LOT SCHEDULING WITH TRANSFER BATCHES IN MULTI-STAGE SHOPS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

FERDA CAN ÇETİNKAYA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN
THE DEPARTMENT OF INDUSTRIAL ENGINEERING

JULY 1996

Approval of the Graduate School of Natural and Applied Sciences.

Prof. Dr. Tayfur Öztürk
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Doctor of Philosophy.

Prof. Dr. Ömer Kırca
Head of Department

This is to 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 Doctor of Philosophy.

Assoc. Prof. Dr. M. Sinan Kayalıgil
Supervisor

Examining Committee Members

Assist. Prof. Dr. M. Selim Aktürk

Assist. Prof. Dr. H. Cemal Akyel

Prof. Dr. F. Canan Çilingir

Assoc. Prof. Dr. M. Sinan Kayalıgil

Prof. Dr. Ömer Kırca

ABSTRACT

LOT SCHEDULING WITH TRANSFER BATCHES IN MULTI-STAGE SHOPS

Çetinkaya, Ferda Can
Ph.D., Department of Industrial Engineering
Supervisor: Assoc. Prof. Dr. M. Sinan Kayalıgil

July 1996, 191 pages

The *production scheduling* problem arises whenever it is necessary to process a set of N jobs on M machines (stages) so that the result is optimal with respect to a given measure of performance. In the traditional scheduling theory of multi-stage discrete parts manufacturing systems, several simplifying assumptions are made; specifically, it is assumed that transfer of items from one stage to another is not allowed until the entire job (lot) is completed, although a job consists of a number of identical items which form a production lot. However, the operations of a job can be overlapped by moving small transfer batches (some portion of the lot) through the system, while the rest of the items of the same job are processed at the current stage. If the sizes of these transfer batches are restricted to remain the same on all machines, the solution is called a *consistent* transfer batch solution. On the other hand, if the sizes of the transfer batches are allowed to vary, the solution will be referred to as a *variable* transfer batch size solution. Also, there are two major cases that are conceptually worthy to distinguish. The first case which is referred to as *lot-streaming* involves overlapping operations for a given job. In the second case which is referred to as *lot-splitting* the movement of transfer batches of a lot is again permitted, but the transfer of batches of the lot can not be intermingled with transfer batches of other lots while they are processed.

If there is a single job to be scheduled by allowing transfer batches, the goal is to find the optimal size of the transfer batches which will minimize given performance measure (e.g., makespan, mean flow time). However, in multi-job problems the size

of the transfer batches of all jobs as well as their movement time between machines, and the sequence of jobs or transfer batches on the machines must be determined simultaneously.

In this thesis, the main concern is to provide polynomial time solution procedures which give the optimal solution to different versions of the single and multi-job scheduling problems with transfer batches in flow shops and open shops. Specifically, we study the single-job streaming problem in which makespan is taken as the measure of schedule performance in a two-stage hybrid flow shop, in a general M-machine flow shop with variable sized transfer batches, in a three-machine flow shop with variable transfer batches and detached setup times on the machines. Also, we study the single-job streaming problem in which mean flow time is taken as the measure of schedule performance in a general M-machine flow shop. Furthermore, the multi-job streaming problems in two-stage flow shop and open shops with the makespan criterion are analyzed. In addition, we consider the lot-splitting problem in a two-machine flow shop in which either transfer time between machines exist or not. The computational complexity of the solution procedures for these problems are also given.

Keywords: Scheduling, lot streaming, lot splitting, transfer batches

ÖZ

ÇOK AŞAMALI ATÖLYELERDE AKTARMA BÖLÜMLÜ KAFİLE ÇIZELGELEMESİ

Çetinkaya, Ferda Can
Doktora, Endüstri Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. M. Sinan Kayalıgil

Temmuz 1996, 191 sayfa

Üretim çizelgeleme problemi, N tane işi M tane makiada (aşamada) işlemek gerektiği zaman doğar ki elde edilen sonuç da belli bir başarım ölçütüne göre optimaldir. Çok aşamalı kesikli parça üretim sistemlerinin geleneksel çizelgeleme teorisinde, birçok basitleştirici yapılır; özellikle, bir iş, bir üretim kafilesini oluşturan özdeş birkaç parçadan meydana gelmesine karşın, parçaların bir aşamadan diğer bir aşamaya transferine bütün iş (kafile) tamamlanana kadar izin verilmediği varsayımı yapılır. Halbuki, aynı işin geri kalan parçaları o anki aşamada işlem görürken, işin operasyonları, küçük transfer bölümlerinin (kafilenin bir kısmı) sistem içinde hareket ettirilmesiyle üstüste bindirilebilir. Eğer bu transfer bölümlerinin hacmi, bütun aşamalarda aynı kalmaya kısıtlanırsa, çözüm, *uyumlu* transfer bölümlü çözüm olarak anılır. Öte yandan, transfer bölümlerinin hacimlerinin değişmesine izin verilirse çözüm, *değişken* hacimli transfer bölümlü çözüm olarak anılır. Ayrıca, ayırdedilmesi kavramsal olarak değecek iki önemli durum vardır. *Kafile aktarma* olarak anılan birinci durum belli bir iş için operasyonların üstüste bindirilmesini içerir. *Kafile bölme* olarak anılan ikinci durumda ise kafilenin transfer bölümlerinin hareketine yine izin verilir, fakat kafilenin bölumlerinin transferi, işlenme esnasında diğer kafilelerin transfer bölümleriyle karıştırılmaz.

Eğer transfer bölümlerine izin verilerek çizelgelenmesi gereken tek bir iş varsa, amaç, işin bitiş süresi, ortalama akış süresi gibi belli bir başarım ölçütünü enazlıyacak transfer bölümlerinin optimal hacmini bulmaktır. Her nasılsa, çok işli problemlerde, işlerin makinalar arasındaki hareket zamanı ile tüm işlerin transfer bölümlerinin

hacmine ve işlerin ya da transfer bölümlerinin makinalardaki sırasına eş zamanlı olarak karar verilmesi gerekir.

Bu tezde esas ilgi, akış tipi atölye ve açık tip atölyelerde, transfer bölümlü tek ve çok işli çizelgeleme problemlerinin değişik versiyonlarına optimal çözüm veren polinom-zamanlı çözüm yöntemleri sağlamaktır. Özellikle, değişken transfer bölümlü ve makinalarda ayrılabilir hazırlık (kurma) zamanlı üç makinalı akış tipi atölyede, değişken hacimdeki transfer bölümlü genel M-makinalı akış tipi atölyede, ve iki aşamalı hibrid akış tipi atölyede, işin bitiş süresinin çizelge başarım ölçütü olarak alındığı tek işli aktarma problemini çalışmaktayız. Ayrıca, genel M-makinalı akış tipi atölyelerde ortalama akış süresinin çizelge başarım ölçütü olarak alındığı tek işli aktarma problemini de çalışmaktayız. Bunlardan başka, tüm işlerin bitiş süresi kriterli iki aşamalı akış tipi atölye ve açık tip atölyelerde çok işli aktarma problemleri analiz edilmektedir. Ayrıca, makinalar arasında transfer zamanının olduğu ve olmadığı durumlarda iki makinalı akış tipi atölyedeki kafile bölme problemini ele almaktayız. Bütün bu problem için önerilen çözüm yöntemlerinin işlemsel karmaşıklığıda sunulmaktadır.

Anahtar kelimeler: Çizelgeleme, kafile aktarma, kafile bölme, transfer bölümleri

To my family

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor Sinan Kayalıgil for his encouragement, guidance and valuable advise in all phases of this study. I am also indebted to the members of the examining committee for accepting to read and review this thesis.

This thesis has been partially written during my visit to Purdue University for eleven months. I would like to thank the Scientific and Technical Research Council of Turkey (TÜBİTAK) for their partial financial support in the scope of NATO Science Fellowship (NATO-A2).

It was a pleasure to meet J.N.D. Gupta who visited Purdue in the same time I did and who conducted joint research with me in Ball State University. His comments on some papers of mine, later incorporated in the thesis, are gratefully acknowledged. I consider our permanent interest in scheduling problems with transfer batches to be a good basis for cooperation.

Moral support from my parents, fiancé Dilek, sister Şeyda, and friends whose names do not appear here have contributed to bring happiness into those hard years and made my life easier.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

A manufacturing organization can be considered as a system which transforms raw materials into finished goods through the use of its manufacturing resources. In many manufacturing (production) environments, the goal of manufacturers is to produce and ship quality products to the customer in a cost-effective and timely manner. To achieve this goal, *production planning and scheduling* activities in the manufacturing firms play an important role. The distinction between planning and scheduling activities is made primarily on the basis of the length of the time period that is considered and degree of detail that is generated. The longer range planning includes the demand forecasts and the general activities that are needed to secure the manufacturing resources required for production. Scheduling covers a shorter period in great detail by assigning specific tasks at the departmental and shop levels. Products and their quantities to be produced in the specified periods are determined through production planning. Then an actual implementation plan as to the time schedule for performing required activities must be established.

Within the framework of the hierarchical approach for production planning and scheduling in multi-stage manufacturing systems, much research has focused on determining the optimal lot size (production quantity) for products in the traditional lot-sizing models basically considring the trade-off between the cost of setting up a machine and the cost of holding inventory of products. On the other hand, in the lower level of the decision making process, the traditional machine scheduling and sequencing models has focused on allocation of the resources to the production activities given the lot size for the products. Most of these models, however, do not consider the possibility of lot sizes at intermediate processing steps being less than the immediate production quantity of the products.

In traditional production scheduling theory, production lots (jobs) are assumed to be indivisible single entities although an entire lot might consist of many identical items. Furthermore, it is implicitly assumed that partial transfer of completed items in

a lot between the work stations on the processing routing of the lot is impossible. But it is quit unreasonable to wait for the entire lot to finish its processing on the current work station, while the downstream work stages may be idle. It should be obvious that processing the entire lot as a single object can lead to large work-in-process inventories between work stations and throughput time (the total elapsed time to complete the manufacturing activities of the lot).

However, permitting partials (which will be called as *transfer batches*) of the entire lot to be transferred to its downstream work station will allow operations overlapping while work proceeds to complete the lot at the upstream work station. The essence of this idea is the use of transfer batches between work stations to increase the movement of work in the manufacturing environment. Clearly, the potential efficiencies come in the form of reduced throughput time, reduced work-in-process inventory, reduced floor space requirements, and reduced size of transfer vehicles. However, additional costs may accrue through additional costs of transportation of transfer batches and through additional cost of control.

To motivate the discussion consider a lot of 100 items to be processed in a three-stage manufacturing environment in which the flow of its operations is unidirectional from stage 1 through stage 3. Assume that the unit processing time at stage 1, 2 and 3 are 1, 3, and 2 minutes. If we do not allow transfer batches, the throughput time is 100(1+3+2)=600 minutes (see Figure 1.1.a). However, if we specify two equal sized transfer batches through all stages, the throughput time decreases to 450 minutes, a reduction of 25% (see Figure 1.1.b). It is clear that the throughput time decreases as the number of transfer batches increases, but the there will be an additional cost of transportation of the second transfer batch.

Over the years different approaches or philosophies have been advanced as methods of production planning and scheduling such as *Manufacturing Resource Planning* (MRP-formerly Material Requirements Planning), *Just-in-time* (JIT) *Production*, and *Optimized Production Technology* (OPT).

MRP systems are perhaps the most widely installed in industry today. For a fixed planning horizon, MRP systems determine the quantities of each item that will be used in the production of a prescribed volume of end products and the times at which each of these items must be purchased or manufactured to meet prescribed due dates for the end products. MRP models the flow of material by assuming that items flow from work station to work station in the same batches that are used in production. However, in MRP systems no thought is given to permitting the use of transfer

batches. On the other hand, the transfer batch concept is likely to be practiced in real settings, and significant improvements can be made through overlapping to the existing MRP framework (Vollman, 1986; Graves and Kostreva, 1986).

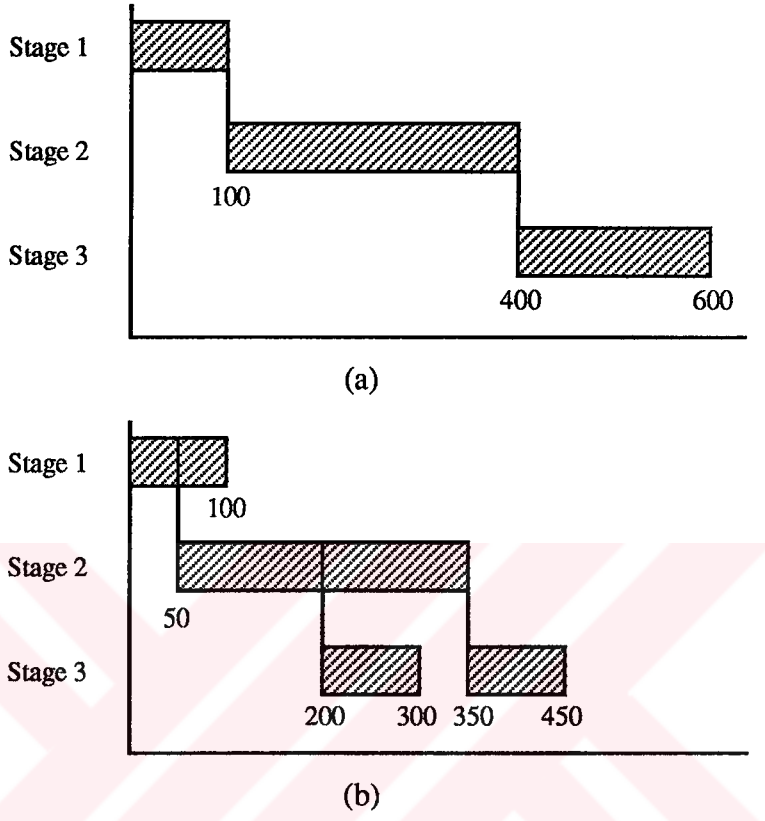


Figure 1.1. Schedule (a) without transfer batches, (b) with transfer batches

JIT production is another production planning and scheduling philosophy that dictates reduced materials inventories and minimum work-in-process (WIP) inventories to aid process improvement and reduce process variability. The obvious benefit of JIT schedules is the reduced capital cost associated with holding inventories both in terms of reduced inventory storage requirements and reduced investment in raw materials and intermediate goods on hand. The more subtle benefits resides in associated improvements in process flow and floor control, particularly with respect to the early detection of rejects and immediate isolation of the associated culprit operation. As a scheduling philosophy, the success of JIT production demands highly reliable suppliers, workforce and repair facilities, since buffer stocks are essentially eliminated. While JIT is commonly misclassified as a "method" that achieves minimal WIP with a lot size of one, in fact, there is no prescriptive theory or method for deriving JIT schedules or achieving JIT goals. However, in many environments,

splitting a production lot of a product into many unit sized sublots (transfer batches) may not be feasible or may be prohibitive because of transportation costs between work stations (Kulonda, 1984; Umble and Srikanth, 1990; Fogarty *et. al.*, 1991). In this context, there may be a need to find the transfer batch sizes larger than unity which can be implemented in the JIT environment.

Optimized Production Technology (OPT) is a proprietary software package that has generated considerable interest as a production scheduling tool since it is established in 1979. OPT can be viewed as an alternative to a comprehensive MRP system for production planning, materials planning and resource planning. While the details of the finite scheduling portion of OPT have been disclosed, promotional literature and examination of OPT schedules stresses the use of transfer batches increases the utilization of the bottleneck machines, which in turn implies the reduction in the throughput time (Jacob, 1984; Lundgrian, 1986). OPT is discussed in more detail in Section 2.2.

In each of the contexts mentioned above, the possibility of using transfer batches creates a host of new problems involving production scheduling. The transfer batch scenario presents the scheduling theory community with a plethora of challenging new theoretical problems. As transfer batches are used routinely in batch manufacturing situations, a number of applied issues arise. A purpose of this thesis is to introduce some ideas that can improve the flow of production by allowing transfer batches through some multi-stage manufacturing environments and investigate some machine scheduling problems with transfer batches as well as their solution methods under various assumptions of production.

1.1 Definitions and Notations in Production Scheduling Problems

A general production scheduling problem may be stated as follows.

Suppose we are given a set $J = \{1,...,N\}$ of N job-lots containing a number of identical items and each job $j \in J$ have k *operations* denoted by $O_{j,1}, O_{j,2}, ..., O_{j,k}$. Furthermore, there is a given set $M = \{M_1,...,M_M\}$ of resources called *machines* which are allocated to perform the operations of jobs in set J. Each job $j \in J$ consists of Q_j identical items (*lot size* of job j), and has $p_{j,m}$ units of *processing time* per item on a machine $m \in M$. A usual convention, which we shall follow

throughout, says that each machine can process at most one job at a time. In this context, production scheduling can be defined as the allocation of available production resources over time to best satisfy some set of criteria. In other words, a scheduling problem is to determine an assignment of tasks over time onto production resources, and a sequence in which is these resources wil perform the tasks.

Numerous schemes have been proposed for categorizing production scheduling problems. Here, we present the following five dimensions for classifying production scheduling problems (Graves, 1981):

1. Production requirements generation,
2. Nature of the requirement specification,
3. Scheduling environment,
4. Technological constraints (or processing complexity)
5. Scheduling criteria.

Production Requirements Generation

The first dimension, production requirements, is a key distinction in production scheduling problems. Production requirements may be generated either directly by customers' orders or indirectly by inventory replenishment decisions. This distinction is often made in terms of an *open shop* (*make-to-order*) versus a *closed shop*. In an open shop all production orders are by customer request and no inventory is stocked; in a closed shop all customer requests are serviced from inventory, and production tasks are generally a result of inventory replenishment decisions.

Nature of the Requirement Specification

Depending upon how the production requirements are generated, the specification of the requirements may be termed as *deterministic* or *stochastic*. For instance, for an open shop, the processing time for each operation of the task may be known, or may be a random variable with specified probability distribution. Similarly, for a closed shop, the customer demand process, which derives the inventory replenishment decisions, may be assumed to be stochastic or deterministic.

Scheduling Environment

The scheduling environment deals with the assumptions on the availability of information on future production requirements. A common distinction is made

between *static* and *dynamic* environments. In a static environment, the scheduling problem is defined with respect to a finite set of fully specified requirements; no additional production requirements will be added to this set, nor will any of the specifications be altered. Conversely, in a dynamic environment, the scheduling problem is defined not only for the known requirements, but also with respect to the anticipation for additional requirements and specifications generated over the future time periods.

Technological Constraints

In defining a scheduling problem the technological constraints on jobs must be specified. Technological constraints are determined principally by the flow pattern of jobs on machines. In this context, the problems studied in scheduling theory can be divided into two large classes. The first class contains the *single-stage* problems, i.e., the problems with either a *single* machine or a number of *parallel* machines. Generally, in these problems, it is assumed that there is only one operation for each job in the job set. There exists three types of parallel machines each being distinguished by their speed factor. If all machines in set M have equal job processing speeds, then we call them *identical* parallel machines. These machines form the simplest class of all parallel machine problems. If the machines differ from each other by a constant speed factor which is independent of the job being processed, then they are called *uniform* parallel machines. If the speeds of the machines depend on the particular job, then they are called *unrelated* parallel machines and the problems associated with these machine environments are perceived to be harder than those with uniform parallel machines. For the problems belonging to that class, it is assumed that the processing of each job consists of only one stage. In other words, it is allowed to use any machine for processing a job and, in principle, a job can be completed whenever it is processed on any one of these machines.

The second class includes the *multi-stage* or *shop* models. Here, it is assumed that every stage consists of only one machine, and the processing of each job requires more than one stage. Normally, all machines are involved in the processing of a job, however, there may exists jobs which have to be processed on only a subset of the machines of the given set M.

The most well-known among the multi-shop models is a *flow shop*. In a flow shop there are M operations of each job in the set J, and each job has the same process *routing* $R=\{M_1,...,M_M\}$. This means that each job has to be processed first on

machine 1, then on machine 2 and so on, until it is processed on the last machine M_M. The first remarkable result on the flow shop was obtained by Johnson (1954).

A more general situation is known as a *job shop*. For this model, it is assumed that each job has a process routing which is given in advance, however the routes are not the same for all jobs as in the case for the flow shop. As a matter of fact, each job may even have a unique route. Another point of the difference between the job and the flow shop is that, for the former model, a route for a particular job may not include all the machines and, on the other hand, some machines may occur in the route more than once since the number of operations of a job may be more than the number of machines in the shop. The paper of Jackson (1956) seems to be the first significant one devoted to this model.

Many practical situations lead to the necessity of studying so-called *open shops* introduced by Gonzalez and Sahni (1976). Here, each job has to be processed on each machine (unless the corresponding value of the processing time is zero), however, the process route is not given in advance for each job but must be chosen, different jobs being allowed to get different routes.

Those three basic models can be generalized by combining some or all of them. For example, if one combines flow and open shops, one gets the model which is known as a *mixed shop*. More precisely, for the mixed shop it is assumed that the set J of jobs is partitioned into two non empty subsets J_1 and J_2. The jobs of the set J_1 have non-fixed process routes (as in open shop) while the jobs of the set J_2 have the fixed process route $R = \{M_1,...,M_M\}$ (as in a flow shop). Using the term "mixed shop" we follow Masuda, Ishi and Nishida (1985).

The more general multi-stage scheduling model which covers all the previous ones was introduced by Strusevich (1989). This model deserves to be called a *super shop*. It is obtained as a result of combining the open and the job shop. Accordingly to that model, the set J of jobs is known to be partitioned into k subsets $J_1, J_2, \ldots, J_k$, $k \geq 3$. For the jobs of the set J_1 the process routes are not given in advance (as in open shops) while the jobs belonging to a particular set J_s, $2 \leq s \leq k$, have the same fixed route specified by a sequence which may be different for each subset; some machines of set M may not occur in these sequences, whilst on the other hand, some of them may occur more than once (as in job shops).

In some practical situations, it is also possible to combine the single and multi-stage environments to represent the most general machining environment. For example, if

someone considers the flow shop in which parallel machines at each stage are available, the model which is known as a *hybrid flow shop* is obtained. Salvador (1973) first recognized this model in the polymer, chemical and petrochemical industries where there are several identical plants each of which can be considered as a flow shop, and the jobs can practically be processed at any of the plants, at any stage of the processing.

Scheduling Criteria

The last dimension, scheduling criteria, indicates the measures upon which schedules are to be evaluated. Two broad classes of criteria are *schedule cost* and *schedule performance*. The cost associated with a particular schedule includes the fixed costs associated with production setups, variable production costs, inventory holding costs, and shortage costs for not meeting deadlines or stocking out, etc. On the other hand, the performance of schedule may be measured in many ways, the percentage of late tasks, the average or maximum tardiness for a set of tasks, and average or maximum flow time for a set of tasks.

In most production environments, the schedule evaluation is based on a mixture of both cost and performance criteria; however, most of the theoretical literature on production scheduling addresses single criteria problems.

Traditionally, in scheduling theory, the schedule performance criteria are penalty functions which depend on some performance measures. For a schedule of jobs one can define the *completion time* of a job j (which is the primary measure since all other measures are determined from it) as the time at which the processing of the last operation of the job is completed. Let $C_{j,i,m}$ be the completion time of the ith transfer batch of job j on machine m, then the completion of job j within the context of scheduling problems with transfer batches is defined as $C_j = \max_{i,m} C_{j,i,m}$.

Let r_j be the time at which job j is ready for processing. Then the *flow time* of job j is defined by

$$F_j = C_j - r_j$$

which is the amount of time job j arrival into and departure from the system. Clearly, the flow time is closely related to the work-in-process (WIP) cost.

If we define d_j as the due date by which it is desirable to complete job j, then the *lateness* of job j is expressed as:

$$L_j = C_j - d_j$$

and measures the conformity of the schedule to a given due date.

The *tardiness* of job j, a positive amount of lateness, is expressed as:

$$T_j = \max\{L_j, 0\}$$

and reflects the fact that, in many situations, distinct penalties and other costs will be associated with positive lateness, but no penalties or benefits will be associated with negative lateness.

The earliness of job j, a negative amount of lateness, is expressed as:

$$E_j = \max\{-L_j, 0\}$$

and reflects the fact that, in some situations, there may be a penalty for finishing job before it is due.

Now, we need to present some of performance criteria objectives.

Makespan : The *makespan* (maximum flow time), which is defined as the time length from beginning of the first operation of the first job to the ending the last operation of the last job, is:

$$F_{\max} = \max_{1 \le j \le N} F_j.$$

If the ready times of all jobs are assumed to be zero, we denote $F_j = C_j$, $j = 1,...,N$.

The schedule with the smallest makespan is often taken as a surrogate for the schedule with the highest utilization. The intuitive idea is that finishing the given set of activities earlier will allow new activities to be started earlier.

Mean (weighted) Flow Time : The *mean flow time*, which is the mean time length during which all jobs remain in the shop, is given by:

$$\overline{F} = \frac{1}{N} \sum_{j=1}^{N} F_j.$$

Let w_j be the weight which is used to indicate the relative importance of job j, then the *weighted mean flow time* is:

$$\overline{F}_w = \frac{1}{N} \sum_{j=1}^{N} w_j F_j.$$

In some circumstances, costs might be related to the number of jobs completed, thus provide incentive for minimizing mean (weighted) flow time, since these costs are directly related to average inventory. In practice, reduced flow time would

probably also provide a competitive sales advantage that would be even more significant.

Maximum Lateness, Tardiness : The *maximum lateness* and *tardiness* are, respectively given by:

$$L_{\max} = \max_{1 \le j \le N} L_j,$$

$$T_{\max} = \max_{1 \le j \le N} T_j.$$

Minimizing maximum tardiness is important when customers tolerate smaller tardiness but become rapidly and progressively more upset for larger ones. Minimizing maximum lateness is primarily important because it is a relatively easy problem and can be used as an aid for solving other problems. For example, minimizing maximum flow time can be accomplished by setting due dates equal to the arrival times and minimizing maximum lateness. Minimizing maximum tardiness can similarly be accomplished by minimizing maximum lateness and truncating to zero if that value is negative. Finally, there is a technical use for the maximum lateness objective in solving complex multi-resource problems with a makespan objective. If we set an arbitrary common due date for all jobs in the problem, then minimizing maximum lateness will minimize the makespan.

An extension of the above single performance criterion objective functions to multi-criteria objectives can be devised in two ways:

Composite objective : $\alpha_1 z_1 + \alpha_2 z_2$, where $\alpha_1, \alpha_2 > 0$ are weights and z_1, z_2 are objective functions.

Primary and secondary objectives : minimize z_1 subject to $z_2 = z_2^*$, where z_2 is the primary objective with z_2^* being its minimum value and z_1 is the secondary objective.

1.2 Basic Concepts in Scheduling Problems with Transfer Batches

Generally, scheduling problems with transfer batches require assumptions about the formation and movement of transfers. The former assumption is related with the size of transfer batches. With this assumption, at one extreme, all transfer batches are unit sized, i.e., the number of transfer batches is equivalent to the number of identical items in the lot (process batch). For instance, a rip saw may produce lumber of different lengths with setups required for different widths; boards are moved to the

next work station one by one as they come off the saw. On the other hand, the size of the sublots are restricted to be larger than unit sized (equivalently, if the number of transfer batches is restricted to be less than the number of identical items in the lot) due to the technological restrictions for moving items between machines. In this case, if each item in a transfer batch is available for processing on the next machine or to be delivered from the shop as soon as it completes processing on the current machine, then we have *item availability* (item flow) assumption on the movement of transfer batches. On the other hand, under *batch availability* (batch flow) assumption, each item in a transfer batch already processed on a machine should wait the completion of all items in the same transfer batch to be processed by the next machine or to be delivered from the shop. Note that item and batch availability assumptions are equivalent in the case of equal sized transfer batches.

As we will discuss in further detail in Chapter 2, formal descriptions of scheduling models with transfer batches have been introduced in several articles, and different types of models exist. Basically, these variations reflect specific constraints imposed on scheduling decisions. These constraints may reflect features of the production technology that must be taken as given, or they may reflect simplifying assumptions that are made for analytic convenience. Some of the constraints which alter the scheduling problem are *constant* versus *variable* number of transfer batches between machines, *consistent* versus *variable* transfer batch sizes, *discrete* versus *continuous* transfer batch sizes, *no idling* versus *intermittent idling* of machines, *finite* versus *infinite* buffers between machines, *lot-streaming* versus *lot-splitting*.

Constant vs Variable Number of Transfer Batches

In general, number of transfer batches of a job between machines are determined by the transportation costs or budget constraints. There is the need to find the size and the maximum number of transfer batches allowable given the transportation costs and budget constraints. However, except Trietsch (1989), the number of transfer batches as a predetermined quantity. Therefore, given the maximum number of transfer batch sizes, the problem reduces to finding their corresponding sizes.

Let $s_{j,m}$ be the number of transfer batches of a job j between machines m and $(m+1)$. In general, number of transfer batches of a job between machines m and $(m+1)$ may differ from the transfer batches between machines $(m+1)$ and $(m+2)$. We call this case as the *variable* number of transfer batches. In some cases, however,

it may be assumed that number of transfer batches of a job between pairs of machines is *constant*, i.e., $s_{j,m} = s_j,\ m = 1,\ldots,M$.

Consistent vs Variable Transfer Batch Sizes

In some manufacturing environments, it may not be cost effective to change the size of the transfer batches after they have completed their processing on each machine. Also, the lack of transfer batch transport of varying sizes may limit the number of times the transfer batch sizes can be altered. In these situations, it is quite reasonable to take into account the batch availability assumption and specify that the transfer batch sizes be constant (*consistent*) over all the machines in the problem, i.e., if we define $q_{j,i,m}$ as the size of the ith transfer batch of job j between machines m and $(m+1)$, then $q_{j,i,m} = q_{j,i},\ m = 1,\ldots,M$.

A special case of consistent sublots is generated by the requirement that all transfer batch sizes be *equal*, i.e., the total work is equally allocated to the transfer batches on all machines. In such a case, transfer batch sizing for a single job is no longer a problem when $s_{j,m} = s_j,\ m = 1,\ldots,M$.

However, it might also be reasonable to use item availability assumption and allow the transfer batches to vary at any point during the processing of the sublots which we will term as *variable* transfer batches.

It is clear that while consistent transfer batches are easier to run as their use reduces bookkeeping, variable sublots may be more effective for certain shop conditions and measures of performance. In some instances, it is also possible to have an *intermediate* version between consistent and variable transfer batches, where transfer batches are allowed to vary with the concept of batch availability.

To motivate the discussion consider an example containing 100 items to be processed in a three-machine flow shop. Machines 1, 2 and 3 take 1, 3 and 2 minutes per item, respectively. If we do not allow transfer batches, the makespan is 100(1+3+2) = 600 minutes. If we specify two *equal-sized* transfer batches through all machines, then the makespan will be 450 minutes, a reduction of 25% (Figure 1.2.a.). On the other hand, if we allow two *consistent* transfer batches of size 60 and 40 units, respectively, we can reduce the makespan to 440 minutes, a total reduction of 26.7% (Figure 1.2.b.). Finally, if we allow two *variable* transfer batches of size 25 and 75 units between machines 1 and 2, and transfer batches of size 60 and 40 units

between machines 2 and 3, then we can reduce the makespan further to 405 minutes, a total reduction of 32.5% (Figure 1.2.c.).

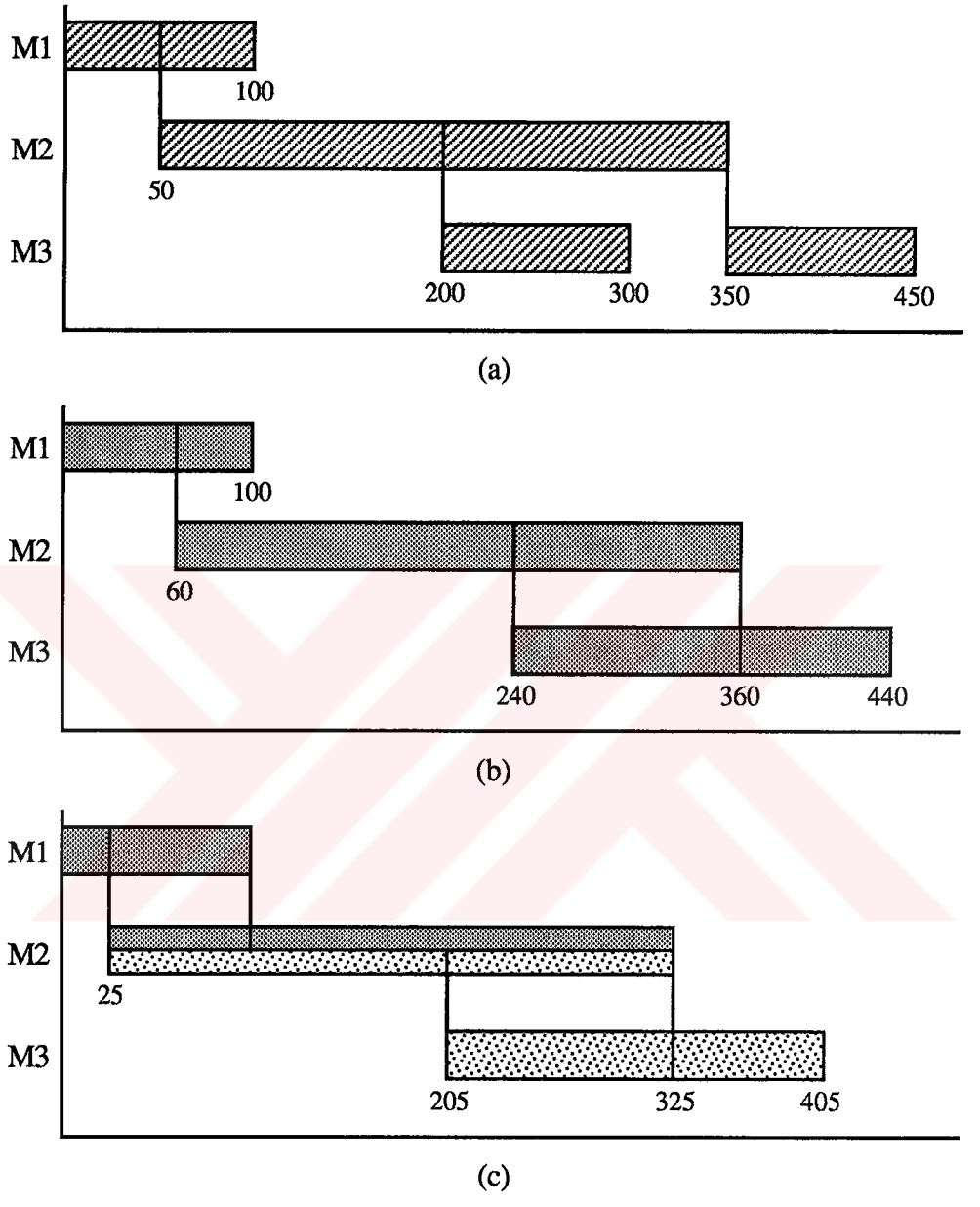


Figure 1.2. (a) Equal-sized, (b) consistent, (c) variable transfer batches

Discrete vs Continuous Transfer Batch Sizes

The scheduling problem with transfer batches can also have alternative solutions by specifying that the transfer batch sizes as either *discrete* (integer) or *continuous*. From a practical point of view, the ultimate goal of most problems is to find the integer sized

transfer batches. However, solving the discrete version of the problem is more difficult than the continuous version in which transfer batch sizes can be fractional. Thus, it is easier to solve continuous version of the problem and use these solutions as approximations to the discrete version of the problem.

In some cases, it may be possible to implement continuous transfer batch sizes if the original lot is large enough so that the solution to the continuous version of the problem may be very close to the optimal solution of the discrete version. Also, in such cases, even a simple rounding of the continuous solutions may provide acceptable and workable transfer batch sizes. Furthermore, by starting with the infinitely divisible case, we hope to develop general insights that will help solve more complicated variations.

No-idling vs Intermittent Idling

In a manufacturing environment, a requirement may be imposed that each machine, once started, must process the entire job (lot) continuously, without being idle. This may be needed in environments where there are prohibitive start-up costs for each machine. The *no-idling* criterion specifies whether or not a machine can be idle between its processing of transfer batches. If the no idling criterion is specified, each machine must process the entire lot continuously without any breaks in processing. In this case, once a machine has completed its processing on a transfer batch, it must start processing the next transfer batch immediately. However, with *intermittent idling* the situation is very different. In this case, a machine can be kept idle between processing transfer batches if restarting the machine is considerably less expensive than the cost of a major set-up per entire lot on the same machine.

No-wait vs Wait-in-process

In certain situations, it may be required that a transfer batch (or process batch) must be processed at each stage immediately after its processing is finished at the preceding stage. This is called as a *no-wait processing* (or continuous flow) within the context of scheduling theory. This constraint arises in real life situations in which each machine has no buffer where partially processed job lots on the preceding machine can temporarily be stored before processing. On the other hand, if this restriction is relaxed we allow transfer batches wait for processing in a buffer having an infinite capacity in front of a machine.

Lot-streaming vs Lot-splitting

In scheduling problems with transfer batches, there are two major cases that are conceptually worthy to distinguish. The first case involves overlapping operations for a given job lot. Here, the production run continues, but before the entire lot is complete, some processed portion (transfer batch) of the production lot is moved ahead to begin downstream operation. We refer to this case as *lot-streaming*. In the second case, however, we again permit the movement of transfer batches of a lot; but transfer batches of the lot can be intermingled with transfer batches of other lots while they are processed. We refer to this case as *lot-splitting*. It is clear that the lot-splitting is not possible in the single-job scheduling problem with transfer batches. But, both cases can be considered in multi-job cases.

In general, multi-job scheduling problem with *lot-streaming* can be defined as follows:

Given the maximum number and type of transfer batches between machines for all jobs, the problem is to determine simultaneously

. the size of transfer batches of the jobs as well as their movement time between machines, and

. the schedule (sequence) of jobs on the machines

so as to minimize a given measure of performance.

On the other hand, multi-job scheduling problem with *lot-splitting* can be defined as:

Given the maximum number and type of transfer batches between machines for all jobs, the problem is to determine simultaneously

. the size of transfer batches of the jobs, and

. the schedule (sequence) of transfer batches on the machines

so as to minimize a given measure of performance.

To illustrate the difference between lot-streaming and lot-splitting, consider a two-job two-machine flowshop problem in which each job is to be split into two sublots (transfer batches). The jobs have lot size of seven and fourteen identical items, respectively. The processing time per item of the first job is 1 and 2 time units on machine 1 and 2, respectively. However, the processing time per item of the second job is 1 and 2 time units on machine 1 and 3, respectively. If we do not allow lot-splitting, but allow lot-streaming, the optimal transfer batch sizes and the job sequence are as shown in Figure 1.3.a, respectively, where J_i denotes the ith transfer batch of job J. However, the size and the sequence of the sublots must be determined

simultaneously when we allow lot-splitting. In this case, makespan is reduced to 57 (which is optimal) as shown in Figure 1.3.b.

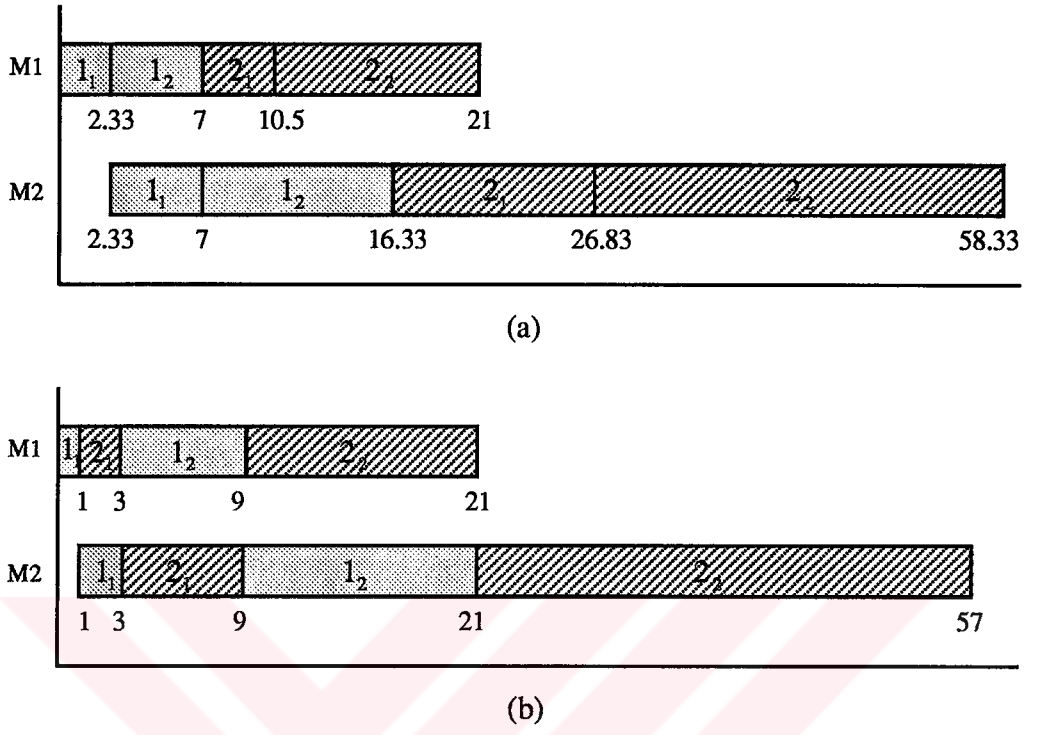


Figure 1.3. Optimal schedules with (a) lot-streaming, (b) lot-splitting.

1.3 An Extended Classification Scheme for the Scheduling Problems with Transfer Batches

Now, we briefly describe a systematic notation that could serve as a basis to classify and facilitate the presentation and discussion of deterministic and static scheduling problems with transfer batches. This classification scheme is adapted from the four-field classification scheme $\sigma|\alpha|\beta|\gamma$ of Conway *et al.* (1967), where σ is the number of jobs, α represents the machine environment, β defines the job characteristics and γ is the objective function.

Field σ:

Here the first field specifies the number of jobs to be scheduled.

$\sigma \in \{1, N\}$:

$\sigma = 1$: single job,

$\sigma = N$: multiple jobs.

Field α:

Here the first field takes the form $\alpha = \alpha_1\ \alpha_2\ \alpha_3\ \alpha_4$.

$\alpha_1 \in \{\varnothing, P, U, R, F, J, O, X, S\}$ is used to code the single-stage models:

$\varnothing$ – single machine,

P – identical parallel machines,

U – uniform parallel machines,

R – unrelated parallel machines,

and the multi-stage models:

F – flow shop,

J – job shop,

O – open shop,

X – mixed shop,

S – super shop.

$\alpha_2 \in \{\varnothing, M\}$ specifies the number of stages in the shop where

$\alpha_2 = \varnothing$: *variable* number of machines,

$\alpha_2 = M$: *fixed* number of machines.

$\alpha_3 \in \{\varnothing, P, U, R\}$:

$\alpha_3 = \varnothing$ and $\alpha_4 = \varnothing$: one machine at each stage of the shop.

$\alpha_3 \in \{P, U, R\}$: parallel machines at each stage as defined above for the single stage models.

$\alpha_4 \in \{\varnothing, m_1,\ldots,m_M\}$ specifies the number of parallel machines at stages.

Field β:

The second field $\beta \subset \{\beta_1,\ldots,\beta_8\}$ indicates various additional assumptions and restrictions on the way jobs are processed.

$\beta_1 \in \{\varnothing, r_j\}$:

$\beta_1 = \varnothing$: All ready times which a job may start processing are *zero*.

$\beta_1 = r_j$: All ready times are *arbitrary* non-negative integers.

$\beta_2 \in \{\varnothing, d_j, d_j = d\}$:

$\beta_2 = \varnothing$: No due dates are assumed to complete jobs.

$\beta_2 = d_j$: *Arbitrary* due dates for jobs.

$\beta_2 = d_j = d$: *Common* due dates for all jobs.

$\beta_3 \in \{\varnothing, p_{j,m} = p, dmn\}$:

$\beta_3 = \varnothing$: All processing times are *arbitrary* non-negative integers.

$\beta_3 = p_{j,m} = p_m$: All jobs have the *same* processing time requirement of time units.

$\beta_3 = dom$: There are *dominance* conditions among the jobs and machines. For example, *ordered* flow shops which will be discussed later.

$\beta_4 \in \{\varnothing, pmtn\}$:

$\beta_4 = \varnothing$: No *preemption* is allowed. *Preemption* means that each operation of a job can be interrupted at any time and resumed later on. If a job has fixed processing route, then after interrupting its operation on a machine, it is not allowed to be processed on any machine until the interrupted operation is completed. On the other hand, if a processing route for a job is not given in advance, then in between interruption and resumption of processing of the interrupted operation of the job may be processed (also with preemption) on any other machine it still has to be processed on.

$\beta_4 = pmtn$: Job preemption is allowed.

$\beta_5 \in \{\varnothing, prec\}$:

$\beta_5 = \varnothing$: *No precedence* constraints are specified.

$\beta_5 = prec$: Jobs are not independent. More precisely, there are some constraints which imply that some operations should precede other operations.

$\beta_6 \in \{\varnothing, S_d, S_a\}$:

$\beta_6 = \varnothing$: No setups are specified.

$\beta_6 = S_d$: *Detached* setup times of jobs. A physical unit (item) from the preceding operation of a job is not required, and setup on a machine can be performed on a downstream machine as soon as this machine is available.

$\beta_6 = S_a$: *Attached* setup times. The setup on a machine may require an item to be completed from the preceding operation. e.g., the setup of a special fixture on a machine usually requires an item to check settings. Note that both detached and attached setups are independent of job sequences.

$\beta_7 \in \{\varnothing, R\}$:

$\beta_7 = \varnothing$: No removal times are specified.

$\beta_7 = R$: *Removal times* are specified. Here, removal time is the time needed to remove the tools, jigs, fixtures, etc., on a machine once the job is processed on that machine.

$\beta_8 \in \{\varnothing, b, NW\}$:

$\beta_8 = \varnothing$: Buffers between stages are *infinite.*

$\beta_8 = b$: Buffers between stages are *finite.*

$\beta_8 = NW$: *No-wait* schedule, i.e., the buffer between stages is zero.

$\beta_9 \in \{\varnothing, NI\}$:

$\beta_9 = \varnothing$: *Intermittent idling* is allowed.

$\beta_9 = NI$: *No idling* is allowed.

$\beta_{10} \subset \{\pi, \theta, \rho\}$:

$\pi \in \{\varnothing, t, t_m, t_j\}$ specifies the *transfer time* of jobs between machines.

$\pi = \varnothing$: transfer times are assumed to be *negligible (or zero).*

$\pi = t$: transfer time for all jobs between all machines is *same.*

$\pi = t_m$: transfer time is the *same* for all jobs on each machine, but may *vary* between machines.

$\pi = t_j$: transfer time between machines is the *same*, but may *vary* for jobs.

$\theta \in \{\varnothing, n_T, n_{T,m}\}$ specifies the *number* of *transporters.*

$\theta = \varnothing$: *unlimited* number of transporters between machines.

$\theta = n_T$: *limited and* the *same* number of transporters between machines.

$\theta = n_{T,m}$: *limited and varying* numbers of transporters among machines.

$\rho \in \{\varnothing, c_T, c_{T,m}\}$ specifies the *capacity of transporters.*

$\rho = \varnothing$: capacity of transporters is *unlimited.*

$\rho = c_T$: capacity of transporters is the same between machines.

$\rho = c_{T,m}$: capacity of transporters can vary among machines.

- $\beta_{11} \subset \{\varnothing, TB(\lambda, \delta, \mu, \psi)\}$

$\beta_{11} = \varnothing$: No transfer batches is allowed.

$\beta_{11} = \{TB(\lambda, \delta, \mu, \psi)\}$: *Transfer batches* are allowed.

$\lambda \in \{strm, splt\}$:

$\lambda = strm$: *Lot-streaming* is allowed.

$\lambda = splt$: *Lot-splitting* is allowed.

$\delta \in \{\varnothing, s_m\}$:

$\delta = \varnothing$: *Constant* number of transfer batches throughout the shop.

$\delta = s_m$: *Variable* number of transfer batches between pairs of machines.

$\mu \in \{V, C, E, U\}$:

$\mu = V$: *Variable* transfer batch sizes.

$\mu = C$: *Consistent* transfer batch sizes.

$\mu = E$: *Equal-sized* transfer batches.

$\mu = U$: *Unit-sized* transfer batches.

$\psi \in \{c, d\}$:

$\psi = d$: Each transfer batch must contain *discrete* number of units.

$\psi = c$: No integrality requirement on the size of the transfer batches,

Field γ:

The third field specifies the single or multi-criteria objective functions to be optimized. Usually minimization of schedule performance measures discussed before is the concern, e.g. $C_{\max}$, $\overline{F}$, $\alpha_1 C_{\max} + \alpha_2 \overline{F}$, etc.

The following examples may serve to clarify this extended classification scheme:

(a) Problem $1|J|TB(s_m, C, d)|\sum q_i C_i$ reads as follows: Streaming of a single-job (lot) with variable number of discrete sized consistent transfer batches between pairs of stages and arbitrary processing times, ready at time zero for processing in a job shop with one machine at each stage, in order to minimize the total (mean) flow time of transfer batches.

Note that in this example specifying lot-streaming in a single-job case is redundant, and the notation *TB* will be sufficient, thus we drop "*strm*" from the second field for the cases denoted $1|\cdot|TB|\cdot$.

(b) Problem $N|F|TB(splt, C, c)|C_{\max}$ stands for: Splitting of multiple jobs with constant number of continuous consistent transfer batches between pairs of stages and arbitrary processing times, ready at time zero for processing in a general flow shop in which there are only one machine at each stage, in order to minimize the makespan of the schedule (completion time of the last transfer batch processed at the last stage).

Throughout the study, the following assumptions are also made:

1. All jobs are available and ready at time zero.
2. Each job has a specified number of transfer batches which is fixed across the stages in the shop.
3. There exists no priority orders within jobs and each job has the same degree of importance.
4. All machines in the system are available at any time. Breakdown or repair of any machine does not occur during the planning period.

1.4 Solution Complexity Aspects of Deterministic Scheduling Problems

Combinatorial analysis is often described as the mathematical study of the selection and arrangement of discrete objects, usually finite in number. In the past, most of the work in this area has been on the existence and number of possible selection or arrangements. A new line of combinatorial investigation has been increasing emphasis on finding the best possible selection or arrangement since early 70s. Each possible configuration is given some kind of score, an optimization process is carried out. This area of mathematics is termed *combinatorial optimization*. The scheduling problems belong to the broader class of combinatorial optimization problems. Hence, they may be analyzed in the same way as the latter.

There is usually a finite number of feasible solutions to each instance of the combinatorial optimization problems. Therefore, it seems intuitive that the fundamental theorem of combinatorial optimization could be employed: examine every feasible solution and choose the best. Unfortunately, there are usually far too many solutions for this approach to be practical. As an example, consider the sequencing of N jobs in a two-machine flow shop in which the processing sequence of the job on both machines are the same and minimizing mean flow time of the jobs is the objective function. It is easy to see that $N!$ possible sequences of jobs must be evaluated to find the optimal sequence. Clearly, running (execution) time of such a complete enumeration is effectively infinite if the number of jobs is large enough. However, we have to be able to do better then complete enumeration. Suppose an algorithm (a procedure for obtaining an optimal solution) is proposed for this combinatorial optimization problem. Here, the most important thing is the effectiveness of this algorithm. One generally accepted convention in the realm of combinatorial optimization is stated as follows:

An algorithm is considered to be effective if it can guarantee to solve any instance of the problem for which it was designed by performing a number of elementary computational steps where this number can be expressed as a polynomial function of the size of the problem.

The *complexity* of an algorithm refers to its execution time for finding a solution, which is usually expressed as a function of the size of the input. For our purpose, we need only to concentrate on the terms of a function that dominate the behavior of the execution time, which is called the *order-of-magnitude notation* $O(\cdot)$. An algorithm is said to have a complexity $O(N^3)$ when there exists a constant c such that the function cN^3 bounds the execution time as a function of input of size N. A solution algorithm whose complexity is bounded by a polynomial in N is a *polynomial-time* algorithm.

Based on the effectiveness of an algorithm, combinatorial optimization problems can be categorized as follows:

(i) Problems for which no algorithm can be written in mathematical and logical sense,

(ii) Problems for which at least one algorithm is known (or an algorithm could be developed), but no known effective algorithms have been reported yet,

(iii) Problems for which effective algorithms are available.

The problems having a polynomial-time algorithm (problems that fall into category (iii) above) are "easy" to solve, and denoted by P (standing for *polynomial*). On the other hand, the problems that are in category (ii) are "hard" to solve, and denoted by NP (standing for *non-polynomial*). NP contains a subset of problems termed *NP-complete*. Each problem in the set has the following properties: (1) it belongs to NP; (2) if an effective algorithm exists for it, then an effective algorithm exists for every problem in NP, i.e., $P=NP$. This means that NP-complete problems are the hardest in NP. It seems unlikely that an effective algorithm exists for any of the NP-complete problems. Hence, the fact that a problem is NP-complete is considered by some to be justification for approximation procedures to be applied to it.

In order to solve *NP hard* problems, one is obliged either to use *exact enumerative algorithms* (algorithms which gives the optimal solution) such as dynamic programming and branch-and-bound algorithms or to use *approximation algorithms*. Enumerative algorithms are used mainly for small instances of the problem or for solving problems of special structure. However, they require great efforts to determine an optimal schedule for large-scale problems that occur in actual situations.

Therefore, by approximation algorithms, it is better to get a near optimal solution with less computational efforts than to get an optimal one with great computational ones. Such an approach is often called a *heuristic approach.*

Some of the results on the complexity of flow shop and open shop scheduling problems are presented in Appendix A. For more detailed discussion on these scheduling problems and others as well as their complexity aspects, one can refer to the books published on the subject by Conway *et al.* (1967), Baker (1974, 1994), Rinnooy Kan (1976), Bellman *et al.* (1982), Dempster *et al.* (1982), French (1982), Morton and Pentico (1993), and review articles which survey the development of scheduling theory by Garey *et al.* (1976), Lenstra *et al.* (1977), Graham *et al.* (1979), Lageweg *et al.* (1982), Herrmann *et al.* (1993).

1.5 Outline of the Thesis

The purpose of this thesis is to analyze the various lot streaming and splitting problems to propose solution methods. The outline of the thesis is as follows.

In Chapter 2 we provide a survey of the recent studies in the area of scheduling with transfer batches with an emphasis on how these problems are modeled and what main aspects of the existing solution approaches can be identified. In addition, we also discuss the philosophy of the Optimized Production Technology (OPT) which underlines the notion of transfer batch-sizing issue.

Chapter 3 is devoted to single-job streaming problems in which makespan is taken as the measure of performance. In Section 3.1, two-stage hybrid flow shop in which stages may consist of several identical parallel machines are analyzed. We will show that the results for the two-machine flow shop lot streaming problem are easily extended for the two-stage hybrid flow shop environment. The general M-machine flow shop streaming problem with variable sized transfer batches is examined in Section 3.2, and we propose a polynomial time solution procedure which gives the optimal transfer batch sizes. Furthermore, in Section 3.3, the three-machine flow shop streaming problem with variable transfer batches and detached setup times on the machines is discussed, and properties of the optimal solution are developed. Finally, we close Chapter 3 by proving that the single-job streaming problem in an M-machine open shop is NP-hard if $3 \leq s < M$ where s is the number of transfer batches.

In Chapter 4, we again consider the single-job streaming problem in a general M-machine flow shop for a different measure of performance which is the mean flow time. We derive the characteristic of the optimal solution with the consistent transfer batch sizes, and then examine the special cases of this problem. In Section 4.1, we first show that the optimal transfer batch sizes are equal sized if the first machine has the longest processing time. Then, in Section 4.2, we propose a polynomial time solution procedure for another special case in which there are only two transfer batches. Third special case in which the number of machines is two is examined with detached setup times on the machines in Section 4.3. Finally, the last special case in which the number of transfer machines is three is analyzed and its optimal solution is derived.

The multi-job streaming problems in two-stage flow and open shops with makespan criterion are analyzed in Chapter 5. We first show that transfer batch sizing and sequencing decisions can be made independently in the two-machine flow shop scheduling with detached setup times and arbitrary transfer batches. Then we will examine the same problem with a no-wait restriction that once the processing of a transfer batch begins on the first machine, its subsequent processing must be carried out with no delay in the passage of the job from machine to machine. Finally, in Chapter 5, we develop the properties of the optimal solution to the multi-job two-machine open shop streaming problem with detached setup times on the machines.

Chapter 6 is devoted to the multi-job splitting problems. Here, we again take the makespan as the performance measure. In Section 6.1 we examine the basic two-machine flow shop problem. After deriving the properties of the optimal solution, we propose an two-level iterative solution procedure which gives very good approximate results as compared to the optimal solution. In Section 6.2, we again consider the two-machine flow shop problem in which transfer times between machines exist. But in this problem, we assume that transfer batch sizes for all jobs are given as unit or equal sized, and develop an optimal solution procedure.

The main results of the thesis and directions for further research are discussed in Chapter 7.

CHAPTER 2

AN OVERVIEW OF LITERATURE ON SCHEDULING PROBLEMS WITH TRANSFER BATCHES

In this chapter, we first provide a survey of the recent studies in the area of scheduling problems with transfer batches with an emphasis on how these problems are modeled and what main aspects can be identified of the solution approaches developed so far. The next issue will be the discussion of the philosophy of the Optimized Production Technology (OPT) which underlines the notion of transfer batching issue, and

2.1 A Review of Literature

Although the concept of overlapping operations has almost universal application in manufacturing systems, there are relatively few analytical studies which address several aspects of transfer batch concept to provide guidance for its use in lot-sizing and machine scheduling and sequencing problems. The previous research on scheduling problems with transfer batches follows two paths. The first, exemplified initially by Szendrovits (1975), treats the single lot scheduling problem by allowing operations overlapping in a multi-stage flow shop to minimize a composite cost function which gives the trade-off between cost of setting up a machine versus cost of holding inventory, on an individual machine basis. We will call this type of scheduling problems as the *lot scheduling problems involving lot-sizing decisions*.

On the other hand, the second path, exemplified first by Baker (1988), is based on the claim that the production lot size for a product is determined in an upper-level of the traditional hierarchical approach in production planning and scheduling activities. Thus, for a given lot size and its maximum number of transfer batches, the problem is to determine the transfer batches and their sequences which minimize a selected schedule performance which we discussed in Chapter 1. This type of scheduling

problems will be referred to as the *lot scheduling problems involving sequencing decisions.*

Within the context of the above classification, in the following subsections we will discuss the recent studies in the area of scheduling problems with transfer batches.

