Optimal and heuristic solutions for the single and multiple batch flow shop lot streaming problems with equal sublots

by

Adar A. Kalir

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Industrial and Systems Engineering

Approved:

“Science is like the joke about the drunk who is looking under a lamppost for a key that he has lost on the other side of the street, because that’s where the light is. It has no other choice”

- Noam Chomsky (14 June 1993)

Optimal and heuristic solutions for the single and multiple batch flow shop lot streaming problems with equal sublots

by

Adar A. Kalir

Dr. Subhash C. Sarin, Chairman

Industrial and Systems Engineering

ABSTRACT

This research is concerned with the development of efficient solutions to various problems that arise in the flow-shop environments which utilize lot-streaming. Lot streaming is a commonly used process of splitting production lots into sublots and, then, of scheduling the sublots in an overlapping fashion on the machines, so as to expedite the progress of orders in production and to improve the overall performance of the production system.

The different lot-streaming problems that arise in various flow-shop environments have been divided into two categories, single-lot problems and multiple-lot problems. Further classification of the multiple-lot problems into the lot streaming sequencing problem (LSSP) and the flow-shop lot-streaming (FSLS) problem is made in this work. This classification is motivated by the occurrence of these problems in the industry. Several variants of these problems are addressed in this research. In agreement with numerous practical applications, we assume sublots of equal sizes. It turns out that this restriction paves the way to the relaxation of several typical limitations of current lot-streaming models, such as assumption of negligible transfer and setup times or consideration of only the makespan criterion. For the single-lot problem, a goal programming (GP) approach is utilized to solve the problem for a unified cost objective function comprising of the makespan, the mean flow time, the average work-in-process (WIP), and the setup and handling related costs. A very fast optimal solution algorithm is proposed for finding the optimal number of sublots (and, consequently, the sublot size) for this unified cost objective function in a general *m*-machine flow shop.

For the more complicated multiple-lot problem, a near-optimal heuristic for the solution of the LSSP is developed. This proposed heuristic procedure, referred to as the Bottleneck Minimal Idleness (BMI) heuristic, identifies and employs certain properties of the problem that are irregular in traditional flow-shop problems, particularly the fact that the sublot sizes eminating from the same lot type and their processing times (on the same machines) are identical. The BMI heuristic attempts to maximize the time buffer prior to the bottleneck machine, thereby minimizing potential bottleneck idleness, while also looking-ahead to sequence the lots with large remaining process time earlier in the schedule. A detailed experimental study is performed to show that the BMI heuristic outperforms the Fast Insertion Heuristic (the best known heuristic for flow-shop scheduling), when modified for Lot Streaming (FIHLS) and applied to the problem on hand.

For the FSLS problem, several algorithms are developed. For the two-machine FSLS problem with an identical sublot-size for all the lots, an optimal pseudo-polynomial solution algorithm is proposed. For all practical purposes (i.e., even for very large lot sizes), this algorithm is very fast. For the case in which the sublot-sizes are lot-based, optimal and heuristic procedures are developed. The heuristic procedure is developed to reduce the complexity of the optimal solution algorithm. It consists of a construction phase and an improvement phase. In the construction phase, it attempts to find a near-optimal sequence for the lots and then, in the improvement phase, given the sequence, it attempts to optimize the lot-based sublot-sizes of each of the lots. Extensions of the solution procedures are proposed for the general *m*-machine FSLS problem.

A comprehensive simulation study of a flow shop system under lot streaming is conducted to support the validity of the results and to demonstrate the effectiveness of the heuristic procedures. This study clearly indicates that, even in dynamic practical situations, the BMI rule, which is based on the proposed BMI heuristic, outperforms existing WIP rules, commonly used in industry, in scheduling a flow-shop that utilizes lot streaming. With respect to the primary performance measure – cycle time (or MFT) – the BMI rule demonstrates a clear improvement over other WIP rules. It is further shown that it also outperforms other WIP rules with respect to the output variability measure, another important measure in flow-shop systems. The effects of several other factors, namely system randomness, system loading, and bottleneck-related (location and number), in a flow-shop under lot streaming, are also reported.

ACKNOWLEDGEMENTS

This is a dream come true for which I first wish to thank my parents, without whom this would not have been possible. Their unconditional love and constant support over the years is something that I cannot thank them enough for. In the same breath, I would like to thank my wife, Rina, for her mental support, patience and understanding and, for her willingness to postpone her own personal development and career, to allow me to pursue my own.

My deepest appreciation goes to my advisor, the Chairman of my committee, Professor Subhash C. Sarin, for his valuable assistance, guidance, and encouragement in bringing this research work to a successful completion. I regard him as a personal friend and I highly commend him for his character and academic achievements.

I am particularly grateful to my dissertation committee member, the Assistant Department Head in the Department of Industrial and Systems Engineering at Virginia Tech, Professor Michael P. Deisenroth, for his good advice, support, and, I dare say, friendship, throughout my graduate studies.

I would like to sincerely thank my dissertation committee member, the Head of the Operations Research program in the Department of Industrial and Systems Engineering at Virginia Tech, Professor Hanif D. Sherali, for being a role model in demonstrating what excellence in teaching and research truly means. I highly commend him for his academic capabilities and accomplishments.

I am also thankful to the remaining members of my dissertation committee, Professor George Ioannou and Professor Philip Huang (of the Management Science program) for their guidance and their interest in my research. A special thanks is also due to Dr. Charles P. Koelling who gracefully agreed to serve on my committee at the last stages of the research.

I take this opportunity to also thank my M.Sc advisor, Dr. Yohanan Arzi, a true colleague and a dear friend. Thanks is also due to Prof. Daniel Sipper, Prof. Avraham Shtub, and Prof. Oded Maimon of the Department of Industrial Engineering at Tel-Aviv University for their support.

Last, but not least, I thank all my colleagues and friends that I have acquired throughout my years at Virginia Tech – John Tester, Mark and Sue Eaglesham, Andre and Karla Ramos, Alexandra and Armando Medina, Cole Smith, Dan Mullins, Vianni Perianayagam, and Michael Greco - to name a few. I would also like to thank the Faculty and the Staff in the Department of Industrial and Systems Engineering – Dr. William Sullivan, Dr. Janis Terpenny, Dr. John Shewchuk, and Lovedia Cole, again, just to name a few.

I am honored to have been a part of such a talented and dedicated staff during my years at Tech. I will cherish this period greatly for the rest of my life.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Background and Motivation

With the approach of the twenty first century, there has been a great deal of discussion among researchers, as well as practitioners, regarding the Factory of the Future (FOF). Many technologies needed for the FOF are already well developed. Others are under active improvement. Still others are quite hazy and need significant development to be considered viable contributors. For example, MRP-II won't schedule the factory of the future (Mather, 1986).

Factories present various problems, many of which have been addressed by researchers with a significant amount of success. To effectively deal with these problems, problems have commonly been divided into three hierarchical categories (Anthony, 1965):

- Strategic planning
- Tactical planning, and
- Operations control

Strategic planning deals with the process of selecting the objectives by which to measure the overall performance of the factory, and the optimal level of the resources required to attain these objectives. The tactical planning level addresses problems pertaining to the effective utilization of the resources in view of the desired objectives. Finally, the operations control level is concerned with the detailed operational and scheduling decisions. The research work described herein falls into the last category.

The importance of this work emerges from the root causes as to why MRP won't schedule the factory of the future, as indicated by Mather. Material Requirements Planning (MRP) and Manufacturing Resources Planning (MRP-II) were the first structured methods to be developed for Production Planning and Control (PPC). Even today, these methods are widely used in the industry. MRP-based methods provide a timely plan for the acquisition of raw materials and their processing, based on the Bill Of Materials (BOM), and the procurement and production Lead Times (LT's) of the end products. MRP-II-based methods further accommodate the limited capacity of the available resources, ensuring that the MRP timely plans would also be feasible. However, **both MRP and MRP-II systems do not allow overlapping of the operations on a batch for processing by sequential machines**. The production batch in these systems is an integral entity which cannot be further broken down. This represents a major limitation of MRP-based systems, from a practical standpoint.

The overlapping of operations for processing by the machines has also been the primary reason for the success of two other PPC methods during the 80's, which rapidly replaced MRP-based systems in the many industries. These two methods are the well-known Just-In-Time (JIT) and Optimized Production Technology (OPT). In contrast to MRP methods, these JIT and OPT methods strongly support the overlapping of operations in production. JIT calls for the elimination of unproductive setup times and for the production of very small (sub-)lot sizes, preferably of size one. OPT is less restrictive than JIT and calls for the splitting of lots into 'transfer batches' – the OPT term for sublots - before and after the bottleneck machine(s).

Both JIT and OPT, although differing in their analysis of production systems, have reached similar conclusions. They have concluded that significant improvements can be attained by breaking down production batches into smaller transfer batches. Somewhat surprisingly, the critical problems, pertaining to the splitting of the production batches into transfer batches and the sequencing (or streaming) of these transfer batches throughout the production system have remained open. Neither JIT nor OPT have addressed these problems explicitly (Jacobs, 1984). This research is motivated by the need to develop practical solutions for these open problems.

JIT and OPT have both, independently, articulated that better system performance would be attained by streaming small portions of work through the system. But they have both fallen short, as far as implementation goes, in certain practical situations. JIT does not address cases in which the setup times cannot be reduced to insignificant levels relative to the processing times. Furthermore, it does not provide a straightforward answer as to what a sufficiently small setup time for JIT to work as expected is. OPT does not provide the means for determining the optimal transfer batch sizes and/or the sequence in which these transfer batches should be processed. OPT merely utilizes transfer batches to save time on the bottleneck resource(s). The utilization of transfer batches, in OPT, is limited to the neighborhood of the bottleneck, and does not extend throughout the system. As we shall see in the literature survey of lot streaming, optimizing the sizes of transfer batches and streaming them throughout the system, as opposed to just near the bottleneck, can further enhance the gains. In addition to the above limitation, OPT also falls short in coping with systems in which several resources experience the same workload (i.e., parallel simultaneous bottlenecks.) OPT postulates that such cases are rare in reality and are not desirable. However, efforts of assembly line balancing, group technology, and other techniques do lead to flow-shop configurations with almost perfect balance and, therefore, such cases should not be overlooked.

To further motivate the need for this research, consider the following simple problem. A single batch of 100 items is to be produced using a three-machine flow line. The routing is {M_1, M_2, M_3} with unit processing times of {2, 1, 3} respectively. Assume that, at time zero, all the machines are available, and that, setup and transfer times are negligible. In traditional batch production, the entire batch is processed before it is transferred from one machine to the next. This results in a makespan of 100(2+1+3)=600 time units. Now suppose that there are no limitations on the transfer of products between the machines. In that case, it would be more efficient to transfer the first item, upon completion on the first machine, to the next machine, instead of letting it await the completion of the remainder of the batch on that machine. By transferring the parts one by one, as soon as the processing is completed on them, the makespan can be reduced to: 1(2+1+3)+99(3)= 303 time units, a reduction of almost 50%!

This result demonstrates the potential benefits that can be gained from the utilization of lot-streaming. However, this result relies heavily on the assumption that the transfer time is negligible. Otherwise, the makespan may increase significantly. For example, let *TT* be the transfer time of any transfer batch size from one machine to the next. Assume that the transfer time is independent of the transfer batch size and that the machine is not operational during transfers (e.g., when the operator of the machine is also used to transfer the items.) Then, the makespan under no splitting becomes:

$$M = 600 + 2 \cdot (TT)$$

whereas the makespan under lot-splitting becomes:

$$M = 303 + 200 \cdot (TT)$$

Clearly, for $TT \geq 1.5$ time units, it becomes undesirable to split the batch into unit-sized transfer batches. So, suppose $TT = 2$. Does this imply that splitting is undesirable? The answer is negative. It may still be desirable to split the batch but, preferably, to larger transfer batches, in order to balance the additional transfer time involved. For example, equal transfer batches, of 10 items each, result in the following makespan:

$$M = 1 \cdot \left[10(2+1+3)\right] + 9 \cdot \left[10(3)\right] + 20 \cdot 2 = 370$$

This makespan is still significantly better than the one attained under no splitting. As we shall demonstrate in the literature review of single batch models, by letting the transfer batches be of unequal sizes, the makespan may be improved even further. This is due to the following two reasons. First, and more obvious, the transfer time can be reduced if we allow unequal transfer batches. In the above example, there is no need to transfer sub-batches of 10 items from the second machine to the third every time that these sub-batches are completed on the second machine. These sub-batches need only be transferred after the completion of every three sub-batches since, only by that time, the third machine completes the processing of the previous sub-batch. The second reason for which unequal transfer batches may be more desirable is that, by doing so, the machines can be more efficiently utilized.

