AN ITERATIVE SOLUTION PROCEDURE FOR IMPROVING FEASIBILITY IN HIERARCHICAL PRODUCTION PLANNING

A MASTER'S THESIS

in

Industrial Engineering

Middle East Technical University

By

Ferda Can ÇETİNKAYA

May 1988

To my parents

Approval of the Graduate School of Natural and Applied Sciences.

Prof.Dr. Alpay ANKARA

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science in Industrial Engineering.

Assoc.Prof.Dr. Murat KÖKSALAN

Acting Chairman of the Department

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science in Industrial Engineering.

Asst.Prof.Dr. Sinan KAYALIGİL

Supervisor

Examining Committee in Charge:

Assoc.Prof.Dr. Nesim ERKİP (Chairman)

Asst.Prof.Dr. Suna KONDAKÇI

Asst.Prof.Dr. Canan SEPİL

Asst.Prof.Dr. Sinan KAYALIGİL

ABSTRACT

AN ITERATIVE SOLUTION PROCEDURE FOR IMPROVING FEASIBILITY IN HIERARCHICAL PRODUCTION PLANNING

ÇETİNKAYA, Ferda Can

M.S. in Industrial Engineering

Supervisor: Asst.Prof.Dr. Sinan KAYALIGİL

May 1988, 135 pages

To provide effective management support for decisions related to production planning and scheduling problems, it is useful to partition the set of decisions into a hierarchical framework. To take the advantage of the hierarchical structure of production planning decisions, hierarchical formulations and solution procedures have been proposed. However, there are some problems arising in these formulations such as the consistency of decisions at all levels of hierarchy and feasibility of aggregate and detailed plans.

In this study, feasibility problem of detailed schedules in a two-stage flow shop production system is analyzed in a hiearchical framework and an iterative solution procedure for feasibility improvement is proposed.

Key words: Hierarchical Production Planning, Production Scheduling, Feasibility, Flow Shops

ÖZET

AŞAMALI ÜRETİM PLANLAMASINDA FİZİBİLİTEYİ GELİŞTİRMEK İÇİN İTERATİF BİR ÇÖZÜM YÖNTEMİ

ÇETİNKAYA, Ferda Can

Yüksek Lisans Tezi, Endüstri Mühendisliği Bölümü

Tez Yöneticisi : Y.Doç.Dr. Sinan KAYALIGİL

Mayıs 1988, 135 sayfa

Üretim planlama ve çizelgeleme problemleri ile ilgili kararlar için etkin yönetim desteği sağlama konusunda, kararlar kümesini, aşamalı bir çerçevede, daha küçük kümelere bölerek ele almak kullanışlıdır. Üretim planlama kararlarının aşamalı yapısından faydalanarak, aşamalı formülasyonlar ve çözüm yöntemleri önerilmektedir. Fakat bu formülasyonlarda ortaya çıkan, hiyerarşinin tüm aşamalarındaki kararların tutarlılığı, bütünleşik ve detaylı planların fizibilitesi gibi sorunlar vardır.

Bu çalışmada, iki aşamalı akış tipi üretim sistemleri için geliştirilen detaylı çizelgelerin fizibilitesi, aşamalı bir çerçevede analiz edilir ve fizibiliteyi sağlamak için iteratif bir çözüm yöntemi önerilir.

Anahtar Kelimeler : Aşamalı Üretim Planlaması, Üretim Çizelgelemesi, Fizibilite, Akış Tipi Üretim Sistemleri

ACKNOWLEDGEMENT

I would like to express my gratitude to my thesis supervisor, Asst.Prof.Dr. Sinan Kayalıgil, for his valuable and patient supervision and guidance throughout this study.

I am grateful to Asst.Prof.Dr. Suna Kondakçı and Assoc.Prof.Dr. Nesim Erkip for their very helpful and valuable comments.

I would like to thank to my friend Mehmet Kılıç for his kind interest and help.

I wish to express my appreciation to my sister Şeyda Akhan for her excellent work she carried out in drawing the figures and forming this draft.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

CHAPTER 1
INTRODUCTION

Production is one of the basic functions of a business. It is neither a beginning nor an end in itself. The intermediate nature of the production activity results in a set of functional links and interdependencies between production and the other functional areas-such as marketing, personnel, and finance-which ought to be coordinated continuously.

In addition to encountering these functional interfaces, production planning is of a hierarchical nature, since each level of the organization participates in the planning process with different emphasis, scope, and planning horizon.

This hierarchy ranges from strategic planning through tactical planning to operations control. As we go from the top level to the tactical and operational levels, the length of planning horizons decreases and the degree of uncertainty decreases. However, the dependence between the functional activities is typically coordinated more at the tactical level than at the operational level. This also hints at the hierarchical information problems associated with production planning since plans at any given level are based on the information before the fact,and then updated according to the information feedback after the fact. Therefore, in a production environment a hierarchical integration of production activities to insure coordination between various organization levels is responsible for developing and executing plans.

Production planning and scheduling in a production environment require production decisions related with the acquisition utilization, and allocation of production resources to best satisfy consumer needs at the desired times, of the required quality, and at a reasonable cost.

In fact, decisions involve complex choices among a large number of alternatives. These choices have to be made by trading-off conflicting objectives under the presence of financial, technological,and marketing constraints. Such decisions are not trivial and model based systems have proven to be of great assistance in supporting managerial actions in this field.

The problem of production planning and scheduling have been approached from two distinct perspectives-attempts have been made to build an all inclusive direct optimization model, and heuristic algorithms have been proposed to take advantage of the hierarchical structure of production planning decisions. These approches are the monolithic and the hierarchical approaches, respectively.

The basic concept of the hierarchical approach should be familiar to most managers since this procedure is frequently used by organizations on an informal basis. Aggregate planning must be accomplished within the capacity restrictions imposed by long term capacity planning. Detailed scheduling should recognize the manpower and inventory restrictions imposed by aggregate and rough cut planning. Yet, when no formal link exists between the planning levels, conflicts are likely. The worst case occurs when higher

level decisions are not communicated to the lower level planners on a timely basis. Lower level planners may find it necessary to routinely violate aggregate restrictions without any regard to the cost implications beyond the scope of their own span of control.

Formal hierarchical planning simultaneously considers the needs of all planning levels and provides an overall solution that disaggregates well. Some hierarchical formulations explicitly recognize the need for feedback from lower levels to the aggregate level. In this way, problems with disaggregation can be addressed so that an overall near optimal solution is still retained.

However, there are some problems arising in the hierarchical formulations such as the consistency of decisions at all levels of the hierarchy and the feasibililty of both aggregate and detailed plans.

In this study, the feasibility problem of detailed schedules in a two-stage flow shop production system in hierarchical production planning framework is analysed and an iterative solution procedure for feasibility improvement is proposed.

In Chapter 2, an overview of the hierarchical production planning systems is given.

The scheduling function and the two-stage flow shop production system is discussed in Chapter 3.

In Chapter 4, lap phasing (operation overlapping) techniques and its special cases for two-stage flow shop is discussed and

solution procedures for improving schedule feasibility in two-stage system by lap phasing are explained. In addition, in this chapter, family scheduling with family setup times times are discussed and a lower bounding procedure is proposed.

Item re-disaggregation problem and solution procedure is explained in Chapter 5. An iterative solution procedure for schedule feasibility improvement in a two-stage production system is also proposed in a hierarchical production planning framework.

In Chapter 6, the solution procedures explained in previous chapters are illustrated by a sample problem and the results are provided.

In Chapter 7, the concluding remarks are presented together with the comments on possible future research.

CHAPTER 2

AN OVERVIEW OF HIERARCHICAL PRODUCTION PLANNING

Two distinct approaches for production planning have appeared in the literature:

The first approach, called the **monolithic approach**, formulates the production planning and scheduling problems as a large mixed integer linear programming that captures the relevant cost and restrictions of a problem. Planning is at a level of detail where set up costs are incurred. Unfortunately, the resulting MILP formulation is too large and computationally prohibitive. In order to derive an approximate solution, considerable amount of work must be performed to transform the problem into an equivalent linear programming model and then solve it by using large scale programming methods which, in many cases, would only give near optimal solutions.

The original work in this area was done by Manne (1958). Many contributions were made by Dzielinski, Baker, and Manne (1963), Dzielinski and Gomory (1965), Goodman (1978), and Lasdon and Terjung (1971).

The obvious alternative to a detailed monolithic approach to production planning is a **hierarchical approach.** The basic design questions of a hierarchical planning system are the partitioning of the overall planning problem and the linkage of the resulting subproblems. An important input to resolve these questions is the number of levels recognized in the product structure. In any

planning period, the subproblems are solved sequentially, with the solutions of subproblems from the upper hierarchy imposing constraints on the lower hierarchy subproblems.

Hierarchical planning and analysis have been the subject of much discussion and conjecture in recent years. It has been asserted by Mesarovic, Macko and Takahora (1970) that hierarchical formulation of complex large-scale system problems is the most desirable for decision making. Such formulations are adaptive so that response to the various facets of complexity such as multi-dimensionality, interactiveness and uncertainty are possible. They are also versatile in bringing out the decomposition of the total system into several functional subsystems.

The basic philosophy of a hierarchical planning approach may be well understood by its essential characteristics. First, the processing of information through the hierarchical disposition of the subsystems is possible. Second, the nature of interaction and interdependence between various subsystems become more transparent. Third, the performance assessment of the whole system through the means-ends or actions-reactions consequences becomes possible through linkage and feedback mechanism.

The specific advantages of using a hierarchical planning method lie in the explicit identification of the various interacting subsystems and emphasis on the study of the functional and behavioural aspects of each subsystem. In summary, it can be said that hierarchical planning approaches can construct an internal, functional image of the various interlinking subsystems that can

produce the same internal effects on the total system but the images and the effects become more transparent to the decision makers.

The advantages of the hierarchical planning as compared to a monolithic one may now be clearer. The primary advantage of the monolithic approach is that it focuses on a well defined model formulation for which optimization is meaningful.On the contrary,the hierarchical approach breaks the production planning problem into subproblems, at best, can only optimize each of these subproblems which results in a total system suboptimization.

The advantages of the hierachical approach can be divided into four distinct categories.

1. The first category considers the cost of data collection to support the model as well as the computational cost of running the model. A major information system may be required to collect demand, productivity and cost data as well as prepare forecasts for thousands of individual items;this is a more costly project than building the production planning system itself. This data must then be reviewed by management. As the number of items increases, this effort can become unwieldy, leading to deterioration of the data used in the planning process and therefore the output. In most cases, this cost of data collection and preparation will far outweigh the cost of computation. This is important to note as the cost of computation continues to decrease and it becomes feasible to solve enormous linear or non-linear programming problems. Aggregation of items can significantly reduce the cost and effort in demand forecasting and data preparation in addition to decreasing

the computational costs.

2. The second category considers the accuracy of the demand data. Unless all items are perfectly correlated, an aggregate forecast of demand will have a reduced variance. In general, one is able to employ more sophisticated techniques and spend more time in obtaining managerial judgement, given the smaller number of forecasts required in hierarchical planning. Since decisions on regular time, overtime, hiring and firing, and other production rate changes are based on the total production quantity demanded, increased forecast accuracy on total demand should improve the decison making process.

3. Another significant feature of hiearchical approach relates to the cognizance of uncertainties. For instance, in any complex manufacturing system, two conflicting pressures are imposed upon managers. First, it may be necessary to act without delay to avoid decision by default. Second, time is needed to better understand the situation. In monolithic planning aproaches, these two situations can not be modelled. Hierarchical planning methods provide intervention and coordination mechanisms which assist in understanding the situation at each level. Intervention may be affected by the resetting of objectives; revision of constraints, parameter values, performance results and decision rules etc. through timely information.

4. Finally, and perhaps from an implementation standpoint, most importantly, aggregation leads to more effective managerial

understanding of the models results. When ten thousands of items are being planned simultaneously, the sensitivity of the results to changes in individual item demands may be complex. There are too many combinations of changes to consider.

2.1. Hierarchical Production Planning Systems :

An important input to a hierarchical production planning system is the number of levels recognised in the product structure. The hierarchical levels represented in the system correspond to the organisational echelons of the firm, which facilitate the interaction between the system outputs and the responsible manager at each level, allowing for appropriate managerial inputs to be made. Hax and Meal (1975) have identifed three levels such as Items, Families and Types. Starting from the lowest level, these are :

1) Items (also called stock-keeping units, SKU) refer to the highest degree of specificity regarding the manufactured or purchased products. Items differ in terms of specification of size, packaging characteristics, color, dimensions, etc.

2) Families (of items) are formed by items pertaining to a same product type and sharing a common manufacturing tooling and set-up cost or a common purchase ordering cost.

3) Product Types are aggregations of families and/or non-family items. Their production quantities are to be determined by an aggregate production plan. Families and non-family items belonging to a type normally have similar costs per unit of

production time, inventory holding costs, labor costs, productivity rates, backorder costs, and seasonalities.

It has been found that these three levels are necessary to characterize the product structure in many batch processing manufacturing environments. However, there may be some practical applications where additional or fewer levels might be required for incorporation into the models. The hierarchical framework considered in this study is based on these three levels of item aggregation. The same principles can be extented to any number of levels by proper definition of subproblems to link the various levels.

2.1.1. A Review of Hierarchical Planning Models:

Many types of hierarchical production planning models have been developed. Some have been implemented and validated in practice. This section first considers theoretical models, then applications.

Hax and Meal, and Bitran, Haas, and Hax Models:

Bitran, Haas, Hax, and Meal have been pioneers in developing the hierarchical approach (Hax and Meal(1975), Bitran and Hax(1977, 1981), Bitran, Haas and Hax(1981,1982), Hax and Candea(1984)). Their models use three levels of aggregation: type,family, and item. The cost structure and the planning horizons of each level can be set independently of the other two levels. But the production decisions at the item level productions are similarly restricted by the family level. Feedback from lower levels is used informally. The quality of the aggregate solution should be judged by the ability to schedule at the item level. If necessary, the aggregate plan can be modified

with a determinable increase in cost.

There are three steps involved in these methods:

(1) The first step is to allocate the total production capacity among product types by means of an aggregate planning model. Their formulation includes the cost of inventory, regular and overtime labor, and other production costs. However, setup costs are not considered at the aggregate level. Their inclusion complicate the formulation and require more detailed demand predictions for the full aggregate horizon; thus losing the principal advantage that the hierarchical approach has over the monolithic approach.

(2) The second step in the planning process is to allocate the production quantities for each product type among families belonging to that type.This is done by family disaggregation model.

(3) Finally, the family production allocation is divided among the items belonging to each family by the item disaggregation model.

These models are analysed in more detail in the following sections.

Bitran, Haas, and Hax (1981,1982) have recently proposed a modification to the Hax-Meal framework for settings where the setup costs are significant. This modification entails a "look-ahead" procedure for solving the scheduling subproblem; however, the

planning subproblem remains unchanged in that it still does not consider its cost impact on the scheduling subproblem.

Jaikumar's Model:

Jaikumar (1974) presents an alternative hierarchical system in which the problem is again decomposed into a planning component and scheduling component. Jaikumar's approach differs from that of Hax and Meal primarily with respect to how the subproblems are linked. Jaikumar uses the solution from the planning subproblem not only to constrain the scheduling problem, but also to define certain scheduling costs based on the shadow prices from the planning subproblem.

These hierarchical approaches seem to work well provided that the primary costs are those costs associated with the aggregate subproblem, that is, the overtime costs and inventory holding costs associated with the product types are dominant, while the family setup costs are secondary in importance and magnitude. However, when the family setup costs are significant, there is a conceptual gap in the original hierarchical framework in that the aggregate planning subproblem does not consider its impact on total setup costs as determined by the family disaggregation subproblem.

Graves' Model:

If more than informal feedback from the lower level models is required, then a Lagrangian approach may be necessary. Graves (1982) has developed a framework for solving a hierarchical production planning problem which formally includes feedback from

the lower levels of aggregation. The method retains more of the information which the monolithic approach contains, yet the mathematical technique reduces the problems of finding a solution. Graves himself has developed only an approximate algorithm for its solution.

Newson's Model:

Newson (1975) proposes and tests an iterative heuristic in which the problem is decomposed into a planning subproblem and a scheduling subproblem. The approach is similar in spirit to that of Graves. The difference between these approaches is in the definition of each subproblem and in the updating procedure of each subproblem at each iteration.

The linking mechanism for the two subproblems mentioned before is an inventory consistency relationship which is priced out by a set of Lagrange multipliers. The best values for the multipliers are found by an iterative procedure which may be Hax-Meal framework. At each iteration, the procedure finds both a lower bound on the optimal value of the production planning problem and a feasible solution from which an upper bound is obtained.

2.2. Models at Different Hierarchical Levels :

2.2.1. Aggregate Production Planning Models (APP):

APP is considered to be the highest level planning decision and used to allocate the total production capacity among product types. The planning horizon for such models cover at least a full

year in order to properly reflect the fluctuating demand requirements for the products. Aggregate planning has been the subjects of discussion by many authors. Linear Programming , Linear Decision Rules, and Management Coefficient Models etc. are some methods encountered in the literature.

2.2.2. Family Disaggregation Models :

At this stage of hierarchical planning system, family dissagregation models allocate the production quantities of types dictated by the aggregate model among the families of the type. Only the results of the first period of planning horizon are disaggregated, thus substantially reducing the required amount of data collection and processing. The main condition to be satisfied at this planning level for a coherent disaggregation is that the sum of the productions of the families in a product type equals the amount dictated by the higher level model for this type. This assures consistency and feasibility among the type and family production decisions. It is at this stage where setup costs are explicitly considered and minimized.

There are four major disaggregation methods which have been proposed in the literature:

(1) Hax and Meal method

(2) Knapsack method

(3) Winters method

(4) Equalization of Run-Out Times.

(1) Hax and Meal Disaggregation Method:

Hax and Meal (1975) have suggested a heuristic approach that:

(i) schedules those families in each product type that must be run in the current period in order to meet the item's service requirements;

(ii) sets initial family run quantities so as to minimize cycle inventory and setup costs;

(iii) adjusts the family run quantities so as to use all the production time allocated to each product type by the aggregate planning model, while observing the item's overstock limits.

(2) Knapsack Method:

Bitran and Hax (1981) have formalized the Hax and Meal (1975) heuristic into a convex knapsack problem that minimises the total setup costs of families. As it is mentioned previously, Bitran *et al.* (1981,1982) have suggested modifications to the knapsack method by means of look forward feasibility rule to reduce the myopic nature of the disaggregation rules and by a routine to improve the performance under high setup costs.

(3) Winters Method:

Winters (1962) examines various alternatives for disaggregating the aggregate production quantities. He recommends a disaggregation procedure in which families are produced in economic order quantities, in the order of their increasing run-out times, until the aggregate total is reached. Unlike the Hax and Meal method, the initial run quantities are not modified, but are treated

as indivisible discrete units and their release point are varied.

(4) Equalization of Run-Out Times:

An obvious alternative disaggregation method is to allocate the production amount determined at the aggregate planning level for a given type in such a way as to equalize the run-out times of all the items belonging to that type. This implies skipping the family level during disaggregation level. Run-out time equalization is a natural disaggregation methodology to be applied at the item level and,therefore , the corresponding literature is presented in the section of item disaggregation.

It is important to mention at this point that when run-out equalization is directly applied at the item level, no consideration is given to the resulting setup costs associated with the family runs. Thus, it might be expected that this disaggregation procedure will generate fairly high setup costs, relative to the other family disaggregation methods which take explicit account of setup costs. The possible advantages of a direct item run-out time equalization are the realization of a high degree of synchronization of the production planning system, and the added simplicity in implementing the hierarchical system.

2.2.3. Item Disaggregation Models:

Hax and Meal (1975) have proposed a heuristic algorithm to equalize the run-out times of the items belonging to each family. The essence of this approach is to allocate the family run quantity so as to maximise the expected time until an item in that family

runs out. Bitran and Hax (1981) have formalized the above heuristic approach by treating the run out time equalization problem as a strictly convex knapsack problem. Another Equalization of Run-out Times (ERT) rule has been suggested as a heuristic for allocating production capacity across a group of items with a common setup cost by Karmarkar (1981).

CHAPTER 3

SCHEDULING FUNCTION AND ANALYSIS OF A TWO-STAGE FLOW SHOP PRODUCTION SYSTEM

As it is mentioned in the previous chapters, the hierarchical nature of the managerial process warrants the use of hierarchical production planning and inventory control systems. In these systems decisions are made in sequence, with each set of decisions at an aggregate level providing constraints within which more detailed lower level decisions are made.

If a manufacturing environment is considered, a typical set of decision levels consists of facilities design, assignment of products to manufacturing plants and distributing warehouses, aggregate planning, inventory control and operations scheduling. The lower the level in the hierarchy, the narrower the scope of the the plan is, the lower the management level involved is, the more detailed the information needed is, and the shorter the planning time horizon is. Thus, operations scheduling is one of the decisions at the bottom of the hierarchical planning system. Its purpose is to make the most detailed scheduling decisions involving the assignment of the operations to specific machines and operators during a given time interval.

Since operations scheduling is the lowest level in a hierarchical structure, it uses decisions made at higher planning levels with respect to two important issues :

1. The productive resources (facilities with machines and workers) are fully specified. It is not within the scope of scheduling to make changes in the production capacity.

2. The jobs to be performed are given to the operations scheduling process and they all must be completed.

Therefore, the function of scheduling becomes relevant in a situation where the nature of the tasks to be scheduled has been described and the configuration of the resources available has been determined.

In practice, of course, the scheduling and planning functions may not be completely independent. In the context of the hierarchical production planning system, the tasks to be carried out are identified and the limits on the amount of resources available are set at the higher planning levels. Then comes the allocation of the available resources to perform the specified tasks according to the information given from the higher levels. The interaction between these two levels might be repeated over several information exchanges before a final planning decision is reached.

Therefore, in a manufacturing environment, the feasibility is an essential problem of the scheduling function since there are limits on the capacity of available resources and the technological restrictions for the order in which tasks can be performed.

To represent the feasibility problem in operations scheduling, a two stage flow shop production system may be a basis

since an optimal solution exists only for F_{max} criterion in the scheduling problem (which is discussed later on) and the complex behaviour of the flow shop may effect the efficient utilization of the existing facilities.

3. 1. A Two-Stage Flow Production System:

A two-stage Flow Shop production system can be thought as a production process in which a number of products has to be processed sequentially through two stages in series.

In such a production environment, there are several tasks to be done before finding out a solution to the sequencing and scheduling problem. These are:

1) establishment of the technological ordering in which a job (a unit of product or a batch of identical units) is to be processed through the stages.

2) specification of the materials, fixtures and tools which are required to process the jobs.

3) determining the economical batch (lot) size, if any, in which a job is to be processed.

4) estimation of the processing time, job transportation, and setup times, if any, for each of the operations comprising the job.

5) setting the time limit by which the job should be completed in response to a delivery date.