2.1.1 Lot Scheduling Problems Involving Lot-sizing Decisions

Until recently, there have been relatively little work on the lot scheduling problems dealt with schedule cost criteria. As we stated before, the first known research emphasizing the importance of overlapping operations made by moving transfer batches through the system is due to Szendrovits (1975). He presents a model to find a lot size of a single product which is produced through a fixed sequence of operations in a multi-stage (serial) system. He permits transfer of equal sized transfer batches (where the number of transfer batches is known and fixed through all stages) to reduce the cycle-time (makespan) of the lot, but no machine idleness between processing of transfer batches is permitted once processing begins on the lot. In his model, the sum of setup, finished products inventory and work-in process inventory costs are minimized while meeting a continuous demand for the product.

However, Goyal (1976) recognizes that the cycle time derived by Szendrovits is a function of two variables, namely production quantity per lot and number of transfer batches per lot. Furthermore, he states that the cycle time decreases as the number of transfer batches increases, as a result, the total cost will decrease. Thus, he claims that there must be a trade off between creating more transfer batches and moving these transfer batches through all the stages. Then he extends the Szendrovits' model to include the cost of moving transfer batches between operations, and provides a procedure which would not only give the optimal cycle time and minimum total cost, but also the optimal number of transfer batches (equivalently transfer batch size) for a given transportation cost which is constant through all the stages and independent form the transfer batch size. Szendrovits (1976) then provides a simplified method of solving this extended model.

Goyal (1977) reconsiders his extended model for a two-stage flow shop in which the lot can be split into unequal sized transfer batches. He proves that both process inventory and cycle time obtained by unequal sized transfer batches is less than the process inventory and cycle time obtained for equal sized transfer batches, and then

proposes a solution procedure which determines both optimal lot size and number of transfer batches.

Szendrovits and Drezner (1980) studies another lot-sizing model with transfer batches. In their model, again a lot is produced through a series of manufacturing stages (flow shop) with a single setup and without interruption at each stage. Transfer of partial lots is allowed between stages, and the transfer batch sizes are assumed to be equal sized at any particular stage, but the optimal number of equal-sized batches may differ across stages. Setup costs, the inventory holding costs and transportation costs influence the optimal transfer batch sizes at various stage. They propose an optimization method for this model.

Truscott (1985) uses the concept of transfer batch to minimize the cycle-time of a lot in a multi-stage manufacturing system. He notes that the use of transfer batches can have positive effects on the incurrence of fixed and carrying costs as well as on cycle-time. The manufacturing system he models is one where there is one lot which can be divided and which must be sent through a sequence of processes. The time required to perform each process is known. He considers two types of processes: processes whose completion time are proportional to the lot size and processes whose completion times are invariant to the lot size. He mentions that processes such as inspection and machining are proportional to the lot size while processes such as setups and transportation are not. In his model, he takes into account the same assumptions of the Szendrovits model. Although his primary objective is to minimize the completion time of the lot, he also has a secondary objective which is to minimize the number of loads between processes. Then, Truscott (1985) provides a heuristic algorithm which schedules the transfer batches on the multi-stage system.

Next, Truscott (1986) extends his model to include constraints on the transportation activities between processes. He considers two types of constraints on transportation activities; capacity constraints on the transport equipment and transportation time requirements on vehicles between processes. He provides algorithms that schedules the transfer batches so that the additional constraints given are taken into account.

In a recent study by Goyal and Gunasekaran (1994), the assumption of no-idling in Szendrovits' earliest work is relaxed, and his model is extended to the case in which each transfer batch must be processed immediately on the next machine (stage) after finishing its work on the current machine without waiting, i.e., no-wait restriction is assumed. Thus, for a given number of transfer batches, they determine the optimal production batch size.

Steiner and Truscott (1993) extends the Szendrovits' earlier work for the M-machine open shop in which the processing sequence of the operations of a lot is not predetermined as in the case of flow shops. In this situation, the process routing of the lot which is the sequence of operations and the transfer batch sizes must be determined simultaneously. Clearly, if the process routing is given, the problem is equivalent to the flow shop problem considered in Szendrovits' work.

On the other hand, if the transfer batch sizes are given, the problem is to determine the process routing for the lot. Steiner and Truscott (1993) have shown that when scheduling equal sized transfer batches in an open shop with continuous work on each machine (i.e., no-idling), the optimal transfer batch size is 1, and the optimal process routing is pyramidal, with respect to minimizing the cycle time (makespan). In a *pyramidal routing* $R_p = \{M_{(1)}, M_{(2)}, ..., M_{(M)}\}$, the job is processed on the machines with a nondecreasing order of unit processing times followed by machines with a nonincreasing order of unit processing times. i.e., the pyramidal routing satisfies the condition $p_{(1)} \leq p_{(2)} \leq ... \leq p_{(m)} \geq p_{(m+1)} \geq ... \geq p_{(M-1)} \geq p_{(M)}$ where $p_{(m)}$ is the unit processing time of the mth machine in the process routing. It is clear that out of $M!$ process routings, 2^{M-1} routings have the pyramid shape, one for each 2-partition of the machine set $M = \{M_1, M_2, ..., M_M\}$, and one of these pyramidal routings gives the optimal sequence of operations. Steiner and Truscott (1993) also considers the total (mean flow time) as the scheduling criterion, and shows that the optimal routing is obtained by sequencing the machines in nondescending order of their unit processing time, and the optimal transfer batch sizes are again of unit sized.

2.1.2 Lot Scheduling Problems Involving Sequencing Decisions

Within the context of scheduling theory, problems allowing the use of transfer batches with schedule performance criteria can be classified into two groups based on the number of jobs involved: *single job* and *multi-job* problems. Although their application areas may be limited, single job streaming problems are used to provide helpful insights and initial results in understanding the nature of multi-job problems. As we will discuss, the solution of multi-job problems potentially involves complicated structures, and thus may require heuristic solution procedures instead of exact ones. Therefore, the solution of the single job problems may also be used as a subproblem in exact or heuristic procedures to solve multi-job problems.

2.1.2.1 Single-job Problems

The current studies on single job problems, exemplified first by Baker (1987), examine the transfer batch sizing problem to allocate work to the given number of transfer batches (which is the constant between pairs of machines) to minimize the makespan of the lot. The notion that the number of transfer batches is given also deserves some comment. From a practical point of view, there might be a limit on the number of transfer batches that the system can track, this would naturally specify the number of transfer batches. From a theoretical point of view, the analysis for finite values of transfer batches shows us just how much a benefit can be attained by using small number of transfer batches, even though an infinite number of transfer batches would in principle be even better.

2.1.2.1.1 Flow Shop Problems

Consistent Transfer Batches

The basic single-job streaming problem in a general M-machine flowshop (which is expressed as $1|F|TB(C,c)|C_{\max}$ in our classification scheme) was first introduced by Trietsch (1988) and Baker (1988) independently. In this problem, the number of transfer batches through the shop is assumed to be given and constant, and furthermore, transfer batches are assumed to be consistent. The objective is selected as the minimization of the completion time of the last transfer batch on last machine M which is defined as the makespan of the job. Baker (1988) provided two linear programming formulations; one using completion times of the transfer batches as decision variables and another using the idle times on each machine as decision variables (see Appendix B). He then specialized the second formulation for the two-machine streaming problem and provided optimal transfer batch sizes as well as the makespan of the problem. In the following discussion, by classifying the problems according to the number of machines considered, we examine the results of Baker as well as the others obtained by the other researchers.

Two-machine Problems

Utilizing the second LP formulation mentioned above, Baker (1988) gives the following result for the solution of the two-machine flow shop streaming problem with consistent transfer batches $1|F2|TB(C,c)|C_{\max}$.

Theorem 2.1 (Baker, 1988) *The following transfer batch sizes are optimal for the* $1|F2|TB(C,c)|C_{\max}$ *problem,*

$$q_i = \pi^{i-1} q_1, \qquad i = 1,\ldots,s$$

$$q_1 = \frac{\pi^s - 1}{\pi - 1} Q$$

where $\pi = p_2 / p_1$.

Baker (1988) also makes some interesting observations about the solution of this problem. First, he notes that the optimal transfer batch sizes are geometric. Further, he notes that if the second machine is faster than the first (has a shorter processing time), the transfer batch sizes will be decreasing in size; however, if the second machine is the slower machine, then the transfer batch sizes will be increasing in size. He also notes that increasing the number of transfer batches reduces the makespan. As will be seen, analogs of these conditions hold true when various other consistent transfer batch models are considered.

Potts and Baker (1989) consider the same problem and prove the following result.

Theorem 2.2 (Potts and Baker, 1989) *In an optimal schedule for the lot streaming problem with two or three machines, the consistent transfer batches are optimal if the batch availability assumption is used.*

This result is interesting since there is a similarity between the results for consistent transfer batches in the lot streaming problem and results for permutation schedules in the traditional multi-job flow shop problem where no transfer batches are allowed. In the traditional problems with two and three machines, the optimal job sequences can be obtained by considering only permutation schedules, which force the processing order of jobs to be the same on each machine. Furthermore, there are instances with four machines where permutation schedules do not yield an optimal solution. For each of these statements about permutation schedules we have a corresponding result for consistent transfer batches.

Based on the above theorem, Potts and Baker (1989) show that in an optimal solution of the single-job streaming problem in two-machine flowshop all transfer batches are *critical*. That is, for any transfer batch k, the makespan is equal to the sum of processing times of transfer batches $1,\ldots,k$ on the first machine and the processing times of transfer batches $k,\ldots,s$ on the second machine; hence changing the size of any transfer batch increases the makespan. Thus, each transfer batch is critical. This result leads us to the solution obtained by Baker (1988). Moreover, in the optimal

solution both machines operates continuously and all transfer batches are critical, the solution of the $1|F2|TB(C,c)|C_{\max}$ problem is also optimal for the problems with restrictions of no-idling $(1|F2|TB(C,c),NI|C_{\max})$ or no-wait $(1|F2|TB(C,c),NW|C_{\max})$.

Trietsch (1989) considers the two-machine problem assuming a budget B on the total transfer cost of transfer batches from one machine to another. In the two-machine problem, there is only one type of transfer, with a transfer cost of c. A budget on the number of transfers is effectively a constraint on the number of transfer batches, i.e., the number of transfer batches is limited by $\lfloor B/c \rfloor$. Thus, Trietsch also obtains the geometric solution in the two-machine case.

Simple examples show that the solution of the discrete version of the two-machine problem, i.e., $1|F2|TB(C,d)|C_{\max}$, in which the transfer batches are integer sized could be substantially different form the best continuous solution. For this purpose, Trietsch (1989) also proposes an iterative (dynamic programming) algorithm of time complexity $O(s^2Q)$ to find the optimal integer transfer batch sizes.

However, a brief computational study of the discrete version of the two-machine problem on randomly generated problems shows that the minimum makespan realizable with integer sized transfer batches is always very close to the one for the continuous version of the problem. In other words, the integrality requirement does not significantly reduce the savings achievable in the makespan by lot streaming.

The optimal solution for the case of two machines involves geometric transfer batch sizes, illustrating that in theory the optimal solution involves different transfer batch sizes. In practice, however, it may be more attractive to use transfer batches of equal size as mentioned in previous chapter. In the case of equal transfer batches, in which $q_i = Q/s$ for $1 \le i \le s$, the makespan is given by

$$C_{\max}^{E} = \max\{\frac{p_1}{s} + p_2, \frac{p_2}{s} + p_1\}Q.$$

On the other hand, the optimal makespan achievable with consistent transfer batches is

$$C_{\max}^{C} = (\frac{\alpha - 1}{\alpha^s - 1} p_1 + p_2)Q.$$

Potts and Baker (1989) compares the makespan $C_{\max}^{E}$ to the optimal makespan $C_{\max}^{C}$ and proves the following result.

Theorem 2.3 (Potts and Baker, 1989) $C_{\max}^{E} / C_{\max}^{C} < 1.09$.

Vickson (1994) treats the optimal lot streaming problem in a two-machine flow shop with setup times. He deals with both detached and attached setup times in this

study. Recall that detached setups can be performed on a downstream machine as soon as this machine is available, but attached setups require a physical unit of the job from the preceding stage (machine).

Vickson (1994) considers the problem with detached setups $(1|F2|TB(C,c),S_d|C_{\max})$. Let S_m be the detached setup time on machine m for $m=1,2$. It is clear that then the optimal solution behaves as though no setup times existed if $S_1 \geq S_2$. Then, the problem is only examined for the case $S_1 < S_2$, and the following result is given by Vickson (1994) .

Theorem 2.4 (Vickson, 1994) *In the problem* $1|F2|TB(C,c),S_d|C_{\max}$,

(i) *if* $p_1Q \leq S_2 - S_1$, *then* $q_1 = Q$, $q_i = 0$ *for* $2 \leq i \leq s$, $C_{\max} = S_2 + p_2Q$,

(ii) *if* $p_1q_1^R \leq S_2 - S_1$, *then* $q_1 = q_1^R = Q(1-\alpha)/(1-\alpha^s)$, $q_i = \alpha^{i-1}q_1$ *for* $2 \leq i \leq s$, $C_{\max} = S_1 + p_1q_1 + p_2Q$,

(iii) *if* $p_1Q > S_2 - S_1$ *and* $p_1q_1^R > S_2 - S_1$, *then* $q_1 = (S_2 - S_1)/p_1$, $q_i = \alpha^{i-1}q_1$ *for* $2 \leq i \leq k$, $q_{k+1} = Q - \sum_{1 \leq i \leq k} q_i$, $q_i = 0$ *for* $k+2 \leq i \leq s$, $C_{\max} = S_2 + p_2Q$

where q_1^R *be the optimal size of the first transfer batch in the relaxed problem* $(1|F2|TB(C,c)|C_{\max})$ *without any setups.*

In the case of attached setups $(1|F2|TB(C,c),S_a|C_{\max})$, Vickson (1994) presents a linear programming formulation which is similar to the one for the detached setup case.

Three-machine Problems

For the special case of $s=2$, Baker (1988) studies the three-machine problem and, by modifying the second LP formulation given in Appendix B, provides the following result to find the optimal solution to the $1|F3|TB(s=2,C,c)|C_{\max}$ problem.

Theorem 2.5 (Baker, 1988) *In the problem* $1|F3|TB(s=2,C,c)|C_{\max}$, *if* $p_2^2 \leq p_1p_3$ *then optimal transfer batch sizes are*

$$q_1 = Q(\beta-1)/(\beta^2-1) = Q(p_1+p_2)/(p_1+2p_2+p_3)$$

$$q_2 = \beta\, q_1$$

where $\beta = (p_2 + p_3)/(p_1 + p_2)$*; otherwise the optimal transfer batch sizes are*

$$q_1 = \begin{cases} Q(\alpha - 1)/(\alpha^2 - 1) = Qp_1/(p_1 + p_2), & \text{if } p_1 \geq p_3, \\ Q(\upsilon - 1)/(\upsilon^2 - 1) = Qp_2/(p_2 + p_3), & \text{if } p_1 < p_3, \end{cases}$$

$$q_2 = \begin{cases} \alpha q_1, & \text{if } p_1 \geq p_3, \\ \upsilon q_1, & \text{if } p_1 < p_3, \end{cases}$$

where $\alpha = p_2/p_1$ *and* $\upsilon = p_3/p_2$.

Glass *et al.* (1994) extend the study of Baker (1988) for the general case of the three-machine flowshop problem with s transfer batches, i.e., the $1|F3|TB(C,c)|C_{\max}$ problem and present a solution procedure which gives the optimal transfer batches. In developing this solution procedure, they also use the network representation of the problem. They determine the general form that the longest or critical path would take in an optimal solution. They define any sub-path of the critical path as a critical segment. Next, they provide the following result.

Theorem 2.6 (Glass *et al.*, 1994) *In any optimal solution of the* $1|F3|TB(C,c)|C_{\max}$ *problem, every transfer batch is critical and all transfer batch sizes are positive.*

Then, machine m was defined to be *critical* from transfer batch i $(1 \leq i \leq s-1)$ in the project network if $(i,m) - (i+1,m)$ is a critical segment, where s is the number of transfer batches in the problem. Next, they provide the following theorem.

Theorem 2.7 (Glass *et al.*, 1994) *There exists a set of transfer batch sizes for which the project network of the* $1|F3|TB(C,c)|C_{\max}$ *problem contains at least two critical machines from transfer batch* i for $1 \leq i \leq s-1$.

Using this theorem, Glass *et al.* then prove the following theorem which gives a procedure for finding the optimal transfer batches.

Theorem 2.8 (Glass *et al.*, 1994) *In the problem* $1|F3|TB(C,c)|C_{\max}$*, if* $p_2^2 \leq p_1 p_3$ *then optimal transfer batch sizes are*

$$q_i = \beta^{i-1} q_1, \qquad i = 2,\ldots,s,$$

$$q_1 = \begin{cases} Q(\beta - 1)/(\beta^s - 1), & \text{if } p_1 \neq p_3, \\ Q/s, & \text{if } p_1 = p_3, \end{cases}$$

where $\beta = (p_2 + p_3)/(p_1 + p_2)$*; otherwise, there exists a transfer batch* k *for which the optimal transfer batch sizes are of the form*

$$q_i = \alpha^{k-i} q_k, \quad i = 1,\ldots,k,$$

$$q_k = \begin{cases} Q/[(\alpha^s - 1)/(\alpha - 1) + (\upsilon^{s-k+1} - 1)/(\upsilon - 1) - 1], & \text{if } p_1 \neq p_2,\ p_2 \neq p_3, \\ Q/[k - 1 + (\upsilon^{s-k+1} - 1)/(\upsilon - 1)], & \text{if } p_1 = p_2,\ p_2 \neq p_3, \\ Q/[(\alpha^s - 1)/(\alpha - 1) + s - k], & \text{if } p_1 \neq p_2,\ p_2 = p_3, \end{cases}$$

$$q_i = \upsilon^{i-k} q_k, \quad i = k,\ldots,s,$$

where $\alpha = p_1 / p_2$ *and* $\upsilon = p_3 / p_2$.

Although Theorem 2.8 gives the general form of the solution when $p_2^2 > p_1 p_3$, there is still a need to determine k, the crossover (pattern changing) transfer batch, in order to find the optimal transfer batch sizes and the corresponding makespan value. Glass *et al.* (1994) provide the following theorem which leads to a method of determining k.

Theorem 2.9 (Glass *et al.*, 1994) *In the problem* $1|F3|TB(C,c)|C_{\max}$*, if* $p_2^2 > p_1 p_3$*, the value of* k *is found by a bisection search in* $O(\log s)$ *time.*

It is easy to see that there is a proportionality relationship between the transfer batch sizes in this three-machine problem as there was in the two-machine problem. In the two-machine problem, the ratio of q_{i+1} / q_i is p_2 / p_1. However, in the three-machine problem, the ratio of q_{i+1} / q_i is $(p_2 + p_3)/(p_1 + p_2)$ if $p_2^2 \leq p_1 p_3$. If $p_2^2 > p_1 p_3$, then $q_{i+1} / q_i = p_2 / p_1$ for $1 \leq i \leq k$ and $q_{i+1} / q_i = p_3 / p_2$ for $k \leq i \leq s$ where k is the crossover transfer batch.

As yet, however, these algorithms have not been extended to more than three machines except the following case in which the number of transfer batches is restricted to be two.

M-machine Two-sublot Problem

A special case of M-machine s-sublot problem was studied by Baker and Pyke (1990) to find the optimal transfer batch sizes by allowing two transfer batches on each machine, i.e., $s = 2$. In their development, they analyze the lot streaming

problem as a project network problem where the objective is to minimize the longest path in the network. A representation of this network is given in Figure 2.1.

Using the network representation of the problem, they sought to minimize the length of a longest path in the network over all possible q_1 values, where q_1 is the size of the first transfer batch. (Note that the specification of q_1 determines q_2 in the two transfer batch problem since $q_2 = Q - q_1$. They find that associated with some subset of machine indices I, there is a path which begins at node (1,1) goes to node $(1,k)$ crosses over from node $(1,k)$ to $(2,k)$ and then continues from node $(2,k)$ to $(2,M)$, where $k \in I$. They show that the associated with this path, there is a range of values for q_1 for which this path is the longest path in the network. They also show how to find the upper and lower bounds on the values the transfer batch size q_1 that would be needed to maintain the path defined by machine k as the longest path in the network. They provide a method of choosing the q_1 value that would minimize this path defined by machine index k.

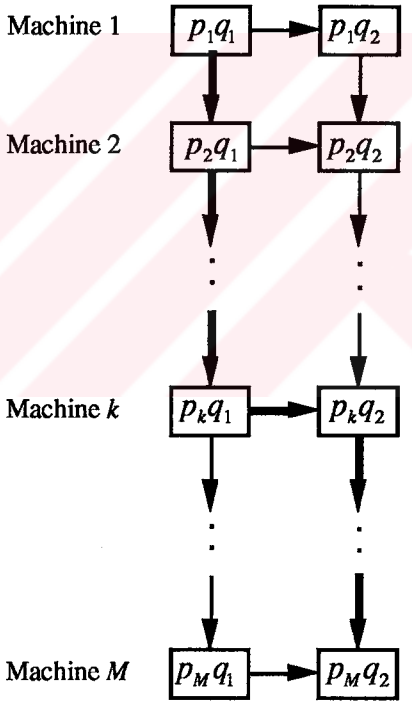


Figure 2.1. Network representation for the M-machine two-sublot problem

However, instead of suggesting searches over all possible longest paths in the network to find the one which will yield the shortest makespan by a judicious choice of q_1, they state that the path which is defined by the machine with the largest (or longest) processing time is always in the path which defines the optimal makespan. In

other words, they claim that the critical path in the network crosses over from the first column of nodes to the second at the machine with the largest processing time. Thus, they devise an algorithm which chooses the path defined by the machine with the longest processing time as the critical path and then picks the q_1 value which minimizes this path. However, although this path is indeed the critical path in many problems, problems do exist where this path is not the critical path. Because of this, Baker and Pyke's algorithm is sub-optimal under certain conditions. This error was first noted in Williams and Tüfekçi (1992).

Williams and Tüfekçi (1992) show that the path defined by the largest machine may not necessarily be the critical path in the network. Furthermore, they provide an algorithm (see Appendix D) with time complexity of $O(M^2)$ which first searches for the critical path, then determines the q_1 value which minimizes the length of this critical path.

Potts and Baker (1989) also compare the optimal makespan of the two-sublot case above with the makespan of two equal transfer batches and gives the following result.

Theorem 2.10 (Potts and Baker, 1989) *Let* $C^E_{\max}$ *and* $C^C_{\max}$ *the optimal makespans of the problems* $1|F|TB(s=2,E,c)|C_{\max}$ *and* $1|F|TB(s=2,C,c)|C_{\max}$, *respectively. Then,* $C^E_{\max} / C^C_{\max} < \frac{9}{8}$.

Variable Transfer Batches

For the two-machine case there is only one set of transfer batches (transfers from machine 1 to machine 2), so the issue of allowing variable transfer batches to minimize the makespan of the job is moot. Hence, by solving the problem $1|F2|TB(C,c)|C_{\max}$ we obtain the minimal makespan for the variable transfer batches case as well. However, for the three or more machine cases, allowing variable transfer batches will reduce the makespan of the schedule. For the three machines case, i.e., $1|F3|TB(V,c)|C_{\max}$, Trietsch and Baker (1993) define the set of machines that operate continuously as the *partition set*, or simply the partition. Machines 1 and 3 are always in the partition since any feasible makespan (thus optimal makepan) can be achieved by scheduling the first and third machines to operate continuously. They state that there is an optimal schedule that is no-wait schedule in which there is no queuing of transfer batches. Then they show that an optimal partition consists of machines 1 and 3 if $p_2^2 \le p_1 p_3$. However, if $p_2^2 > p_1 p_3$, then the optimal partition consists of all three machines. In this case, the problem decomposes into two subproblems. The first

subproblem for machines 1 and 2 determines the transfers from machine 1 to machine 2, and the second subproblem for machines 2 and 3 determines the transfers from machine 2 to 3. As a result, they obtain the following result.

Theorem 2.11 (Trietsch and Baker, 1993) *The optimal solution to the* $1|F3|TB(V,c)|C_{\max}$ *is obtained by applying the solution of the problem* $1|F3|TB(C,c)|C_{\max}$ *if* $p_2^2 \le p_1 p_3$, *and the solution of the problem* $1|F2|TB(C,c)|C_{\max}$ *to each pair of machines consecutively if* $p_2^2 > p_1 p_3$.

Trietsch and Baker (1993) also consider the discrete version of the three machine problem with variable transfer batches. They extend the notion of decomposition in the problem $1|F3|TB(V,c)|C_{\max}$ to the problem $1|F3|TB(V,d)|C_{\max}$ in an analogous fashion. If $p_2^2 > p_1 p_3$, each subproblem is solved by the two machine procedure for the discrete version $1|F2|TB(C,d)|C_{\max}$, as described earlier. On the other hand, when $p_2^2 \le p_1 p_3$, they extend the two machine procedure to the three machines.

Another version of three-machine problem with variable transfer batches is the case in which variable transfer batches are permitted but no-idling restriction (i.e., the machines work continuously without any idle time between processing of transfer batches) is required which is represented by $1|F3|TB(C,c),NI|C_{\max}$, transfers between each pair of machines are determined independently of the rest of the schedule, as if there were only two machines in the model. In the $1|F3|TB(C,c),NI|C_{\max}$ problem, the initial idle time on the last machine is equal to the time required for the first transfer batch at each machine preceding the last. Adding this initial idle time to the processing required on the last machine yields the makespan. The resulting makespan is given by

$$C_{\max} = [(\sum_{m=1}^{M-1} \frac{p_{m+1} - p_m}{(p_{m+1}/p_m)^s - 1}) + p_m]Q.$$

Then, we have the following result.

Theorem 2.12 (Baker and Trietsch, 1993) *The optimal solution to the* $1|F3|TB(C,c),NI|C_{\max}$ *is obtained by applying the solution of the problem* $1|F2|TB(C,c)|C_{\max}$ *to each pair of machines consecutively.*

Note that this result is also valid to the discrete version of the problem, $1|F3|TB(C,d),NI|C_{\max}$.

Baker and Jia (1993) examine the improvement in the makespan that results from increasing the number of transfer batches allowed for the three-machine lot streaming problem. The results of their study describes that the improvement in the makespan

shows diminishing returns as the number of transfer batches increases. They examine the various lot streaming procedures for different constraints such as consistent, variable and equal transfer batches when there are 2, 3, 5, 8, and 10 transfer batches allowed. For example, they observe that more than half of the potential benefit from 10 transfer batches is obtained with just two transfer batches, and roughly 80% of the benefit is obtained with three transfer batches. Thus a small number of transfer batches is sufficient to achieve most of the benefits of lot streaming, no matter which procedure is used.

A general mixed integer programming model formulation for the M-machine flow shop problem which handles both consistent and variable transfer batches is proposed by Benli (1994). This model also handles a more realistic case in which the number of transfer batches between machines can vary. The formulation is basically a periodic review model with variable period lengths, which are the decision variables (see Appendix C for the details).

Potts and Baker (1989) introduce an *intermediate* version between consistent and variable transfer batches, which may yield intermediate makespans for more than three machines. Under this version, transfer batches are allowed to vary, but only if all upstream transfer batches that contribute items to a new transfer batch have been delivered by the time a new transfer batch is started. Potts and Baker show that for three machines their version is identical to consistent transfer batches. For more than three machines they provide an example where the makespan is shorter than it is under consistent transfer batches, but where it can be shown that the makespan could be improved even further under variable transfer batches.

2.1.2.1.2 Open Shop Problems

In open shop lot streaming problems, from the view of processing route of transfer batches, there are two cases to be considered. In the first case, all transfer batches of a job is restricted to follow the same order of machines for processing, thus we will call this case as *single routing* problem. In this problem, the decision to be made is to determine both the routing of the job and the transfer batch sizes which minimizes the given measure of performance. Note that if the routing is pre specified, the problem is equivalent to the flow shop lot streaming problem. In the second case, we relax the above restriction to achieve more flexibility by allowing different routings for each transfer batch of the job, i.e., the transfer batches may follow different order of machines for processing. We call this type of problems as *variable routing* problems.

In this case, we need to determine the transfer batch sizes as well as the schedule of transfer batches on different machines.

Theorem 2.13 (Glass *et al.*, 1994) *If the number of transfer batches is greater than or equal to the number of machines, i.e.,* $s \geq M$, *consistent transfer batches with sizes*

$$q_i = \begin{cases} Q/s & \text{if } 1 \leq i \leq M, \\ 0 & \text{if } m < i \leq M, \end{cases}$$

are optimal to the single-job with s *transfer batches* M*-machine open shop makespan minimization problem. Furthermore, the processing order of transfer batch* i *is* $(M_i,\ldots,M_M,M_1,\ldots,M_{i-1})$, *and the corresponding minimum makespan is* $C_{\max} = \max_{1 \leq m \leq M}\{p_m\}Q$.

Note that, in each of the M equal length of intervals within the interval $[0,C_{\max}]$ each machines processes exactly one of the M transfer batches, and hence the processing time intervals on any machine m do not overlap (See Figure 2.3 in which machine 3 has the longest unit processing time). From this theorem, it is clear that time complexity of finding minimum makespan is $O(M)$ and time complexity to output optimal transfer batch sizes is $O(s)$, hence time complexity to output an optimal schedule is $O(sM)$.

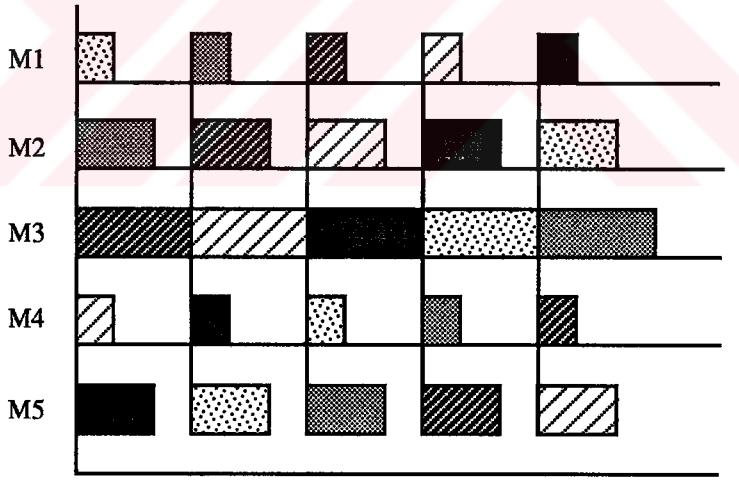


Figure 2.3. Optimal schedule if $s \geq M$.

Glass et al. (1994) also give the following result for a special case of the problem when $s = 2$.

Theorem 2.14 (Glass *et al.*, 1994) *If* $s = 2$, *consistent transfer batches with sizes* $q_1 = q_2 = \frac{Q}{2}$ *are optimal to the* M*-machine open shop makespan minimization*

problem. Furthermore, the algorithm of Gonzales and Sahni (1976) generates an optimal schedule with minimum makespan of $C_{\max} = \min\{\sum_{i=1}^{M} \frac{1}{2} p_i \ , \ \max_{1 \le m \le M} \{p_m\}\} Q$.

Note that both minimum makespan and an optimal schedule are found from the algorithm of Gonzales and Sahni (1976) in $O(M)$ time if $s = 2$. However, Glass *et. al* (1994) states that the complexity of the M-machine open shop lot streaming problem is unknown for $3 \le s < M$. In Chapter 3, we will show that this problem is NP-Hard.

2.1.2.1.3 Job Shop Problems

Jacobs and Bragg (1988) use a simulation model to study the effects of lot streaming in a closed job shop. Their experiments suggest that the use of transfer batches can be very effective in reducing mean flow time and that lot streaming can provide much better results. However, Jacobs and Bragg create transfer batches by splitting the original lot into equal transfer batches and do not examine other ways of allocating the work. They also show that the time saved by allowing lot streaming can more than compensate for extra setups incurred when production lots are split, so that small lot sizes can be used even in the presence of setup times.

In general, if a single job has M or less operations need to be processed on different machines in an M-machine job shop, it is easy to see that single job streaming problem in a job shop is equivalent to the flow shop lot streaming problem. Thus, the single job streaming problem only differs from the flow shop lot streaming problem if some operations require the same machine for processing. Based on this fact, the first analytical study on the lot streaming problem in a two-machine job shop environment is performed by Glass *et al.* (1994) for a single job split into s consistent transfer batches having three operations in which two of the operations require the same machine. In this case, it is clear that the only type of lot streaming for a single job which remains to be analyzed is that in which the first and third operations require machine 1 and machine 2 is required for the second operation.

Glass *et al.* (1994) show that this problem also can be easily solved using the results for the three-machine flow shops which is discussed before as follows. Let p_k be the processing time of kth operation. Relax the restriction that the first and third operations require the same machine, i.e., introduce a dummy machine 3 to handle processing for the last operation, and solve the resultant three-machine flow shop lot streaming problem. Then, without increasing the makespan, convert this schedule into

an optimal schedule with no idling on machines 1 and 3 by right shifting the transfer batches. By a transformation, any overlapping intervals of processing on machine 1 and 3 is eliminated so that the resulting schedule is feasible for the original problem.

Glass *et al.* (1994) prove that the relaxation algorithm above is optimal for the original problem with a resulting schedule having a makespan of $\max\{C^R_{\max}, (p_1 + p_3)Q\}$. Note that the flow shop relaxation algorithm generalizes to an M-machine job shop lot streaming problem in which number of operations is $M+1$ and the first and the last operation of the job is performed at the same machine. Unfortunately, there is no available algorithm to solve the general M-machine ($M > 3$) job shop lot streaming problem for which the variable transfer batches may be required since there is no algorithm even to solve the M-machine flow shop lot streaming problem with variable transfer batches.

2.1.2.2 Multi-job Problems

In the literature, we observe that much research has been conducted for the single-job problems, and there are relatively few analytical studies which address the several aspects of the multi-job problems and provide guidance for their use. Here, we again follow a similar path of discussion as we did for the single-job problems. We will first analyze the flow shop problems and then other multi-stage problems.

2.1.2.2.1 Flow Shop Problems

As we mentioned in Chapter 1, one of the concept in scheduling problems with transfer batches is the assumption of item availability. With this assumption, individual items in a job become available for processing at the next machine as soon as they are finished on the current machine, and it is our benefit to form unit sized transfer batches if there is no restriction on the number of transfer batches. In other words, the number of transfer batches will be equal to the lot size of the job. i.e., $s_j = Q_j$ for $1 \le j \le N$. As a consequence, the transfer batch sizing is no longer a problem so that the analysis is focused on the sequencing of transfer batches as well as their corresponding jobs.

The earliest studies which consider the multi-job scheduling problems with unit sized transfer batches starts with Vickson and Alfredson (1992) and Çetinkaya and Kayalıgil (1992). Analogously to the classical flow shop makespan minimization

problem, Vickson and Alfredson (1992) identify each transfer batch of all jobs as distinct jobs and proved the following result for the $N|F|TB(splt,U)|C_{\max}$ problem.

Theorem 2.15 (Vickson and Alfredson, 1992) *The optimal schedule of the* $N|F|TB(splt,U)|C_{\max}$ *problem uses*

(i) the same ordering of transfer batches on the first two machines, and

(ii) the same ordering of transfer batches on the last two machines.

As an immediate result of the theorem above, they give the following corollary.

Corollary 2.1 (Vickson and Alfredson, 1992) *In the* $N|F3|TB(splt,U)|C_{\max}$ *problem, there is an optimal schedule of permutation type: the same transfer batch ordering applies on all machines.*

Vickson and Alfredson (1992) observed that the solution of the problem with unit sized transfer batches in a two-machine flow shop can also be obtained from the Johnson's algorithm by identifying each unit as a distinct job. Their following result implies that there exists an optimal schedule with no lot-splitting, i.e., lot-streaming is sufficient in the optimal solution.

Theorem 2.16 (Vickson and Alfredson, 1992) *In the optimal schedule of the problem* $N|F2|TB(splt,U)|C_{\max}$, *job* i *precedes job* j *in an optimal schedule if* $\min(p_{i,1}, p_{j,2}) \le \min(p_{j,1}, p_{i,2})$ *where* $p_{j,m}$ *is the processing time per item of job* j *on machine* m, $m = 1,2$.

Çetinkaya and Kayalıgil (1992) extend the study of Vickson and Alfredson (1992) for the cases in which there are *detached* or *attached* setups are incurred. As we defined before, a setup is called as detached if the setup is performed on a machine as soon as this machine available. On the other hand, if a setup requires a physical unit (item) from the preceding machine, then it is called as attached (See Figure 2.2). Using our extended classification scheme of the scheduling problems, this problem is denoted by $N|F2|TB(strm,U),S_d$ or $S_a|C_{\max}$.

Çetinkaya and Kayalıgil (1992) define the following variables in a right-shifted processing timetable for a job j:

RI_j = run-in delay of job j = latest delay time of a setup for job j on machine 2 without affecting the completion time of the job.

RO_j = run-out delay of job j = difference between the completion times on machine 1 and machine 2 of job j.

Then, for a job j, if the setup is detachable, then the run-in and run-out delays are given as follows:

$$RI_j = \max\{(S_{j,2} - S_{j,1}), p_{j,1}, p_{j,2} - (B_j - A_j)\} - (S_{j,2} - S_{j,1})$$

$$RO_j = RI_j + (S_{j,2} - S_{j,1}) + (B_j - A_j).$$

Now, assume that the setup for a job is attached. Following a similar logic used for detached setup case, we have the run-in and run-out delay for a job as follows:

$$RI_j = S_{j,1} + p_{j,1} + \max\{0, (p_{j,2} - p_{j,1}) - S_{j,2} - (B_j - A_j)\}$$

$$RO_j = RI_j + (S_{j,2} - S_{j,1}) + (B_j - A_j).$$

Then, once the run-in and run-out delays of the jobs an optimal sequence of the jobs is determined by the following theorem.

Theorem 2.17 (Çetinkaya and Kayalıgil, 1992) *Job i precedes job j in an optimal schedule of the problem $N|F2|S_d$ or $S_a, TB(strm, E, c)|C_{\max}$ if*

$$\min(RI_i, RO_j) \le \min(RI_j, RO_i).$$

Using Theorem 2.17, Çetinkaya and Kayalıgil construct an optimal schedule which is similar to the Johnson's rule.

Vickson and Alfredson (1992) also consider the multi-job three-machine scheduling problem with unit-sized transfer batches and present the following results for some special cases of the problem by an analogy to the special cases of the classical three-machine scheduling problem $N|F3|\ |C_{\max}$ without transfer batches.

Theorem 2.18 (Vickson and Alfredson, 1992) *In the $N|F3|TB(splt,U)|C_{\max}$ problem, an optimal job sequence is obtained using Johnson's Algorithm with processing times $(p_{j,1} + p_{j,2}, p_{j,2} + p_{j,3})$ if*

(a) machine 1 dominates machine 2, i.e., $p_{i,1} \ge p_{j,2}$, $\forall i, j$; or
(b) machine 3 dominates machine 2, i.e., $p_{i,3} \ge p_{j,2}$, $\forall i, j$.

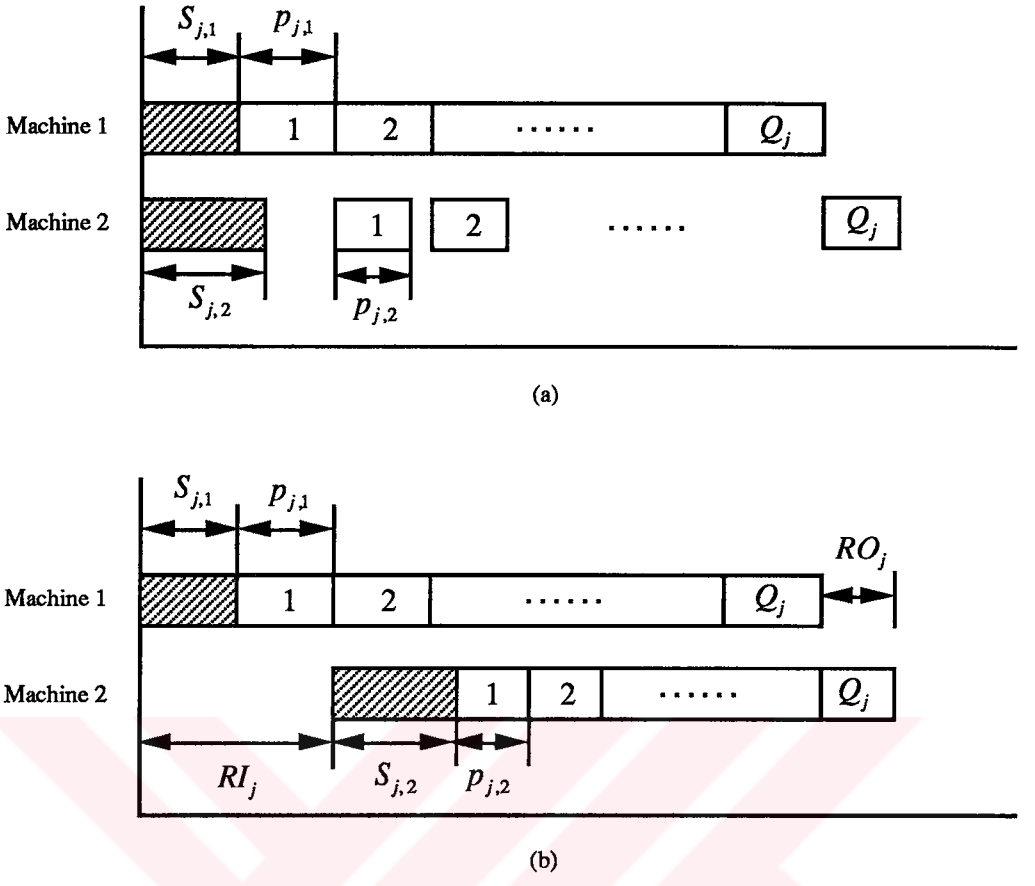


Figure 2.2. (a) Detached setups (b) attached setups

Theorem 2.19 (Vickson and Alfredson, 1992) *In the* $N|F3|TB(splt,U)|C_{max}$ *problem, if*

(a) machine 2 dominates machine 1 (i.e., $p_{i,2} \geq p_{j,1}$, $\forall i,j$*), the problem can be solved by letting each of the* N *jobs to be first, sequencing the remaining* $(N-1)$ *jobs using Johnson's Algorithm with processing times* $(p_{j,2},p_{j,3})$ *and by then choosing the best of the resulting* N *schedules.*

(b) machine 2 dominates machine 3 (i.e., $p_{i,2} \geq p_{j,3}$, $\forall i,j$*), the problem can be solved by letting each of the* N *jobs to be last, sequencing the initial* $(N-1)$ *jobs using Johnson's Algorithm with processing times* $(p_{j,1},p_{j,2})$ *and by then choosing the best of the resulting* N *schedules.*

2.1.2.2.2 Open Shop Problems

To our knowledge, there is only one analytical study which treats the multi-job streaming problem in open shops by Şen *et. al.* (1995). The authors consider a two-

machine open shop problem and extend it for the case each job is split into pre specified number of transfer batches. They observe that lot streaming or lot splitting is beneficial if there exists a job k such that $P_{k,1} + P_{k,2} > \max\{\sum_{1 \le j \le N} P_{j,1}, \sum_{1 \le j \le N} P_{j,2}\}$ where $P_{j,m}$ is the total processing time of job j on machine m. In Chapter 4, we extend this study to the case in which there are detached setups for each job.

2.1.2.2.3 Job Shop Problems

Solving the multi-job lot streaming problem optimally is an *NP*-Hard problem and can only be done for very small instances of the problem, Dauzere-Peres and Lasserre (1993) propose a two-level model. In the upper level (called *lot streaming* level), given a fixed number of transfer batches and the ordering of transfer batches on the machines (i.e., the sequence is fixed), optimal transfer batch sizes are determined by solving a simple Linear Programming problem which minimizes the makespan. In the lower level (called *scheduling* level), given the fixed transfer batch sizes, an ordering of the transfer batches on the machines is determined by minimizing the makespan in a classical job shop where the transfer batches are processed independently. However, the scheduling problem that must be solved in the lower lever still is *NP*-Hard. For this purpose, Dauzere-Peres and Lasserre (1993) use a heuristic method. They also report that numerical experiments on a sample show the great improvements that can be achieved with lot streaming. In few iterations, the makespan becomes closer to a lower bound with often few transfer batches, suggesting that the procedure is efficient. Although more transfer batches are needed when jobs become larger, their experiments show that the required number of transfer batches is slowly increasing.

2.2 Optimized Production Technology (OPT) and its Philosophy

This section is devoted to describe an approach called Optimized Production Technology (OPT) which was originated in the late 70s and early 80s as an extension of the MRP approach to the understanding management of a manufacturing system. The approach contains many of the insights which underlie the Japanese Kanban System which is an important element in Just-in-time (JIT) manufacturing.

From the OPT perspective there is one goal for a manufacturing company: to make money. All activities in the business are means to achieve this goal. Progress towards the achievement of this goal is measured in terms of net profit, return on investment,

and cash flow. At the operational level, OPT identifies three important criteria that are useful in evaluating manufacturing progress namely, *throughput*, *inventory*, and *operating expenses*.

Throughput is the rate at which manufacturing business generates money through selling finished goods. *Inventory* is defined as the raw materials, components and finished goods that have been paid for by the business but have not, as yet, been sold. *Operating costs* are the cost of converting inventory into throughput. Changes in any of these three elements result in changes in the financial measurements listed above. The goal of manufacturing, as understood by OPT, is to increase throughput while simultaneously decreasing inventory and operating expenses through realistic and optimized schedules.

The basic concepts in OPT philosophy was developed by Eliyahu Goldratt, and are embodied in a set of commonsense rules which are to some extent sound principles for any manufacturing system. An analytical technique (which is a proprietary product) implementing these rules have been computerized to give a software product called OPT/SERVE by Creative Output of Milford, Connecticut. The ten rules of OPT may be applied to the manufacturing organization without recourse to the software system. We now review each of these rules starting with those in relation to manufacturing bottlenecks.

2.2.1 Bottlenecks

The evolution of OPT's philosophy depends on the ideology called Theory of Constraints (TOC) by Goldratt and his associates. The theory of constraints has many of the same underpinnings as linear programming, but it adds some more operational concepts for dealing with constraining situations. Within the context of theory of constraints, the resources in a manufacturing facility are considered as constraints. These resources are classified into bottleneck and non-bottleneck resources. A *bottleneck* can be defined as a point or storage in the manufacturing process that holds down the amount of product that a factory can produce. It is where the flow of materials being worked on, narrows to a thin stream (Bylinsky, 1983).

OPT argues that the non-bottleneck should work at a reduced level of utilization, sufficient to support the bottleneck while at the same time preventing a build-up of work-in-process at the bottleneck station, and the bottleneck should work at 100 percent utilization. With regard to non-bottleneck resources, not all of their time can be

used effectively and some of their time is therefore considered as enforced idle time. The first rule of OPT derives from the insight and is as follows:

Rule 1 : *The level of utilization of a non-bottleneck is determined not by its own potential, but by some other constraint in the system.*

Traditionally, utilization and activation were considered to be the same. However, the pioneers of OPT make an important distinction between doing the required work (what we shall do, activation) and performing work not needed at a particular time (what we can do, utilization). Thus, it is vitally important to schedule all non-bottleneck resources within the manufacturing system based on the constraints of the system, which are usually bottlenecks. Then, the next rule is stated as follows:

Rule 2 : *Utilization and activation of a resource are not synonymous.*

2.2.2 Setup Times

Another important aspect of shop floor activities is the *setup time*, the time needed to set up the tools, jigs, fixtures, etc., for a task processing on a resource. However, there is a difference between the setup times on a bottleneck and those of a non-bottleneck resource. If we can save an hour of setup time on a bottleneck resource, we gain an hour of processing time. Relating this to the fact that bottlenecks are limiting constraint on other resources and on the system as a whole, an hour of production gained at a bottleneck has far reaching implications. It can be equivalent to an increased hour of production and throughput for the total system.

At a non-bottleneck resource, however, we have the elements, namely, setup, processing, and idle time. Clearly, if we can save an hour of setup time, we gain an hour of idle time as the bottlenecks still constrain the capability of the non-bottleneck. Consequently, an hour saved at an non-bottleneck is likely to be of no real value. There is however one advantage to reducing setup times at non-bottleneck machines. Due to a lower setup time, more setups can be used and the batch (lot) sizes can be reduced. While a smaller lot size of itself does not increase throughput, it tends to reduce inventory levels and operating expenses.

The next three OPT rules are deduced from the above points.

Rule 3 : *An hour lost a bottleneck is an hour lost for the total system.*

Rule 4 : *An hour saved at an non-bottleneck is just a mirage.*

Rule 5 : *Bottlenecks govern both throughput and inventory in the system.*

2.2.3 Lot (Batch) Sizes

Another important factor on the shop floor is lot (batch) size. It is closely linked to the inventory and throughput of the organization. Traditionally, one lot size was determined as being optimal for the manufacturing process, and splitting of lots into transfer batches and overlapping of batches were traditionally discouraged in manufacturing planning. However, from the OPT perspective, there are two types of lot sizes namely process batch and transfer batch. The *process batch* defines the number of items processed at a certain stage before a new setup is necessary. The *transfer batch* defines the number of items transferred together as a batch between operations. Therefore, the process and transfer batches are the lot sizes from the resource and parts point of view, respectively. OPT argues that process batch and transfer batch should not necessarily be the same size. Thus, the next OPT rule is derived from this distinction as

Rule 6 : *The transfer batch may not, any many times should not, be equal to the process batch.*

The lot-sizing issue is closely related to a scheduling approach in OPT. The basic concept is to move material as quickly as possible through nonbottleneck work centers until it reaches the bottleneck. There, work is scheduled for maximum efficiency (large batches). Thereafter, work again moves at maximum speed to finished goods. What this means for lot-sizing is very small transfer batches to and from the bottleneck, with a large process batch at the bottleneck. This implies that the process batch size at different work centers should not be the same. In the OPT approach, the process batch size should not be fixed, potentially vary by operation and over time. The process batch size is established dynamically for each operation and balances inventory cost, setup costs, component flow requirements and the needs for managerial control and flexibility. The next rule is derived from this point as

Rule 7 : *The process batch should be variable, not fixed.*

2.2.4 Lead Times and Priorities

MRP uses lead times to offset from a defined due date to calculate the time to start production or to release purchase orders. Essentially, MRP uses the estimated lead

times to determine the order in which jobs are processed. Priorities are assigned to jobs and those with the higher priorities are processed first. The estimated lead time is in turn dependent on the estimated queuing time for each operation. Once priorities have been established, the capacity of the production process is verified to see if the plan can be met. However, the important interaction between priority and capacity is not examined. Priority and capacity are essentially considered sequentially, not simultaneously. OPT, however, argues that lead times are not fixed and further that lead times are not known a priori, but depend on the sequencing at the limited capacity or bottleneck resources. Exact lead times and hence priorities can not be determined in a capacity bound situation. Hence, OPT suggest the following rule.

Rule 8 : *Capacity and priority should be considered simultaneously, not sequentially.*

The eight rules above focus primarily on the operational level and particularly the scheduling of work through the shop floor. The next two OPT rules are concerned with the performance measures used to monitor the effectiveness of the shop floor.

Rule 9 : *Balance flow not capacity.*

Rule 10: *The sum of local optima is not equal to the optimum of the whole.*

These ideas and rules of OPT were used to create a software package which was claimed would generate an optimum schedule for a given manufacturing system. The OPT software system is based on a closely guarded algorithm in which the details of this procedure were never made public, and the claim of optimality can not be verified. What is clear that the software generates schedules by progressively recognizing bottlenecks in the manufacturing system and ensuring that these bottlenecks are kept working continuously.

CHAPTER 3

SINGLE-JOB STREAMING PROBLEMS WITH THE MAKESPAN CRITERION

Although their application areas may be limited, single-job streaming problems can be useful in understanding the nature of multi-job problems. These problems may also be used as a sub problem in exact or heuristic procedures to solve the multi-job problems. In this chapter, several single-job streaming problems are studied with the minimization of the makespan as their criteria. In Section 3.1, two-stage hybrid flow shop will be the production environment in which stages may consist of several identical parallel machines. Then, in Section 3.2, we examine the general M-machine flow shop streaming problem with s variable sized transfer batches and propose a polynomial time solution procedure which gives the optimal transfer batch sizes. In Section 3.3, the three-machine flow shop streaming problem with s variable sized transfer batches and detached setup times on the machines is discussed and properties of the optimal solution are developed. Finally, we will close this chapter by proving that the single-job streaming problem in an M-machine open shop is NP-hard if $3 \leq s < M$ where s is the number of transfer batches.

3.1 Two-stage Hybrid Flow Shop Problem

As it is seen in Chapter 2, most of the results on streaming problems consider the production process as multiple stages in series (flow shops) in which each stage consists of a single machine. However, in many practical problems, the case is usually a mixture (hybrid combination) of different types of production process. Hence, a realistic sequencing problem is the investigation of hybrid flow shop where each job has to be processed through a number of stages in series (i.e., unidirectional flow of material) and at any stage there may be more than one machine available.

It has been shown by Gupta (1988) that two-stage hybrid flow shop makespan minimization problem without transfer batches is NP-complete. Gupta (1988) and Gupta and Tunc (1991) present approximate solution algorithms which minimize

makespan in a two-stage hybrid flow shop when there are identical machines at the first and second stage, respectively. Brah and Hunsucker (1991) present a branch and bound algorithm for minimizing the makepan in a M-stage hybrid flow shop. However, in these studies, transfer batches are not considered.

Figure 3.1 depicts three possible configurations of the hybrid flowshop problems considered in this section. The parallel machines could be only at one of the two stages (as in Figures 3.1(a) and 3.1(b)) or at both stages as in Figure 3.1(c). Depending on the shop configuration and the number of parallel machines at each stage, a single machine may process different number of transfer batches.

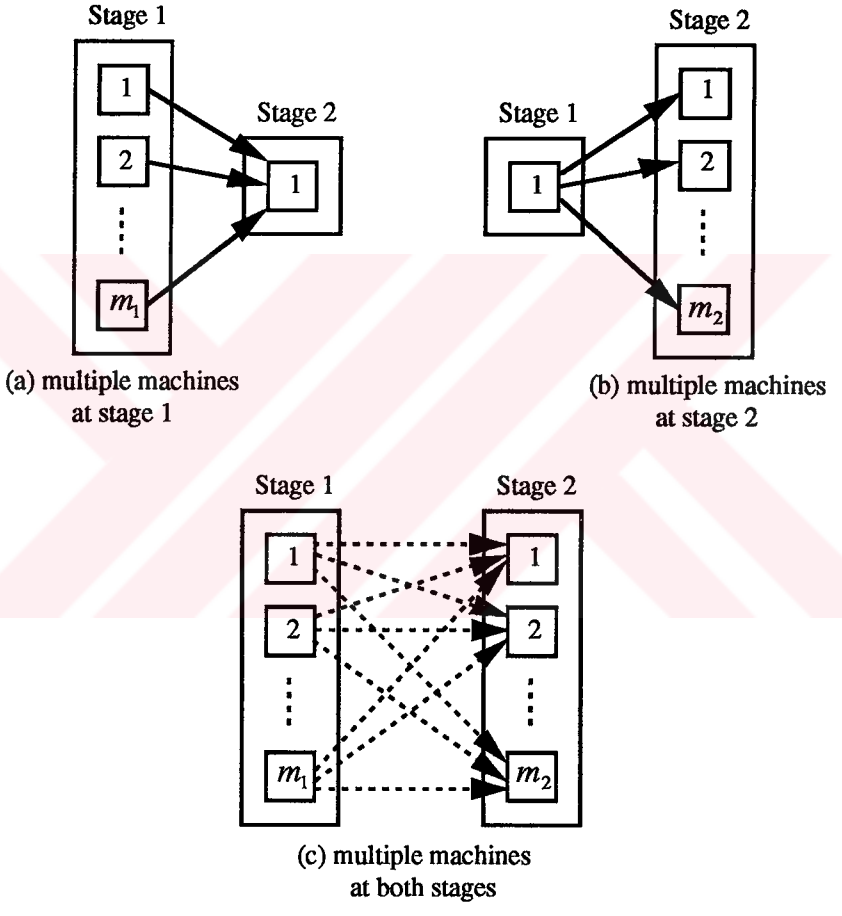


Figure 3.1. Two-stage hybrid flowshop structures.

Using the classification scheme described in Chapter 1, the single-job streaming problem in a two-stage hybrid flowshop with multiple identical parallel machines is denoted by $1|F2P(m_1,m_2)|TB|C_{\max}$ where m_1 and m_2 denote the number of identical parallel machines at stages 1 and 2, respectively.

Before discussing the lot streaming procedures, it is useful to discuss two equivalent problems. We define the *inverse* of $1|F2P(m_1,m_2)|TB|C_{\max}$ problem to be the problem $1|F2P(m_2,m_1)|TB|C_{\max}$ in which processing times of the job at the first stage and second stage are reversed. Then, we state the following result.

Theorem 3.1 *The two-stage hybrid flowshop lot streaming problem and its reverse are equivalent.*

Proof : Consider any feasible schedule σ for $1|F2P(m_1,m_2)|q|C_{\max}$ lot streaming problem. Let $C_{i,1}(\sigma)$ and $C_{i,2}(\sigma)$ denote the completion time of the first and second operations of transfer batch i $(i=1,..,s)$, respectively.

For the inverse problem, construct the schedule σ' in which the first and second operations of transfer batch i $(i=1,..,s)$ are scheduled in the intervals $[C_{\max}-C_{i,2}(\sigma),\ C_{\max}-C_{i,2}(\sigma)+p_2q_i]$ and $[C_{\max}-C_{i,1}(\sigma),\ C_{\max}-C_{i,1}(\sigma)+p_1q_i]$, respectively. The feasibility of σ ensures that σ' is feasible. Moreover, the makespans for these two schedules in their respective problems are identical, i.e., $C_{\max}(\sigma)=C_{\max}(\sigma')$. Similarly, any feasible schedule for $1|F2P(m_2,m_1)|q|C_{\max}$ converts into a feasible schedule for $1|F2P(m_1,m_2)|q|C_{\max}$ with the same makespan. Thus, the two problems are equivalent. ∎

3.1.1 Multiple Machines at Only One Stage

We now consider lot streaming problems in a hybrid flowshop where only one of the two stages contains identical parallel machines. Without loss of generality, we will assume that the multiple machines are at the second stage.