As evident by the example, the problem of the determination of optimal transfer batch sizes is a problem worthy of study. Several versions of this problem are addressed in this research. Although JIT and OPT have not provided specific answers to this problem, they have given rise to many essential problems, in the area of lot streaming, that have not been addressed before and have finally started drawing the deserved attention.

Certainly, lot streaming as a research area is not new. Its first appearance in the literature dates back more than thirty years ago, when Reiter (1966) first coined the term. Although his work, at the time, was very limited, because of the limited computer capabilities and the absence of PPC tools, it was still an identification of a possible avenue to improving efficiency in production systems. As implied in the discussion thus far, lot streaming is the process of splitting production lots into sublots, and then scheduling these sublots in an overlapping fashion on the machines (or more generally, the resources), in order to accelerate the progress of orders in production (Baker and Jia, 1993), and to improve the overall performance of the production system.

Research in the area of lot streaming was not conducted in the 70's. There were minor exceptions in this regard that will be mentioned in the literature survey. It was not until the late 80's and the early 90's, after extensive research into JIT and OPT had been carried out, that the area of lot streaming was re-discovered. Some progress has definitely been made during the past few years. Analytical and simulation-based works have contributed to a better understanding of the impact of lot streaming in various manufacturing environments. Truscott (1986) mentions several potential benefits of lot streaming:

- Reduction of production lead times (thus, better due-date performance)
- Reduction of WIP inventory, and associated WIP costs
- Reduction of interim storage and space requirements ; and
- Reduction of material handling system (MHS) capacity requirements

Simulation studies and industry-based reports, published in recent years, have confirmed that the above benefits can indeed be achieved via lot streaming in various batch production environments. Following are some examples:

- Job shops (Dauzere and Lasserre, 1997 ; Smunt et. al., 1996)
- Flow shops and serial production systems (Wu and Egbelu, 1994)
- Flexible assembly systems (FAS) (Sohlenius et. al., 1989)
- Group Technology (GT) cells (Vembu and Srinivasan, 1995)
- Cellular Manufacturing Systems (CMS) (Logendran and Ramakrishna, 1995)

Still, as far as research is concerned, many of the lot streaming problems remain poorly understood (Vickson, 1995). The next Section describes two such problems. These two problems are the focus of this research.

1.2 Statement of the Problem

The problem that is addressed in this research can be stated as follows. Given a batch, or a set of batches, to be processed in a flow shop, determine sublots of equal size and the associated sequence in which to process the sublots, so as to minimize makespan and other performance criteria.

This problem can be divided into two cases. One pertains to the determination of the sequence of the sublots, given that the sublot size had been pre-determined. The second pertains to the determination of the sequence of the sublots as well as their size. The former case will be referred to as the Lot Streaming Sequencing Problem (LSSP) while the latter will be referred to as the Flow Shop Lot Streaming (FSLS) problem. In both, the sublots may or may not intermingle while streamed throughout the system. These two different problems correspond to two different practical applications, which will be described in more detail in Section 1.4. The following characteristics apply to both cases:

- There are N lots to be processed on an m-machine flow shop.
- The lots can be split into sublots (of unknown size) during production.
- The sequence in which a machine processes the lots (or sublots) is unknown (but all the lots follow the same machine ordering.)
- The machines require setup before the start of sublots' processing.
- The lots are of different sizes.
- The sublots of each lot are of equal size.

The above characteristics constitute an extremely difficult-to-solve sequencing problem, with the additional complication resulting from the unknown sublot sizes in the FSLS problem. As will be discussed in the Literature Review (Chapter 2), the traditional m-machine flow shop sequencing problem (i.e., without lot streaming) is known to be NP-hard. Thus, under lot streaming, if the sublots are considered independently, a large-scale complicated flow shop sequencing problem results, which is notoriously NP-hard. The assumptions that are made in the models developed in this research comply with the traditional assumptions of regular flow-shop sequencing problems. These assumptions are as follows:

- The machines are available for continuous processing.
- Preemption is not allowed, i.e., once the processing of a sublot has begun it cannot be stopped.
- The setup times and the unit processing times are deterministic and known in advance.
- The lots considered for sequencing are available at the beginning of the planning period (this assumption is relaxed in Chapter 5.)
- The machines cannot process more than one sublot at any time.

This research makes a first attempt to deal with the general problem of sequencing multiple lots in a multiple-machine flow-shop under lot-streaming with non-negligible setup times and unknown sublot sizes. In accordance with the two applications, consideration is restricted to the case of equal sublots. Justifications for this restriction are provided in the literature review. In some cases, as will be discussed later, this restriction makes the problem even more complicated.

To further justify the need for the proposed research, it is noted that the single most important problem facing production managers, on a daily basis, is how to effectively carry out daily production. Daily production is typically derived from the Master Production Schedule (MPS) that specifies lot sizes and due dates for a time period of several days work, normally a month. The MPS does not specify the sequence in which the lots should be produced and/or the transfer batches that should be utilized. These important decisions, which may significantly impact the performance of the entire production system and, in fact, of the entire facility, are unfortunately made, on a daily basis, by production managers who do not possess the proper tools for making the best decisions.

As explained in the previous Section, JIT and OPT have not provided sufficient and broad means to cope with these decisions. JIT suggests solutions only for particular cases in which the transfer and the setup times are negligible. OPT suggests solutions by focusing on the bottleneck, but does not explain explicitly how to determine the transfer batch sizes and their sequence.

This research is geared towards developing and providing effective, and yet practical, solutions for the flow-shop lot-streaming problems, to enable production managers to make better decisions when faced with these types of problems. Although the same problems arise in almost every batch production system, this research is particularly concerned with general flow-shop systems. Flow-shop systems are commonly used in the industry for the production of a single product or a family of similar products that are in high demand. In a flow shop system, machines are organized in serial and the lots flow from one machine to the next in the same order. Production lines, assembly lines, and flow lines all fall into the category of flow-shop systems.

1.3 Research Objectives

The primary objective of this research is to study the effects of lot streaming in single and multiple batch flow-shop systems and to provide optimal and heuristic solutions for a variety of lot-streaming problems that arise in these systems. From a practical standpoint, this research is geared toward developing and providing effective solutions for these problems to enable production managers to make better decisions when faced with them. In view of the above, the following objectives are pursued:

- To analyze and evaluate the extent of the potential benefits of lot streaming with respect to various objective functions
- To study the effects of the problem parameters on the potential benefits of lot streaming with respect to various objective functions
- To solve the single-batch lot-streaming problem under various scenarios of non-negligible transfer and setup times, and for various objective functions
- To develop an efficient optimum seeking algorithm for the single-batch lot-streaming problem with respect to a unified objective function
- To effectively extend the results developed for the single-batch lot-streaming problem to the multiple-batch lot-streaming problem
- To suggest efficient heuristics for the solution of the multiple-batch lot-streaming problem with respect to the makespan criterion
- To study the effects of lot streaming in a dynamic flow shop with respect to time-invariant objective functions (e.g., mean flow time, output variability, etc.)

1.4 Contributions of Research

This research makes a first attempt to deal with the general problem of sequencing multiple lots in a multiple-machine flow shop that utilizes lot-streaming, with the consideration of non-negligible sublot-attached setup times. Meaningful results and useful insights are provided for this problem. In particular, the proposed research makes the following contributions to the literature for the single-lot multiple-machine flow-shop problem:

- Analysis of the extent of the potential benefits via the use of lot streaming with respect to the three commonly used performance measures. These measures are:
    1. Makespan (i.e., the total completion time of all the lots.)
    2. Mean Flow Time (MFT).
    3. Average WIP level.
- Closed-form formulae for the optimal (equal) sublot size with respect to both operational and economical objective functions, under various forms of non-negligible transfer times.
- An optimal solution algorithm of complexity $O(m)$ computations and $O(m^2)$ comparisons for the objective of minimizing the makespan, for the case of non-negligible sublot-attached setup times, and its modifications for implementation for the minimization of production cost as well.
- An optimal solution algorithm to obtain the optimal sublots, and a quick approximation scheme, for the most general case of an objective function that considers the weighted sum of the makespan, the mean flow time, the average work-in-process, the setup, and the material handling associated costs.

For the multiple-lot multiple-machine flow-shop, the contributions of this research are as follows:

- Analysis of the extent of potential benefits via the use of lot streaming in a multiple-lot, multiple-machine flow shop with respect to the same three measures listed above.
- A near-optimal heuristic procedure for the lot streaming sequencing problem (LSSP) under both cases of intermingling and no sublot intermingling.

- A fairly fast optimal solution algorithm for the two-machine flow shop lot streaming (FSLS) problem with identical sublot size.
- An optimal solution algorithm for the two-machine FSLS problem with lot-specific sublot sizes, and a heuristic that offers a faster solution for practical large scale problems in which the optimal solution algorithm may require too much time due to its complexity.
- Extensions of the optimal and heuristic solution procedures developed for two-machine FSLS problems, that are expected to produce near-optimal solutions for general *m*-machine FSLS problems.
- A study of the effects of the problem parameters on the quality of the solution and the performance of the proposed procedures. These parameters are:
    1. The number of machines in the flow shop
    2. The number of lots considered for sequencing
    3. The percentage of setup times
    4. The sublot size
- A comprehensive simulation study that demonstrates the effectiveness of the proposed procedures in practical dynamic situations with respect to two primary measures:
    1. Cycle time (or MFT)
    2. Output variability
- A study of the effects of the following factors in a dynamic lot streaming environment:
    1. System randomness
    2. System loading
    3. Bottleneck-related factors (location and multiple bottlenecks)

1.5 Lot Streaming Applications

In this Section, two applications are described. These applications correspond to the two lot streaming problems discussed earlier, namely the Lot-Streaming Sequencing Problem (LSSP) and the Flow-Shop Lot-Streaming (FSLS) problem.

The first application is the Surface Mount Technology (SMT) production line. SMT production lines are widely used today for manufacturing Printed Circuit Boards (PCBs). A typical SMT production line consists of the following machines in serial:

- Screening machine
- Pick and Place (P&P) machines, the number may vary from one up to as many as eight in serial
- Visual (or automated) inspection station
- Reflow oven

Unprocessed PCBs of a given lot are loaded onto a buffer at the beginning of the line. They are then transferred one by one via a conveyor from one machine to the next along the SMT line. The conveyor also serves as a small intermediate buffer between the machines. **Fig. 1.1** depicts a schematic SMT line. Each P&P machine along the line performs, basically, the same task. Components are sequentially picked by the header from the back of the machine, where component-feeders have been pre-installed, and are then placed in the correct locations on the PCB.

Assembly Line Balancing (ALB) of the P&P machines, which are almost always the bottleneck in the process, is clearly essential to minimize the idle time and the cycle time of the line. However, the problem of balancing SMT lines is a complicated one, and even if solved optimally, additional technological constraints may prevent the feasibility of a perfect balance. Moreover, the same problem arises for different types of PCBs and thus, the imbalance varies from one PCB to the other.