The majority of research on two-stage flow shop scheduling has centered on the sequencing problem. This problem, of course, is only a part of the overall production control problem in such a system. However, not much attention has been paid to conform the regulations and adjustment of production activities with the plans.

System Description :

Figure 3.1 represents the model for the two stage flow shop production system.

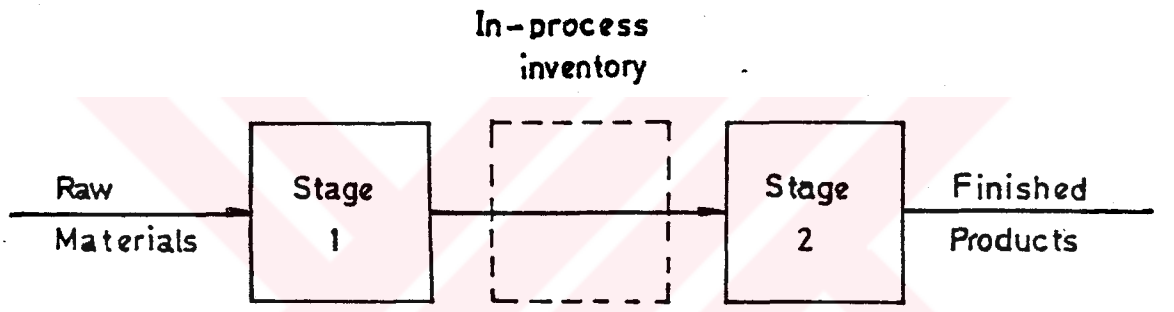


Figure 3.1. Two stage flow shop production system

Two-stage flow shop models that have been developed are based on a set of assumptions, the purpose of most of which is to simplify the analysis of the system. The assumptions include those placed on the characteristics of the jobs, stages, and processing, setup and transportation times. These assumptions are :

1. *Assumptions regarding jobs :*

1.1. Each job is processed first at stage 1 and then at stage 2.

1.2. Each job, once started, must be performed to

completion, that is, no job cancellation occurs.

1.3. Each job, once started processing at a stage, must be performed to completion before another job can start processing at that stage, that is no preemptive priorities.

1.4. Each job may not be processed by more than one stage at a time. This eliminates lap-phasing in which the same job is started on the succeeding stage as soon as some units are available from the preceding stage. This assumption is relaxed later on.

1.5. Each job may have to wait between stages and hence, in-process inventory is permitted.

2. *Assumptions regarding stages :*

2.1. Each stage consists of only one manufacturing facility.

2.2. Each stage in the shop operates independently.

2.3. Each stage is continuously available for assignment, during the scheduling period under consideration, without any interruption such as machine breakdowns or maintenance.

3. *Assumptions regarding processing, setup and transportation times :*

3.1. Processing times for jobs are known and finite at

each stage.

3.2. For each job, a setup time is incurred in setting up a stage. This setup time is known and finite at each stage.

3.3. Setup time is independent of the production sequence.

3.4. Transportation times may be considered negligible. Therefore, if the processing of a job is complete at stage 1, its second operation can start immediately, as soon as stage 2 is available.

3.5. Processing times may implicitly include setup times, if any.

The above assumptions indicate how explicitly the conditions of the scheduling problem should be defined before a conceptual model can be used. As a consequence of relaxing one or more of these assumptions, different versions of the basic model can be formulated.

3. 2. Measure of Performance:

The solution of the two-stage flow shop scheduling problem demands an explicit statement of a criterion or set of criteria. A study of the criteria proposed in the literature indicates that a wide variety of performance measures are employed. The variety of different scheduling situations and the prospects of obtaining solutions have usually influenced the choice of criteria.

A performance measure which is employed for a two-stage flow shop in this study is the makespan, or the total amount of time required to completely process all the jobs. For such a system, the makespan is the maximum flow time :

$$F_{max} = \max_{1 \le i \le n} \{ F_i \}$$

where, F_i is the flow time of job i

For a flow shop system, total delay is made up of run-in delay, between-jobs delays and run-out delay. It may be stated without proof that if the sum of run-in delay and between jobs delays at the second stage in a two-stage flow shop or the run-out delay at stage 1 is minimum, the corresponding sequence is optimal with respect to makespan (Figure 3.2).

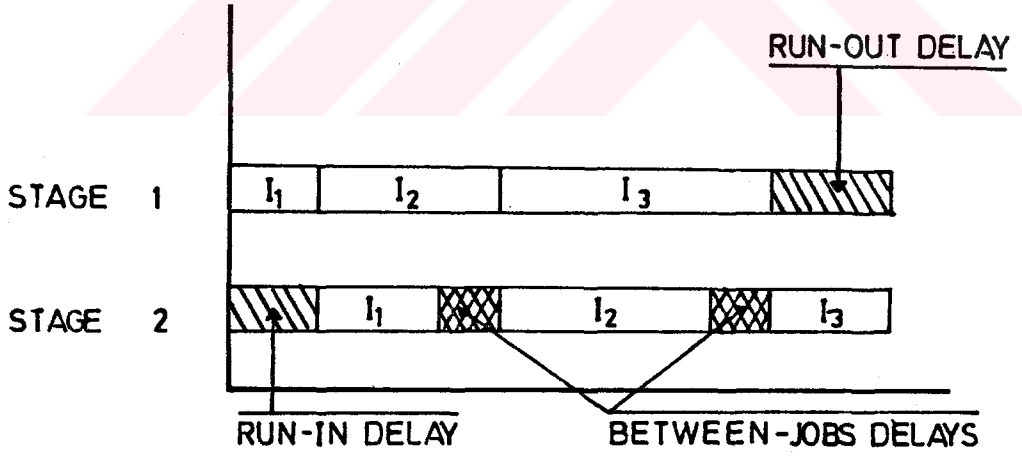


Figure 3.2. Delays in flow shops

It is obvious that efficient scheduling will allow a given work load to be performed in a short time, and thus maximize the utilization of the equipment. On the other hand, poor scheduling is costly, since it will always lead to the occurance of delay.

Therefore, the makespan is an important measure to evaluate the schedule which should allow the given work load to be performed in an allocated period of time.

3. 3. Permutation Schedules:

In a flow shop, if the arbitrary sequence of jobs at each stage is considered, then with m stages and n jobs a total of $(n!)^m$ schedules may be constructed. There are, however, two important theorems that serve to reduce the number of schedules which have to be considered (proofs may be found in Baker(1974) and French(1982)).

Theorem 1:

When scheduling to minimize any regular measure of performance in a static deterministic flow shop, it is sufficient to consider only schedules in which the same job ordering is imposed at the first two stages.

Theorem 2:

When scheduling to minimize makespan in the static deterministic flow shop, it is sufficient to consider only schedules in which the same job ordering is imposed at stages 1 and 2, and the same job ordering at stages m-1 and m.

A permutation schedule for a flow shop is a schedule in which the order of jobs is identical at all stages. By the above theorems, the followings can be stated as corollaries:

Corollary 1: In a two-stage flow shop, the optimal schedule with respect to any regular measure of performance is a permutation schedule.

Corollary 2: In a three-stage flow shop, the makespan is minimized by a permutation schedule.

In the following section, a mathematical programming model of the two-stage permutation flow shop problem is given.

3. 4. Mathematical Programming Model ($N/2/P/F_{max}$):

The mathematical programming model of the permutation flow shop problem for a two stage production system can be formulated as follows :

i) *Decision Variables :*

$$X_{ik} = \begin{cases} 1, \text{ if job } i \text{ is scheduled in the } k\text{th position of the processing sequence} \\ 0, \text{ otherwise} \end{cases}$$

for $i = 1,2,\ldots,N$ and $k = 1,2,\ldots,N$.

I_k = idle time (between-jobs delays) of stage 2 between the completion of job in the k th position of the processing sequence and the start of job in the (k+1) th position.

for $k = 1,2,\ldots,N-1$ since there is no idle time in between jobs at stage 1.

W_k = the time that the job in the k th position of the processing sequence must spend between completion at stage 1 and starting to be processed at stage 2.

for k = 1,2,...,N.

ii) *Constraints :*

a) Exactly one job is scheduled in the k th position :

$$\sum_{i=1}^{N} X_{ik} = 1 \qquad \text{for } k = 1,2,...,N. \qquad (1)$$

b) Each job is scheduled in exactly one position :

$$\sum_{k=1}^{N} X_{ik} = 1 \qquad \text{for } i = 1,2,...,N. \qquad (2)$$

c) The times τ_k between completion of the job in the k th position in the sequence at stage 1 and the start of job in the (k+1)th position in the sequence at stage 2 must be well defined. In Figure 3.3 the relationships between the variables in the model are indicated.

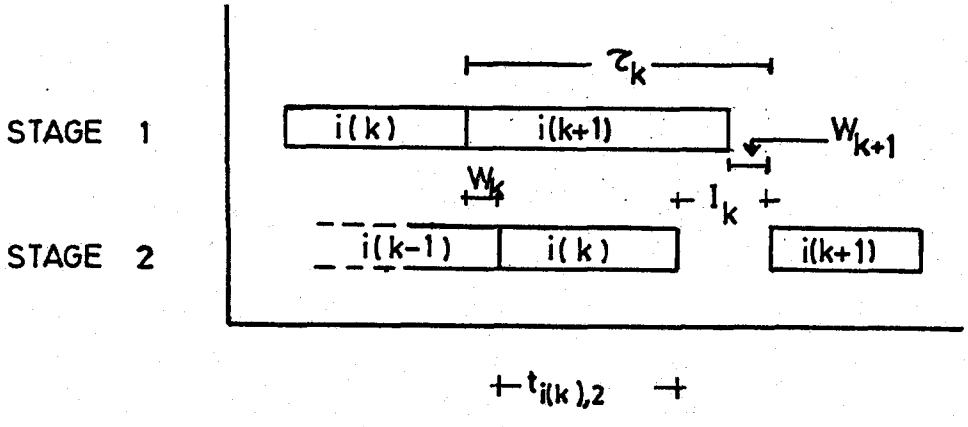


Figure 3.3. The relationships between the variables in the integer programme.

From the figure it is clear that

$$\tau_k = W_k + t_{i(k),2} + I_k = t_{i(k+1),1} + W_{k+1} \qquad (3)$$

where,

$t_{i(k),2}$ is the total processing time of job i in the k th position in the processing sequence at stage 2.

Since $X_{i,k}$ is one for i=i(k) and zero for other i's, hence $t_{i(k),2}$ is expressed in terms of the $X_{i(k)}$ as follows :

$$t_{i(k),2} = \sum_{i=1}^{N} X_{i,k} \cdot t_{i,2} \qquad (4)$$

Similarly,

$$t_{i(k+1),1} = \sum_{i=1}^{N} X_{i,k+1} \cdot t_{i,1} \qquad (5)$$

Thus the equation (3) may be rewritten as :

$$W_k - W_{k+1} + \sum_{i=1}^{N} X_{i,k}\, t_{i,2} - \sum_{i=1}^{N} X_{i,k+1}\, t_{i,1} + I_k = 0 \qquad (6)$$

The above constraints hold for k=1,2,...,(N-1).

Therefore, the constraints (1), (2), and (6) together with integrality of X_{ik} and nonnegativity of W_k and I_k form the entire constraint set.

iii) *Objective function :*

Objective function of this model is to minimize the total idle time at stage 2, equivalently to minimize the makespan. This idle time is given by the sum of the between-jobs delays plus the run-in delay at stage 2. Thus, the model seeks to minimize :

$$Z(i(1),\ldots, i(N)) = \sum_{k=1}^{N-1} I_k + t_{i(1),1}$$

However, run-in delay at stage 2 can be expressed in terms of binary variables X_{ik} as follows :

$$t_{i(1),1} = \sum_{i=1}^{N} X_{i1} \cdot t_{i1}$$

Hence,

$$Z(i(1),\ldots,i(N)) = \sum_{i=1}^{N-1} I_k + \sum_{i=1}^{N} X_{i1} \cdot t_{i1}$$

Therefore the MIP model becomes

minimize $Z(i(1),\ldots,i(N))$

subject to constraints (1), (2), (6) and nonnegativity and integrality constraints.

Since the speed with which integer programs can be solved depends upon the number of variables and constraints in the problem, it is informative to count these in the above formulation. Counting the variables first, there are

N^2	integer variables X_{ik}
N-1	real variables I_k
N-1	real variables W_k
$N^2 + 2(N - 1)$	variables in total

Next counting the constraints, there are

N	constraints of type (1)
N	constraints of type (2)
N-1	constraints of type (3)
3N-1	constraints in total

When there are M stages in a flow shop, the total number of variables and constraints become $N^2 + 2(M-1)(N-1)$ and $MN + N - M + 1$, respectively. When practical applications are considered, there is no advantage in translating scheduling problem into integer programming problem.

3. 5. Johnson's Algorithm for $N/2/P/F_{max}$:

Johnson (1954) constructed an algorithm which gives a solution procedure for sequencing n jobs, all simultaneously available, in a two stage flow shop so as to minimize the maximum flow time.

Theorem 3: (Johnson's Rule)

In an optimum processing sequence, job i precedes job j if

$$\min \{ t_{i1}, t_{j2} \} \leq \min \{ t_{i2}, t_{j1} \}$$

(See Baker (1974) for the proof and the algorithm).

Using the algorithm optimal schedule is found as the permutation of jobs such that the earlier jobs in the processing sequence have relatively shorter processing times at stage 1, whereas the later jobs have relatively shorter processing times at stage 2.

Considering the scheduling property of Johnson's algorithm, it can be observed that the optimal schedule consists of respective optimal schedules of two disjoint set of jobs. The first optimal schedule S_1 is a permutation such that the jobs having relatively shorter processing times at stage 1 is ordered in nondecreasing values of their processing times at stage 1. The second optimal schedule, S_2 is a permutation such that the jobs having relatively shorter processing times at stage 2 are ordered in nonincreasing values of their processing times at stage 2.

3. 5. 1. Lower Bounds on the Maximum Flow Time (F_{max}):

A permutation schedule for a two-stage flow shop problem discussed in the previous sections may be obtained by the application of Johnson's algorithm. In fact, before applying the algorithm one can find out whether there is a feasible schedule (a schedule is feasible if a set of jobs can be processed in an allocated production time) or not. To do this lower bounds may be useful tools since they give a brief idea about the value on the maximum flow time. If the lower bound is greater than the value of period length under consideration (time allocated to process all the jobs), it can be stated that the production of jobs with the current lot quantities is not possible.

By examining the principle ideas of the Johnson's Algorithm, lower bounds on the makespan can be derived.

Let t_{i1} be the processing time at stage 1 of the i th job in the processing sequence. A similar processing time t_{i2} for stage 2 may be defined.

It is clear that the last job can not be completed earlier than the time required to process all N jobs at stage 1 plus the time needed to perform the second operation of the last job.

$$F_{max} \geq \sum_{i=1}^{N} t_{i1} + t_{N2}$$

Similarly, the last job can not be completed in less time than it takes to process all N jobs at stage 2 plus the time caused by the delay before stage 2 can begin. Thus,

$$F_{max} \geq \sum_{i=1}^{N} t_{i2} + t_{11}$$

The sums of the processing times in the above equations are entirely unaffected by the ordering of the jobs. The bounds are only affected by the choice of t_{N2} and t_{11}.

The nature of the Johnson's algorithm suggests that t_{11} and t_{N2} are the smallest of the values in the set of jobs for schedules S_1 and S_2, respectively.

Therefore, the lower bound on the maximum flow time is

$$LB_{F_{max}} = \max \{ \sum_{i=1}^{N} t_{i1} + t_{N2} , \sum_{i=1}^{N} t_{i2} + t_{11} \}$$

CHAPTER 4

LAP-PHASING AND FAMILY SCHEDULING

A production schedule for a two-stage lot production system discussed in the previous chapter may be obtained by the application of Johnson's algorithm. In fact, before applying the algorithm, one can find out whether or not there is a feasible schedule. To do this, the lower bounds on the maximum flow time may be introduced. If the lower bound is greater than the production period length under consideration, it can easily be stated that the production of all items with the current production run quantities is not possible. In such a case, the run quantities must be re-arranged. That is, there is a need for re-disaggregation for all items.

Although the lower bound on the makespan is less than the value of the period length in which all the jobs should be processed, after applying the Johnson's algorithm an infeasible solution may be found. In such a case, to improve the feasibility:

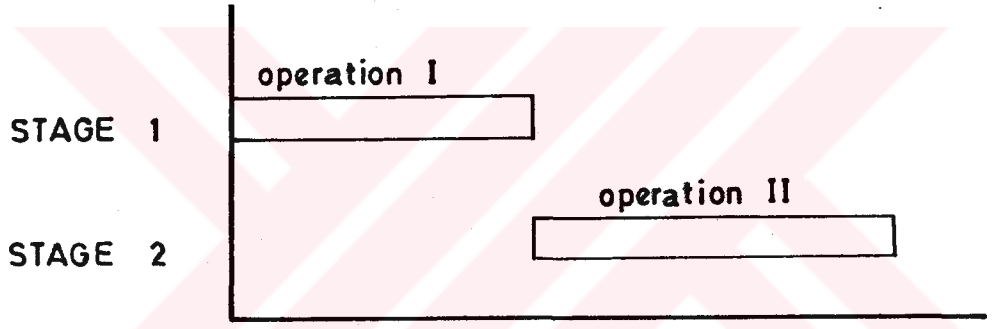
1. large lots may be split into sublots, and

2. the assumption 1.4 which is stated in the Chapter 3 may be relaxed. i.e. lap-phasing is allowed.

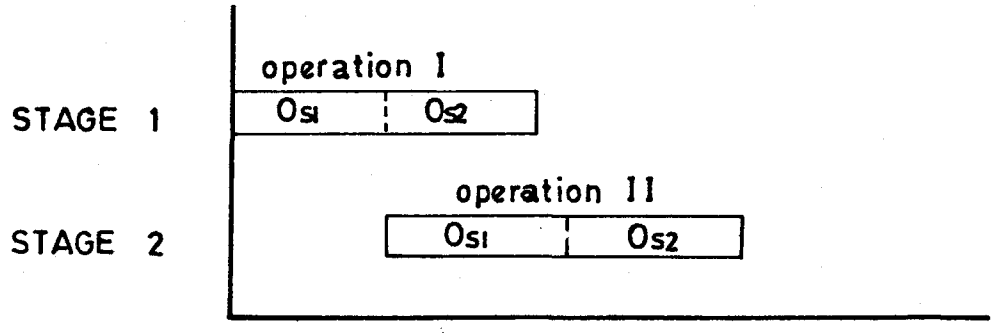
As a consequence, completed sublots may proceed to the subsequent operation without waiting for the full lot to be processed. In such an overlapping structure, the assumption 1.3 (no preemptive schedule) is still valid.

4. 1. Scheduling by Lap-Phasing (Operation Overlapping):

In many practical situations where jobs consist of large lots, it is possible to start processing of an item of a lot at the succeeding stage before the completion of the processing of the full lot at the preceding stage. A small sublot may be processed without waiting for the whole lot to be processed at stage 1. This structure allows to reduce the waiting times at second stage, consequently the maximum flow time (Figure 4.1).



4.1 (a) Operations nonoverlapping



4.1 (b) Operations overlapping

Figure 4.1. Operations nonoverlapping and overlapping

Because the production system under study consists of many jobs having large lots of items, operation overlapping can reduce the maximum flow time (makespan) significantly.

In practice, batch transfering costs between stages may take an important role. To avoid increase in transferring costs, it is possible to restrict the size of sublots transferred to the second stage. Therefore, minimum quantity on the number of identical units in a sublot may be introduced into the production system.

For a given sequence of jobs in a two-stage flow shop system, processing of a job at stage 2 can start immediately as long as stage 2 is available. This means that two cases can be identified:

Case 1 :

Processing at stage 2 can start when the second stage is available. i.e.,

$$C_{i1} > C_{i-1,2}, \quad W_i = 0 \text{ and } I_{i-1,i} = C_{i1} - C_{i-1,2}$$

where C_{ij} is the completion time of i th job at stage j,

W_i is the waiting time of i th job at stage 2,

$I_{i-1,i}$ is the idle time at stage 2 between the completion of (i-1)th job and the start of i th job in the sequence.

Lap-phasing can be applied to this case since sublots may proceed to the subsequent operation without waiting for the full lot to be processed. Since at stage 1 a job can start processing immediately, a sublot having

$$Q_s = [(C_{i-1,2} - C_{i-1,1}) - S_{i1}] / p_{i1}$$

units of item may proceed to the second operation without waiting where,

S_{i1} is the setup time incurred at stage 1 before processing item i (Note that the setup time is only incurred prior to the first sublot),

p_{i1} is the unit processing time of item i at stage 1.

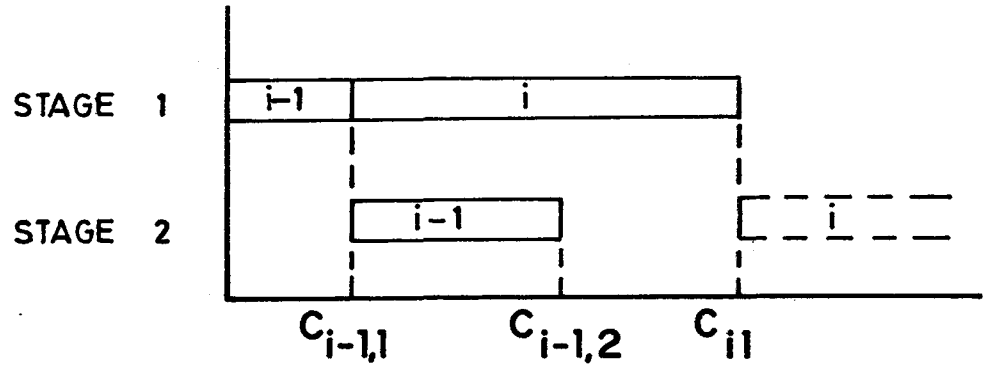
Case 2 :

Processing at stage 2 can not start because of unavailability of the second stage. i.e.

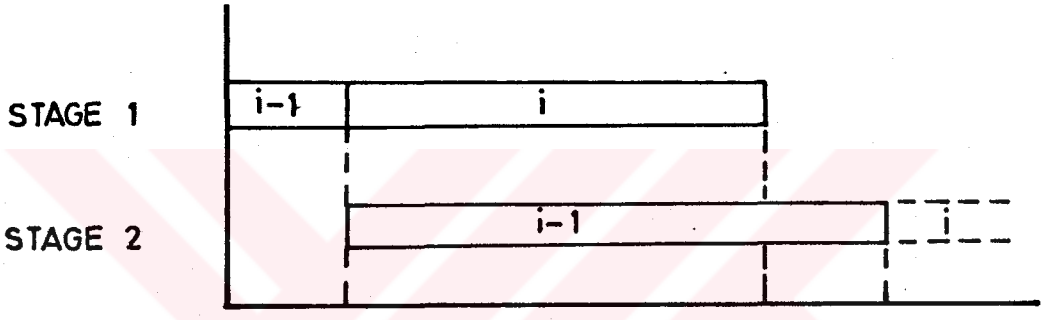
$$C_{i1} < C_{i-1,2}, \qquad W_i = C_{i-1,2} - C_{i1}$$

In such a case there is no need to apply the lap-phasing since it is not possible to start processing of an item of a lot at stage 2 before the completion of the processing of the full lot at stage 1 given the assumptions.