Theorem 3.2 *The optimal transfer batch sizes for the* $1|F2P(1,m_2)|TB|C_{\max}$ *problem are*

$$q_1=(\frac{\alpha-1}{\alpha^s-1})Q,$$

$$q_i=\alpha^{i-1}q_1,\qquad i=2,\ldots,s,$$

where $\alpha=\frac{p_2}{p_1m_2}$, $Q=\sum_{1\le i\le s}q_i$, and the optimal makespan is

$$C_{\max}=p_1q_1+(p_2/m_2)Q.$$

Proof: Let q_i be the size of the ith transfer batch transferred from the first stage to the second one. Then a lower bound LB_1 on $C_{\max}$ based on the first transfer batch in the sequence is given as the sum of

. the processing time of the first transfer batch at stage 1, and

. the earliest completion time of all transfer batches at stage 2, which is achieved by equal allocation of the total work of the entire lot equally among m_2 parallel machines. Thus,

$$C_{\max} \geq LB_1 = p_1 q_1 + p_2(q_1 + \ldots + q_s)/m_2 = p_1 q_1 + p_2 Q/m_2.$$

Similarly, another lower bound based on the second transfer batch processed at stage 1 is given as the sum of the processing time of the first two transfer batches at stage 1 and the earliest completion time of the transfer batches 2 through s at stage 2, i.e.,

$$C_{\max} \geq LB_2 = p_1(q_1 + q_2) + p_2(q_2 + \ldots + q_s)/m_2.$$

Generally, for any transfer batch i, we can write

$$C_{\max} \geq LB_i = p_1 \sum_{k=1}^{i} q_k + (p_2/m_2) \sum_{k=i}^{s} q_k.$$

It follows that

$$C_{\max} \geq \max_{1 \leq i \leq s} \{p_1 \sum_{k=1}^{i} q_k + (p_2/m_2) \sum_{k=i}^{s} q_k\}. \tag{3.1}$$

In the view of above developments, the lot streaming problem can be formulated by the following linear programming model.

$$\begin{aligned}
(P)\quad \text{minimize} \quad & C_{\max} \\
\text{subject to} \quad & C_{\max} \geq p_1 \sum_{k=1}^{i} q_k + (p_2/m_2) \sum_{k=i}^{s} q_k, \qquad i = 1,2,\ldots,s \qquad (3.2) \\
& \sum_{i=1}^{s} q_i = Q \qquad (3.3) \\
& q_i \geq 0, \quad i = 1,2,\ldots,s.
\end{aligned}$$

A feasible schedule to the above linear program is obtained by assuming that each of the inequalities in (3.2) is satisfied as equality, i.e., there is no intermittent idle time between processing of transfer batches on all machines at stage 2. In such a case, the total idle time at stage 2 is $I = p_1 q_1$ and the adjacent pair of transfer batch sizes must satisfy the following relationship

$$p_1 \sum_{k=1}^{i} q_k + (p_2/m_2) \sum_{k=i}^{s} q_k = p_1 \sum_{k=1}^{i-1} q_k + (p_2/m_2) \sum_{k=i-1}^{s} q_k, \qquad i = 2,\ldots,s,$$

or equivalently,

$$q_i = (\tfrac{p_2}{p_1 m_2}) q_{i-1}, \quad i = 2,\ldots,s. \tag{3.3}$$

Then, solving the set of simultaneous equations (3.2) and (3.3) yields

$$q_1 = (\frac{\alpha - 1}{\alpha^s - 1}) Q, \tag{3.4}$$

$$q_i = \alpha^{i-1} q_1, \quad i = 2,\ldots,s, \tag{3.5}$$

$$C_{\max} = p_1 q_1 + (p_2 / m_2) Q \tag{3.6}$$

where $\alpha = \frac{p_2}{p_1 m_2}$.

Hence, we have shown that the solution given by the theorem is feasible. Note that this solution is a solution for the two-stage flow shop lot streaming with a single machine at each stage, where the p_1 is the processing time of stage 1 and $\frac{p_2}{m_2}$ is the pseudo-processing time of stage 2.

Now, to prove the optimality of this solution, we construct the following dual of the problem P as

$$(D) \quad \text{Min} \quad Q\omega$$

$$\text{s. to} \quad p_1 \sum_{k=i}^{s} \lambda_k + (p_2 / m_2) \sum_{k=1}^{i} \lambda_k - \omega \geq 0, \quad i = 1,2,\ldots,s \tag{3.7}$$

$$\sum_{i=1}^{s} \lambda_i = 1 \tag{3.8}$$

$$\lambda_i \geq 0, \quad i = 1,2,\ldots,s$$

ω unrestricted in sign

Now, as before, we find a feasible solution to the problem D by assuming that all constraints defined by (3.7) are satisfied as equalities. It follows that a feasible solution to the dual problem is obtained when

$$\lambda_i = (\tfrac{p_1 m_2}{p_2}) \lambda_{i-1}, \quad i = 2,\ldots,s. \tag{3.9}$$

Then solving the set of simultaneous equations (3.7) and (3.9) yields

$$\lambda_1 = \alpha^{s-1} (\frac{\alpha - 1}{\alpha^s - 1}), \tag{3.10}$$

$$\lambda_i = \alpha^{1-i} \lambda_1, \quad i = 2,\ldots,s, \tag{3.11}$$

$$\omega = p_1(\frac{\alpha-1}{\alpha^s-1}) + \frac{p_2}{m_2} \tag{3.12}$$

where $\alpha = \frac{p_2}{p_1 m_2}$.

The feasible solutions to the problem P and its dual D have the same objective function value. Therefore, from the duality theory, it follows that (3.4) and (3.5) is an optimal solution to the primal problem P and (3.11) and (3.12) is an optimal solution to the dual problem D. Thus, we conclude that the solution given by (3.4) and (3.5) optimally solves $1|F2P(1,m_2)|TB|C_{\max}$ problem. ∎

From Theorems 3.1 and 3.2, we have the following result.

Corollary 3.1 *The optimal transfer batch sizes for the* $1|F2P(m_1,1)|TB|C_{\max}$ *problem are*

$$q_1 = (\frac{\alpha-1}{\alpha^s-1})Q,$$
$$q_i = \alpha^{i-1}q_1, \quad i = 2,\ldots,s,$$

where $\alpha = \frac{p_1}{p_2 m_1}$, *and the optimal makespan is*

$$C_{\max} = (p_1 / m_1)Q + p_2 q_s.$$

3.1.2 Multiple Machines at Both Stages

Now, we consider the single-job streaming problem in a two-stage hybrid flow shop with identical parallel machines at both stages.

Theorem 3.3 *The optimal transfer batch sizes for the* $1|F2P(m_1,m_2)|TB|C_{\max}$ *problem are*

$$q_1 = (\frac{\alpha-1}{\alpha^s-1})Q,$$
$$q_i = \alpha^{i-1}q_1, \quad i = 2,\ldots,s,$$

where $\alpha = \frac{p_2 m_1}{p_1 m_2}$.

Proof: Let q_i be the size of the transfer batch processed and transferred to the second stage in the ith position. Moreover, let $q_{i,l}$ be the portion of q_i which is processed by the lth machine at stage 1. Since there m_1 identical parallel machines exist at the first stage, amount of work for the ith transfer batch can be at most allocated to m_1 identical parallel machines. Then it is clear that $\sum_{1\le l\le m_1} q_{i,l} = q_i$ and $q_{i,l} \ge 0$ for $1 \le l \le m_1$.

The completion time of the ith transfer batch at the first stage is given by

$$C_{i,1} = p_1 \max_{1 \le l \le m_1} \{\sum_{k=1}^{i} q_{k,l}\}, \qquad i = 1,...,s$$

Then a lower bound, based on the ith transfer batch, on the makespan of transfer batches at the second stage is given as the sum of

. the completion time of the ith transfer batch at stage 1, and

. the earliest completion time of the transfer batches i through s at stage 2, which is achieved by equal allocation of the total work for the transfer batches i through s among m_1 identical parallel machines at stage 2. Thus,

$$C_{\max} \ge LB_i = C_i + p_2(q_i + ... + q_s)/m_2 = p_1 \max_{1 \le l \le m_1} \{\sum_{k=1}^{i} q_{k,l}\} + (p_2/m_2)\sum_{k=i}^{s} q_k . \qquad (3.13)$$

Hence, the streaming problem can be formulated by the following linear programming model:

$$(P) \quad \text{Min} \quad C_{\max}$$

$$\text{s. to} \quad C_{\max} \ge p_1 \sum_{k=1}^{i} q_{k,l} + (p_2/m_2)\sum_{k=i}^{s} q_k, \quad i = 1,2,...,s; \; l = 1,2,...,m_1 \qquad (3.14)$$

$$\sum_{l=1}^{m_1} q_{i,l} = q_i, \qquad i = 1,2,...,s \qquad (3.15)$$

$$\sum_{i=1}^{s} q_i = Q \qquad (3.16)$$

$$q_i \ge 0, \;\; q_{i,l} \ge 0, \;\; i = 1,2,...,s, \;\; l = 1,2,...,m_1.$$

A feasible solution to the above model is obtained by assuming that each of the inequality in (3.14) is satisfied as equality. In this case, $q_{i,l} = q_i / m_1$ for all $1 \le i \le s$, $1 \le l \le m_1$, i.e., the processing work of each transfer batch is equally allocated to the parallel machines at stage 1. This result is also valid for the processing of transfer batches at the second stage. Thus, the transfer batch sizes are

$$q_1 = (\frac{\alpha - 1}{\alpha^s - 1})Q, \qquad (3.17)$$

$$q_i = \alpha^{i-1} q_1, \qquad i = 2,...,s, \qquad (3.18)$$

where $\alpha = \frac{p_2 m_1}{p_1 m_2}$.

Using the analysis similar to that in the earlier case examined in Theorem 3.2, it can be shown that this solution is an optimal solution to the problem $1|F2P(m_1,m_2)|TB|C_{\max}$. ∎

From Theorems 3.2 and 3.3, and Corollary 3.1, we have the following result.

Corollary 3.2 *In the optimal solution of the* $1|F2P(m_1,m_2)|TB|C_{\max}$ *problem,*

. every machine at a stage is used to process all transfer batches,

. there is no intermittent idle time between processing of the transfer batches at both stages,

. no transfer batch waits for processing between the first and second stage.

Corollary 3.2 above shows that the concepts and results related to critical machines and critical transfer batches used by Glass *et al.* (1994) also extends to the hybrid flowshop lot streaming problems. Thus, we can state that in the optimal solution of the $1|F2P(m_1,m_2)|TB|C_{\max}$ problem, every transfer batch is critical.

Another interesting result is given by the following corollary.

Corollary 3.3 *Two-stage hybrid flow shop lot streaming problem* $1|F2P(m_1,m_2)|TB|C_{\max}$ *reduces to the two-stage flow shop lot streaming problem* $1|F2|TB|C_{\max}$ *in which the processing time at stage* j *is* $\frac{p_j}{m_j}$.

To illustrate the results above, consider the following example problem.

Example 3.1

Let $p_1 = 1$, $p_2 = 3$, $Q = 60$, and $s = 4$.

Case a. Let $m_1 = 1$, $m_2 = 3$.

In this case, the optimal transfer batches are equal each other since $\alpha = \frac{p_2}{p_1 m_2} = \frac{3}{(1)(3)} = 1$, i.e., $q_i = Q/s = 15$ for $1 \le i \le 4$, and the corresponding optimal makespan is 75 (see Figure 3.2).

Case b. Let $m_1 = 2$, $m_2 = 3$.

In this case, $\alpha = \frac{p_2 m_1}{p_1 m_2} = \frac{(3)(2)}{(1)(3)} = 2$. Thus, the optimal transfer batch sizes are $q_1 = (\alpha - 1)Q/(\alpha^s - 1) = 4$, $q_2 = 8$, $q_3 = 16$, and $q_4 = 32$, respectively, and the corresponding optimal makespan is 62 (see Figure 3.3).

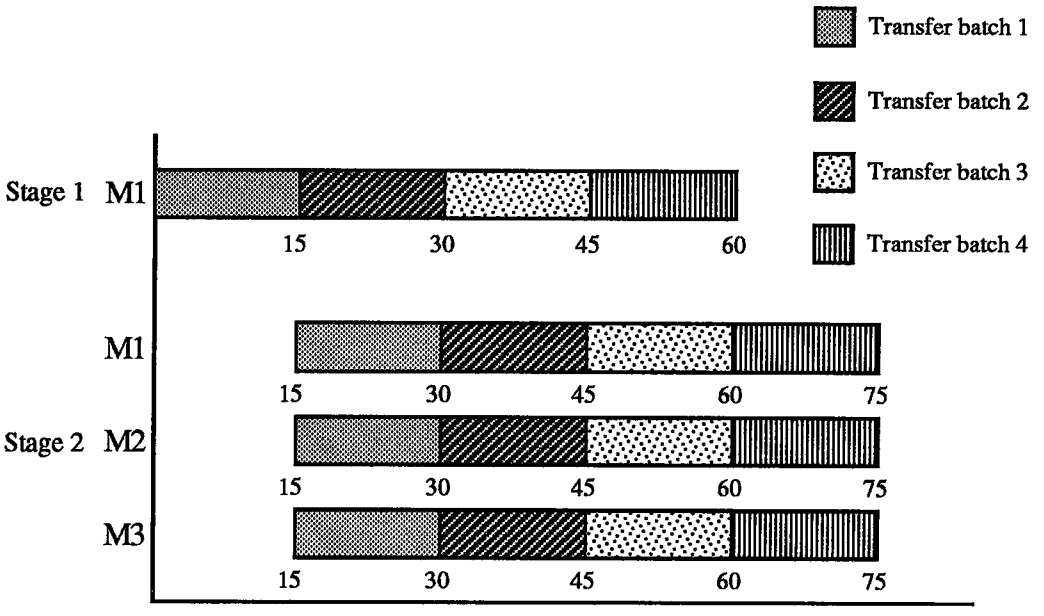


Figure 3.2. Optimal schedule in case (a) of Example 3.1.

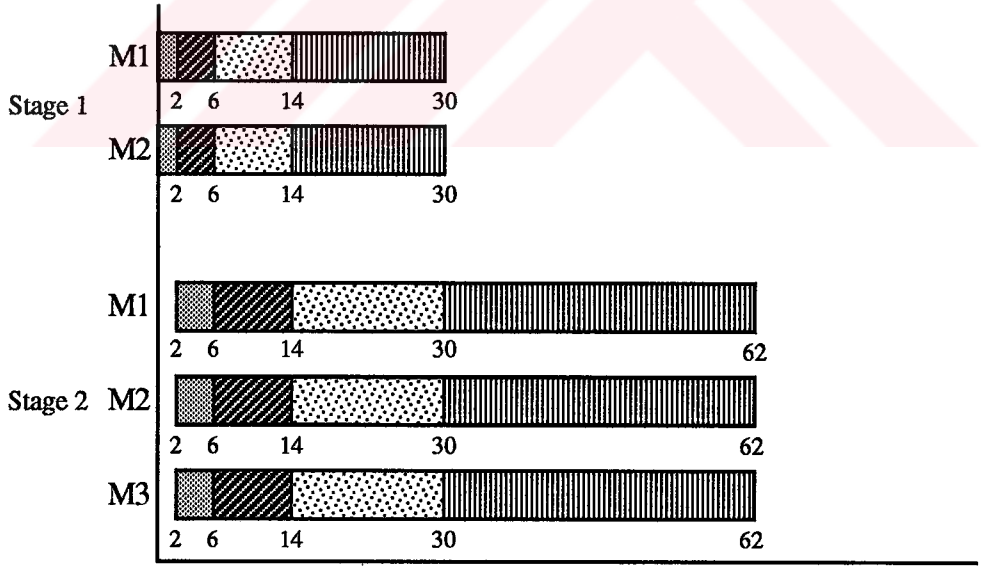


Figure 3.3. Optimal schedule in case (b) of Example 3.1.

3.2 M-machine Flow Shop Problem with Variable Transfer Batches

As mentioned in Chapter 2, the mathematical formulation for the $1|F|TB(V,c)|C_{max}$ problem which was given by Benli (1994) is a mixed integer programming model with linear constraints and objective function, and requires $4M^2(s-1)+M(s-2)$ continuous variables, $M^2(s-1)+M$ binary variables and $4M^2(s-1)+5M$ constraints where M is the number of machines and s is the number of transfer batches. For a small sized problem with 5 machines and 5 transfer batches between each pairs of transfer batches, it is clear that we need to solve a mixed integer model with 415 continuous variables, 105 binary variables and 425 constraints. Thus, this model can not be used efficiently to solve the problem. However, the properties of the three-machine flow shop streaming problem with s variable transfer batches derived by Trietsch and Baker (1993) and method of solving the M-machine flow shop streaming problem with two consistent transfer batches proposed by Williams and Tufekci (1992) can be used to devise a solution procedure without solving a mathematical programming model.

In Chapter 2, while discussing the three-machine flow shop streaming problem with s variable transfer batches, we mentioned that the set of machines operating continuously without idle time between processing of transfer batches is defined as the *partition set*. With respect to the processing times on the machines, there are two partition sets, first one containing machines 1 and 3, and the second containing all machines. Note that the first and last machines are always in the partition sets since any feasible makespan can be achieved by scheduling the first and last machines to operate continuously. With variable transfer batches, there is an optimal solution that is a no-wait schedule with consistent transfer batches among each pair of machines in the partition set. If the first partition set {1,3} is optimal, then the optimal solution of the problem with consistent transfer batches is also optimal for the variable transfer batch case. On the other hand, if the second partition set {1,2,3} is optimal, then the optimal solution for the problem with variable transfer batches decomposes into two subproblems. The transfer batches between machine 1 and 2 are determined by the optimal solution of the first two-machine problem with consistent transfer batches. In the second subproblem, the transfer batches between machine 2 and 3 are determined by the optimal solution of the last two-machine problem with consistent transfer batches.

These properties lead us to obtain a solution for our problem in which there exists an optimal partition set, and all transfer batches between consecutive pairs of machines within the partition set are consistent. Thus, the problem is to simultaneously find the optimal partition set and the corresponding transfer batches.

Theorem 3.4 *The optimal solution to the problem* $1|F|TB(V,c)|C_{\max}$ *can be found in* $O(sM^2)$ *time.*

Proof: Let $\{1,...,j,k,...,M\}$ be the optimal partition set in which j and k are the indices of the two machines working continuously such that $j<k$. Also, let $q_{i,j}$ be the size of the ith transfer batch which is consistent over machine j through k.

To have a no-wait schedule between machines j and k working continuously, the following must hold

$$(\sum_{m=j+1}^{k} p_m)\, q_{i,j} = (\sum_{m=j}^{k-1} p_m)\, q_{i+1,j}, \qquad \text{for } i=1,...,s-1 \tag{3.19}$$

where p_m is the unit processing time on machine m.

From equation (3.19), let

$$z = \frac{q_{i,j}}{q_{i+1,j}} = \frac{P(j,k-1)}{P(j+1,k)}, \qquad \text{for } i=1,...,s-1 \tag{3.20}$$

where $P(j,k) = (\sum_{m=j}^{k} p_m)$. Also, note that

$$\sum_{i=1}^{s} q_{i,j} = Q. \tag{3.21}$$

Using (3.20) and (3.21), it is easy to verify that

$$q_{i,j} = (\frac{z^{s-i}}{1+z+z^2+...+z^{s-1}})Q, \qquad \text{for } i=1,...,s. \tag{3.22}$$

From (3.22), it is obvious that $q_{1,j}$ increases as z is increased since $\frac{\partial q_{1,j}}{\partial z} > 0$ for all $z>0$. Thus, in order to minimize $q_{1,j}$, the following must be true.

$$z = \min_{j<m\le M}\left\{\frac{P(j,m-1)}{P(j+1,m)}\right\} = \frac{P(j,k-1)}{P(j+1,k)}. \tag{3.23}$$

From (3.23), it is easy to see that the time complexity to determine the machines in the partition set is $O(M^2)$, however, time complexity to determine the optimal

consistent transfer batch sizes for every pair of machines in the partition set is $O(s)$. Thus, the problem can be solved optimally in $O(sM^2)$. ∎

Based on the results above, a polynomial time algorithm can be given to solve the problem $1|F|TB(V,c)|C_{\max}$ optimally.

Algorithm 3.1

Step 1 Let the first machine be the initial machine in the partition, i.e., set $j=1$. Also let $I=0$.

Step 2 (i) Let $$z=\min_{j<m\le M}\left\{\frac{P(j,m-1)}{P(j+1,m)}\right\}=\frac{P(j,k-1)}{P(j+1,k)}$$

where $P(j,k)=(\sum_{m=j}^{k} p_m)$ and k, $k>j$, is the largest index for which the equation above is true. Then, machine k is the next machine in the partition set after machine j.

(ii) The optimal transfer batch sizes which are consistent from machine j through k are

$$q_{i,j}=(\frac{z^{s-i}}{1+z+z^2+\ldots+z^{s-1}})Q, \qquad \text{for } i=1,\ldots,s.$$

(iii) Let $I=I+q_{1,j}\,P(j,k-1)$.

Step 3 If $k<M$, set $j=k$ and go to Step 2; otherwise, the optimal makespan is $C_{\max}=I+p_M Q$, and stop.

Example 3.2

To illustrate the algorithm above, consider a four-machine flow shop problem in which unit processing times on the machines are 1, 1, 3, and 3, respectively. Suppose a lot of 300 units will be processed in the shop, three transfer batches are allowed between machines.

In Step 1, we set $j=1$, i.e., machine 1 is the first feasible machine in the partition set. Also, let $I=0$. Then, in Step 2,

$$z=\min\left\{\frac{p_1}{p_2},\ \frac{p_1+p_2}{p_2+p_3},\ \frac{p_1+p_2+p_3}{p_2+p_3+p_4}\right\}=\min\left\{\frac{1}{1},\ \frac{1+1}{1+3},\ \frac{1+1+3}{1+3+3}\right\}$$

$$= \min\left\{1, \frac{1}{2}, \frac{5}{7}\right\} = \frac{1}{2}$$

corresponds to machine 3 which is the next feasible machine in the partition set, i.e., $k = 3$. Also, the transfer batch sizes from machine 1 through 3 are

$$q_{1,1} = \left(\frac{z^3}{1+z+z^2+z^3}\right)Q = 20$$
$$q_{2,1} = \left(\frac{z^2}{1+z+z^2+z^3}\right)Q = 40$$
$$q_{3,1} = \left(\frac{z}{1+z+z^2+z^3}\right)Q = 80$$
$$q_{4,1} = \left(\frac{1}{1+z+z^2+z^3}\right)Q = 160.$$

Then the idle time on machine 3 before the processing of the first transfer batch is $I = I + (p_1 + p_2)q_{1,3} = 0 + 2(20) = 40$. Since $k = 3 < M = 4$, we set $j = k = 3$ and repeat Step 2. Then, we have

$$z = \left\{\frac{p_3}{p_4}\right\} = \left\{\frac{1}{1}\right\} = 1$$

corresponds to machine 4 which is the next feasible machine in the partition set, i.e., $j = 4$. Also, the transfer batch sizes from machine 3 through 4 are

$$q_{1,3} = \left(\frac{z^3}{1+z+z^2+z^3}\right)Q = \left(\frac{1}{4}\right)300 = 75$$
$$q_{2,3} = 75$$
$$q_{3,3} = 75$$
$$q_{4,3} = 75.$$

Then the idle time on machine 4 before the processing of the first transfer batch is $I = I + (p_3)q_{3,3} = 40 + 3(75) = 265$. Since $k = M = 4$, we have the optimal partition set and the corresponding transfer batch sizes with a makespan of $C_{\max} = I + p_M Q = 1165$ as shown in Figure 3.4.

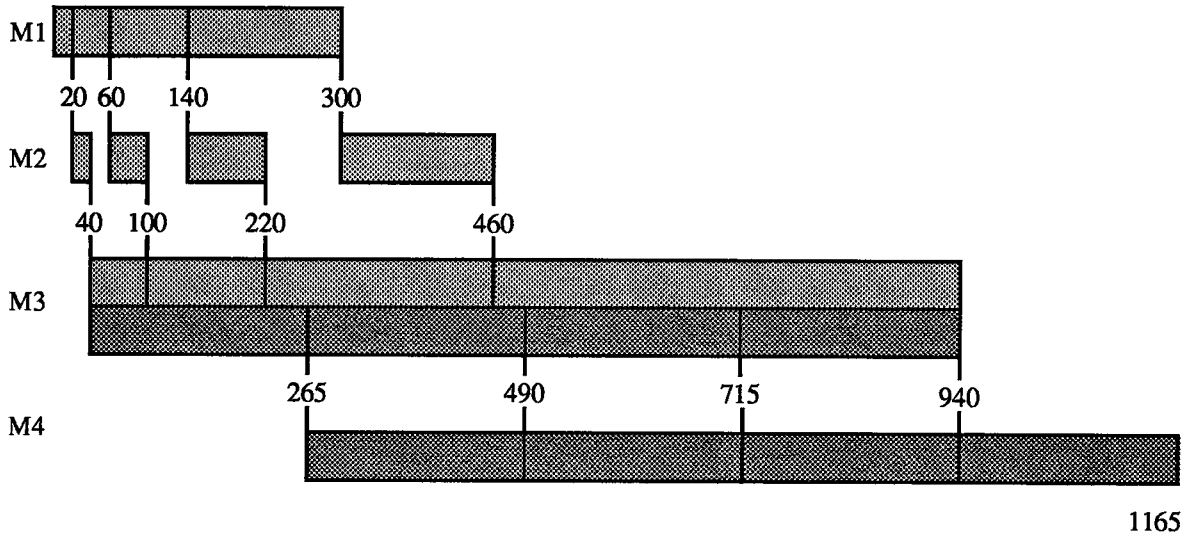


Figure 3.4. Optimal schedule in Example 3.2.

3.3 Three-machine Flow Shop Problem with Setup Times and Variable Transfer Batches

In this section, single-job streaming problem $1|F3|TB(V,c),S_d|C_{\max}$ with variable transfer batches in a three-machine flow shop in which detached setup times exist before processing the whole lot will be studied. The following lemmas can further simplify the problem.

Lemma 3.1 *Let $C'_{\max}$ be the optimal makespan for the alternative lot streaming problem (P') in which the setup on any machine m is replaced by $S'_m = \max\{S_m - S_1, 0\}$ (for all $m = 1,...,M$). Then $C_{\max} = S_1 + C'_{\max}$, and the optimal transfer batches for the alternative problem are also optimal for the original problem.*

Proof: Assume that the setups on all machines start at time zero. If the setup time on machine m $(m = 2,...,M)$ is greater than the setup time on the first machine, i.e., $S_m > S_1$, then the machine m will be busy for the setup during the interval $[0,S_1]$ in which the first machine is set up, and its setup will continue during the interval $[S_1, S_m - S_1]$. Let S'_m be the *reduced* setup time required on machine m after the interval $[0,S_1]$. Thus, $S'_m = S_m - S_1$.

However, if the setup on machine m is less or equal to the setup time on the first machine, i.e., $S_m \le S_1$, then there won't be a setup on machine m after the interval $[0,S_1]$. Thus, $S'_m = 0$. Hence, it is clear that $S'_m = \max\{S_m - S_1, 0\}$ for a machine m

$(m = 2,...,M)$. Thus, without loss of generality, we can say that our lot streaming problem is equivalent to an alternative lot streaming problem in which the processing on the first machine starts at time S_1 and the setup times on machine m is $S'_m = \max\{S_m - S_1, 0\}$. Hence, $C_{\max} = S_1 + C'_{\max}$. ∎

As an immediate result of Lemma 3.1, we have the following corollary.

Corollary 3.4 *If $S'_m = 0$ for every machine m in the alternative lot streaming problem (P'), then the problem (P') as well as the original problem is equivalent to the relaxed problem without any setups.*

Lemma 3.2 *Let $S'_h = \max\{S'_m : 1 \le m \le M\} \ne 0$ be the reduced setup time of the machine h having the smallest machine index. Then, in the optimal schedule of the alternative lot streaming problem (P'), the setup on the machine m for $m = h+1,...,M$ is not critical, i.e., it is sufficient to consider only the setups on the machines 1 through h.*

Proof: Proof is trivial. By definition, since $S'_h \ge S'_m$ for $m = h+1,...,M$, the setup on machine m is not completed later than setup on the machine h is finished. Thus, the setups on the machines succeeding machine h do not have any affect on the start of processing the first transfer batch on these machines. Therefore, they are not critical. ∎

Lemma 3.3 *Let $C''_{\max}$ be the optimal makespan for the alternative lot streaming problem (P'') in which the reduced setup time on the last machine M, S'_m is replaced by zero. Then*

$$C'_{\max} = \max\{C''_{\max}, S'_M + p_M Q\}.$$

Proof: Let I_M be the total idle time on the last machine in the optimal schedule for problem (P''). Without loss of generality, we can assume that all transfer batches transferred to the last machine are processed on this machine as late as possible without increasing the makespan of the optimal schedule for problem (P''). Thus, I_M represents the latest start time for processing the first transfer batch on the last machine. Then, $C''_{\max} = I_M + p_m Q$.

If $S'_M \le I_M$, then S'_m time units of setup on the last machine is finished before the first transfer batch starts its processing on that machine, and thus

$C'_{\max} = C''_{\max} = I_M + p_M Q$. Hence, we can use the optimal transfer batch sizes obtained for problem (P'') as the optimal transfer batch sizes to the problem (P').

However, if $S'_M > I_M$, then we can use the optimal schedule for problem (P'') for the problem (P'), and, in this case, $C'_{\max} = C''_{\max} = S'_M + p_M Q$.

Thus the optimal makespan for the problem (P') is $C'_{\max} = \max\{C''_{\max}, S'_M + p_M Q\}$.

Corollary 3.5 *The original three-machine problem with setups is equivalent* either *to an alternative problem* (P'') *with no setup time on machines 1 and 3 and setup time* $S'_2 = S_2 - S_1 > 0$ *on machine 2* or *to the relaxed problem without any setups.*

Now, we restrict our attention to first case in Corollary 3.5, i.e., the case in which $S'_1 = S'_3 = 0$ and $S'_2 > 0$.

Remark 3.1 In the alternative problem (P''), if $S'_2 > p_1 Q$ then it is clear that there is no need for lot streaming between machines 1 and 2., i.e., the whole lot can be transferred to the second machine in a single lot. However, allowing transfer batches between machines 2 and 3 will result in a reduced makespan. Then the second machine works continuously, and the alternative problem (P'') reduces to a two-machine lot streaming problem (on machines 2 and 3) with a setup time S'_2 on machine 2 and zero on machine 3.

Then we have the following result.

Theorem 3.5 *Let* $q^R_{1,1}$ *be the optimal size of the first transfer batch from machine 1 to machine 2 in the relaxed problem without any setups. In the alternative problem* (P''), *if* $S'_2 \le p_1 q^R_{1,1}$, *then the optimal transfer batch sizes between machines in the relaxed problem are also optimal for the problem* (P'').

Proof: If $S'_2 \le p_1 q^R_{1,1}$, then the reduced setup on machine 2, S'_2, is not critical in the optimal solution. That is, the start of the first transfer batch processing on machine 2 is not affected by the reduced setup on machine 2. There is sufficient time to complete the setup before the start of processing on machine 2. Thus, the optimal transfer batch sizes between machines in the relaxed problem are also optimal for the problem (P'').

On the other hand, the alternate problem (P'') may really be different from the relaxed problem if $S_2' > p_1 q_{1,1}^R$. In this case, similar to the relaxed problem, we may distinguish two cases, depending on whether $(p_2)^2 \geq p_1 p_3$ or $(p_2)^2 < p_1 p_3$.

Case 1 $(p_2)^2 \geq p_1 p_3$ and $S_2' > p_1 q_{1,1}^R$.

From the discussion in Chapter 2, we know that, in the optimal solution of the three-machine relaxed problem, the problem decomposes into two subproblems: a subproblem for machines 1 and 2, which determines the transfers from machine 1 to machine 2, and another subproblem for machines 2 and 3, which determines the transfers from machine 2 to machine 3. Between machines 1 and 2, the transfer batch sizes are determined by solving the two-machine problem with only processing times on machines 1 and 2. Between machines 2 and 3, the transfer batch sizes are determined by solving the two machine problem with processing times on machines 2 and 3. Furthermore, in the optimal structure of the schedule obtained in this way, we observe that all machines work continuously, i.e., there is no idle time between processing of transfer batches on each machine. Based on these observations, we have the following result.

Theorem 3.6 *In the alternative problem* (P''), *if* $(p_2)^2 \geq p_1 p_3$ *and* $S_2' > p_1 q_{1,1}^R$, *then the optimal variable transfer batch sizes between machines in the relaxed problem are also optimal for the problem* (P'').

Proof: Let I_2 be the idle time before machine 2 starts processing in the relaxed problem. Note that $I_2 = p_1 q_{1,1}^R$. Since $S_2' > p_1 q_{1,1}^R$, then $S_2' > I_2$ which implies that start of the continuous processing of the second machine is delayed by $S_2' - I_2$ time units. Thus, the transfer batch sizes determined in the first subproblem of the relaxed problem are still optimal for our problem (P''), and transfer batch sizes determined in the second subproblem does not change with the increase in the start of second machine, thus, they are still optimal for our problem (P''). ∎

Case 2 $(p_2)^2 < p_1 p_3$ and $S_2' > p_1 q_{1,1}^R$.

Let us first give the following lemma which will be used to distinguish the possible cases in the following developments.

Lemma 3.4 *Let $q_{1,1}^T$ be the optimal size of the first transfer batch from machine 1 to machine 2 when the first two machines are only considered without any setups. If $(p_2)^2 < p_1 p_3$, then $q_{1,1}^R < q_{1,1}^T$.*

Proof: In Chapter 2, we have seen that the optimal size of the first transfer batch in a two-machine streaming problem without any setups is given as follows:

$$q_{1,1}^T = (\frac{1}{1+\alpha+\alpha^2+...+\alpha^{s-1}})Q \tag{3.24}$$

where $\alpha = p_2 / p_1$, and the optimal size of the first transfer batch in the three-machine relaxed problem for $(p_2)^2 < p_1 p_3$ without any setups is given by

$$q_{1,1}^R = (\frac{1}{1+\beta+\beta^2+...+\beta^{s-1}})Q \tag{3.25}$$

where $\beta = (p_2 + p_3) / (p_1 + p_2)$.

Then we need to verify whether

$$\frac{1}{1+\beta+\beta^2+...+\beta^{s-1}} < \frac{1}{1+\alpha+\alpha^2+...+\alpha^{s-1}}$$

is true. Equivalently, we can show

$$(\beta-\alpha)+(\beta^2-\alpha^2)+...+(\beta^{s-1}-\alpha^{s-1}) > 0. \tag{3.26}$$

Now, let us take the difference between $\beta = (p_2 + p_3) / (p_1 + p_2)$ and $\alpha = p_2 / p_1$. Hence,

$$\beta - \alpha = \frac{p_2 + p_3}{p_1 + p_2} - \frac{p_2}{p_1} = \frac{p_1 p_3 - (p_2)^2}{p_1(p_1 + p_2)} > 0 \tag{3.27}$$

since $(p_2)^2 < p_1 p_3$. Thus, $\beta > \alpha$, which implies that inequality (3.26) is true. Hence, our assumption $q_{1,1}^R < q_{1,1}^T$ is must hold. ∎

Using the result of Lemma 3.4, we may distinguish two subcases:

Subcase 1: $p_1 q_{1,1}^R < p_1 q_{1,1}^T \le S_2'$.

Theorem 3.7 *In the alternative problem (P''), if $(p_2)^2 < p_1 p_3$ and $p_1 q_{1,1}^R < p_1 q_{1,1}^T \le S_2'$, then the optimal variable transfer batch sizes between machines in the relaxed problem are also optimal for the problem (P'').*

Proof : As we have seen in Chapter 2, in the optimal structure of the two-machine streaming problem without any setups, there is no idle time between processing of transfer batches on the second machine. Assume that we use the optimal transfer batch sizes of the two-machine streaming problem without any setups for transfers between machines 1 and 2. If $p_1 q_{1,1}^T \le S_2'$, then the start of the continuous processing of the second machine is delayed by $S_2' - p_1 q_{1,1}^T$ time units, and there will not be idle time between processing of transfer batches. Thus, the transfer batch sizes between machines 1 and 2 in the two-machine problem are still optimal for our problem (P''). Since second machine operates continuously, between machines 2 and 3, the transfer batch sizes are determined by solving the two-machine problem with processing times on machines 2 and 3. Furthermore, in the optimal structure of the schedule obtained in this way, we observe that all machines work continuously. ∎

Subcase 2: $p_1 q_{1,1}^R < S_2' < p_1 q_{1,1}^T$.

Unfortunately, we have not able to obtain a polynomial time solution procedure for this subcase. Thus, it is an open problem for the future research.

Example 3.3

Let $p_1 = 3$, $p_2 = 1$, $p_3 = 1$, $Q = 6$ and $s = 2$. Then it is clear that $(p_2)^2 < p_1 p_3$. Using equation (3.25), the optimal solution for the relaxed problem without any setups is obtained with consistent transfer batches, and the transfer batch sizes are $q_{1,1}^R = q_{1,2}^R = (\frac{p_1+p_2}{p_1+2p_2+p_3})Q = 4$ and $q_{2,1}^R = q_{2,2}^R = Q - q_{1,1}^R = 2$ with a makespan of $C_{\max} = (p_1 + p_2)q_{1,1}^R + p_3 Q = 22$.

Case a. Let $S_2' = 10$.
In this case, we use Theorem 3.6 since $S_2' = 10 < p_1 q_{1,1}^R = 12$. Then the consistent transfer batch solution is optimal (see Figure 3.5).

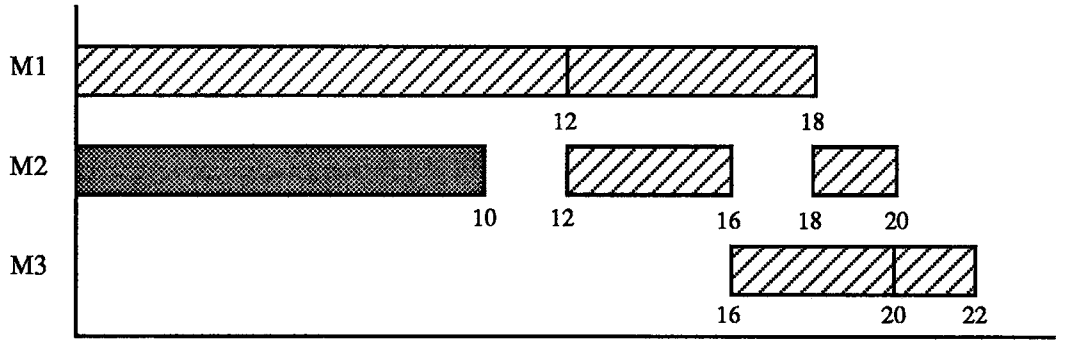


Figure 3.5. Optimal schedule in case(a) of Example 3.3.

Case b. Let $S_2' = 13.5$.

In this case, the solution obtained for Case a. is no longer optimal since $S_2' = 13.5 > p_1 q_{1,1}^R = 12$. Using equation (3.24), the optimal size of the first transfer batch from machine 1 to machine 2 is obtained as $q_{1,1}^T = (\frac{p_1}{p_1+p_2})Q = 4.5$ when the first two machines are only considered. Since $p_1 q_{1,1}^T = S_2' = 13.5$, the problem decomposes into a pair of 2-machine problems according to Theorem 3.8. Between machines 1 and 2, the transfer batches are determined by solving the 2-machine problem with processing times p_1 and p_2. The optimal transfer batch sizes are $q_{1,1} = 4.5$ and $q_{2,1} = 1.5$. Between machines 2 and 3, transfer batches are determined by solving the 2-machine problem with processing times p_2 and p_3. Thus, the optimal transfer batch sizes are equal, i.e., $q_{1,2} = q_{2,2} = 4.5$. The corresponding makespan is $C_{\max} = S_2' + p_2 q_{1,2} + p_3 Q = 22.5$ (see Figure 3.6).

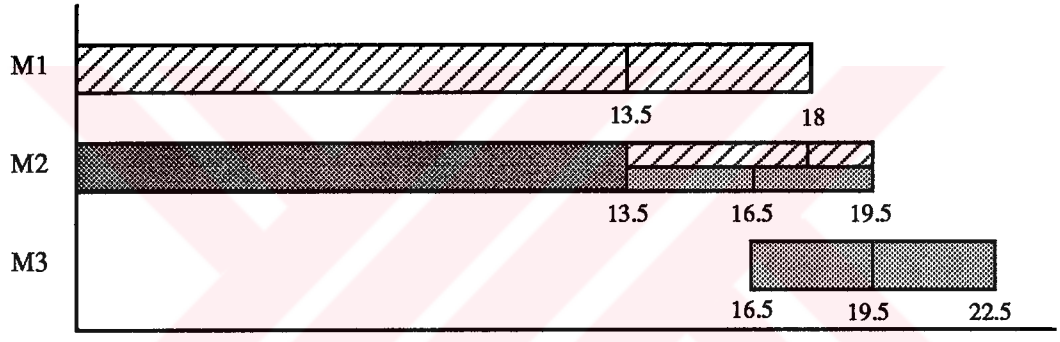


Figure 3.6. Optimal schedule in case (b) of Example 3.3.

3.4 M-machine Open Shop Problem

In this section, we will prove that the single-job streaming problem in an M-machine open shop is NP-hard if $3 \le s < M$ where s is the number of continuous sized transfer batches. Using our classification scheme, this problem is denoted by $1|O|TB(3 \le s < M, c)|C_{\max}$. Before giving the theorem, let us discuss an important characteristic of the open shops which will be used in the following theorem.

Let us assume that we have an M-machine N-job open shop makespan minimization problem without transfer batches, in which the processing time of job j on machine m is $P_{j,m}$. Since there is no specified machine order in open shops, there is one-to-one correspondence between schedules (thus makespans) for this original

problem and the problem in which the machines are interpreted as jobs, and jobs as machines (Glass *et al.*, 1994). Thus, we obtain an equivalent problem of N-machine M-job open shop makespan minimization problem in which the processing time of job m on machine j is $P_{j,m}$. This "duality" is illustrated in Figure 3.7. The first Gantt chart represents a 2-machine 4-job schedule, while the second one represents 4-machines 2-job schedule from the jobs point of view. The corresponding makespan values in each case are the same.

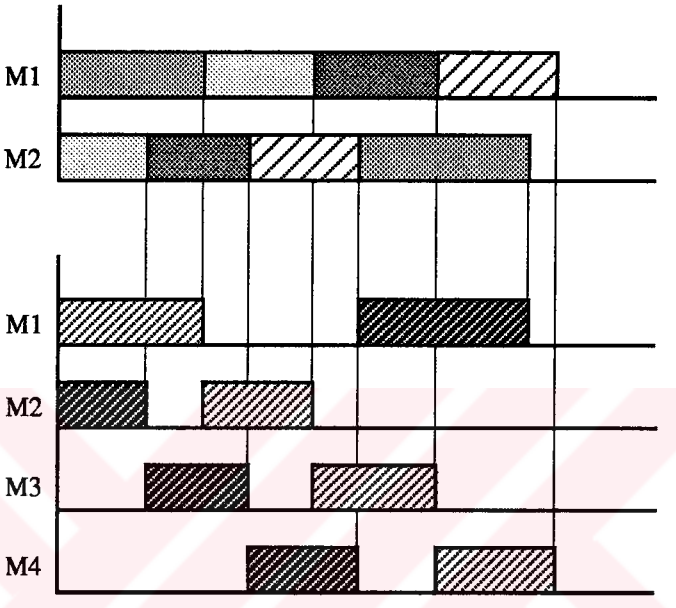


Figure 3.7. Duality in open shop makespan minimization problem

Theorem 3.8 *The* $1|O|TB(3 \le s < M, c)|C_{\max}$ *problem is NP-hard.*

Proof: Assume that consistent transfer batches of equal sizes are specified. Let $P_{i,m}$ be the processing time of transfer batch i on machine m. It is clear that $P_{i,m} = P_m$ for $i = 1,...,s$ since transfer batches are of equal size. We first examine the special case with $s = 3$ transfer batches of this problem denoted by (P). Using the property of open shop makespan minimization problem, we may create the dual of this special case as 3-machine M-sublot problem (denoted by (D)) in which $P_{i,m} = P_i$ for $m = 1,...,3$. It is clear that the problems (P) and (D) have the same makespan value.

To prove that problem (D) is *NP*-hard we make use of the following *NP*-complete problem called PARTITION (Garey and Johnson, 1979):

A multiset $S=\{A_1,A_2,...,A_M\}$ of given positive integers, A_i, $1\le i\le M$ and $\sum_{A_i\in S}A_i=2B$ is said to have a PARTITION if there exists a subset $U\subset\{1,2,...,M\}$ such that $\sum_{i\in U}A_i=B$.

We prove that if our problem (D) is polynomially solvable, then so is PARTITION. From the PARTITION problem for $S=\{A_1,A_2,...,A_M\}$, we construct the following instance of the open shop problem with $M+1$ transfer batches, 3-machines and $P_{i,m}=P_i$ for $1\le i\le M+1$, $1\le m\le 3$. The processing times of transfer batches P_i are defined as follows: $P_i=A_i$, $1\le i\le M$ and $P_{M+1}=B$.

Now we show that the above open shop problem has a schedule with $C_{\max}=3B$ if and only if S has a partition.

(a) If S has a partition U, then there is a schedule with $C_{\max}=3B$. Such a schedule is shown in Figure 3.8.

(b) If S has no partition, then all schedules for our open shop problem must have a completion time greater than $3B$. This is shown by contradiction. Assume that there is a schedule for the open shop problem with $C_{\max}=3B$. Since no transfer batch can be processed simultaneously on two machines, the transfer batch $M+1$ has to be processed at all times (somewhere on one machine). Given that the schedule is non preemptive, there must be a machine m on which the transfer batch $M+1$ starts processing at time B and is completed at time $2B$. Since we assume that $C_{\max}=3B$, on machine m a subset of transfer batches has to be processed between 0 and time B and the rest of transfer batches are processed from $2B$ till $3B$ on machine m. But this would constitute a partition for S.

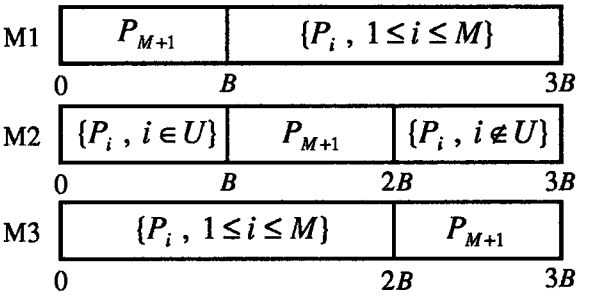


Figure 3.8. Optimal schedule when S has a partition.

Clearly, the above result can be generalized for the cases in which $s\ge 4$, thus the M-machine open shop makespan minimization problem with equal sized transfer batches is NP-hard if $3\le s<M$, i.e., no polynomial time solution exists for the

problem. To complete this complexity analysis we note that the problem with equal transfer batches is a special case of the problem with consistent transfer batches unrestricted in size. Therefore, the problem is also *NP*-hard for the consistent transfer batches case as well. ∎

Table 3.1 below summarizes the solution status of single-job streaming problems with the makespan criterion which exist in the literature along with the problems investigated in this section.

TABLE 3.1. Summary of the solution status of single-job streaming problems with the makespan criterion

Problem	Complexity†	Reference
$1\|F2\|TB(C \text{ or } V,c)\|C_{\max}$	$O(s)$	Baker (1988), Potts and Baker (1989)
$1\|F2\|TB(C \text{ or } V,d)\|C_{\max}$	$O(s^2Q)$	Trietsch (1989)
	$O(s\log Q)$	Vickson (1994)
$1\|F2\|TB(C \text{ or } V,c),NI\|C_{\max}$	$O(s)$	Baker (1988), Potts and Baker (1989)
$1\|F2\|TB(C \text{ or } V,c),NW\|C_{\max}$	$O(s)$	Baker (1988), Potts and Baker (1989)
$1\|F2\|TB(C,c),S_d \text{ or } S_d\|C_{\max}$	$O(s)$	Vickson (1994)
$1\|F2P(m_1,m_2)\|TB(C,c)\|C_{\max}$	$O(s)$	Section 3.1 of this thesis
$1\|F3\|TB(C,c),p_2^2 \le p_1p_3\|C_{\max}$	$O(s)$	Glass *et al.* (1994)
$1\|F3\|TB(C,c),p_2^2 > p_1p_3\|C_{\max}$	$O(\log s)$	Glass *et al.* (1994)
$1\|F3\|TB(V,c)\|C_{\max}$	$O(s)$	Trietsch and Baker (1993)
$1\|F3\|TB(V,d)\|C_{\max}$	$O(s^2Q)$	Trietsch and Baker (1993)
$1\|F3\|TB(V,c),S_d\|C_{\max}$	$O(s)$	Section 3.3 of this thesis
$1\|F\|TB(C,c \text{ or } d)\|C_{\max}$	*open*	
$1\|F\|TB(s=2,C,c)\|C_{\max}$	$O(M^2)$	Baker and Pyke (1990), Williams and Tüfekçi (1992)
$1\|F\|TB(V,c)\|C_{\max}$	$O(sM^2)$	Section 3.2 of this thesis
$1\|F\|TB(V,c),NI\|C_{\max}$	$O(sM)$	Trietsch and Baker (1993)
$1\|F\|TB(V,d),NI\|C_{\max}$	$O(s^2QM)$	Trietsch and Baker (1993)
$1\|O\|TB(s \ge M,c)\|C_{\max}$	$O(sM)$	Glass *et al.* (1994)
$1\|O\|TB(s=2,c)\|C_{\max}$	$O(s)$	Glass *et al.* (1994)
$1\|O\|TB(3 \le s < M,c)\|C_{\max}$	*NP*	Section 3.4 of this thesis

† s: number of transfer batches, M: number of machines, Q: lot size, *NP* : *NP*-hard, *open*: Complexity unknown

CHAPTER 4

SINGLE-JOB STREAMING PROBLEMS WITH THE MEAN FLOW TIME CRITERION

In solving the single-job streaming problems, the measure of performance is often taken as makespan. However, a better measure of performance may well be the minimization *mean (or total) flow time* criterion in a shop, defined as the mean of the completion time of the last operations of each transfer batch (Kropp and Smunt, 1990). The choice of the mean flow time criterion is easily justified, as stated by Dobson *et al.* (1984). This criterion is an effective surrogate for manufacturing lead times; it minimizes work-in-process inventories, enables the scheduler to respond better to disruptions, reduces the safety stocks required by downstream operations. The following results presented in this section are partially reported in Çetinkaya and Gupta (1994).

Adapting the identical definitions as in makespan minimization problems, the single-job M-machine lot streaming problem with s continuous and consistent transfer batches to minimize the mean (total) flow time $|1|F|TB(C,c)|\sum q_i C_i$ can be described by the following convex programming model:

$$\text{Min} \quad \frac{1}{Q}\sum_{i=1}^{s} C_{i,m} q_i$$

$$\text{s.to} \quad C_{i,m} \geq C_{i,m-1} + p_m q_i, \qquad i = 1,\ldots,s,\ m = 1,\ldots,M$$

$$C_{i,m} \geq C_{i-1,m} + p_m q_i, \qquad i = 1,\ldots,s,\ m = 1,\ldots,M$$

$$\sum_{i=1}^{s} q_i = Q,$$

$$q_i \geq 0, \qquad i = 1,\ldots,s$$

$$C_{i,0} = C_{0,m} = 0, \qquad i = 1,\ldots,s,\ m = 1,\ldots,M$$

where q_i and C_i are the size and the completion time of the ith transfer batch, respectively.

The first constraint prevents the completion time of the first transfer batch on the first machine from being less than the time it takes to process the first transfer batch on the first machine. The second set of constraints prevents transfer batch i from beginning its processing on machine m before transfer batch $i-1$ has completed its processing on machine m. The third set of constraints restricts transfer batch i from beginning its processing on machine m before it has completed its processing on machine $m-1$. The fourth type of constraint ensures that the sum of all transfer batches constitutes one production lot (job). Of course, the last two constraint sets are non-negativity constraints on the completion times and transfer batch sizes. Note that this LP formulation has $(2s-1)M-s+2$ constraints and $(M+1)s$ variables.

Note that if the transfer batch sizes are required to be integer, the model remains same with integrality restriction on the variable q_i, thus becomes a mixed integer nonlinear programming model. It can be seen that the objective function is a sum of piece wise quadratic (hence convex) functions. Furthermore, for the general problem above, it may not be possible to develop a polynomial time algorithm which gives the optimal solution. Therefore, we examine some specific cases in which polynomial time solutions exist. The solution properties of these specific problems can be used to devise heuristics for the general problem.

Definition 4.1 The M-machine N-job flowshop is called an *ordered flowshop* (Smith et al, 1975) if the following two properties are satisfied:

(i) If a particular job has a smaller processing time on any machine than does a second job on the same machine, this implies that the processing time of this first job is less than or equal the processing time of the second job on all machines.

(ii) If a job has its rth smallest processing time on some machine m, $m=1,2,...,M$, this implies that every other job will have its rth smallest processing time on the same machine m where $r=1,2,..,M$.

The four-job three-machine flowshop problem given in Table 4.1 can be used as an illustration of the ordered flow shop. In this problem, it is easy to see that the order of processing times of jobs on every machine is same, i.e., $t_{1,m} < t_{2,m} < t_{3,m} < t_{4,m}$ for every machine $m=1,2,3$, which implies the first property in Lemma 4.2. Furthermore, the shortest processing time for every job is on machine 1, the next smallest processing time for every job is on machine 3, and the longest processing

time for every job is on machine 2, i.e., $t_{j,1} < t_{j,3} < t_{j,2}$ for $j = 1,2,3$, which implies the second property in Lemma 4.2. Thus, the problem given in Table 4.1 satisfies the properties of an ordered flowshop.

TABLE 4.1. Ordered Flow Shop Processing Times

	Machine		
Job	1	2	3
1	1	3	2
2	3	5	4
3	5	8	6
4	9	12	10

Using results of Smith *et al.* (1975, 1976), Panwalker and Khan (1979) gives the following result for the ordered flow shops.

Lemma 4.1 *Let S be the sequence of N jobs (lots) in which jobs are arranged in nondecreasing order of processing times, i.e.,*

$$t_{1,m} \le t_{2,m} \le \ldots \le t_{N,m}, \qquad m = 1,2,\ldots,M$$

where $t_{i,m}$ is the processing time of the job in the ith position. Also, let S' be any sequence other than sequence S. If the longest processing time for every job in an ordered flow shop problem occurs on machine h, then

$$C_{i,h} \le C'_{i,h}, \qquad i = 1,2,\ldots,N$$

where $C_{i,h}$ and $C'_{i,h}$ denote the completion time of job in the ith position on machine h for sequences S and S', respectively.

Lemma 4.2 *The single-job lot streaming problem with consistent transfer batches satisfies the characteristics of the ordered flowshop defined above.*

Proof : Proof is obvious from two facts. First, if a transfer batch has the kth smallest processing time on a machine m, then it has the kth smallest processing time on every other machine. Second, if a transfer batch has its kth smallest processing time on machine m, then every other transfer batch has the kth smallest processing time on machine m. ∎

Using Lemma 4.1 and 4.2, we will prove the following result for a general single-job M-machine lot streaming problem with s consistent transfer batches to minimize the mean (total) flow time.

Theorem 4.1 *In the optimal solution of the problem* $1|F|TB(C)|\sum q_i C_i$, *transfer batches are arranged in nondecreasing order of their sizes, i.e.,*

$$q_i \le q_{i+1}, \qquad i = 1,2,\ldots,s$$

where q_i *is size of the transfer batch processed in the* i*th position.*

Proof: Let $S = \{1,2,\ldots,k-1,\ k,\ k+1,\ k+2,\ldots,s\}$ be the sequence of given s transfer batches in which transfer batches are arranged in nondecreasing order of their sizes, i.e., $q_i \le q_{i+1}$ for $1 \le i \le s-1$. Let $C_{i,m}$ be the completion time of the transfer batch processed in the ith position on machine m in this sequence. Also, let h denote the machine with the longest unit processing time, i.e., $p_h = \max_{1 \le m \le M}\{p_m\}$. Then we have

$$C_{1,h+m} = C_{1,h} + (\sum_{r=1}^{m} p_{h+r})q_1, \qquad m = 1,2,\ldots,M-h.$$

Similarly,

$$C_{2,h+m} = \max\{C_{2,h+m-1},\ C_{1,h+m}\} + p_{h+m}q_2, \qquad m = 1,2,\ldots,M-h. \tag{4.1}$$

From the properties of ordered flowshops given in Definition 4.1 and the characteristic of the sequence S (i.e., $q_i \le q_{i+1}$ for $1 \le i \le s-1$), it can be easily shown that

$$C_{1,h+m} \le C_{2,h+m-1}, \qquad m = 1,2,\ldots,M-h. \tag{4.2}$$

Using (4.2) in (4.1), we get

$$C_{2,h+m} = C_{2,h+m-1} + p_{h+m}q_2$$

Thus,

$$C_{2,h+m} = C_{2,h} + (\sum_{r=1}^{m} p_{h+r})q_2, \qquad m = 1,2,\ldots,M-h. \tag{4.3}$$

By similar reasoning, in general,

$$C_{i,h+m} = C_{i,h} + (\sum_{r=1}^{m} p_{h+r})q_i, \qquad i = 1,2,\ldots,s;\ m = 1,2,\ldots,M-h. \tag{4.4}$$

Hence, from (4.4) we obtain

$$C_{i,M} = C_{i,h} + (\sum_{m=h+1}^{M} p_m)q_i, \qquad i = 1,2,\ldots,s. \tag{4.5}$$

Using (4.5) we write the total flow time for the sequence S

$$F(S)=\sum_{i=1}^{s} q_i\, C_{i,M}=\sum_{i=1}^{s} q_i\,(C_{i,h}+(\sum_{m=h+1}^{M} p_m)q_i)=\sum_{i=1}^{s} q_i\, C_{i,h}+(\sum_{m=h+1}^{M} p_m)\sum_{i=1}^{s}(q_i)^2$$

$$=\sum_{i=1}^{k-1} q_i\, C_{i,h}+q_k\, C_{k,h}+q_{k+1}\, C_{k+1,h}+\sum_{i=k+2}^{s} q_i\, C_{i,h}+(\sum_{m=h+1}^{M} p_m)\sum_{i=1}^{s}(q_i)^2. \qquad (4.6)$$

Now, let $S'=\{1,2,...,k-1,\ k+1,\ k,\ k+2,...,s\}$ be another sequence which is identical to sequence S, except any arbitrary transfer batches k and $k+1$, $1\le k\le s-1$, are interchanged. Let $F(S')$ be the total flow times for this sequence.