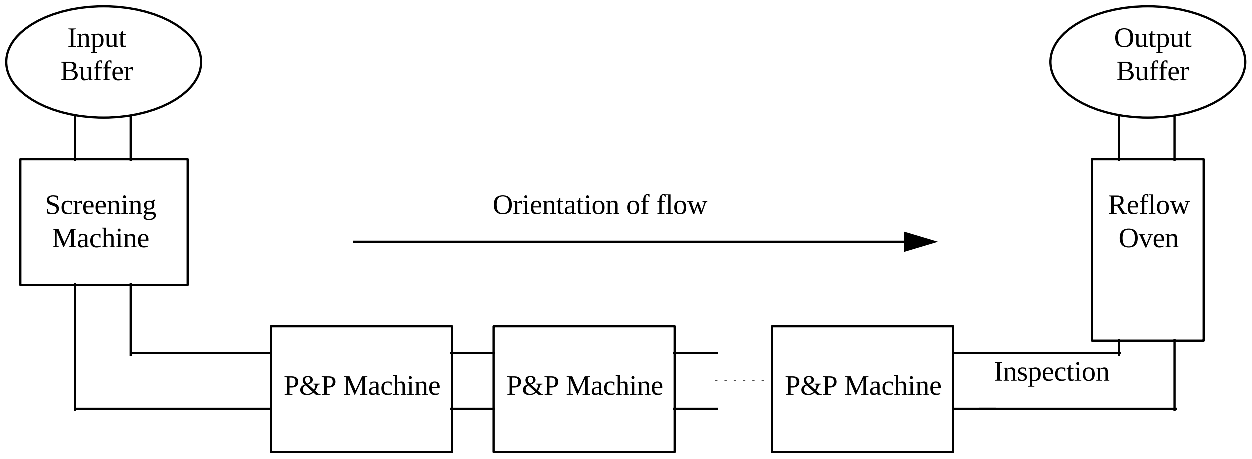


Fig. 1.1. Schematic SMT production line.

The problem characteristics are as follows:

- The lots are streamed one-by-one (i.e., unit-sized sublots are utilized.)
- The lots are not intermingled.
- The transfer time is negligible compared to the processing time.
- The setup of the P&P machines (i.e., the loading of the component feeders) is only required for the entire lot, and can be done in parallel to the processing of the current lot. Therefore, it can be considered negligible as well, as long as the total processing time of the current lot is larger than the setup time of the next lot.

The SMT sequencing problem can succinctly be stated as follows: Determine a sequence for the **lots** (no intermingling), given that they are streamed one by one, in order to minimize the makespan (i.e., the total completion time of all the lots listed for the next shift.) Hence, it is an example for the Lot Streaming Sequencing Problem (LSSP).

The second application considers a special case of the production of wafers in Semi-Conductor (SC) manufacturing. In general, the production process of wafers is performed via a re-entrant flow shop system, in which the sublots re-enter the same machines several times throughout their production. As a first attempt at this problem, we only consider the special case of semi-conductor processes involving uni-directional flow patterns (i.e., no re-entrance.) Under this assumption, the flow shop experiences mixed sublots (i.e., lots are allowed to intermingle) of equal sizes. The sublots are transferred between the machines using a device called ‘cassette’. Each cassette contains a fixed amount of wafers. The wafers within a single cassette are processed and moved together throughout the shop. Some machines process the wafers in the cassette one-by-one (still, the transfer is only done after all the wafers within the cassette have been processed) while other process them altogether. In both cases, setup of the machine is required before the start of processing. Thus, the characteristics of this problem can be summarized as follows:

- The lots are streamed in equal-sized sublots, using cassettes.
- Sublots of different lot types can intermingle.
- Setup is required prior to the processing of every sublot.
- The transfer time is negligible compared to the processing time.

The key questions constituting this problem are as follows: (a) How should the **sublots** be sequenced, and (b) What should the **sublot size** be, in order to optimize a given objective function. Due to the significantly longer processing times of the lots in SC manufacturing, criteria such as minimizing the average WIP and the mean flow time are considered at least as important as the makespan minimization criterion.

1.6 Organization of Research

This work contains six chapters. Introduction to the problem, the research objective, and two possible applications of the work have been presented in Chapter 1. In Chapter 2, a review of the literature on lot streaming and other related areas is provided. Chapter 3 presents detailed analysis and results, obtained for the single-batch lot streaming problem. Optimal and heuristic solutions for the multiple-batch lot streaming problem are developed and presented in Chapter 4. Both Chapter 3 and Chapter 4 address the static version of the problem. In contrast, Chapter 5 presents simulation-based analysis of the dynamic lot streaming problem. A summary of the research and concluding remarks are made in Chapter 6.

Chapter 2: Literature Survey

The objective of this chapter is to provide a review of the published literature on lot streaming as well as other strongly related areas. These areas include production planning and control (PPC) methodologies which support the splitting of lots (JIT, CONWIP, and OPT) and flow-shop sequencing and scheduling. The review of the literature begins with a historical overview of the progress made in the area of lot streaming during the past thirty years. This is followed by a review of the PPC methodologies that support the possibility of lot-splitting. Emphasis is given to the concept in which lot streaming is performed via each of the methods. Then, a review of the existing heuristics for the regular flow-shop sequencing problem is presented and, lastly, an up-to-date detailed coverage of the research on lot streaming is provided. The literature review of the latter is summarized in two Sections. The first Section deals with single batch models while the second deals with the more general case of multiple batch models.

2.1 Historical Review

The term lot streaming first appeared in the literature more than thirty years ago, when Reiter (1966) coined the term "lot streaming" in his article "*A System for Managing Job-Shop Production*". Although his work at that time was very limited, a breakthrough was achieved regarding the identification of a possible avenue to improve the efficiency of production systems. Somehow, possibly due to the complication involved in it, developers of the MRP system, in the early 70's, had chosen to ignore the possibility of splitting lots. In MRP-based systems for production planning and control, lots are considered the smallest integral entities to flow throughout the production system. Generally speaking, research in the field of lot streaming was neglected in the 70's. However, there were some exceptions. Szendrovits (1975) analyzed a single batch multiple machine makespan problem, with equal sublots of known size, and under the assumption of continuous processing, i.e., no idleness between any two consecutive sublots on any machine. Goyal (1976) continued the work of Szendrovits, and developed a scheme for obtaining the optimal sublot sizes (which were assumed to be known in Szendrovits' model.)

As the 80's brought two new and revolutionary approaches, namely JIT and OPT, which strongly supported the streaming of smaller portions of lots throughout the production system, research into lot streaming was re-discovered. Insights gained into JIT and OPT have enhanced the need of quantitative methods to determine optimal sublots (or transfer batches, as they are referred to in OPT.) Vollmann (1986) even categorized OPT as simply being an enhancement of MRP, as it provides the possibility of splitting lots, although, in addition to transfer batches, OPT supports other features pertaining to the scheduling and the utilization of bottlenecks, that MRP-II packages do not. According to a broad interpretation of the OPT principles, these transfer batches need not necessarily be of equal size, but no further details regarding their determination are available in the public domain (Trietsch, 1989). Judging by the output of the OPT software, it seems that equal sizes of sublots are used in OPT (Jacobs, 1984).

Meanwhile, after gaining an enormous deal of insights into JIT, a more generalized and powerful concept - CONWIP (CONstant Work-In-Process) - had emerged in 1990, as an alternative to JIT. According to CONWIP, it may be better to have some level of WIP throughout the production system, as long as this level is strictly preserved.

In the late 80's and the early 90's, after extensive research had been carried out to cover almost all aspects of JIT and OPT, it became apparent that the concept of lot streaming, a major issue related to both of them, was neglected. From this point on, research has drawn its attention directly to the question of lot streaming in various production systems. The early analytical works of Truscott (1986), Trietsch (1987), and Baker (1988) were mainly concerned with the impact of a single lot, split into sublots, on system performance, where the production system under examination was typically limited to a two-machine flow shop. Subsequently, two research directions have evolved. The first pertains to the analysis of more complex systems, such as systems with multiple batches or job shop systems (El-Najdawi (1994) ; Doutriaux and Sarin (1996)). The second pertains to the generation of insights using simulation-based studies (Wagner and Ragats (1994) ; Hancock (1991)). These works have shed some light on our understanding of the impact of lot streaming in various manufacturing environments.

2.2 Existing PPC Methodologies

In this Section, three production planning and control (PPC) methodologies are briefly described. These are the well-known just-in-time (JIT), constant work-in-process (CONWIP), and optimized production technology (OPT). For each, the objectives, principles, and production control methods are discussed. Emphasis is given to the concepts of lot-streaming in these methodologies.

2.2.1 Just-in-time (JIT) and Kanban

The JIT philosophy originated in the plants owned by Toyota, a giant Japanese automobile manufacturer. JIT is a straightforward philosophy and its logic is easy to follow. Nevertheless, it is not straightforward that it guarantees achieving the ultimate goal, which is profit maximization. Unlike the theory of constraints (TOC), which will be discussed next, JIT is not a direct derivative of the above goal.

JIT evolved in Toyota due to the need to eliminate waste. Toyota defined waste as "anything other than the minimum amount of equipment, materials, parts, and working time absolutely essential to production" (Hay, 1988). Thus, JIT is concerned with the entire production system and not merely with material scrap, rework, or line fallout. Via JIT, waste is anything that is not necessary for the manufacturing process or is in excess. For example, labor hours spent reworking products because of poor quality as well as buffer inventories required to store defective parts awaiting repair are considered waste (Hernandez, 1989). By absolute minimum resources, it is also meant that one supplier is preferred as long as that supplier has enough capacity, that there should be no safety stock held in storage, and that, there should be no excess lead times for raw materials, and no work that does not add value to the product.

There are three main objectives for JIT (Suzaki, 1987). These objectives are homogeneous in the sense that they can be applied to a diversity of organizations within industries that differ greatly from one another (Cheng and Podolsky, 1996). These three objectives are :

- To increase the organization's competitiveness, over the long run, by systematically optimizing the manufacturing processes.
- To increase the degree of efficiency, within the production process, by achieving greater levels of productivity while minimizing the associated costs of production.
- To reduce production costs by reducing the level of wasted materials, time, and effort involved in the production process.

To accomplish its main objectives, JIT calls for the following conditions to be met:

- Elimination of unproductive processing time, mainly setup time.
- Elimination of process variability, i.e., (almost) deterministic processing times and reliable resources (via preventive maintenance and total production maintenance.)
- Elimination of non-value added work, which leads to simplified manufacturing processes.
- Elimination of material waste via perfect quality and quality control
- Elimination of stock and inventories by using small procurement lead times through constant reliable vendors.

Thus, any necessary implementation of JIT should first address the above issues to determine whether or not they are achievable. The JIT philosophy postulates that if the above conditions are satisfied, customer demands can be met within a very short time frame, almost just-in-time, and with great certainty. This, in turn, ensures customer satisfaction, and is achieved with minimal production costs (minimal WIP associated costs, minimal production costs, and minimal setup costs.)

The PPC method which supports the JIT philosophy is known as "Kanban." The word Kanban means "signal" in English. Kanban, similar to MRP-based systems, is basically used to establish the scheduling of operations, the quantity of the product to be produced, and the direction of production flow (Cheng and Podolsky, 1996). In Kanban, the transfer unit between each two successive machines is usually referred to as a container. Each container is designated for one specific part type. Production on a machine does not start unless the following three conditions are satisfied:

- The machine is ready to work on the next part.
- There is an available (full) container of parts, to be worked on, from the preceding machine.
- There is an available (empty) container, for completed parts, to be transferred to the following machine upon completion.

To ensure that work is not initiated unless the above conditions are met, card tags which specify the type and amount of product(s) that the next process should withdraw from the preceding process, are attached to the container(s) (Monden, 1983.) Since the last condition dictates that production cannot start if there is no empty container, this system is referred to as a 'pull' system. Production on the downstream machines authorizes the production on the upstream machines, by 'pulling' additional parts to work on. Production on the last machine starts only when a customer order has been placed. This concept allows to produce only the quantity that is required, and thus helps eliminate unnecessary inventory which would otherwise be stockpiled.

Obviously, the number of containers makes a physical upper bound on the WIP level in the system. The WIP level affects the throughput time and the total production cost and therefore, the determination of this number is of great importance. Several methods have been developed to determine the appropriate number of Kanbans to be used in the system. The interested reader is referred to Rees et al. (1987) and Fukukawa and Hong (1993) for examples. We now turn our attention to the Kanban approach to addressing three important factors of production planning and control systems. These are:

- The determination of lot sizes.
- The effect of setup time.
- The use of overlapping operations throughout the system.