The above cases may be illustrated graphically in Figure 4.2.a and 4.2.b, respectively.



4.2 (a) Second stage is available.



4.2 (b) Second stage is not available.

Figure 4.2. Operation at stage two

4. 2. Decomposition of Jobs into Two Sets:

Two sets of jobs are identified previously as set S_1 that contains the jobs having shorter processing times at stage 1 and set S_2 that contains the jobs having shorter processing times at stage 2. The optimal schedule is a permutation of items such that the earlier jobs in the processing sequence are the elements of set S_1 whereas the later jobs are the elements of set S_2.

Applying the lap-phasing to the respective schedules of two sets of jobs, the following propositions can be stated :

Proposition 1:

For each job in set S_1, the sublot sizes will be nondecreasing in size except the last sublot size.

Proof:

Since the processing times at stage 2 is greater than the processing times at stage 1, the difference between the completion times at two stages increases after the initial sublot is processed in the system. Then, the second sublot size will be greater than the first one. Since each time the difference between the completion times at two stages increases, the sublots will have more units of item compared to the previous sublots. Although the difference between the completion times of two stages before the processing of the last sublot may allow to have a large last sublot size, a small sublot size may be choosen to prevent the excessive production of the item considered.

Proposition 2:

For each job in set S_2, the sublot sizes will be nonincreasing in size.

Proof :

Proof is similar to the previous one. The difference between the completion times at two stages decreases as the number of sublots increases.

Proposition 3:

To avoid the delays at stage 2 , given the sequence of jobs, there exists a lower limit on the number of sublots for each item to

be processed.

Proof :

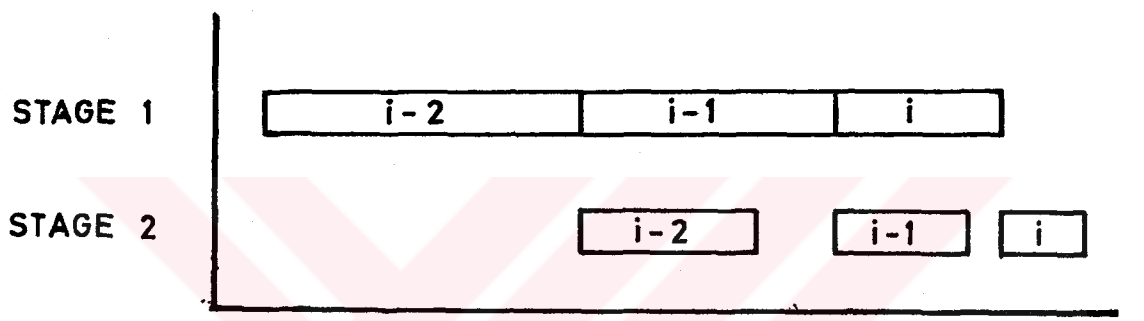
As it is mentioned in the previous section, the size of each sublot that is sent ahead to the second stage is determined by the completion time difference between stages after processing the previous item or sublot. To avoid the delay at the second stage, the sublot sizes must be chosen such that the processing of the sublots are completed before the completion time of the previous sublot (or the previous item if the sublot is the initial sublot) at stage 2. If the number of items processed in the initial sublot increases, the time difference between two stages increases. As a result, the number of items in the second sublot becomes more than the previous one. In this manner, the number of sublots decreases since each time the sublot sizes are chosen as big as possible without causing delay at stage 2. The number of sublots determined this way is a lower limit. One can increase the number of sublots without causing delays at stage 2. However, it is impossible to decrease the number of sublots without increasing the idle time since any increase in the sublot sizes will cause delays at stage 2.

4. 3. Reverse Scheduling :

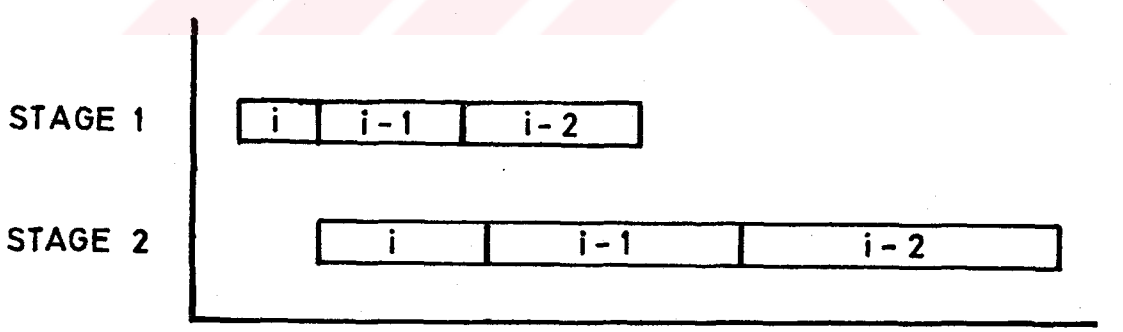
It was observed before that the application of lap-phasing to the two sets of jobs decomposed forced independent analysis of two sets (i.e. S_1 and S_2). The symmetrical argument (reversing the time scale) and the interchange in the processing times at stages might simplify the solution obtained by lap-phasing.

A **reverse schedule** may be defined as the permutation schedule that treats the jobs by interchanging the processing time at both stages and reversing the time scale. i.e. the time runs backwards from completion towards job initiation.

Let us consider the schedule in Figure 4.3.a. The reverse schedule for the jobs in this figure may be obtained as shown in Figure 4.3.b :



4.3 (a) Original schedule



4.3 (b) Reverse Schedule

Figure 4.3. Original and reverse schedules (Delay at stage two).

If the schedule in Figure 4.3.b is reversed again, the schedule in Figure 4.4 is obtained. When the two schedules in figure 4.3.a and Figure 4.4 , original and schedule obtained by reversing, are compared it can be noticed that the maximum flow time does not change. The only difference is that the jobs coming before the last job are shifted to the right in the reversed schedule in Figure 4.4. If the job i-2 is considered as the first in the processing sequence, the run-in delay is increased, however the total delay remains the same.

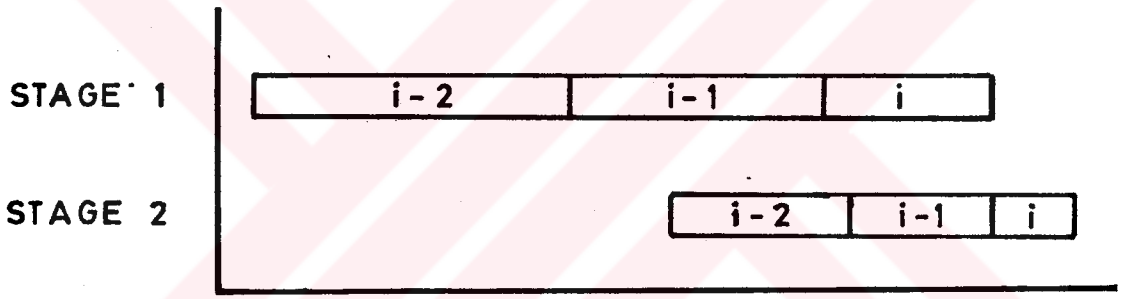
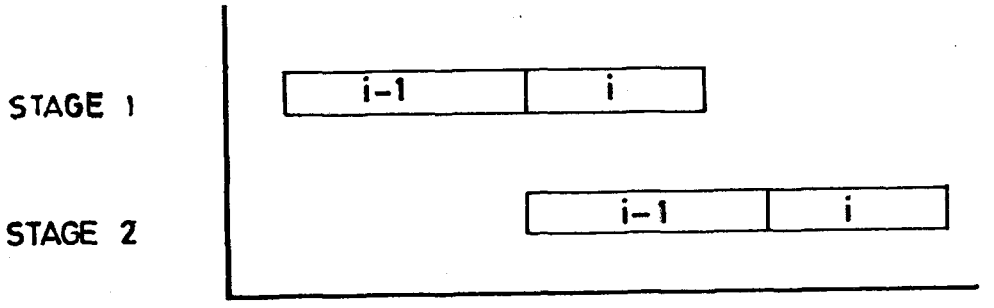
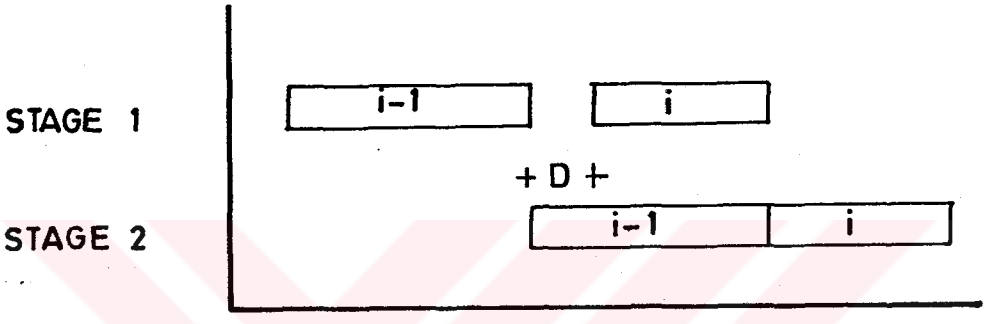


Figure 4.4. Schedule obtained by reversing the reversed schedule (Delay at stage two).

Now, consider a schedule having a delay at stage 2. Suppose that this original schedule and the schedule obtained by reversing the original one are illustrated in Figures 4.5.a and 4.5.b, respectively.



4.5 (a) Original schedule



4.5 (b) Schedule obtained by reversing

Figure 4.5. Original and Reversed Schedules (No delay at stage two).

If the above schedules are considered, it is observed that the maximum flow time is again unchanged. However, there is a delay between (i-1)th and i th jobs at stage 1. There is no need to start the processing of i th job after waiting "D" units of time at stage 1. Therefore, the jobs processed after (i-1)th job may be left shifted in the schedule obtained by reversing. This yields the original schedule at stage 1.

In general, by reverse scheduling,

(1) the maximum flow time is unchanged,

(2) between-jobs delays are eliminated from stage 2,

(3) run-out delay of stage 1 in the reversed schedule or the run-in delay in the schedule obtained by reversing is the total delay at stage 2 in the original schedule.

4. 3. 1. Application of the Reverse Scheduling to the Two-Stage Flow Shop :

STEP 0 : - Form sets S_1 and S_2.

- Apply the following steps to the respective schedules for sets S_1 and S_2 independently.

STEP 1 : For set S_1 :

- Find a permutation schedule such that the jobs are in nondecreasing value of their processing times at stage 1.

- Find the completion times of the last job in the processing sequence at stage 1 and 2.

STEP 2 : For set S_2 :

- Find a permutation schedule such that the jobs are in nonincreasing value of their processing times at stage 2.

- Find the completion times of the last job in the processing sequence at stage 1 and 2.

STEP 3 : For each set :

- Compute the differences between the completion times of the last job at stage 1 and 2.
  (i.e. run-out delays of the respective schedules)

STEP 4 : Compute the time difference between the run-out delays in the respective schedules for two sets.

STEP 5 : Convert the reverse schedule of set S_2 to the original one (job sequence will be reversed while reverse schedule is converted).

STEP 6 : To prevent the unnecessary idle times at stage 1, shift the jobs to the left (Job shifting is always possible since the run-out delay in reversed schedule for set S_2 is less than the run-out delay in the schedule for set S_1).

STEP 7 : If the run-out delay in the reverse schedule for set S_2 is greater than the run-out delay in the schedule for set S_1.

- Compute the maximum flow time of the schedule for all jobs in set S_1 and S_2 as :

  F_{max} = (Maximum Flow Time of the Schedule for Set S_1) + (Time Difference Between Run-out Delays) + (Sum of the Processing Times of Jobs in Set S_2 at Stage 2)

Otherwise,

- Compute the maximum flow time of the schedule for all jobs in sets S_1 and S_2 as :

  F_{max} = (Maximum Flow Time of the Schedule for Set S_1) + (Sum of the

Processing Times of Jobs in Set S_2 at Stage 2)

4. 3. 2. Example : Reverse Scheduling

The above algorithm can be illustrated by an example. The processing times for five jobs at each of two stages are shown in Table 4.1.

Table 4.1.

	Processing Times	
Job	Stage 1	Stage 2
i	t_{i1}	t_{i2}
1	4	3
2	1	2
3	8	5
4	5	6
5	2	3

Applying Johnson's algorithm the minimum makespan sequence is found as 2-5-4-3-1 . The Gantt chart for this sequence, shown in Figure 4.6, reveals a makespan of 24 time units.

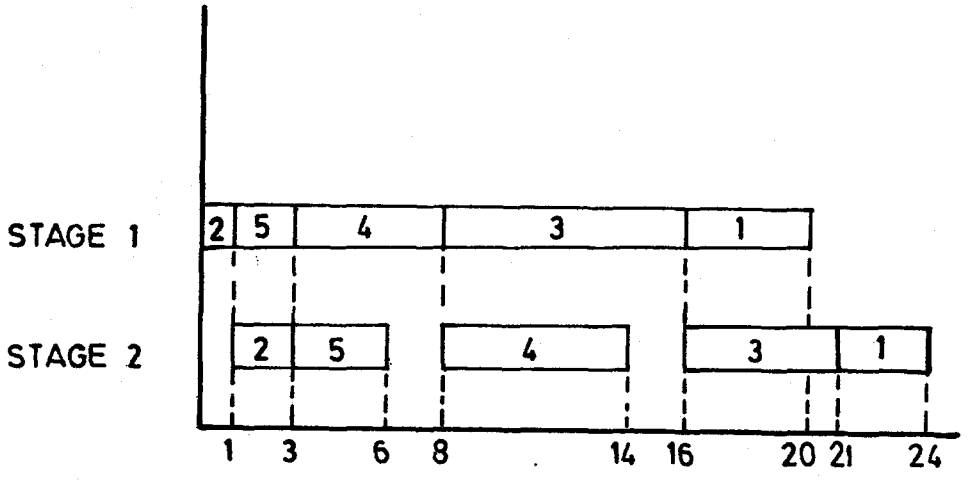


Figure 4.6. Gantt Chart for the 5/2/P/F_{max} example.

Applying the reverse scheduling algorithm presented above, two sets are constructed as :

$$S_1 = \{ 2,4,5 \} \quad \text{and} \quad S_2 = \{ 1,3 \}$$

The minimum makespan sequence for two sets are 2-5-4 and 1-3 , respectively.

Let C_j^k be the completion time of the last job in the sequence for set k at stage j and,

R^k be the run-out delay occurs in the schedule for set k. Then,

$$C_1^1 = 8 \ , \ C_2^1 = 14 \quad \text{and} \quad C_1^2 = 8 \ , \ C_2^2 = 16$$

Note that the completion times of jobs in set S_2 are given from the reversed schedule.

The result of step 3 is the following run-out delays :

$$R^1 = C_2^1 - C_1^1 = 14 - 8 = 6 \quad \text{and,}$$

$$R^2 = C_2^2 - C_1^2 = 16 - 8 = 8$$

Since $R^2 > R^1$ there would be a delay of 2 time units before processing job 3 at stage 2. The permutation schedule for the jobs in set S_2, obtained by reverse scheduling, is shown in Figure 4.7.

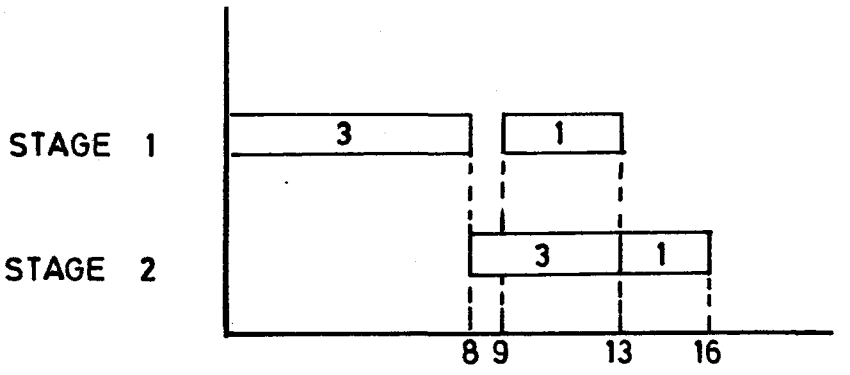
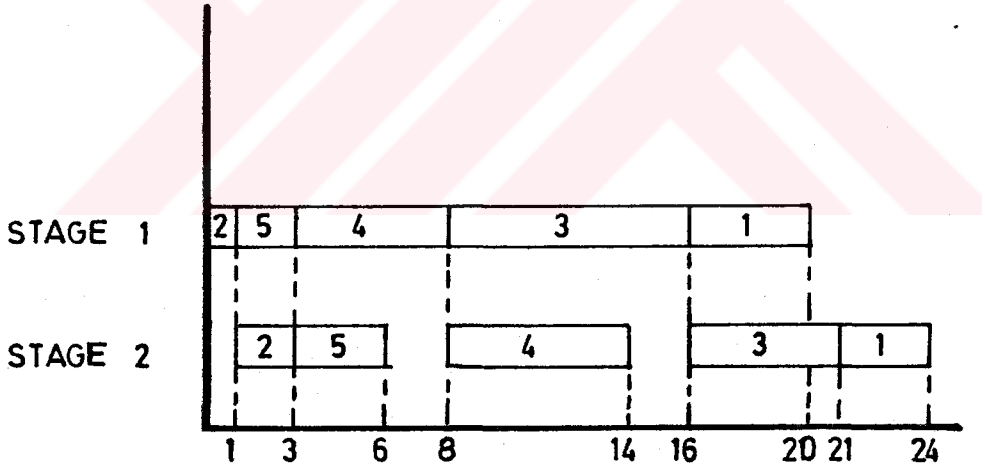


Figure 4.7. Permutation schedule for jobs in set S_2 (By reverse scheduling).

After left shifting of jobs at stage 1 of the above schedule a permutation schedule for all jobs in two sets can be depicted in the following figure :



$$F_{max} = C_2^1 + (R^2 - R^1) + \sum_{i=1}^{5} t_{i2} = 14 + 2 + 8 = 24$$

Figure 4.8. Permutation schedule for all jobs in example (By reverse scheduling).

By reverse scheduling, it is demonstrated that the outcomes of the schedules, original and reversed are the same when two schedules in Figures 4.6 and 4.8 are compared. This allows us to evaluate the job set S_2 as the first set S_1, and consequently the application of Lap-Phasing method can be facilitated for both sets in a similar way hence simplifying the feasibility improvement process.

4. 4. Some Possible Cases That Occur in Lap-Phasing:

When the lower bounds on the sublot sizes (minimum sublot sizes) are considered, one can state some cases. Since the sublot sizes will be nondecreasing for each job in set S_1, it is enough to check the initial and the last sublot sizes to compare with the minimum lot size. Therefore, there will be four cases to be considered in lap-phasing of the jobs in set S_1 :

Case 1 : Both initial and the last sublot sizes are greater than the minimum required sublot size.

Case 2 : Initial sublot size is greater than the minimum sublot size, however, the last sublot size is less.

Case 3 : Both initial and the last sublot sizes are less than the minimum sublot size.

Case 4 : Initial sublot size is less than the minimum sublot size although the last sublot size is greater.

Case 1 :

Both initial and the last sublot sizes of a job are greater than the minimum sublot size. i.e.

$$Q_{i1} \geq m_i \quad \text{and} \quad Q_{i,s_i} \geq m_i$$

where,

Q_{i1} is the size of the initial sublot of job (item) i,

m_i is the minimum sublot size of item i,

s_i is the number of sublots of item i,

Q_{i,s_i} is the size of the last sublot of item i,

h_i is the amount of item i.

Since $p_{i2} \geq p_{i1}$; $Q_{ij} \geq Q_{i,j-1}$ and $Q_{ij} \geq m_i$ for $j = 2,..,s_i-1$

$$Q_{i,s_i} = h_i - \sum_{j=1}^{s_i-1} Q_{ij} \geq Q_{i,s_i-1} \quad \text{or} \quad Q_{i,s_i} < Q_{i,s_i-1}$$

This case is graphically represented in Figure 4.9.

In such a case :

(i) all sublots can be accepted as they are, or

(ii) if it is possible, the identical sublots (sublots having equal units of item i) can be formed.

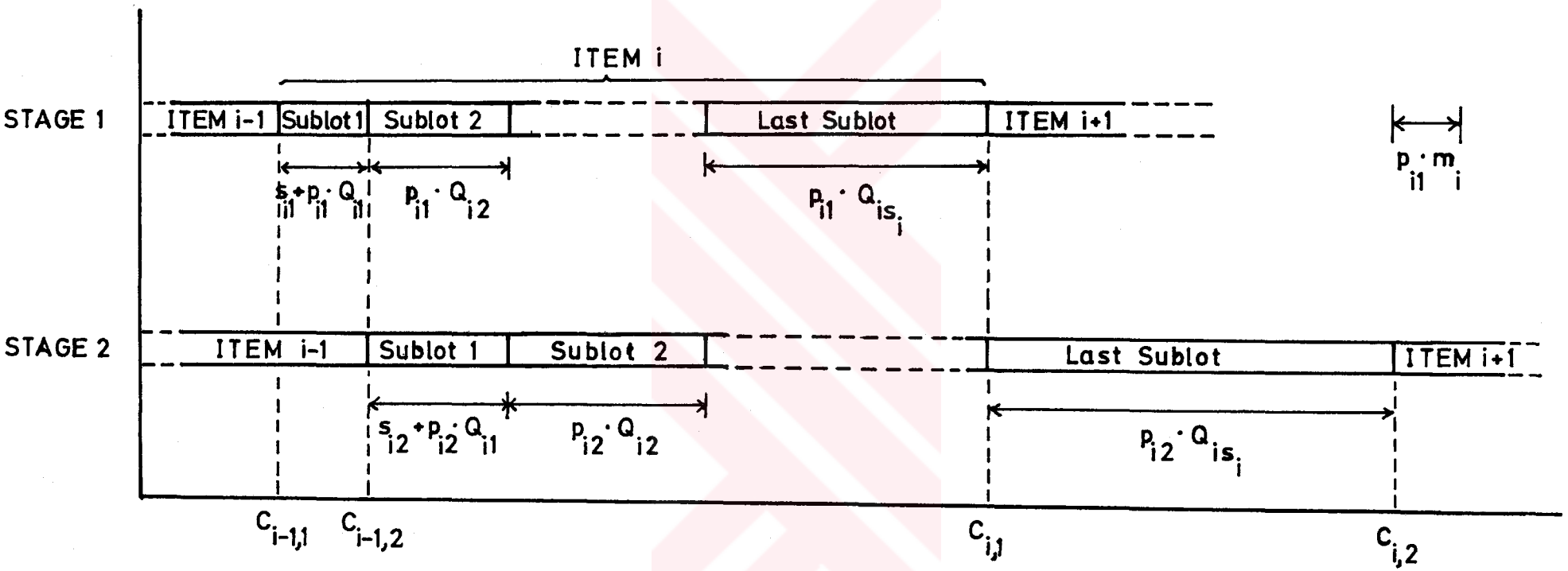


Figure 4.9. Graphical Representation of Case 1.