It is clear that $F(S')=F(S)$ if $q_k=q_{k+1}$. Thus we will assume $q_k<q_{k+1}$ and show that this pairwise interchanging of transfer batches will not improve the total flow time for the sequence S, i.e., if $F(S')\ge F(S)$, then transfer batch $k+1$ can not directly precede transfer batch k in an optimal sequence. Thus, it is obvious that the sequence S is optimal arrangement of transfer batches.

Now, consider the sequence S'. Let q'_i and $C'_{i,m}$ be the size and completion time of the transfer batch processed in the ith position on machine m in the sequence S'. It is obvious that

$$\begin{aligned} &q'_i=q_i, && 1\le i\le k-1\\ &q'_k=q_{k+1}, &&\\ &q'_{k+1}=q_k, &&\\ &q'_i=q_i, && k+2\le i\le s. \end{aligned}$$

Then,

$$C'_{i,M}=C'_{i,h}+(\sum_{m=h+1}^{M} p_m)\,q'_i=C_{i,h}+(\sum_{m=h+1}^{M} p_m)\,q_i, \qquad 1\le i\le k-1 \qquad (4.7)$$

since $C'_{i,h}=C_{i,h}$ for $1\le i\le k-1$.

$$C'_{k,M}=C'_{k,h}+(\sum_{m=h+1}^{M} p_m)\,q'_k=C'_{k,h}+(\sum_{m=h+1}^{M} p_m)\,q_{k+1} \qquad (4.8)$$

$$C'_{k+1,M}\ge C'_{k+1,h}+(\sum_{m=h+1}^{M} p_m)\,q'_{k+1}=C'_{k+1,h}+(\sum_{m=h+1}^{M} p_m)\,q_k \qquad (4.9)$$

$$C'_{i,M}\ge C'_{i,h}+(\sum_{m=h+1}^{M} p_m)\,q'_i=C'_{i,h}+(\sum_{m=h+1}^{M} p_m)\,q_i \qquad k+2\le i\le s. \qquad (4.10)$$

Using expressions (4.7)-(4.10), we write the total flow time for the sequence S'

$$F(S') = \sum_{i=1}^{s} q'_i C'_{i,M} = \sum_{i=1}^{s} q'_i (C'_{i,h} + (\sum_{m=h+1}^{M} p_m) q'_i) = \sum_{i=1}^{s} q'_i C'_{i,h} + (\sum_{m=h+1}^{M} p_m) \sum_{i=1}^{s} (q'_i)^2$$
$$\geq \sum_{i=1}^{k-1} q_i C_{i,h} + q_{k+1} C'_{k,h} + q_k C'_{k+1,h} + \sum_{i=k+2}^{s} q_i C'_{i,h} + (\sum_{m=h+1}^{M} p_m) \sum_{i=1}^{s} (q_i)^2. \qquad (4.11)$$

Hence, from (4.6) and (4.11), we have

$$F(S') - F(S) = q_{k+1} C'_{k,h} + q_k C'_{k+1,h} - q_k C_{k,h} - q_{k+1} C_{k+1,h} + \sum_{i=k+2}^{s} q_i (C'_{i,h} - C_{i,h}). \qquad (4.12)$$

For a transfer batch in the ith position, completion time on machine h is given by

$$C_{i,h} = \max_{1 \leq m \leq h} \{C_{i-1,m} + \sum_{r=m}^{h} p_r q_i\}, \qquad i = 1,2,...,s \qquad (4.13)$$

for a sequence S. Then we have two cases to be investigated:

Case 1: $C_{k,h} = C_{k-1,c} + (\sum_{r=c}^{h} p_r) q_k$, and

$$C_{k+1,h} = C_{k,c} + (\sum_{r=c}^{h} p_r) q_{k+1} = C_{k-1,c} + p_c q_k + (\sum_{r=c}^{h} p_r) q_{k+1}.$$

This case corresponds to the case in which the completion time of transfer batches k and $k+1$ on machine h are both determined by the same critical machine c, $c \leq h$. Then, in the sequence S', we have

$$C'_{k,h} \geq C_{k-1,c} + (\sum_{r=c}^{h} p_r) q'_k = C_{k-1,c} + (\sum_{r=c}^{h} p_r) q_{k+1} \qquad (4.14)$$

since $q'_k = q_{k+1} > q_k$, which implies that the completion time of the transfer batch in the kth position on machine h in sequence S' is determined from either by machine c or a machine c', $c+1 \leq c' \leq h$. Furthermore, we have

$$C'_{k+1,h} \geq \max\{C_{k-1,c} + (\sum_{r=c}^{h} p_r) q'_k,\ C_{k-1,c} + p_c q'_{k+1} + (\sum_{r=c}^{h-1} p_r) q'_{k+1}\} + p_h q'_{k+1}$$
$$= C_{k-1,c} + \max\{p_h q_{k+1} + (\sum_{r=c}^{h-1} p_r) q_{k+1},\ p_c q_k + (\sum_{r=c}^{h-1} p_r) q_k\} + p_h q_k$$
$$C'_{k+1,h} \geq C_{k-1,c} + (\sum_{r=c}^{h} p_m) q_{k+1} + p_h q_k \qquad (4.15)$$

since $q_{k+1} > q_k$ and $p_h \geq p_c$.

Hence, using (4.12), we obtain

$$
\begin{aligned}
F(S')-F(S) \geq\ & q_{k+1}\,[C_{k-1,c}+(\sum_{r=c}^{h} p_r)\,q_{k+1}]+q_k\,[C_{k-1,c}+(\sum_{r=c}^{h} p_r)\,q_{k+1}+p_h\,q_k] \\
& -q_k\,[C_{k-1,c}+(\sum_{r=c}^{h} p_r)\,q_k]-q_{k+1}\,[C_{k-1,c}+p_c\,q_k+(\sum_{r=c}^{h} p_r)\,q_{k+1}] \\
& +\sum_{i=k+2}^{s} q_i(C'_{i,h}-C_{i,h}) \\
=\ & (\sum_{r=c+1}^{h} p_r)\,q_k\,q_{k+1}-(\sum_{r=c}^{h-1} p_r)\,(q_k)^2+\sum_{i=k+2}^{s} q_i(C'_{i,h}-C_{i,h}) \\
=\ & (p_h\,q_{k+1}-p_c\,q_k)\,q_k+(\sum_{r=c+1}^{h-1} p_r)\,q_k\,(q_{k+1}-q_k)+\sum_{i=k+2}^{s} q_i(C'_{i,h}-C_{i,h})
\end{aligned}
\tag{4.16}
$$

The first two terms in (4.16) are always greater than zero since $q_{k+1} > q_k$ and $p_h \geq p_c$. From Lemmas 4.1 and 4.2, since $C'_{i,h} \geq C_{i,h}$, $1 \leq i \leq s$, the last term in (4.16) is also greater than or equal to zero. Thus, $F(S')-F(S) > 0$.

Case 2: $C_{k,h} = C_{k-1,c}+(\sum_{r=c}^{h} p_r)\,q_k$, and

$$C_{k+1,h} = C_{k,c'}+(\sum_{r=c'}^{h} p_r)\,q_{k+1} \geq C_{k,c}+(\sum_{r=c}^{h} p_r)\,q_{k+1}$$

where $1 \leq c' \leq c-1$.

This case corresponds to the case in which the completion time of transfer batches k and $k+1$ on machine h are determined by different critical machines c and c', $1 \leq c' \leq c-1$. Then, in both sequences S and S', we consider machine c' to compute the completion time of transfer batches k and $k+1$ on machine h. In the sequence S, we have

$$C_{k,h} \geq C_{k-1,c'}+(\sum_{r=c'}^{h} p_r)\,q_k$$

or equivalently,

$$C_{k,h} = C_{k-1,c'}+(\sum_{r=c'}^{h} p_r)\,q_k+\varepsilon_k. \tag{4.17}$$

where $\varepsilon_k \geq 0$. Also, we have

$$C_{k+1,h} = C_{k,c'}+(\sum_{r=c'}^{h} p_r)\,q_{k+1} = C_{k-1,c'}+p_{c'}\,q_k+(\sum_{r=c'}^{h} p_r)\,q_{k+1}. \tag{4.18}$$

On the other hand, in the sequence S', we have

$$C'_{k,h} \geq C_{k-1,c'}+(\sum_{r=c'}^{h} p_r)\,q'_k = C_{k-1,c'}+(\sum_{r=c'}^{h} p_r)\,q_{k+1}$$

or equivalently,

$$C'_{k,h} = C_{k-1,c'} + (\sum_{r=c'}^{h} p_r)\, q_{k+1} + \varepsilon'_k \tag{4.19}$$

where $\varepsilon'_k \geq 0$. Furthermore, we have

$$\begin{aligned} C'_{k+1,h} &\geq \max\{C_{k-1,c'} + (\sum_{r=c'}^{h} p_r)\, q'_k,\ C_{k-1,c'} + p_{c'}\, q'_{k+1} + (\sum_{r=c'}^{h-1} p_r)\, q'_{k+1}\} + p_h\, q'_{k+1} \\ &= C_{k-1,c'} + \max\{p_h\, q_{k+1} + (\sum_{r=c'}^{h-1} p_r)\, q_{k+1},\ p_{c'}\, q_k + (\sum_{r=c'}^{h-1} p_r)\, q_k\} + p_h\, q_k \\ C'_{k+1,h} &\geq C_{k-1,c'} + (\sum_{r=c'}^{h} p_m)\, q_{k+1} + p_h\, q_k \end{aligned} \tag{4.20}$$

since $q_{k+1} > q_k$ and $p_h \geq p_{c'}$.

Hence, using (4.12), we obtain

$$\begin{aligned} F(S') - F(S) &\geq q_{k+1}\,[C_{k-1,c'} + (\sum_{r=c'}^{h} p_r)\, q_{k+1} + \varepsilon'_k] + q_k\,[C_{k-1,c'} + (\sum_{r=c'}^{h} p_r)\, q_{k+1} + p_h\, q_k] \\ &\quad - q_k\,[C_{k-1,c'} + (\sum_{r=c'}^{h} p_r)\, q_k + \varepsilon_k] - q_{k+1}\,[C_{k-1,c'} + p_{c'}\, q_k + (\sum_{r=c'}^{h} p_r)\, q_{k+1}] \\ &\quad + \sum_{i=k+2}^{s} q_i (C'_{i,h} - C_{i,h}) \\ &= (\sum_{r=c'+1}^{h} p_r)\, q_k\, q_{k+1} - (\sum_{r=c'}^{h-1} p_r)\, (q_k)^2 + \varepsilon'_k\, q_{k+1} - \varepsilon_k\, q_k + \sum_{i=k+2}^{s} q_i (C'_{i,h} - C_{i,h}) \\ &= (p_h\, q_{k+1} - p_{c'}\, q_k)\, q_k + (\sum_{r=c'+1}^{h-1} p_r)\, q_k\, (q_{k+1} - q_k) + \varepsilon'_k\, q_{k+1} - \varepsilon_k\, q_k \\ &\quad + \sum_{i=k+2}^{s} q_i (C'_{i,h} - C_{i,h}) \end{aligned} \tag{4.21}$$

The first two terms in (4.21) are always greater than zero since $q_{k+1} > q_k$ and $p_h \geq p_{c'}$. Furthermore, $\varepsilon'_k \geq \varepsilon_k$ since $q'_k = q_{k+1} > q_k$. Thus, $\varepsilon'_k\, q_{k+1} - \varepsilon_k\, q_k \geq 0$. From Lemma 4.2, since $C'_{i,h} \geq C_{i,h}$ $k+2 \leq i \leq s$, the last term in (4.21) is also greater than or equal to zero. Thus, $F(S') - F(S) > 0$.

In both cases, $F(S') > F(S)$, which is necessary for the optimality of sequence S.

4.1 M-machine Problem with $p_1 = \max_{1 \leq m \leq M}\{p_m\}$

The following theorem gives the optimal integer transfer batch sizes for a special case of M-machine s-sublot problem where the first machine has the longest unit processing time.

Theorem 4.2 *In the problem* $|1|F|TB(C,d)|\sum q_i C_i$, *if* $p_1 = \max_{1\le m\le M} p_m$, *then the optimal integer transfer batch sizes are*

$$q_i = \begin{Bmatrix} \lfloor Q/s \rfloor, & i = 1,\ldots,k \\ \lceil Q/s \rceil, & i = k+1,\ldots,s \end{Bmatrix} \tag{4.22}$$

where $k = s - Q + s(\lfloor Q/s \rfloor)$, $\lfloor x \rfloor$ *is the greatest integer less than or equal to* x, *and* $\lceil x \rceil$ *is the smallest integer greater than or equal to* x.

Proof: Let q_i be the optimal integer size of the ith transfer batch. Using Theorem 4.1, the completion time of the ith transfer batch on the last machine in an optimal solution can be written as

$$C_{i,M} = p_1 \sum_{v=1}^{i} q_v + (\sum_{m=2}^{M} p_m) q_i \tag{4.23}$$

since $C_{i,m} \le C_{i+1,m-1}$ for $m = 2,\ldots,M$. Thus, the total flow time is given by

$$\begin{aligned} F &= \sum_{i=1}^{s} q_i C_{i,M} = \sum_{i=1}^{s} q_i [p_1 \sum_{v=1}^{i} q_v + (\sum_{m=2}^{M} p_m) q_i] \\ &= p_1 \sum_{i=1}^{s} q_i \sum_{v=1}^{i} q_v + (\sum_{m=2}^{M} p_m) \sum_{i=1}^{s} (q_i)^2 \\ &= (p_1/2) \sum_{i=1}^{s} (q_i)^2 + (p_1/2)(\sum_{i=1}^{s} q_i)^2 + (\sum_{m=2}^{M} p_m) \sum_{i=1}^{s} (q_i)^2 \\ &= [(p_1/2) + (\sum_{m=2}^{M} p_m)] \sum_{i=1}^{s} (q_i)^2 + (p_1/2) Q^2. \end{aligned} \tag{4.24}$$

Then, it is sufficient to minimize $\sum_{1\le i\le s} q_i^2$ since the second term in (4.24) is a constant. It follows that the minimum value of this expression is achieved by equal sized transfer batches, i.e., $q_i = Q/s$ for $1 \le i \le s$ since $\sum_{1\le i\le s} q_i = Q$. If Q/s is integer, then the current solution is optimal for both continuous and discrete version of the problem. However, in the case of non integer solution (i.e., Q/s is not an integer), it is relatively easy to obtain the integer transfer batch sizes. In this case, we will first show that to minimize the sum of square of the integer transfer batch sizes it is sufficient to minimize the sum of squares of the difference between the integer and continuous transfer batch sizes, i.e.,

$$\sum_{i=1}^{s} [q_i - (Q/s)]^2 = \sum_{i=1}^{s} [(q_i)^2 - 2(Q/s) q_i + (Q/s)^2]$$

$$= \sum_{i=1}^{s} (q_i)^2 - 2(Q/s)\sum_{i=1}^{s} q_i + \sum_{i=1}^{s} (Q/s)^2$$

$$= \sum_{i=1}^{s} (q_i)^2 - 2(Q/s)Q + s(Q/s)^2$$

$$= \sum_{i=1}^{s} (q_i)^2 - (Q^2/s). \qquad (4.25)$$

Thus, minimizing the first term in (4.25) is equivalent to minimizing the first term in (4.24).

Now, let $f = (Q/s) - \lfloor Q/s \rfloor$. Then, $\lceil Q/s \rceil = 1 - f + (Q/s)$. It is clear that the minimum sum of squares of the difference between the integer and continuous transfer batch sizes is obtained if some k of the transfer batch sizes equals $\lfloor Q/s \rfloor$ and $(s-k)$ equals $\lceil Q/s \rceil$. In other words,

$$Q = k\lfloor Q/s \rfloor + (s-k)\lceil Q/s \rceil. \qquad (4.26)$$

Substituting $\lfloor Q/s \rfloor = (Q/s) - f$ and $\lceil Q/s \rceil = 1 - f + (Q/s)$ into (4.26) and making some arrangements yields

$$k = s(1-f). \qquad (4.27)$$

Note that $k < s$ since $0 < 1 - f < 1$.

Now, we need to check whether k in the equation above is integer or not. Substituting $f = (Q/s) - \lfloor Q/s \rfloor$ into the equation yields

$$k = s(1-f) = s(1-(Q/s)+\lfloor Q/s \rfloor) = s - Q + s(\lfloor Q/s \rfloor). \qquad (4.28)$$

It is clear that k becomes integer since all the terms on the right hand side of equation (4.28) are integers.

From Theorem 4.1, we know that transfer batch sizes are arranged in non-decreasing order, hence an optimal integer solution is obtained if the first k of the transfer batches equals $\lfloor Q/s \rfloor$ and remaining $(s-k)$ equals $\lceil Q/s \rceil$. ∎

4.2 M-machine Problem with Two Transfer Batches

In this section, we analyze the problem for a special case in which there are only two transfer batches. Let $q_1 = q$ and $q_2 = Q - q$. Then the completion times of the first and second transfer batches on the last machine are

$$C_{1,M} = (\sum_{m=1}^{M} p_m)q, \qquad (4.29)$$

$$C_{2,M} = \max_{1\le m\le M}\{(\sum_{k=1}^{m} p_k)q + (\sum_{k=m}^{M} p_k)(Q-q)\}, \tag{4.30}$$

respectively. Thus, the problem reduces to a minimization problem with a single decision variable which is the size of the first transfer batch , i.e.,

$$\text{Min} \quad F = \sum_{i=1}^{s} q_i C_{i,M} = (\sum_{m=1}^{M} p_m)q^2 + (Q-q)\max_{1\le m\le M}\{(\sum_{k=1}^{m} p_k)q + (\sum_{k=m}^{M} p_k)(Q-q)\}$$

$$\text{s.to} \qquad 0 \le q \le Q.$$

The following developments lead to a $O(M^2)$ algorithm to optimally solve the problem.

Lemma 4.3 (Baker and Pyke, 1990) *Machine f is a feasible machine (the machine that determines the completion time of the second transfer batch) for $q_{f,L} \le q \le q_{f,U}$ if $q_{f,L} \le q_{f,U}$ where*

$$q_{f,L} = Q \cdot \max_{1\le m<f}\{\sum_{k=m}^{f-1} p_k / [\sum_{k=m}^{f-1} p_k + \sum_{k=m+1}^{f} p_k]\}, \tag{4.31}$$

and

$$q_{f,U} = Q \cdot \max_{f<m\le M}\{\sum_{k=f}^{m-1} p_k / [\sum_{k=f}^{m-1} p_k + \sum_{k=f+1}^{m} p_k]\}. \tag{4.32}$$

By Lemma 4.3, it is clear that the quadratic behavior of the total flow time function with respect to q changes at break points, i.e., at a point for which the feasible machine changes from one to another. Thus, F is a piece-wise quadratic function over q. Therefore, in order to obtain the stationary point of the quadratic function within the region $[q_{f,L}, q_{f,U}]$, the following model must be solved.

$$\text{Min} \quad F_f(q) = (\sum_{m=1}^{M} p_m)q^2 + (\sum_{m=1}^{f} p_m)q(Q-q) + (\sum_{m=f}^{M} p_m)(Q-q)^2 \tag{4.33}$$

$$\text{s.t.} \quad Q \cdot \max_{1\le m<f}\{\sum_{k=m}^{f-1} p_k / [\sum_{k=m}^{f-1} p_k + \sum_{k=m+1}^{f} p_k]\} \le q \le Q \cdot \max_{f<m\le M}\{\sum_{k=f}^{m-1} p_k / [\sum_{k=f}^{m-1} p_k + \sum_{k=f+1}^{m} p_k]\}$$

If we ignore the constraint in the model and setting $\partial F(q)/\partial q = 0$, then the solution to the unconstrained problem is obtained as

$$\tilde{q}_f = (1 + \frac{p_f - T_f}{p_f + 2(T - T_f)}) \cdot \frac{Q}{2} \tag{4.34}$$

where $T = \sum_{m=1}^{M} p_m$ and $T_f = \sum_{m=1}^{f} p_m$. Since $F_f(q)$ is a convex function,

(i) $F_f(\tilde{q}_f) < F_f(q_{f,L}) < F_f(q_{f,U})$ if $\tilde{q}_f < q_{f,L}$,
(ii) $F_f(q_{f,L}) \geq F_f(\tilde{q}_f) \leq F_f(q_{f,U})$ if $q_{f,L} \leq \tilde{q}_f \leq q_{f,U}$,
(iii) $F_f(q_{f,L}) > F_f(q_{f,U}) > F_f(\tilde{q}_f)$ if $\tilde{q}_f > q_{f,U}$.

Hence, the optimal solution to the constrained model becomes

$$q_f^* = \begin{cases} q_{f,L} & \text{if} \quad \tilde{q}_f < q_{f,L} \\ \tilde{q}_f & \text{if} \quad q_{f,L} \leq \tilde{q}_f \leq q_{f,U} \\ q_{f,U} & \text{if} \quad \tilde{q}_f > q_{f,U} \end{cases} \tag{4.35}$$

A solution procedure for solving the problem may be as follows. For each machine, we check whether the machine is feasible or not. If so, the value of q and the corresponding total flow time are computed from (4.33), (4.34) and (4.35). Then, take the best of the weighted total flow times determined for each feasible machine at its optimum. It is easy to see that the time complexity of this procedure is $O(M^2)$. However, we can develop a better solution procedure by analyzing the special structure of the problem.

Lemma 4.4 *Machine h with smallest index satisfying $p_h \geq \max_{1 \leq j \leq M} p_j$ is a feasible machine.*

Proof: Let B_h and A_h be the indices for the adjacent feasible machines of machine h (denoted by $B_h \leftrightarrow h \leftrightarrow A_h$) such that $B_h < h$, $A_h > h$. Furthermore, assume that if $h = 1$, then $B_1 = 0$ is a dummy feasible machine such that $p_0 = 0$. Similarly, if $h = M$, then $A_M = M + 1$ is the next dummy feasible machine such that $p_{M+1} = 0$. Then, from (4.31) and (4.32), the lower and upper bounds on q for which machine h is feasible are written as

$$q_{h,L} = \{\sum_{m=B_h}^{h-1} p_m / [\sum_{m=B_h}^{h-1} p_m + \sum_{m=B_h+1}^{h} p_m]\} \cdot Q = (1 + \frac{T_h - T_{B_h}}{T_h - T_{B_h} + p_{B_h} - p_h})^{-1} Q \tag{4.36}$$

$$q_{h,U} = \{\sum_{m=h}^{A_h-1} p_m / [\sum_{m=h}^{A_h-1} p_m + \sum_{m=h+1}^{A_h} p_m]\} \cdot Q = (1 + \frac{T_{A_h} - T_h}{T_{A_h} - T_h + p_h - p_{A_h}})^{-1} Q \tag{4.37}$$

where $T_{B_h} = \sum_{m=1}^{B_h} p_m$, $T_h = \sum_{m=1}^{h} p_m$ and $T_{A_h} = \sum_{m=1}^{A_h} p_m$.

Substituting $T_h = T_{B_h} + \alpha + p_h$ and $T_{A_h} = T_h + \beta + p_{A_h}$ (where $\alpha = \sum_{m=B_h+1}^{h-1} p_m$ and $\beta = \sum_{m=h+1}^{A_h-1} p_m$) into corresponding expressions (4.36) and (4.37) yields

$$q_{h,L} = Q(1 + \frac{\alpha + p_h}{\alpha + p_{B_h}})^{-1} \tag{4.38}$$

$$q_{h,U} = Q(1 + \frac{\beta + p_{A_h}}{\beta + p_h})^{-1} \tag{4.39}$$

Case 1: $p_h > p_m$ for $1 \le m \le M$, $m \ne h$ (i.e., there exists only one machine having the maximum processing time). Then, from (4.38), it is obvious that $q_{h,L} < Q/2$ since $\alpha \ge 0$ and $p_h > p_{B_h}$. Similarly, equation (4.39) gives $q_{h,U} > Q/2$ since $\beta \ge 0$ and $p_h > p_{A_h}$. Hence, $q_{h,L} < Q/2 < q_{h,U}$.

Case 2: $p_h > p_m \ \forall m < h$, and $p_h \ge p_m \ \forall m > h$ (i.e., there exists more than one machine having the same maximum processing time). Let h be the smallest machine index for which machine h has the maximum processing time. Then, from (4.38), we get $q_{h,L} < Q/2$ since $\alpha \ge 0$ and $p_h > p_{B_h}$. Similarly, equation (4.39) gives $q_{h,U} = Q/2$ since $\beta \ge 0$ and $p_h = p_{A_h}$. Hence, $q_{h,L} < q_{h,U} = Q/2$.

Therefore, in both cases, it is clear that machine h with the smallest index having the maximum processing time is a feasible machine since $q_{h,L} < q_{h,U}$ in both cases.

Lemma 4.5 *Let f be the index of a feasible machine. Also, B_f and A_f are two adjacent feasible machines such that $B_f \leftrightarrow f \leftrightarrow A_f$, $B_f < f$, $A_f > f$. Then, $p_{B_f} < p_f < p_{A_f}$ for all $f < h$ where h is the machine with smallest index having the maximum processing time.*

Proof: Consider the feasible machine f such that $A_f = h$. For machine f to be feasible, by definition, $q_{f,L} \le q_{f,U}$. Using equations (4.31) and (4.32) we can write the relation between $q_{f,L}$ and $q_{f,U}$ as

$$(1 + \frac{T_f - T_{B_f}}{T_f - T_{B_f} + p_{B_f} - p_f})^{-1} Q \le (1 + \frac{T_h - T_f}{T_h - T_f + p_f - p_h})^{-1} Q$$

which can be simplified to

$$(T_h - T_f) p_{B_f} - T_h p_f \le (T_{B_f} - T_f) p_h - T_{B_f} p_f. \tag{4.40}$$

Adding $T_{B_f} p_{B_f}$ to both sides of (4.40) and rearranging it we obtain

$$p_{B_f} \le p_f + (T_f - T_{B_f})(p_f - p_h)/(T_h - T_f). \tag{4.41}$$

The second term in the right hand side of (4.41) is less than zero since $T_f - T_{B_f} > 0$, $T_h - T_f > 0$ and $p_f - p_h < 0$. Hence, it is clear that $p_{B_f} < p_f < p_h$. Therefore, in general, for every feasible machine $f < h$, we get $p_{B_f} < p_f < p_{A_f}$ which is the desired result. ∎

Lemma 4.6 *As the index of a feasible machine increases then its corresponding $\tilde{q}$ value decreases.*

Proof: Let B_h and h are two adjacent feasible machines such that $B_h < h$ and machine h with smallest index satisfying $p_h \ge \max_{1 \le j \le M} p_j$. Using equation (4.34), the difference between $\tilde{q}_{B_h}$ and $\tilde{q}_h$ can be written as

$$\tilde{q}_{B_h} - \tilde{q}_h = (Q/2K)[p_{B_h}(2T - T_h) + p_h(T_{B_h} - 2T) + 2T(T_h - T_{B_h})] \tag{4.42}$$

where $K = [p_{B_h} + 2(T - T_{B_h})][p_h + 2(T - T_h)] > 0$ since $T \ge T_h > T_{B_h}$.

Then, substituting $T_h = T_{B_h} + \alpha + p_h$ into (4.42) and rearranging yields

$$\tilde{q}_{B_h} - \tilde{q}_h = (Q/2K)[p_{B_h}(2T - T_{B_h}) + p_h(2T_{B_h} - p_{B_h}) + \alpha(2T - p_{B_h})]. \tag{4.43}$$

Since $2T - T_{B_h} > 0$, $2T_{B_h} - p_{B_h} > 0$, $2T - p_{B_h} > 0$, and $\alpha \ge 0$, it is clear that the right hand side of (4.43) is greater than zero which implies that $\tilde{q}_{B_h} > \tilde{q}_h$. By induction, this relation is also true between all pairs of adjacent feasible machines. Therefore, in general, it is obvious that as the index of a feasible machine increases then its corresponding $\tilde{q}$ value decreases. ∎

Theorem 4.3 *To find an optimal solution to the $1|F|TB(s=2,C)|\sum q_i C_i$ problem, it is sufficient to consider the feasible machines 1 through h where h is the machine with smallest index having the maximum processing time.*

Proof: From (4.31) and (4.32), it is clear that machine 1 is always a feasible machine in every problem since $q_{1,U} > q_{1,L}$. Equation (4.34) gives us $\tilde{q}_1 = Q/2$, and, from Lemma 4.6, it is obvious that $\tilde{q}_h \le Q/2$. Then, in the proof of Lemma 4.4, we have $\tilde{q}_h \le Q/2 \le q_{h,U}$. Hence, from equation (4.35), we get $q_h^* \le Q/2$. Therefore, to find the optimal value of q it is sufficient to consider the feasible machines 1

through h where h is the machine with smallest index having the maximum processing time. ∎

Theorem 4.4 *For the* $1|F|TB(s=2,C)|\sum q_i C_i$ *problem, the optimal size* q^* *of the first transfer batch is*

(i) $q^* = q_f^* = \tilde{q}_f$ if $q_{f,L} < \tilde{q}_f < q_{f,U}$;
(ii) $q^* = q_f^* = q_{f,L} = q_{B_f}^* = q_{B_f,U}$ if $\tilde{q}_f \le q_{f,L} = q_{B_f,U} < \tilde{q}_{B_f}$;
(iii) $q^* = q_f^* = q_{f,U} = q_{A_f}^* = q_{A_f,L}$ if $\tilde{q}_{A_f} < q_{A_f,L} = q_{f,U} \le \tilde{q}_f$

where f *is the index for a feasible machine such that* $f \le h$.

Proof: Case (i): $q_{f,L} < \tilde{q}_f < q_{f,U}$.

Take b as the index for an adjacent machine such that $b < B_f$. Then,

$$q_{b,U} < q_{B_f,U} = q_{f,L} < \tilde{q}_f < \tilde{q}_{B_f} < \tilde{q}_b$$

and

$$q_b^* = q_{b,U} < q_{B_f}^* = q_{B_f,U} < \tilde{q}_f$$

since $\tilde{q}_f < \tilde{q}_{B_f} < \tilde{q}_b$ for all feasible machines such that $b < B_f$. Thus,

$$F(q_b^*) > F(q_{B_f}^*) = F(q_{B_f,U}) = F(q_{f,L}) > F(\tilde{q}_f).$$

Similarly,

$$\tilde{q}_a < \tilde{q}_{A_f} < \tilde{q}_f < q_{f,U} = q_{A_f,L} < q_{a,L}$$

and

$$\tilde{q}_f < q_{A_f}^* = q_{r,L} < q_a^* = q_{a,L}$$

since $\tilde{q}_a < \tilde{q}_{A_f} < \tilde{q}_f$ for all feasible machines such that $a > A_f$. Hence,

$$F(q_b^*) > F(q_{B_f}^*) > F(\tilde{q}_f) < F(q_{A_f}^*) < F(q_a^*) \text{ and } q^* = q_f^* = \tilde{q}_f.$$

Case (ii): $\tilde{q}_f \le q_{f,L} = q_{B_f,U} < \tilde{q}_{B_f}$.

Following a similar analysis given in Case (i), it can be shown that

$$q^* = q_f^* = q_{f,L} = q_{B_f}^* = q_{B_f,U},$$

$$F(q_b^*) > F(q_{B_f}^*) = F(q_{B_f,U}) = F(q_{f,L}) = F(q_f^*) < F(q_a^*)$$

for all feasible machines such that $b < B_f$ and $a > f$. In other words, the optimal solution is at a break point in which feasible machines B_f and f give the same total weighted flow time.

Case (iii): $\tilde{q}_{A_f} < q_{A_f,L} = q_{f,U} \le \tilde{q}_f$.

Case (ii) and (iii) are the same if we replace the terms f and B_f in Case (ii) with A_f and f, respectively. The proof is then similar. ∎

The discussion above assumes that transfer batch sizes are continuous. However, in many practical situations, integer values of transfer batch sizes are needed. In our case, it is relatively easy to get integer transfer batch sizes when there are only two transfer batches. Using Theorems 4.3 and 4.4, and Lemma 4.5, we present the following algorithm to find the optimal continuous solution of the total flow time problem with two transfer batches as well as its optimal integer solution.

Now, let us define the following variables:

$\lfloor q^* \rfloor$ = the greatest integer less than or equal to the optimal continuous size q^* of the first transfer batch,

$\lceil q^* \rceil$ = the smallest integer greater than or equal to the optimal continuous size q^* of the first transfer batch,

$F_f(q)$ = the total flow time determined by using feasible machine f if the size of the first transfer batch is q units such that

$$F_f(q) = Tq^2 + q(Q-q)T_f + (T - T_f + p_f)(Q-q)^2$$

where $T = \sum_{m=1}^{M} p_m$ and $T_f = \sum_{m=1}^{f} p_m$.

Algorithm 4.1

Step 1. a $h = 1,\ T_1 = p_1,\ m = 2.$

b Set $T_m = T_{m-1} + p_m$. If $p_m > p_h$ then set $B_m = h,\ h = m.$

c If $m < M$, then set $m = m+1$, and return to step 1(b).

Step 2. If $h = 1$, then $q^* = \tilde{q}_1 = Q/2$, go to step 4;
otherwise, set $T = T_M,\ f = h,\ q_{f,U} = Q/2.$

Step 3. a Compute $\tilde{q}_f = (1 + \dfrac{p_f - T_f}{p_f + 2(T - T_f)}) \cdot \dfrac{Q}{2}.$

b If $\tilde{q}_f \ge q_{f,U}$ then $q^* = q_f^* = q_{f,U}$, go to step 4;
otherwise, set $q_{\max} = 0,\ c = B_f.$

c If $c = 0$ then set $q_{f,L} = q_{\max}$, go to step 3(e);

otherwise, set $q_{c,U} = (1 + \frac{T_f - T_c}{T_f - T_c + p_c - p_f})^{-1} Q$.

d If $q_{\max} \geq q_{c,U}$ then set $c = B_c$, and return to step 3(c);
otherwise, set $q_{\max} = q_{c,U}$, $B_f = c$, and return to step 3(c).

e If $\tilde{q}_f \geq q_{f,L}$ then $q^* = q_f^* = \tilde{q}_f$, go to step 4;
otherwise, set $A_{B_f} = f$, $f = B_f$, return to step 3(a).

Step 4. a If q^* is an integer, then $F^* = F_f(q^*)$, stop;
otherwise,
If $\lfloor q^* \rfloor \geq q_{f,L}$ then set $F^L = F_f(\lfloor q^* \rfloor)$;
otherwise, set $F^L = F_{B_f}(\lfloor q^* \rfloor)$.
If $\lceil q^* \rceil \leq q_{f,U}$ then set $F^U = F_f(\lceil q^* \rceil)$;
otherwise set $F^U = F_{A_f}(\lceil q^* \rceil)$.

b If $F^L < F^U$ then set $q^* = \lfloor q^* \rfloor$, $F^* = F^L$;
otherwise, set $q^* = \lceil q^* \rceil$, $F^* = F^U$.
Stop.

The steps of Algorithm 4.1 can be explained as follows. Initially, machine 1 is considered as the machine having the maximum processing time. The sum of processing time of machines 1 through m is computed in Step 1(b). If the processing time of machine m is less than or equal to the processing time of machine h having maximum processing time, then machine m can not be a candidate feasible machine and the next machine is evaluated. However, if machine m is a candidate to be feasible, then the index of machine h is set to m. Step 1(b) is repeated until the last machine is evaluated. Clearly, if machine 1 has the maximum processing time, the optimal solution is already on hand, and the optimal size of the first and second transfer batches are equal. If not, the initial feasible machine for evaluation is set to machine h. Note that the upper bound on the size of the first transfer batch in which the machine having the maximum processing time is feasible is set to $Q/2$ since $\tilde{q}_h \leq Q/2$ and $q_{h,U} \geq Q/2$ (by Theorem 4.3). If $\tilde{q}_f \geq q_{f,U}$ for feasible machine f, then the optimal size (continuous solution) of the first transfer batch equals $q_{f,U}$ (by Theorem 4.4). If not, the next feasible machine B_f which is adjacent to machine f is searched to find $q_{f,L}$ value. If $\tilde{q}_f \geq q_{f,L}$, then the optimal size (continuous solution) of the first transfer batch equals $\tilde{q}_f$ (by Theorem 4.4). If machine f does not give the optimal solution, then Step 3 is repeated for the feasible machines until the optimal continuous solution is found.

If the optimal size of the first transfer batch found in Step 3 is an integer, then we terminate. If not, we check whether $\lfloor q^* \rfloor$ is still in the region in which the total flow time value is determined by feasible machine f or not. If so, the total flow time is computed by $F_f(q)$ for $q = \lfloor q^* \rfloor$; otherwise, $\lfloor q^* \rfloor$ lies in the region in which the total flow time value is determined by the adjacent feasible machine B_f, and the total flow time is computed by $F_{B_f}(q)$ for $q = \lfloor q^* \rfloor$. Then, a similar comparison is made for $\lceil q^* \rceil$. Finally, $\lfloor q^* \rfloor$ or $\lceil q^* \rceil$ is selected as the optimal integer size of the first transfer batch by comparing their corresponding total flow time values.

The following numerical example illustrates the algorithm given above.

Example 4.2

Consider the following 6-machine flowshop problem where $Q = 99$.

machine (m)	1	2	3	4	5	6
processing time (p_j)	1	2	1	3	5	3

Following the steps 1 and 2, we find that the machine having the largest processing time, $h = 5$. In addition, we have $T = T_6 = 15$, $f = 5$, $q_{5,U} = Q/2 = 49.5$, and

m	1	2	3	4	5	6
T_m	1	3	4	7	12	15
B_m	-	1	-	2	4	-

where machines 1, 2, 4 and 5 (by Lemma 4.5) are candidate feasible machines as shown in the following figure.

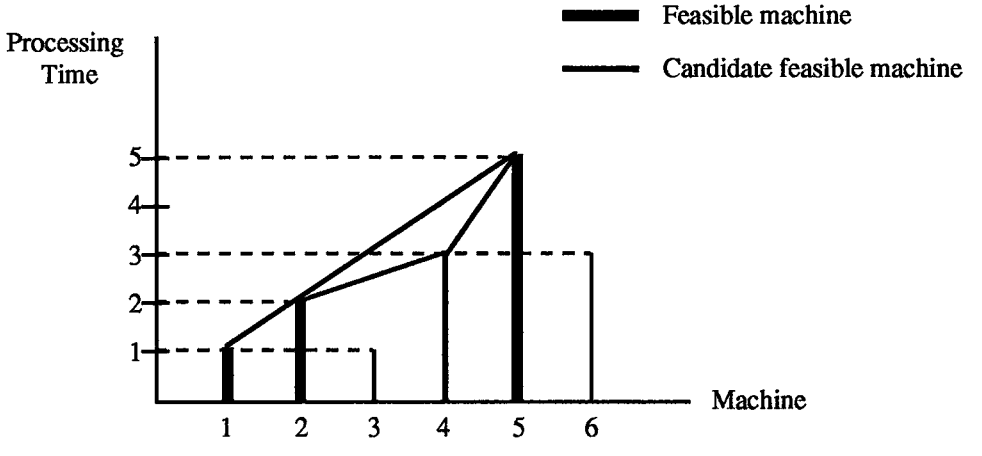


Figure 4.1. Feasible machines

Since $\tilde{q}_5 = 18 < q_{5,U} = 49.5$, the adjacent feasible machine to machine 5 is searched in step 3. Thus, machine 2 is found as the adjacent feasible machine to machine 5. Then, the revised vector of candidate feasible machines is obtained as follows.

m	1	2	3	4	5	6
B_m	-	1	-	-	2	-

$f = 2$ and $A_2 = 5$ since $\tilde{q}_5 = 18 < q_{5,L} = 39.6$. Thus, the optimal continuous size of the first transfer batch is $q^* = q_5^* = q_2^* = q_{2,U} = q_{5,L} = 39.6$ units, and the corresponding optimal total flow time is 79,976 since $q_{2,U} = 39.6 < \tilde{q}_2 = 47.6$. Note that this solution is at the break point in which feasible machines 2 and 5 are both critical, and work continuously in the optimal solution, i.e., there is no idle time between processing of transfer batches on these machines.

From Step 4(a), we get $\lfloor q^* \rfloor = 39$ and $F^L = F_2(\lfloor q^* \rfloor) = 80{,}235$ since $\lfloor q^* \rfloor = 39 > q_{2,L} = 33$. Similarly, from Step 4(b), we get $\lceil q^* \rceil = 40$ and $F^U = F_5(\lceil q^* \rceil) = 80{,}168$ since $\lceil q^* \rceil = 40 > q_{2,U} = 39.6$.

Since $F^U < F^L$, $q^* = \lceil q^* \rceil = 40$. i.e., the optimal integer size of first and second transfer batches are 40 and 59 units, respectively, and the corresponding total weighted flow time is 80,168 (See Figure 4.2).

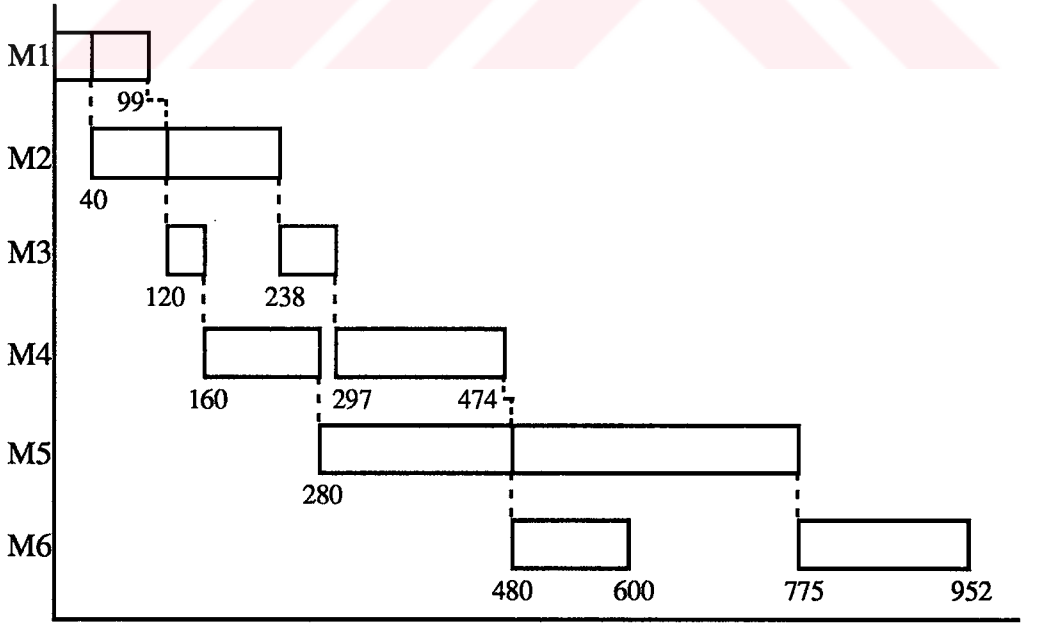


Figure 4.2. Optimal schedule in Example 4.2.

4.3 Two-machine Problem

We now consider the $1|F2|TB(C,c)|\sum q_iC_i$ problem. It is clear that Theorem 4.2 gives the optimal transfer batch sizes if $p_1 \geq p_2$. Thus we only analyze the second case where $p_1 < p_2$. The following theorem is conjectured by Çetinkaya and Gupta (1994) and proved later by Şen *et al.* (1994).

Theorem 4.5 *In the optimal solution of the two-machine s-sublot flowshop lot streaming problem, if* $p_1 < p_2$

(i) there is no idle time between processing of transfer batches on machine 2, and

(ii) there exists a transfer batch k *for which the optimal transfer batch sizes are of the form*

$$q_i = \pi^{i-1}q_1, \qquad i = 1,\ldots,k;$$
$$q_i = q_{k+1}, \qquad i = k+2,\ldots,s$$

if $q_k \leq q_{k+1} \leq \pi q_k$ where $\pi = p_2 / p_1$ and $k < s$.

A simple algorithm based on the Theorem 4.5 is given as follows:

Algorithm 4.2

Step 1 Set $\pi = p_2 / p_1$, $k = 1$.

Step 2 Compute

$$q_1 = \frac{(\frac{\pi^k-1}{\pi-1}) + \pi(s-k)}{(\frac{\pi^k-1}{\pi-1})^2 + (s-k)(\frac{\pi^{2k}-1}{\pi^2-1})}Q,$$

$$q_k = \pi^{k-1}q_1$$

$$q_{k+1} = \frac{Q - q_1(\sum_{i=1}^{k}\pi^{i-1})}{s-k}.$$

Step 3 If $q_k \leq q_{k+1} \leq \pi q_k$ then the optimal transfer batch sizes are

$$q_i = \begin{cases} \pi^{i-1}q_1 & for \quad 1 \leq i \leq k, \\ q_{k+1} & for \quad k+1 \leq i \leq s, \end{cases}$$

stop; otherwise, set $k = k+1$, and go to step 2.

The following example illustrates the above solution procedure.

Example 4.3

Assume that a lot of 280 items will be split into 10 sublots and processed in a two-machine flowshop where the processing times are 1 and 2 at machine 1 and 2, respectively. It is easy to verify that the smallest transfer batch index which satisfies the condition in step 3 is three. i.e., $k=3$. Hence, from step 2, we obtain $q_1=5$, $q_k=q_3=20$ and $q_{k+1}=q_4=35$ units. Since $q_{k+1}=q_4=35<\pi q_k=(2)(20)=40$, it follows that the optimal transfer batch sizes are $q_1=5$, $q_2=10, q_3=20$, and $q_4=\ldots=q_{10}=35$ units, and the corresponding optimal total weighted flow time is 88,900.

Now, as an extension of the two-machine s-sublot lot streaming problem to minimize the mean flow time, we assume that there are detached setups on the machines before the processing of the lot. Then, using our classification scheme, this problem is denoted by $1|F2|TB(C,c),S_d|\sum q_iC_i$.

Let S_1 and S_2 be the detached setup times on machine 1 and 2, respectively. If we set $S_1=S_2=0$, then this is a problem with no setup times, which we refer to as the *relaxed problem*. The following theorem can simplify the original problem.

Lemma 4.7 *Let (P') be an alternative lot streaming problem in which the setup on machine i is replaced by $S_i'=\max\{S_i-S_1,0\}$ for $i=1,2$. Then the optimal transfer batch sizes for the alternative problem are also optimal for the original problem.*

Proof: Assume that the setups on both machines start at time zero. If the setup time on machine 2 is greater than the setup time on machine 1, i.e., $S_2>S_1$, then machine 2 will be busy for the setup during the interval [0, S_1] in which the first machine is set up, and the setup on the second machine will continue during the interval $[S_1, S_2-S_1]$. Let S_i' be the *reduced* setup time required on machine i after the interval $[0,S_1]$. Thus, $S_2'=S_2-S_1$ and $S_1'=0$.

However, if the setup on machine 2 is less or equal to the setup time on the first machine, i.e., $S_2\le S_1$, then there won't be a setup on machine 2 after the interval $[0,S_1]$. Thus, $S_1'=S_2'=0$.

Hence, it is clear that $S_i'=\max\{S_i-S_1,0\}$ for a machine i. Thus, without loss of generality, we can say that our original lot streaming problem is equivalent to an

alternative lot streaming problem in which the processing on the first machine starts at time S_1, and the setup times on machine i is $S_i' = \max\{S_i - S_1, 0\}$. ◆

Corollary 4.1 *If $S_2' = 0$, then it is clear that the new (alternative) problem (P') as well as the original problem is equivalent to the relaxed problem without any setups.*

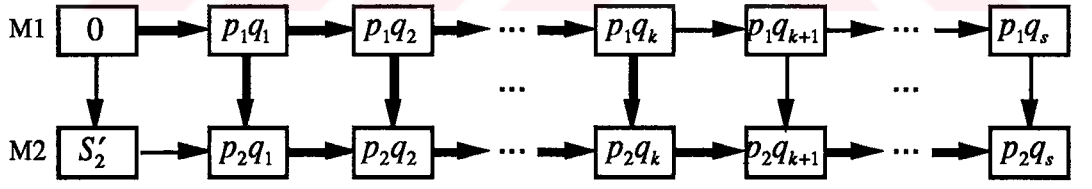
In the remainder of this section, we restrict our attention to a case where the setup time on the second machine is greater than the setup time on the first machine, i.e., $S_2 > S_1$. In other words, we will deal with the alternative problem in which $S_2' \neq 0$.

Theorem 4.6 *Let q_1^R be the optimal size of the first transfer batch in the relaxed problem without any setups. If $0 < S_2' \leq p_1 q_1^R$, then the optimal transfer batch sizes in the relaxed problem are also optimal for the alternative problem (P').*

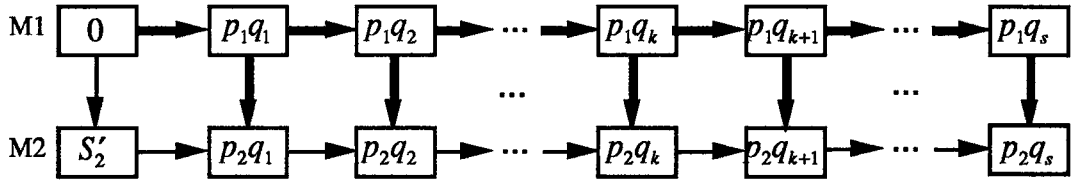
Proof: Omitted since it is trivial. ◆

Based on Theorem 4.6, we present the network of optimal transfer batch sizes for the problem (P') when $0 < S_2' \leq p_1 q_1^R$ as shown in Figure 4.3. The heavy and dashed arrows represent the critical and non-critical segments, respectively.

However, alternative problem (P') may really be different from relaxed problem if $S_2' > p_1 q_1^R$. Then we have the following cases to be investigated with respect to the relation between the processing times on machines.



(a) $p_1 < p_2$



(b) $p_1 \geq p_2$

Figure 4.3. Network of optimal transfer batch sizes when $0 < S_2' \leq p_1 q_1^R$.

Case 1 $p_1 < p_2$

Lemma 4.8 *In the optimal solution of the problem* (P')*, if* $p_1 < p_2$ *and* $p_1 q_1^R < S_2'$*, then there is no idle time between setup and the processing of the first transfer batch on the second machine.*

Proof: From the optimal structure of the relaxed problem without any setups, we know that the first transfer batch is always critical and there is no idle time between processing of transfer batches on machine 2. Now, assume that there exists an optimal solution $q=(q_1,q_2,...,q_s)$ for the alternative problem (P') such that $S_2' < p_1 q_1$. In this case, it is clear that $p_1 q_1^R < S_2' < p_1 q_1$. However, if this solution is optimal, then its structure must be same as the structure of the optimal solution for the relaxed problem without any setups since setup is not effective and thus can be ignored. Then, not only their structures, the solution $q=(q_1,q_2,...q_s)$ and the solution for the relaxed problem must also be equivalent in terms of transfer batch sizes, i.e., $q_i = q_i^R$ for $1 \le i \le s$, which is a contradiction to the fact that $S_2' > p_1 q_1^R$. Thus, the solution $q=(q_1,q_2,...,q_s)$ can not be optimal for the alternative problem (P'), and the processing of the first transfer batch on machine 1 must be finished before the setup on machine 2 is completed, which implies that there is no idle time between setup and the processing of the first transfer batch on the second machine. ∎

We have the following subcases to be investigated.

Subcase 1.1: $p_1(Q/s) \le S_2'$.

Theorem 4.7 *In the optimal solution of the problem* (P')*, if* $p_1 < p_2$ *and* $p_1(Q/s) \le S_2'$ *then the transfer batch sizes are equal, i.e.*

$$q_i = \frac{Q}{s}, \qquad i=1,2,...,s.$$

Proof: The completion time of the transfer batch i on the second machine is given by

$$C_{i,2} = \sum_{u=1}^{i} (I_u + p_2 q_u) \tag{4.44}$$

where I_u is the idle time, if any, between processing of transfer batch u and $u-1$ on the second machine. Note that if $u=1$ then I_u is defined as the idle time between processing of the first transfer batch and the finishing of the setup on the second machine. Thus,

$$\sum_{i=1}^{s} q_i C_i = \sum_{i=1}^{s} q_i [\sum_{u=1}^{i} (I_u + p_2 q_u)] = \sum_{i=1}^{s} q_i \sum_{u=1}^{i} I_u + p_2 \sum_{i=1}^{s} q_i \sum_{u=1}^{i} q_u$$
$$= \sum_{i=1}^{s} q_i \sum_{u=1}^{i} I_u + (p_2/2) \sum_{i=1}^{s} q_i^2 + (p_2/2)(\sum_{i=1}^{s} q_i)^2. \qquad (4.45)$$

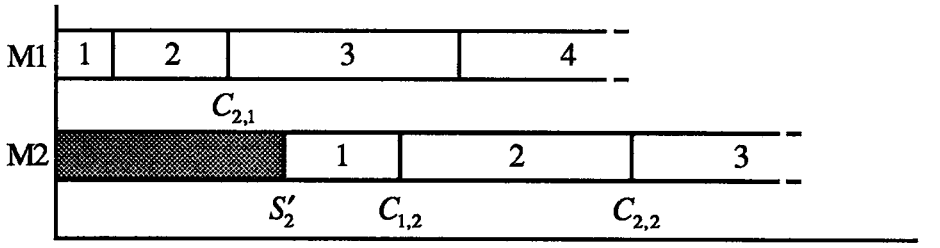
It is clear that the second term in (4.45) is minimized if all transfer batches are equal sized, i.e., $q_i = \frac{Q}{s}$ for $1 \le i \le s$. On the other hand, the first term is minimized when the total idle time on the second machine is equal to zero, i.e., $\sum_{u=1}^{s} I_u = 0$. By Lemma 4.8, we have $I_1 = 0$. If $q_i = \frac{Q}{s}$ for $1 \le i \le s$, then $I_i = 0$ for $1 \le i \le s$ and thus $\sum_{u=1}^{s} I_u = 0$ since $S_2' \ge p_1(Q/s)$ and $p_1 < p_2$, which implies that the first term in the equation above is also at its minimum. Therefore, it is obvious that $\sum_{i=1}^{s} q_i C_i$ is minimized when $q_i = \frac{Q}{s}$ for $1 \le i \le s$. ∎

Subcase 1.2: $p_1 q_1^R < S_2' < p_1(Q/s)$.

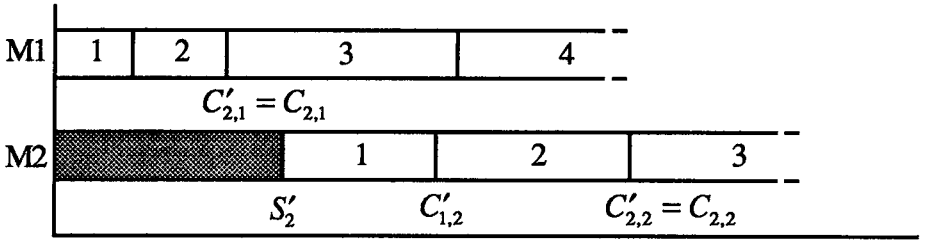
Lemma 4.9 *In the optimal solution of the problem (P'), the size of the first transfer batch is* $q_1 = S_2'/p_1$ *when* $p_1 < p_2$ *and* $p_1 q_1^R < S_2' < p_1(Q/s)$.

Proof: Assume that we have the optimal solution $q^R = (q_1^R, q_2^R, ..., q_s^R)$ for the relaxed problem without any setups. From the optimal structure of this solution, we know that transfer batch sizes are in non-decreasing order and there is no idle time between processing of transfer batches on machine 2. If we add the reduced setup time S_2' into the schedule of this solution, the completion time of every transfer batch increases by S_2' as shown in Figure 4.4. Now, we construct a solution (with primes denoting its transfer batches) by increasing the size of the transfer batch 1 by ε, decreasing the size of the transfer batch 2 by ε, and keeping the size of all other transfer batches unchanged. Let us denote the resulting solution by $q' = (q_1', q_2', q_3^R, ..., q_s^R)$ in which $q_1' = q_1^R + \varepsilon$, $q_2' = q_2^R - \varepsilon$, and $0 < \varepsilon < q_2^R - q_1^R$.

Let $C_{i,m}$ and $C_{i,m}'$ be the completion time of the transfer batch i on machine m, $m = 1,2$ in the solution q^R and q', respectively. Note that in the new solution $C_{i,2}' = C_{i,2}$ for $3 \le i \le s$.



(a)



(b)

Figure 4.4. Transfer batch completion (a) in the solution $q^R = (q_1^R, q_2^R, ..., q_s^R)$, (b) in the solution $q' = (q_1', q_2', q_3^R, ..., q_s^R)$.

Furthermore, let $F(q_1^R, q_2^R)$ be the contribution of transfer batches 1 and 2 to the total flow time $F(q^R)$ in the relaxed solution q^R. Then,

$$\begin{aligned} F(q^R) &= F(q_1^R, q_2^R) + F(q_3^R, ..., q_s^R) \\ &= C_{1,2}q_1^R + C_{2,2}q_2^R + F(q_3^R, ..., q_s^R) \\ &= (S_2' + p_2 q_1^R)q_1^R + [S_2' + p_2(q_1^R + q_2^R)]q_2^R + F(q_3^R, ..., q_s^R) \\ &= S_2'(q_1^R + q_2^R) + p_2(q_1^R)^2 + p_2(q_1^R + q_2^R)q_2^R + F(q_3^R, ..., q_s^R) \end{aligned} \tag{4.46}$$

where $F(q_3^R, ..., q_s^R)$ is the contribution of transfer batches 3 through s to the total flow time $F(q^R)$.