Kanban production calls for small lot sizes. This requirement becomes especially important in the case of mixed production (i.e., multiple batches.) By utilizing small lot sizes, resources can be quickly changed over to meet other goals of production. However, while small lot sizes are desired in JIT, they should not be so small that the setup cost is spread over very few items, resulting in an increase of the cost per unit (Cheng and Podolsky, 1996). In addition to the requirement of small lot sizes, forecasting is taken into consideration to smooth out the production of lots. For example, if a large demand is anticipated after two successive periods in which a small demand is anticipated, then lot sizes are determined in a manner that would enable excess production during the first of these two periods to reduce the variability in lot sizes. Typically, this is referred to as leveled scheduling or production smoothing. The sequence in which the multiple batches are worked on at each machine is typically determined by their due-date, i.e., in the order of their urgency. The closer the due-date, the higher is the priority.

As mentioned above, setup times are a major issue in JIT. Setups are costly, time-consuming, and wasteful and therefore setup reduction is needed to pave the way for every other element of JIT, from small lot sizes, leveled scheduling (or production smoothing), to overlapping operations, to pull systems, and even to quality (Hay, 1988). The common method that has been used by Japanese firms to deal with setup reduction is the makeup of setup reduction teams, which analyze setup activities (external and internal) and improve the setup process. One way is to try and do as much of the setup as possible while the machine is running (Fisher, 1995). However, as setup can never be completely eliminated, a question that arises pertains to what a small setup time is. Intuitively, it can be argued that the point in which the ratio of setup time to processing time becomes insignificant reflects the JIT requirement of elimination of setup. Sometimes, more specific thumb rules are provided to interpret the JIT requirement, such as the one provided in Fisher (1988): “Establish rapid setups (less than 10 minutes) for most machines and lines”.

Complementary to the setup issue in JIT is the issue of overlapping operations. JIT calls for the facility to be laid out by product rather than by function. The equipment is semi-dedicated to a family of products and it is arranged in the order of the operations performed on that family of products. Then, JIT asks for a one-at-a-time flow from one machine to the next. This mechanism creates a flow whereby each operation on the succeeding machine starts as soon as it is completed on the current machine. In effect, the batch becomes a single item (Hay, 1988).

Several shortcomings are cited in the literature with regard to JIT. Most of these shortcomings are not really concerned with the JIT philosophy, rather they are really concerned with the cultural differences and the resistance to change, which can become severe barriers to a successful implementation of JIT in western countries (Klein, 1989). However, there are also a few drawbacks of the philosophy of JIT itself. One argument addresses the issue of flexibility of

JIT-based systems to deal with increases in demand over a short period of time (Hall, 1989). Although the JIT system is intended to respond quickly to such changes, it would not be able to compete with its rival competitors which do keep a certain amount of safety stock. It has also been speculated that JIT is an industry specific approach that is especially efficient for repetitive manufacturing but is rather limited for other applications. Fisher (1995) states that, although some principles of JIT can be applied in job-shop systems, its application in such systems is limited since it is the repetitive activities that cause most waste.

In the proposed work, we are particularly interested in two aspects of JIT that have been discussed above. One is the setup and the other is the overlapping of operations. With respect to these two aspects, we raise the following issues which, among others, motivate this research:

- JIT does not handle cases where setup times cannot be reduced to a level at which they become insignificant relative to processing times. Furthermore, there is no straightforward answer in JIT to the question of what these sufficiently small setup times are.
- JIT calls for overlapping operations and, more specifically, for splitting of the lots to unit-sized sublots. But, is this the most efficient way to stream the lots throughout the production system? Maybe this type of transfer incurs a higher material handling cost than what is actually required and, therefore, implies 'waste'?
- Given that the lots should be transferred one-at-a-time, what should their sequence be? Is it better to complete a certain lot or, maybe, it is preferred to intermingle the sublots and advance the lots simultaneously?

2.2.2 Constant work-in-process (CONWIP)

The CONWIP concept was first introduced by Spearman et al. (1990) for a single production line. They have shown that in the cases of variable processing times and/or unreliable machines, the Kanban policy tends to build up WIP upstream of the bottleneck machine. However, by maintaining a higher (but still, constant) WIP level the system results in a higher bottleneck utilization and as a consequence, a higher throughput. As explained in the previous sub-section dealing with JIT and Kanban, the Kanban policy requires the elimination of process variability (Monden, 1983). CONWIP realizes that this requirement may not always be met and, therefore, relaxes it. There are several other factors which motivate the use of CONWIP over Kanban. Spearman and Zazanis (1988) argue that any of the following conditions is detrimental to the success of Kanban implementation:

- Large product mixture (which affects the repetitiveness of the production system.)
- Long setup times
- Process variability
- Unbalanced workload

Thus, it is not purely due to high variability that CONWIP is preferable over Kanban. If setup times cannot be drastically reduced, or if the production environment is not highly repetitive, or even if workload cannot be closely balanced among the resources, Kanban may perform poorly compared to expectations. In fact, the inventors of Kanban have been well aware of the impact of the above factors and, therefore, stated that the elimination of all of them is a prerequisite to a successful implementation. CONWIP addresses all of these issues at once by modifying a single principle of the Kanban policy. While Kanban calls for keeping the WIP level minimal, preferably “zero”, in CONWIP this principle is relaxed, and is replaced by the following: “keep the WIP level at a constant level”. As a result, in addition to the robustness of system performance to process variability demonstrated by Spearman et al. (1990), the Kanban’s necessary condition of negligible setup times is no longer necessary, since a certain level of WIP is permitted. This WIP ensures faster response to market demand, because some work has already been carried out and incoming orders need not necessarily be processed from the very beginning (unlike Kanban), thus allowing setup times to not be negligible. For the same reason, CONWIP also handles cases of large product mixture better than Kanban.

One question that immediately comes to mind is how should the constant WIP level be determined in CONWIP. As will be demonstrated later in our review of lot streaming methods, the requirement for negligible setup times leads to the optimality of unit-sized sublots, as Kanban suggests. However, if setup times are not negligible, larger sublot sizes should be considered, which lead to larger WIP, in agreement with CONWIP. Recently, Sarin and Greco (1997) have proposed a procedure to determine the desired WIP level when the sequence of the batches is known in advance. Obviously, there is a strong correlation between the optimal WIP level and the sequence of the batches. Thus, the question of sequencing arises as well. Both Greco (1996) and Herer and Masin (1997) address this question.

CONWIP, similar to Kanban, operates as a pull system, i.e., the start of a batch is triggered by the completion of another batch (the preceding batch in Kanban.) In order to keep the WIP at a constant level, CONWIP, as opposed to Kanban, utilizes a closed production system approach. This implies that a fixed number of containers is utilized throughout the production system at all times. Same as in Kanban, empty containers, transferred back from an upstream machine, authorize production on downstream machines. However, unlike Kanban, an empty container coming from the last machine to the beginning of the production system authorizes production as well.

Research into closed queuing networks as well as simulation studies of production systems have provided strong evidence to support the superiority of pull systems over push systems (see Spearman and Zazanis, 1988 ; Hopp and Spearman, 1991). Thus, CONWIP maintains the pull principle introduced in Kanban. A performance comparison of CONWIP (a pull system) versus MRP (a push system) using simulation in an actual plant environment is provided by Roderick et al. (1994). They show that, by utilizing CONWIP, the plant's WIP can be reduced while ensuring the same levels of throughput as in MRP.

CONWIP is as easy to implement as Kanban and, under special circumstances, results in better system performance (due date, makespan.) However, its weakness is in its increased WIP levels. To summarize, JIT and CONWIP have both been developed to cope with the dynamic changes in product demands in a growing 'produce-to-order' environment. While changes in product demand could easily destroy schedules built by MRP systems, JIT and CONWIP are insensitive to such changes since it is primarily the actual demand, and not the forecast, that triggers production. However, for JIT to be efficient, several conditions pertaining to setup times, process variability, and repetitiveness must be met first. In CONWIP, these requirements are relaxed by allowing a certain (but fixed) level of WIP. Nevertheless, with regard to shop-floor control, similar issues arise:

- CONWIP does not address the issue of breaking the lots (batches) to sublots throughout the production.
- The issue of the sequencing of the lots (or the sublots) remains unresolved in CONWIP.
- WIP is not considered an objective in CONWIP, as it is predetermined. Thus, the primary measure utilized by this method is the throughput. It may, however, be possible to vary the WIP level around its constant level, proposed by CONWIP procedures, to further improve system performance, with respect to the average WIP as well. This, however, results in an open queuing network instead of a closed one, as suggested in CONWIP.

2.2.3 The Theory Of Constraints (TOC) and Optimized Production Technology (OPT)

The methodology of Optimized Production Technology (OPT) is a part of a more general philosophy, known as the Theory Of Constraints (TOC). TOC has been widely used by many companies throughout the United States in the late 80's and is still used widely today. It was developed by Dr. Eliahu Goldratt during the early 80's. First, the ideas at the operations control level were introduced in the famous books "The Goal" (1984) and "The Race" (1986). Then, a more general philosophy was added in the form of what had become to be known as the "theory of constraints". The theory of constraints starts from the very roots of existence of most organizations - their ultimate goal. Every action taken by any part of the organization should be judged by its impact on the goal. This obviously implies that the system's goal must first be defined and that a set of measures are derived, which would enable to judge the impact of any local decision on this global goal.

Once the goal and a set of measures have been established, the natural question to be asked is: what is preventing the system from further improvements in performance versus its goal? The answer to this question lies at the heart of the theory of constraints. Practical systems have very few constraints and at the same time must have at least one constraint (Goldratt, 1990). Therefore, TOC proposes a systematic approach for continuous improvement, achieved by implementing the following five steps:

- Identify the system's constraints. These are the constraints which limit the entire system.
- Decide how to exploit the system's constraints.
- Subordinate everything else to the above decision.
- Elevate the system's constraints.
- If, in the previous step, a constraint has been broken, go back to step 1, i.e., repeat from the beginning.

It is a common belief that OPT has emerged from the dissatisfaction of industry with existing PPC methods, in particular MRP (Trietsch, 1987 ; Vollmann, 1986). MRP systems were found to be subject to what is well known as the "MRP syndrome". The MRP syndrome is an expression used to describe the phenomenon of continuously increasing WIP, which occurs in

MRP systems. This phenomenon is caused because MRP is a highly parameter-sensitive method. It is sensitive to changes in lead-times, both procurement and production lead-times, and it is also sensitive to product demand. For the given lead-time values and product demand, MRP sets up a timely production plan for raw materials and WIP to be processed. Consequently, when these values change (as expected in a dynamic production environment), raw materials and WIP keep on accumulating along the production system without being consumed. This, in turn, causes the production lead-times to further increase due to additional WIP, and the process goes on.

This syndrome of MRP systems has been (at least partially) resolved in OPT. In OPT, the production plan is derived based, almost solely, on bottleneck considerations (In that respect, MRP systems use much more data than what is really needed.) The production plan, via OPT, is obtained in the following manner: all the stations preceding the bottleneck station are backward scheduled, i.e., the production plan starts with the production on the bottleneck station and back-propagates through the upstream machines, so as to fully utilize the bottleneck station. All the stations following the bottleneck station are forward scheduled, i.e., the production plan starts with the production on the bottleneck station and propagates to the downstream machines. To summarize, OPT aims at developing a timely production plan which fully utilizes the bottleneck resource.

One clear advantage of OPT over MRP is that, it is considerably less sensitive to changes in the input parameters than MRP. OPT requires much less data than MRP. It results in the same production plan in many cases where MRP does not. Another advantage of OPT is that it provides a method to reduce WIP by smoothing the flow to and from the bottleneck station. This is done by using small transfer batches for the bottleneck station. In contrast, MRP systems do not allow the splitting of batches.

Goldratt (1990) postulates that the extensive body of literature regarding the determination of optimal lot sizes ignores one simple fact which changes the entire understanding of the problem. In lot sizing problems, optimal lot sizes are realized based on minimizing two major cost components, namely inventory carrying costs and setup costs. Seemingly, there exists a trade-off between the two. Large batch sizes are required to save setup time while small batch sizes are required to reduce carrying cost of inventory. But, does this really mean that the best we can do is to minimize the sum of the two? According to Goldratt, there is no real conflict between the two. We can actually minimize both of them separately. To do that, Goldratt differentiates between a process batch and a transfer batch. A process batch is defined as the number of items grouped together for the purpose of processing them one after the other while a transfer batch is defined as the number of items grouped together for the purpose of transferring them from one machine to the next, along the production system. Now, it is clear that we can simultaneously use large process batch sizes, to cut setup costs, and small transfer batches, to cut inventory carrying costs. As Goldratt summarizes it, "the entire problem that bothered so many people for more than 50 years, was merely due to the improper terminology. The solution is obvious. We should strive to maximize process batches on the bottleneck while, at the same time, minimize transfer batches everywhere".