Check for the Possibility of Identical Sublots :

Let n_i be the lower limit on the number of sublots of item i and define e_i as :

$$e_i = h_i / n_i$$

where,

$$h_i = \sum_{j=1}^{s_i} Q_{ij}$$

For $n_i = s_i$, if e_i is an integer value, and

$$Q_{i1} \geq e_i \quad \text{and} \quad e_i \geq m_i$$

then for item i, n_i identical sublots having e_i units of the item i can be formed.

If e_i is not integer, then n_i identical sublots can not be formed. However, it is possible to have identical sublots having Q_{i1} units of item i or less.

Let n_f be the number of identical sublots having Q_{i1} units of item i. i.e.,

$$n_f = h_i / Q_{i1}$$

If n_f is an integer value, then n_f identical sublots having Q_{i1} units of item i can be formed. If not, it must be checked whether there exists an integer number e_i such that

$$h_i / e_i \text{ is integer} \quad \text{and} \quad Q_{i1} > e_i \geq m_i$$

If there exists more than one solution to the above, the one

having the maximum value should be the number of units in each sublot.

The following procedure states the computational routine of the check for the possible identical sublots :

Procedure 1 (Identical Sublots):

Step 1 : Let $n_i = s_i$,

Step 2 : Let $e_i = h_i / n_i$

Step 3 : If e_i is integer such that

$$Q_{i1} \geq e_i \geq m_i$$

n_i identical sublots having e_i units of item i can be formed, Stop.

Otherwise, goto step 4.

Step 4 : Let $n_f = h_i / Q_{i1}$

Step 5 : If n_f is integer then n_f identical sublots having Q_{i1} units of item i, Stop.

Otherwise, goto step 6

Step 6 : Find an integer value for e_i such that

h_i / e_i is integer and $Q_{i1} > e_i \geq m_i$

Step 7 : If there exist a value for e_i, then h_i/e_i identical sublots having e_i units of item i can be formed, Stop.

Otherwise,

identical sublots can not be formed.

Stop.

Let us now illustrate above through an example. Consider the following problem where,

$C_{i-1,1}$ = Completion time of (i-1) th item in sequence at stage 1
= 910

$C_{i-1,2}$ = Completion time of (i-1) th item in sequence at stage 2
= 1200

s_{i1} = setup time incurred at stage 1 before processing i th item
= 20

s_{i2} = setup time incurred at stage 2 before processing i th item
= 22

p_{i1} = unit processing time of i th item at stage 1
= 5

p_{i2} = unit processing time of i th item at stage 2
= 7

h_i = Lot size of i th item (production run quantity)
= 216 units

m_i = minimum sublot size of i th item
= 40 units

s_i = number of sublots of item i
= 3

Q_{i1} = number of units of item in sublot 1
= 54

Q_{i2} = number of units of item in sublot 2
= 80

Q_{i3} = number of units of item in sublot 3
= 82

The graphical representation of this example is given in Figure 4.10.

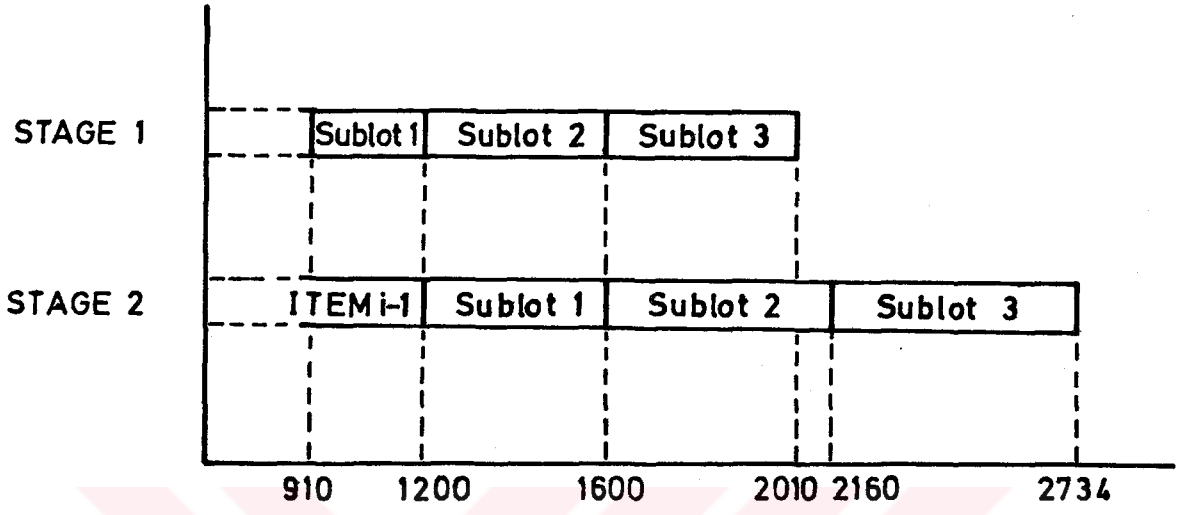


Figure. 4.10. Graphical Representation of Sublot Scheduling in Sample Problem Illustrating Procedure 1.

If procedure 1 is applied to check for the possibility of identical sublots, its steps will be as folows:

S. 1 : $n_i = s_i = 3$ sublots

S. 2 : $e_i = h_i / n_i = 216 / 3 = 72$ units

S. 3 : e_i is integer. Since it is greater than the initial sublot size, it is not possible that three identical can be formed. Goto step 4.

S. 4 : Let $n_f = h_i / Q_{i1} = 216 / 54 = 4$

Since n_f is integer, 4 identical sublots having 54 units of item i can be formed.

Case 2 :

Initial sublot size is greater than the minimum sublot size, however, the last sublot size is less. i.e.

$$Q_{i1} \geq m_i \quad \text{and} \quad Q_{is_i} < m_i$$

Then, it can be concluded that the intermediate sublots are also greater than the minimum sublot size. That is,

Since $p_{i2} \geq p_{i1}$, $Q_{ij} \geq Q_{i,j-1}$ and $Q_{ij} \geq m_i$ for

$$j = 2,..,s_i-1$$

This case is graphically represented in Figure 4.11.

At this point, since all the sublot sizes are greater than the minimum sublot size except the last sublot, there may exist three possible situations where ;

(1) Initial sublot size is greater than the possible identical size e_i of n_i sublots where the identical sublot size is greater than the minimum sublot size.

(2) Initial sublot size is greater than the possible identical size e_i of n_i sublots where the identical sublot size is less than the minimum sublot size.

(3) Initial sublot size is less than the possible identical size e_i of n_i sublots.

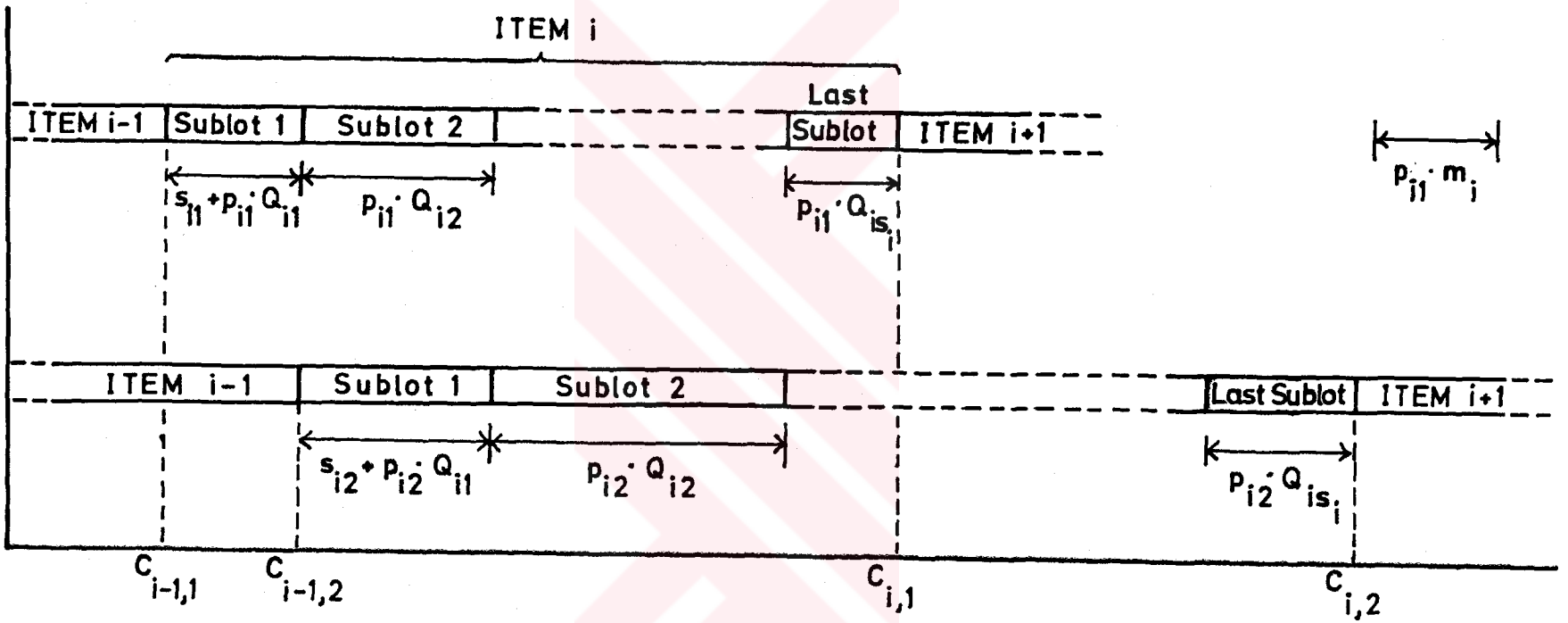


Figure 4.11. Graphical Representation of Case 2.

Situation 2. 1 : Initial sublot size is greater than the possible identical size e_i of n_i sublots where the identical sublot size is greater than the minimum sublot size. i.e.

$$Q_{i1} \geq e_i \quad \text{and} \quad e_i \geq m_i$$

At this point, a decision must be made whether to

(i) form identical sublots (if possible), or

(ii) re-arrange the sublot sizes and try to eliminate the possible delays between the processing of sublots at stage 1.

For the first choice, apply the procedure given in case 1 to check whether all sublot sizes can be made greater than the minimum sublot size by forming identical sublots without causing delays at stage 1.

For the second choice, the last sublot size is set to the minimum sublot size requirement and, starting from the sublot before the last one, all previous sublot sizes are re-arranged.

Based on the idea stated above, the following extra notation and definitions are required before stating the procedure :

k : index for the sublots

h_{left} : number of units of item which is not allocated to any one of the sublot

Procedure 2 (Re-arrangement of the Sublot Sizes):

Step 1 : Set $k = n_i$, $Q_{ik} = m_i$, $h_{left} = h_i - Q_{ik}$

Step 2 : (i) Decrease the number of sublots which are not re-arranged yet.

$$k = k - 1$$

(ii) Compute the identical sublot size as

$$e_i = h_{left} / k$$

Step 3 : For the sublots which are not re-arranged, check whether the identical sublot sizes can be formed or not.

If $e_i > Q_{i1}$ then

(i) Add one more unit of item i to the last sublot k-1 which is already re-arranged

$$k = k + 1$$

$$Q_{ik} = Q_{ik} + 1$$

$$h_{left} = h_{left} - 1$$

(ii) Goto step 2

Otherwise, goto step 4.

Step 4 : Check whether it is possible to have identical sublot sizes for the sublots which are not re-arranged yet.

If $e_i = \lfloor e_i \rfloor$ then

(where $\lfloor e_i \rfloor$ denotes the largest integer less than or equal to e_i)

(i) k identical sublots having e_i units of item i can be formed.

$$Q_{ik} = \ldots = Q_{i1} = e_i$$

(ii) Stop

Otherwise, goto step 5.

Step 5 : (i) Take the sublot size of the last sublot which is already re-arranged as the sublot size for sublot k.

$$Q_{ik} = Q_{i,k-1}$$

$$h_{left} = h_{left} - Q_{ik}$$

(ii) Goto Step 2

This procedure may be illustrated by the example given in case 1. Now, suppose that the lot size is 149 units instead of 216. This time, the sublots is as follows:

$$Q_{i1} = 54 \quad , \quad Q_{i2} = 80 \quad , \quad Q_{i3} = 15$$

The last sublot size is less than the minimum sublot size and the identical sublot size for three sublots is greater than the minimum sublot size. That is,

$$Q_{i3} = 15 < m_i = 40 \quad \text{and} \quad e_i = 149 / 3 \approx 49.67 > m_i = 40$$

The possible identical sublots may be first checked by applying PROCEDURE 1. The steps are:

S. 1 : $n_i = 3$

S. 2 : Although $Q_{i1} = 54 > e_i \approx 49.67 > m_i = 40$ it is not integer. So, 3 identical sublots can not be formed.

S. 3 : $n_f = 149 / 54 \approx 2.76$

S. 4 : Since n_f is not integer, identical sublots having Q_{i1} units of item can not be formed.

S. 5 : Since identical sublots can not be formed in range $[m_i, Q_{i1}]$, Procedure 2 may be applied. Its steps are as folows :

S. 1 : k = number of sublots which are not re-arranged = 3

$Q_{i3} = m_i = 40$

$h_{left} = h_i - Q_{i3} = 149 - 40 = 109$

S. 2 : $k = k - 1 = 3 - 1 = 2$

$e_i = h_{left} / k = 109 / 2 = 54.5$

S. 3 : $e_i = 54.5 > Q_{i1} = 54$

$k = k + 1 = 2 + 1 = 3$

$Q_{i3} = Q_{i3} + 1 = 40 + 1 = 41$

$h_{left} = h_{left} - 1 = 109 - 1 = 108$

S. 2 : $k = 3 - 1 = 2$

$e_i = 108 / 2 = 54$

S. 3 : Since $e_i = Q_{i1} = 54$ goto step 4.

S. 4 : $e_i = |e_i| = 54$. i.e. e_i is integer and two identical sublots having 54 units can be formed.

$Q_{i1} = Q_{i2} = 54$ units and $Q_{i3} = 41$ units

As a result, identical sublots can be formed by applying Procedure 2.

Situation 2. 2 : Initial sublot size is greater than the possible identical size e_i of n_i sublots where the identical sublot size is less than the minimum sublot size. i.e.

$$Q_{i1} \geq e_i \quad \text{and} \quad e_i < m_i$$

In such a situation, it can be stated that the identical sublots can not be formed without causing a delay at the second stage. Since the size of the possible identical sublots is less than the minimum sublot size requirement, production with identical sublots having m_i units of item i is also impossible. Therefore, a delay can not be avoided to satisfy the minimum sublot size requirement. To minimize the delay at stage 2, since the last sublot size is less than the minimum sublot size requirement, the last sublot might be dropped and distributed to the previous sublots.

The delay at stage 2 may be represented by the following inequality and illustrated by Figure 4.12.

$$\left[\begin{array}{c}\text{Completion time of}\\ \text{sublot k-1 at stage 2}\end{array}\right] \leq \left[\begin{array}{c}\text{Completion time of}\\ \text{sublot k at stage 1}\end{array}\right]$$

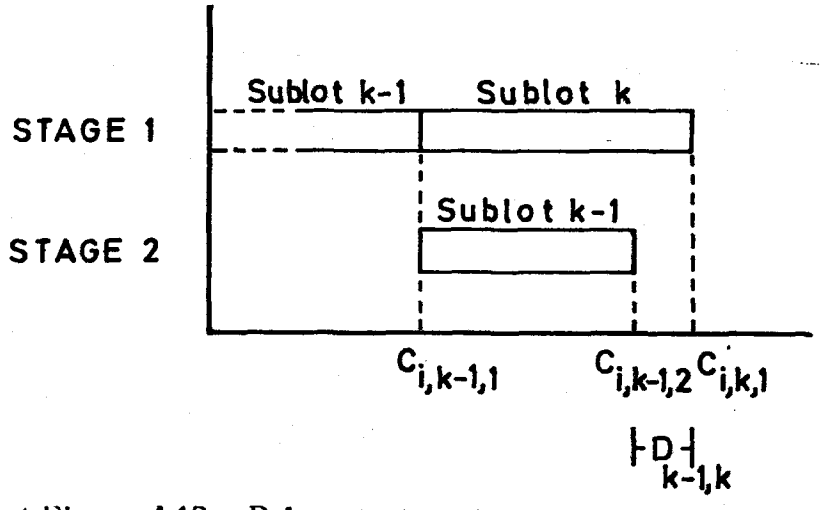


Figure 4.12. Delay at stage two.

Then, the above inequality may be rewritten as an equality

$$C_{i,k-1,2} + D_{k-1,k} = C_{ik1} \qquad \text{for } k = 2,\ldots,n_i-1 \quad (1)$$

where,

C_{ik1} is the completion time of sublot k of item i at stage 1,

$D_{k-1,k}$ is the delay between sublots k-1 and k,

Since $C_{i,k-1,2} = C_{i,k-1,1} + t_{i,k-1,2} + p_{i2} \cdot X_{k-1}$ (2)

and

$$C_{i,k,1} = C_{i,k-1,1} + t_{i,k,1} + p_{i1} \cdot X_k \quad (3)$$

where,

$t_{i,k-1,j}$ is the total processing time of sublot k-1 at stage j,

p_{ij} is the unit processing time of item i at stage j,

X_k is the amount of last sublot allocated to sublot k.

Then, the equation (1) may be written by using the equations (2) and (3) as follows:

$$t_{i,k-1,2} + p_{i2} \cdot X_{k-1} + D_{k-1,k} = t_{i,k,1} + p_{i1} \cdot X_k \qquad \text{for } k = 2,\ldots,n_i-1 \quad (4)$$

Since the last sublot is also allocated to the initial sublot, there will be a delay between the start of processing of the initial sublot of item i and the completion of the processing of the last sublot of item i-1 at stage 2. This can be represented as follows:

$$C_{i-1,2} + d = C_{i11} + p_{i1} \cdot X_1 \qquad (5)$$

where,

$C_{i-1,2}$ is the completion time of item i-1 at stage 2

d is the delay between the processing of item i-1 and the inital sublot of item i

In addition, sum of allocated amounts should be equal to the last sublot size which will be dropped. i.e.,

$$\sum_{i=1}^{n_{i-1}} X_k = Q_{i,n_i} \qquad (6)$$

Then, the problem is formulated as a MIP model whose objective function is to minimize the total delay that occurs at stage 2 because of dropping the last sublot.

MIP model becomes

$$\text{minimize} \quad d + \sum_{k=2}^{n_{i-1}} D_{k-1,k}$$

subject to constraints (4),(5),(6) and nonnegativity and integrality

At this point, to determine the sublot sizes, the following procedure is suggested :

Procedure 3 (Lot size allocation) :

Step 1 : Relax the integrality constraints from the model.

Step 2 : Solve the model by the Simplex algoritm.

Step 3 : If the solution is all integer that satisfies the constraints, then goto Step 5.

Otherwise,

Step 4 : Round off the resulting solution to one that is all integer and satisfies the constraints.

Step 5 : Add the allocated sublot sizes to all sublots

Step 6 : Stop

This procedure may be illustrated by the example given in situation 2.1. Now, suppose that the minimum sublot size is 51 units instead of 40. This time, the last sublot and identical sublot sizes are both less than the minimum sublot size. That is,

$Q_{i3} = 15 < m_i = 40$ and $e_i = 149 / 3 \approx 49.67 < m_i = 51$

Then, the mixed integer model for this example becomes :

Minimize $d + D_{1,2}$

subject to

$$7 X_1 + D_{1,2} = 5 X_2$$

$$d = 5 X_1$$

$$X_1 + X_2 = 15$$

$$d, D_{1,2}, X_1, X_2 \geq 0 \text{ and integer}$$

The solution for this model is

$X_1 = 6$, $X_2 = 9$, $d = 30$, $D_{1,2} = 3$

This result shows that i th item is processed in two sublots having 60(= 54+6) and 89(= 80+9) units, respectively. Consequently, delay between the processing of (i-1) th item and the initial sublot

of i th item, and delay between sublots 1 and 2 become 30 and 3 time units, respectively. In Figure 4.13, this result is graphically represented.

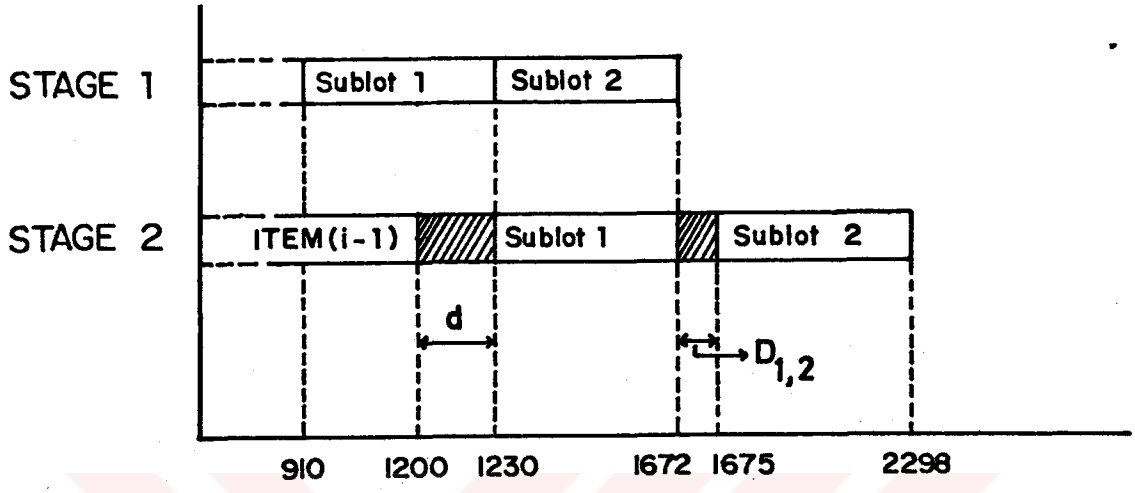


Figure. 4.13. Graphical Representation of Sublot Scheduling in Sample Problem Illustrating Procedure 3.

Situation 2. 3 : Initial sublot size is less than the identical size e_i of n_i sublots. i.e.