Similarly, let $F(q_1', q_2')$ be the contribution of transfer batches 1 and 2 to the total flow time $F(q')$ in the new solution q'. Then, we have

$$\begin{aligned} F(q') &= F(q_1', q_2') + F(q_3^R, ..., q_s^R) \\ &= C'_{1,2}q' + C'_{2,2}q' + F(q_3^R, ..., q_s^R) \\ &= (S_2' + p_2 q_1')q_1' + [S_2' + p_2(q_1' + q_2')]q_2' + F(q_3^R, ..., q_s^R) \\ &= S_2'(q_1^R + q_2^R) + p_2(q_1^R)^2 + p_2(q_1^R + q_2^R)q_2^R + p_2\varepsilon(q_1^R - q_2^R + \varepsilon) + F(q_3^R, ..., q_s^R) \end{aligned} \tag{4.47}$$

since $q_1' = q_1^R + \varepsilon$ and $q_2' = q_2^R - \varepsilon$. Hence, using (4.46) and (4.47), we obtain

$$F(q') - F(q^R) = p_2\varepsilon[\varepsilon - (q_2^R - q_1^R)]. \tag{4.48}$$

We know that $0 < \varepsilon < q_2^R - q_1^R$. Then, $F(q') < F(q^R)$, which implies that the new solution is better than the relaxed solution. Using the solution q', we again construct another new solution by increasing the size of the transfer batch 2 by ε $(0 < \varepsilon < q_3^R - q_2')$, decreasing the size of the transfer batch 3 by ε, and keeping the size of all other transfer batches unchanged. That is, we have the solution $q'' = (q_1', q_2'', q_3'', q_4^R ..., q_s^R)$ in which $q_2'' = q_2' + \varepsilon$ and $q_3'' = q_3^R - \varepsilon$. If $F(q'') < F(q')$, then we can continue to create another solution by the same way. We repeat the same process up to a point where the total flow time does not improve. At this point, we again go to the first transfer batch and repeat the same procedure. Hence, the size of the first transfer batch converges to S_2' / p_1. Note that size of the first transfer batch can not be greater than S_2' / p_1 due to Lemma 4.8. ∎

Using the results of Lemmas 4.8 and 4.9, we state the following theorem.

Theorem 4.8 *Let q_i be the optimal size of the ith transfer batch $(2 \le i \le s)$ in the optimal solution of the problem (P') when $p_1 < p_2$, $p_1 q_1^R < S_2' < p_1(Q/s)$, and $q_1 = S_2' / p_1$. Then the optimal transfer batch sizes satisfy the condition*

$$\alpha q_i \ge q_{i+1}, \qquad i = 2,...,s-1$$

where $\alpha = p_2 / p_1$.

Proof: Assume that there exists an optimal solution $q = (q_1, q_2, ..., q_s)$ in which some transfer batches does not satisfy the property given above, i.e., $\alpha q_i < q_{i+1}$ for some transfer batch i, $i = 2,...,s-1$. Let k be the first transfer batch which violates the property, i.e., $k = \min_{2 \le i \le s-1} \{i | \alpha q_i < q_{i+1}\}$.

Now, we construct a solution (with primes denoting its transfer batches) by increasing the size of the transfer batch k by ε, decreasing the size of the transfer batch $k+1$ by ε, and keeping the size of all other transfer batches unchanged. Let us denote the resulting solution by $q' = (q_1, q_2, ..., q_k', q_{k+1}', ..., q_s)$ in which $q_k' = q_k + \varepsilon$, $q_{k+1}' = q_{k+1} - \varepsilon$, and $\varepsilon > 0$.

Let $C_{i,m}$ and $C_{i,m}'$ be the completion time of the transfer batch i on machine m, $m = 1,2$ in the solution q and q', respectively. Note that in the new solution the completion time of the transfer batch $k-1$ on both machines remains unchanged (i.e., $C_{k-1,2}' = C_{k-1,2}$), which guarantees that the size of the transfer batches 1 through $k-1$ is the same as the one in the solution q. Similarly, the completion time of the transfer batch $k+1$ on machine 1 does not change with the new solution q' (i.e.,

$C'_{k+1,1} = C_{k+1,1}$). Then, to keep the size of the transfer batches $k+2$ through s unchanged, we only need to check whether or not the completion of the transfer batch $k+1$ changes in the new solution q'. Thus, the new solution q' is only feasible if $C'_{k+1,2} \le C_{k+1,2}$.

Furthermore, let $F(q)$ and $F(q')$ be the corresponding total flow times of these solutions. If the new solution is feasible, then it is sufficient to show that $F(q') < F(q)$, which implies that the solution q can not be optimal. There are two possible cases to be examined:

Case 1: $C_{k+1,1} \le C_{k,2}$.

This case corresponds to the no-idling on machine 2 between processing of transfer batches k and $k+1$ in the solution q as illustrated by Figure 4.5(a). In the corresponding new solution q' which is shown in Figure b. , the completion time of the transfer batch $k+1$ on machine 2 is

$$\begin{aligned} C'_{k+1,2} &= C'_{k,2} + p_2 q'_{k+1} \\ &= C_{k-1,2} + p_2 q'_k + p_2 q'_{k+1} \\ &= C_{k-1,2} + p_2(q_k + \varepsilon) + p_2(q_{k+1} - \varepsilon) \\ &= C_{k+1,2} \end{aligned} \tag{4.50}$$

which implies that the new solution q' is feasible.

In the solution q, the contribution of transfer batches k and $k+1$ to the total flow time $F(q)$ is

$$\begin{aligned} F(q_k, q_{k+1}) &= C_{k,2} q_k + C_{k+1,2} q_{k+1} \\ &= [C_{k-1,2} + p_2 q_k] q_k + [C_{k-1,2} + p_2(q_k + q_{k+1})] q_{k+1} \\ &= p_2 (q_k)^2 + (q_k + q_{k+1}) C_{k-1,2} + p2(q_k + q_{k+1}) q_{k+1}. \end{aligned} \tag{4.51}$$

On the other hand, the contribution of transfer batches k and $k+1$ to the total flow time $F(q')$ in the new solution q' is

$$\begin{aligned} F(q'_k, q'_{k+1}) &= C'_{k,2} q'_k + C'_{k+1,2} q'_{k+1} \\ &= p_2 (q'_k)^2 + (q'_k + q'_{k+1}) C_{k-1,2} + p_2(q'_k + q'_{k+1}) q'_{k+1}. \end{aligned} \tag{4.52}$$

Substituting $q'_k = q_k + \varepsilon$ and $q'_{k+1} = q_{k+1} - \varepsilon$ into (4.52), we obtain

$$\begin{aligned} F(q'_k, q'_{k+1}) = {} & p_2 (q_k)^2 + (q_k + q_{k+1}) C_{k-1,2} + p_2(q_k + q_{k+1}) q_{k+1} \\ & + p_2(q_{k+1} - q_k)\varepsilon - p_2 \varepsilon^2. \end{aligned} \tag{4.53}$$

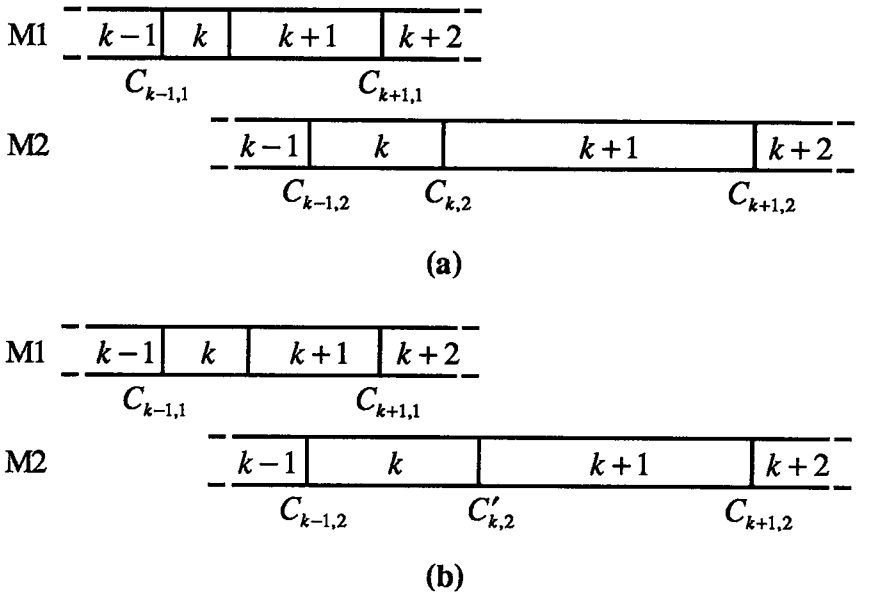


Figure 4.5. Transfer batch completion (a) in the solution $q=(q_1,q_2,\ldots,q_s)$, (b) in the solution $q'=(q_1,q_2,\ldots,q'_k,q'_{k+1},\ldots,q_s)$.

Then, from (4.51) and (4.53), we obtain

$$\begin{aligned} F(q_k,q_{k+1})-F(q'_k,q'_{k+1}) &= p_2(q_{k+1}-q_k)\varepsilon-p_2\varepsilon^2 \\ &= p_2(q_{k+1}-q_k-\varepsilon)\varepsilon. \end{aligned} \tag{4.54}$$

We know that $\alpha=p_2/p_1>1$ and $\alpha q_i<q_{i+1}$ in the solution q, then $q_i<q_{i+1}$. Therefore, it is clear that $F(q_k,q_{k+1})-F(q'_k,q'_{k+1})>0$ for some $\varepsilon>0$. Hence, $F(q)>F(q')$, which implies that the solution q can not be optimal.

Case 2: $C_{k+1,1}>C_{k,2}$.

This case corresponds to the idleness on machine 2 between processing of transfer batches k and $k+1$ in the solution q as illustrated by Figure 4.6(a). Using the corresponding new solution q' which is shown in Figure 4.6(b), the completion time of the transfer batch $k+1$ on machine 2 is written as

$$\begin{aligned} C'_{k+1,2} &= C'_{k+1,2}+p_2q'_{k+1} \\ &= C_{k+1,1}+p_2(q_{k+1}-\varepsilon) \\ &= C_{k+1,1}+p_2q_{k+1}-p_2\varepsilon \\ &= C_{k+1,2}-p_2\varepsilon \end{aligned} \tag{4.55}$$

since $C'_{k+1,1}=C_{k+1,1}$ and $C_{k+1,2}=C_{k+1,1}+p_2q_{k+1}$. Hence, the new solution q' is feasible.

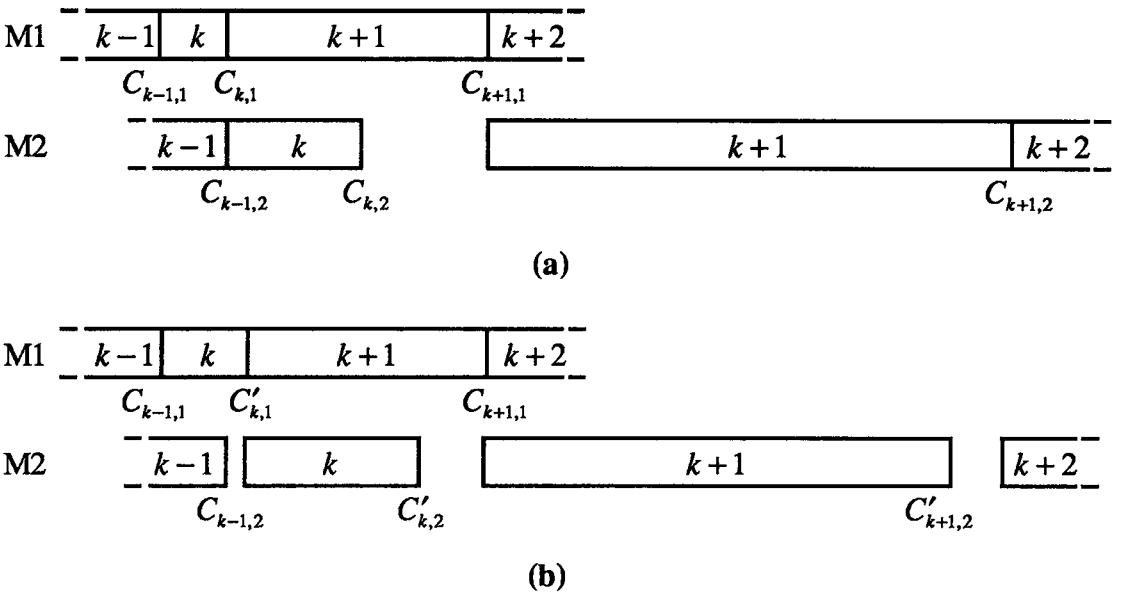


Figure 4.6. Transfer batch completion (a) in the solution $q=(q_1,q_2,\ldots,q_s)$, (b) in the solution $q'=(q_1,q_2,\ldots,q'_k,q'_{k+1},\ldots,q_s)$.

Similar to the first case, the contribution of transfer batches k and $k+1$ to the total flow time $F(q)$ in the solution q is expressed as

$$\begin{aligned}F(q_k,q_{k+1}) &= C_{k,2}q_k + C_{k+1,2}q_{k+1}\\ &= [C_{k-1,1}+(p_1+p_2)q_k]q_k + [C_{k+1,1}+p_2q_{k+1}]q_{k+1}\\ &= [C_{k-1,1}+(p_1+p_2)q_k]q_k + [C_{k-1,1}+p_1(q_k+q_{k+1})+p_2q_{k+1}]q_{k+1}\\ &= (p_1+p_2)(q_k)^2 + (q_k+q_{k+1})C_{k-1,1} + p_1(q_k+q_{k+1})q_{k+1} + p_2(q_{k+1})^2.\end{aligned} \tag{4.56}$$

On the other hand, the contribution of transfer batches k and $k+1$ to the total flow time $F(q')$ in the new solution q' is

$$\begin{aligned}F(q'_k,q'_{k+1}) &= C'_{k,2}q'_k + C'_{k+1,2}q'_{k+1}\\ &= (p_1+p_2)(q'_k)^2 + (q'_k+q'_{k+1})C_{k-1,1} + p_1(q'_k+q'_{k+1})q'_{k+1} + p_2(q'_{k+1})^2.\end{aligned} \tag{4.57}$$

Substituting $q'_k = q_k+\varepsilon$ and $q'_{k+1} = q_{k+1}-\varepsilon$ into (4.57), we obtain

$$\begin{aligned}F(q'_k,q'_{k+1}) &= (p_1+p_2)(q_k)^2 + (q_k+q_{k+1})C_{k-1,1} + p_1(q_k+q_{k+1})q_{k+1} + p_2(q_{k+1})^2\\ &\quad +(p_1+2p_2)(q_{k+1}-q_k)\varepsilon - (p_1+2p_2)\varepsilon^2.\end{aligned} \tag{4.58}$$

Then, from (4.56) and (4.57), we obtain

$$F(q_k, q_{k+1}) - F(q'_k, q'_{k+1}) = (p_1 + 2p_2)(q_{k+1} - q_k)\varepsilon - (p_1 + 2p_2)\varepsilon^2$$
$$= (p_1 + 2p_2)(q_{k+1} - q_k - \varepsilon)\varepsilon. \qquad (4.59)$$

We know that $\alpha = p_2 / p_1 > 1$ and $\alpha q_i < q_{i+1}$ in the solution q, then $q_i < q_{i+1}$. Therefore, it is clear that $F(q_k, q_{k+1}) - F(q'_k, q'_{k+1}) > 0$ for some $\varepsilon > 0$. Hence, $F(q) > F(q')$, which implies that the solution q can not be optimal.

Therefore, in an optimal schedule, $\alpha q_i \geq q_{i+1}$, $i = 2,...,s-1$. ∎

As an immediate result of Theorem 4.8, we can write the following corollary.

Corollary 4.2 *In the optimal solution of the problem* (P'), *if* $p_1 < p_2$ *and* $p_1 q_1^R < S'_2 < p_1(Q/s)$ *then there exists a transfer batch* k *such that*

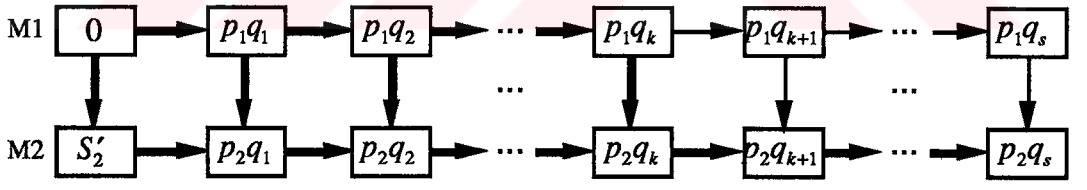
(i) $q_i = \alpha^{i-1} q_1$, $\quad i = 1,...,k$

(ii) $q_k \leq q_{k+1} \leq \alpha q_k$

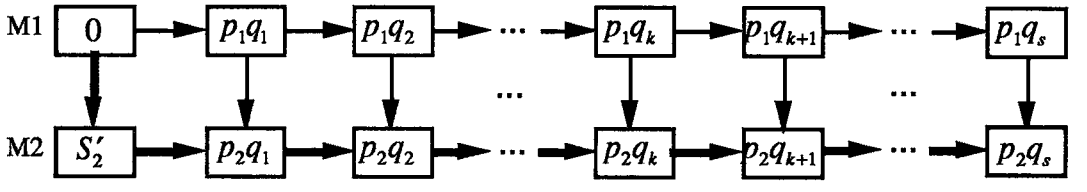
(iii) $q_i = q_{k+1}$, $\quad i = k+1,...,s$

where $\alpha = p_2 / p_1$ *and* $q_1 = S'_2 / p_1$.

Based on the developments above, we present the network of optimal transfer batch sizes for all cases of the problem (P') when $p_1 < p_2$ as shown in Figure 4.7 and an algorithm which determines the corresponding optimal transfer batch sizes.



(a) $p_1 q_1^R < S'_2 < p_1(Q/s)$



(b) $p_1(Q/s) \leq S'_2$

Figure 4.7. Network of optimal transfer batch sizes for $p_1 < p_2$, and $p_1 q_1^R < S'_2$.

Example 4.4

Suppose that a lot of 100 identical units will be split into 5 sublots, i.e., $Q = 100$, $s = 5$, and assume that the setup time on the first machine is less than the setup time on the second machine, i.e., $S_1' = 0$. Furthermore, the unit processing times on machine 1 and 2 are 1 and 2 time units, respectively, i.e., $p_1 = 1$, $p_2 = 2$. Then the optimal solution for the relaxed problem without any setups is obtained with transfer batch sizes $q_1^R = 6.59$, $q_2^R = 13.19$, $q_3^R = 26.37$, and $q_4^R = q_5^R = 26.92$.

Case a. Let $S_2' = 5$.

Since $S_2' = 5 < p_1 q_1^R = 6.59$, the solution to the relaxed problem is also optimal for this case as illustrated in Figure 4.8.

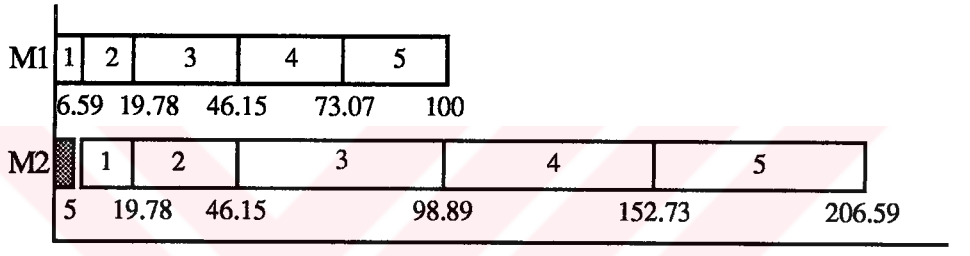


Figure 4.8. Optimal schedule in case (a) of Example 4.4.

Case b. Let $S_2' = 25$.

Since $S_2' = 25 > p_1(Q/s) = 20$, the solution obtained for Case (a) is no longer optimal. Note that $p_1(Q/s) = 20 > p_1 q_1^R = 6.59$. Then, from Theorem 4.7, the optimal solution is obtained with equal sized transfer batches as illustrated in Figure 4.9, i.e., $q_i = Q/s = 20$ for $1 \le i \le 5$.

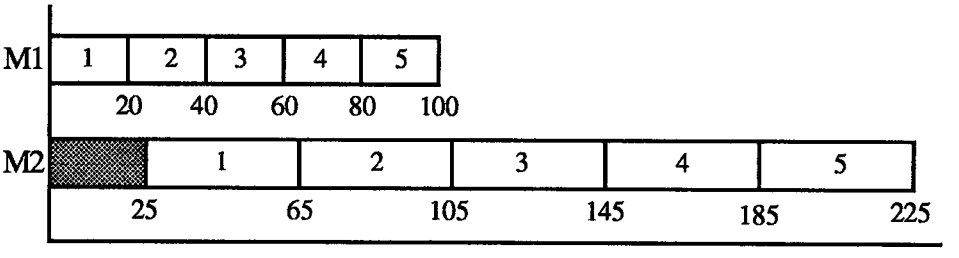


Figure 4.9. Optimal schedule in case (b) of Example 4.4.

Case c. Let $S_2' = 10$.

Since $p_1 q_1^R = 6.59 < S_2' < p_1(Q/s) = 20$, Corollary 4.2 gives the optimal solution in which transfer batch 3 is the pattern-changing transfer batch and, $q_1 = S_2'/p_1 = 10$, $q_2 = \alpha q_1 = 20$, $q_i = 23.33$ for $3 \le i \le 5$. The optimal solution for this case can be seen from Figure 4.10.

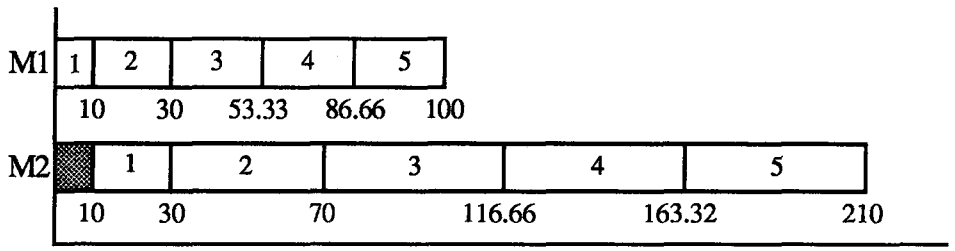


Figure 4.10. Optimal schedule in case (c) of Example 4.4.

Case 2 $p_1 \ge p_2$

Theorem 4.9 *In the optimal solution of the problem* (P'), *if* $p_1 \ge p_2$ *and* $S_2' \ge p_1 Q$ *then the transfer batch sizes are equal, i.e.*

$$q_i = \frac{Q}{s}, \qquad i = 1,2,\ldots,s.$$

Proof: The completion time of the transfer batch i on the second machine is given by

$$C_{i,2} = \sum_{u=1}^{i} (I_u + p_2 q_u) \tag{4.60}$$

where I_u is the idle time between processing of transfer batch u and $u-1$ on the second machine. Note that if $u=1$ then I_u is defined as the idle time between processing of the first transfer batch and the finishing of the setup on the second machine. Thus,

$$\sum_{i=1}^{s} q_i C_i = \sum_{i=1}^{s} q_i [\sum_{u=1}^{i} (I_u + p_2 q_u)] = \sum_{i=1}^{s} q_i \sum_{u=1}^{i} I_u + p_2 \sum_{i=1}^{s} q_i \sum_{u=1}^{i} q_u$$

$$= \sum_{i=1}^{s} q_i \sum_{u=1}^{i} I_u + (p_2/2) \sum_{i=1}^{s} q_i^2 + (p_2/2)(\sum_{i=1}^{s} q_i)^2. \tag{4.61}$$

It is clear that the second term in (4.61) is minimized if all transfer batches are equal sized, i.e., $q_i = \frac{Q}{s}$ for $1 \le i \le s$. On the other hand, the first term is minimized when the total idle time on the second machine is equal to zero, i.e., $\sum_{u=1}^{s} I_u = 0$. If $q_i = \frac{Q}{s}$ for $1 \le i \le s$, then $I_1 = 0$ and $\sum_{u=2}^{s} I_u = 0$ since $S_2' \ge p_1 Q > p_1 (Q/s)$ and $p_1 \ge p_2$, which implies that the first term in the equation above is also at its minimum. Therefore, it is obvious that $\sum_{i=1}^{s} q_i C_i$ is minimized when $q_i = \frac{Q}{s}$ for $1 \le i \le s$ (see Figure 4.11(a)). ∎

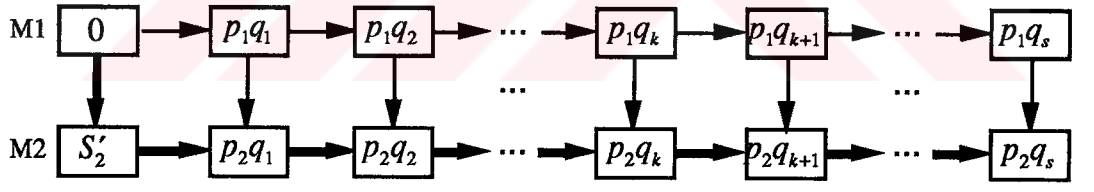
Conjecture 4.1 *In the optimal solution of the problem, if* $p_1 \ge p_2$ *and* $p_1 q_1^R < S_2' < p_1 Q$ *then there exists a transfer batch* k *such that*

(i) $q_i = q_{i-1}, \qquad i = 2,\ldots,k$

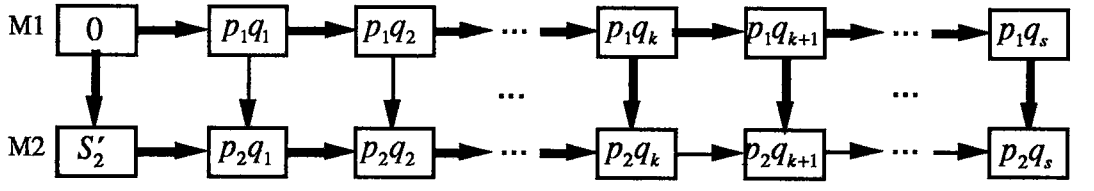
(ii) $\alpha q_k \ge q_{k+1} \ge q_k$

(iii) $q_i = q_{k+1}, \qquad i = k+1,\ldots,s$

where $\alpha = p_2 / p_1$ *and* $q_k = \frac{S_2 - S_1}{p_1} + (k-1)(\alpha - 1) q_1$ *(see Figure 4.11(b)).*



(a) $p_1 Q \le S_2'$



(b) $p_1 q_1^R \le S_2' < p_1 Q$

Figure 4.11. Network of optimal transfer batch sizes for $p_1 \ge p_2$ and $p_1 q_1^R < S_2'$.

4.4 Three-machine Problem

In this section, we analyze the characteristics of the lot streaming schedules for which the total flow time of transfer batches for a single job in a three-machine flow shop by extending the results of Çetinkaya and Gupta (1994) and Şen *et al.* (1994).

Following the approach of Glass *et. al* (1994), three-machine s-sublot problem $1|F3|TB(C,c)|\sum q_iC_i$ can be represented by a network shown in Figure 4.12. In the network, node (i,m) represents the processing time of transfer batch i on machine m. Also, the arcs in the network represent the precedence relationships between transfer batches and between machines. That is, the directed arc from node (i,m) to $(i+1,m)$ represents the constraint that machine m can not start to process transfer batch $i+1$ until it has completed transfer batch i. The directed arc from node (i,m) to $(i,m+1)$ represents the constraint that transfer batch i can not be processed on machine $m+1$ until it is completed on machine m.

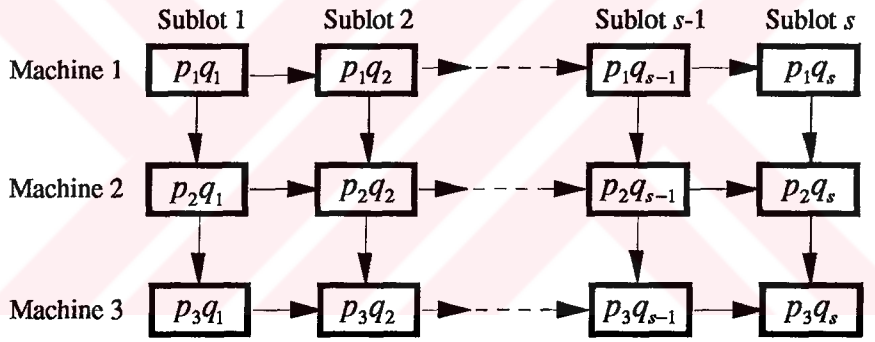


Figure 4.12. Network representation of three-machine problem

Definition 4.3 Machine (stage) m is *critical from transfer batch* i $(1 \le i \le s-1)$ if the arc from (i,m) to $(i+1,m)$ is a critical segment.

Definition 4.4 Transfer batch i is *critical* if the arc from $(i,m)$to $(i,m+1)$is a critical segment for some machine m $(1 \le m \le 2)$.

Theorem 4.10 *In the optimal solution of the* $1|F3|TB(C,c)|\sum q_iC_i$ *problem, there exists a transfer batch* k *such that machine with the longest processing time stays critical for transfer batches* $k,k+1,\ldots,s$.

Proof: Consider an optimal solution. Let the machine with the longest processing time, p^*, be critical from a transfer batch k (existence of such a k is obvious from Definition 4.3). Proof is trivial if $p^* = 3$.

If $p^* = 1$, then,

$$p_1 \sum_{i=k+1}^{s} q_i + p_2 q_s > p_2 \sum_{i=k}^{l} q_i + p_3 \sum_{i=l}^{s-1} q_i \qquad \text{for any } l,\ k < l \le s-1$$

since $p_1 \sum_{i=k+1}^{l+1} q_i \ge p_2 \sum_{i=k}^{l} q_i$ and $p_1 \sum_{i=l+2}^{s} q_i + p_2 q_s > p_3 \sum_{i=l}^{s-1} q_i$.

If $p^* = 2$, then

$$p_2 \sum_{i=k+1}^{s} q_i > p_2 \sum_{i=k+1}^{l} q_i + p_3 \sum_{i=l}^{s-1} q_i \qquad \text{for any } l,\ k < l \le s-1$$

since $p_2 \sum_{i=k+1}^{l} q_i$ is common on both sides and $p_2 \sum_{i=l+1}^{s} q_i > p_3 \sum_{i=l}^{s-1} q_i$. ∎

From this theorem, we immediately get the following corollary.

Corollary 4.3 *In the optimal solution of the* $1|F3|TB(C,c)|\sum q_i C_i$ *problem, machine with the longest processing time,* p^*, *operates with no idling from a transfer batch* k *that is critical up to the last transfer batch* s.

Therefore, the longest path contains

$(k,p^*)-(k+1,p^*)-\ldots-(s,p^*)-\ldots-(s,m)$ if $p^* \neq m$,

and $(k,m)-(k+1,m)-\ldots-(s,m)$ if $p^* = m = 3$

where k is a transfer batch that p^* is critical from.

Lemma 4.10 *In the* $1|F3|TB(C,c)|\sum q_i C_i$ *problem, if the last stage has the longest processing time, the idleness between any pair of transfer batches* $r-1$ *and* r *can be removed without violating the nondecreasing property of transfer batches.*

Proof: Without loss of generality, let the earliest (i.e., leftmost occurrence) idleness occurs between transfer batches $r-1$ and r. This means

$$(p_1 + p_2) q_1 + p_3 \sum_{i=1}^{j} q_i \ge L_{j+1,2} \qquad \text{for } j = 1,2,\ldots,r-2$$

and

$$(p_1 + p_2) q_1 + p_3 \sum_{i=1}^{r-1} q_i < L_{r,2}$$

where $L_{i,j}$ is the longest path (1,1)-...-(i, j).

There are three alternative cases for the longest path $L_{r,2}$.

Case 1: $L_{r,2} = L_{r-1,1} + (p_1 + p_2)q_r$.

For idleness on machine 3 to be the earliest time on machine 3,

$$q_r > \frac{p_3 + p_2}{p_1 + p_2} q_{r-1}$$

must hold since $L_{r-1,1} + p_2 q_{r-1} \le (p_1 + p_2)q_1 + p_3 \sum_{j=1}^{r-2} q_j$ as no idleness occurs before transfer batch $r-1$.

Let us define new transfer batch sizes as $q'_{r-1} = q_{r-1} + \Delta_1$ and $q'_r = q_r - \Delta_1$, $\Delta_1 > 0$. Equating longest paths for no idleness on machine 3 with these new transfer batch sizes, we obtain

$$L_{r-1,1} + (p_1 + p_2)q_r + p_1\Delta_1 - (p_1 + p_2)\Delta_1 = (p_1 + p_2)q_1 + p_3 \sum_{j=1}^{r-1} q_j + p_3\Delta_1$$

$$L_{r-1,1} + (p_1 + p_2)q_r - (p_1 + p_2)q_1 - p_3 \sum_{j=1}^{r-1} q_j = (p_3 + p_2)\Delta_1$$

$$\Delta_1 = \frac{L_{r-1,1} + (p_1 + p_2)q_r - (p_1 + p_2)q_1 - p_3 \sum_{j=1}^{r-1} q_j}{p_3 + p_2}$$

$$= \frac{L_{r,2} - (p_1 + p_2)q_1 - p_3 \sum_{j=1}^{r-1} q_j}{p_3 + p_2} \le \frac{(p_1 + p_2)q_r - (p_3 + p_2)q_{r-1}}{p_3 + p_2}$$

Then we need to check whether $q'_r \ge q'_{r-1}$. i.e.,

$$q_r - \frac{p_1 + p_2}{p_3 + p_2} q_r \ge \frac{p_1 + p_2}{p_3 + p_2} q_r - q_{r-1},$$

or equivalently,

$$q_r - \frac{2(p_1 + p_2)}{p_3 + p_2} q_r \ge -q_{r-1}.$$

Substituting $\frac{p_3 + p_2}{p_1 + p_2} q_{r-1}$ for q_r (which was shown before to be at most equal to q_r) in the second term on the left hand side of the above inequality yields $q_r \ge q_{r-1}$. Therefore, the property of nondecreasing transfer batch sizes is not violated when Δ_1 is used to update the two transfer batches q_{r-1} and q_r. This case has an exception

where the longest path changes after q'_{r-1} and q'_r replaces q_{r-1} and q_r, respectively. This will be studied later as an occurrence of Case 2.

Case 2: $L_{r,2} = L_{r-1,2} + p_2 q_r$.

For idleness on machine 3 to be the earliest time on machine 3,

$$q_r > \frac{p_3}{p_2} q_{r-1}$$

must hold since $L_{r-1,2} < (p_1 + p_2)q_1 + p_3 \sum_{j=1}^{r-2} q_j$ as no idleness occurs before transfer batch $r-1$.

Let us define new transfer batch sizes as $q'_{r-1} = q_{r-1} + \Delta_2$ and $q'_r = q_r - \Delta_2$, $\Delta_2 > 0$. Equating longest paths for reduced idleness on machine 3 with these new transfer batch sizes, we obtain

$$L_{r-1,1} + p_2(q_{r-1} + q_r) + p_1\Delta_2 + p_2\Delta_2 - p_2\Delta_2 = (p_1 + p_2)q_1 + p_3 \sum_{j=1}^{r-1} q_j + p_3\Delta_2$$

$$\Delta_2 = \frac{L_{r-1,1} + p_2(q_{r-1} + q_r) - (p_1 + p_2)q_1 - p_3 \sum_{j=1}^{r-1} q_j}{p_3 - p_1} \leq \frac{p_2 q_r - p_3 q_{r-1}}{p_3 - p_1}$$

Then we need to check whether $q'_r \geq q'_{r-1}$. i.e.,

$$q_r - \frac{p_2}{p_3 - p_1} q_r + \frac{p_3}{p_3 - p_1} q_{r-1} \geq q_{r-1} + \frac{p_2}{p_3 - p_1} q_r - \frac{p_3}{p_3 - p_1} q_{r-1}$$

or equivalently,

$$q_r - \frac{2p_2}{p_3 - p_1} q_r \geq q_{r-1} - \frac{2p_3}{p_3 - p_1} q_{r-1}.$$

Substituting $\frac{p_3}{p_2} q_{r-1}$ for q_r (which was shown before to be at most equal to q_r) in the second term on the left hand side of the above inequality yields $q_r \geq q_{r-1}$. Therefore, the property of nondecreasing transfer batch sizes is not violated when Δ_2 is used to update the two transfer batches q_{r-1} and q_r.

If $L_{r,2} = L_{r-1,1} + (p_1 + p_2)q_r$ (Case 1)

but $L'_{r,2} = L'_{r-1,1} + p_2(q'_{r-1} + q'_r) = L'_{r-2,1} + p_1 q'_{r-1} + p_2(q'_{r-1} + q'_r)$ (Case 2)

where $L'_{i,j}$ is the longest path length with $q'_{r-1} = q_{r-1} + \Delta_1$ and $q'_r = q_r + \Delta_1$ used.

$$L'_{r,2} = L_{r-2,1} + p_1 q_{r-1} + p_1\Delta_1 + p_2(q_{r-1} + \Delta_1 + q_r - \Delta_1)$$
$$= L_{r-2,1} + p_1 q_{r-1} + p_1\Delta_1 + p_2(q_{r-1} + q_r)$$
$$= L_{r-2,1} + p_1 q_{r-1} + p_2 q_{r-1} + p_2 q_r + p_1\Delta_1.$$

Thus, after $q'_{r-1} = q_{r-1} + \Delta_1$ and $q'_r = q_r - \Delta_1$.
although

$$(p_1 + p_2)q_1 + p_3\sum_{i=1}^{r-2} q_i + p_3 q'_{r-1} = L'_{r,1} + p_2 q'_r$$

it may still be true that $(p_1 + p_2)q_1 + p_3\sum_{i=1}^{r-2} q_i + p_3 q'_{r-1} < L'_{r,2}$, i.e., less amount of idleness remains between the same transfer batches. Then

$$q''_{r-1} = q'_{r-1} + \Delta_2,$$
$$q''_r = q'_r - \Delta_2,$$
$$\Delta_2 > 0$$

can be made and the remaining idleness can be removed as in Case 2.

Case 3: $L_{r,2} = L_{r-2,2} + p_2(q_{r-1} + q_r)$.

For idleness on machine 3 to be the first time on machine 3,

$$q_r > \frac{p_3}{p_2} q_{r-1}$$

must hold similar to Case 2.

Equating longest paths for reduced idleness on machine 3 with these new transfer batch sizes, we obtain

$$L_{r-2,2} + p_2(q_{r-1} + q_r) + p_2\Delta_3 - p_2\Delta_3 = (p_1 + p_2)q_1 + p_3\sum_{j=1}^{r-1} q_j + p_3\Delta_3$$

$$\Delta_3 = \frac{L_{r,2} - (p_1 + p_2)q_1 - p_3\sum_{j=1}^{r-1} q_j}{p_3} \le \frac{p_2 q_r - p_3 q_{r-1}}{p_3}$$

Let $q'_{r-1} = q_{r-1} + \Delta_3$ and $q'_r = q_r - \Delta_3$, $\Delta_3 > 0$. Then we need to check whether $q'_r \ge q'_{r-1}$. i.e.,

$$q_r - \frac{p_2}{p_3} q_r + q_{r-1} \ge q_{r-1} + \frac{p_2}{p_3} q_r - q_{r-1},$$

or equivalently,

$$q_r \ge \frac{2p_2}{p_3} q_r - q_{r-1}.$$

Similar to before it can be shown that $q_r \geq q_{r-1}$. Therefore, the property of nondecreasing transfer batch sizes is not violated if Δ_3 is used to update the two transfer batches q_{r-1} and q_r. The proof is complete. ∎

Theorem 4.11 *In the optimal solution of the* $1|F3|TB(C,c)|\sum q_i C_i$ *problem, machine with the longest processing time,* p^**, operates with no idling from transfer batch 1 through the first transfer batch* k *that it is critical from.*

Proof: Trivial if $p^* = 1$. Take $p^* = 2$. Assume that earliest idling on machine 2 before the first transfer batch that, p^* is critical, is between transfer batches $r-1$ and r. Then it is true that

$$p_2 \sum_{i=1}^{r-1} q_i < p_1 \sum_{i=2}^{r} q_i .$$

But we have

$$p_2 \sum_{i=1}^{j} q_i \geq p_1 \sum_{i=2}^{j+1} q_i , \qquad j = 1,2,\ldots,r-2.$$

Thus we obtain $p_2 q_{r-1} < p_1 q_r$ (or equivalently, $q_r > \frac{p_2}{p_1} q_{r-1}$), and the total flow time can be written as

$$F = F_{r-2}^{-} + [p_1 q_1 + p_2 \sum_{i=1}^{r-2} q_i + (p_2 + p_3) q_{r-1}] q_{r-1} + [p_1 \sum_{i=1}^{r} q_i + (p_2 + p_3) q_r] q_r + F_{r+1}^{+}$$

where $F_j^{-} = \sum_{i=1}^{j} q_i C_i$ and $F_j^{+} = \sum_{i=j}^{s} q_i C_i$.

Now, let us define Δ such that, $p_2 \sum_{i=1}^{r-1} q_i + \Delta = p_1 \sum_{i=2}^{r} q_i$ and update transfer batches q_{r-1} and q_r as $q_r' = q_r - \frac{\Delta}{p_2}$ and $q_{r-1}' = q_{r-1} + \frac{\Delta}{p_2}$, respectively.

This update does not change the sum on the right and it still preserves

$$p_1 \sum_{i=2}^{j} q_i > p_2 \sum_{i=1}^{j-1} q_i \quad \text{for } j \geq r+1;$$

but makes

$$p_1 \sum_{i=2}^{r-2} q_i + p_1 q_{r-1}' + p_1 q_r' = p_2 \sum_{i=1}^{r-2} q_i + p_2 q_{r-1}' .$$

Hence, it removes the earliest idleness on machine 2. Moreover, it improves F since, F^-_{r-2} and F^+_{r+1} are not affected; but,

$$[p_1q_1 + p_2\sum_{i=1}^{r-2} q_i + (p_2+p_3)q_{r-1})]\frac{\Delta}{p_2} < [p_1\sum_{i=1}^{r} q_i + (p_2+p_3)q_r)]\frac{\Delta}{p_2}.$$

Therefore, the first idling between $r-1$ and r can be removed to improve total flow time. Then, the next idling (if any) can be removed as it becomes the first idling. Similarly, all idling in stage 2 before transfer batch k can be removed to improve total flow time.

Now, take $p^* = 3$. Suppose the earliest idleness on machine 3 occurs between transfer batches $r-1$ and r where $r < k$ and r is the first transfer batch machine 3 is critical from. Then there are three cases as in Lemma 4.10.

Case 1: $L_{r,2} = L_{r-1,1} + (p_1+p_2)q_r$.

To reduce idleness between $r-1$ and r

$$\Delta_1 = \frac{L_{r-1,1} + (p_1+p_2)q_r - (p_1+p_2)q_1 - p_3\sum_{j=1}^{r-1} q_j}{p_3+p_2}$$

should be used. Let us compute the change in the total flow time, ΔF, after the update to reduce the earliest idleness at the third stage.

$$\begin{aligned}
\Delta F &= [(p_1+p_2)q_1 + p_3\sum_{j=1}^{r-1} q_j + p_3\Delta_1](q_{r-1}+\Delta_1) - [(p_1+p_2)q_1 + p_3\sum_{j=1}^{r-1} q_j]q_{r-1} \\
&= [p_3q_{r-1} + (p_1+p_2)q_1 + p_3\sum_{j=1}^{r-1} q_j + p_3\Delta_1]\Delta_1 \\
&= [p_3(q_{r-1}+\Delta_1) - p_2q_r - 2p_3q_r]\Delta_1 \\
&< 0
\end{aligned}$$

since $q_{r-1} + \Delta_1 < q_r$ by Lemma 4.10. Therefore, total flow time improves by removing the earliest idleness.

After a Case 1 application for the update, the longest path $L_{r,2}$ may change to $L'_{r,2} = L'_{r-1,1} + p_2(q'_{r-1} + q'_r)$ and there may still remain idleness between transfer batches $r-1$ and r on the third machine. This may be removed by an application of Case 2 and it will be shown to improve total flow time as well.

Case 2: $L_{r,2} = L_{r-1,2} + p_2 q_r$.

To remove idleness between $r-1$ and r

$$\Delta_2 = \frac{L_{r-1,1} + p_2(q_{r-1} + q_r) - (p_1 + p_2)q_1 - p_3 \sum_{j=1}^{r-1} q_j}{p_3 - p_1}$$

should be used. Let us compute the change in the total flow time, ΔF, after the update to remove the earliest idleness at the third stage.

$$\begin{aligned}\Delta F &= [p_3 q_{r-1} + (p_1 + p_2)q_1 + p_3 \sum_{j=1}^{r-1} q_j + p_3 \Delta_2]\Delta_2 \\ &\quad + [(p_1 q_r - p_3 q_r - L_{r-1,1} - p_1 \Delta_2 - p_2(q_{r-1} + q_r) - p_3(q_r - \Delta_2)]\Delta_2 \\ &= p_3(q_{r-1} + \Delta_2) + p_1 q_r - 2p_3 q_r \\ &< 0\end{aligned}$$

since $p_1 < p_3$ and $q_{r-1} + \Delta_2 < q_r$ by Lemma 4.10. Therefore, total flow time improves by removing the earliest idleness.

Case 3: $L_{r,2} = L_{r-2,2} + p_2(q_{r-1} + q_r)$.

To reduce idleness between $r-1$ and r

$$\Delta_3 = \frac{L_{r,2} - (p_1 + p_2)q_1 - p_3 \sum_{j=1}^{r-1} q_j}{p_3}$$

should be used. Let us compute the change in the total flow time, ΔF, after the update to reduce the earliest idleness at the third stage.

$$\begin{aligned}\Delta F &= [p_3 q_{r-1} + (p_1 + p_2)q_1 + p_3 \sum_{j=1}^{r-1} q_j + p_3 \Delta_3]\Delta_3 \\ &\quad + [-p_3 q_r - L_{r-2,2} - p_2(q_{r-1} + q_r) - p_3(q_r - \Delta_3)]\Delta_3 \\ &= p_3(q_{r-1} + \Delta_3) - 2p_3 q_r < 0\end{aligned}$$

since $p_1 < p_3$ and $q_{r-1} + \Delta_3 < q_r$ by Lemma 4.10. Therefore, total flow time improves by removing the earliest idleness.

After a Case 3 application for the update, the longest path $L_{r,2}$ may change to $L'_{r,2} = L'_{r-1,1} + p_2(q'_{r-1} + q'_r)$ and there may still remain idleness between transfer batches $r-1$ and r on the third machine. This may be removed by an application of Case 2 and it was shown to improve total flow time as well. ∎

Corollary 4.4 *In the optimal solution of the* $1|F3|TB(C,c)|\sum q_iC_i$ *problem, machine with the longest processing time operates with no-idling.*

Proof: Result of Theorem 4.10 and 4.11.

Theorem 4.12 *In the optimal solution of the* $1|F3|TB(C,c)|\sum q_iC_i$ *problem, the longest path to the completion of any transfer batch* i *is such as to make*

either *(i) all machines are critical from transfer batch* i,

or *(ii) all transfer batches are critical from machine 3.*

Proof: As it is defined before let p^* be the machine with longest processing time.

Case 1: $p^* = 1$.

The longest path length for transfer batch s is given by

$$L_{s,3} = p_1 \sum_{i=1}^{s} q_i + (p_2 + p_3) q_s .$$

Suppose that

$$L_{j,3} = p_1 \sum_{i=1}^{k} q_i + p_2 \sum_{i=k}^{l} q_i + p_3 \sum_{i=l}^{j} q_i , \qquad k \le l \le j \text{ with either } k \ne l \text{ or } l \ne j.$$

Then we have

$$p_2 \sum_{i=k}^{l-1} q_i + p_2 q_l + p_3 \sum_{i=l}^{j-1} q_i > p_1 \sum_{i=k+1}^{l} q_i + p_2 q_j + p_1 \sum_{i=l+1}^{j} q_i$$

which is a contradiction since

$$p_1 \sum_{i=k+1}^{l} q_i > p_2 \sum_{i=k}^{l-1} q_i \qquad \text{if } p^* = 1,$$

$$p_2 q_j > p_2 q_l \qquad \text{if } l < j,$$

$$p_1 \sum_{i=l+1}^{j} q_i > p_3 \sum_{i=l}^{j-1} q_i \qquad \text{if } p^* = 1.$$

Case 2: $p^* = 2$.

The longest path length for transfer batch s is given by

$$L_{s,3} = p_1 q_1 + p_2 \sum_{i=1}^{s} q_i + p_3 q_s .$$

Suppose that

$$L_{j,3} = p_1 q_1 + p_2 \sum_{i=1}^{l} q_i + p_3 \sum_{i=l}^{j} q_i , \quad l < j.$$

Then we have

$$p_3 \sum_{i=l}^{j-1} q_i > p_2 \sum_{i=l+1}^{j} q_i$$

which is a contradiction since $p^* = 2$.

Case 3: $p^* = 3$.

Proof is trivial due to Theorem 4.11 (all transfer batches are critical from machine 3). ∎

Theorem 4.13 *In the* $1|F3|TB(C,c)|\sum q_i C_i$ *problem, if the stage with the longest processing time is the last stage* ($p^* = 3$), *then in the optimal solution, either stage 2 stays critical from transfer batch 1 through a transfer batch* k ($1 \le k \le s$) *or stage 1 stays critical in a similar manner. In other words, it is necessary to have no-wait and no-idling on machine 2 between transfer batches 1 and a transfer batch* k *in an optimal solution. If* $p^* = 2$, *then in the optimal solution, stage 1 stays critical from transfer batch 1 through a transfer batch* k ($1 \le k \le s$).

Proof:

Case 1: $p^* = 3$ and $p_2^2 > p_1 p_3$.

Let all transfer batches be arranged in an arbitrary nondecreasing order. It is sufficient to show that total flow time does not increase, making stage 2 stay critical between transfer batch 1 and a transfer batch k ($1 \le k \le s$) while nondecreasing transfer batch sizes are preserved.

Suppose initially transfer batch sizes are updated such as to produce no idling on machine 3. This can be done without violating the nondecreasing transfer batch sizes, as shown in Theorem 4.11 and Corollary 4.4.

We will first show that idling on machine 2, if any, can be removed up to a transfer batch k' without increasing the total flow time.

Let l be the earliest transfer batch with delay on machine 2, then

$$p_1 q_1 + p_2 \sum_{i=1}^{l} q_i < p_1 \sum_{i=1}^{l} q_i + p_2 q_l.$$

Let q_{l-1} and q_l be updated as $q'_{l-1} = q_{l-1} + \Delta$ and $q'_l = q_l - \Delta$, $\Delta > 0$. For no delay, the following must hold

$$\Delta = \frac{p_1}{p_2} \sum_{i=2}^{l} q_i - \sum_{i=1}^{l-1} q_i.$$

Since all delay on machine 3 was removed

$$p_1 q_1 + p_2 \sum_{i=1}^{l} q_i \le p_1 q_1 + p_2 q_1 + p_3 \sum_{i=1}^{l-1} q_i$$

after the update

$$p_1q_1 + p_2\sum_{i=1}^{l} q_i \le p_1q_1 + p_2q_1 + p_3\sum_{i=1}^{l-2} q_i + p_3(q_{l-1} + \Delta)$$

still holds (i.e., any waiting remains).

Let C_i be the completion time of transfer batch i on the last machine. Assuming no delay on machine 2 for the previous transfer batch $(l-1)$ and a possible wait, change in total flow time due to the update is

$$\begin{aligned}
\Delta F &= C'_{l-1}\, q'_{l-1} + C'_l\, q'_l - C_{l-1}\, q_{l-1} - C_l\, q_l \\
&= [p_1q_1 + p_2q_1 + p_3\sum_{i=1}^{l-1} q_i + p_3\Delta](q_{l-1} + \Delta) + [p_1q_1 + p_2q_1 + p_3\sum_{i=1}^{l} q_i](q_l - \Delta) \\
&\quad -[p_1q_1 + p_2q_1 + p_3\sum_{i=1}^{l-1} q_i]q_{l-1} - [p_1q_1 + p_2q_1 + p_3\sum_{i=1}^{l} q_i]q_l \\
&= \Delta(p_3q_{l-1} + p_1q_1 + p_2q_1 + p_3\sum_{i=1}^{l-1} q_i + p_3\Delta - p_1q_1 - p_2q_1 - p_3\sum_{i=1}^{l} q_i) \\
&= \Delta[p_3(q_{l-1} + \Delta) - p_3q_l] \\
&\le 0
\end{aligned}$$

since $q_{l-1} + \Delta \le q_l$, otherwise $k' = l-1$ and the update would be terminated.

We will next show that wait on machine 2, if any, can be removed up to a transfer batch $k \le k'-1$ without increasing the total flow time.

Let l $(2 \le l \le k'-1)$ be the earliest transfer batch with wait on machine 2, then

$$p_1q_1 + p_2\sum_{i=1}^{l} q_i < p_1q_1 + p_2q_1 + p_3\sum_{i=1}^{l-1} q_i .$$

Let q_l and q_{l+1} be updated as $q'_l = q_l + \Delta$ and $q'_{l+1} = q_{l+1} - \Delta$, $\Delta > 0$. For no wait, the following must hold

$$\Delta = \frac{p_3}{p_2}\sum_{i=1}^{l-1} q_i - \sum_{i=2}^{l} q_i .$$

Since all delay on machine 2 was removed

$$p_1q_1 + p_2\sum_{i=1}^{l} q_i > p_1\sum_{i=1}^{l} q_i + p_2q_l$$

after the update

$$\begin{aligned}
p_1q_1 + p_2\sum_{i=1}^{l} q_i + p_2\Delta - p_1\sum_{i=1}^{l} q_i - p_1\Delta - p_2q_l - p_2\Delta &= p_2\sum_{i=1}^{l-1} q_i - p_1\sum_{i=2}^{l} q_i - p_1\Delta \\
&= p_2\sum_{i=1}^{l-1} q_i - p_1\sum_{i=2}^{l} q_i - p_1(\frac{p_3}{p_2}\sum_{i=1}^{l-1} q_i - \sum_{i=2}^{l} q_i)
\end{aligned}$$

$$= p_2 \sum_{i=1}^{l-1} q_i - (\frac{p_1 p_3}{p_2}) \sum_{i=1}^{l-1} q_i$$
$$= (\frac{p_2^2 - p_1 p_3}{p_2}) \sum_{i=1}^{l-1} q_i$$
$$> 0$$

since $p_2^2 > p_1 p_3$.

Therefore, no delay is introduced on machine 2. With delays removed from stage 2 up to transfer batch k' and a Δ such as to make $q_l' \le q_{l+1}'$ (otherwise, $k = l-1$ and removing wait should be terminated). Hence, change in total flow time due to the update is

$$\Delta F = C_l' q_l' + C_{l+1}' q_{l+1}' - C_l q_l - C_{l+1} q_{l+1}$$
$$= [p_1 q_1 + p_2 q_1 + p_3 \sum_{i=1}^{l} q_i + p_3 \Delta](q_l + \Delta) + [p_1 q_1 + p_2 q_1 + p_3 \sum_{i=1}^{l+1} q_i](q_{l+1} - \Delta)$$
$$- [p_1 q_1 + p_2 q_1 + p_3 \sum_{i=1}^{l} q_i] q_l - [p_1 q_1 + p_2 q_1 + p_3 \sum_{i=1}^{l+1} q_i] q_{l+1}$$
$$= \Delta(p_3 q_l + p_1 q_1 + p_2 q_1 + p_3 \sum_{i=1}^{l} q_i + p_3 \Delta - p_1 q_1 - p_2 q_1 - p_3 \sum_{i=1}^{l+1} q_i)$$
$$= \Delta[p_3(q_l + \Delta) - p_3 q_{l+1}]$$
$$\le 0$$

since $q_l + \Delta \le q_{l+1}$, otherwise $k = l$ and the update would be terminated.

Case 2: $p^* = 3$ and $p_2^2 \le p_1 p_3$.

Let all transfer batches be arranged in an arbitrary nondecreasing order. It is sufficient to show that total flow time does not increase, making stage 2 stay critical between transfer batch 1 and a transfer batch k $(1 \le k \le s)$ while nondecreasing transfer batch sizes are preserved.

Suppose initially transfer batch sizes are updated such as to produce no idling on machine 3. This can be done without violating the nondecreasing transfer batch sizes, as shown in Theorem 4.11 and Corollary 4.4.

We will first show that wait on machine 1, if any, can be removed starting from the earliest occurrence up to a transfer batch k without increasing the total flow time.

Let l $(2 \le l \le s-1)$ be the earliest transfer batch with wait on machine 1, then

$$p_1 \sum_{i=1}^{l} q_i + p_2 q_l < p_1 \sum_{i=1}^{l-1} q_i + p_2 q_{l-1} + p_2 q_l.$$

Let q_l and q_{l+1} be updated as $q'_l = q_l + \Delta$ and $q'_{l+1} = q_{l+1} - \Delta$, $\Delta > 0$. For no-wait on machine 1, the following must hold

$$p_1 \sum_{i=1}^{l} q_i + p_2 q_l + (p_1 + p_2)\Delta \geq p_1 \sum_{i=1}^{l-1} q_i + p_2 q_{l-1} + p_2 q_l + p_2 \Delta$$

$$p_1 \Delta \geq p_2 q_{l-1} - p_1 q_l$$

$$\Delta \geq \frac{p_2}{p_1} q_{l-1} - q_l.$$

This update may induce delay on machine 3 between transfer batches l and $l+1$ which may require another incrementing of transfer batch l at the expense of decrementing transfer batch $l+1$. However, these secondary updates, if any, does not induce wait for transfer batch l on machine 1 (According to Theorem 4.10 Case 1 for $p^* = 3$ with $r-1 = l$) as transfer batch l gets even larger than needed.