To further exploit why OPT calls for the smoothing of production in the neighborhood of the bottleneck station via transfer batches, we now introduce the key principles of accounting that are embedded within the theory of constraints. As explained earlier, the ideas are derived from the organization's ultimate goal. Normally, the goal would be profit maximization. In order to achieve the goal, the theory of constraints suggests to focus on (only) three cost-related components:

- Throughput - the rate in which the system generates money through sales
- Inventory - money invested in purchasing, for future sales
- Operating Expense - money spent in turning Inventory into Throughput

TOC realizes that there is always a system constraint which determines the throughput as well as the minimal level of inventory and the minimal operating expenses required. When the system constraint happens to be one of the manufacturing resources, which is typically the case, then the constraint is referred to as a bottleneck resource. The fact that the constraint determines the throughput is obvious. The fact that it also determines inventory and operating expense levels is less obvious and requires additional explanation. OPT argues that the level of utilization of every non-bottleneck resource is determined by the bottleneck resource and not by its own potential. Therefore, time saved at a non-bottleneck resource is just a mirage.

Following these principles, it should be apparent that, in order to gain further improvements in the goal, the system constraint must be identified and dealt with first. Other non-bottleneck resources are not important (at least not immediately.) Viewing these principles, it also becomes clear why OPT is merely satisfied with the smoothing of production in the neighborhood of the bottleneck station. It assumes that time saved at a non-bottleneck resource is just a mirage. But, is this assumption really true? Only when the resources are not versatile because if they are, then an hour saved on a non-bottleneck resource can be used to increase the capacity of the bottleneck resource. It seems that OPT itself contradicts this assumption, of non-versatile resources, made by the TOC, because it recognizes the versatility of resources (human operators) in “The Goal”.

OPT suffers from several other deficiencies as well. First, from the discussion thus far, it is apparent that OPT is sensitive to the data pertaining to the bottleneck resource. In fact, inaccurate data about the bottleneck can cause OPT to develop very poor schedules. Secondly, the idea of splitting batches is limited to smoothing the production around the bottleneck resource and is not extended beyond it. Moreover, it is reported that transfer batches via the OPT software are of equal size (Jacobs, 1984), although better smoothing results can be achieved by using unequal transfer batches. Fox (1983) also uses an example of equal transfer batches to illustrate how OPT works. To conclude, OPT introduces the idea of splitting large process batches before and after the bottleneck station to small transfer batches, in order to fully utilize the bottleneck resource. However, several unresolved issues in OPT, which motivate this research, are as follows:

- OPT does not provide means neither for the determination of the sizes of the transfer batches nor for the determination of the sequence in which the transfer batches are to be processed.
- The concept of transfer batches is merely utilized to save time on the bottleneck resource and is not extended beyond it. However, as we shall see in the literature survey of lot streaming, optimizing the sizes of transfer batches and streaming them throughout the system can further enhance the gains.
- OPT ignores flexibility and versatility in making the assumption that only the bottleneck resource should be considered for time savings.
- OPT ignores systems in which several or all the resources experience similar workload. OPT postulates that such cases are rare in reality but efforts of assembly line balancing, group technology and other techniques all lead to shop floors with nearly perfect balance.

In a recent paper by Russell and Fry (1997), the impact of several factors on system performance in OPT, observed via simulation, has been examined. Their conclusion is as follows: “the impact of lot splitting has yet to be studied. We found that the splitting of process batches into smaller transfer batches nearly always improved shop performance criteria. Further, simulation results indicate that the shop inventory levels necessary to achieve comparable service levels is a function of the capacity balance between bottlenecks and non-bottlenecks, i.e. capacity slack. As capacity slack increases, thereby reducing capacity balance, shop inventory levels show a corresponding decrease.” The impact of these factors are further investigated in this research.

2.3 *Flow Shop Sequencing and Scheduling*

In this research, the issue of streaming multiple lots in a flow-shop production system is addressed. In doing so, some of the results build up on previous results, obtained in the area of classical flow-shop sequencing and scheduling. Thus, it is necessary to provide a brief review of the available literature in this area. The primary motivation is that it may be possible to extend existing results and algorithms, available for job scheduling in flow shops, to the case of lot streaming.

The terminology 'flow shop' is used to describe a production system in which lots follow the same routing on the machines. The static flow shop sequencing problem denotes the problem of determining the best sequence of the lots (known in advance) on each of the machines in the flow-shop. The schedule is the resulting timely plan of the sequence(s). Various objectives can be used to determine the best sequence but, the majority of the research considers the minimization of the makespan (i.e., the total completion time of the entire list of lots) as the primary objective. Other objectives that can be found in the literature of flow shops are flowtime related (e.g., minimal mean flowtime), due-date related (e.g., minimal maximum lateness), and cost related (e.g., minimal total production cost).

The earliest analytical results for flow shop sequencing are due to Johnson (1954). He has shown that, in a two-machine flow shop, an optimal sequence can be constructed as follows: Given the pair $\{a_i, b_i\}$, the processing time of each lot *i* on each machine *j* (*j*=1,2), the optimal sequence on both the machines is the one in which the following condition is satisfied for any two lots *i*,*k*:

$$\text{if: } \min\{a_i, b_k\} \le \min\{a_k, b_i\} \text{ then lot } i \text{ precedes lot } k \qquad (2.3.1\text{-}1)$$

The above condition is known as Johnson's rule. One way to implement the rule is to first arrange the lots with $a_i \le b_i$ (i.e., for which machine 2 is dominant) in increasing order of a_i and then to arrange the remaining lots in decreasing order of b_i. In other words, the lots are sequenced according to a shortest processing time (SPT) rule on M_1 and a longest processing time (LPT) rule on M_2.

An immediate result of the above solution is that the same sequence is utilized for both the first and the second machine, i.e., the sequence need not be modified. In general, when only the sequence on the first machine is considered, and an assumption is made that the same sequence among the jobs applies throughout the flow shop, that sequence is termed a “permutation sequence”. Although permutation sequences need not be optimal in general *m*-machine flow shops, it has become a tradition to assume identical processing sequence on the machines and to look for the best permutation sequence (Lawler et al, 1993). Conway et al (1967) have observed that, due to the inverse property of the makespan objective, there exists an optimal sequence which is identical on M_1 and M_2 as well as on M_{m-1} and M_m. Hence, for the three-machine flow shop, there must exist an optimal permutation sequence.

Unfortunately, Johnson’s rule cannot be generalized to yield optimal sequences for flow shops with more than two machines. Special problems of three machines in which one machine dominates the others can be reduced to equivalent two-machine problems. However, if there does not exist a dominant machine, the problem is strongly NP-hard (for a proof, see Pinedo, 1996.) Therefore, in flow shops which consist of more than two machines, the majority of research has mainly focused on developing efficient heuristics.

Page (1961) was the first to introduce the concept of a slope index in prioritizing lots. Palmer (1965) then adopted this idea and proposed the following slope index to be utilized for lot sequencing in a general *m*-machine flow shop:

$$\begin{aligned} s_i &= (m-1)\cdot p_{i,m} + (m-3)\cdot p_{i,m-1} + (m-5)\cdot p_{i,m-2} + \ldots \\ &\quad -(m-5)\cdot p_{i,3} - (m-3)\cdot p_{i,2} - (m-1)\cdot p_{i,1} \end{aligned} \tag{2.3.1-2}$$

Where s_i is the slope index of lot *i*. The sequence is constructed in the decreasing order of the slope indices of the lots. In the spirit of Johnson’s SPT(1)-LPT(2) rule, Palmer’s rule gives priority to lots having higher tendency to progress from short times to long times. However, for m=2, Palmer’s rule does not coincide with Johnson’s rule and therefore does not guarantee optimality even for the two-machine case.

Gupta (1971) investigated cases in which Johnson's rule is optimal for three-machine flow shops. He proposes the following slope index for the general *m*-machine flow shop based on his analysis:

$$s_i = \frac{e_i}{\min\limits_{1 \le k \le m-1} \left\{ p_{ik} + p_{i,k+1} \right\}} \tag{2.3.1-3}$$

where:

$$e_i = \begin{cases} 1 & \text{if } p_{i1} < p_{i,m} \\ -1 & \text{if } p_{i1} \ge p_{i,m} \end{cases}$$

Campbell et al (1970) proposed a simple heuristic extension of Johnson's rule to the *m*-machine flow shop problem. This extension is known in the literature as the CDS heuristic. The heuristic constructs at most (*m*-1) different sequences from which the best sequence is chosen. Each sequence corresponds to the application of Johnson's rule on a new two-machine problem that is derived from the original problem in the following manner:

The new processing time of lot *i* on the first machine is: $p'_{i1} = \sum_{j=1}^{k} p_{ij}$

The new processing time of lot *i* on the second machine is: $p'_{i2} = \sum_{j=1}^{k} p_{i,m-j+1}$

For each $k = 1,2,...,m-1$ a (possibly) different sequence is generated. The best makespan from the corresponding schedules is identified and the respective sequence is chosen. Notice that for m=2, the CDS heuristic reduces to Johnson's rule and is thus optimal.

Several other algorithms have been proposed over the years, see for example Dannenbring (1977) ; Turner and Booth (1987) and Osman and Potts (1989). The latter propose to use the random search technique of simulated annealing for the problem. Nawaz et al (1983) have proposed the fast insertion heuristic. The heuristic is referred to as FIH (fast insertion heuristic) or sometimes as NEH (Nawaz, Enscore, Ham). An extensive comparison of 16 various heuristics, presented by Park (1981), reveals that the fast insertion heuristic performs significantly better than the others. In particular, he concluded that "it is clear that NEH is the least unbiased and the best-operated of the heuristics tested on the small static flow shop problems ... and CDS is the next best”. For large problems, it is indicated that “NEH is the least biased and the most effective”.   Lawler et al (1993) also indicated that “the current champions are the fast insertion heuristic (FIH) and the less efficient simulated annealing algorithm of Osman and Potts”.

Because of its strong relationship to this research, we now describe the FIH heuristic in detail. The heuristic is based on the assumption that a lot with more total processing time on all the machines should be given higher priority than a lot with less processing time. In that spirit, the two lots with the highest processing times are selected first and the best partial sequence of these two lots is determined by evaluating the two possible partial sequences. The relative position of these two lots, with respect to each other, is fixed in the remaining steps of the algorithm. Next, the lot with the third highest total process time is selected and again, all the possible partial sequences, created by inserting that lot in the beginning, middle, and end of the partial sequence established so far, are evaluated. The best partial sequence is then fixed and the process is repeated with the fourth highest process time lot, etc. The heuristic can be implemented using the procedure presented next.

<u>The Fast Insertion Heuristic (FIH) for flow shop sequencing problems</u>

Step 1. For each lot *i*, compute:

$$T_i = \sum_{j=1}^{m} p_{ij}$$

Step 2. Arrange the lots in a LIST, in decreasing order of T_i.

Step 3. Pick the first two lots on the LIST.

Find the best partial sequence of these two lots, and make it the 'current sequence'

Set i=3

Step 4. Pick the lot in the *i*-th position on the LIST.

Insert it in all possible positions in the 'current sequence' and evaluate the sequence.

Make the best resultant partial sequence the 'current sequence'.

Step 5. If $i \le n$, then set $i = i + 1$ and go to Step 4.