If $Q_{i1} < e_i$, then $e_i > m_i$ since $Q_{i1} > m_i$. Thus, this case may be represented as $m_i < Q_{i1} < e_i$

Since increases in Q_{i1} lead to some delays, the identical sublots having Q_{i1} or less units of item i can be formed. However, the identical sublot size can be decreased down to the minimum sublot size requirement. Therefore, the identical sublot size is in between the range m_i and Q_{i1} units of item i. To form identical sublot sizes the following procedure is suggested :

Procedure 4 (Forming sublots in range $[m_i, Q_{i1}]$):

Step 1 : Let $n_1 = h_i / Q_{i1}$

Step 2 : Check whether n_1 identical sublots having Q_{i1} units can be formed.

If $n_1 = |n_1|$ then Goto Step 4

Otherwise, goto step 3.

Step 3 : n_1 identical sublots having Q_{i1} units of item i can be formed.

(i) $Q_{i1} = \ldots = Q_{i,n_i} = Q_{i1}$

(ii) Stop

Step 4 : Let $n_m = h_i / m_i$

Step 5 : Check whether n_m identical sublots having m_i units can be formed.

If $n_m = |n_m|$ then

(i) $n_m = |n_m|$

(ii) Goto Step 7

Otherwise, goto step 6.

Step 6 : n_m identical sublots having m_i units of item i can be formed.

(i) $Q_{i1} = \ldots = Q_{i,n_m} = m_i$

(ii) Stop

Step 7 : Find a non-negative integer value for k such that

$e_i = h_i / k$

where,

e_i and k are integers and $n_1 \leq k \leq n_m$

Step 8 : If there exists more than one solution for k satisfying the conditions in Step 7, choose k^* as a lot size for identical sublots such that

$$k^* = \min \{ k \}$$

(i) $Q_{ik} = e_i$ for $k = 1,\ldots, k^*$

(ii) Stop

Otherwise, goto step 9.

Step 9 : Check whether it is possible to have sublots where there is no delay between their processing at stage two.

(i) Let $f = Q_{i1} - m_i$

(ii) $h_{left} = h_i - n_m \cdot m_i$

(iii) If $h_{left} > f \cdot n_m$ then

(a) n_m-1 identical sublots having Q_{i1} units of item i are formed i.e.

$$Q_{ik} = Q_{i1} \quad \text{for } k = 1,\ldots,n_1$$

(b) The last sublot n_1 will have a lot size of

$$Q_{i1} + (h_i - n_1 \cdot Q_{i1}) \text{ units.}$$

(c) Stop.

Otherwise, goto step 10.

Step 10 : Let $k = 1$; $l = n_m$; $Q_{ik} = m_i$

Step 11 : Allocate h_{left} to the sublots

(i) Let $e_d = h_{left} / l$

(ii) If $e_d = |e_d|$ then

$$Q_{ik} = \ldots = Q_{i,n_m} = Q_{ik} + e_d$$

Stop.

Otherwise, goto step 12.

Step 12 : (i) $e_d = |e_d| + 1$

$Q_{ik} = Q_{ik} + e_d$

$h_{left} = h_{left} - e_d$

$l = l - 1$

$k = k + 1$

(ii) Goto Step 11

This procedure may be illustrated by the example given in case 1. Now, suppose that the lot size is 172 units instead of 216. This time, the sublots is as follows:

$$Q_{i1} = 54 \quad , \quad Q_{i2} = 80 \quad , \quad Q_{i3} = 38$$

The last sublot size is less than the minimum sublot size and the identical sublot size for three sublots is greater than the initial sublot size. That is,

$$Q_{i3} = 38 < m_i = 40 \quad \text{and} \quad e_i = 172 / 3 \approx 57.33 > Q_{i1} = 40$$

Then, the identical sublot sizes can be formed which is in between the range 40 and 54 units of item following the steps of the procedure 4 as :

S. 1 : Let $n_1 = h_i / Q_{i1} = 172 / 54 \approx 3.185$

S. 2 : Since n_1 is not integer, n_i identical sublots having 54 units can not be formed.

Set $n_1 = |3.185| = 3$

S. 4 : $n_m = h_i / m_i = 172 / 40 = 4.3$

S. 5 : Since n_m is not integer, n_m identical sublots having 40 units can not be formed.

Set $n_m = |4.3| = 4$

S. 7 : For $k = n_m = 4$, $e_i = h_i/k = 172/4 = 43$ is integer

S. 8 : Four identical sublots having 43 units can be formed. That is,

$$Q_{i1} = Q_{i2} = Q_{i3} = Q_{i4} = e_i = 43 \text{ units}$$

Case 3 :

Both initial and the last sublot sizes are less than the minimum sublot size. i.e.

$$Q_{i1} < m_i \quad \text{and} \quad Q_{i,s_i} < m_i$$

In such a case, there is a possibility of getting intermediate sublots that satisfies the minimum sublot size requirement.

This case is graphically represented in Figure 4.14.

At this point, a decision must be made whether to

(i) accept the delay that occurs before the processing of the initial sublot at stage 2 because of the increase in the lot size to satisfy the minimum sublot size requirement, or

(ii) change the position of item i in the processing sequence as the last item to be processed.

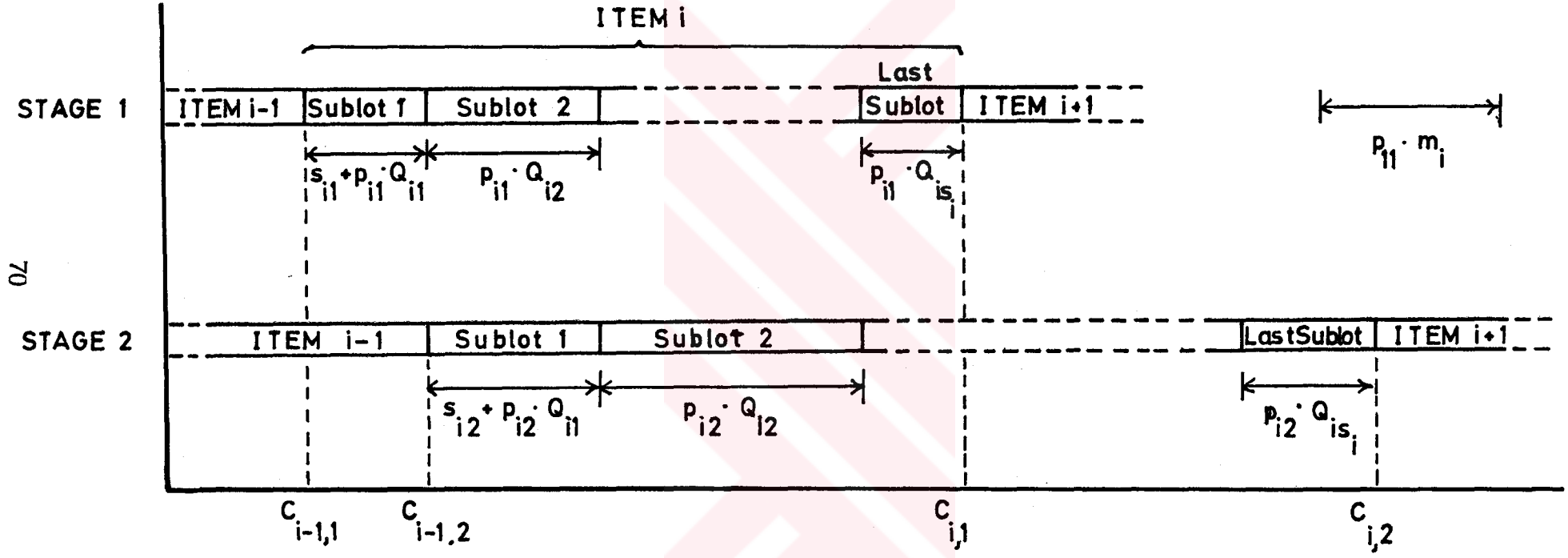


Figure 4.14. Graphical Representation of Case 3.

For the first choice, the following procedure may be followed:

Procedure 5:

Step 1 : (i) Let $q = m_i$

(ii) Set total delay at stage 2 to zero.

Step 2 : $k = \lfloor h_i / q \rfloor$

$h_{left} = h_i - k \cdot q$

Step 3 : Check whether k identical sublots having q units of item i can be formed.

If $h_{left} = 0$ then

(i) k identical sublots having q units of item i can be formed.

$Q_{ik} = \ldots = Q_{ik} = q$

(ii) Stop.

Otherwise, goto step 4.

Step 4 : (i) For the first k-1 sublots :

$Q_{i1} = \ldots = Q_{i,k-1} = q$

Step 5 : (ii) For the last sublot k :

$Q_{ik} = q + h_{left}$

Step 6 : Check whether there is a delay at stage 2 before processing the last sublot k.

Step 7 : If there is no delay,

(i) Compute the total delay at stage 2 for the current schedule.

(ii) Compare the total delays of the current and previous schedules.

If current total delay is less than the previous one, then the sublot sizes are

$Q_{i1} = \ldots = Q_{i,k-1} = q$ and $Q_{ik} = q + h_{left}$

Stop.

Otherwise,

$Q_{i1} = \ldots = Q_{i,k-1} = q-1$ and

$Q_{ik} = q + h_{left} + k$

Stop.

Otherwise, goto step 8.

Step 8 : (i) Compute the total delay at stage 2

(ii) $q = q + 1$

(iii) Goto Step 2

This procedure may be illustrated by the example given in

case 1. Now, suppose that the minimum sublot size is 85 units instead of 40. This time, the initial and last sublot sizes are both less than the minimum sublot size. That is,

$$Q_{i1} = 54 < m_i = 85 \quad \text{and} \quad Q_{i3} = 82 < m_i = 85$$

The steps of the procedure are followed as

S. 1 : Let $q = m_i = 85$

S. 2 : $k = \lfloor h_i / q \rfloor = \lfloor 216 / 85 \rfloor = \lfloor 2.541 \rfloor = 2$

$h_{left} = h_i - k \cdot q = 216 - 2(85) = 46$

S. 3 : Since $h_{left} = 0$, 2 identical sublots having 85 units of item i can not be formed.

S. 4 : Since $k = 2$, $Q_{i1} = q = 85$ units

S. 5 : $Q_{i2} = q + h_{left} = 85 + 46 = 131$ units

S. 6 : There is a delay at stage 2, which is 38 time units, before processing the last sublot.

S. 7 : Goto step 8

S. 8 : $q = q + 1 = 85 + 1 = 86$

Repeating the procedure for q equals to 86, 87, 88 units, there are delays at stage 2 before processing the last sublot. For q equals to 89, there is no delay at the second stage before processing the last sublot. However, total delay becomes 175 time units. Since this amount of delay at stage 2 is greater than the total delay observed in the schedule where total delay equals to 172 time units. Then, the initial and the last sublot sizes are 88 and 128 units, respectively. That is,

$$Q_{i1} = 88 \text{ units} \quad \text{and} \quad Q_{i2} = 128 \text{ units}$$

The solution for this problem is graphically illustrated in Figure 4.15.

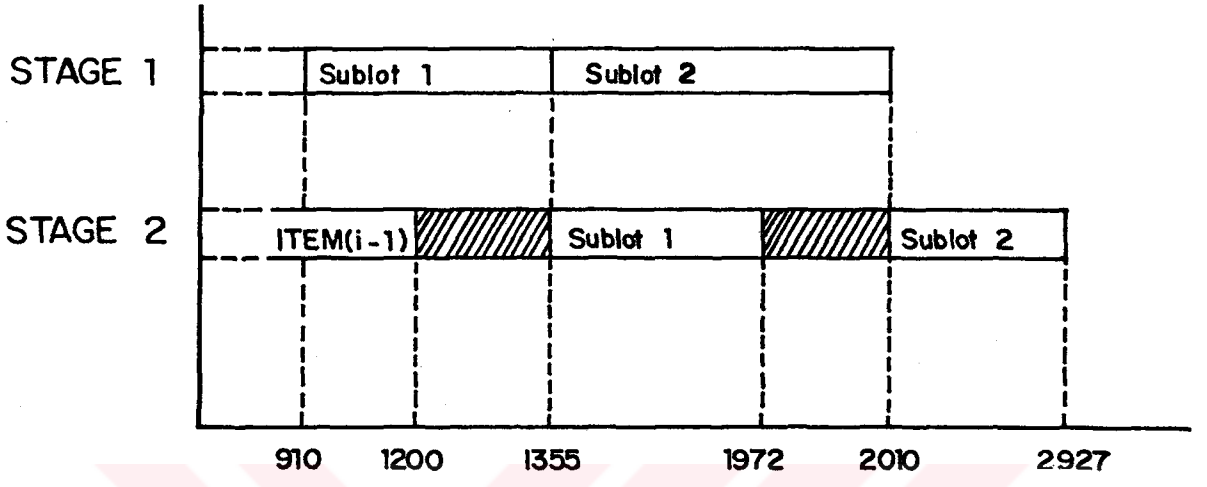


Figure. 4.15. Graphical Representation of Sublot Scheduling in Sample Problem Illustrating Procedure 5.

Case 4 :

Initial sublot size is less than the minimum sublot size although the last sublot size is greater. That is,

$$Q_{i1} < m_i \ , \quad Q_{ij} \geq Q_{i,j-1} \quad \text{and} \quad Q_{ij} \geq m_i \ \text{for} \ j = 2,..,s_i-1$$

This case is graphically represented in Figure 4.16.

In such a case procedure 5 is also valid since the initial sublot size is less than the minimum sublot size.

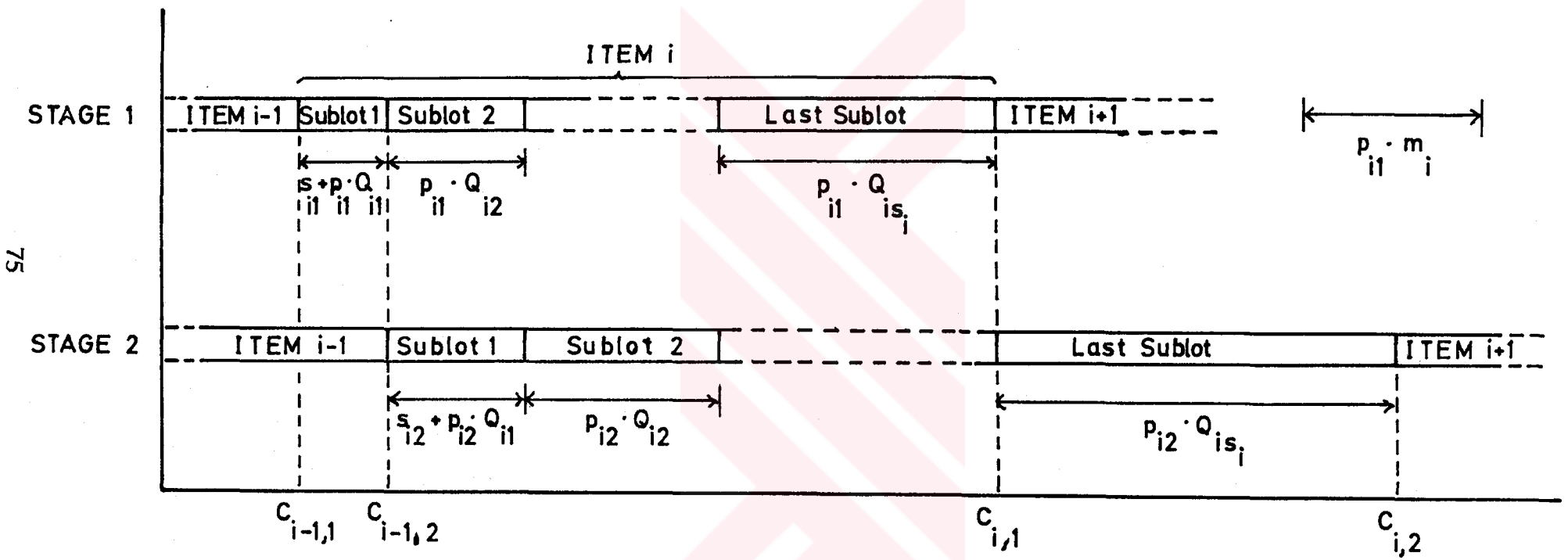


Figure 4.16. Graphical Representation of Case 4.

4. 5. Integrating Family Setups to The Lower Bounds on The Maximum Flow Times

In the previous chapters, the lower bound on the maximum flow time for a permutation schedule in a two stage flow shop system is derived as:

$$LB_{F_{max}} = \max \{ F^1_{max} , F^2_{max} \}$$

where,

$$F^1_{max} = \sum_{i=1}^{N} t_{i1} + t_{N2}$$

$$F^2_{max} = \sum_{i=1}^{N} t_{i2} + t_{11}$$

t_{ij} is the total processing time of the i th job (item) in the processing sequence at stage j.

While applying the lap-phasing to the respective schedules of two sets of jobs(items) in a family, the items, which are processed in the first position in their respective schedules, have initial sublots which are equal to their minimum batch sizes. These initial sublots may be considered as first and last jobs (items) to be processed in a set of jobs(items) belonging to a family. Therefore, the definitions related to the first and last jobs may be changed as:

t_{11} is the processing time of the initial sublot of the first job (item) in the processing sequence at stage 1,

t_{N2} is the processing time of the last sublot of the last job (item) in the processing sequence at stage 1,

The above bound can be represented in a different way by taking the difference of the possible maximum flow times. This result is stated in the following corollary:

Corollary 3 :

For a permutation schedule in a two-stage flow shop system, the difference between the possible lower bounds on the maximum flow times is:

$$\Theta = F^1_{max} - F^2_{max} = \left(\sum_{i=1}^{N} t_{i1} + t_{N2} \right) - \left(\sum_{i=1}^{N} t_{i2} + t_{11} \right)$$

$$= \sum_{i=2}^{N} t_{i1} - \sum_{i=1}^{N-1} t_{i2}$$

Then, the following is true:

$$\text{If } \sum_{i=2}^{N} t_{i1} \geq \sum_{i=1}^{N-1} t_{i2} \text{ then } F^1_{max} > F^2_{max} \text{ and } LB_{F_{max}} = F^1_{max}$$

Otherwise,

$$LB_{F_{max}} = F^2_{max}$$

For the discussion of how the lower bound on the run out delay of a schedule can be determined, the following definition may be introduced :

Let R be the run out delay of the schedule,

LB_R be the lower bound on the run out delay.

Now, consider a schedule where $F^1_{max} > F^2_{max}$. We have shown that

$$LB_{F_{max}} = F^1_{max}$$

Thus, for such a schedule, the lower bound on the run out delay is the time required to perform the second operation of the last job (item) in the sequence. Explicitly,

$$LB_R = LB_{F_{max}} - \sum_{i=1}^{N} t_{i1} = F^1_{max} - \sum_{i=1}^{N} t_{i1}$$

$$= \left(\sum_{i=1}^{N} t_{i1} + t_{N2} \right) - \sum_{i=1}^{N} t_{i1}$$

$$= t_{N2}$$

For $F^1_{max} < F^2_{max}$, the corresponding lower bound on the run out delay is

$$LB_R = LB_{F_{max}} - \sum_{i=1}^{N} t_{i1} = F^2_{max} - \sum_{i=1}^{N} t_{i1}$$

$$= \left(\sum_{i=1}^{N} t_{i2} + t_{11} \right) - \sum_{i=1}^{N} t_{i1}$$

$$= \sum_{i=1}^{N} t_{i2} - \sum_{i=2}^{N} t_{i1}$$

So far, the lower bounds on the maximum flow time and the run out delay for a schedule of items in a family have been derived. When the setup times, which are incurred in setting up stages for processing each family, are introduced into the system, corresponding lower bounds may be derived. At this point, due to the existence of two different lower bounds, there exists two cases to be considered in the derivation of new lower bounds considering the setup times of families.

Case 1 :

The sum of the processing times at stage 1, of the items from the second in the sequence to the last, is greater than or equal to the sum of the processing times at stage 2 of the items from the first in the sequence to the (N-1)st in the sequence. i.e.

$$\sum_{i=2}^{N} t_{i1} \geq \sum_{i=1}^{N-1} t_{i2}$$

Case 2 :

The opposite of case 1. i.e.,

$$\sum_{i=2}^{N} t_{i1} < \sum_{i=1}^{N-1} t_{i2}$$

Case 1 :

This case may be explicitly written as

$$\sum_{i=2}^{N} t_{i1} \geq \sum_{i=1}^{N-1} t_{i2}$$

In such a case, there may exist three possible situations where:

(1) Sum of family setup time at stage 1 and processing time of the first item in sequence at stage 1 is greater than the family setup time at stage 2.

(2) Sum of family setup time and processing time of the first item at stage 1 is less than the family setup time at stage 2 and

$$\sum_{i=2}^{N} t_{i1} - \sum_{i=1}^{N-1} t_{i2} \geq \delta$$

where,

$$\delta = S_2 - (S_1 + t_{11})$$

S_1 is the setup time incurred in setting up stage 1 for a family,

S_2 is the setup time incurred in setting up stage 2 for a family,

(3) Sum of family setup time and processing time of the first item at stage 1 is less than the family setup time at stage 2 and

$$\sum_{i=2}^{N} t_{i1} - \sum_{i=1}^{N-1} t_{i2} < \delta$$

Situation 1 :

This situation may be explicitly expressed as

$$S_1 + t_{11} \leq s_2$$

and graphically represented in Figure 4.17.

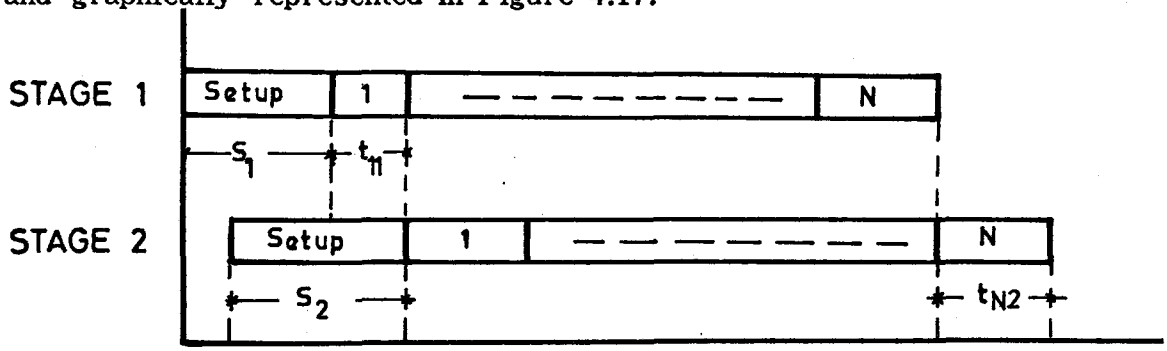


Figure 4.17. Family scheduling where setup time at stage 2 is less than the sum of processing time of the first item and setup time at stage 1. (Case 1)

Then the corresponding lower bound on the maximum flow time becomes

$$LB_{F_{max}} = \left(\sum_{i=1}^{N} t_{i1} + t_{N2} \right) + S_1$$

Situation 2 :

This situation may be explicitly expressed as follows:

$$S_1 + t_{11} < S_2 \quad \text{and} \quad \sum_{i=2}^{N} t_{i1} - \sum_{i=1}^{N-1} t_{i2} \geq \delta$$

In such a situation, since $\sum_{i=2}^{N} t_{i1} - \sum_{i=1}^{N-1} t_{i2} \geq 0$,

there may exist a run-in delay at stage 2. If the above difference is greater than δ, the lower bound on the maximum flow time becomes

$$LB_{F_{max}} = \left(\sum_{i=1}^{N} t_{i1} + t_{N2} \right) + S_1$$

The situation may be illustrated by Figure 4.18.