Now, let us express the change in total flow time:

$$\Delta F = C'_l q'_l + C'_{l+1} q'_{l+1} - C_l q_l - C_{l+1} q_{l+1}.$$

Assuming delay is induced on machine 3 between transfer batches l and $l+1$, after the update

$$\Delta F = [p_1 q_1 + p_2 q_1 + p_3 \sum_{i=1}^{l} q_i + p_3 \Delta](q_l + \Delta)$$

$$+ [p_1 \sum_{i=1}^{l} q_i + p_2 (q_l + q_{l+1}) + p_3 q_{l+1} + p_1 \Delta - p_3 \Delta](q_{l+1} - \Delta)$$

$$- [p_1 q_1 + p_2 q_1 + p_3 \sum_{i=1}^{l} q_i] q_l - [p_1 q_1 + p_2 q_1 + p_3 \sum_{i=1}^{l+1} q_i] q_{l+1}.$$

Dividing ΔF by Δ, as $p_1 q_1 + p_2 q_1 + p_3 \sum_{i=1}^{l+1} q_i \geq p_1 \sum_{i=1}^{l} q_i + p_2 (q_l + q_{l+1}) + p_3 q_{l+1}$

$$\frac{\Delta F}{\Delta} \leq p_3 q_l + p_1 q_1 + p_2 q_1 + p_3 \sum_{i=1}^{l} q_i + p_3 \Delta + p_1 q_{l+1} - p_3 q_{l+1}$$

$$- p_1 \sum_{i=1}^{l} q_i - p_2 (q_l + q_{l+1}) - p_3 q_{l+1} - p_1 \Delta + p_3 \Delta$$

$$\frac{\Delta F}{\Delta} \leq p_3 q_l + p_3 \Delta - 2 p_3 q_{l+1} + 2 p_3 \Delta + p_1 q_{l+1} - p_1 \Delta$$

as $p_1 \sum_{i=1}^{l} q_i + p_2 (q_l + q_{l+1}) \geq p_1 q_1 + p_2 q_1 + p_3 \sum_{i=1}^{l} q_i$

$$\frac{\Delta F}{\Delta} \leq p_3 (q_l + \Delta) - 2 p_3 (q_{l+1} + \Delta) + p_1 (q_{l+1} - \Delta)$$

$$\frac{\Delta F}{\Delta} \leq p_3 (q_l - q_{l+1}) + q_{l+1} (p_1 - p_3)$$

$$\Delta F \leq 0$$

since $q_{l+1} \geq q_l$ and $p_3 \geq p_1$. Therefore, the total flow time does not increase after the update.

Assuming no-delay induced on machine 3 between transfer batches l and $l+1$,

$$C'_{l+1} = p_1 q_1 + p_2 q_1 + p_3 \sum_{i=1}^{l+1} q_i$$

Hence,

$$\frac{\Delta F}{\Delta} = p_3 q_l + p_1 q_1 + p_2 q_1 + p_3 \sum_{i=1}^{l} q_i - p_1 q_1 - p_2 q_1 - p_3 \sum_{i=1}^{l+1} q_i$$

$$= p_3 (q_l - q_{l+1} + \Delta)$$

$$\Delta F \leq 0$$

since $q_{l+1} - q_l \geq \Delta$, otherwise transfer batch sizes will not be nondecreasing in size.

In all the above if the update q'_l and q'_{l+1} results in violating the nondecreasing transfer batch sizes at any l, it means the last transfer batch for no wait on machine 1 is arrived, and $k = l-1$.

Case 3: $p^* = 2$.

If $p^* = 2$ then in the optimal solution, stage 1 stays critical from transfer batch 1 through a transfer batch $k-1 (1 \leq k \leq s)$. In other words, it is necessary to have no-wait on machine 1 between transfer batches 1 and transfer batch k in an optimal solution.

Suppose initially transfer batch sizes are updated such as to produce no idling on machine 2. This can be done without violating the nondecreasing transfer batch sizes, as shown in Theorem 4.11 and Corollary 4.4.

We will show that wait on machine 1, if any, can be removed starting from the earliest occurrence up to a transfer batch k beyond which nondecreasing transfer batch sizes can not be removed, without increasing the total flow time.

Let l $(2 \leq l \leq s-1)$ be the earliest transfer batch with wait on machine 1, then

$$p_1 \sum_{i=1}^{l} q_i + p_2 q_l < p_1 \sum_{i=1}^{l-1} q_i + p_2 q_{l-1} + p_2 q_l.$$

Let q_l and q_{l+1} be updated as $q'_l = q_l + \Delta$ and $q'_{l+1} = q_{l+1} - \Delta$, $\Delta > 0$. For no wait on machine 1, as shown before (strict equality is needed not to induce delay on machine 2)

$$\Delta = \frac{p_2}{p_1} q_{l-1} - q_l.$$

Now, let us express the change in total flow time:

$$\Delta F = C_l' q_l' + C_{l+1}' q_{l+1}' - C_l q_l - C_{l+1} q_{l+1}.$$

Then

$$\begin{aligned}\Delta F = &[p_1 \sum_{i=1}^{l} q_i + (p_2 + p_3) q_l + (p_1 + p_2 + p_3)\Delta](q_l + \Delta) \\ &+[p_1 \sum_{i=1}^{l-1} q_i + p_2 q_{l-1} + p_2(q_l + q_{l+1}) + p_3 q_{l+1} - p_3\Delta](q_{l+1} - \Delta) \\ &-[p_1 \sum_{i=1}^{l-1} q_i + p_2 q_{l-1} + (p_2 + p_3) q_l] q_l \\ &-[p_1 \sum_{i=1}^{l-1} q_i + p_2 q_{l-1} + p_2(q_l + q_{l+1}) q_l + p_3 q_{l+1}] q_{l+1}.\end{aligned}$$

Dividing ΔF by Δ, as

$$p_1 \sum_{i=1}^{l} q_i + (p_2 + p_3) q_l + (p_1 + p_2)\Delta = p_1 \sum_{i=1}^{l-1} q_i + p_2 q_{l-1} + (p_2 + p_3) q_l$$

$$\begin{aligned}\frac{\Delta F}{\Delta} &= p_3 q_l + p_1 \sum_{i=1}^{l} q_i + (p_2 + p_3) q_l + (p_1 + p_2 + p_3) q_l \Delta \\ &\quad - p_3 q_{l+1} - p_1 \sum_{i=1}^{l-1} q_i - p_2 q_{l-1} - p_2(q_l + q_{l+1}) - p_3 q_{l+1} + p_3\Delta \\ &= p_3(q_l - q_{l+1}) + p_3(q_l - q_{l+1} + 2\Delta) - p_2(q_{l+1} - \Delta) - p_2 q_{l-1} + p_1(q_l + \Delta)\end{aligned}$$

$$\Delta F \le 0$$

since $q_{l+1} \ge q_l + 2\Delta$ and $p_2 \ge p_1$. Therefore, the total flow time does not increase after the update. The proof is complete. ∎

Based on the developments above (Theorem 4.11, 4.12 and 4.13), the network of optimal transfer batch sizes are illustrated by Figure 4.13, 4.14, and 4.15, where heavy and dotted lines show critical and non-critical segments, respectively.

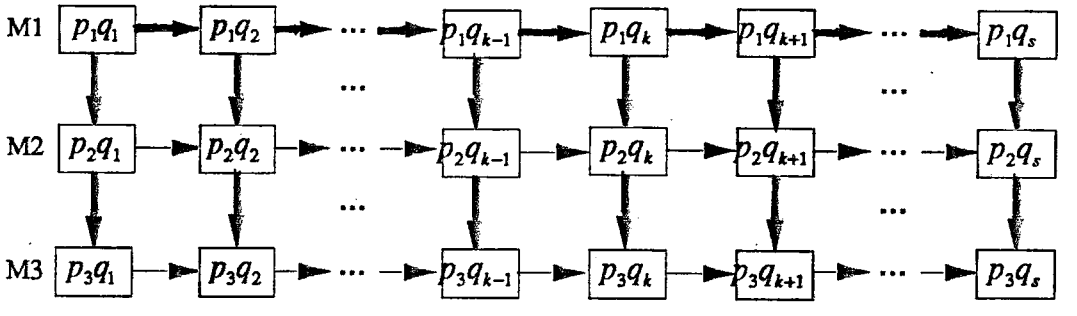


Figure 4.13. Network of optimal transfer batch sizes for $p_1 = \max_{1 \le m \le 3}\{p_m\}$.

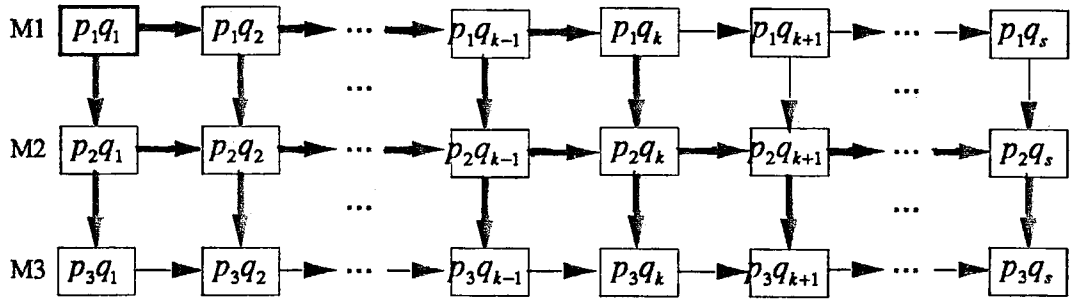
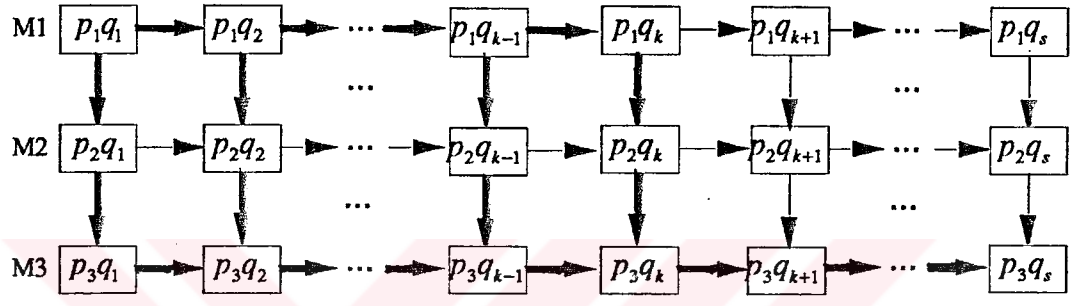
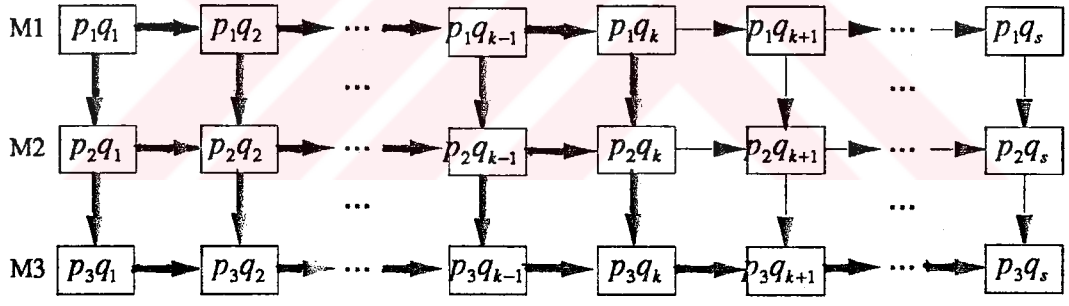


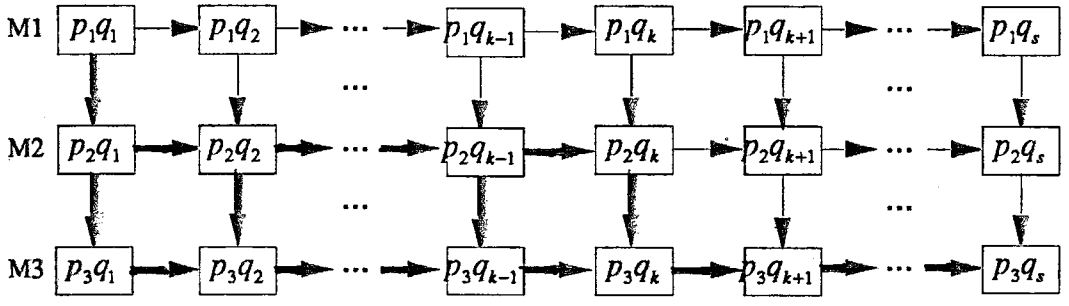
Figure 4.14. Network of optimal transfer batch sizes for $p_2 = \max_{1 \le m \le 3}\{p_m\}$.



(a) $(p_2)^2 < p_1p_3$



(b) $(p_2)^2 = p_1p_3$



(c) $(p_2)^2 > p_1p_3$

Figure 4.15. Network of optimal transfer batch sizes for $p_3 = \max_{1 \le m \le 3}\{p_m\}$.

By knowing the properties of the optimal solution to the three-machine total flow time problem, we can find the optimal transfer batch sizes in each case.

Theorem 4.14 *In the* $1|F3|TB(C,c)|\sum q_i C_i$ *problem, if* $p_2 = \max_{1\le m\le 3}\{p_m\}$ *or* $p_3 = \max_{1\le m\le 3}\{p_m\}$, *then in the optimal solution there exists a transfer batch* k *such that*

$$\begin{aligned} &(i) && q_i = \alpha q_{i-1}, && i = 2,\ldots,k \\ &(ii) && q_k \le q_{k+1} \le \alpha q_k \\ &(iii) && q_i = q_{i+1}, && i = k+1,\ldots,s-1 \end{aligned}$$

and the following transfer batch sizes are optimal

$$\begin{aligned} q_i &= \alpha^{i-1} q_1, && i = 1,\ldots,k; \\ q_i &= \frac{Q - (\frac{\alpha^k - 1}{\alpha - 1})q_1}{s-k}, && i = k+1,\ldots,s \end{aligned}$$

where

$$q_1 = \frac{(\frac{\alpha^k - 1}{\alpha - 1}) - \theta(s-k)}{(\frac{\alpha^k - 1}{\alpha - 1})^2 + (\frac{\alpha^{2k} - 1}{\alpha^2 - 1})(s-k)} Q$$

$$\alpha = \begin{cases} \dfrac{p_2}{p_1}, & \text{if } p_2 = \max\limits_{1\le j\le 3}\{p_j\} \\ \dfrac{p_3}{p_1}, & \text{if } p_3 = \max\limits_{1\le j\le 3}\{p_j\},\ p_2^2 \ge p_1 p_3, \\ \dfrac{p_2 + p_3}{p_1 + p_2}, & \text{if } p_3 = \max\limits_{1\le j\le 3}\{p_j\},\ p_2^2 < p_1 p_3 \end{cases}$$

$$\theta = \begin{cases} \dfrac{p_1}{p_2 + 2p_3}, & \text{if } p_2 = \max\limits_{1\le j\le 3}\{p_j\} \\ \dfrac{p_1 + p_2}{p_3}, & \text{if } p_3 = \max\limits_{1\le j\le 3}\{p_j\} \end{cases}$$

Proof: The results (i) and (ii) are consequences of Theorem 4.13 and Corollary 4.4, respectively. To prove (iii) we need to consider the following two cases:

Case 1: $p_2 = \max_{1\le m\le 3}\{p_m\}$.

From the network of optimal transfer batch sizes (see Fig. 4.14), using Theorem 4.12 the completion time of each transfer batch on the third machine is

$$C_{i,3} = p_1 q_1 + p_2 \sum_{j=1}^{i} q_i + p_3 q_i, \qquad i = 1,\ldots,s$$

and the total flow time is

$$F(q)=\sum_{i=1}^{s}C_{i,3}q_i=\sum_{i=1}^{s}(p_1q_1+p_2\sum_{j=1}^{i}q_i+p_3q_i)q_i$$

$$=p_1q_1\sum_{i=1}^{s}q_i+p_2\sum_{i=1}^{s}\sum_{j=1}^{i}q_jq_i+p_3\sum_{i=1}^{s}q_i^2$$

$$=p_1Qq_1+(\tfrac{p_2}{2})\sum_{i=1}^{s}q_i^2+(\tfrac{p_2}{2})(\sum_{i=1}^{s}q_i)^2+p_3\sum_{i=1}^{s}q_i^2$$

$$=p_1Qq_1+(\tfrac{p_2+2p_3}{2})\sum_{i=1}^{s}q_i^2+(\tfrac{p_2}{2})Q^2.$$

If there exists a transfer batch k satisfying property (i) and Corollary 4.4, then the total flow time can be rewritten as

$$F_k(q)=p_1Qq_1+(\tfrac{p_2+2p_3}{2})(\sum_{i=1}^{k}q_i^2+\sum_{i=k+1}^{s}q_i^2)+(\tfrac{p_2}{2})Q^2$$

$$=p_1Qq_1+(\tfrac{p_2+2p_3}{2})\sum_{i=1}^{k}(\alpha^{i-1}q_1)^2+(\tfrac{p_2+2p_3}{2})\sum_{i=k+1}^{s}q_i^2+(\tfrac{p_2}{2})Q^2.$$

$$=p_1Qq_1+(\tfrac{p_2+2p_3}{2})(\tfrac{\alpha^{2k}-1}{\alpha^2-1})q_1^2+(\tfrac{p_2+2p_3}{2})\sum_{i=k+1}^{s}q_i^2+(\tfrac{p_2}{2})Q^2. \quad (4.62)$$

Now, assume that q_1 is known for a given value of k. Then it is sufficient to minimize the third term in (4.62) which can be achieved if $q_i=q_{i+1}$, $i=k+1,\ldots,s-1$.

Using property (i), (ii) and $\sum_{i=1}^{s}q_i=Q$, we obtain

$$q_{k+1}=\frac{Q-(\frac{\alpha^k-1}{\alpha-1})q_1}{s-k}. \quad (4.63)$$

Substituting (4.64) into (4.62) yields

$$F_k(q_1)=p_1Qq_1+(\tfrac{p_2+2p_3}{2})(\tfrac{\alpha^{2k}-1}{\alpha^2-1})q_1^2+(\tfrac{p_2+2p_3}{2})\sum_{i=k+1}^{s}\{(Q-(\tfrac{\alpha^k-1}{\alpha-1})q_1)/(s-k)\}^2+(\tfrac{p_2}{2})Q^2$$

$$=(\tfrac{p_2+2p_3}{2(s-k)})\{(\tfrac{\alpha^k-1}{\alpha-1})^2+(\tfrac{\alpha^{2k}-1}{\alpha^2-1})(s-k)\}q_1^2-(\tfrac{p_2+2p_3}{s-k})\{(\tfrac{\alpha^k-1}{\alpha-1})-(\tfrac{p_1}{p_2+2p_3})(s-k)\}Qq_1$$

$$+\{(\tfrac{1}{s-k})(\tfrac{p_2+2p_3}{2})+\tfrac{p_2}{2}\}Q^2 \quad (4.64)$$

From property (i), it is clear that

$$q_k=\alpha^{k-1}q_1. \quad (4.65)$$

For a given value of a transfer batch k satisfying all properties above, the optimal size of the first transfer batch is obtained by setting $\partial F_k(q_1)/\partial q_1=0$ as

$$q_1=\frac{(\frac{\alpha^k-1}{\alpha-1})-(\frac{p_1}{p_2+2p_3})(s-k)}{(\frac{\alpha^k-1}{\alpha-1})^2+(\frac{\alpha^{2k}-1}{\alpha^2-1})(s-k)}Q \quad (4.66)$$

Case 2: $p_3=\max_{1\le m\le 3}\{p_m\}$

In this case, the completion time of each transfer batch on the third machine is written as

$$C_{i,3} = p_1 q_1 + p_2 q_1 + p_3 \sum_{j=1}^{i} q_i, \qquad i = 1,...,s,$$

and the total flow time is

$$\begin{aligned} F(q) &= \sum_{i=1}^{s} C_{i,3} q_i = \sum_{i=1}^{s} \{(p_1 + p_2) q_1 + p_3 \sum_{j=1}^{i} q_i\} q_i \\ &= (p_1 + p_2) q_1 \sum_{i=1}^{s} q_i + p_3 \sum_{i=1}^{s} \sum_{j=1}^{i} q_i^2 \\ &= (p_1 + p_2) Q q_1 + (\tfrac{p_3}{2}) \sum_{i=1}^{s} q_i^2 + (\tfrac{p_3}{2}) (\sum_{i=1}^{s} q_i)^2 \\ &= (p_1 + p_2) Q q_1 + (\tfrac{p_3}{2}) \sum_{i=1}^{s} q_i^2 + (\tfrac{p_3}{2}) Q^2. \end{aligned}$$

If there exists a transfer batch k satisfying (i) and Corollary 4.4, then total flow time is rewritten as

$$\begin{aligned} F_k(q) &= (p_1 + p_2) Q q_1 + (\tfrac{p_3}{2}) (\sum_{i=1}^{k} q_i^2 + \sum_{i=k+1}^{s} q_i^2) + (\tfrac{p_3}{2}) Q^2 \\ &= (p_1 + p_2) Q q_1 + (\tfrac{p_3}{2}) \sum_{i=1}^{k} (\alpha^{i-1} q_1)^2 + (\tfrac{p_3}{2}) \sum_{i=k+1}^{s} q_i^2 + (\tfrac{p_3}{2}) Q^2 \\ &= (p_1 + p_2) Q q_1 + (\tfrac{p_3}{2}) (\tfrac{\alpha^{2k} - 1}{\alpha^{2k} - 1}) q_1^2 + (\tfrac{p_3}{2}) \sum_{i=k+1}^{s} q_i^2 + (\tfrac{p_3}{2}) Q^2. \end{aligned} \tag{4.67}$$

We again assume that q_1 is known for a given value of k. Then it is sufficient to minimize the third term in (4.67) which can be achieved if $q_i = q_{i+1}$, $i = k+1,...,s-1$. The proof follows similar to Case 1. ∎

Theorem 4.15 *The transfer batch k satisfying properties of the optimal solution given in Theorem 4.14 can be found in $O(s)$.*

Proof: Without loss of generality, we give the proof for the case $p_2 = \max_{1 \le m \le 3} \{p_m\}$ only since the proof for the case $p_3 = \max_{1 \le m \le 3} \{p_m\}$ can be obtained similarly. For a given value of k, substituting (4.63) and (4.65) into property (ii) in Theorem 4.14 yields

$$q_L(k) \le q_1 \le q_U(k) \tag{4.68}$$

where $q_L(k) = Q / \{(\frac{\alpha^k - 1}{\alpha - 1}) + (s-k)\alpha^k\}$ and $q_U(k) = Q / \{(\frac{\alpha^k - 1}{\alpha - 1}) + (s-k)\alpha^{k-1}\}$.

It is clear that $q_L(k) = q_U(k+1)$. Then, using (4.64), the total flow time function then is written as

$$F(q) = F_k(q_1),$$

if

$$R_{k+1} \le q_1 \le R_k \tag{4.69}$$

where $R_k = Q / \{(\frac{\alpha^k - 1}{\alpha - 1}) + (s-k)\alpha^{k-1}\}$ and $F(q) = F_k(q_1)$.

To find the optimal value of q_1, we need to check whether or not a transfer batch k for $k = 1,2,...,s$ satisfies the condition in (4.69) and compare their corresponding total flow time functions. That is, $F(q_1^*) = \min_k \{F_k(q_1) | R_{k+1} \le q_1 \le R_k\}$ where q_1 as in expression (4.66). Hence, it is clear that transfer batch k satisfying properties of the optimal solution given in Theorem 4.14 can be found in $O(s)$. ∎

Remark 4.1 If $p_2 = \max_{1 \le m \le 3}\{p_m\}$ and $p_3 = 0$, then the problem reduces to a two-machine problem where $p_2 > p_1$.

We now present an algorithm to determine the transfer batch k which provides the optimal transfer batch sizes.

Algorithm 4.4

Step 1 Set $h = \arg\max_{1 \le j \le 3}\{p_j\}$.

Step 2 If $h = 1$, then the optimal transfer batch sizes are $q_i = Q/s$, $i = 1,...,s$; stop.

else (i) set

$$\alpha = \begin{cases} \frac{p_2}{p_1} & \text{if } h=2 \\ \frac{p_3}{p_1} & \text{if } h=3,\ p_2^2 \ge p_1 p_3, \\ \frac{p_2+p_3}{p_1+p_2} & \text{if } h=3,\ p_2^2 < p_1 p_3 \end{cases} \qquad \theta = \begin{cases} \frac{p_1}{p_2+2p_3} & \text{if } h=2 \\ \frac{p_1+p_2}{p_3} & \text{if } h=3 \end{cases}$$

$$k^* = 0,\ F_{k^*} = \infty,\ k = 1$$

(ii) set $R_k = Q / \{(\frac{\alpha^k - 1}{\alpha - 1}) + (s-k)\alpha^{k-1}\}$.

Step 3 Set $a = (\frac{\alpha^k - 1}{\alpha - 1})^2 + (\frac{\alpha^{2k} - 1}{\alpha^2 - 1})(s-k)$, $b = (\frac{\alpha^k - 1}{\alpha - 1}) - \theta(s-k)$, $q_1 = (\frac{b}{a})Q$,

$$R_{k+1} = Q / \{(\frac{\alpha^{k+1} - 1}{\alpha - 1}) + (s-k-1)\alpha^k\}.$$

Step 4 If $R_{k+1} \le q_1 \le R_k$, then

(i) if $k < s$, then set

$$F_k(q_1) = \begin{cases} \frac{p_2+2p_3}{2(s-k)}\, a\, (q_1)^2 - \frac{p_2+2p_3}{s-k}\, b\, q_1\, Q + (\frac{p_2+2p_3}{2(s-k)} + \frac{p_2}{2})\, Q^2 & \text{if } h=2 \\ \frac{p_3}{2(s-k)}\, a\, (q_1)^2 - \frac{p_3}{s-k}\, b\, q_1\, Q + (\frac{p_3}{2(s-k)} + \frac{p_3}{2})\, Q^2 & \text{if } h=3 \end{cases}$$

else set

$$F_k(q_1) = \begin{cases} (\frac{p_2+2p_3}{2})(\frac{\alpha^{2k} - 1}{\alpha^2 - 1})(q_1)^2 + p_1\, q_1\, Q + (\frac{p_2}{2})\, Q^2 & \text{if } h=2 \\ (\frac{p_3}{2})(\frac{\alpha^{2k} - 1}{\alpha^2 - 1})(q_1)^2 + (p_1 + p_2)\, q_1\, Q + (\frac{p_3}{2})\, Q^2 & \text{if } h=3 \end{cases}$$

(ii) if $F_k(q_1) < F_{k^*}$, then set $k^* = k$, $F_{k^*} = F_k(q_1)$.

Step 5 Set $k = k + 1$.

If $k > s$, then the optimal transfer batch sizes are

$$q_i^* = \alpha^{i-1} q_1^* \qquad i = 1,\ldots,k^*;$$

$$q_i^* = \frac{Q - (\frac{\alpha^{k^*}-1}{\alpha-1})q_1^*}{s-k^*} \qquad i = k^*+1,\ldots,s.$$

else go to step 3.

Example 4.5

The algorithm above is illustrated here in a five transfer batch example. Suppose that a lot consists of 100 identical items to be split into 5 transfer batches and processed in a three-machine flow shop. Let the unit processing times on machines be 2, 3 and 7, respectively.

Since $h = 3$ and $p_2^2 < p_1 p_3$, it is clear that $\theta = (p_1 + p_2)/p_3 = 5/7$ and $\alpha = (p_2 + p_3)/(p_1 + p_2) = (3+7)/(2+3) = 2$. Following the steps of the algorithm we obtain:

k	$q_{1,k}^*$	R_k	R_{k+1}	$R_{k+1} \le q_1 \le R_k$	$F_k(q_1^*)$	k^*	F_{k^*}
1	-37.143	11.111	20.000	No			
2	3.571	6.667	11.111	No			
3	6.122	4.348	6.667	Yes	46,530	3	46,530
4	4.608	3.226	4.348	No			
5	3.226	3.226	3.226	Yes	49,032		

Hence, we have the following optimal transfer batch sizes (see Figure 4.16):

$$q_1^* = 6.122,\ q_2^* = 12.245,\ q_3^* = 24.490,\ q_4^* = q_5^* = 28.571.$$

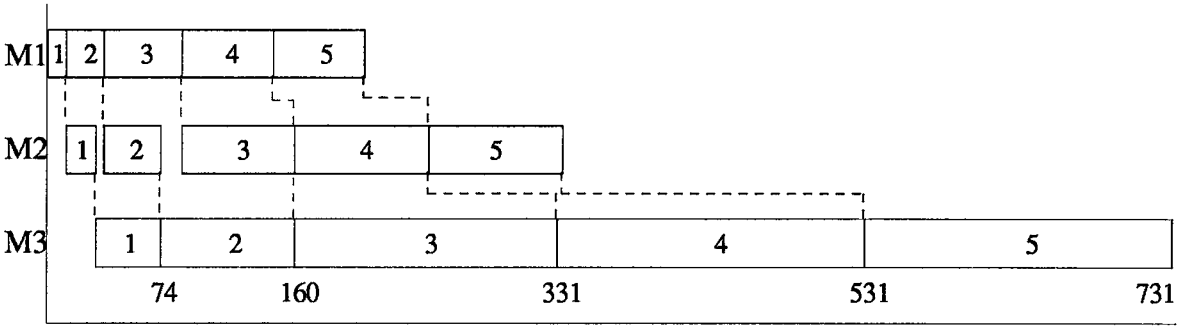


Figure 4.16. Optimal schedule of transfer batches in Example 4.5.

We have shown that the structure of the optimal network depends on the relative size of the job processing times on the three machines. We first proved that there is no idle time between processing of transfer batches on the machine with the longest processing time. When machine 1 has the longest processing time, it is critical for all transfer batches, and equal transfer batch sizes are optimal. When the second machine has the longest processing time, machine 1 and machine 2 are both critical from transfer batch 1 through transfer batch k (a pattern changing transfer batch), and the optimal transfer batch sizes follow a geometric progression up to some transfer batch k. Transfer batch sizes after that are equal to each other. If the machine with the longest processing time is the last one, either machines 1 and 3, or machines 2 and 3, or all three machines are critical from transfer batch 1 through some transfer batch k. A similar geometric progression of transfer batch sizes is also valid for batches 1 through some k.

In some cases, *no-wait* schedule may be required (i.e., a transfer batch can be processed on each machine immediately after its processing is finished on the preceding machine). It can be easily seen that from the structure of optimal networks, the no-wait requirement is satisfied for all transfer batches if $p_1 = \max_{1 \le m \le 3}\{p_m\}$. When $p_2 = \max_{1 \le m \le 3}\{p_m\}$ or $p_3 = \max_{1 \le m \le 3}\{p_m\}$, this requirement is only satisfied for transfer batch 1 up to k. However, on machine 1, shifting the start of each transfer batch from $k+1$ up to s to right (without creating any idle time on the machine with the longest processing time) will achieve the no-wait schedule. For instance, in Example 4.5, the no-wait requirement is satisfied for transfer batches 1 through 3 (see Figure 4.16). Shifting the start of transfer batches 4 and 5 to right on machine 1 without changing their start times on the third machine is possible as shown in Figure 4.17.

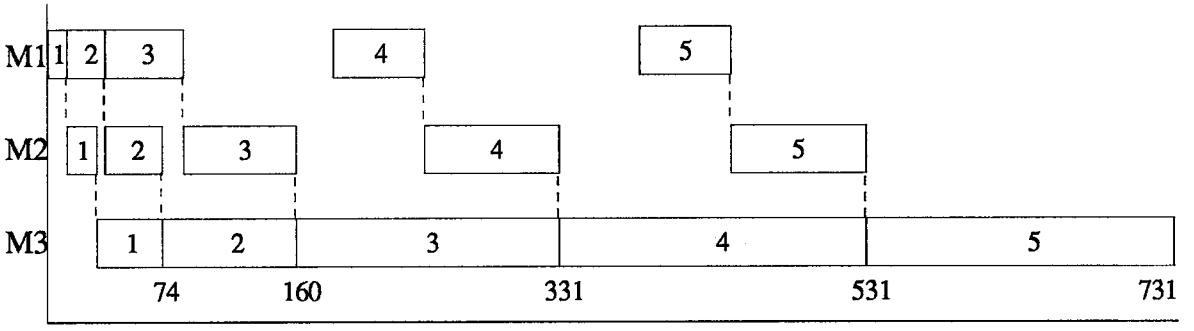


Figure 4.17. Optimal no-wait schedule of transfer batches in Example 4.5.

Table 4.2 below summarizes the solution status of single-job streaming problems with the mean flow time criterion which exist in the literature along with the problems investigated in this section.

TABLE 4.2. Summary of the solution status of single-job streaming problems with the mean flow time criterion

Problem	Complexity†	Reference
$1\|F2\|TB(C,c)\|\sum q_iC_i$	$O(s)$	Çetinkaya and Gupta (1994), Şen *et al.* (1994)
$1\|F2\|TB(C,c),NW\|\sum q_iC_i$	$O(s)$	Section 4.4 of this thesis
$1\|F2\|TB(C,c),S_d\|\sum q_iC_i$	$O(s)$	Section 4.3 of this thesis
$1\|F3\|TB(C,c)\|\sum q_iC_i$	$O(s)$	Section 4.4 of this thesis
$1\|F3\|TB(C,c),NW\|\sum q_iC_i$	$O(s)$	Section 4.4 of this thesis
$1\|F\|TB(C,c \text{ or } d)\|\sum q_iC_i$	*open*	
$1\|F\|TB(C,c \text{ or } d),p_1=\max\{p_m\}\|\sum q_iC_i$	$O(s)$	Section 4.1 of this thesis
$1\|F\|TB(s=2,C,c \text{ or } d)\|\sum q_iC_i$	$O(M^2)$	Section 4.2 of this thesis
$1\|F\|TB(V,c \text{ or } d)\|\sum q_iC_i$	*open*	

† s: number of transfer batches, M: number of machines, *open*: Complexity unknown

CHAPTER 5

MULTI-JOB STREAMING PROBLEMS

In the previous chapter, the single-job lot streaming problems are studied. On the other hand, a more realistic case is the problem with multiple jobs. At this level of generality the problem appears to be very difficult when the number of jobs is more than one. This is due to the fact that the transfer batch sizing and sequencing decisions must be made simultaneously. In this chapter, various multi-job lot streaming problems are studied, and exact solution procedures are given to minimize the makespan of the sequence for the corresponding problems.

5.1 Flow Shop Problems

5.1.1 Two-machine Problem with Detached Setup Times

In this section, we consider the two-machine flow shop problem in which the size of the transfer batches are not restricted to be unit or equal sized. If we define s_j as the number of transfer batches allowed, then, in our case, $2 \le s_j < Q_j$ for $1 \le j \le N$. Using our classification scheme, the problem is denoted by $N|F2|TB(strm,C,c)|C_{\max}$. The results presented below are reported in Çetinkaya (1994).

Now, suppose that our problem is decomposed into the sequencing problem of jobs given the transfer batch sizes and the transfer batch sizing problem given a sequence of jobs. Let us examine these two sub problems.

Sequencing Problem

Assume that we have already determine the transfer batch sizes for all jobs. The following theorem gives the optimal sequence of jobs given the transfer batch sizes.

Theorem 5.1 *Let $S_{j,m}$ and $R_{j,m}$ be the detached setup time and removal time for job j on machine m, $(m=1,2)$, respectively. Given the transfer batch sizes, in an*

optimal schedule for the problem $N|F2|TB(strm,C,c),S_d,R|C_{\max}$, *job* i *precedes job* j *if*

$$\min(RI_i, RO_j) \le \min(RI_j, RO_i)$$

where

$$RI_j = S_{j,1} - S_{j,2} + Z_j,$$
$$RO_j = R_{j,2} - R_{j,1} + B_j - A_j + Z_j,$$
$$Z_j = \max_{1\le u\le s_j}\{\sum_{i=1}^{u} p_{j,1}q_{j,i} - \sum_{i=1}^{u-1} p_{j,2}q_{j,i}\},$$
$$A_j = p_{j,1}Q_j,$$
$$B_j = p_{j,2}Q_j$$

for $j = 1,2,...,N.$

Proof: Let (j) be a job which is sequenced in the jth position for a schedule. Furthermore, let $C_{(j),m}$ and $C_{(j,i),m}$ denote the completion times of job (j) and its ith transfer batch on machine m, respectively. Then the completion time of job (j) on machine 1 is given by

$$C_{(j),1} = C_{(j-1),1} + S_{(j),1} + A_{(j)} + R_{(j),1}, \tag{5.1}$$

where $C_{(0),1} = 0.$

Similarly, we have

$$C_{(j,i),1} = C_{(j-1),1} + S_{(j),1} + \sum_{k=1}^{i} p_{(j),1}q_{(j,k)} \quad \text{for } i = 1,2,...,s_{(j)}. \tag{5.2}$$

Then, the completion time of the first transfer batch of (j) on machine 2 is expressed as

$$\begin{aligned} C_{(j,1),2} &= \max\{C_{(j-1),2} + S_{(j),2}, C_{(j,1),1}\} + p_{(j),2}q_{(j,1)} \\ &= \max\{C_{(j-1),2} + S_{(j),2}, C_{(j-1),1} + S_{(j),1} + p_{(j),1}q_{(j,1)}\} + p_{(j),2}q_{(j,1)} \end{aligned} \tag{5.3}$$

where $C_{(0),2} = 0.$

Similarly, we obtain

$$\begin{aligned} C_{(j,2),2} = \max\{&C_{(j-1),2} + S_{(j),2}, C_{(j-1),1} + S_{(j),1} + \max_{1\le u\le 2}\{\sum_{k=1}^{u} p_{(j),1}q_{(j,k)} - \sum_{k=1}^{u-1} p_{(j),2}q_{(j,k)}\} \\ &+ \sum_{i=1}^{2} p_{(j),2}q_{(j,i)}. \end{aligned} \tag{5.4}$$

Repeating this process yields

$$C_{(j,s_{(j)}),2} = \max\{C_{(j-1),2} + S_{(j),2}, C_{(j-1),1} + S_{(j),1} + \max_{1\le u\le s_{(j)}}\{\sum_{k=1}^{u} p_{(j),1}q_{(j,k)} - \sum_{k=1}^{u-1} p_{(j),2}q_{(j,k)}\}\}$$
$$+ B_{(j)} \qquad (5.5)$$

where $B_{(j)} = \sum_{i=1}^{s_{(j)}} p_{(j),2}q_{(j,i)} = p_{(j),2}Q_{(j)}$.

Thus, the completion time of job (j) on machine 2 is expressed as

$$\begin{aligned} C_{(j),2} &= \max\{C_{(j-1),2} + S_{(j),2}, C_{(j-1),1} + S_{(j),1} + \max_{1\le u\le s_{(j)}}\{\sum_{k=1}^{u} p_{(j),1}q_{(j,k)} - \sum_{k=1}^{u-1} p_{(j),2}q_{(j,k)}\}\} \\ &\quad + B_{(j)} + R_{(j),2} \\ &= \max\{C_{(j-1),2}, C_{(j-1),1} + S_{(j),1} - S_{(j),2} + \max_{1\le u\le s_{(j)}}\{\sum_{k=1}^{u} p_{(j),1}q_{(j,k)} - \sum_{k=1}^{u-1} p_{(j),2}q_{(j,k)}\}\} \\ &\quad + S_{(j),2} + B_{(j)} + R_{(j),2}. \end{aligned} \qquad (5.6)$$

By successive application of the expression above using $C_{(j-1),1}$ from (5.1), the makespan of the jobs is obtained as follows:

$$C_{\max} = C_{(N),2} = \max_{0\le u\le N}\{\sum_{j=1}^{u} RI_{(j)} - \sum_{j=1}^{u-1} RO_{(j)}\} + \sum_{j=1}^{N}(S_{(j),2} + B_{(j)} + R_{(j),2}) \qquad (5.7)$$

where

$$RI_{(j)} = S_{(j),1} - S_{(j),2} + Z_{(j)},$$
$$RO_{(j)} = R_{(j),2} - R_{(j),1} + B_{(j)} - A_{(j)} + Z_{(j)},$$
$$Z_{(j)} = \max_{1\le u\le s_{(j)}}\{\sum_{i=1}^{u} p_{(j),1}q_{(j,i)} - \sum_{i=1}^{u-1} p_{(j),2}q_{(j,i)}\}.$$

Since the second part in (5.7) is a constant which is independent of the sequence, an equivalent problem is to minimize the first part which indicates the total idle, I, on machine 2. Hence, the objective function to be minimized becomes

$$I = \max_{0\le u\le N}\{\sum_{j=1}^{u} RI_{(j)} - \sum_{j=1}^{u-1} RO_{(j)}\}.$$

The form of I is the same as in Johnson's expression (see Johnson, 1954). Hence, an optimal sequence of the jobs can be obtained by job i preceding job j if $\min(RI_i, RO_j) \le \min(RI_j, RO_i)$. ∎

Transfer Batch-sizing Problem

Now, we assume that we have an arbitrary sequence of jobs. The following theorem gives the optimal transfer batch sizes for any job in the sequence.

Theorem 5.2 *In any arbitrary sequence of jobs in the problem* $N|F2|TB(strm,C,c),S_d,R|C_{\max}$, *the optimal transfer batch sizes of a job in position* (j) *should be*

$$q_{(j,i)} = (\alpha^{i-1} / \sum_{k=1}^{s_{(j)}} \alpha^{k-1}) Q_{(j)} = \alpha^{i-1} (\frac{\alpha - 1}{\alpha^{s_{(j)}} - 1}) Q_{(j)} \quad i = 1,2,...,s_{(j)};\ j = 1,2,...,N \qquad (5.8)$$

where $\alpha = p_{(j),2} / p_{(j),1}$.

Proof: From (5.7) it is clear that the minimization of the makespan, $C_{(N),2}$, through transfer batch-sizing for a given sequence, calls for the minimization of the idle time between processing transfer batches on machine 2, $Z_{(j)}$ for each job separately. That is, transfer batch sizes for any sequence (hence the optimal sequence) that minimize the makespan are identical to those transfer batch sizes that minimize individual job makespans independently. Hence, solving transfer batch-sizing problem once and for all would not worsen the solution no matter what the sequence is. Therefore, the following linear program is solved for every job (j):

$$\text{Min} \quad Z_{(j)} = \max_{1 \le u \le s_{(j)}} \{\sum_{i=1}^{u} p_{(j),1} q_{(j,i)} - \sum_{i=1}^{u-1} p_{(j),2} q_{(j,i)}\} \qquad (5.9)$$

$$\text{s.to} \quad \sum_{i=1}^{s_{(j)}} q_{(j,i)} = Q_{(j)} \qquad (5.10)$$

$$q_{(j,i)} \ge 0\ , \qquad i = 1,2,...,s_{(j)}.$$

The model above can be rewritten as

$$\text{Min} \quad Z_{(j)}$$

$$\text{s.to} \quad Z_{(j)} \ge p_{(j),1} q_{(j,1)}$$

$$Z_{(j)} \ge p_{(j),1} q_{(j,1)} + p_{(j),1} q_{(j,2)} - p_{(j),2} q_{(j,1)}$$

$$\vdots$$

$$Z_{(j)} \ge p_{(j),1} q_{(j,1)} + ... + p_{(j),1} q_{(j,s_{(j)})} - p_{(j),2} q_{(j,1)} - ... - p_{(j),2} q_{(j,s_{(j)}-1)}$$

$$\sum_{i=1}^{s_{(j)}} q_{(j,i)} = Q_{(j)}$$

$$q_{(j,i)} \ge 0\ , \qquad i = 1,2,...,s_{(j)}.$$

When we examine the structure of the model, it can be seen that the solution is trivial. That is, the minimum value of $Z_{(j)}$ is achieved when it is equal to $p_{(j),1} q_{(j,1)}$. Accordingly, in the optimal solution, the inequality constraints must be satisfied as equalities. This reveals that the adjacent transfer batches must be in a unique proportion as

$$q_{(j,i)} = \alpha q_{(j,i-1)} = \alpha^{i-1} q_{(j,1)} \quad i = 1,2,\ldots,s_{(j)} \tag{5.11}$$

where $\alpha = p_{(j),2} / p_{(j),1}$.

Substituting (5.17) into (5.16) yields

$$q_{(j,1)} = Q_{(j)} / \sum_{k=1}^{s_{(j)}} \alpha^{k-1}. \tag{5.12}$$

Using (5.11) and (5.12), we obtain the optimal transfer batch sizes of a job in any position in any arbitrary sequence as

$$q_{(j,i)} = (\alpha^{i-1} / \sum_{k=1}^{s_{(j)}} \alpha^{k-1}) Q_{(j)} = \alpha^{i-1} (\frac{\alpha - 1}{\alpha^{s_{(j)}} - 1}) Q_{(j)} \qquad i = 1,2,\ldots,s_{(j)}$$

which is the same as Baker's expression given to find the optimal transfer batch sizes of a single job. ∎

Theorems 5.1 and 5.2 lead to the following corollary.

Corollary 5.1 *Partitioning each job independently into optimal transfer batches and then sequencing the jobs optimally yields an optimal solution to the* $N|F2|TB(strm,C,c),S_d,R|C_{\max}$ *problem.*

The current results, in fact, can also be interpreted in the framework of the time-lag model described by Mitten (1959). In the time-lag model, the basic two-machine flow shop problem which is proposed by Johnson (1954) is considered by allowing arbitrary time lags. A start-lag for a job is defined as the required delay between the start of its first and second operations. Analogously, a stop-lag is the required delay between the completion of its first and second operations. In fact, when we consider the lot streaming problem of job j by itself, from Theorem 5.2, it is clear that there is no idle time between processing of transfer batches on the second machine. If a semi-active timetable in which every activity is started as early as possible is taken into account, the idle time on machine 2 may occur between the setup and processing activities of a job. However, delaying the setup on machine 2 so as to eliminate the idle time between setup and processing yields a continuous processing of all activities on the second machine for a job. Note that this shifting does not affect the completion time of the job on the second machine. Then, RI_j can be treated as a start-lag which gives the largest delay between the start of setups on machines 1 and 2. Similarly, RO_j can be thought of as a stop-lag for the completion of the removal operations on machines 1 and 2. Therefore, Mitten's algorithm for the time-lag model can also be applied to our case in which transfer batches are allowed by using time lags RI_j and RO_j for every job j.

A method of finding integer transfer batch sizes

In principle, a job would consist of a number of identical items and, thus it may be reasonable to require integer transfer batch sizes. An immediate consequence of this requirement is that the above linear model becomes an mixed integer programming problem. However, without solving an mixed integer programming model, it is possible to find the optimal integer transfer batch sizes by modifying the algorithm proposed by Trietsch (1989).

From our linear programming model given in Theorem 5.2, we have

$$Z_{(j)} \geq \sum_{k=1}^{i} p_{(j),1} q_{(j,k)} - \sum_{k=1}^{i-1} p_{(j),2} q_{(j,k)} = p_{(j),1} q_{(j,i)} + (p_{(j),1} - p_{(j),2}) \sum_{k=1}^{i-1} q_{(j,k)} \tag{5.13}$$

for $i = 1,2,\ldots,s_{(j)}$.

Since we want to obtain integer transfer batch sizes, each $q_{(j,i)}$ must be the largest integer which satisfies the inequality (5.13). That is,

$$q_{(j,i)} \leq \left\lfloor [Z_{(j)} - (p_{(j),1} - p_{(j),2}) \sum_{k=1}^{i-1} q_{(j,k)}] / p_{(j),1} \right\rfloor \tag{5.14}$$

where $\lfloor x \rfloor$ denotes the largest integer less than or equal to x.

Since we have a constraint (5.11) on the sum of transfer batch sizes, the following recursive formula is given to determine the size of transfer batch i

$$q_{(j,i)} \leq \min\left\{ \left\lfloor [Z_{(j)} - (p_{(j),1} - p_{(j),2}) \sum_{k=1}^{i-1} q_{(j,k)}] / p_{(j),1} \right\rfloor, Q_{(j)} - \sum_{k=1}^{i-1} q_{(j,k)} \right\} \tag{5.15}$$

If $\sum_{i=1}^{s_{(j)}} q_{(j,i)} < Q_{(j)}$, then there must exist at least one transfer batch which should include at least one more item. Hence, the current value of $Z_{(j)}$ must increase. The smallest increment can be given as

$$\delta = p_{(j),1} \min_i (1 - f_{(j,i)}) \tag{5.16}$$

where $f_{(j,i)}$ is the fraction of the next item that machine 1 will have being processed when the ith transfer batch is sent to the second machine, and is given as

$$f_{(j,i)} = [Z_{(j)} - (p_{(j),1} - p_{(j),2}) \sum_{k=1}^{i-1} q_{(j,k)}] / p_{(j),1} - q_{(j,i)} \tag{5.17}$$

in which $q_{(j,i)}$ is obtained from (5.15).

In the light of developments above, an algorithm for determining the optimal integer-valued transfer batch sizes for a job (j) works as follows. Select the total idle time of the continuous version as the initial value of $Z_{(j)}$. Then, compute the size of the first transfer batch from (5.15). Repeat this calculation for all transfer batches. If the cumulative number of items in the transfer batches is equal to the lot size of the job, then the current transfer batch sizes are optimal. However, when this condition is not satisfied, the current value of $Z_{(j)}$ is incremented by (5.16). Repeat these steps until the optimality condition is satisfied.

Hence, using the results above, we may propose the following algorithm to solve the problem optimally.

Algorithm 5.1

Step 1 For all elements of job set, $J = \{1,2,...,N\}$,

(i) determine the size of each transfer batch as

$$q_{j,i} = k_{j,i} Q_j \qquad i = 1,2,...,s_j$$

$$\text{where } k_{j,i} = \begin{cases} 1/s_j & \text{if sublots are selected as equal sized} \\ \alpha^{i-1}\left(\dfrac{\alpha-1}{\alpha^{s_j}-1}\right) & \text{otherwise} \end{cases},$$

and $\alpha = p_{j,2} / p_{j,1}$.

(ii) If $q_{j,i}$ are all integers, then go to Step 2; otherwise go to Step 1(iii).

(iii) If transfer batches are selected as equal sized

(a) determine the size of each transfer batch as $q_{j,i} = Q_j / h_j$ for $i = 1,2,...,h_j$ where h_j is the largest integer less than or equal to s_j such that Q_j / h_j becomes an integer;

(b) let $Z_j = \max\{p_{j,1}q_{j,1} + p_{j,2}Q_j,\ p_{j,1}Q_j + p_{j,2}q_{j,s_j}\} - p_{j,2}Q_j$.

Otherwise,

(a) let $Z_j = \max\limits_{1 \le u \le s_j}\{\sum\limits_{i=1}^{u} p_{j,1}q_{j,i} - \sum\limits_{i=1}^{u-1} p_{j,2}q_{j,i}\}$;

(b) for each transfer batch, $i = 1,2,...,s_j$, compute

$$q_{j,i} \le \min\left\{\left\lfloor [Z_j - (p_{j,1} - p_{j,2})\sum_{k=1}^{i-1} q_{j,k}] / p_{j,1} \right\rfloor, Q_j - \sum_{k=1}^{i-1} q_{j,k}\right\},$$

$$f_{j,i} = [Z_j - (p_{j,1} - p_{j,2})\sum_{k=1}^{i-1} q_{j,k}] / p_{j,1} - q_{j,i}$$

(c) if $\sum_{i=1}^{s_j} q_{j,i} = Q_j$, then the current transfer batch sizes, $q_{j,i}$, are integers, go to Step 2;
otherwise, set $Z_j = Z_j + p_{j,1} \min\limits_i (1 - f_{j,i})$, go to Step (b).

Step 2 (i) For all jobs, compute

$$A_j = p_{j,1}Q_j,$$
$$B_j = p_{j,2}Q_j,$$
$$RI_j = S_{j,1} - S_{j,2} + Z_j,$$
$$RO_j = R_{j,2} - R_{j,1} + B_j - A_j + Z_j.$$

(ii) Decompose the job set J into two parts as:

$$J_A = \{j : RI_j \le RO_j\} \text{ and } J_B = \{j : RI_j > RO_j\}.$$

Step 3 (i) Sort the job numbers of set J_A in non-decreasing order of RI_j and denote the resulting list by J_A^*.

(ii) Sort the job numbers of set J_B in non-increasing order of RO_j and denote the resulting list by J_B^*.

Step 4 The optimal sequence of job set is $J^* = \{J_A^*, J_B^*\}$ and its makespan is

$$C_{\max} = \max_{0 \le u \le N} \{\sum_{j=1}^{u} RI_{(j)} - \sum_{j=1}^{u-1} RO_{(j)}\} + \sum_{j=1}^{N} (S_{(j),2} + B_{(j)} + R_{(j),2}).$$

Example 5.1

To illustrate Algorithm 5.1, consider a five-job scheduling problem in which transfer batches for all jobs are not restricted to be equal sized. The relevant set of data is shown in Table 5.1.

TABLE 5.1. Production data for five-job problem

Job	Lot size	Number of sublots	Unit processing time		Setup time		Removal time	
			Machine 1	Machine 2	Machine 1	Machine 2	Machine 1	Machine 2
j	Q_j	s_j	$p_{j,1}$	$p_{j,2}$	$S_{j,1}$	$S_{j,2}$	$R_{j,1}$	$R_{j,2}$
1	30	3	1	1	5	10	5	5
2	10	2	3	2	10	25	5	15
3	18	2	2	1	15	10	10	10
4	25	3	1	2	15	5	10	5
5	14	2	3	4	20	15	15	5

Step 1(i) of the algorithm provides integer-valued transfer batch sizes for all jobs except job 4. Initially, transfer batch sizes for job 4 are 3.57, 7.14 and 14.29 units. However, by applying Step 1(iii), integer transfer batch sizes are obtained as 4, 8 and 13 units which are shown in Table 5.2. Then, the job set, J, is decomposed into: $J_A = \{1, 2, 4\}$ and $J_B = \{3, 5\}$ with their sorted lists $J_A^* = \{2, 1, 4\}$ and $J_B^* = \{5, 3\}$. Hence, the optimal schedule illustrated in Figure 5.1 is obtained as $J^* = \{2, 1, 4, 5, 3\}$ having a makespan of 282 time units with 3 units of total idle on machine 2.

TABLE 5.2. Transfer batch sizes

j	i	$q_{j,i}$	$S_{j,1} - S_{j,2}$	$R_{j,2} - R_{j,1}$	$B_j - A_j$	Z_j	RI_j	RO_j
1	1	10	- 5	0	0	10	5	10
	2	10						
	3	10						
2	1	6	- 15	10	- 10	18	3	18
	2	4						
3	1	12	5	0	- 18	24	29	6
	2	6						
4	1	4	10	- 5	25	4	14	24
	2	8						
	3	13						
5	1	6	5	- 10	14	18	23	22
	2	8						

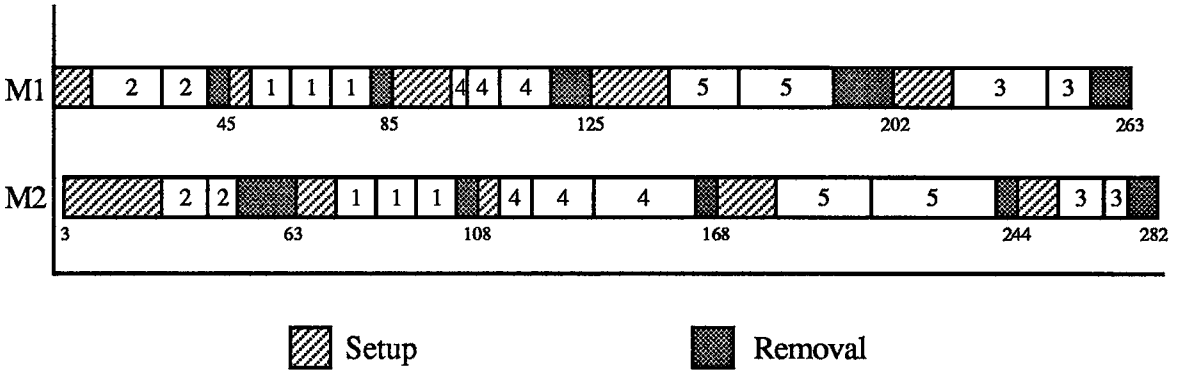


Figure 5.1. Optimal schedule in Example 5.1.

5.1.2 Two-machine Problem with No Wait-in-process

In this section, we consider two-machine flow shop with a constraint that once the processing of a transfer batch begins on the first machine, its subsequent processing must be carried out with no delay in the passage of the job from machine to machine. If necessary, the start of transfer batch processing is delayed on the first machine so that the transfer batches need not wait for processing on the second machine. Figure 5.2 illustrates the no-wait restriction for a three-job example. Using the classification scheme given in Chapter 1, the problem that will be considered is denoted by $N|F2|TB(strm,C,c),NW|C_{\max}$.

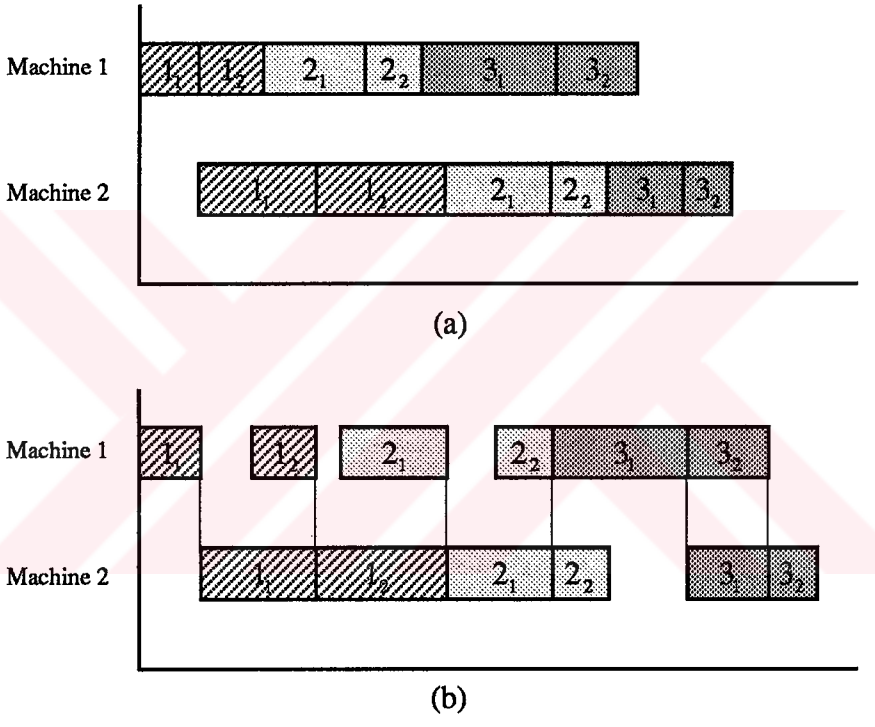


Figure 5.2. Schedule of jobs (a) with wait-in-process, (b) with no-wait-in-process

Now, suppose that we, in advance, know the transfer batch sizes for all jobs. Then, in the following theorem, we show that the current problem given the transfer batch sizes is reduced to the Gilmore-Gomory Traveling Salesman Problem and solved in $O(N \log N)$.