The complexity of the FIH heuristic is of order $O(n^2)$ since, at each iteration of step 4, at most *n* positions are considered and this step is repeated (*n*-*2*) times. It can be further shown that the number of enumerations in the algorithm is:

$$\frac{n \cdot (n+1)}{2} - 1$$

When using heuristics to approach a problem, an important question that arises pertains to the quality of the solutions generated. A common way to check the quality of the solutions generated is to compare those with some known, preferably tight, lower bounds. For the flow shop sequencing problem, several such bounds are available in the literature. The bounds differ in the amount of information and the computational effort required to utilize them. Two well known bounds are the 'job-based bound' and the 'machine-based bound'. For additional bounds, see Baker (1974), Pinedo (1995), and Santos et al (1995). The job-based bound states that the makespan is at least as large as the highest total processing time of any job on all the machines:

$$LB = \max\left\{\sum_{i=1}^{n} p_{ij}\right\} \quad \forall j = 1,..,m \tag{2.3.1-4}$$

Accordingly, the machine-based bound states that the makespan is at least as large as the highest total processing time of all the jobs on any machine:

$$LB = \max\left\{\sum_{j=1}^{m} p_{ij}\right\} \quad \forall i = 1,..,n \tag{2.3.1-5}$$

Variations of these bounds are used in the sequel to test the proposed heuristics for the lot-streaming sequencing problem (LSSP).

2.4 Lot Streaming

2.4.1 Terminology and definitions

Lot streaming is the process of splitting production lots into sublots, and then scheduling the sublots in an overlapping fashion on the machines (or more generally, the resources), in order to accelerate the progress of orders in production (Baker and Jia, 1993 ; Chen and Steiner, 1997), and to improve the overall performance of the production system.

In different contexts, the term "lot" is also referred to as a "batch" or a "job". While the terms "lot" and "batch" are commonly used to describe a group of parts of the same type, going through the same production processes, the term "job" is more often used to describe a computational task to be performed by a computer processor. These terms are used interchangeably in the sequel. Consequently, "transfer lot", "sublot", and "transfer batch" are all used to describe the portions of an original lot (batch), which are streamed (or split) throughout the production system.

Lot streaming is sometimes confused with classical preemption (Vickson and Alfredsson, 1992). However, the two are not the same. Preemption refers to the possibility to stop the processing of a job while it is being processed prior to its completion, presumably in order to make changes in the sequence. In contrast, lot streaming does not pertain to stopping any processing that has begun. Lot streaming refers to the opportunity of streaming portions of the jobs to downstream machines as processing of the jobs on upstream continues. The processing of the jobs on the upstream machines is never stopped and no changes to the sequence are made. Lot streaming is thus a special sequencing problem in which each job is known to be comprised of identical items and the total processing time of the job is the sum of the processing times of all the items in it. Having this point clarified, we now turn to a review of the existing literature on lot streaming.

We differentiate between models which deal with a single lot (single batch models) and those which deal with multiple lots (multiple batch models). Within each of these two categories, the current research is further classified to models that consider equal sublots and models that do not.

2.4.2 Single batch models

2.4.2.1 Problem statement

The problem of the single batch models can be stated as follows: given a certain number of transfers (or, alternatively, an indirect constraint on this number), what are the optimal sublot sizes to be used, to stream the batch through the production system, in order to optimize a given objective function (typically, but not necessarily, makespan.) A generalization of this problem is possible if the number of transfers is considered to be a decision variable as well.

Single batch models are valid in their own right but they are also motivated by the reasoning that they provide a useful insight into the more complicated case of multiple batches. Further, it is expected that methods developed for the single batch case can then be used for multiple batches by simply applying them to each batch independently. However, as we shall see, the problem of multiple batches is much harder and a straightforward implementation of the single batch models is most likely not possible.

Since the issue of sublot sizes is addressed by the single batch models, the characteristics of the transfer mechanism are essential. Obviously, different problems arise when different transfer mechanisms are utilized. The capacity of the transfer devices (and the entire material handling system) must be considered in order to determine the rate at which the sublots can be moved throughout the system. The shape and size of the transfer devices are also important factors as they limit the sublots from exceeding a certain size. Finally, the cost associated with each move (or usage of the transfer device) limits the total number of sublots to be used.

In addition to the limitations associated with the transfer mechanism, the question of whether to use equal-sized sublots or not is a strategic question that needs to be addressed by production managers. Clearly, there is an advantage in not restricting the sublot sizes, at least initially, because such a restriction further limits the space of feasible solutions. However, in many cases, equal-sized sublots are a prerequisite since they are easier to track, manage, and control. In single batch models, two other types of sublot sizes are considered in addition to equal sublots. These two types are referred to as ‘consistent’ and ‘variable’ sublots. By ‘consistent’ it is meant that the sublots are of the same size on all the machines and therefore only the initial sizes (the sublot sizes on the first machine) need to be determined. In that case, the identity of each sublot can be established by associating a number with it, because the sublots would have the same size throughout the system. A more general case would be if it is allowed to further split the sublots as they are processed on the machines. This is the 'variable' sublot case. In that case, the number of sublots on each machine may change and therefore, to establish the optimal sublot sizes, two numbers should be specified - one to indicate the (serial) machine and one to indicate the sublot number emanating from that machine.

An additional constraint when considering a single batch is regarding the processing of the machines. A restriction can be imposed that the machines must process the sublots continuously. In other words, it is desired that, once a machine has begun its processing on the first sublot, subsequent sublots will be processed without any idle time incurred. Such a requirement is not uncommon in manufacturing processes (e.g., to avoid cooling, or chemical deterioration) and may be a consequence of economical and/or non-economical considerations (such as safety and contamination.) Note that continuous processing can also be interpreted as follows: the machine should not idle in between the processing of the sublots. The 'continuous processing' requirement and the 'no-idling' requirement are identical.

Before we present a classification of the existing research on single batch models, there is one final issue, a modeling issue, that requires attention. This issue addresses the integrality of the sublots. As we shall see, an additional restriction of integrality makes the resulting mathematical models substantially harder to solve (Potts and Baker, 1989.) Typically, such a restriction would result in a mixed integer linear programming (MILP) problem for which a polynomial solution is not known.

2.4.2.2 Classification of existing single batch models

Next, a classification of the majority of single batch models investigated so far is provided. Following the classification proposed by Trietsch and Baker (1993), the classification given here is made via four criteria. These criteria are as follows:

- Number of machines (*m*)
- Sublot types: *V, C, or E* (Variable, Consistent, or Equal, respectively)
- Idling (*II*) or No Idling (*NI*)
- Solution type: Continuous (decision) variables (*CV*) or Discrete (*DV*)

Table 2.1 summarizes the research achievements in the case of single batch models. Table 2.1(a) presents the research on single batch models for the two- and three-machine flow shop while Table 2.1(b) presents the research on single batch models for the m-machine flow shop. In the sequel, research into the problems provided in Table 2.1, as well as additional research (not covered in Table 2.1) is presented. We divide the existing research of single batch models into two groups. The first group consists of models which restrict the analysis to equal sublots while the other group consists of models that do not impose that restriction.

Table 2.1. Summary of single batch models

Table 2.1(a). Single batch models for the two- and three-machine flow shop

Seq. No.	No. of Machines	Sublot Type	Idling	Solution Type	Basic Approach Utilized
1	2	C (or V)	NI (or II)	CV	Closed-form formula for the sublot sizes: $q = \frac{p_2}{p_1}$; $L_1 = \frac{Q \cdot (q-1)}{q^n - 1}$; $L_j = q^{j-1} \cdot L_1$ (Trietsch, 1989)
2	2	C	II	DV	Recursive formula on the sum of sublots $S_r = \sum_{k=1}^{r} L_{jk} \Rightarrow$ $S_r \leq \min\left\{ \frac{M - p_2\left(Q - S_{r-1}\right)}{p_1}, Q \right\}$ (Trietsch and Baker, 1993)
3	3	C	II	CV	Network representation (Glass et al., 1992)
4	3	V	II	CV	(When the second machine is dominant) Decomposition to two 2/C/II/CV (Trietsch and Baker, 1993)
5	3	V	II	DV	(When the second machine is dominant) Recursive formula on the sum of sublots, as in 2/C/II/DV (Trietsch and Baker, 1993)

Table 2.1(b). Single batch models for the *m*-machine flow shop

Seq. No.	No. of Machines	Sublot Type	Idling	Solution Type	Basic Approach Utilized
1	m	V	NI	CV	Decomposition to (m-1) pairs of 2/C/NI/CV problems (Baker, 1988)
2	m	C	II	CV	Linear Programming formulation $\min\ \{t_{mn}\}$ $s.t.\ \ t_{11} \geq p_1 \cdot L_1$ $t_{jk} \geq t_{j-1,k} + p_j \cdot L_k \quad \forall \text{j} = 2,\ldots,\text{m}\ ;\ \forall \text{k} = 1,\ldots,\text{n}$ $t_{jk} \geq t_{j,k-1} + p_j \cdot L_k \quad \forall \text{j} = 1,\ldots,\text{m}\ ;\ \forall \text{k} = 2,\ldots,\text{n}$ $\sum_{k=1}^{n} L_k = Q\ ;\ t_{jk},\ L_k \geq 0 \quad \forall \text{j},\text{k}$ (Trietsch and Baker, 1993)
3	m	C	II	DV	Integer Programming formulation, with the L_j integers in the above formulation (Trietsch and Baker, 1993)
4	m	E	II	DV	Closed-form formula for the makespan: $M = \frac{Q}{n}\left[\sum_{j=1}^{m} p_j + \left((n-1)\,p_{\max}\right)\right]$ Optimal sublot size: $L^* = 1$ Baker (1988)
5	m	E	NI	DV	Closed-form formula for the makespan: $M = \frac{Q}{n}\left[\sum_{j=1}^{m} p_j + \left((n-1)\sum_{j=1}^{m}(p_j - p_{j-1})d(p_j - p_{j-1})\right)\right]$ Optimal sublot size: $L^* = 1$ Szendrovitz (1975)

2.4.2.3 Equal sublots

We start our review of single batch models with the case of equal sublots. The literature in this category is rather limited. Baker (1988) gives the following closed-form formula for the makespan of a single batch, split into *n* equal sublots, in an *m*-machine-flow-shop:

$$M = \frac{Q}{n}\left[\sum_{j=1}^{m} p_j + \left((n-1)p_{\max}\right)\right] \qquad (2.4.2.3\text{-}1)$$

where:

M -	Makespan
Q -	Batch size
n -	Number of sublots in the batch
m -	Number of machines, j=1..m
p_j -	Processing time of machine j ; $p_{\max} \equiv \max\limits_{1\le j\le m}\{p_j\}$

In this case the optimal continuous solution is:

$$n \to \infty$$
$$M = Q \cdot p_{\max}$$

However, this is clearly infeasible since the lot is not infinitely divisible. The discrete solution is to have Q sublots of unit size:

$$n^* = \max\{n\} = Q$$
$$M = (Q-1)\cdot p_{\max} + \sum_{j=1}^{m} p_j \qquad (2.4.2.3\text{-}2)$$

The expression can be extended for cases when a 'no-idling' constraint is imposed on the machines (i.e., the machines must operate continuously):

$$M = \frac{Q}{n}\left[\sum_{j=1}^{m} p_j + \left((n-1)\sum_{j=1}^{m}(p_j - p_{j-1})d(p_j - p_{j-1})\right)\right] \qquad (2.4.2.3\text{-}3)$$

where:

$d(p_j - p_{j-1}) = p_j - p_{j-1}$, if $p_j - p_{j-1} \ge 0$, and 0 otherwise

Exp. (2.4.2.3-3) is due to Szendrovitz (1975). As mentioned earlier, such a restriction of 'no-idling' can be imposed when the process itself dictates it (e.g., to avoid cooling, or chemical deterioration.)

Potts and Baker (1989) provide an upper bound for the worst performance of equal sublots relative to consistent sublots, with respect to a makespan objective:

$$\frac{M^E}{M^C} < 1.53$$

where:

M^E - optimal makespan using Equal sublots

M^C - optimal makespan using Consistent sublots

Unfortunately, no additional analysis is provided as to how tight this bound is. For *n*=2 (i.e., two sublots) and for *m*=2 (i.e., two machines), the bounds are significantly lower ($\frac{9}{8}$ and 1.09, respectively) and are also tight.

Some issues that have not been addressed by researchers in this case are:

- The effect of (non-negligible) transfer and setup times on the optimal sublot sizes and the optimal number of sublots.
- The optimal solution for objectives other than makespan, such as a cost-based objective.

2.4.2.4 Unequal sublots

Most of the research into single batch models falls into the category of unequal sublots. Although this problem is more difficult to solve, a significant progress has been made. Due to the complication associated with variable sublots and the difficulties in managing production systems which utilize them, the vast majority of the research of unequal sublots considers only consistent sublots.