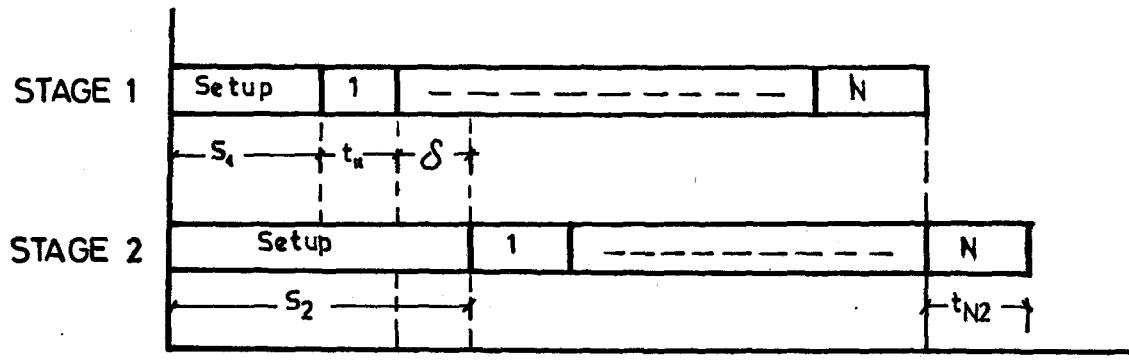


Figure 4.18. Family scheduling where setup time at stage 2 is greater than the sum of processing time of the first item and setup time at stage 1. (Case1)

Situation 3:

This situation is the opposite of the second situation. It may be explicitly expressed as follows:

$$S_1 + t_{11} < S_2 \quad \text{and} \quad \sum_{i=2}^{N} t_{i1} - \sum_{i=1}^{N-1} t_{i2} < \delta$$

At this point, since $\delta > \sum_{i=2}^{N} t_{i1} - \sum_{i=1}^{N-1} t_{i2}$, the lower bound on the maximum flow time is greater than the previous lower bounds by an amount of $\delta - \left(\sum_{i=2}^{N} t_{i1} - \sum_{i=1}^{N-1} t_{i2} \right)$ time units.

Therefore,

$$LB_{F_{max}} = [(\sum_{i=1}^{N} t_{i1} + t_{N2}) + S_1] + [\delta - (\sum_{i=2}^{N} t_{i1} - \sum_{i=1}^{N-1} t_{i2})]$$

$$= \sum_{i=1}^{N} t_{i2} + S_2$$

Case 2 :

This case may be explicitly written as

$$\sum_{i=2}^{N} t_{i1} < \sum_{i=1}^{N-1} t_{i2}$$

In such a case, there may exist two possible situations where:

(1) Family setup at stage 2 is finished before the completion of the processing the first item at stage 1.

(2) The first item in the sequence waits until the completion of setup incurred in setting up the second stage.

Situation 1 :

The first item in the sequence can be processed at stage 2 without waiting. That is, setup at stage 2 is completed before the completion of the processing the first item at stage 1. This may be explicitly expressed as

$$S_1 + t_{11} \geq S_2$$

Then, the lower bound on the maximum flow time is

$$LB_{F_{max}} = \left(\sum_{i=1}^{N} t_{i2} + t_{11} \right) + S_1$$

Situation 2:

The first item in the sequence waits until the completion of the family setup incurred in setting up the second stage. That is,

$$S_1 + t_{11} < S_2$$

Consequently, the lower bound on the maximum flow time becomes

$$LB_{F_{max}} = \sum_{i=1}^{N} t_{i2} + S_2$$

Both of the above situations may be illustrated by Figures 4.19 and 4.20, respectively.

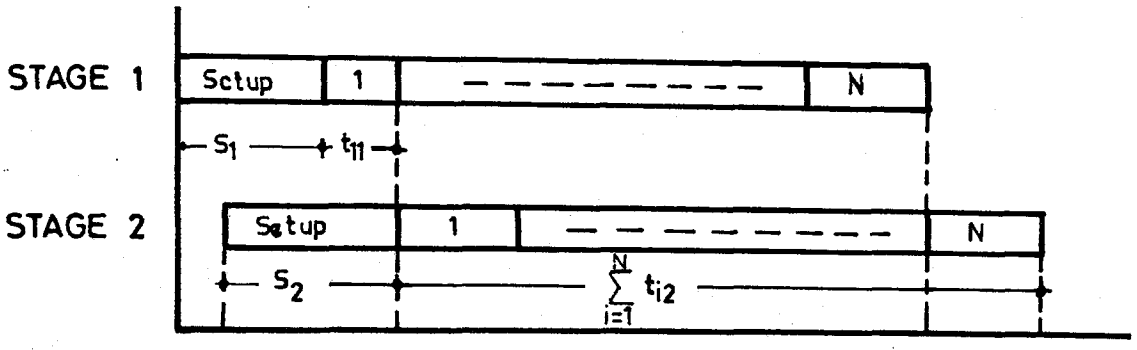


Figure 4.19. Family scheduling where setup time at stage 2 is less than the sum of processing time of the first item and setup time at stage 1. (Case 2)

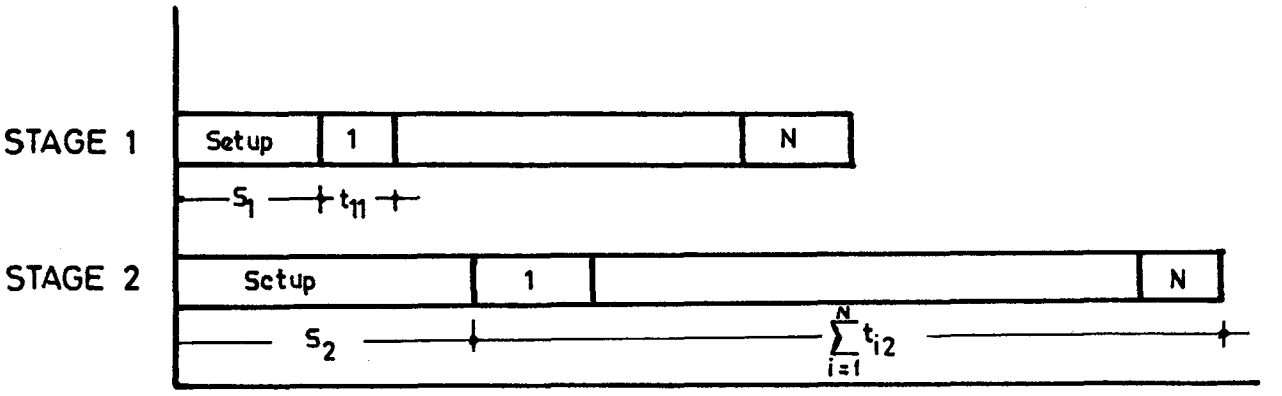


Figure 4.20. Family scheduling where setup time at stage 2 is less than the sum of processing time of the first item and setup time at stage 1. (Case 2)

Proposition 4:

If $\sum_{i=2}^{N} t_{i1} - \sum_{i=1}^{N-1} t_{i2} \geq \delta$ in any of the cases

Then $LB_{F_{max}} = \sum_{i=1}^{N} t_{i1} + t_{N2} + S_1$

Proof :

For case 1. situation.1, since $\sum_{i=2}^{N} t_{i1} \geq \sum_{i=1}^{N-1} t_{i2}$

and $\delta \leq 0$,

$$\sum_{i=2}^{N} t_{i1} - \sum_{i=1}^{N-1} t_{i2} \geq \delta$$

Therefore, $LB_{F_{max}}$ holds true.

For case 1. situation.2, it is trivial.

Proposition 5:

If $\sum_{i=2}^{N} t_{i1} - \sum_{i=1}^{N-1} t_{i2} < \delta$ in any of the cases

Then $LB_{F_{max}} = \sum_{i=1}^{N} t_{i2} + S_2$

Proof:

For case.1 situation.3, proof is trivial. For case 2, since

$$\sum_{i=2}^{N} t_{i1} - \sum_{i=1}^{N-1} t_{i2} < 0 \quad \text{and} \quad \delta \geq 0,$$

then the following expression is true :

$$\sum_{i=2}^{N} t_{i1} - \sum_{i=1}^{N-1} t_{i2} < \delta$$

Consequently, $LB_{F_{max}}$ holds true.

4. 5. 1. Lower Bounds on the Run-Out Delays:

As a consequence of the propositions 4 and 5, the following corollaries may be stated to derive the lower bounds on the run-out delays of a family.

Corollary 4:

For $\sum_{i=2}^{N} t_{i1} - \sum_{i=1}^{N-1} t_{i2} \geq \delta$,

$$LB_R = LB_{F_{max}} - (\sum_{i=1}^{N} t_{i1} + S_1)$$

$$= \left(\sum_{i=1}^{N} t_{i1} + t_{N2} + S_1 \right) - \left(\sum_{i=1}^{N} t_{i1} + S_1 \right)$$

$$= t_{N2}$$

where,

$\left(\sum_{i=1}^{N} t_{i1} + S_1 \right)$ is the completion time at stage 1

Corollary 5 :

For $\sum_{i=2}^{N} t_{i1} - \sum_{i=1}^{N-1} t_{i2} < \delta$,

$$LB_R = LB_{F_{max}} - \left(\sum_{i=1}^{N} t_{i1} + S_1 \right)$$

$$= \left(\sum_{i=1}^{N} t_{i2} + S_2 \right) - \left(\sum_{i=1}^{N} t_{i1} + S_1 \right)$$

$$= \sum_{i=1}^{N} (t_{i2} - t_{i1}) + (S_2 - S_1)$$

4. 6. Scheduling of Families :

When more than one family is to be scheduled, starting from the second family in a sequence, the processing of the succeeding families need not wait until the completion of the processing of the previous families. Therefore, any family processing can start prior to the completion of the previous family equal to the run out delay

of the preceding family. Based on this idea, families could be sequenced in nonincreasing order of their lower bounds on the run out delays.

The following corollaries can be stated as a consequence of sequencing the families in nonincreasing order of their lower bounds on the run out delays:

Corollary 6 :

For any family f with

$$\sum_{i=2}^{N} t_{i1} - \sum_{i=1}^{N-1} t_{i2} < \delta$$

$$LB_{F_{max}}(F) = C_p + LB_{F_{max}}^{f} - \min\{ LB_R , |\delta| \} \text{ if } \delta < 0$$

otherwise, $LB_{F_{max}}(F) = C_p + LB_{F_{max}}^{f}$

where,

F is the set of families containing family f and families prior to the family f,

$LB_{F_{max}}(F)$ is the lower bound on the maximum flow time time to process families in set F,

C_p is the completion time of preceding family at stage 2,

$LB_{F_{max}}^{f}$ is the lower bound on the maximum flow time of to process family f,

LB_{R_p} is the lower bound on the run out delay of the preceding family,

$|\delta|$ is the absolute value of δ.

Corollary 7:

For any family f with

$$\sum_{i=2}^{N} t_{i1} - \sum_{i=1}^{N-1} t_{i2} \geq \delta$$

$$LB_{F_{max}}(F) = C_p + LB_{F_{max}}^{f} - \Gamma \qquad \text{if} \quad \delta < 0$$

where,

$$\Gamma = \min \{ \max \{0, LB_{R_p} - |\delta|\}, \Theta \} + \min \{ LB_{R_p}, |\delta| \}$$

otherwise, $LB_{F_{max}}(F) = C_p + LB_{F_{max}}^{f} - \min \{ LB_{R_p}, \Theta \}$

Family scheduling can be illustrated by an example. Consider a scheduling problem of three families whose items and their respective total processing times at each of the two stages and, the family setup times are shown in Table 4.2.

Table 4.2. Processing times of items and family setup times

Family	Item	Processing times Stage 1	Stage 2	Family setup times Stage 1	Stage 2
1	1	5	4	4	6
	2	6	6		
	3	11	7		
	4	1	3		
	5	7	5		
	6	2	8		
2	1	4	3	8	12
	2	1	2		
	3	5	4		
	4	2	8		
	5	5	6		
3	1	4	2	10	5
	2	11	5		
	3	7	8		
	4	5	4		
	5	2	5		

Applying the Johnson's Algorithm to each of the families, the minimum makespan sequences are found as:

Family	Sequence
1	4-6-2-3-5-1
2	2-4-5-3-1
3	5-3-2-4-1

From propositions 1 and 2, and corollaries 2 and 3, the lower bound solutions for the maximum flow time and the run out delays are found. A tabular representation of the parameters and the lower bounds is shown in Table 4.3.

Table 4. 3. Parameters of example (Family Scheduling)

Family	Θ	δ	$LB_{F_{max}}$	LB_R
1	2	1	40	4
2	-4	3	35	10
3	5	-7	41	2

The Gantt charts for the sequence of items in each family are shown in Figure 4.21.

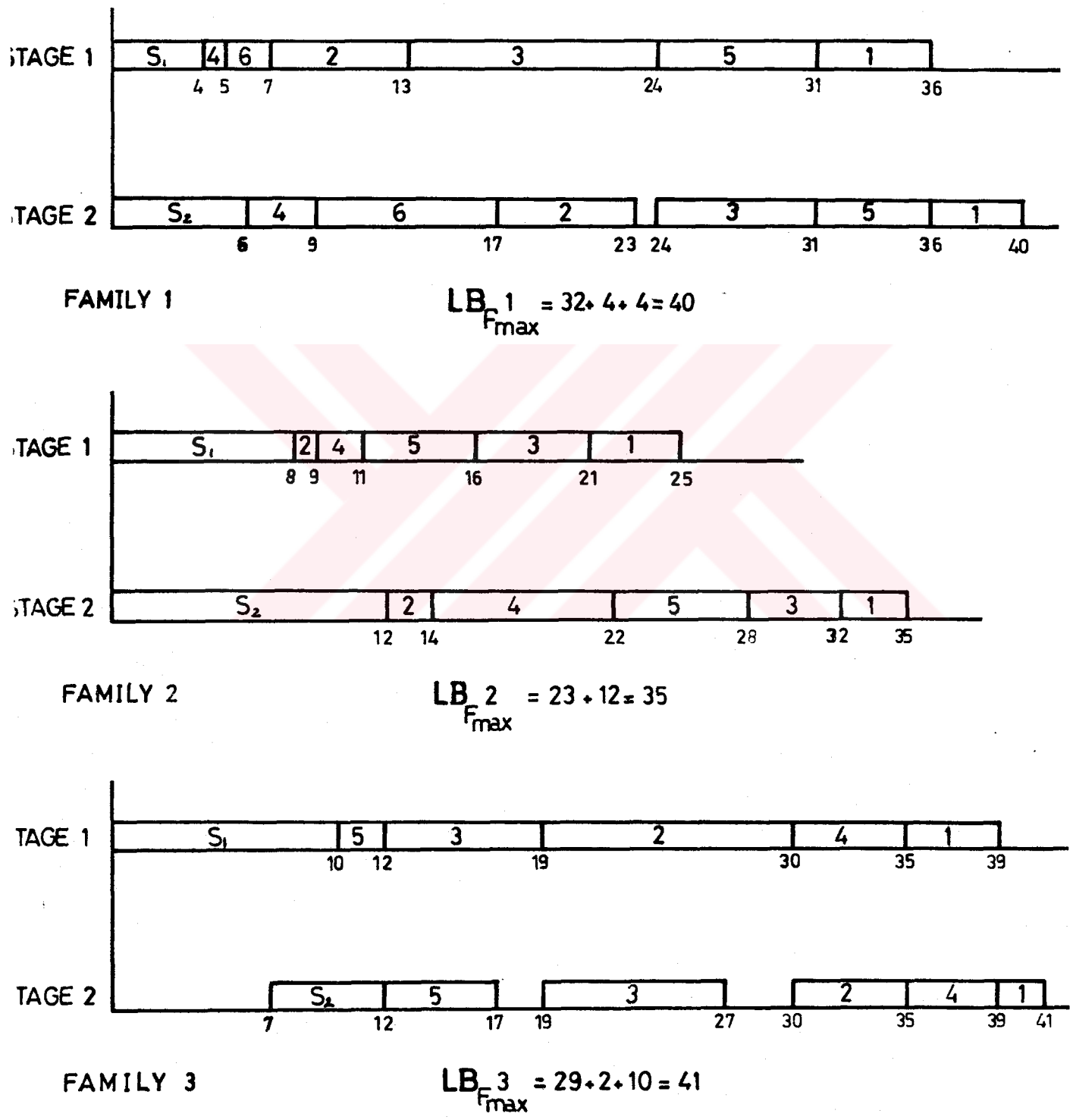


Figure 4.21. Gantt Charts of Families

Sequencing the families in nonincreasing order of their lower bounds on the run out delays leads to the sequence 2-1-3. The lower bound on the total maximum flow time for all families will be the summation of each family lower bounds on flow times. That is,

$$LB_{F_{max}}(F) = \sum_{f \in F} LB_{F_{max}}^{f}$$

where,

f is index for families,

F is the set of families.

Then,

$$LB_{F_{max}}(F) = LB_{F_{max}}^{1} + LB_{F_{max}}^{2} + LB_{F_{max}}^{3}$$

$$= 40 + 35 + 41$$

$$= 116$$

However, family processing can start prior to completion of the preceding family equal to the run out delay of the preceding family. Therefore, after processing family 2,

since $\Theta_1 = 2 > \delta_1 = 1$ and $\delta > 0$, then for $F = \{ 2,1 \}$

$$LB_{F_{max}}(F) = C_2 + LB_{F_{max}}^{1} - \min \{ LB_{R_2}, |\delta_1| \}$$

$$= 35 + 40 - \min \{ 10, |1| \}$$

$$= 75 - 1$$

$$= 74$$

$$C_1 = LB_{F_{max}}(F) = 74 \quad , \quad LB_{R_1} = C_1 - \left(\sum_{\substack{f \in F \\ i \in f}} t_{i1} + S_1 \right)$$

$$= 74 - [\ (17 + 8) + (32 + 4)\ ]$$

$$= 74 - (\ 25 + 36\ )$$

$$= 74 - 61$$

$$= 13$$

Since $\Theta_3 = 5 > \delta_3 = -7$ and $\delta < 0$, then for F = { 2,1,3 }

$$\Gamma = \min \{\ \max \{0, LB_{R_1} - |\delta_3|\}\ ,\ \Theta_3\ \} + \min \{\ LB_{R_1}\ ,\ |\delta_3|\ \}$$

$$= \min \{\ \max \{0,\ 13 - 7\ \}\ ,\ 5\ \} + \min \{\ 13, 7\ \}$$

$$= \min \{\ \max \{0,\ 6\}\ ,\ 5\ \} + 7$$

$$= \min \{\ 6,5\ \} + 7$$

$$= 5 + 7$$

$$= 12$$

$$LB_{F_{max}}(F) = C_1 + LB_{F_{max}}^{3} - \Gamma = 74 + 41 - 12$$

$$= 103$$

Therefore, sequencing families in the order 2-1-3 reduce the maximum flow time by 116 - 103 = 13 time units.

In fact, there are six possible sequences for families. These sequences and their respective lower bounds on maximum flow times are as follows:

Family sequence	Maximum flow time	
1 - 2 - 3	104	
1 - 3 - 2	112	
2 - 1 - 3	103	(*)
2 - 3 - 1	105	
3 - 1 - 2	115	
3 - 2 - 1	115	

The results shows us that sequencing families in non-increasing order of their run out delays gives the best improvement on the maximum flow time.

CHAPTER 5

ITEM DISAGGREGATION AND PROPOSED SOLUTION PROCEDURE FOR IMPROVING FEASIBILITY IN A TWO-STAGE PRODUCTION SYSTEM

This chapter is related with the item disaggregation problem and the proposed solution procedure for improving feasibility of production schedules in a two-stage system. The former section aims at disaggregating families into items as in Bitran and Hax (1977) approach. On the other hand, the later section aims at re-disaggregating families into items in case where infeasible schedules are obtained. In the last section, a complete solution procedure for improving feasibility is given.

5. 1. Item Disaggregation :

As it is stated in Chapter 2, Bitran, Haas, Hax and Meal consider the hierarchical production planning models at three levels of aggregation: type, family and item. The family production allocation is divided among the items belonging to each family by the item disaggregation model. Their proposed heuristic algorithms aim to equalize the the run out times of the items belonging to each family. The essence of these approaches is to allocate the family run quantity so as to minimize the expected time until an item in that family runs out.

Bitran and Hax (1981) have formalized the above heuristic approach by treating the run out time equalization problem as a strictly convex knapsack problem.

Problem P_f

$$\text{Minimize} \quad 1/2 \sum_{i \in I^\circ} \left[\frac{Y_f^* + \sum_{i \in I^\circ} (AI_i - SS_i)}{\sum_{i \in I^\circ} \sum_{t=1}^{L} d_{i,t}} - \frac{Z_i + AI_i - SS_i}{\sum_{t=1}^{L} d_{i,t}} \right]^2$$

subject to

$$\sum_{i \in I^\circ} Z_i = Y_f$$

$$Z_i \leq OS_i - AI_i$$

$$Z_i \geq \max \{0, \sum_{t=1}^{L} d_{it} - AI_i + SS_i\}$$

where,

Z_i is the number of units to be produced of item i,

AI_i, SS_i, OS_i are, respectively, the available inventory, the safety stock, and the overstock limit of item i.

$d_{i,t}$ is the forecasted demand for item i in period t

Y_f^* is the total amount to be allocated for all items belonging to family f (It was determined by the family disaggregation model).

L is the length of planning horizon,

I° is the set of indices of all the items belonging to family f.

The first constraint of problem P_f requires consistency in the disaggregation from family to items. The last two constraints are the upper and lower bounds for the item run quantities.

The two terms inside the square bracket of the objective function represent, respectively, the run out time for family f and run out time for an item i belonging to family f. The minimization of the square of the differences of the run out times will make those quantities as close as possible. (The term 1/2 in front of the objective function is for a computational convenience.)

The following transformation will help in simplifying the problem formulation:

$$h_i = Z_i + AI_i - SS_i$$

$$D_i = \sum_{t=1}^{L} d_{i,t}$$

$$D = \sum_{i \in I^\circ} D_i$$

$$h = Y_f^* + \sum_{i \in I^\circ} (AI_i - SS_i)$$

$$ub_i = OS_i - SS_i$$

$$lb_i = \max \{ 0, \sum_{t=1}^{L} d_{i,t} - AI_i + SS_i \} + AI_i - SS_i$$

The problem P_f is equivalent to the following problem.