Theorem 5.3 *The* $N|F2|TB(strm,C,c),NW|C_{\max}$ *problem can be solved in* $O(N \log N)$ *if the transfer batch sizes of all jobs are known in advance.*

Proof: Let (j) be a job which is sequenced in the jth position for a no-wait schedule. Then the total idle time on the first machine, I, can be expressed as the sum of total idle time between jobs and total idle time between transfer batches of jobs. That is,

$$I = \sum_{j=1}^{N-1} \max\{p_{(j),2}\, q_{(j,s_{(j)})} - p_{(j+1),1}\, q_{(j+1,1)}\ ,\ 0\} + \sum_{j=1}^{N} \sum_{k=1}^{s_{(j)}-1} \max\{p_{(j),2} q_{(j,k)} - p_{(j),1} q_{(j,k+1)}\ ,\ 0\}. \tag{5.18}$$

Then the makespan of this schedule, which is the sum of the total idle time on machine 1 and the processing time of the last transfer batch of the last job on machine 2, is given by

$$C_{\max} = I + \sum_{j=1}^{N} p_{(N),2}\, Q_{(j)} + p_{(N),2}\, q_{(N,s_{(N)})}$$

$$= \sum_{j=1}^{N-1} \max\{p_{(j),2}\, q_{(j,s_{(j)})} - p_{(j+1),1}\, q_{(j+1,1)}\ ,0\} + \sum_{j=1}^{N} \sum_{k=1}^{s_{(j)}-1} \max\{p_{(j),2} q_{(j,k)} - p_{(j),1} q_{(j,k+1)}\ ,0\} + \sum_{j=1}^{N} p_{(j),1}\, Q_{(j)} + p_{(N),2}\, q_{(N,s_{(N)})}. \tag{5.19}$$

When the transfer batch sizes for all jobs are known in advance, the second and third terms in (5.19) can be ignored since they are independent of the sequence and constant. Thus, it is sufficient to minimize the sum of the first and last terms in (5.19).

In turn, the problem of minimizing the sum of the first and last terms can be reduced to the *Traveling Salesman Problem* (*TSP*). In order to transform the schedule to a *Hamiltonian* tour, a dummy job "0" with $p_{(0),1} = p_{(0),2} = 0$, i.e. with zero processing time on both machines, is introduced. The dummy job marks the beginning and the end of a schedule.

Let us define the distance matrix $D = (d_{i,j})$ of the *TSP* as

$$d_{i,j} = \max\{b_i - a_j, 0\},\quad 0 \le i \le N,\quad 0 \le j \le N,\quad i \ne j \tag{5.20}$$

where $a_j = p_{j,1} q_{j,1}$ and $b_j = p_{j,2}\, q_{j,s_j}$ for $0 \le j \le N$.

Note that $d_{0,j} = \max\{b_0 - a_j, 0\} = 0$ since $b_0 = 0$, and $d_{j,0} = \max\{b_j - a_0, 0\} = b_j$ since $a_0 = 0$. Then, the length of the tour, denoted by L_{TSP}, can be expressed as follows:

$$L_{TSP} = d_{(0),(1)} + \sum_{j=1}^{N-1} d_{(j-1),(j)} + d_{(N),(0)} = \sum_{j=1}^{N-1} d_{(j-1),(j)} + d_{(N),(0)}. \qquad (5.21)$$

The above matrix $D = (d_{i,j})$ is known as *large matrix* in *TSP* literature. As shown by Gilmore and Gomory (1964) and Gilmore *et al.* (1986), The *TSP* restricted to (pseudo) large matrices is known to be solvable in $O(N \log N)$, and this completes the proof. ∎

Remark 5.1 The above result is also true if there are detached setup times and removal times in the problem (denoted by $N|F2|TB(strm,C,c),NW,S_d,R|C_{\max}$). The only change will be in the definition of the total idle time between pair of jobs and accordingly in the matrix D. Thus the expression (5.18) is rewritten as

$$I = \sum_{j=1}^{N-1} \max\{[p_{(j),2} q_{(j,s_{(j)})} + R_{(j),2} - R_{(j),1}] - [S_{(j+1),1} + p_{(j+1),1} q_{(j,1)} - S_{(j+1),2}]\ ,\ 0\}$$
$$+ \sum_{j=1}^{N} \sum_{k=1}^{s_{(j)}-1} \max\{p_{(j),2} q_{(j,k)} - p_{(j),1} q_{(j,k+1)}\ ,\ 0\} \qquad (5.22)$$

which leads to the change in the definition of $D = (d_{i,j})$ with $a_j = S_{j,1} + p_{j,1} q_{j,1} - S_{j,2}$ and $b_j = p_{j,2}\, q_{j,s_j} + R_{j,2} - R_{j,1}$ for $0 \le j \le N$.

Now assume that we don't know transfer batch sizes in advance. Then it is clear that transfer batch-sizing and job sequencing problems must be solved simultaneously. However, when we consider each job independently, the optimal transfer batch sizes which satisfy the no-wait constraint and also minimize the makespan of every individual job are obtained by Baker's formula that we mentioned in Chapter 2. That is, for a job j, the optimal transfer batch sizes have the following relation:

$$q_{j,i} = \alpha\, q_{j,i-1}, \qquad i = 2,\ldots,s_j \qquad (5.23)$$

where $q_{j,1} = (\dfrac{\alpha - 1}{\alpha^{s_j} - 1}) Q_{(j)}$, $\alpha = p_{j,2} / p_{j,1}$.

If we take into account the relation above in equation (5.19), the second term becomes zero since

$$p_{(j),2}\, q_{(j,k)} - p_{(j),1}\, q_{(j,k+1)} = 0 \qquad (5.24)$$

for $1 \le j \le N$ and $1 \le k \le s_{(j)} - 1$. Thus, $C_{\max} = L_{TSP} + \sum_{j=1}^{N} p_{j,1}\, Q_j$.

However, we are not sure that the solution of the *TSP* problem with the transfer batch sizes that are computed from (5.23) for all jobs (i.e., splitting each job independently into optimal transfer batches and then sequencing the jobs optimally by the Gilmore and Gomory algorithm) is optimal to the problem in which transfer batch sizes and the sequence of jobs are found simultaneously. On the other hand, in each of the limited number of problems solved by this approach, we obtained the optimal solution. At this point, we could not able to prove this result and leave it as a conjecture. However, the approach may also be used as an heuristic to solve the problem. Therefore, this issue may require further analysis.

Conjecture 5.1 *The* $N|F2|TB(strm,C,c),NW,S_d,R|C_{\max}$ *problem is solved by Algorithm 5.2 optimally.*

Remark 5.2 Note that the conjecture gives the optimal sequence of jobs where transfer batches are assumed to be equal sized.

Algorithm 5.2

Step 1 Apply the first step of Algorithm 5.1.

Step 2 For all jobs, compute $a_j = S_{j,1} + p_{j,1}q_{j,1} - S_{j,2}$ and $b_j = p_{j,2}\, q_{j,s_j} + R_{j,2} - R_{j,1}$.

Step 3 Calculate elements of distance matrix $D = (d_{i,j})$ for the *TSP* as

$$d_{i,j} = \begin{cases} \max\{b_i - a_j, 0\} & \text{if} \quad i \neq j \\ \infty & \text{if} \quad i = j \end{cases}.$$

Step 4 Apply Gilmore-Gomory algorithm to solve the *TSP* problem. Determine the corresponding sequence of jobs.

Example 5.2

To illustrate Algorithm 5.2, let us reconsider Example 5.1. By applying the first step of the algorithm, we obtain the integer sized transfer batches as shown in Table 5.2. In the second and third steps, we determine

j	a_j	b_j
1	5	10
2	3	18
3	29	6
4	14	21
5	23	22

$$D = \begin{array}{c|cccccc} i\backslash j & 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & \infty & 0 & 0 & 0 & 0 & 0 \\ 1 & 10 & \infty & 7 & 0 & 0 & 0 \\ 2 & 18 & 13 & \infty & 0 & 4 & 0 \\ 3 & 6 & 1 & 3 & \infty & 0 & 0 \\ 4 & 21 & 16 & 18 & 0 & \infty & 0 \\ 5 & 22 & 17 & 19 & 0 & 8 & \infty \end{array}$$

The solution of the *TSP* gives us the optimal sequence of jobs as $J^* = \{2,4,5,3,1\}$ with a makespan of 291 time units of which 18 and 12 units of time are the total idle times on the first and second machines, respectively.

5.2 Open Shop Problems

In this section, we will deal with the multi-job two-machine open shop lot streaming problem with detached setup times. Our objective is to minimize the makespan of the schedule. Remember that in open shops each job has to be processed on each machine, however, the process route is not given in advance for each job but must be chosen, different jobs being allowed to follow different routes. It is well known that the multi-job two-machine classical open shop problem without any setup times in which the objective to be minimized is the makespan is optimally solvable in linear time by the Algorithm G&S proposed by Gonzales and Sahni (1976). We give here a simplified version of this algorithm.

Algorithm G&S

Step 1 Decompose a given set of jobs $J = \{1,2,...,N\}$ into two subsets as

$$J_A = \{j \mid P_{j,1} \le P_{j,2}\} \text{ and } J_B = \{j \mid P_{j,1} > P_{j,2}\}$$

where $P_{j,m}$ is the total processing time of Q_j identical items of job j on machine m.

Step 2 Choose a job u such that $\max\{P_{u,1}, P_{u,2}\} = \max\{\max_{j \in J_A}\{P_{j,1}\}, \max_{j \in J_B}\{P_{j,2}\}\}$.

Step 3 If $u \in J_A$, then

(i) Let $J'_A = J_A - \{u\}$.

(ii) In each subset J'_A and J_B, sequence the jobs in any arbitrary order.

(iii) Construct the optimal schedule by the sequence $\{J'_A, J_B, u\}$ on machine 1 and $\{u, J'_A, J_B\}$ on machine 2 in which job u is processed first on machine 2 and then on machine 1, and the processing order of other jobs is machine 1 and machine 2.

If $u \in J_B$, then

(i) Let $J'_B = J_B - \{u\}$
(ii) In each subset J'_B and J_A, sequence the jobs in any arbitrary order.
(iii) Construct the optimal schedule by the sequence $\{u, J'_B, J_A\}$ on machine 2 and $\{J'_B, J_A, u\}$ on machine 1 in which job u is processed first on machine 1 and then on machine 2, and the processing order of other jobs is machine 2 and machine 1.

The optimal makespan is $C_{\max} = \max\{\sum_{j=1}^{N} P_{j,1}, \sum_{j=1}^{N} P_{j,2}, \max_{1 \le j \le N}\{P_{j,1} + P_{j,2}\}\}$.

It is easy to see that finding the index u in Step 2 is possible in linear time and hence the whole algorithm runs in $O(N)$ time.

In the rest of our discussion, without loss of generality, we will assume that $u \in J_A$ since similar results can be obtained with the interchanged role of machines 1 and 2, and of the subsets J_A and J_B if $u \in J_B$.

Now, we investigate some characteristics of the schedule obtained by the Algorithm G&S.

Lemma 5.1 *One of the following three paths in the optimal schedule obtained by Algorithm G&S must be critical.*

$$\begin{aligned}
&\textit{path (i)} : \quad P_{j,1}\ (\forall j \in J'_A) \rightarrow P_{j,1}\ (\forall j \in J_B) \rightarrow P_{u,1} \\
&\textit{path (ii)} : \quad P_{u,2} \rightarrow P_{j,2}\ (\forall j \in J'_A) \rightarrow P_{j,2}\ (\forall j \in J_B) \\
&\textit{path (iii)} : \quad P_{u,2} \rightarrow P_{u,1}
\end{aligned}$$

where $u \in J_A = \{j \mid P_{j,1} \le P_{j,2}\}$, $J'_A = J_A - \{u\}$ *and* $J_B = \{j \mid P_{j,1} > P_{j,2}\}$.

Proof: Let (i) be the job in the ith position of the sequence on machine 1 of the optimal schedule obtained by Algorithm G&S in which job u (defined before) is in the last position. Then we have only $N-1$ forms of paths as candidate for being critical except the three paths defined in the statement of the lemmas. They have the following form:

$$P_{(1),1} \rightarrow \cdots \rightarrow P_{(i),1} \rightarrow P_{(i),2} \rightarrow \cdots \rightarrow P_{(N-1),1} \quad , \quad i = 1,2,\ldots,N-1.$$

We will show that none of these $N-1$ forms of paths can have a length greater than the length of paths defined by (i), (ii), and (iii).

If job $(i) \in J_A$, then the path has the length

$$\sum_{j=1}^{i} P_{(j),1} + \sum_{j=i}^{N-1} P_{(j),2} \leq \sum_{j=1}^{i-1} P_{(j),2} + P_{(i),2} + \sum_{j=i}^{N-1} P_{(j),2}$$

$$\leq \sum_{j=1}^{i-1} P_{(j),2} \tag{5.25}$$

which the length of path (ii).

If job $(i) \in J_B$, then the path has the length

$$\sum_{j=1}^{i} P_{(j),1} + \sum_{j=i}^{N-1} P_{(j),2} \leq \sum_{j=1}^{i-1} P_{(j),2} + P_{(i),1} + \sum_{j=i}^{N-1} P_{(j),2}$$

$$\leq \sum_{j=1}^{i-1} P_{(j),1} \tag{5.26}$$

which the length of path (iii). ∎

Lemma 5.1 gives a useful insight into the structure of Algorithm G&S which we will use in the following theorems to derive the optimal schedule of the problem with setup times. On the other hand, Lemma 5.1 gives a very simple proof of the correctness of Algorithm G&S.

Let us define the length of all operations of a job j by

$$L(j) = S_{j,1} + S_{j,2} + P_{j,1} + P_{j,2} \quad \text{for} \quad 1 \leq j \leq N.$$

Furthermore, we define the length of all operations on machine 1 and 2 by

$$L(M1) = \sum_{j=1}^{N} (S_{j,1} + P_{j,1})$$

and

$$L(M2) = \sum_{j=1}^{N} (S_{j,2} + P_{j,2}),$$

respectively. We first give the following theorem which shows the case on which there is no need for lot streaming in the optimal schedule of the problem.

Theorem 5.4 *If* $\max\{L(M1), L(M2)\} \geq \max_{1 \leq j \leq N}\{L(j)\}$, *then an optimal schedule for the multi-job two-machine open shop lot streaming problem with setup times* $N|O2|TB(strm,C,c),S_d|C_{\max}$ *can be found by Algorithm G&S without streaming any job.*

Proof: According to Lemma 5.1, there are three paths in the solution from which at least one is critical. These paths have the following length:

path (i): $$\sum_{j=1}^{N}(S_{j,1}+P_{j,1})=L(M1) \tag{5.27}$$

path (ii): $$\sum_{j=1}^{N}(S_{j,2}+P_{j,2})=L(M2) \tag{5.28}$$

path (iii): $$S_{k,1}+S_{k,2}+P_{k,1}+P_{k,2}=L(k) \text{ for some } k=1,2,...,N. \tag{5.29}$$

By the assumption of the theorem, i.e., $\max\{L(M1)\ ,\ L(M2)\} \geq \max_{1\leq j\leq N}\{L(j)\}$, one of the two paths (i) and (ii) must be critical with corresponding length of $L(M1)$ and $L(M2)$, respectively. Because both $L(M1)$ and $L(M2)$ are lower bounds for the value of the makespan of the problem, an optimal schedule can be found by Algorithm G&S in which the optimal makespan is $C_{\max}=\{L(M1)\ ,\ L(M2)\}$. Thus, this solution can not be improved by any means (including streaming one or more jobs). ∎

Corollary 5.2 *In the problem* $N|O2|TB(strm,C,c),S_d|C_{\max}$, *lot streaming is beneficial if* $\max_{1\leq j\leq N}\{L(j)\} > \max\{L(M1)\ ,\ L(M2)\}$.

From now on, in our discussion, we restrict our attention to the case $\max_{1\leq j\leq N}\{L(j)\} > \max\{L(M1)\ ,\ L(M2)\}$. The following theorem further simplifies our problem.

Theorem 5.5 *In the optimal schedule for the problem* $N|O2|TB(strm,C,c),S_d|C_{\max}$, *there can be at most one job* k *such that*

$$L(k) > \max\{L(M1)\ ,\ L(M2)\}.$$

Proof: The expression (5.27) is rewritten as

$$L(M1)=S_{k,1}+P_{k,1}+\sum_{\substack{j=1\\ j\neq k}}^{N}(S_{j,1}+P_{j,1}).$$

Since $L(k) > \max\{L(M1)\ ,\ L(M2)\}$, we have

$$L(M1) < L(k). \tag{5.30}$$

Substituting $L(k)=S_{k,1}+S_{k,2}+P_{k,1}+P_{k,2}$ into (5.30) yields

$$\sum_{\substack{j=1\\ j\neq k}}^{N}(S_{j,1}+P_{j,1}) < S_{k,2}+P_{k,2}. \tag{5.31}$$

Similarly, the expression (5.28) is rewritten as

$$L(M2) = S_{k,2} + P_{k,2} + \sum_{\substack{j=1 \\ j \neq k}}^{N} (S_{j,2} + P_{j,2}).$$

Since $L(k) > \max\{L(M1)\ ,\ L(M2)\}$, we have

$$L(M2) < L(k). \tag{5.32}$$

Substituting $L(k) = S_{k,1} + S_{k,2} + P_{k,1} + P_{k,2}$ into (5.32) yields

$$\sum_{\substack{j=1 \\ j \neq k}}^{N} (S_{j,2} + P_{j,2}) < S_{k,1} + P_{k,1} \tag{5.33}$$

Adding both sides of (5.31) and (5.33), we obtain

$$\sum_{\substack{j=1 \\ j \neq k}}^{N} (S_{j,1} + S_{j,2} + P_{j,1} + P_{j,2}) < S_{k,1} + S_{k,2} + P_{k,1} + P_{k,2} = L(k) \tag{5.340}$$

which implies that there can be no job k' with $L(k') > \max\{L(M1)\ ,\ L(M2)\}$. ∎

Corollary 5.3 *In the problem* $N|O2|TB(strm,C,c),S_d|C_{\max}$, *only job* k *is streamed if* $L(k) > \max\{L(M1)\ ,\ L(M2)\}$.

Theorem 5.6 *An optimal schedule to the problem* $N|O2|TB(strm,C,c),S_d|C_{\max}$ *is obtained by the sequence* $\{k, J'\}$ *on machine 1 and* $\{J', k\}$ *on machine 2 if job* k *is processed first on machine 1 and then on machine 2 where* $J' = J - \{k\}$, $L(k) > \max\{L(M1)\ ,\ L(M2)\}$, *and the jobs in set* J' *are arbitrarily sequenced.*

Proof: Assume that we have only job k, and this job is optimally streamed by taking into account its setup type, detached or attached, to minimize its completion time. Furthermore, let RI_k be the run-in delay of job k which is the latest delay time of a setup for job k on machine 2 without increasing the completion time of the job, C_k. Also, let RO_k be the run-out delay of job k which is the difference between its processing completion times on machine 1 and 2 (See Figure 5.3).

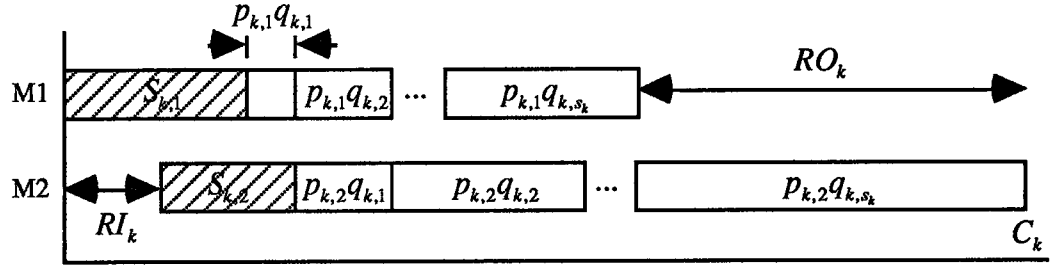


Figure 5.3. Optimal schedule for job k with detached setups.

Now, consider the schedule defined by the statement of the theorem. Then there are four cases to be considered:

Case 1: $RI_k \geq \sum_{j \in J'} (S_{j,2} + P_{j,2})$ and $RO_k \geq \sum_{j \in J'} (S_{j,1} + P_{j,1})$.

In this case, all machine 2 operations (setup and processing) of the jobs in set J' can be performed within the time interval [0, RI_k] without delaying the start of setup for job k, and all operations of these jobs on machine 1 can be done within the time interval [$C_k - RO_k$, C_k]. Thus, the corresponding makespan of the schedule is equal to the completion time of job k on machine 2, i.e., $C_{\max} = C_k$.

Case 2: $RI_k \geq \sum_{j \in J'} (S_{j,2} + P_{j,2})$ and $p_{k,2}\, q_{k,s_k} < \sum_{j \in J'} (S_{j,1} + P_{j,1})$.

In this case, as in the previous case, all machine 2 operations (setup and processing) of the jobs in set J' can be performed within the time interval [0, RI_k] without delaying the start of setup operation for job k. However, all operations of these jobs on machine 1 can not be done within the time interval [$C_k - RO_k$, C_k] since their operations length is greater than RO_k, i.e., their operations on machine 1 is completed after the completion of the job k. Thus, the corresponding makespan is equal to the total time of the operations of all jobs on machine 1, i.e., $C_{\max} = L(M1) = \sum_{j \in J} (S_{j,1} + P_{j,1})$.

Case 3: $RI_k < \sum_{j \in J'} (S_{j,2} + P_{j,2})$ and $RO_k \geq \sum_{j \in J'} (S_{j,1} + P_{j,1})$.

In this case, all machine 2 operations of the jobs in set J' can not be performed within the time interval [0, RI_k], and thus cause a delay on the start of setup operation for job k as well as the completion time of job k. However, their operations on machine 1 is completed before the completion of the job k. Thus, the corresponding makespan is equal to the total time of the operations of all jobs on machine 2, i.e., $C_{\max} = L(M2) = \sum_{j \in J} (S_{j,2} + P_{j,2})$.

Case 4: $RI_k < \sum_{j \in J'} (S_{j,2} + P_{j,2})$ and $RO_k < \sum_{j \in J'} (S_{j,1} + P_{j,1})$.

In this case, all machine 2 operations of the jobs in set J' can not be performed within the time interval [0, RI_k], and thus cause a $\{\sum_{j \in J'} (S_{j,2} + P_{j,2})\} - RI_k$ time units delay on the start of setup operation for job k as well as the completion time of job k. On the other hand, their operations on machine 1 may or may not be completed before

the completion of the job k. If $RO_k - RI_k + \sum_{j\in J'}(S_{j,2}+P_{j,2}) > \sum_{j\in J'}(S_{j,1}+P_{j,1})$, then the corresponding makespan is $C_{\max} = L(M2)$; otherwise, $C_{\max} = L(M1)$.

From the above cases, it is clear that the makespan is obtained as

$$C_{\max} = \max\{C_k,\ L(M1),\ L(M2)\}.$$

Since C_k is minimized by streaming job k, $C_{\max}$ is the minimum makespan for our problem which implies that the schedule defined by the statement of the theorem is optimal. ∎

Using a similar approach followed above, we may give the following theorem to the case where job k satisfying the property in Corollary 5.3 is processed first on machine 2 and then on machine 1.

Theorem 5.7 *An optimal schedule to the problem* $N|O2|TB(strm,C,c),S_d|C_{\max}$ *is obtained by the sequence* $\{J', k\}$ *on machine 1 and* $\{k, J'\}$ *on machine 2 if job* k *is processed first on machine 2 and then on machine 1 where* $J' = J - \{k\}$, $L(k) > \max\{L(M1),\ L(M2)\}$, *and the jobs in set* J' *are arbitrarily sequenced.*

Note that the minimum makespan values for the schedules defined in Theorems 5.6 and 5.7 may not be the same if job k is first processed on machine 1 and then on machine 2, or processed first on machine 2 and then on machine 1. Therefore, it is obvious that if $C_{\max} \le C'_{\max}$ then an optimal schedule for the multi-job two-machine open shop lot streaming problem with setups is the one given in Theorem 5.6; otherwise, it is the one given in Theorem 5.7 where $C_{\max}$ and $C'_{\max}$ are the minimum makespans for the schedules obtained by Theorems 5.6 and 5.7, respectively.

Example 5.3

The optimal schedule is illustrated here in a three-job example with the following data given in Table 5.3. We will assume that all setups are detachable.

TABLE 5.3 Production data for three-job example

Job	Lot size	Number of sublots	Unit processing time		Setup time	
			Machine 1	Machine 2	Machine 1	Machine 2
1	10	2	1	1	5	5
2	70	3	2	1	10	40
3	15	2	1	1	5	10

In this example, $\max_{1\le j\le 3}\{L(j)\} = L(2) = 260 > \max\{L(M1), L(M2)\} = \max\{185, 160\}$, thus job 2 is streamed. Let us assume that the routing of job 2 is from machine 2 to machine 1. Then the optimal transfer batch sizes are $q_{2,1} = 1$, $q_{2,2} = 2$, and $q_{2,3} = 4$. From Theorem 5.6, we can have a job sequence {1, 3, 2} (as well as {3, 1, 2}) on machine 1 and {2, 1, 3} on machine 2, with a makespan $C_{\max} = C_2 = 190$ units of time. On the other hand, if the routing of job 2 is from machine 1 to machine 2. Then the optimal transfer batch sizes are $q_{2,1} = 4$, $q_{2,2} = 2$, and $q_{2,3} = 1$. From Theorem 5.7, we have a job sequence {2, 1, 3} on machine 1 and {1, 3, 2} on machine 2, with a makespan $C'_{\max} = L(M1) = 185$ units of time. These schedules are illustrated in Figure 5.4. Since $C'_{\max} < C_{\max}$, the optimal schedule is the one given by Theorem 5.7, i.e., $C^*_{\max} = C'_{\max} = 185$.

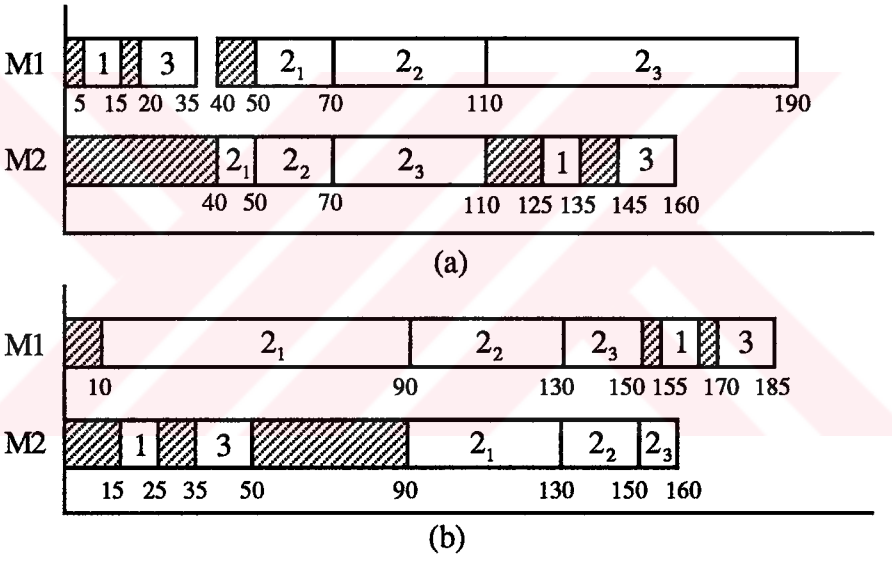


Figure 5.4. Route for job 2 is (a) from machine 2 to machine 1, (b) from machine 1 to machine 2

Table 5.4 below summarizes the solution status of multi-job streaming problems with the makespan criterion which exist in the literature along with the problems investigated in this section.

TABLE 5.4. Summary of the solution status of multi-job streaming problems with the makespan criterion

Problem	Complexity†	Reference
$N\|F2\|TB(strm,U)\|C_{\max}$	$O(N\log N)$	Vickson and Alfredson (1992)
$N\|F2\|TB(strm,U),S_d$ or $S_a\|C_{\max}$	$O(N\log N)$	Çetinkaya and Kayalıgil (1992)
$N\|F2\|TB(strm,C,c),S_d,R\|C_{\max}$	$O(N\log N)$	Section 5.1.1 of this thesis
$N\|F2\|TB(strm,C,c),NW,S_d,R\|C_{\max}$	$O(N\log N)$	Section 5.1.2 of this thesis
$N\|O2\|TB(strm,C,c)\|C_{\max}$	$O(N)$	Şen *et al.* (1995)
$N\|O2\|TB(strm,C,c),S_d\|C_{\max}$	$O(N)$	Section 5.1.2 of this thesis

† N: number of jobs

CHAPTER 6

MULTI-JOB SPLITTING PROBLEMS

In the previous chapter, we study the lot streaming problems with multiple jobs in flow shops and open shops. In these problems, we assume that the objective function to be minimized is the makespan and the transfer batches of a job can not be intermingled with the transfer batches of other jobs. As it is demonstrated by Potts and Baker (1989), makespan of the optimal schedule in multi-job lot streaming problems can be reduced by allowing transfer batches to be sequenced independently, even for simple problems involving only two machines and two jobs, especially when the setups are negligible or zero. Moreover, even with the equal sized transfer batches, it may be optimal to allow lot splitting when the number of machines is more than two (Vickson and Alfredson, 1992).

In the literature, to our knowledge, there is only one study which addresses the lot splitting problem in a two and three machine flow shop (Tan, 1996). This results from the complicated interleaving structure of the problem. Actually, the problem is to determine simultaneously transfer batch sizes and their sequences on the machines. However, despite the difficulty of an exact analysis, the development of heuristics that allow job splitting seems worthwhile. For this reason, in this chapter, we analyze the two-machine flow shop lot splitting problem to minimize the makespan for two different production environments.

6.1 Two-machine Flow Shop Problem

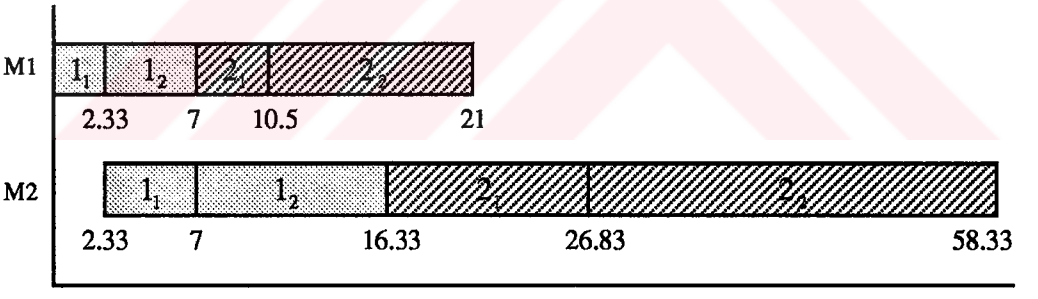
In this section, we consider the two-machine flow shop makespan minimization problem in which jobs that are split into transfer batches and the transfer batches of jobs can be intermingled in the schedule. According to the classification scheme described in Chapter 1, this problem is denoted by $N|F2|TB(splt,V,c)|C_{\max}$. Before presenting our results, it might be interesting to give the example of Potts and Baker (1989).

Example 6.1

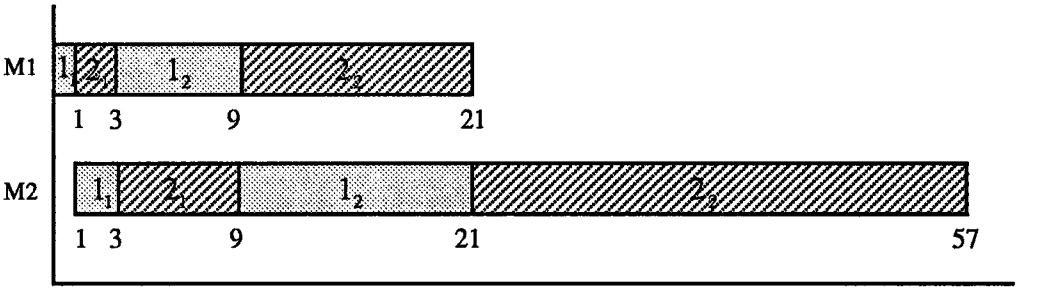
Consider a two-job problem in which each job is to be split into two transfer batches (See Table 6.1.). If we do not allow job splitting, but allow lot streaming, the optimal transfer batches sizes and the job sequence with a makespan of 58.33 are as shown in Table 6.1 and Figure 6.1(a), respectively. However, the size and the sequence of the transfer batches must be determined simultaneously when we allow lot splitting. In this case, makespan is reduced to 57 (which is optimal) as shown in Figure 6.1(b), and the optimal transfer batch sizes are as shown in Table 6.1. Although the reduction in makespan is 2.28 percent, the difference may be significant as the number of jobs gets large.

TABLE 6.1. Two-machine two-sublot problem

Job	Lot size	Unit processing time		Optimal transfer batch sizes: With lot-streaming		With lot-splitting	
		Machine 1	Machine 2	Sublot 1	Sublot 2	Sublot 1	Sublot 2
1	7	1	2	2.33	4.67	1	6
2	14	1	3	3.50	10.50	2	12



(a)



(b)

Figure 6.1. Optimal schedules with (a) lot-streaming, (b) lot-splitting.

Now, assume that we use the transfer batches obtained by allowing lot streaming for the first case, and then find their optimal sequence by applying Johnson's Rule. As shown in Figure 6.2, the makespan of this schedule is again 58.33 although the sequence of the transfer batches in this schedule is equivalent to the one given in the optimal schedule. This implies that this hierarchical solution scheme may not give better solutions than the solution of the problem with lot streaming. However, it may be good starting point to develop a heuristic or an optimal solution procedure as we will discuss later on.

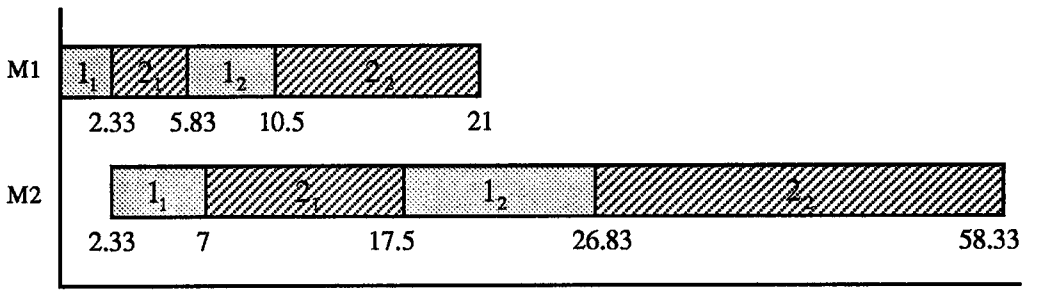


Figure 6.2. Lot-splitting schedule obtained by streaming each job independently.

6.1.1 Properties of the Optimal Solution

Assume that we have a set of jobs $J = \{1,...,N\}$ in which each job $j \in J$ having Q_j identical items will be split into s_j transfer batches. The following theorem gives a basic property of the optimal solution for the two-machine flow shop lot splitting problem.

Theorem 6.1 (Tan, 1996) *In the optimal solution to the multi-job lot splitting problem in a two-machine flowshop* $N|F2|TB(splt,V,c)|C_{max}$, *the optimal sequence of transfer batches are of permutation type, and the transfer batch sizes are consistent.*

Let job set J be decomposed into two distinct sets J_A and J_B such that $J_A = \{j: p_{j,1} \le p_{j,2}\}$ and $J_B = \{j: p_{j,1} > p_{j,2}\}$ where $p_{j,m}$ is the processing time per item of job j on machine m, $m = 1,2$. Then, we state another property of the optimal solution in the following theorem.

Theorem 6.2 *In the optimal solution to the* $N|F2|TB(splt,V,c)|C_{max}$ *problem, the transfer batches of jobs in set* $J_A = \{j: p_{j,1} \le p_{j,2}\}$ *are not to be intermingled with the*

transfer batches of jobs in set $J_B = \{j: p_{j,1} > p_{j,2}\}$. *That is, the processing of jobs in set* J_A *is completed before the processing of jobs in set* J_B.

Proof: From Theorem 6.1, we know that all transfer batches are consistent in the optimal solution. Now, assume that we know the transfer batch sizes for all jobs in advance. Then, the problem can be considered as an ordinary two-machine flow shop problem with K jobs, where $K = \sum_{j \in J} s_j$ is the total number of transfer batches in all jobs. That is, regard the problem as that of scheduling K transfer batches using Johnson's Algorithm. If a job is an element of set $J_A = \{j: p_{j,1} \le p_{j,2}\}$, then its transfer batches remain also in set J_A. Similarly, this condition is also true for the transfer batches of jobs in set J_B. In optimal sequence of the ordinary two-machine flow shop problem, the jobs in set J_A precedes the jobs in set J_B, thus the transfer batches of jobs in set J_A are not be intermingled with the transfer batches of jobs in set J_B. ∎

Corollary 6.1 *Let* q_i *and* $p_{i,m}$ *be the size and corresponding unit processing time on machine* m *of the transfer batch in the* i*th position of an optimal sequence of the* $N|F2|TB(splt,V,c)|C_{\max}$ *problem. Then, the transfer batches of jobs in set* J_A *are processed in non-decreasing order of* $p_{i,1}\, q_i$ *values, i.e.,*

$$p_{1,1}q_1 \le p_{2,1}q_2 \le \ldots \le p_{i,1}q_i \le \ldots \le p_{K_A,1}q_{K_A},$$

and the transfer batches of jobs in set J_B *are processed in non-increasing order of* $p_{i,2}\, q_i$ *values, i.e.,*

$$p_{K_A+1,2}q_1 \ge p_{K_A+2,2}q_2 \ge \ldots \ge p_{K,2}q_K,$$

where K_A *and* K_B *are the total number of transfer batches in sub sets* J_A *and* J_B, $K = K_A + K_B$.

Now, we present the following result which simplifies the solution and the size of the problem.

Corollary 6.2 *The* $N|F2|TB(splt,V,c)|C_{\max}$ *problem decomposes into two sub problems as*

sub problem P_A*: finding simultaneously size and sequence of transfer batches in set* J_A,

sub problem P_B*: finding simultaneously size and sequence of transfer batches in set* J_B.

Proof: From Theorem 6.2, we know that the processing order of sets is J_A and J_B. Then, without loss of generality, we may assume that there are two artificial jobs, namely A and B, which correspond to job set J_A and J_B, respectively. Using the notion of run-in and run-out delays for a job, also assume that we have a right-shifted schedules for the artificial jobs A and B. Then the makespan of the artificial problem is given by

$$\begin{aligned} C_{\max} &= \max\{RI_A, RI_A + RI_B - RO_A\} + \sum_{j\in J} p_{j,2}Q_j \\ &= \max\{RO_A, RI_B\} + RI_A - RO_A + \sum_{j\in J} p_{j,2}Q_j \\ &= \max\{RO_A, RI_B\} + RI_A - [RI_A + \sum_{j\in J_A}(p_{j,2} - p_{j,1})Q_j] + \sum_{j\in J} p_{j,2}Q_j \\ &= \max\{RO_A, RI_B\} + \sum_{j\in J_A} p_{j,1}Q_j + \sum_{j\in J_B} p_{j,2}Q_j \end{aligned}$$

since the artificial job A is processed first in the optimal sequence.

Since the second and third terms in the above equation are constant which is independent of the sequence, we need only to minimize the first term. This goal is achieved if RO_A and RI_B are both minimized at the same time. In other words, the corresponding makespans $C^A_{\max}$ and $C^B_{\max}$ of the jobs in set J_A and J_B must be minimized independently since $RO_A = C^A_{\max} - \sum_{j\in J_A} p_{j,2}Q_j$ and $RI_B = C^B_{\max} - \sum_{j\in J_B} p_{j,2}Q_j$. Therefore, the original problem can be decomposed into two sub problems as stated above. ∎

Now, let us only consider the sub problem P_A and define its *inverse* P_B. If the processing time for each job $j \in J_A$ on machine m is $p_{j,3-m}$ for $m = 1,2$, then $p_{j,1} \geq p_{j,2}$ for each job j in the new sub problem which is equivalent to the sub problem P_B. This fact implies that it is sufficient to develop optimal solution to the sub problem P_A since the sub problem P_B can be solved using the equivalency result of a sub problem and its inverse. Thus, from now on, we only restrict our attention to the sub problem P_A.

6.1.2 A Two-level Iterative Procedure

In this section, we decompose the sub problem P_A into two sub problems called *transfer batch-sizing* and *sequencing*, respectively, and propose a two-level iterative procedure to obtain its solution. Let,

N_A = number of jobs in job set J_A

J_j = job j in job set $J_A = \bigcup_{j=1}^{N_A} J_j$

Q_j = number of identical items in job J_j

s_j = number of consistent transfer batches allowed for job J_j

K_A = total number of transfer batches allowed, i.e., $K_A = \sum_{j \in J_A} s_j$

σ = sequence of K_A transfer batches

i = ith transfer batch in sequence σ

q_i = size of the ith transfer batch in sequence σ

$p_{j,m}$ = unit processing time for job J_j on machine m, $m = 1,2$

$\overline{q}$ = vector of transfer batch sizes such that $\overline{q} = \{q_i : i = 1,...,K_A\}$

In the *transfer batch-sizing* problem, given a sequence of transfer batches σ, the optimal vector of transfer batch sizes $\overline{q}(\sigma)$ are determined. On the other hand, in the *sequencing* problem, given a vector of transfer batch sizes $\overline{q}$, the optimal sequence of transfer batches $\sigma(\overline{q})$ are determined.

Transfer Batch-sizing Subproblem

For a given sequence of K_A transfer batches, σ, the total idle time on machine 2 is given as

$$I = \max_{1 \le i \le K_A} \{\sum_{k=1}^{i} p_{k,1} q_k - \sum_{k=1}^{i-1} p_{k,2} q_k\} = \max_{1 \le i \le K_A} \{\sum_{k=1}^{i-1} (p_{k,1} - p_{k,2}) q_k + p_{i,1} q_i\}.$$

Then, the optimal vector of transfer batch sizes $\overline{q}(\sigma)$ can be determined by solving the following linear programming model $BS(\sigma)$ which minimizes the total idle time on machine 2.

$BS(\sigma)$: Min I

s.to $I \ge \sum_{k=1}^{i-1} (p_{k,1} - p_{k,2}) q_k + p_{i,1} q_i$, $i = 1,...,K_A$

$\sum_{i \in J_j} q_i = Q_j$, $j = 1,...,N_A$

$I \ge 0$, $q_i \ge 0$, $i = 1,...,K_A$

The first set of constraints ensure that the total idle time of machine 2 before processing the ith transfer batch in sequence σ is at least as large as the difference between the sum of processing times of the first i transfer batches on machine 1 and the sum of processing times of the first $i-1$ transfer batches on machine 2. For each job, the second set of constraints ensure that the sum of transfer batch sizes is equal to the lot size of the job. Note that in an optimal solution, even if there are s_j number of

transfer batches allowed for job J_j, some of them may be empty, which means that less than s_j transfer batches were necessary. Also, note that this formulation has $K_A + N_A$ constraints and $K_A + 1$ variables which is easily solved with any Linear Programming package.

Sequencing Subproblem

By Corollary 6.1, for a given vector of transfer batch sizes $\overline{q}$, an optimal sequence of transfer batches $\sigma(\overline{q})$ can be determined in $O(K_A \log K_A)$ time by sorting transfer batches in nondecreasing order of their processing times on machine 1. Let us call this problem as $S(\overline{q})$.

Based on the transfer batch-sizing and sequencing subproblems, we propose an iterative procedure to obtain a solution of problem P_A. But, we first introduce the following notation:

k = iteration number

$\overline{q}^{(k)}(\sigma^{(k)})$ = for a given sequence $\sigma^{(k)}$, the vector of optimal transfer batch sizes obtained by solving the problem $BS(\sigma^{(k)})$ in iteration k

$\sigma^{(k)}(\overline{q}^{(k)})$ = for a given vector of transfer batch sizes $\overline{q}^{(k)}$, the optimal sequence of transfer batches obtained by solving the problem $S(\overline{q}^{(k)})$ in iteration k

$I(\sigma^{(k)})$ = minimum total idle time which corresponds to the sequence $\sigma^{(k)}$

L = list of alternative sequences of equal total idle time

Now, we are ready to present the two-level iterative procedure as follows:

Algorithm 6.1

Step 0 (Initialization) (i) Set $L = \varnothing$, $\sigma^{(0)} = \varnothing$ and $I(\sigma^{(0)}) = +\infty$.

(ii) Set $k = 1$. An initial sequence $\sigma^{(1)}$ is fixed.

Step 1 (Batch-sizing) (i) Solve problem $BS(\sigma^{(k)})$ to obtain $\overline{q}^{(k)}(\sigma^{(k)})$ and $I(\sigma^{(k)})$.

(ii) If $I(\sigma^{(k)}) < I(\sigma^{(k-1)})$, then $L = \sigma^{(k)}$.

(iii) If $I(\sigma^{(k)}) = I(\sigma^{(k-1)})$, then $L = L \cup \sigma^{(k)}$.

Step 2 (Sequencing) (i) For a given vector of transfer batch sizes $\overline{q}^{(k)}(\sigma^{(k)})$, find a sequence σ by solving the problem $S(\overline{q}^{(k)}(\sigma^{(k)}))$.

(ii) If $\sigma \notin L$, then goto Step 2(iii), otherwise, stop.

(iii) If $I(\sigma) > I(\sigma^{(k)})$, then stop; otherwise, set $k = k+1$, $\sigma^{(k)} = \sigma$, $I(\sigma^{(k)}) = I(\sigma)$ and goto Step 1.

Note that L is used to avoid cycles in successive iterations of the algorithm, and can have at most two sequences before the algorithm stops. In Step 2, our goal is to find a better sequence which is not in the list L. If such a sequence does not exist, we terminate with the solution left in the list L.

Theorem 6.4 *Algorithm 6.1 terminates in a finite number of iterations with a solution* $(\sigma^*, \overline{q}^*, I^*)$.

Proof: Let σ_A be the set of all possible $K_A!$ sequences of transfer batches. An optimal total idle time $I(\sigma)$ on machine 2 is associated to each sequence $\sigma \in \sigma_A$ by solving the problem $BS(\sigma)$. Since, by construction of the algorithm, a sequence can not be picked more than once, I value at each iteration is monotone decreasing, and the set σ_A is finite, the procedure stops after a finite number of iterations with a sequence of transfer batches σ^*, a vector of transfer batch sizes $\overline{q}^*$, and an associated total idle time I^*. ∎

As it is shown in the initialization step of the algorithm, we need an initial sequence of transfer batches. We may start with any arbitrary sequence such as the one obtained with equal transfer batches for all jobs. However, to obtain a better initial sequence, suppose that we apply the single-job streaming procedure to all jobs independently as mentioned in Chapter 2, and find the corresponding optimal sequence of transfer batches by using Johnson's Algorithm. Let us denote this sequence by σ_{Strm}. Note that σ_{Strm} is feasible solution with respect to the property for the sub problem P_A given in Corollary 6.1. Thus, we set $\sigma^{(1)} = \sigma_{strm}$ in the initialization step of the algorithm.

Example 6.2

To illustrate Algorithm 6.1, consider the following three-job problem in which each job has a lot size of 100 items and split into two sublots (transfer batches).

	Processing time per item	
Job	Machine 1	Machine 2
1	2	4
2	1	5
3	3	5

To obtain an initial sequence, assume that each job is independently split into two sublots. That is, the size of the ith sublot of job j is

$$q_{j,i} = Q(\alpha - 1)/(\alpha^{s_j} - 1), \qquad j = 1,2,3;\ i = 1,2$$

where $\alpha = p_{j,2} / p_{j,1}$, $s_j = 2$ for $j = 1,2,3$. Then, we have

	Sublot sizes		Processing time of sublots on machine 1	
Job	Sublot 1	Sublot 2	Sublot 1	Sublot 2
1	33.33	66.67	66.67	133.33
2	16.67	83.33	16.67	83.33
3	37.50	62.50	112.50	187.50

Based on the processing times of the sublots on machine 1, we obtain an initial sequence

$$\sigma^{(1)} = \sigma_{Strm} = \{2_1,\ 1_1,\ 2_2,\ 3_1,\ 1_2,\ 3_2\}$$

where j_i is the ith sublot of job j. We set $L = \varnothing$, $\sigma^{(0)} = \varnothing$ and $I(\sigma^{(0)}) = +\infty$.

Iteration 1 :

In Step 1, we construct the transfer batch-sizing problem for the sequence $\sigma^{(1)}$ as follows:

$BS(\sigma^{(1)})$: Min I

s.to
$$I - q_1 \geq 0$$
$$I + 4q_1 - 2q_2 \geq 0$$
$$I + 4q_1 + 2q_2 - q_3 \geq 0$$
$$I + 4q_1 + 2q_2 + 4q_3 - 3q_4 \geq 0$$
$$I + 4q_1 + 2q_2 + 4q_3 + 2q_4 - 2q_5 \geq 0$$
$$I + 4q_1 + 2q_2 + 4q_3 + 2q_4 + 2q_5 - 3q_6 \geq 0$$
$$q_1 + q_3 = 100$$
$$q_2 + q_5 = 100$$
$$q_4 + q_6 = 100$$
$$I \geq 0,\ \ q_i \geq 0,\ i = 1,\ldots,6$$

The solution of the problem $BS(\sigma^{(1)})$ gives the minimum total idle time on machine 2 which corresponds to the sequence $\sigma^{(1)}$ as $I(\sigma^{(1)}) = I = 9.09$ and the following vector of sublot sizes

$$\bar{q}^{(1)}(\sigma^{(1)}) = \{q_1,\ldots,q_6\} = \{9.09,\ 22.73,\ 90.91,\ 0.00,\ 77.27,\ 100.00\}.$$

In a tabular form, the sublot sizes and their corresponding processing times on machine 1 is given as:

	Sublot sizes		Processing time of sublots on machine 1	
Job	Sublot 1	Sublot 2	Sublot 1	Sublot 2
1	22.73	77.27	45.46	154.54
2	9.09	90.91	9.09	90.91
3	0.00	100.00	0.00	300.00

Then, the list of sequences becomes $L = \sigma^{(1)}$ since $I(\sigma^{(1)}) = 9.09 < I(\sigma^{(0)}) = +\infty$. In Step 2(i), we solve the sequencing problem $S(\overline{q}^{(1)}(\sigma^{(1)}))$ with processing times of the sublots on machine 1 and obtain the following optimal sequence σ for a given vector of sublots $\overline{q}^{(1)}(\sigma^{(1)})$:

$$\sigma = \{3_1, 2_1, 1_1, 2_2, 1_2, 3_2\} \notin L$$

with a total idle time $I(\sigma) = 9.09$. Then we set $k = k+1 = 2$, $\sigma^{(2)} = \sigma$, $I(\sigma^{(2)}) = I(\sigma) = 9.09$ and go to Step 1 for the next iteration.

Iteration 2 :

In Step 1, we construct the new transfer batch-sizing problem $BS(\sigma^{(2)})$ for the sequence $\sigma^{(2)}$ and its solution gives the minimum total idle time on machine 2 which corresponds to the sequence $\sigma^{(2)}$ as $I(\sigma^{(2)}) = 5.46$ and the following vector of sublot sizes

$$\overline{q}^{(2)}(\sigma^{(2)}) = \{1.82, 9.09, 22.73, 90.91, 77.27, 98.18\}.$$

In a tabular form, the sublot sizes and their corresponding processing times on machine 1 is given as:

	Sublot sizes		Processing time of sublots on machine 1	
Job	Sublot 1	Sublot 2	Sublot 1	Sublot 2
1	22.73	77.27	45.46	154.54
2	9.09	90.91	9.09	90.91
3	1.82	98.18	5.46	294.54

Then, the list of sequences becomes $L = \sigma^{(2)}$ since $I(\sigma^{(2)}) = 5.46 < I(\sigma^{(1)}) = 9.09$. In Step 2(i), we solve the sequencing problem $S(\overline{q}^{(2)}(\sigma^{(2)}))$ with processing times of the sublots on machine 1 and obtain the following optimal sequence σ for a given vector of sublots $\overline{q}^{(2)}(\sigma^{(2)})$:

$$\sigma = \{3_1,\ 2_1,\ 1_1,\ 2_2,\ 1_2,\ 3_2\} \in L = \sigma^{(2)}$$

with a total idle time $I(\sigma) = 5.46$. Thus, we stop with a sequence $\sigma^* = \sigma^{(2)}$ of transfer batches, $I^* = I(\sigma^{(2)}) = 5.46$, and the vector of transfer batch sizes $q^* = \overline{q}^{(2)}(\sigma^{(2)})$ are as shown in the table above. Note that the corresponding makespan of this schedule is $C_{\max} = I^* + \sum_j p_{j,2} Q_j = 5.46 + 1400 = 1405.46$. However, as shown in the following table, the optimal makespan value of the lot-splitting problem is $C^*_{\max} = 1405$. Thus, our makespan value obtained using Algorithm 6.1 is only 0.032% worse than the optimal makespan value.

	Optimal makespan, $C^*_{\max}$
Without transfer batches	1500
With lot-streaming	1416.67
With lot-splitting	1405

For a limited number of problems, we tested our algorithm and perform only one iteration in each problem. For a given number of jobs (2,3,4 or 5 jobs), 10 problems are randomly generated where half of them allow 2 sublots and the other half allows 3 sublots for each job. The total processing times of jobs on the machines are randomly generated in the interval [10,100]. In most cases, the makespan is very close to the optimum value with the average 0.4 % deviation from the optimum makespan value. In these problems, the largest deviation we obtain is approximately 0.9 % of the optimum solution. From a practical point of view, these results indicate that our approach with only one iteration can have potential for being implemented easily.

6.2 Two-machine Flow Shop Problem with Transfer Time Between Machines

In this section, we consider the lot splitting problem in a two-machine flow shop problem but in this case transfer batches are assumed to be unit or equal sized for all jobs and there is a transfer time to move a transfer batch from machine 1 to machine 2. Furthermore, we assume that there is one transporter in the system (like an AGV). Note that the transporter is called an AGV (automated guided vehicle) for convenience only; it need not be automated nor should it be guided. It picks a completed transfer batch from the first machine, travels to the second machine in t units of time, delivers the transfer batch and takes r more time units to return to the first machine.

Furthermore, if the AGV is not available at the time when a transfer batch completed on the first machine, the machine becomes blocked until the transfer batch is removed by the AGV. When the AGV reaches machine 2, it delivers the transfer batch to the machine if it is free; otherwise it puts the transfer batch in a waiting line.

Let $p_{j,m}$ be the processing time of a transfer batch of job j, $(1 \le j \le N)$, on machine m, $(m = 1,2)$. Then, we have the following cases investigated:

Case 1: $p_{j,1} \ge t + r$ for all jobs, $1 \le j \le N$.
Case 2: $p_{j,1} < t + r$ for some job j, $1 \le j \le N$.

Case 1 $p_{j,1} \ge t + r$ for all jobs, $1 \le j \le N$.

Theorem 6.5 *If $p_{j,1} \ge t + r$ for all jobs, $1 \le j \le N$, all transfer batches of the same job are processed sequentially, i.e., no transfer batch of every job is intermingled with the transfer batches of other jobs, and job i precedes job j if*

$$\min\{p_{i,1}, p_{j,2}\} \le \min\{p_{j,1}, p_{i,2}\}.$$

Proof: Assume that we allow all transfer batches of all jobs to be scheduled independently, one by one. In this case, the problem involves the scheduling of $K = \sum_{i=1}^{N} s_j$ distinct objects where s_j is the number of transfer batches of a job j. Let (k) be a transfer batch which is sequenced in the kth position for a sequence of K transfer batches. Furthermore, let $C_{(k),m}$ denote the completion time of transfer batch (k) on machine m. Then the completion time of transfer batch (k) on machine 1 is given by

$$C_{(k),1} = \sum_{u=1}^{k} C_{(u-1),1} + p_{(k),1} = \sum_{u=1}^{k} p_{(u),1}$$

since $p_{(k),1} \ge t + r$ for $1 \le k \le K$. In other words, machine 1 is never blocked since the unit processing time of all jobs (processing time of all transfer batches) on machine 1 are at least equal to $t + r$. By the same reasoning, the completion time of transfer batch (k) on machine 2 is expressed as

$$C_{(k),2} = \max\{C_{(k-1),2}\ ,\ C_{(k),1} + t\} + p_{(k),2}.$$

By successive application of the expression above substituting $C_{(k),1}$, we obtain

$$C_{\max} = C_{(K),2} = \max_{0 \le u \le N} \{\sum_{k=1}^{u} p_{(k),1} - \sum_{k=1}^{u-1} p_{(k),2}\} + t + \sum_{k=1}^{N} p_{(k),2}.$$

Since the second and third terms in the expression above are constant which are independent of the sequence, it is sufficient to minimize the first term. Thus, in an optimal sequence of the transfer batches, transfer batch k is processed before transfer batch l if $\min\{p_{k,1}, p_{l,2}\} \le \min\{p_{l,1}, p_{k,2}\}$. If k and l belong to same job, then the inequality becomes equality which implies that all transfer batches of the same job are processed sequentially, i.e., in the optimal sequence, no transfer of a job is intermingled with the transfer batches of other jobs, and job i precedes job j if $\min\{p_{i,1}, p_{j,2}\} \le \min\{p_{j,1}, p_{i,2}\}$. ∎

Case 2 $p_{j,1} < t + r$ for some job j, $1 \le j \le N$.

Theorem 6.6 *If $p_{j,1} < t + r$ for some job j, $1 \le j \le N$, then, in the optimal sequence, either no transfer batch of every job is intermingled with the transfer batches of other jobs (i.e., jobs are processed sequentially), or at most one job is split and one of its transfer batches is intermingled with the other jobs.*

Proof: The proof is based on a known result. If we are considering the classical two machine flowshop scheduling problem without transfer batches, then it is known that for a specified pre sequence (one or more jobs), it is optimal to order the remaining jobs according to Johnson's rule.

Let $T = \{1,2,...,K\}$ be the set of all transfer batches and σ_1 be the sequence of transfer batches obtained by Johnson's rule with the modified processing times (denoted by primes) $p'_{j,1} = \max\{p_{j,1}, t + r\}$ and $p_{j,2}$, i.e., in sequence σ_1, transfer batch i precedes transfer batch j if $\min\{p'_{i,1}, p_{j,2}\} \le \min\{p'_{j,1}, p_{i,2}\}$.