In an early research conducted by Truscott (1986), a branch and bound solution approach is proposed to the problem of minimizing the makespan in a two-machine-flow-shop, where the transfer is performed by a limited number of transporters. The limited availability of the transporter makes it necessary to decrease the number of sublots to the minimum, as well as to fully utilize the transporter. Under the assumption that it takes each transporter *CT* time units to complete a cycle (i.e., load, transfer, unload, and return), the number of feasible transfers during the makespan is bounded (in other words, the number of sublots, *n*, is determined by this mechanism.) and an iterative procedure can thus be utilized to determine the optimal number of sublots. However, the optimal solution can also be obtained explicitly, by setting the first sublot to the size of:

$$\min\left\{\frac{CT}{p_2}, Q\right\} \tag{2.4.2.4-1}$$

and then setting the remaining sublots accordingly, using geometrically increasing sublot sizes:

$$L_j = L_1 \cdot q^{j-1} \tag{2.4.2.4-2}$$

where: $q = \frac{p_2}{p_1}$

Here, it is assumed, without loss of generality, that $p_2 \geq p_1$, i.e., $q \geq 1$. In case that $\frac{CT}{p_2} < 1$, the first sublot should be set to one. The solution specifies that the first sublot has to be large enough to last on the second machine to enable the transporter to return to it before (or immediately as) the current sublot is completed on that machine. Generalizing this result for the case of k transporters is straightforward (Trietsch and Baker, 1993). The first *k* sublots must satisfy the following:

$$L_1 + L_2 + \ldots + L_k \geq \frac{CT}{p_2} \tag{2.4.2.4-3}$$

which leads to the following optimal sublot sizes:

$$\begin{aligned} L_1 &= \frac{CT}{p_2} \cdot (1 + q + \ldots + q^{k-1}) = \frac{CT}{p_2} \cdot \frac{1-q}{1-q^k} && \text{if } q \neq 1 \\ L_1 &= \frac{CT}{p_2 \cdot k} && \text{if } q = 1 \end{aligned} \tag{2.4.2.4-4}$$

For more than two machines (m>2), the limited transporting equipment problem appears to be difficult to solve, and has not been dealt with (Trietsch and Baker, 1993).

Trietsch (1989) examines a variation of the above problem where a budget constraint is imposed on the number of transfers. He assumes that each transfer costs a fixed amount of C money units, while the total available budget is B. It is also assumed, in his model, that each transfer takes TT time units (independent of the sublot size) and that the machines must be setup prior to processing. To obtain optimal sublot sizes, the case of zero setup or transfer times is analyzed first. It is shown that, for the case similar to the above, the optimal sublot sizes follow the geometrical relationship (2.4.4-2). Consequently, if the number of possible transfers (i.e., sublots) is bounded by $u = \left\lfloor \frac{\mathrm{B}}{\mathrm{C}} \right\rfloor$ (the rounded down value), then the optimal size of the first sublot is:

$$L_1 = Q \cdot (1 + q + \ldots + q^{u-1}) = Q \cdot \frac{1-q}{1-q^u} \qquad \text{if } q \neq 1$$
$$L_1 = \frac{Q}{u} \qquad \text{if } q = 1 \tag{2.4.2.4-5}$$

If the first sublot (or, alternatively, the last sublot) turns out to be less than one, then the solution can be obtained by setting it to one. In the presence of setup and transfer times, the first sublot is simply set to at least:

$$L_1 = \frac{su_2 - su_1 - TT}{p_1} \tag{2.4.2.4-6}$$

where:

su_j - The setup time of machine j.

For the case of multiple batches, Trietsch proposes a straightforward generalization of the single batch model solution in which multiple batches are processed sequentially, each according to the geometrical progression specified in (2.4.4-2). The size of the first sublot of each batch (except the first batch) is determined by adding the time difference between the completion times of the previous batch on the second machine and the first machine to the difference between the setups of the next batch on the two machines (2.4.4-6). The rest of the computation remains the same. For the case of multiple machines (*m>2*), a greedy heuristic is proposed, but no analysis is carried out to support its performance.

Recently, Chen and Steiner (1997) study the structural properties of schedules which minimize the makespan for a single lot with detached setup times in a flow shop. They show that it is possible to find the optimal solution with *s* sublots in O(log *s*) time for the three-machine flow shop case.

Potts and Baker (1989) derive some general properties of the lot streaming problem. They first provide a proof that there exists an optimal schedule (i.e., minimum makespan) in which the same sublot sizes are processed on the first pair of machines, and consequently, via the inverse property (which applies for the makespan objective, but not the others, such as flowtime), on the last pair of machines. It follows that, for *m*=2 and *m*=3, an optimal schedule in which the sublots are of consistent size exists (and this is true for any number of batches, N.) For $m \geq 4$, a counter-example is given to show that, the optimal schedule is not necessarily composed of consistent sublots. Another example is given in Trietsch and Baker (1993). For *m*=2, the optimal sublots are consistent, and it is shown that the makespan for that case is as follows:

$$M = p_1 + \frac{1}{p_1}\left(\frac{1-q^{n+1}}{1-q^n}\right) \tag{2.4.2.4-7}$$

where: $q = \frac{p_2}{p_1}$

Without loss of generality, let $p_1 = 1$. The makespan then becomes a function of *q* as follows:

$$M = 1 + \frac{1-q^{n+1}}{1-q^n} \tag{2.4.2.4-8}$$

where: $q = p_2$.

This formula shows that, as $n \to \infty$, $M \to 1+q$. The lower limit on *M* is $\max\{1, q\}$, which is the processing time of the bottleneck machine. Note, however, that the underlying assumption in this analysis is that, the batch is infinitely divisible.

Baker and Pyke (1990) propose a computationally efficient procedure to obtain the optimal consistent sublots for the *m*-machine-flow-shop, under the restriction of two sublots only, i.e., *n*=2. They also examine heuristics for more than two sublots. Trietsch and Baker (1993) summarize the work on single batch models. They propose a framework for the classification of single batch models. They also provide extensions to some of the existing models. In particular, they address two issues:

- Obtaining integer solutions from optimal continuous solutions.
- Variable sublots.

To obtain integer solutions from optimal continuous solutions, an algorithm is provided for the case of *m*=2, which is then extended to *m*=3. The algorithm works as follows:

Define:

$S_r = \sum_{k=1}^{r} L_{jk}$ (- the sum of the first *r* sublots.)

$LS_r = M - p_2\left(Q - S_{r-1}\right)$ (- the latest start time of the r-th sublot on machine 2.)

LS_r is the latest time at which the r-th sublot can begin on machine 2, and still allow for all remaining work to be finished on machine 2 by time *M*.

For feasibility, we must complete the r-th sublot on machine 1, no later than LS_r:

$$p_1 \cdot S_r \le M - p_2\left(Q - S_{r-1}\right)$$

$$\Rightarrow \quad S_r \le \min\left\{\frac{M - p_2\left(Q - S_{r-1}\right)}{p_1}, Q\right\} \qquad (2.4.2.4\text{-}9)$$

This is a recursive formula with respect to S_r, for which $S_0 = 0$, and S_r is the largest feasible integer (i.e., $\lfloor S_r \rfloor$, the rounded down integer.) If $S_n < Q$, this implies that M is infeasible. Then, *M* is increased as follows:

Define: $f_r = S_r - \lfloor S_r \rfloor$, the fraction of the part that machine 1 will have processed when the r-th sublot is transferred. The appropriate increment to *M* is:

$$p_1 \cdot \min_r\{1 - f_r\} \qquad (2.4.2.4\text{-}10)$$

which is the minimal possible increment. This process continues until a feasible integer solution is obtained. Procedure of the same nature is introduced for the case of limited transporting equipment as well. When the transporters are of limited capacity, it is shown that if the transporter's capacity (UB) is binding, the solution calls for equal sublots of size UB (with a possibly smaller last sublot.)

The second issue that is dealt with by Trietsch and Baker (1993) is the issue of variable sublots. To motivate the need to investigate the performance of variable sublots, they show that for *m*>2 consistent sublots are not necessarily optimal. Breaking the sublots further between each pair of machines can result in better makespan. Their work, with regard to variable sublots extends the previous works of Baker (1988) and Glass et. al. (1992).

Baker (1988) develops expressions to obtain the optimal variable sublots when *m*=3, and *n*=2 sublots. Glass et al. (1992) provide a thorough analysis for *m*=3, and consistent sublots. They observe that the optimal sublots are consistent if the second machine is dominated, i.e., if: $p_2^2 \leq p_1 \cdot p_3$. This condition is also referred to as the 'no-wait' condition since it is equivalent to the requirement that the sublots would not wait on the second machine (see Trietsch and Baker, 1993 for a proof.) Sriskandarajah and Wagneur (1995) have provided the discrete optimal solution when the 'no-wait' condition holds on the second machine of a two-machine flow shop.

Encouraged by the above findings, Trietsch and Baker (1993) investigate the other case of a three-machine flow shop, where the second machine dominates the other two. They show that in this case, the optimal solution of variable sublots, can be obtained by decomposing the problem of *m*=3 into two sub-problems of *m*=2, namely {machines 1,2} ; {machines 2,3}, and solving each of these sub-problems independently. We note that the case of variable sublots has not been dealt with thoroughly because of the obvious physical disadvantages of implementing it. It also complicates the analysis since if the sublot size is not kept anymore while being transferred from one machine to the next, there is a need to incorporate this in the notation. Thus, the notation utilized in such cases is as follows:

L_{jl} - The size of the *l*-th sublot emanating from machine *j*

x_{jl} - The portion of the *l*-th sublot emanating from machine *j*, relative to the original batch size, and $x_{jl} = \frac{L_{jl}}{Q}$.

Trietsch and Baker (1993) also discuss LP formulations for the single batch problem in case of consistent sublots. In general, the (continuous) optimal consistent sublot sizes can be obtained by solving a Linear Programming problem as follows:

$$\begin{aligned}
&\min \ \{t_{mn}\} \\
&s.t. \\
&t_{11} \ge p_1 \cdot L_1 \\
&t_{jk} \ge t_{j-1,k} + p_j \cdot L_k \qquad \forall \mathrm{j} = 2,\ldots,\mathrm{m}\ ;\ \forall \mathrm{k} = 1,\ldots,\mathrm{n} \\
&t_{jk} \ge t_{j,k-1} + p_j \cdot L_k \qquad \forall \mathrm{j} = 1,\ldots,\mathrm{m}\ ;\ \forall \mathrm{k} = 2,\ldots,\mathrm{n} \\
&\sum_{k=1}^{n} L_k = Q \\
&t_{jk},\ L_k \ge 0 \qquad \forall \mathrm{j},\mathrm{k}
\end{aligned} \tag{2.4.2.4-11}$$

where:

t_{jk} - The completion time of the *k*-th sublot on machine *j*.

The first constraint ensures that the time to complete the first sublot on machine 1 is at least the time it takes to process it on machine 1. The second set of constraints assures that the completion time of the k-th sublot on machine j is at least equal to the time to complete it on machine j-1 (the preceding machine) plus the time to process it on machine j. The third set of constraints specifies that the completion time of the k-th sublot on machine j must at least be equal to the time to complete sublot k-1 (the preceding sublot) on machine j plus the time to process sublot k on machine j.

This formulation contains ($2 \cdot m \cdot n - n - m + 2$) constraints and $\left[n \cdot (m+1)\right]$ variables. A more compact formulation utilizes the idle times as decision variables. In both cases, however, an LP approach to the problem provides only little insight, and does not exploit the special structure of it (Baker and Jia, 1993). In particular, we note that in order to model the simpler case of equal sublots via LP, another set of constraints would have to be introduced which specifies that the sublots are all equal. Thus, the LP cannot be utilized to simplify the model in that case. It is important to note that both formulations assume that the number of sublots, *n,* is known. In case *n* is unknown, it is necessary to incorporate additional constraints that would limit *n* from becoming infinitely large.