Problem P_f

$$\text{Minimize} \qquad 1/2 \sum_{i \in I^\circ} \left[\frac{h}{D} - \frac{h_i}{D_i} \right]^2$$

$$\text{subject to:} \qquad \sum_{i \in I^\circ} h_i = h$$

$$lb_i \leq h_i \leq ub_i \quad , \; i \in I^\circ$$

The following algorithm can be proved to be optimal to solve problem P_f (see Bitran and Hax (1981)).

Initialization

$$h_i = \frac{D_i h}{D} \quad , \quad i \in I^\circ$$

$$I^\circ_+ = \{ i \in I^\circ : h_i \geq ub_i \} \; ; \; I^\circ_- = \{ i \in I^\circ : h_i \leq lb_i \}$$

$$\triangle^\circ = \sum_{i \in I^\circ_+} (h^\circ_i - ub_i) \; ; \; \triangledown^\circ = \sum_{i \in I^\circ_-} (lb_i - h^\circ_i)$$

Case 1. $\triangle^\circ \geq \triangledown^\circ$

It can be shown that $h_i^* = ub_i$, $i \in I^\circ_+$, together with the solution to (P1) below, will solve problem P_f:

$$\text{Minimize} \qquad 1/2 \sum_{i \in I^1} \left[\frac{h}{D} - \frac{h_i}{D_i} \right]^2 \qquad \text{(P1)}$$

subject to: $\sum_{i \in I^1} h_i = h^1$

$lb_i \leq h_i \leq ub_i \quad , \quad i \in I^1$

where,

$I' = I^\circ - I^\circ_+ \quad ; \quad lb_i' = lb_i \quad , \quad i \in I^\circ_-$

$$lb_i' = \frac{D_i h}{D} \quad , \quad i \in (I^\circ - I^\circ_+ - I^\circ_-)$$

$$h^1 = h - \sum_{i \in I^\circ_+} ub_i$$

An algorithm to solve (P1)

Let $lb_i' = lb_i \quad , \quad i \in I^\circ_-$

$lb_i' = D_i \, h \, / \, D \, , \, i \in \{I^1 - I^\circ_-\}$

and $h_i^* \, , \, i \in I^1$ be the optimal solution to (P1).

Case 2. $\nabla^\circ > \Delta^\circ$

It can be shown that $h_i^* = lb_i \, , \, i \in I^\circ_-$, together with the solution to (P2). below, will solve problem P_f:

$$\text{Minimize} \qquad 1/2 \sum_{i \in I^1} \left[\frac{h}{D} - \frac{h_i}{D_i} \right]^2 \qquad \text{(P1)}$$

subject to: $\sum_{i \in I^1} h_i = h^1$

$lb_i \leq h_i \leq ub_i' \quad , \quad i \in I^1$

where,

$$I' = I^{\circ} - I^{\circ}_{-} \quad ; \quad ub_i = ub'_i \ , \ i \in I^{\circ}_{+}$$

$$ub'_i = \frac{D_i h}{D} \ , \ i \in (I^{1} - I^{\circ}_{+})$$

$$h^{1} = h - \sum_{i \in I^{\circ}} lb_i$$

An algorithm to solve (P2)

Solve in the same way as the flow chart (P1), but use ub'_i instead of ub_i and use lb_i instead of lb'_i.

To illustrate the model by an example, consider a problem whose data is given in a tabular form as

Table 5. 1. Data for the example problem (item disaggregation)

Item i	Total Demand D_i	Lower bound lb_i	Upper bound ub_i
1	100	90	150
2	200	20	150
3	500	20	200
4	100	20	50
5	400	40	500

Total amount to be allocated among the items = 1000 units

If we solve this problem by applying the above algorithm, the optimal run quantities and run out times are as follows:

Table 5. 2. Results of item disaggregation

Item i	Run quantity h_i	Run out time h_i / D_i
1	110	1.1
2	200 (u)	1.0
3	200 (u)	0.4
4	50 (u)	0.5
5	440	1.1

where,

h / D = family run out time ≈ 0.769

(u) means that the run quantity of the item equals to its upper bound.

5. 2. Item Re-disaggregation:

As it is discussed in the previous chapter, feasibility on a production schedule may be improved by splitting large lots of items into sublots and allowing lap-phasing in the system. However, in some cases these measures may be insufficient to overcome infeasibility problem. Items with their current production run quantities can not be produced in an allocated production time. In such a case, the run quantities must be re-arranged. That is, there is a need for re-disaggregating the family run quantities among all items belonging to that family.

Since the run quantities of all items in a family is determined by allocating the family run quantity among its items, if this allocated amount is reduced, there is a possibility of getting feasible schedules. However, while doing this reduction, since family run quantity is reduced for the current production period, the number of setups for the family increases. Therefore, decrease in family run quantity is performed by decreasing the run quantities of the items which requires more production time in the system. If the item whose current run quantity is decreased for re-disaggregation is chosen such that

$$\text{Arg} \quad \underset{i}{\text{Min}} \{ [(F_{max} - T) / u_i] / D_i \}$$

where,

i is the index for item in the family,

F_{max} is the maximum flow time of the schedule before re-disaggregation,

T is the allocated production time (available capacity),

D_i is the total demand for item i,

u_i is the sum of unit processing times at both stages.

the reduced amount in the run quantity of the item is very small with respect to its total demand. That is, the item run-out time is reduced by such a decrease in run quantity.

In the following procedure, a solution algorithm based on the above idea is given.

Procedure (Item Re-disaggregation):

STEP 1 : Compute the excessive production time by subtracting the available production time from the maximum flow time of the family. i.e.,

$$\alpha = F^{f}_{max} - T_f$$

where,

α is the excessive production time,

F^{f}_{max} is the maximum flow time of family f,

T_f is the available production time allocated for processing family f.

STEP 2 : For each item in family f,

(i) Compute the sum of unit processing times at both stages. i.e.,

$$u_i = p_{i1} + p_{i2} \qquad i = 1, \ldots, N_f$$

where, N_f is the number of items in family f.

(ii) Compute the total processing time for all items in family f as

$$\Omega = \sum_{i=1}^{N^f} h_i \cdot u_i + \sum_{i=1}^{N^f} (s_{i1} + s_{i2})$$

STEP 3 : Find the proportion of the maximum flow time to the total processing time of all items in family f as

$$p = F^{f}_{max} \; / \; \Omega$$

STEP 4 : (i) Compute the number of units of item which can be processed in the excessive production time considering the total unit processing time of the items. i.e.

$$\beta_i = \alpha \; / \; u_i \qquad i = 1, \ldots, N_f$$

(ii) List all items in increasing order of

$$\{ \beta_i \; / \; D_i \} \qquad i = 1, \ldots, N_f$$

(iii) Consider the first item in the list. ie., $r = 1$

Let m be the item number in the r th position in the list.

STEP 5 : Find the required decrease in run quantity of item f as

$$\beta_m \; . \; (1 \; / \; p)$$

where,

$\beta_m \; . \; (1 \; / \; p)$ is the decrease in the run out quantity of item m, consequently decrease in total amount of units to be allocated for items in family f.

STEP 6 : Compute the maximum decrease in run out quantity of item m.

$$\Delta = h_m - lb_m$$

STEP 7 : If $\Delta \geq (\beta_m \cdot 1 / p)$, then

(i) Solve problem P_f for $h = h - [\beta_m \cdot (1 / p)]$ and $ub_m = h_m - [\beta_i \cdot (1 / p)]$

(ii) Find the lower bound on the maximum flow time of all items having modified run quantities.

(iii) If $LB_F{}^f_{max} > T_f$, then

- Update the excessive production time as

$$\alpha = LB_F{}^f_{max} - T_f$$

- Update β_m value

- Goto step 5.

Otherwise,

- Find the maximum flow time which gives a feasible schedule for family f.

- Stop.

Otherwise,

(i) Find the modified upper bound level for item m as

$$ub_m = h_m - \Delta$$

(ii Solve problem P_f for $h = h - \Delta$

(ii) Update the excessive production time as

$$\alpha = \alpha - (u_m \cdot p \cdot h_m)$$

(iii) - Set $r = r + 1$

- Update β_i values for $i = r, \ldots, N_f$

(v) Goto step 5.

To illustrate the item re-disaggregation procedure by an example, consider a problem whose data is given in a tabular form as

Table 5. 3. Data for the example problem (item re-disaggregation)

Item	Total Demand	Lower Bound	Upper Bound	Setup time / Processing time Stage 1	Stage 2	Minimum batch size
i	D_i	lb_i	ub_i	s_{i1} / p_{i1}	s_{i2} / p_{i2}	m_i
1	100	20	100	20 / 5	30 / 7	10
2	200	40	150	50 / 8	50 / 5	20
3	500	50	300	30 / 10	20 / 10	25
4	100	10	100	20 / 2	40 / 8	10
5	400	50	400	20 / 8	20 / 5	40
6	300	30	200	10 / 3	20 / 9	10

S_1 = family setup time at stage 1 = 200

S_2 = family setup time at stage 2 = 400

h = 1000 units T =7800

(Assume that available inventory and safety stock are set to zero level)

If we solve this problem by applying the item disaggregation algorithm, the optimal run quantities and run-out times are as follows:

Table 5. 4. Results of item re-disaggregation

Item i	Run quantity h_i	Run out time h_i / D_i
1	64	0.64
2	127	0.635
3	300 (u)	0.6
4	64	0.64
5	225	0.6375
6	191	0.637

where,

h / D = family run out time ≈ 0.625

After finding the sequence for items as 4-1-6-3-5-2 , maximum flow time is found as

$$F_{max} = 8169$$

Since the available production time is not sufficient to process all items in that family, items should be re-disaggregated, The steps of the item re-disaggregation procedure will be as follows:

S. 1 : The excessive production time = $\alpha = F_{max} - T$

$= 8169 - 7800$

$= 369$

S. 2 : (i)

Item i	Total unit processing time $u_i = p_{i1} + p_{i2}$
1	12
2	13
3	20
4	10
5	13
6	12

(ii) $\Omega = 14666 + 330 = 14996$

S. 3 : $p = F_{max} / \Omega = 8169 / 14996 \approx 0.544$

S. 4 :

Item i	β_i	β_i / D_i
1	369 / 12 = 30.75	0.3075
2	369 / 13 ≈ 28.38	0.1419
3	369 / 20 = 18.45	0.0360
4	369 / 10 = 36.9	0.3690
5	369 / 13 ≈ 28.38	0.0710
6	369 / 12 = 30.75	0.1025

Item List = { 3,5,6,2,1,4 }

m = 1

S. 5 : $\beta_m . (1/p) = 18.45 (1/0.544) \approx 34$ units

S. 6 : $\Delta = h_3 - lb_3 = 300 - 50 = 250$ units

S. 7 : Since $\Delta = 250 > 34$, then

item disaggregation for $h = 1000 - 34 = 966$ units

and $ub_3 = h_3 - 34 = 300 - 34 = 266$ units results with

Item (i)	1	2	3	4	5	6
Run quantity (h_i)	64	127	266	64	255	191
Run out time (h_i/D_i)	.64	.635	.532	.64	.6375	.637

Considering the total processing times of each item at both stages for the modified run quantities, the processing is unchanged. The lower bound on the maximum flow time is

$$LB_{F_{max}} = S_2 + \sum_{i=1}^{6} (s_{i2} + p_{i2} \cdot h_i) = 400 + 7429 = 7829$$

Since $LB_{F_{max}} = 7829 > T = 7800$, excessive production time is

$$\alpha = LB_{F_{max}} - T = 7829 - 7800 = 29$$

The updated value for β_3 is

$$\beta_3 = \alpha \ / \ u_i = 29 \ / \ 20 = 1.45$$

S. 5 : $\beta_m \cdot (1/p) = 1.45 \ (1/0.544) \approx 3$ units

S. 6 : $\Delta = h_3 - lb_3 = 266 - 50 = 216$ units

S. 7 : Since $\Delta = 216 > 3$, then

item disaggregation for $h = 966 - 3 = 963$ units

and $ub_3 = h_3 - 34 = 266 - 3 = 263$ units results with

Item (i)	1	2	3	4	5	6
Run quantity (h_i)	64	127	263	64	255	191
Run out time (h_i/D_i)	.64	.635	.526	.64	.6375	.637

$$LB_{F_{max}} = 400 + 7399 = 7799$$

Since $LB_{F_{max}} = 7799 < T = 7800$, then feasible schedule with sequence { 4,1,6,3,5,2 } is given in Table 5.5.

Table 5. 5. Results of item re-disaggregation and scheduling

Item	Minimum batch size	Sublot or batch size
4	10	10
		54
1	10	64
6	10	95
		96
3	25	89
		87
		87
5	40	85
		85
		85
2	20	24
		51
		32
		20

$F_{max} = 7799$

5. 3. Proposed Solution Procedure for Feasibility Improvement:

After the discussion of feasibility improvement by item re-disaggregation, we now present the complete procedure for

feasibility improvement in a two-stage flow shop in an hierarchical framework. The procedure considers scheduling of families in a given type (set of families) and items in families simultaeneously.

STEP 1 : For each family,

1.1. Apply item disaggregation algorithm to allocate family run quantity among its items.

1.2. Compute the total processing times of all items.

1.3. Decompose the items into two sets that contain the items having shorter total processing times at stage 1 and at stage 2, respectively.

1.4. Apply Johnson's Algorithm to find a processing sequence of items in each of two sets of items.

1.5. Compute the lower bounds on the maximum flow time and run-out delay including family setup times for permutation schedule that contains all items in that family.

1.6. Check for the existence of a feasible schedule for all items in that family that should be processed in an allocated production time for that family. That is,

If $LB_F{}^f_{max} < T_f$ then,

(i) Goto Step 1.1. for the next family.

Otherwise,

(i) Put the family into a list that contains families which can not be

processed in their allocated production time (i.e., capacity is insufficient).

(ii) Goto Step 1.1. for next family.

STEP 2 : Compute total production time required for processing all families by summing the lower bounds on the maximum flow time for each family.

STEP 3 : Check whether all families can be processed in an allocated production time for all families. That is,

$$\text{If } \sum_{f \in F} LB_F{}^f{}_{max} > \sum_{f \in F} T_f \text{ , then}$$

(i) Order all families in nonincreasing value of their lower bounds on the run-out delays.

(ii) Find the lower bound on the maximum flow time for processing all families where family processing can start prior to the completion of the preceding family.

(iii) If lower bound found above on the maximum flow time is still greater than the production time allocated for all families, then apply item re-disaggregation procedure to form feasible schedules.

Otherwise, Goto Step 4.

STEP 4 : Apply reverse scheduling and lap-phasing procedures.

STEP 5 : Compute total production time required for processing all families.

STEP 6 : Check whether all families can be processed in an allocated production time. That is,

$$\text{If } \sum_{f \in F} F^{f}_{max} > \sum_{f \in F} T_{f} \text{ , then}$$

(i) Order all families in nonincreasing value of their run-out delays.

(ii) Find the maximum flow time for processing all families where family processing can start prior to the completion of the preceding family.

(iii) If the maximum flow time is still greater than the production time alllocated for all families, then apply item re-disaggregation procedure to form feasible schedules.

Otherwise,

The current schedules for all families are feasible.

CHAPTER 6

SAMPLE PROBLEM

In this chapter, we present a sample problem to illustrate the scheduling feasibility problem in a two-stage flow shop and the solution procedure for improving feasibility in hierarchical production planning.

Consider a manufacturing environment where 25 different items are processed at two manufacturing stages following the same flow pattern through these stages. They are first processed at stage one and then at stage two.

These different items are grouped into four families where all items in a family share a common setup. A tabular representation of data for this example problem is provided in table 6.1, and table 6.2.

Table 6.1. Family Setup Times and Run Quantities

Family	Setup Times		Run Quantity	Available
	Stage 1	Stage 2		Production Time
f	S_1	S_2	h	T_f
1	200	400	500	5000
2	350	200	500	4500
3	500	500	800	6000
4	100	500	400	4000

Table 6.2. Total Demand in a Planning Period, Bounds on the Run Quantities, Minimum Production Batch Sizes of Items

Family	Item	Total Demand	Lower Bound	Upper Bound	Setup Time / Processing Time Stage 1	Setup Time / Processing Time Stage 2	Minimum Batch Size
f	i	D_i	lb_i	ub_i	s_{i1} / p_{i1}	s_{i2} / p_{i2}	m_i
1	1	100	20	80	20 / 5	30 / 8	20
	2	80	15	60	50 / 8	50 / 5	10
	3	200	40	200	30 / 9	20 / 10	20
	4	150	30	130	20 / 7	40 / 8	15
	5	75	15	75	10 / 8	20 / 5	5
	6	100	20	50	20 / 10	25 / 12	20
	7	200	40	100	30 / 8	30 / 8	10
2	1	200	20	200	20 / 10	20 / 2	20
	2	140	20	70	30 / 4	25 / 10	10
	3	80	15	80	10 / 5	10 / 3	10
	4	120	40	100	40 / 2	40 / 10	25
	5	50	30	50	15 / 8	10 / 7	15
	6	150	25	100	20 / 14	35 / 10	10
3	1	160	40	100	30 / 4	30 / 8	20
	2	500	25	300	30 / 8	40 / 7	15
	3	200	20	200	40 / 5	20 / 2	10
	4	100	25	50	20 / 9	20 / 3	10
	5	120	20	120	50 / 10	35 / 5	15
	6	110	35	80	40 / 12	35 / 14	30
	7	60	20	60	10 / 9	20 / 6	20
	8	140	30	110	20 / 4	40 / 2	30
4	1	100	20	100	40 / 5	40 / 3	20
	2	200	45	200	45 / 7	30 / 5	30
	3	100	50	100	20 / 9	20 / 11	20
	4	125	25	100	35 / 10	30 / 16	15

The following are the steps of the procedure applied to the example problem:

STEP 1 : For family 1

1.1. If the item disaggregation algorithm is applied for disaggregating the run quanity of family 1, the following steps are followed:

S.1 : The set of items ,I°, contains the items whose run quantities are not determined yet. This set is

$I^\circ = \{ 1,2,3,4,5,6,7 \}$

Initial run quantities are computed from $h_i^\circ = D_i . h / D$ and given in table 6.3.

Table 6.3. Initial Run Quantities of Items in Family 1

i	1	2	3	4	5	6	7
h_i°	55.2	44.2	110.5	82.9	41.4	55.2	110.5

$I_+^\circ = \{ 6,7 \}$ $I_-^\circ = \phi$

Since set I_-° is empty, items 6 and 7 will be processed at their upper bounds. That is, $h_6^* = 50$ units and $h_7^* = 100$ units

S.2. Since case 2 of the algorithm occurs,

$I^1 = I^\circ - I_+^\circ = \{1,2,3,4,5,6,7\} - \{6,7\}$

$= \{1,2,3,4,5\}$

The updated lower bounds for the items in set I^1 are given in table 6.4.

Table 6.4. Updated Lower Bounds on Run Quantities of Items

i	1	2	3	4	5
lb_i	55.2	44.2	110.5	82.9	41.4

The updated family run quantity is

$$h^1 = h - (h_6^* + h_7^*) = 500 - (50 + 100)$$
$$= 350$$

S.1. Applying the same algorithm to the items whose run quantities are not determined yet, the following run quantities are obtained as in the following table (table 6.5.)

Table 6.5. Run Quantities of Items for All Items in Family 1

Item i	Run Quantity h_i^*	Run out Time h_i^* / D_i
1	58	.58
2	46	.575
3	116	.58
4	87	.58
5	43	.573
6	50	.5
7	100	.5

$$h / D = 500 / 905 \approx .55$$

Since $I^\circ = \emptyset$. then the current run quantities for items are optimal.

1.2. The total processing times for items are given in table 6.6.

Table 6.6. Total Processing Times of Items in Family 1.

Item	Total Processing Times Stage 1	Stage 2
i	$t_{i1} = s_{i1} + p_{i1} \cdot h_i^*$	$t_{i2} = s_{i2} + p_{i2} \cdot h_i^*$
1	310	494
2	418	280
3	1074	1180
4	629	736
5	354	235
6	520	625
7	830	830
	4135	4380

1.3. Two sets of items are:

$S^1 = \{ 1,4,6,7 \}$ $S^2 = \{ 2,3,5 \}$

1.4. Processing sequences for sets are respectively:

1 - 6 - 4 - 7 and 3 - 2 - 5

1.5. Processing time of first sublot of first item at stage 1 and processing time of last sublot of last item at stage 2 in the sequence are:

$$s_{11} + p_{11} \cdot Q_{11} = 20 + 5\,(20) = 120$$

and

$$p_{72} \cdot m_7 = \begin{bmatrix} \text{processing time} \\ \text{of item 5} \end{bmatrix} \times \begin{bmatrix} \text{minimum} \\ \text{batch size} \\ \text{of item 5} \end{bmatrix}$$

$$= 5\,(5) = 25$$

where, $Q_{11} = m_1 = 20$ units

1.6. By taking the first sublot of the first item in sequence as the first job and the last sublot of the last item i sequence as the last job to be processed, we create two dummy items

which are processed first and last in the schedule, respectively. Therefore, the number of jobs to be processed increases from 7 to 9. The lower bound on the maximum flow time and run out delay for family 1 is computed as

$$\sum_{i=2}^{9} t_{i1} = (310-120) + 418 + 1074 + 629 + 354 + 520 + 830$$

$$= 4015$$

$$\sum_{i=1}^{8} t_{i2} = 494 + 280 + 1180 + 736 + 235 + 625 + + (830-25)$$

$$= 4355$$

$$\delta = S_2 - (S_1 + t_{11}) = 400 - (200 + 120)$$

$$= 80$$

Since $\sum_{i=2}^{9} t_{i1} - \sum_{i=1}^{8} t_{i2} = -340 < \delta = 80$, then

$$LB_{F^1_{max}} = \sum_{i=1}^{9} t_{i2} + S_2 = 4380 + 400 = 4780$$

$$LB_R = \sum_{i=1}^{9} (t_{i2} - t_{i1}) + (S_2 - S_1)$$

$$= (4380 - 4135) + (400 - 200)$$

$$= 445$$

1.7. Since $LB_{F^1_{max}} = 4780 < T_1 = 5000$, the current run quantities of the items may be feasible.

For families 2, 3, and 4, the computational results in step 1 are provided in appendix (see table I.1, I.2, I.3, I.4, and I.5.)

STEP 2 : Total production time required for processing all families is

$$LB_{F}{}^{1}_{max} + LB_{F}{}^{2}_{max} + LB_{F}{}^{3}_{max} + LB_{F}{}^{4}_{max} = 4780 + 4432 + 6607 + 3964$$

$$= 19783$$

Total allocated production time for processing all families is

$$\sum_{f \in F} T_f = 5000 + 4500 + 6000 + 4000 = 19500$$

STEP 3 : Since $\sum_{f \in F} LB_{F}{}^{f}_{max} = 19783 > \sum_{f \in F} T_f = 19500$

(i) List of families in nonincreasing order with respect to their lower bounds on the run out delays is given in table 6.7.