Suppose that we move a transfer batch (k) in the kth position $(2 \le k \le K)$ of the sequence σ_1 to the first position in σ_1 and create a new sequence σ_k. By this way, the modified processing times of the remaining $K-1$ transfer batches are not affected. This implies that the processing order of transfer batches in σ_k will be same as in σ_1.

If $p_{(k),1} < t + r$, then $p'_{(k),1}$ reduces from $t + r$ to $p_{(k),1}$. Hence, according to Johnson's rule with modified processing times, $C_{\max}(\sigma_k) \le C_{\max}(\sigma_1)$ where $C_{\max}(\sigma_k)$ and $C_{\max}(\sigma_1)$ are the corresponding makespans of sequences σ_k and σ_1, respectively. On the other hand, if $p_{(k),1} \ge t + r$, then the modified processing time of

the transfer batch (k) remains same, i.e., $p'_{(k),1} = p_{(k),1}$. But, in this case, moving transfer batch (k) to the first position contradicts to Johnson's rule with modified processing times. In other words, $C_{\max}(\sigma_k) > C_{\max}(\sigma_1)$. Thus, the only way to improve the makespan is to move a transfer batch (k) in the kth position $(2 \le k \le K)$ of the sequence σ_1 to the first position in σ_1 if $p_{(k),1} < t + r$. By this way, we create at most $K-1$ different sequences in addition to σ_1. However, it is sufficient to create at most $N-1$ different sequences by bringing only one transfer batch of each job j with $p_{j,1} < t + r$ since processing times of the transfer batches of job j are equal. This implies that remaining $s_j - 1$ transfer batches of job j are sequenced in an unbroken string, i.e., they are processed sequentially. Thus, one of these N sequences (at most) gives the optimal sequence with minimum makespan. ∎

Based on Theorems 6.5 and 6.6, we propose the following algorithm.

Algorithm 6.2

Step 1 Let $p'_{j,1} = \max\{p_{j,1},\ t + r\}$ for $j \in J = \{1,2,...,N\}$.

Step 2
(i) Decompose the job set J into two parts as:
$$J_A = \{j \mid p'_{j,1} \le p_{j,2}\} \text{ and } J_B = \{j \mid p'_{j,1} > p_{j,2}\}.$$
(ii) Sort the job numbers of set J_A in non-decreasing order of $p'_{j,1}$, and denote the resulting list by J'_A.
(iii) Sort the job numbers of set J_B in non-increasing order of $p_{j,2}$, and denote the resulting list by J'_B.
(iv) Let $\sigma_1 = \{J'_A, J'_B\}$. Calculate the corresponding makespan $C_{\max}(\sigma_1)$.
(v) If $p'_{j,1} = p_{j,1}$ for every job $j \in J$, then σ_1 is an optimal sequence, stop; otherwise go to Step 3.

Step 3
(i) Set $j = 2$.
(ii) If $j > N$, go to Step 4; otherwise, let the job in the jth position of σ_1 be job z.
(iii) If $p_{z,1} \ge t + r$, set $C_{\max}(\sigma_j)$ to infinity, and go to Step 3(iv); otherwise move a transfer batch of job z to the first position in σ_1 to obtain sequence σ_j. Calculate $C_{\max}(\sigma_j)$ and go to Step 3(iv).
(iv) Set $j = j + 1$. Return to Step 3(ii).

Step 4 If $C_{\max}(\sigma_k) = \min_{1 \le j \le N}\{C_{\max}(\sigma_j)\}$, then σ_k is an optimal sequence and $C_{\max}(\sigma_k)$ is the corresponding minimal makespan.

Example 6.3

To illustrate Algorithm 6.2, consider a three-job problem in which the travel time t equals to 2 units of time from machine 1 to machine 2 and r equals to 1 unit of time for the return from machine 2 to machine 1. Step 1 of the Algorithm 6.2 to calculate $p'_{j,1}$ is already included in the data shown below:

TABLE 6.2. Data for the three-job problem

Job j	Number of sublots s_j	Processing time per sublot Machine 1 $p_{j,1}$	Machine 2 $p_{j,2}$	Modified processing time per sublot Machine 1 $p'_{j,1}$
1	3	3	4	3
2	3	2	3	3
3	3	2	2	3

Step 2 gives us the sequence $\sigma_1 = \{1_1, 1_2, 1_3, 2_1, 2_2, 2_3, 3_1, 3_2, 3_3\}$ in which j_i denote the ith transfer batch of job j. The makespan $C_{\max}(\sigma_1)$ for this sequence is 32. The sequences and their corresponding makespans obtained in step 3 can be summarized as follows:

$$\sigma_2 = \{\mathbf{2_1}, 1_1, 1_2, 1_3, 2_2, 2_3, 3_1, 3_2, 3_3\} \qquad C_{\max}(\sigma_2) = 31$$

$$\sigma_3 = \{\mathbf{3_1}, 1_1, 1_2, 1_3, 2_1, 2_2, 2_3, 3_2, 3_3\} \qquad C_{\max}(\sigma_3) = 32$$

Since $C_{\max}(\sigma_2) = \min\{32, 31, 32\} = 31$, sequence σ_2 is optimal (see Figure 6.3).

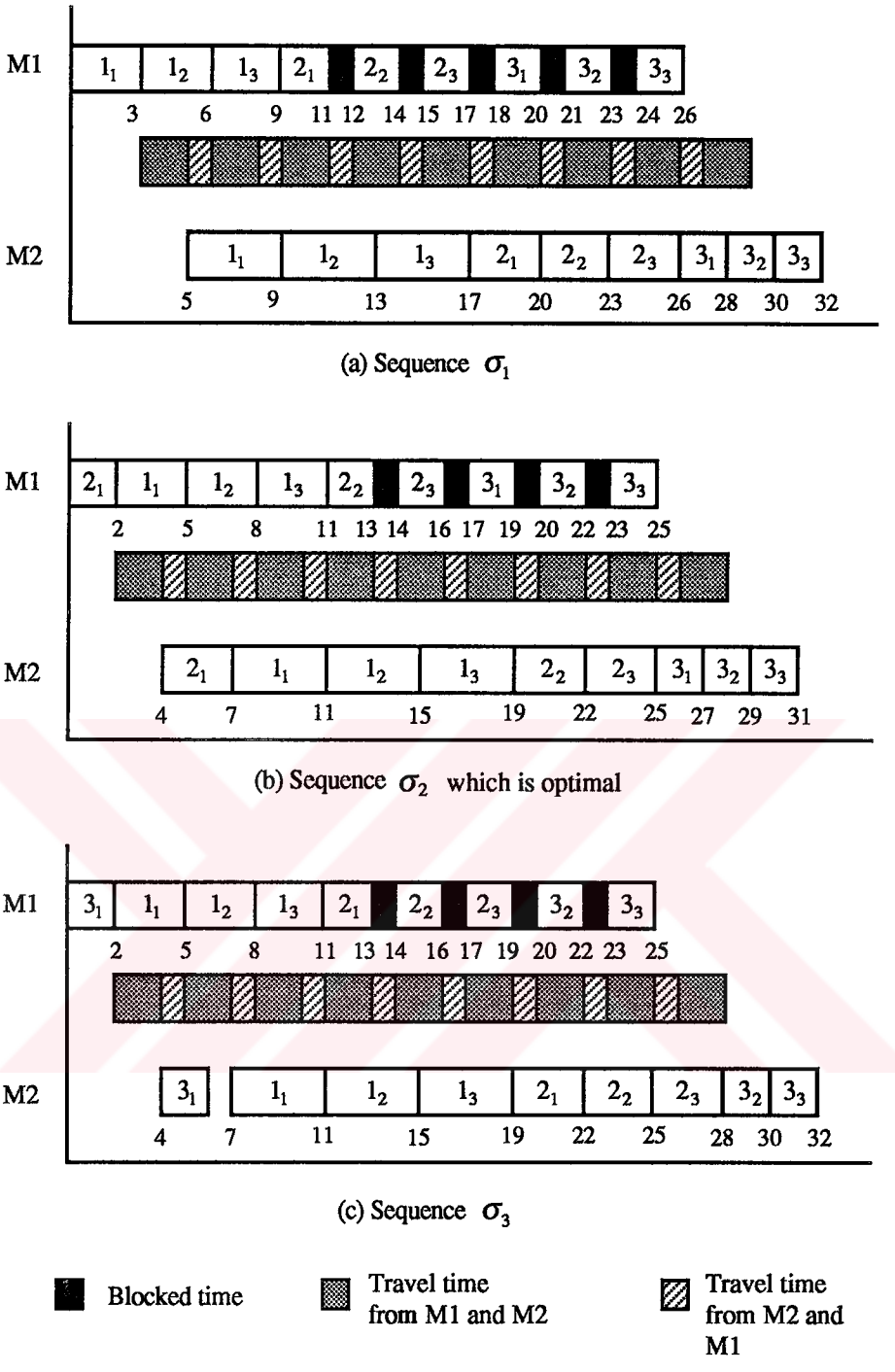


Figure 6.3. Schedules obtained in Example 6.3.

CHAPTER 7

CONCLUSIONS AND FURTHER RESEARCH

The advances in manufacturing automation and the changing nature of business competitiveness created demand for viable production systems. Production scheduling with transfer batches is one such avenue of research. By utilizing transfer batch concept in production scheduling, decreased throughput time of an order can be achieved, which in turn equates to decreased costs. These savings benefit not only the manufacturing company, but the customer as well. Also, there is an increase in production capacity, which allows more orders to be processed over a given period of time. It appears that the use of transfer batches in scheduling can offer significant improvements in the overall efficiency and productivity of a production system.

In this thesis, we have investigated number of scheduling problems for multi-stage shops in which transfer batches between stages are allowed, identified their well solvable cases and derived the basic solution characteristics. For this purpose, we first dealt with single-job streaming problems, later extended our results to the multi-job streaming and splitting problems. New insights have been gained as well, not only for single-job problems, but also for multi-job problems. These insights should be useful in improving our ability to deal with more complex problems.

In this thesis, we first examined some single-job streaming problems in which the measure of performance is taken as the makespan of the job. One of these problems is the two-stage hybrid flow shop problem where stages may consist of multiple identical parallel machines. We proved that the results for the two-machine flow shop problem in which there is only one machine at both stages can easily be extended to our case. According to the characteristic of the optimal solution, processing work of each transfer batch must be equally allocated to the multiple machines on each stage. Furthermore, the optimal schedule is a no-wait and no-idle schedule as in the case of two-machine problem with only one machine at both stages.

We next studied the general M-machine flow shop streaming problem with variable sized transfer batches by extending the results of the study by Trietsch and

Baker (1993) for the three-machine flow shop problem with variable transfer batch sizes. Here, the problem is to simultaneously find the optimal partition set of machines that operates continuously without any idle time between processing of transfer batches and corresponding optimal transfer batches. We showed that the formation of the partition set depends on the relative sizes of the unit processing times on the machines. Furthermore, the optimal transfer batches between any pair of machines in the partition set are consistent and have a geometric progression. Based on these properties of the optimal solution, we proposed a polynomial time algorithm.

In Section 3.3, the streaming problem in a three-machine flow shop with variable transfer batches and detached setup times on the machines was discussed. We identified several cases which depend on the relation between processing and setup times. Except only one case, we proved that either the optimal transfer batches are consistent through all three machines or variable in size. We left the unproved case to be studied as a future research topic.

In the last section of Chapter 3, we investigated the solution complexity of the general M-machine open shop problem in which the number of transfer batches is greater than or equal to three and less than the number of machines in the shop. We proved that this problem is NP-hard.

In Chapter 4, we again considered single-job streaming problems with consistent transfer batches in a general M-machine flow shop for a different measure of performance which is the mean flow time of the transfer batches. We first showed that streaming problem with consistent transfer batches for the flow shops satisfies the properties of the ordered flow shops. Based on this fact, we proved that in the optimal solution of our new problem transfer batches must be ordered in non-decreasing in their size. Then we investigated some special cases of the general problem. In the first case in which the first machine has the longest unit processing time, equal transfer batch sizes gives the optimal solution. In another case, we assumed that only two transfer batch are allowed between machines. We derived the properties of the optimal solution and proposed a search procedure which is polynomial time. We examined the two-machine case along with its extension with detached setup time on the machines. We then derived the properties of the optimal solution except for only one case which was conjectured. Hence, the proof of the conjecture is a promising direction for further research.

The last special case discussed in Chapter 4 is the three-machine case with consistent transfer batches. We first proved that in the optimal solution of the

problem, machine having the longest unit processing time operates continuously without any idle time between processing of transfer batches as in the case of two-machine. We then proved that when machine 2 or 3 has the longest processing time transfer batches follow a geometric progression from the first transfer batch through a pattern changing transfer batch and the remaining transfer batches are equal each other.

In Chapter 5, we analyzed the multi-job streaming problems in two-machine flow and open shops with makespan criterion. We first showed that transfer batch sizing and sequencing decisions are made independently in the two-machine flow shop problem. Then we conjectured that this independent analysis of transfer batching and sequencing decisions are also true for the case with the no-wait-in-process restriction on the processing of transfer batches. Note that the conjecture is optimal when the transfer batch sizes are given or equal sized.

The next topic in Chapter 5 was the extension of the results obtained by Şen *et al.* (1995) for the two-machine open flow shop problem in which the routing of job is fixed to the same production environment, but detached setup times exist. We obtained similar results to that of Şen *et al.* (1995). In the optimal solution, either there is no need for lot streaming or at most one job will be streamed. Based on this property, we also showed that the optimal schedule can be obtained by using the algorithm proposed by Gonzalez and Sahni (1976) for the classical two-machine open shop scheduling problem.

In Chapter 6, we considered the multi-job splitting problems in two-stage flow shops. Here, we again took the makespan as the performance measure. In the first section of this chapter, we examined the basic two-machine flow shop problem. We first decomposed the jobs into two groups: jobs whose unit processing time on the first machine is less or equal to its processing time on the second machine, and jobs whose unit processing time on the first machine is greater than its processing time on the second machine. Then we proved that the problem decomposes into two sub problems in which the first sub problem is to find simultaneously size and sequence of transfer batches in the first job set, and the second sub problem is again to find simultaneously size and sequence of transfer batches in the second job set. We also showed that knowing the properties of the optimal solution to one of these sub problems is sufficient to solve the overall problem. We then proposed a two-level solution procedure which gives very good approximate results as compared to the optimal solution.

In Section 6.2, we considered two-machine flow shop problem in which transfer times between machines exist. But, in this problem, we assumed that transfer batch sizes for all jobs are given as unit or equal sized. We proved that in the optimal solution of this problem either no job is split or at most one job is split and one of its transfer batches is intermingled with the other jobs.

There are several related research directions that are worthy of mention.

• The notion of machine dominance (identification of bottleneck machine) is a critical issue in the solution of single-job streaming problem with consistent or variable transfer batches as shown in the two and three-machine flow shops to minimize both makespan and mean (total) flow time. As of yet, no low order polynomial time algorithms have been proposed for these problems other than the standard linear and quadratic programming models. The generalization of the notion of dominance to M machines would provide useful insights to have algorithms which solves the problem in a quick and straightforward manner. Thus, the development of such low order polynomial algorithms which will solve these problems either optimally or near optimally and their empirical assessment may be a subject of future research.

• In multi-job streaming problems, unfortunately, machine dominance results of traditional scheduling theory for flow shops containing more than two machines are not directly applicable. A dominant machine (i.e., non-bottleneck) for single-job streaming machine may turn out to be bottleneck in a multi-job streaming problem. Thus transfer batch sizing problem is not independent from the job sequencing problem, and they must be solved simultaneously. Furthermore, polynomial time algorithms available for the classical flow shop scheduling problems are not directly applicable for the multi-job streaming problems. However, as we present a two-level iterative solution procedure to the multi-job splitting problem, a hierarchical approach enables transfer batch sizing and sequencing to be integrated for the case of multi-job streaming and splitting problems. Hence, further research is required to address the integrated solution procedures either exact or heuristic.

• Just as the two-machine analysis represents a key building block in the optimization of special cases of the general M-machine problem with multiple jobs, we might expect that it could also play a role in the development of heuristic procedures for the general case. The heuristic algorithm of Champbell et al. (1970) could be adapted readily to the lot streaming model. Thus, one opportunity for future research would be a study of heuristic solutions to the general lot streaming problem. The optimizing algorithm of Lageweg et al. (1978) could be adapted as well. This algorithm could

exploit available theory to solve multi-machine streaming problems with setup times and equal sized batches. If we can not develop such techniques, then it may be convenient to impose some simplifying constraint in order to exploit the M-machine solutions mentioned earlier. In this case, it would be helpful to know something about how suboptimal such solutions could be. Thus, the development of worst-case bound for specially-constrained solutions to the lot streaming problem as early steps in this direction described by Potts and Baker (1989) would be desirable.

- Another important area for future research is to investigate the complexity of various models. Specifically, the general complexity of problems $1|F|TB(C,c \text{ or } d)|C_{\max}$ and $1|F|TB(C \text{ or } V,c \text{ or } d)|\sum q_i C_i$ is open at this stage.

- Major drawback of scheduling problems with transfer batches is that the number of transfer batches is mostly taken as a parameter rather than a decision variable. In these problems, it is assumed that material handling equipment (transporter) between stages (machines) in the shop is always available and transfer times are negligible. However, in reality, the capacity of the transporter is limited and there is a move-time between stages, thus affects the formation of transfer batches. Hence, further research is necessary to rework on the existing models to include this realistic aspect of the multi-stage systems.

- Most of the research on scheduling problems with transfer batches considers transfer batches either consistent or variable throughout the shop. However, combinations of consistent and variable transfer batch models, where the transfer batches would be held consistent across subsets of machines, but allowed to change between subsets, would balance the cost of tracking different sized transfer batches with the improvement of scheduling criteria associated with variable transfer batches. Thus, this issue needs to be studied.

- The different objective functions can be easily incorporated into the models within the framework of scheduling with transfer batches. For example, consider the single job streaming problem with consistent transfer batches. Suppose that there is a due date for the job at which all transfer batches must be completed, and our objective is to minimize the number of tardy transfer batches.

- Although we need integer valued (discrete) transfer batch sizes for most practical applications, much less is known about the discrete version of the streaming problems.

• There is also a need for mathematical models to allow for inclusion of more realistic aspects of multi-stage systems. For example, an extention of the model proposed by Benli (1994) would be the costruction of a general model involving detached or attached setups, transfer time between machines and variable transfer batch sizes.

• In this thesis, we only investigated the scheduling problems for static environments. However, transfer batch concept should be studied for the dynamic environments in which the scheduling problem is defined also with respect to the anticipation for additional requirements and specifications generated over the future time period.

• Transfer batches also play an important role in the design of material handling systems. The determination of appropriate transfer batch sizes are crucial in deciding the tradeoffs between material handling efficiency and the expected work in process inventories in the system. Thus, this topic may be another research area.

• The design of an interactive human-computer integrated scheduling system which allows several aspects of operations overlapping would be desirable.

• Within the framework of Group Technology (GT), an identification of families of components is made so that a group of machines is made available only for the production or processing of the members of that family of components. The primary motivation for GT is the reduction of setup costs which are incurred during the transition of production from one component or product type to another. However, the use of GT to reduce setup costs does not preclude the use of other techniques which may result in a reduction of cycle time and work-in-process inventory. In fact, manufacturing cells, which result from the imposition of GT, are opportune settings in which lots can be divided into sublots (transfer batches) so that the processing or production of these transfer batches can be achieved simultaneously. Thus, in this context there is an opportunity to apply the transfer batch concept.

• The basic problem with production planning is that the sequencing requirements on the resources can not easily accommodated, whereas machine scheduling can not explicitly handle lot sizing. Future significance of transfer batches may possible lie in the integration of lot-sizing and machine scheduling.

REFERENCES

Achugbue, J.O., and Chin, F.Y. (1982), "Scheduling the Open Shop to Minimize Mean Flow Time", *SIAM Journal on Computing*, 11, 709-720.

Baker, K.R. (1974), *Introduction to Sequencing and Scheduling*, John Wiley, New York, NY.

Baker, K.R. (1988), "Lot Streaming to Reduce Cycle Time in a Flow Shop", Working Paper No: 203, The Amos Tuck School of Business Administration, Dartmouth College, Hanover, New Hampshire.

Baker, K.R. (1992), *Elements of Sequencing and Scheduling*, The Amos Tuck School of Business Administration, Dartmouth College, Hanover, New Hampshire.

Baker, K.R. and Pyke, D.F. (1990), "Solution Procedures for the Lot Streaming Problem", *Decision Sciences*, 21, 475-491.

Baker, K.R., and Jia, D. (1993), "A Comparative Study of Lot Streaming Procedures", *OMEGA International Journal of Management Science*, 21, 561-566.

Bellman, R., Esogbue, A.O., and Nabeshima, I. (1982), *Mathematical Aspects of Scheduling and Applications*, Oxford, Pergamon Press.

Benli, Ö. (1994), "Streaming a Single Job in a Flow Shop", Research Report IEOR-9409, Department of Industrial Engineering, Bilkent University, Ankara.

Brah, S.A., and Hunsucker, J.L. (1991), "Branch and Bound Algorithm for the Flow Shop with Multiple Processors", *European Journal of Operational Research*, 51, 88-99.

Cho, Y., and Sahni, S. (1980), "Preemptive Scheduling of Independent Jobs with Release and Due Times on Open, Flow and Job Shops", *Operations Research*, 29, 511-522.

Conway, R.W., Maxwell, W.L., and Miller, L.W. (1967), *Theory of Scheduling*, Addision-Wesley, Reading, Massachusetts.

Çetinkaya, F.C. (1994), "Lot Streaming in a Two-stage Flow Shop with Setup, Processing and Removal Times Separated", *Journal of the Operational Research Society*, 45, 1445-1455.

Çetinkaya, F.C., and Gupta, J.N.D. (1994), "Flowshop Lot Streaming to Minimize Total Weighted Flow Time", Research Memorandum No. 94-24, School of Industrial Engineering, Purdue University, West Lafayette, Indiana.

Çetinkaya, F.C., and Gupta, J.N.D. (1995), "Optimal Lot Streaming in a Two-stage Hybrid Flowshop", Technical Report No: 95-01, Department of Industrial Engineering, Middle East Technical University, Ankara.

Çetinkaya, F.C., and Kayalıgil, M.S. (1992), "Unit Sized Transfer Batch Scheduling with Setup Times", *Computers and Industrial Engineering*, 22, 177-183.

Dauzere-Peres, S., and Lasserre, J.R. (1993), "An Iterative Procedure for Lot Streaming in Job Shop Scheduling", *Computers and Industrial Engineering*, 25, 231-234.

Deal, D.E., and Hunsucker, J.L. (1991), "The Two-stage Flowshop Scheduling Problem with M Machines at Each Stage", *Journal of Information and Optimization Sciences*, 12, 407-417.

Dempster, M.A.H., Lenstra, J.K., and Rinnooy Kan, A.H.G. (1982), *Deterministic and Stochastic Scheduling*, D. Reidel Publishing, Dordrecht, Holland.

Dobson, G, Karmarkar, U.S., and Rummel, J.L. (1987), "Batching to Minimize Flow Times on One Machine", *Management Science*, 33, 784-799.

Flynn, B.B. (1987), "Repetitive Lots: The Use of a Sequence Dependent Set-Up Time Scheduling Procedure in Group Technology and Traditional Shops", *Journal of Operations Management*, 7.

Fogarty, D.W., Blackstone, J.H., Hoffman, T.R. (1991), *Production and Inventory Management*, South-Western Pub. Co., Cincinnati, Ohio.

French, S. (1982), *Sequencing and Scheduling*, Ellis Horwood, Chichester, England.

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

Garey, M.R., Johnson, D.S., and Sethi, R. (1976), "The Complexity of Flowshop and Jobshop Scheduling", *Mathematics of Operations Research*, 11, 117-129.

Gilmore, P.C., and Gomory, R.E. (1964), "Sequencing a None State-variable Machine: a solvable case of traveling salesman problem", Operations Research, 12, 665-679.

Gilmore, P.C., Lawler, E.L., and Shmoys, D.B. (1986), "Well-solved Special Cases", In: *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization* (E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys eds.), Willey, New York, 87-143.

Glass, C.A., Gupta, J.N.D., and Potts, C.N. (1994), "Lot Streaming in Three-stage Production Processes", *European Journal of Operational Research*, 75, 378-394.

Gonzalez, T., and Sahni, S. (1976), "Open Shop Scheduling to Minimize Finish Time", *Journal of the Association for Computing Machinery*, 23, 665-679.

Gonzalez, T., and Sahni, S. (1978), "Flowshop and Jobshop Schedules: Complexity and Approximation", *Operations Research*, 26, 36-52.

Goyal, S.K. (1976), "Note on Manufacturing Cycle Time Determination for a Multi-stage Economic Production Quantity Model", *Management Science*, 23, 332-333.

Goyal, S.K. (1977), "Determination of Optimum Production Quantity for a Two-stage Production System", *Operational Research Quarterly*, 28, 865-870.

Goyal, S.K., and Gunasekaran, A. (1994), "Determination of Economic Production Quantity in a Multi-stage Production System", *Production Planning and Control*, 5, 499-503.

Graham, R.L., Lawler, E.L., Lenstra, J.K., and Rinnooy Kan, A.H.G. (1979), "Optimization and Approximation in Deterministic Scheduling and Sequencing: a survey", *Annals of Discrete Mathematics*, 5, 287-326.

Graves, S.C. (1981), "A Review of Production Scheduling", *Operations Research*, 29, 646-675.

Graves, S.C., and Kostreva, M.M. (1986), "Overlapping Operations in Material Requirements Planning", *Journal of Operations Management*, 6, 283-294.

Gupta, J.N.D. (1988), "Two-stage Hybrid Flowshop Scheduling Problem", *Journal of the Operational Research Society*, 39, 359-364.

Gupta, J.N.D., and Tunç, E.A. (1991), "Schedules for a Two-stage Hybrid Flowshop with Parallel Machines at the Second Stage", *International Journal of Production Research*, 29, 1489-1502.

Herrmann, W.J, Chung-Yee, L., Snowdon, J.L. (1993), "A Classification of Static Scheduling Problems", *Complexity in Numerical Optimization*, edited by P.M. Pardalos, World Scientific Publishing Co.

Jackson, J.R. (1956), "An Extension of Johnson's Results on Job Scheduling", *Naval Research Logistics Quarterly*, 1, 61-68.

Jacob, F.R. (1984), "OPT Uncovered: Many Production Planning and Scheduling Concepts can be Applied with or without the Software", *Industrial Engineering*, 16, 32-41.

Jacob, F.R., and Bragg, D.J. (1988), "Repetitive Lots: Flow-time Reductions Through Sequencing and Dynamic Batch Sizing", *Decision Sciences*, 19, 281-294.

Johnson, S.M. (1954), "Optimal Two and Three-stage Production Schedules with Setup Times Included", *Naval Research Logistics Quarterly*, 1, 61-68.

Kayalıgil, M.S., and Çetinkaya, F.C. (1991), "İki Safhalı Taşıma Süreli Akış Tipi Atelyede Kafile Aktarma ve Çizelgelemesi (Lot Streaming and Scheduling in a Two-stage Flow Shop with Transportation Time)", *Endüstri Mühendiliği*, 4, 3-11 (in Turkish).

Kropp, D.H., and Smunt, T.L. (1990), "Optimal and Heuristic Models for Lot Splitting in a Flow Shop", *Decisions Sciences*, 21, 691-709.

Kulonda, D.J. (1984), "Overlapping Operations- A Step Toward Just-in-time Production", *Readings in Zero Inventory*, APICS 27th Annual International Conference, Falls Church, VA. 78-80.

Lageweg, B.J., Lawler, E.L., Lenstra, J.K., and Rinnooy Kan, A.H.G. (1982), "Computer Aided Complexity Classification of Combinatorial Problems", *Communications of the Association for Computing Machinery*, 25, 817-822.

Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., and Shmoys, D.B. (1989), "Sequencing and Scheduling: Algorithms and Complexity", Report 8945/A, Econometric Institute, Erasmus University, Rotterdam.

Lenstra, J.K., Rinnooy Kan, A.H.G., and Brucker, P. (1977), "Complexity of Machine Scheduling Problems", *Annals of Discrete Mathematics*, 1, 343-362.

Liu, C.Y., and Buflin, R.L. (1986), 'On the Complexity of Open-shop Scheduling Problems", *Operations Research Letters*, 4, 71-74.

Lundgrian, R. (1986), "What is this thing called OPT?", *Production and Inventory Management*, 27, 2-12.

Masuda, T., Ishii, H., and Nishida, T. (1985), "The Mixed Shop Scheduling Problem", *Discrete Applied Mathematics*, 11, 175-186.

Mitten, L.G. (1959), "Sequencing n Jobs on Two Machines with Arbitrary Time Lags", *Management Science*, 5, 293-298.

Miyazaki, S., and Nishiyama, N. (1980), "Analysis for minimizing weighted mean flow-time in flow-shop scheduling", *Journal of the Operations Research Society of Japan*, 23, 118-131.

Monma, C.L., and Rinnooy Kan, A.H.G. (1983), "A Concise Survey of Efficiently Solvable Special Cases of the Permutation Flowshop Problem", *RAIRO Recherche Operationelle*, 17, 105-119.

Morris, J.S., and Tersine, R.J. (1989), "A Comparison of Cell Loading Practices in Group Technology', Journal Manufacturing Operations Management, 2.

Morton, T.E., and Pentico, D.W. (1993), *Heuristic Scheduling Systems: with Applications to Production Systems and Project Management*, John Wiley and Sons Inc., New York.

Panwalker, S.S., and Khan, A.W. (1976), "An Ordered Flow-shop Sequencing Problem with Mean Completion Time Criterion", *International Journal of Production Research*, 14, 631-635.

Potts, C.N., and Baker, K.R. (1989), "Flow Shop Scheduling with Lot Streaming", *Operations Research Letters*, 8, 297-303.

Potts, C.N., and Van Wassenhove, L. N. (1992), "Integrating Scheduling with Batching and Lot-Sizing: a Review of Algorithms and Complexity", *Journal of the Operational Research Society*, 43, 395-406.

Proust, C., Gupta, J.N.D., and Deschamps, V. (1991), "Flow Shop Scheduling with Setup, Processing and Removal Times Separated", *International Journal of Production Research*, 29, 479-493.

Reither, S. (1966), "A System for Managing Job Shop Production", *Journal of Business*, 34, 371-393.

Rinnooy Kan, A.H.G. (1976), *Machine Scheduling Problems: Classification, Complexity and Computations*, The Hague: Martinus Nijhoff.

Roeck, H. (1984a), "Three-machine No-wait Flow Shop Problem is NP-complete", *Journal of the Asosociation for Computing Machinery*, 31, 336-345.

Roeck, H. (1984b), "Some New Results in Flow Shop Scheduling", *Z. Oper. Res.*, 28, 1-16.

Salvador, M.S. (1973), "A Solution to a Special Case of Flow Shop Scheduling Problems", Symposium of the Theory and Scheduling and Applications. Edited by S.E. Elmaghraby. Springer-Verlag. New York.

Santos, C., and Magazine, M. (1985), "Batching in Single Operation Manufacturing Systems", *Operations Research Letters*, 4, 99-103.

Sassani, F. (1990), "A Simulation Study on Performance Improvement of Group Tevhnology Cells", *International Journal of Production Research*, 28.

Smith, M.L., Panwalker, S.S., and Dudek, R.A. (1975), "Flow-shop sequencing problem with ordered processing time matrices", *Management Science*, 21, 544-549.

Steiner, G., and Truscott, W.G. (1993), "Batch Scheduling to Minimize Cycle Time, Flow Time, and Processing Cost", *IIE Transactions*, 25, 90-97.

Strusevich, V.A. (1990), "Two Machine Flow Shop No-wait Scheduling Problem with Setup, Processing and Removal Times Separated", Report 9094/A, Econometric Institute, Erasmus University, Rotterdam.

Sule, D.R., and Huang, K.Y. (1983), "Sequency on Two and Three Machines with Setup, Processing and Removal Times Separated", *International Journal of Production Research*, 21, 723-732.

Szendrovits, A.Z. (1975), "Manufacturing Cycle Time Determination for a Multi-stage Economic Production Quantity Model", *Management Science*, 22, 298-308.

Szendrovits, A.Z. (1976), "On the Optimality of Sub-batch Sizes for a Multi-stage EPQ Model-A Rejoinder", *Management Science*, 23, 334-338.

Szendrovits, A.Z. (1978), "A Comment on Determination of Optimum Production Quantity for a Two-stage Production System", *Journal of Operational Research Society*, 29, 1017-1020.

Szendrovits, A.Z., and Drezner, Z. (1980), "Optimizing Multi-stage Production with Constant Lot Size and Varying Number of Batches", *OMEGA International Journal of Management Science*, 8, 623-629.

Şen, A., Topaloğlu, E., and Benli , Ö. (1994), "Optimal Streaming of a Single Job in a Two Stage Flow Shop", Research Report No. IEOR-9413, Department of Industrial Engineering, Bilkent University, Ankara.

Şen, A., Benli , Ö., and Akyel, C.H. (1995), "Açık Atelyelerde Kafile Aktarma/Lot Streaming in Open Shops", Paper presented at the 17th National OR/IE Congress, Department of Industrial Engineering, METU, Ankara.

Tan, T. (1996), "Modelling and Experimentation for Lot Splitting in Flow Shops", M.S. Thesis, Department of Industrial Engineering, METU, Ankara.

Trietsch, D. (1987), "Optimal Transfer Lots for Batch Manufacturing: A Base Case and Extension", Technical Report No: NPS-54-87-010, Naval Postgraduate School, Monterey, California.

Trietsch, D. (1989), "Polynomial Transfer Lot Sizing Techniques for Batch Processing on Consecutive M Machines", Technical Report No: NPS-54-89-011, Naval Postgraduate School, Monterey, California.

Trietsch, D., and Baker, K.R. (1993), "Basic Techniques for Lot Streaming", *Operations Research* , 41, 1065-1076.

Truscott, W.G. (1985), "Scheduling Production Activities in Multi-stage Batch Manufacturing Systems", *International Journal of Production Research*, 23, 315-328.

Truscott, W.G. (1986), "Production Scheduling with Capacity-Constrained Transportation Activities", *Journal of Operations Management*, 6, 333-348.

Umble, M., and Srikanth, M. (1990), *Synchronous Manufacturing*, South-Western Publishing Co., Cincinnati.

Vickson, R.G. (1993), "Optimal Lot Streaming for Multiple Products in a Two-machine Flow Shop", European Journal of Operational Research, (forthcoming).

Vickson, R.G., and Alfredsson, B.E. (1992), "Two-and Three-machine Flow Shop Scheduling Problems with Equal Sized Transfer Batches", *International Journal of Production Research*, 30, 1551-1574.

Vollman, T. (1986), "OPT as An Enhancement to MRP II", *Production and Inventory Management*, 38-46.

Williams, E.F., and Tüfekçi, S. (1992), "Polynomial Time Algorithms for the Lot Streaming Problem", Research Report 92-10, Department of Industrial and Systems Engineering, University of Florida, Gainesville, Florida.

Yoshida, T., and Hitomi, K. (1979), "Optimal Two-stage Production Scheduling with Setup Times Separated", *AIIE Transactions*, 11, 261-263.

APPENDIX A

COMPLEXITY OF FLOW SHOP AND OPEN SHOP SCHEDULING PROBLEMS

TABLE A.1. Complexity of flow shop scheduling problems

Problem	Complexity†	Reference
$N\|F2\|\|C_{\max}$	$O(N \log N)$	Johnson (1954)
$N\|F2\|pmtn\|C_{\max}$	$O(N \log N)$	Johnson (1954), Gonzalez and Sahni (1978)
$N\|F2\|S_d\|C_{\max}$	$O(N \log N)$	Yoshida and Hitomi (1979)
$N\|F2\|S_d, R\|C_{\max}$	$O(N \log N)$	Sule and Huang (1983)
$N\|F2\|NW\|C_{\max}$	$O(N \log N)$	Gilmore *et. al* (1986)
$N\|F2\|NW, S_d, R\|C_{\max}$	$O(N \log N)$	Strusevich (1990)
$N\|F2\|NI\|C_{\max}$	$O(N \log N)$	Conway *et. al* (1967)
$N\|F3\|\|C_{\max}$	*NP*	Garey *et. al* (1976)
$N\|F3\|pmtn\|C_{\max}$	*NP*	Gonzalez and Sahni (1978)
$N\|F3\|NW\|C_{\max}$	*NP*	Roeck (1984a)
$N\|F2\|\|\sum C_j$	*NP*	Garey *et. al* (1976)
$N\|F2\|pmtn\|\sum C_j$	*open*	Lawler *et. al* (1989)
$N\|F2\|NW\|\sum C_j$	*NP*	Roeck (1984b)
$N\|F3\|pmtn\|\sum C_j$	*NP*	Graham *et. al* (1979)
$N\|F2\|\|L_{\max}$	*NP*	Lenstra *et. al* (1977)
$N\|F2\|pmtn\|L_{\max}$	*NP*	Cho and Sahni (1981)
$N\|F2\|NW\|L_{\max}$	*NP*	Roeck (1984b)

† *NP*: *NP*-hard, *open* : complexity of the corresponding problem is unknown.

TABLE A.2. Complexity of open shop scheduling problems

Problem	Complexity	Reference
$N\|O2\|\|C_{\max}$	$O(N)$	Gonzalez and Sahni (1976)
$N\|O2\|pmtn\|C_{\max}$	$O(N)$	Gonzalez and Sahni (1976)
$N\|O3\|\|C_{\max}$	*NP*	Gonzalez and Sahni (1976)
$N\|O2\|\|\sum C_j$	*NP*	Achugbue and Chin (1982)
$N\|O2\|pmtn\|\sum C_j$	*open*	Lawler *et. al* (1989)
$N\|O3\|pmtn\|\sum C_j$	*NP*	Liu and Bulfin (1986)
$N\|O2\|\|L_{\max}$	*NP*	Lawler *et. al* (1982)
$N\|O2\|pmtn\|L_{\max}$	$O(N \log N)$	Lawler *et. al* (1982)

APPENDIX B

LINEAR PROGRAMMING MODELS PROPOSED BY BAKER (1988) FOR THE PROBLEM $1|F|TB(C,c)|C_{\max}$

Model 1:

$$\text{Min} \quad C_{s,M}$$

$$\text{s.to} \quad C_{1,1} \geq p_1 q_1$$

$$C_{i,m} \geq C_{i-1,m} + p_m q_i, \qquad i = 2,\ldots,s; \;\; m = 1,\ldots,M$$

$$C_{i,m} \geq C_{i,m-1} + p_m q_i, \qquad i = 1,\ldots,s; \;\; m = 2,\ldots,M$$

$$\sum_{i=1}^{s} q_i = Q$$

$$C_{i,m} \geq 0, \qquad i = 1,\ldots,s; \;\; m = 1,\ldots,M$$

$$q_i \geq 0, \qquad i = 1,\ldots,s.$$

where Q = lot size,

s = number of consistent transfer batches into which the lot can be sub-divided,

q_i = size of the ith transfer batch,

M = number of machines,

p_m = unit processing time of machine m.

$C_{i,m}$ = completion time of transfer batch i on machine m.

In this formulation, the goal is to find the minimum value of $C_{s,M}$, which represents the completion time of the last transfer batch s (makespan of the lot), the last transfer batch, on machine M, the last machine. The first constraint prevents the completion time of the first transfer batch on the first machine from being less than the time it takes to process the first transfer batch on the first machine. The second set of constraints prevents transfer batch i from beginning its processing on machine m before transfer batch $i-1$ has completed its processing on machine m. The third set of constraints restricts transfer batch i from beginning its processing on machine m before it has completed its processing on machine $m-1$. The fourth type of constraint

ensures that the sum of all transfer batches constitutes one production lot (job). Of course, the last two constraint sets are non-negativity constraints on the completion times and transfer batch sizes. Note that this LP formulation has $(2s-1)M-s+2$ constraints and $(M+1)s$ variables.

Model 2:

$$\text{Min} \quad \sum_{i=1}^{s} I_{i,M}$$

$$\text{s.to} \quad \sum_{h=1}^{i}(I_{h,m}-I_{h,m-1})+(p_m-p_{m-1})\sum_{h=1}^{i-1}q_h-p_{m-1}q_i \geq 0, \qquad i=1,\ldots,s; \quad m=2,\ldots,M$$

$$\sum_{i=1}^{s} q_i = Q$$

$$I_{i,m} \geq 0, \qquad i=1,\ldots,s; \quad m=1,\ldots,M$$

$$q_i \geq 0, \qquad i=1,\ldots,s.$$

where $I_{i,m}$ = idle time on machine m preceding processing of the ith transfer batch,

In this formulation, transfer batch start times are considered rather than their completion times and uses the idle times on the machines as decision variables. Note that there need be no idle time on the first machine and thus $I_{i,1}=0$ for $1 \leq i \leq s$. Furthermore, the first transfer batch does not have to be delayed on any machine, thus $I_{1,m}=(p_1+p_2+\ldots+p_{m-1})q_1$.

The first set of constraints ensures that the idle time of machine m immediately before it processes transfer batch i having size of q_i units is at least as large as the difference between the completion time of transfer batch $i-1$ on machine m and transfer batch i on machine $m-1$. Again, the second constraint ensures that the sum of all transfer batches constitutes a production lot. The last two constraints are non-negativity constraints on the idle times and transfer batch sizes. This formulation has $sM-s+1$ constraints and sM variables. Obviously, the size of this formulation is smaller than the LP model 1.

APPENDIX C

MIXED INTEGER PROGRAMMING MODEL PROPOSED BY BENLİ (1994) FOR THE PROBLEM $1|F|TB(V,c)|C_{\max}$

A general M-machine flow shop model which handles both consistent and variable transfer batches cases is proposed by Benli (1994). The formulation is basically a periodic review model with variable period lengths, which are the decision variables in the model. In this formulation, the number of transfers allowed on each machine is not restricted to be the same as in the previous models we have discussed so that it is a more flexible model for the single-job flow shop lot streaming problem. On each machine, there are s_m time points (i.e., the number of transfer batches) when the transfer batches can occur from machine m to machine $m+1$, where $h = \sum_{1 \le m \le M} s_m$ is the total number of transfers in the shop. The time points at which the transfer occurs is denoted by T_t, $t = 1,...,h$, where T_0 be the time at which the processing of the job starts on machine 1. Then the periods are defined by the time intervals $[T_{t-1}, T_t]$, $t = 1,...,h-1$. Furthermore, the following variables are defined:

$X_{m,t}$: Amount of production during the time interval $[T_{t-1}, T_t]$ (or, equivalently, period t) on machine m,

$Q_{m,t}$: Size of the transfer batch from machine m to machine $m+1$ at time T_t (i.e., end of period t).

In front of each machine, the items that has been transferred from the preceding machine are stored in an input buffer, and the processed items, if not transferred to the next machine are stored in an output buffer of the machine. In order to achieve this inventory balance, the following two additional inventory variables are defined as:

$I_{m,t}$: Amount of items in the input buffer of machine m at the end of period t,

$O_{m,t}$: Amount of items in the output buffer of machine m at the end of period t.

The earliest time period on machine m in which the production can take place is period m, and the latest time period in which the production can take place is $h-M+m$. Thus, the periods $t = m,...,h-M+m$ are the active periods for machine m. In order to indicate whether or not a transfer takes place from machine m at the

end of period t, a set of binary variables, one for each active period on every machine, is defined as

$Y_{m,t}$ = 1 if a transfer takes place from machine m at the end of period t, i.e.,
$Y_{m,t} = 1$ if $Q_{m,t} > 0$,
= 0, otherwise.

Then, the following mixed integer linear program which minimizes the time at which the last transfer batch completes processing on the last machine M, T_h as

$$\text{Min} \quad T_h$$

$$\text{s.to} \quad I_{m,t-1} + Q_{m-1,t-1} = I_{m,t} + X_{m,t}, \quad m = 1,\ldots,M, \quad t = m,\ldots,h-M+m$$

$$O_{m,t-1} + X_{m,t} = O_{m,t} + Q_{m,t}, \quad m = 1,\ldots,M, \quad t = m,\ldots,h-M+m$$

$$p_m X_{m,t} \le T_t - T_{t-1}, \quad m = 1,\ldots,M, \quad t = m,\ldots,h-M+m$$

$$\sum_{t=1}^{h} Y_{m,t} \le s_m, \quad m = 1,\ldots,M$$

$$Q_{m,t} \le \mu Y_{m,t}, \quad m = 1,\ldots,M, \quad t = m,\ldots,h-M+m$$

$$T_t \ge 0, \quad \forall t$$

$$I_{m,t},\ O_{m,t},\ X_{m,t},\ Q_{m,t} \ge 0, \quad \forall m,t$$

$$Y_{m,t} \in \{0,1\}, \quad \forall m,t$$

where $T_t = I_{m,m-1} = I_{m,h-M+m} = O_{m,m-1} = O_{m,h-M+m} = 0$, $Q_{0,0} = Q_{M,h} = Q$, p_m is the unit processing time on machine m, μ is a very large number or the capacity of the material handling equipment, and Q is the lot size of the job.

In this formulation, the first two sets of constraints are the inventory balance equations for the input and output buffers, respectively. The third set of constraints are the machine capacity constraints restricting the amount that can be produced in a period. In the fourth set of constraints, the total number of transfer batches from machine m is bounded by s_m. The fifth set of constraints ensure that transfers can take place only if they are indicated to do. Naturally, the last three sets of constraints are the non-negativity and integrality restrictions. Note that this formulation has $2M(2(h-M)-1)+h$ continuous variables, $M(h-M+1)$ binary variables and $M(4(h-M)+5)$ constraints.

APPENDIX D

SOLUTION PROCEDURE PROPOSED BY WILLIAMS AND TÜFEKÇİ (1992) FOR THE PROBLEM $1|F|TB(s=2,C,c)|C_{\max}$

Step 1 Find a machine index k such that

$$p_k > p_m \text{ for all } m = 1,\ldots,k-1$$
$$p_k \geq p_m \text{ for all } m = k+1,\ldots,M$$

if $d_k \leq 0$, or

$$p_k > p_m \text{ for all } m = k+1,\ldots,M$$
$$p_k \geq p_m \text{ for all } m = 1,\ldots,k-1$$

if $d_k > 0$. Index k is the initial feasible index.

Step 2 (a) If $d_k \leq 0$ then

(i) Find $z = \min_{k<m\leq M}\left\{\frac{P(k,m-1)}{P(k+1,m)}\right\} = \frac{P(k,r-1)}{P(k+1,r)}$

where r, $k < r \leq M$, is the largest index for which the equation above is true and $P(k,v) = \sum_{m=k}^{v} p_m$. Index r is a feasible index.

(ii) Calculate $d_r = P(1,r-1) - P(r+1,M)$.

(iii) If $d_r > 0$ then go to step 3,
otherwise set $k = r$ and repeat Step 2a.

Otherwise go to Step 2b.

(b) (i) Find $z = \max_{1\leq m<k}\left\{\frac{P(m,k-1)}{P(m+1,k)}\right\} = \frac{P(r,k-1)}{P(r+1,k)}$

where r, $1 \leq r < k$, is the smallest index for which the equation above is true. Index r is a feasible index.

(ii) Calculate d_r.

(iii) If $d_r \leq 0$ then go to step 3,
otherwise set $k = r$ and repeat Step 2b.

Step 3 Let $q_1 = (\frac{z}{1+z})Q$. The optimal makespan is $C_{\max} = d_k\, q_1 + Q\, P(k,M)$.

APPENDIX E

SOLUTION PROCEDURE PROPOSED BY JOHNSON (1954) FOR THE PROBLEM $N|F2\|C_{\max}$

Let (j) be a job which is sequenced in the jth position for a permutation schedule in a flow shop. Then the maximum completion time (makespan) for this schedule is given by

$$C_{\max} = C_{(N)} = \max_{1 \le u_1 \le u_2 \le \ldots \le u_{M-1}} \{\sum_{k=1}^{u_1} P_{(k),1} + \sum_{k=u_1}^{u_2} P_{(k),2} + \ldots + \sum_{k=u_{M-1}}^{N} P_{(k),M}\}.$$

From the expression above, it follows that, for the $N|F2\|C_{\max}$ problem, one gets

$$C_{\max} = C_{(N)} = \max_{1 \le u \le N} \{\sum_{k=1}^{u} P_{(k),1} + \sum_{k=u}^{N} P_{(k),2}\} = \max_{1 \le u \le N} \{\sum_{k=1}^{u} P_{(k),1} - \sum_{k=1}^{u-1} P_{(k),2}\} + \sum_{k=1}^{N} P_{(k),2}.$$

An algorithm for finding the optimal permutation schedule which minimizes the function above and, hence, solves the $N|F2\|C_{\max}$ problem was offered by Johnson (1954). It can be described as follows.

Johnson's Algorithm

Step 1 Decompose the job set J into two parts as:

$$J_A = \{j : P_{j,1} \le P_{j,2}\} \quad \text{and} \quad J_B = \{j : P_{j,1} > P_{j,2}\}.$$

Step 2 (i) Sort the job numbers according to the non-decreasing order with respect to $P_{j,1}$, and denote the resulting list as J_A^*;

(ii) Sort the job numbers according to the non-increasing order with respect to $P_{j,2}$, and denote the resulting list as J_A^*.

Step 3 The optimal sequence of jobs is $J^* = \{J_A^*, J_B^*\}$.

CURRICULUM VITAE

Personal

Date of birth	: January 13, 1963
Place of birth	: Erzincan, Turkey
Sex	: Male
Nationality	: Republic of Turkey

Education

M.S. Industrial Engineering Department, METU, Ankara, Turkey, 1988

Thesis Title : An Iterative Solution Procedure for Improving Feasibility in Hierarchical Production Planning
Supervisor : Assist. Prof. Dr. M. Sinan Kayalıgil

B.S. Industrial Engineering Department, METU, Ankara, Turkey, 1985

Positions

Teaching Assistant	Ind. Eng. Dept., METU. October 1985-February 1996 (on leave from Ph.D program and METU during January-March 1993 for military service).
Visiting Researcher	School of Industrial Engineering, Purdue University, Indiana, USA; Management Department, Ball State Univ., Indiana, USA. August 1993-June 1994 (on leave from Ph.D program and METU).
Editorial Assistant	Transactions on Operational Research (Journal of the Operational Research Society of Turkey). 1992-1993, 1995-present.

Publications

Refereed Journals

Çetinkaya, F.C. (1994), "Lot Streaming in a Two-stage Flow Shop with Set-up, Processing and Removal Times Separated", *Journal of Operational Research Society*, Vol. 45, No.12, pp. 1445-1455.

Çetinkaya, F.C., and M.S. Kayalıgil (1992), "Unit Sized Transfer Batch Scheduling with Setup Times", *Computers and Industrial Engineering*, Vol.22, No.2, pp. 177-183.

Kayalıgil, M.S., and F.C. Çetinkaya (1991), "Lot Streaming and Scheduling in a Two-stage Flow Shop with Transportation Times", *Industrial Engineering*

(Journal of the Chamber of Mechanical Engineers of Turkey), Vol.4, No.19, pp. 3-11 (in Turkish).

Proceedings

Çetinkaya, F.C. (1992), "Lot Scheduling with Transfer Batches in a Two-stage Flow Shop", *Proceedings of the 7th International Working Seminar on Production Economics*, Vol.2, pp. 129-143, Igls/Innsbruck, AUSTRIA.

Kayalıgil, M.M., and F.C. Çetinkaya (1991), "Kafile Bölünmeli ve Makinalar Arası Taşima Süreli Akış Tipi Atölye Çizelgelemesi (Flowshop Scheduling with Lot Streaming and Transportation Time between Machines)", *Proceedings of the 13th National OR/IE Conference*, pp. 241-245, Ankara, Turkey (in Turkish).

Köksalan, M.M., and F.C. Çetinkaya (1988), "Design and Implementation of a Solid Waste Collection System", *Proceedings of the 11th National OR Conference*, Vol.1, pp. 115-129, İstanbul, Turkey (in Turkish).

Working Papers

Çetinkaya, F.C., and J.N.D. Gupta (1995), "Optimal Lot Streaming in a Two-stage Hybrid Flowshop", *Technical Report No.95-01*, Industrial Engineering Department, METU, Ankara.

Çetinkaya, F.C., and J.N.D. Gupta (1994), "Flowshop Lot Streaming to Minimize Total Weighted Flow Time", *Research Memorandum No.94-24*, School of Industrial Engineering, Purdue University, Indiana, USA.

Çetinkaya, F.C., and M.S. Kayalıgil (1993), "Integrating Sublot-sizing and Scheduling Decisions in a Two-machine Flow Shop", *Technical Report No.93-13*, Industrial Engineering Department, METU, Ankara.

Çetinkaya, F.C. (1991), "Lot Scheduling with Transfer Batches where Setup, Processing and Removal Times are Separable and Sequence Independent", *Technical Report No.91-11*, Industrial Engineering Department, METU, Ankara.

Çetinkaya, F.C., and M.S. Kayalıgil (1991), "Unit Sized Transfer Batch Scheduling in a Two-stage Flow Shop with Sequence Independent Setup Times", *Technical Report No.91-03*, Industrial Engineering Department, METU, Ankara.

Kayalıgil, M.S., and F.C. Çetinkaya (1991), "Lot Streaming in a Two-stage Manufacturing System with Transportation Time Restriction", *Technical Report No.91-04*, Industrial Engineering Department, METU, Ankara (in Turkish).

Çetinkaya, F.C. (1987), "A Note on the Efficient Implementation of Johnson's Scheduling Algorithm", *Technical Report No.87-12*, Industrial Engineering Department, METU, Ankara.

Applied Research Project Reports

Arıkan, M.S., Çetinkaya, F.C., and B. Kaftanoğlu (1989), "A System Analysis and Computer Integrated Manufacturing Design for İLSAN Medicine Industry Corporation", *Final Report*, Machine Design and Manufacturing Research Institute, METU, Ankara.

Çetinkaya, F.C., Kırca, Ö., and M.M. Köksalan (1989), "A System Analysis for the Reorganization of BARMEK Holding Corporation", *Final Report*, Systems Sciences Research Institute, METU, Ankara.

Arıkan, M.S., Çetinkaya, F.C., Gökçay, G.F., Göksu, R., Kaftanoğlu, B., Köksalan, M.M., Oğuz, M., Soyupak, M., Sürücü, G., and Yahşi, S.O. (1987), "A Solid Waste Management System Design and Implementation for the Municipality of BURSA", *Final Report*, Machine Design and Manufacturing Research Institute, METU, Ankara.

Benli, Ö.S., Çetinkaya, F.C., Dağlı, C., Hacısalihoğlu, G., Saatçioğlu, Ö., and A.T. Şatır (1986), "Reorganization of HEMA Gear Industry Corporation", *Final Report*, Systems Sciences Research Institute, METU, Ankara.

Paper Presentations

Çetinkaya, F.C., and J.N.D. Gupta, "Lot Streaming to Minimize Total Flow Time", *16th National OR/IE Conference*, Bilkent University, Ankara, Turkey, July 1994; Industrial Engineering Department, METU, Ankara, July 1994.

Çetinkaya, F.C., and R. Uzsoy, "Lot Streaming in a 2-stage Hybrid Flowshop", *TIMS/ORSA Joint National Meeting*, Boston, USA, April 1994.

Çetinkaya, F.C., "Lot Scheduling with Transfer Batches in a Two-stage Flow Shop", *7th International Working Seminar on Production Economics*, Igls/Innsbruck, AUSTRIA, February 1992.

Kayalıgil, M.S., and F.C. Çetinkaya, "Flowshop Scheduling with Lot Streaming and Transportation Time between Machines", *13th National OR/IE Conference*, METU, Ankara, Turkey, June 1990 (in Turkish).

Köksalan, M.M., and F.C. Çetinkaya, "Design and Implementation of a Solid Waste Collection System", *11th National OR Conference*, Marmara University, İstanbul, Turkey, June 1987 (in Turkish).

Applied Research Projects

"System Analysis and Computer Integrated Manufacturing Design"

Project Coordinator	: Prof. Dr. Bilgin Kaftanoğlu (Mechanical Eng. Dept., METU)
Sponsored by	: İLSAN Medicine Industry Corporation, İstanbul, Turkey
Research Center	: Machine Design and Manufacturing Research Institute, METU January 1989-August 1989

"System Analysis for the Reorganization of BARMEK Holding Corporation"

Project Coordinator	: Prof. Dr. Ömer Kırca (Industrial Eng. Dept., METU)
Sponsored by	: BARMEK Holding Corporation, Ankara, Turkey
Research Center	: Systems Sciences Research Institute, METU September 1988-January 1989

"Solid Waste Management System Design and Implementation"

Project Coordinator	: Prof. Dr. Bilgin Kaftanoğlu (Mechanical Eng. Dept., METU)
Sponsored by	: Municipality of Bursa, Bursa, Turkey
Research Center	: Machine Design and Manufacturing Research Institute, METU February 1986-April 1987

"Reorganization of HEMA Gear Industry Corporation"

Project Coordinator	: Prof. Dr. Ömer Saatçioğlu (Industrial Eng. Dept., METU)
Sponsored by	: HEMA Gear Industry Corporation, Ankara, Turkey
Research Center	: Systems Sciences Research Institute, METU September 1985-March 1986

International Journals Served as a Referee

Computers and Industrial Engineering, European Journal of Operational Research, IIE Transactions, Naval Research Logistics, Operations Research Letters, Transactions on Operational Research

Course Taught

IE 405 Facilities Planning and Layout (for non-IE students), Ind. Eng. Dept., METU, Spring 1990

Courses Assisted (all at METU)

IE 102 Industrial Engineering Orientation
IE 262 Engineering Statistics
IE 320 Production Planning I
IE 332 Human Factors Engineering I
IE 351 Operations Research I
IE 407 Fundamentals of Operations Research I
IE 408 Fundamentals of Operations Research II
IE 419 Lot and Order Scheduling
IE 422 Seminars in Manufacturing Systems
IE 423 Production Planning II
IE 424 Project Evaluation
IE 428 Systems Design
IE 471 System Simulation
IE 485 Technology Management

Professional Memberships

Operational Research Society of Turkey
Chamber of Mechanical Engineers of Turkey

Fellowship and Awards

. *A Grant in the Scope of the Encouragement Program of the International Scientific Publications*, Scientific and Technical Research Council of Turkey (TÜBİTAK), May 1995.

. *NATO Science Fellowship (NATO-A2)*, TÜBİTAK, February 1994-June 1994.

. *A Grant in the Scope of the Encouragement Program of the International Scientific Publications*, METU, February 1993.