Table 6.7. Family Sequence With Respect to Lower Bounds on the Run Out Delays

Sequence No	1	2	3	4
Family	4	1	2	3

(ii) After processing family 4,

Since $\Theta_1 = -340 < \delta_1 = 80$ and $\delta_1 > 0$

$$LB_{F}{}^{}_{max}(F) = C_4 + LB_{F}{}^{1}_{max}$$

$$= 3964 + 4780$$

$$= 8744$$

The lower bound on the run out delay after processing family 1 is updated as

$LB_{R_1} = LB_{F_{max}}(F)$ - Completion time at stage 1

$= 8744 - [\ (100+3218) + (200+4135)\ ]$

$= 8744 - 7653$

$= 1091$

$F = \{\ 4,1,2\ \}$

Since $\Theta_2 = 505 > \delta_2 = -240$ and $\delta_2 < 0$

$\Gamma = \min\{\max\{0, LB_{R_1} - |\delta_2|\}, \Theta_2\} +$

$\min\{LB_{R_1}, |\delta_2|\}$

$= \min\{\max\{0, 1091 - |-240|\}, 976\} +$

$\min\{1091, |-240|\}$

$= 745$

Then the lower bound on the maximum flow time for set of family $F \in \{\ 4,1,2\ \}$ is updated as

$LB_{F_{max}}(F) = C_1 + LB_{F_{max}}^2 - \Gamma = 8744 + 4432 - 745$

$= 12431$

The lower bound on the run out delay after processing family 2 is updated as

LB_{R_2} = $LB_{F_{max}}(F)$ - Completion time at stage 1

= 12431 - [7653 + (350+4052)]

= 12431 - 12055

= 376

F = { 4,1,2,3 }

Since $\Theta_3 = 976 > \delta_3 = -110$ and $\delta_2 < 0$

$\Gamma = \min \{\max\{0, LB_{R_2} - |\delta_3|\}, \Theta_3\} +$

$\min \{LB_{R_2}, |\delta_3|\}$

$= \min \{\max \{0, 376 - |-110|\}, 976\} +$

$\min\{376, |-110|\}$

= 376

Then the lower bound on the maximum flow time for set of family F $\in$ { 4,1,2,3 } is updated as

$LB_{F_{max}}(F) = C_2 + LB_{F_{max}}^{3} - \Gamma = 12431 + 6607 - 376$

= 18662

(iii) Since $LB_{F_{max}}(F) = 18662 < \sum_{f \in F} T_f = 19500$

Goto step 4.

STEP 4 : The batch sizes for all items in all families are provided in appendix (see table I.6, I.7, I.8, and and I.9)

Note: while applying lap phasing and reverse scheduling, family setup times are not considered.

STEP 5 : Now, add setup times to the maximum flow times of each family. The maximum flow times are found as

Family	:	1	2	3	4
Maximum Flow Time	:	4780	4432	6607	3964

Step 6 : Schedule of families gives a maximum flow time which equals to the updated lower bound on the maximum flow time. That is, makespan is 18662.

The Gantt charts for all all items in families and family schedule are provided in appendix (see figure I.1, I.2, I.3, I.4 and I.5)

Since the current solution is feasible, then the procedure is terminated.

Without applying lap phasing, maximum flow times of all families are as follows:

Family	:	1	2	3	4
Maximum Flow Time	:	5258	4589	7384	4397

Schedule of families gives a maximum flow time which equals to 21628 time units.

Then the percent decrease in maximum flow times of families are:

Family	:	1	2	3	4
Percent decrease in makespan	:	3.0	3.5	10.5	9.8

The total reduction in maximum flow time for processing all families becomes 13.7 per cent.

CHAPTER 7

CONCLUSION

The hierarchical approach is one of two distinct approaches for production planning and scheduling problems. Partitioning of the overall planning problem and the linkage of the resulting problems is the basic philosophy of a hierarchical planning approach. Formal planning in hierarchical framework simultaneously considers the needs of all planning levels and provides an overall solution that disaggregates as well. However, there are some problems arising in the hierarchical formulations such as the consistency of decisions at all levels of hierarchy and the feasibility of both aggregate and detailed plans.

The purpose of this study is to develop and evaluate an iterative solution procedure for improving feasibility in a two stage flow shop environment in Hax-Meal's hierarchical framework.

Lot splitting and lap-phasing is taken as the basic procedure for feasibility improvement. The nature of the two stage flow shop system is exploited by decomposing items into two sets. It was observed that the application of lap-phasing to the two sets of jobs decomposed causes independent analysis of two sets. The symmetrical analysis (reversing the time scale) and the interchange in the processing times at stages simplify the solution obtained by lap-phasing.

A further improvement in the feasibility of the plans is achieved by sequencing families in the nonincreasing order of the corresponding lower bounds on their run out delays.

In cases where the improvements obtained by lap-phasing and sequencing families according to their lower bounds on the run out delay are not sufficient a solution procedure for disaggregating the items is proposed. The basic idea of this procedure is based on the systematic reduction of item run quantities. The items whose run quantities are reduced are chosen as the items which requires more production time in the system and the reduction is very small with respect to items total demand.

The hierarchical planning procedures proposed are suitable for implementation since the problems are sequentially solved and at each decision levels feedback from lower levels to the upper levels is provided.

The proposed solution procedures are applied to a sample problem for a manufacturing environment where 25 different items are grouped into into 4 families. A significant reduction, which is approximately 14 per cent, in maximum flow time for processing all families is obtained.

Extensions of the current study might concentrate on the development of lap-phasing procedures for flow shops having more than two stages. The nature of two stage flow shop system makes it easy to decompose the jobs(items) into two sets and the sequencing

problem can be solved by Johnson's algorithm optimally. Since for flow shops having more than two stages, the sequence of jobs and the job splitting become more complex problem.

Another further study may be with respect to how types and families are defined. That is, what is the best way to aggregate items into families and families into types for achieving scheduling feasibility. The effects of aggregation over product is a potential area of reseach.

It is important to consider the number and the length of subperiods of a medium term planning model. The effects of aggregation over time may also be carried out as a research area.

Finally, it was observed that for re-disaggregation, there is a need for a reduction in run-out quantities of items to achieve a feasible schedule. Since the reduction of run-out quantities increases the number of production cycles hence setups. A further study on the trade off between scheduling feasibility and the work-in-process inventory, safety stocks and the setup times (or costs) may lead to interesting findings.

REFERENCES

Baker, K.R., *Introduction to Sequencing and Scheduling*, John Wiley and Sons Inc., New York, 1974.

Bitran, G.R., and A.C. Hax, "On the Design of Hierarchical Production Planning Systems," *Decision Sciences*, Vol.8, No.1, pp. 28-55, 1977.

Bitran, G.R., E.A. Haas, and A.C. Hax, "Hierarchical Production Planning : A Single Stage System," *Operations Research*, Vol.29, No.4, pp.717-743, 1981.

Bitran, G.R., E.A. Haas, and A.C. Hax, "Hierarchical Production Planning : A Two-Stage System," *Operations Research*, Vol.29, No.2, pp.232-251, 1982.

Bitran, G.R., and A.C. Hax, "Disaggregation and Resource Allocation Using Convex Knapsack Problems with Bounded Variables," *Management Science*, Vol.27, No.4, pp. 431-441, April 1981.

Buffa,E.S., and W.H. Taubert, *Production-Inventory Systems: Planning and Control*, Irwin, Homewood, Ill., 1972.

Dzielinski, B.P., Baker, C.T., and Manne, A.S., "Simulation Tests of Lot size Programming," *Management Science*, Vol.9, No.2, pp. 229-258,1963.

Dzielinski, B.P., and Gomory, R.E., "Optimal Programming of Lot Sizes, Inentory and Labor Allocations," *Management Science*, Vol. 11, No.9, pp. 874-890, 1965.

French, S., *Sequencing and Scheduling : An Introduction to the Mathematics of the Job Shop*, John Wiley ans Sons Inc., New York, 1982.

Goodman, G.J., "Optimal Aggregation of Multi-Item Production Smoothing Models," *Management Science*, Vol.24, No.16, pp.1733-1739, December 1978.

Graves, Stephan C., "Using Lagrangean Techniques to Solve Hierarchical Production Planning Problems," *Management Science*, Vol.28, No.3, pp. 260-275, March 1982.

Hax, A.C., and H.C. Meal, "Hierarchical Integration of Production Planning and Scheduling," M.A. Geisler (ed.), *Studies in Management Sciences*, Vol.1 Logistics, North-Holland, Amsterdam, pp. 53-69, 1975.

Hax, A.C., and D. Candea, *Production and Inventory Management*, Prentice-Hall, Inc., Eaglewood Cliffs,New Jersey, 1984.

Jaikumar, R., "An Operational Optimization Procedure for Production Scheduling," *Computers and Operations Research*, Vol.1, No.2, pp. 191-200, Aug, 1974.

Johnson, S.M., "Optimal Two and Three Stage Production Schedules with Setup Time Included," *Nav. Res. Log. Quar.*, Vol.1, pp.61-68, 1954.

Karmarkar, U.S., "Equalization of Runout Times," *Operations Research*, Vol.29, No. 4, pp. 757-768, July/Aug. 1981.

Lasdon, L.S., and R.C. Terjung, "An Efficient Algorithm for Multi-Item Scheduling," *Operations Research*, Vol.19, No.4, pp.946-969, July-August 1971.

Manne, A.S., "Programming to Economic Lot Sizes," *Management Science*, Vol.4, No.2, pp. 115-135, 1958.

Mesarovic, M.D., Macko, D., Takahora, Y., *Theory of Hierarchical Multi-Level Systems*, Academic Press, New York, 1970.

Newson, E.F.P., "Multi-Item Lot Size Scheduling by Heuristic Part I : With Fixed Resources, Part II : With Variable Resources," *Management Science*, Vol.21, No.10, pp. 1186-1203, June 1975.

Winters, P.R., "Constrained Inventory Rules for Production Smoothing," *Management Science*, Vol.8., No.4, pp. 470-481, 1962.

APPENDIX

Table I.1. Run Quantities and Run Out Times of Items in Families.

Family f	Family Run out Time h / D	Item i	Run Quantity h_i	Run Out Time h_i / D_i
1	.55	1	58	.58
		2	46	.575
		3	116	.58
		4	87	.58
		5	43	.573
		6	50 (u)	.5
		7	100 (u)	.5
2	.676	1	147	.735
		2	70 (u)	.5
		3	59	.7375
		4	88	.73
		5	37	.74
		6	100 (u)	.67
3	.576	1	93	.58125
		2	291	.582
		3	116	.58
		4	50 (u)	.5
		5	70	.583
		6	64	.582
		7	35	.583
		8	81	.579
4	.761	1	76	.76
		2	152	.76
		3	76	.76
		4	95	.76

Table I.2. Total Processing Times of Items in Families

Family f	Item i	Total Processing Times Stage 1 t_{i1}	Stage 2 t_{i2}
1	1	310	494
	2	418	280
	3	1074	1180
	4	629	736
	5	354	235
	6	520	625
	7	830	830
		4135	4380
2	1	1490	314
	2	310	725
	3	305	187
	4	216	920
	5	311	269
	6	1420	1035
		4052	3450
3	1	402	774
	2	2358	2077
	3	620	252
	4	470	170
	5	750	385
	6	808	931
	7	325	230
	8	344	202
		6077	5021
4	1	420	268
	2	1109	790
	3	704	856
	4	985	1550
		3218	3464

Table I.3. Processing Sequence of Items in Families.

Family	Processing Sequence Set 1	Set 2
2	{ 4 - 2 }	{ 6 - 1 - 5 - 3 }
3	{ 1 - 6 }	{ 2 - 5 - 3 - 7 - 8 - 4 }
4	{ 3 - 4 }	{ 2 - 1 }

Table I.4. Processing Times of the First and Last Jobs in the Sequence at Stage 1 and 2, respectively.

Family	Processing Time of the Initial Sublot of First Item in Sequence	Processing Time of the Last Sublot of Last Item in Sequence
1	20 + (5)(20) = 120	(5)(5) = 25
2	30 + (2)(25) = 90	(3)(10) = 30
3	30 + (4)(20) = 110	(3)(10) = 30
4	20 + (9)(20) = 200	(3)(20) = 60

Table I.5. Lower Bounds on the Maximum Flow Time and Run Out Delay for Families.

Family	Number of Items N_f	$\sum_{i=2}^{N_f+2} t_{i1}$	$\sum_{i=1}^{N_f+1} t_{i2}$	δ	$LB_{F_{max}}^{f}$	LB_R^f
1	7	4015	4355	80	4780	445
2	6	3962	3420	-240	4432	30
3	8	5967	4991	-110	6607	30
4	4	3018	3404	200	3964	646

Table I.6. Schedule for Family 1

Item	Sequence No	Sublot or Batch Size	Total Processing Times Stage 1	Stage 2
1	1	20	120	190
		30	190	304
2	6	18	194	140
		28	224	140
3	5	54	516	560
		62	558	620
4	3	55	405	480
		32	224	256
5	7	8	74	60
		5	40	25
		5	40	25
		5	40	25
		5	40	25
		5	40	25
		5	40	25
		5	40	25
6	2	28	300	361
		22	220	264
7	4	60	510	510
		40	320	320

Table I.7. Schedule for Family 2

Item	Sequence No	Sublot or Batch Size	Total Processing Times Stage 1	Stage 2
1	4	52	540	124
		95	950	190
2	2	70	310	725
3	6	19	105	67
		10	50	30
		10	50	30
		10	50	30
		10	50	30
4	1	25	90	290
		63	126	630
5	5	16	143	122
		21	168	147
6	3	100	1420	1035

Table I.8. Schedule for Family 3

Item	Sequence No	Sublot or Batch Size	Total Processing Times Stage 1	Stage 2
1	1	20	110	190
		47	188	376
		26	104	208
2	3	106	878	782
		105	1480	1295
3	5	116	620	252
4	8	10	110	50
		30	270	90
		10	90	30
5	4	70	750	385
6	2	32	424	483
		32	384	448
7	6	35	325	230
8	7	81	344	202

Table I.9. Schedule for Family 4

Item	Sequence No	Sublot or Batch Size	Total Processing Times Stage 1	Stage 2
1	4	23	155	109
		33	165	99
		20	100	60
2	3	51	402	285
		59	413	295
		42	294	210
3	1	20	200	240
		26	234	286
		30	270	330
4	2	31	345	526
		31	310	496
		33	330	528

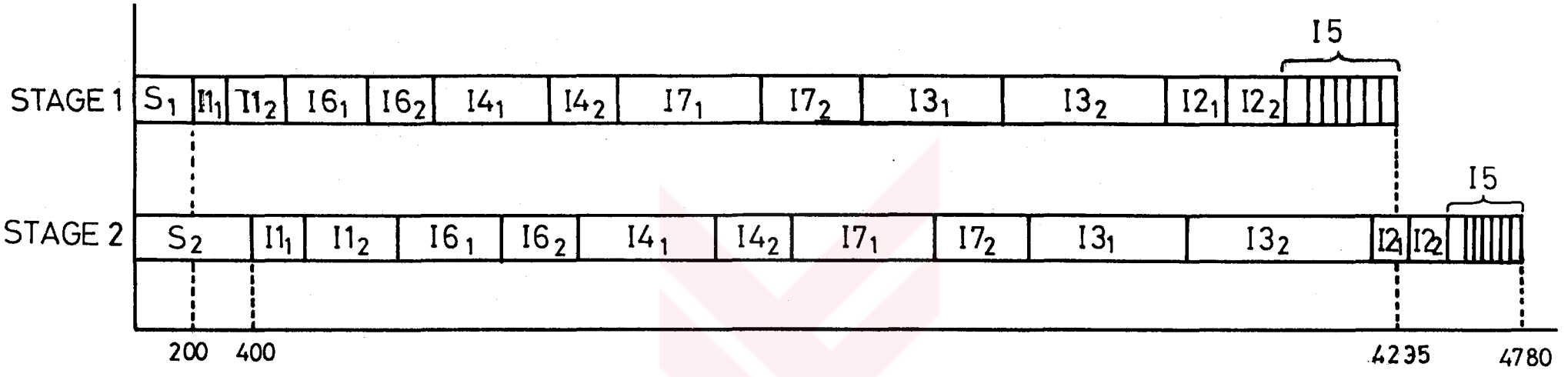


Figure I.1. Gantt chart for Family 1

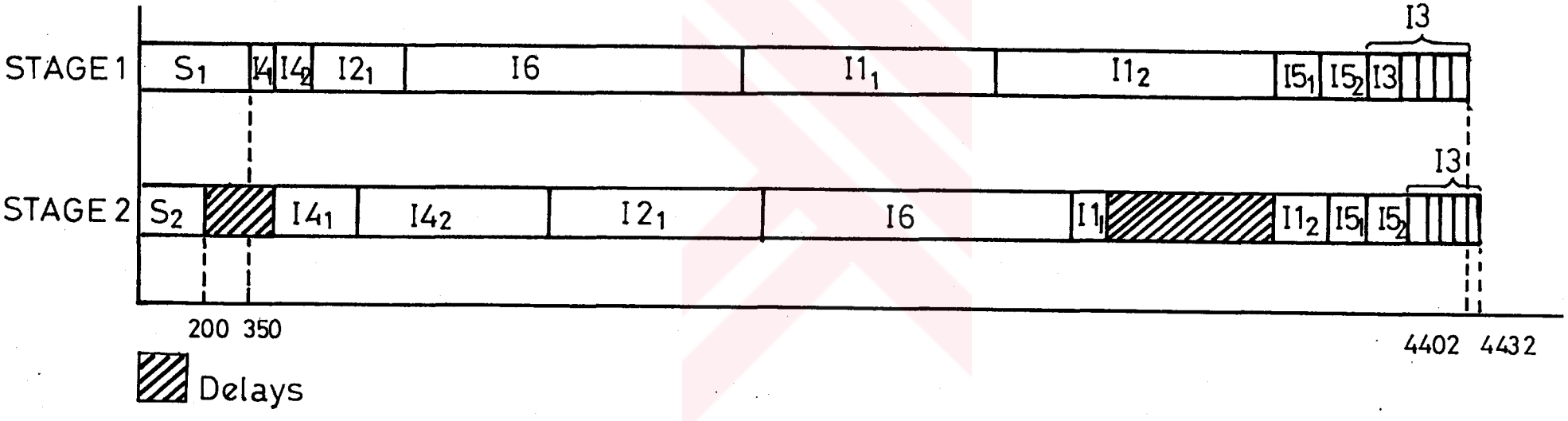


Figure I.2. Gantt chart for Family 2

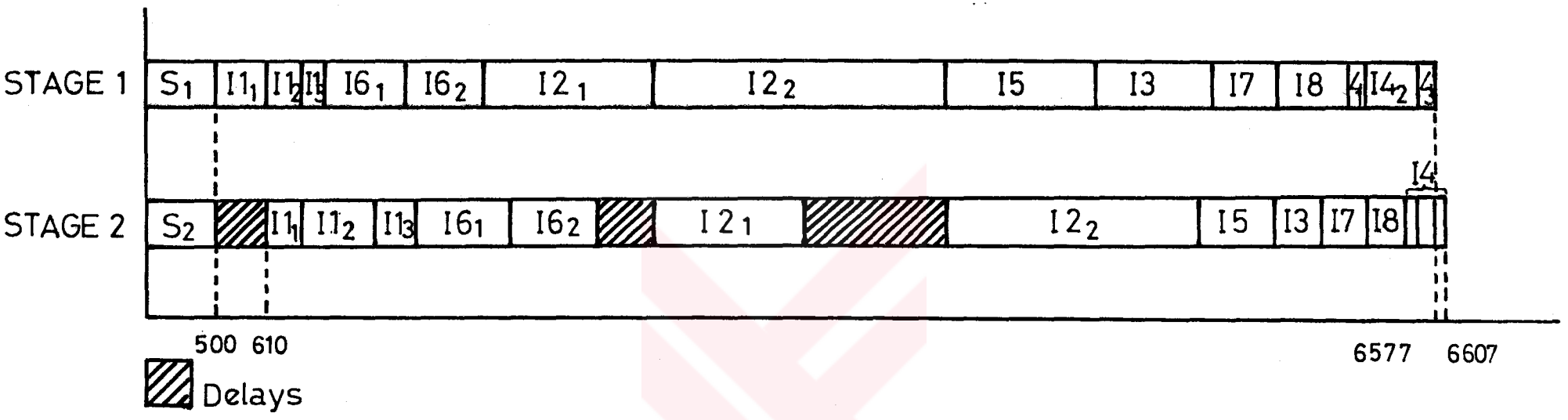


Figure I.3. Gantt chart for Family 3

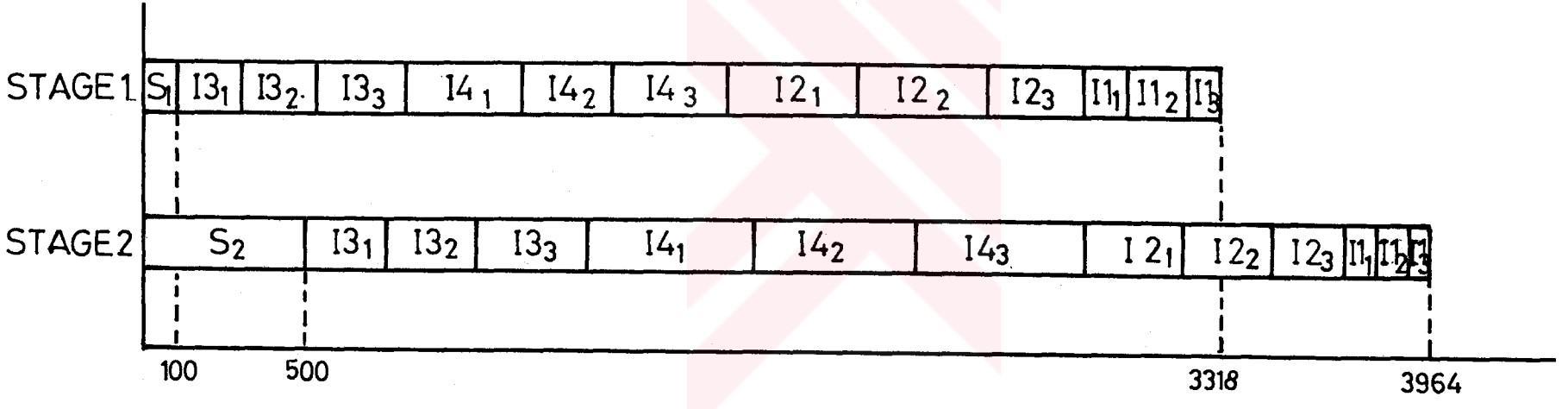


Figure I.4. Gantt chart for Family 4

Figure I.5. Schedule of families 1, 2, 3, and 4