Without exception, research presented thus far has been concerned with the objective of minimizing the makespan. Kropp and Smunt (1990) present a quadratic programming formulation for minimizing a different objective function. They use the mean flow time (MFT) of the sublots as an objective. Imposing the requirement of integral sublots makes their formulation become an integer programming (thus, computationally intractable.) They provide several reasons why it is better to solve the LP problem instead:

- It is computationally solvable
- it is the belief that the rounded continuous solution (which preserves feasibility) is reasonably close to the optimal discrete solution
- A wider variety of settings can be considered.

In their work, they carry out a full factorial experiment to test the effects of the following factors on the optimal sublot sizes:

- The number of sublots, n=2, 3, and 4
- The number of machines, m=2, 3, 4, and 5 ; and
- The ratio of setup time to total processing time = $0 \div 4$

They draw several conclusions based on their experiments. First, they conclude that as expected, when the setup times dominate the processing times (i.e., ratio > 1) the effect of lot streaming becomes inconsequential. Second, for relatively small setup times (ratio << 1) they observe that as the setup times increase and n>2, sublots of equal sizes (except the first sublot) tend to be optimal. They summarize with a justification for the use of equal sublots. In addition to the above result, they provide two other important reasons to support the use of equal sublots. First, a standardized container size is more convenient. Secondly, using equal sublots would likely reduce the number of implementation errors.

Recently, Eynan and Li-Chung (1997) have also considered a lot-splitting model where the objective is to maximize the net present value of the total revenue collected at the delivery of the sublots. Their objective deviates from the makespan in two respects. First, it considers the mean flow time (and not the total processing time of the entire lot.) Second, it is economical and not operational. In their work, they assume that the unit manufacturing time follows a learning curve. An algorithm is developed to solve the problem and they show that two 'convenient' operational approaches, namely the equal sublot size approach and the equal time interval approach are nearly optimal.

Baker and Jia (1993) present further results concerning the impact of equal and consistent sublots on system performance. Performance comparisons of equal and consistent sublots in a three-machine-flow-shop are provided. Cases where the second machine dominates as well as cases where it is dominated are considered. Their results indicate the following:

- As the number of sublots increases, the ratio $\frac{M^E}{M^C}$ is approaching 1, implying that it is less attractive to use consistent sublots and not equal sublots.
- $\frac{M^E}{M^C}$ is slightly lower in cases where the second machine is dominated in comparison with cases where it dominates.
- As expected, the improvements in the makespan, *M*, are diminishing as the number of sublots increases. Moreover, roughly 80% of the benefit from improved makespan can be obtained by using only three sublots! (*n*=3). Therefore, a small number of sublots is sufficient to achieve most of the benefits of lot streaming, regardless of the technique used. This finding coincides with Kropp and Smunt (1990).

2.4.3 Multiple batch models

2.4.3.1 Problem statement

In this case, research has been mainly concerned with the following problem: given the sublot sizes, what is the optimal sequence (and corresponding schedule) which optimizes a given objective function. The objective function that is typically used is the makespan, as in the case of single batch models (Cetinkaya, 1994). Clearly, this problem is strongly related to the classic sequencing and scheduling problem, since the model in this case addresses the sequence and not the sublot sizes. The reasoning for this has been that single batch models can be used to obtain the optimal sublot sizes for the batches. However, recently, several papers have also considered the sublot sizes to be determined simultaneously with the sequence (Dauzere-Peres and Lasserre, 1997), but the research in that area is still very limited.

A few papers have considered variations of the lot-sizing problem with lot streaming as a secondary issue (e.g., Moily, 1986 ; Pinto and Rao, 1992). These papers are also discussed later. However, most of the literature on lot-sizing techniques considers the problem of finding the optimal lot sizes, independent of the manner in which these lots are then produced. Given these lot sizes, the issue of lot streaming becomes significant, since the routing of the lots and the possible ordering of lots on the machines are essential (Dauzere-Peres and Lasserre, 1993).

Several issues arise when considering multiple batches, that could have been ignored in single batch models. First, it may be beneficial to let the sublots intermingle, i.e., to form a sequence of sublots in which sublots of the same lot are not necessarily processed one after another in an unbroken manner. If this is indeed the case, then it rules out the possibility of using a hierarchical approach, as suggested in the previous section, to solve the problem in two phases (at the first phase of which optimal sublots of each batch are obtained while at the second phase the optimal sequence of lots is realized.)

The second issue, which relates to the first, is the setup. Normally setup would be incurred in between the processing of any two different sublots. While in the case of a single batch, setup in between lots is not an issue since only a single lot is considered, it becomes an important issue in the case of multiple batches, in particular, if intermingling is allowed. In a sequence of intermingled sublots, setup would be incurred not only between the lots but also between any two different sublots. Clearly, larger setup times would cause intermingling to be less attractive.

The third issue is concerned with the determination of the number of sublots. If consistent (or yet variable) sublots are to be used and the sublot sizes are to be determined by the model, a decision variable for each lot, representing the number of sublots in the lot, would be required. However, for the case of equal sublots, only a single additional decision variable would be required, to represent the sublot size. Since the lot sizes are known, the sublot size would also determine the number of sublots in each lot.

2.4.3.2 Equal sublots

When setup times are ignored (i.e., assumed negligible or absorbed in the processing times) and there is no limit on the number of transfer batches and the transfer times are also negligible, it is optimal to use equal unit-sized sublots. A proof of optimality can be found in Vickson and Alfredson (1992). However, this is clearly an unrealistic case. Potts and Baker (1989) argue that a justification for using equal sublots, in the case of multiple lots, is the following: sequencing the lots optimally, then splitting each lot independently to optimal sublots -- does not guarantee optimality. They provide an example, in which even for the simplest case of N=2, m=2, the hierarchical (or sequential) solution is not optimal. Under these circumstances, the effectiveness of the solutions resulting from considering unequal sublot sizes is questionable, and it may be better to consider equal sublots that guarrantee near optimal solutions instead. Nevertheless, several approaches have utilized this hierarchy to obtain a solution, usually for the more general case of unequal sublots (see Cetinkaya, 1994).

Vickson and Alfredsson (1992) consider multiple batches on two and three machines with equal sublots. In the two-machine case, they show that there exists an optimal schedule in which sublots from the same batch are processed continuously (i.e., not intermingled with sublots from other batches.) Their proof is based on Johnson's rule (Johnson, 1954) described in section 2.3.1. Applying Johnson's rule on the sublots $\{1,\ldots,n_1,1,\ldots,n_2,\ldots,1,\ldots,n_N\}$, it easily follows that if a sublot of lot i results in being first, then all the n_i sublots of lot i are scheduled first in an unbroken string.

In addition, Vickson and Alfredsson (1992) show that, in the presence of transfer time, *TT*>0, there still exists an optimal schedule in which the sublots are not intermingled. They also provide insights into the three-machine case and present an empirical study of a two-machine-flow-shop (*m*=2), in which the following values are tested:

- Number of batches, *N*=5, 8, 10, 15, and 20
- Number of sublots, *n*=2, 6, and 10.

Their conclusions are as follows:

- Relative improvements in the makespan are as high as 25%. It is their belief that improvements may be higher as the number of machines increases
- Even larger improvements are achieved in regard to mean flow time. However, the improvement rate decreases as *N* increases, or as less transfer batches are utilized (i.e., *n* decreases).

Baker (1995) extends the analytical work of Vickson and Alfredsson (1992), for the two-machine-flow-shop, by incorporating setup times into the model. In his model, he uses results from scheduling theory of models with time lags, on a lot streaming model with time lags that accounts for setups which are attached to the processing (the time lag model, however, accommodates both attached and detached setups.) The analysis relies heavily on the fact that, in a two-machine-flow-shop, the optimal schedule is a permutation schedule (i.e., the sequence is the same on both machines.) This need not be the case when there are more than two machines. Thus, the sequencing algorithm developed for the two-machine-case does not, in general, extend to three machines. However, extensions for special cases of the three-machine-flow-shop are provided.

Kropp et al. (1988) evaluate the ‘flag’ heuristic, in a 10-machine-flow-shop, via a simulation model. In a ‘flag’ heuristic, a single part is accelerated throughout the system to setup the machines prior to the arrival of the lot (or the sublots). This type of heuristic assumes that setups are detached, i.e., that setup can be performed without having the entire sublot that needs to be processed on hand. They conclude that, unless there is a clear flow dominance (i.e., bottleneck), lot-splitting offers only little benefit. However, when there is a clear flow dominance, then using a ‘flag’ policy and equal sublots is better than just using equal sublots.

El-Najdawi and Kleindorfer (1993) provide a framework for a multi-stage, multi-product flow shop system, generalizing earlier CCSP results, as well as the single product results of Szendrovits (1975). Their model is formulated to minimize total costs for all products while meeting stationary given demands. El-Najdawi (1994) notes that, under circumstances of lot streaming, the CCSP is more complex. In this case, seeking a common cycle for the entire set may be a bad compromise. In fact, if the ratio of setup cost to holding cost of one product is much larger than the rest, then forcing this one product to be produced in every cycle that the rest are produced is very inefficient. A more efficient procedure would be to produce this product less often than the rest. He proposes a computationally efficient lot-splitting heuristic to solve the Multi-Cyclic Scheduling Problem in a multi-stage, multi-product production system. The model formulated aims to minimize the total costs of setup and WIP, where WIP costs are time weighted.

Notice that thus far, the vast majority of the research has considered the makespan as the objective to be minimized. Other objectives have been negleted. This is, presumably, due to the complexity of the models involving other types of objectives. One exception is the work by Wagner and Ragatz (1994), who investigate, via simulation, whether job splitting results in a better due date performance, in a five-machine-open-job-shop. In the open job shop system, flow time reductions are solely a result of overlapping operations, rather than saved setups from processing similar sublots in sequence. They examine equal sublots of different sizes. Their conclusion is not trivial at all: lot splitting provides improvements that are essentially independent of the setup time(!), as long as the utilization (i.e., the ratio: $\frac{\text{Setup Time + Processing Time}}{\text{Available Time}}$) is constant. They have shown that, in some cases, lot streaming can result in substantial reductions in mean tardiness (up to 39%), as well as in the number of tardy lots. They conclude that, lot streaming is a simple, yet very effective technique, for improving the performance of an open job shop. It also happens to be robust (as long as the utilization level is kept) thus, it can be combined with other shop floor control techniques.

2.4.3.3 Unequal sublots

One way to simultaneously determine both the sublot sizes and their sequence is to analyze the setup times associated with each machine and each sublot type. This approach does not apply for the single batch models, simply because the setup is, typically, not considered for a single lot (i.e., in between sublots of the same type.) Obviously, as the setup time increases, it becomes more beneficial to increase the sublot size as well, thus decreasing the number of setups required for all the sublots of the same type. A two-machine-flow-shop with setup and removal times is studied by Cetinkaya (1994). The objective considered is to minimize the makespan, and an optimal sublot sizing and scheduling algorithm is presented. The number of sublots is assumed to be given. Pyreddy (1996) proposes a heuristic extension to the single batch optimal soltion algorithm, developed by Trietsch (1989), for the two machine multiple batch lot streaming problem with setups. The idea is to obtain the discrete sublot sizes via Trietsch's procedure for each lot independently, and then, to apply a variation of Johnson's rule to sequence the lots in a non-intermingled manner.

Pinto and Rao (1992) consider the lot-sizing problem of minimizing the inventory holding costs, in a flow shop, while satisfying the production capacity limitations and the external demands. Their constraints allow for transferring lots in sublots. They assume that:

- Transfer times are negligible
- Products are produced in a cyclic fashion, i.e., the optimal schedule may be found by considering a single cycle.
- The demand, for all products, is known and deterministic.

These assumptions are typical under the CCSP mentioned earlier. They propose a heuristic procedure which obtains lot sizes, their sequence, and the corresponding schedule(including the transfer batches.) These three are obtained in a sequential fashion rather than simultaneously. They highlight the significant impact of unit-sized sublots. However, they indicate that, practically, containers are transferred from one machine to another only when full. Therefore, the lot sizes obtained by their procedure are modified to be integral multiples of the container size.

Dauzere-Peres and Lasserre (1993) propose an iterative procedure to obtain both the sublot sizes and the schedule, in a general job-shop. Their procedure alternates between solving for (a) the optimal sublot sizes, for a fixed number of sublots and a fixed sequence of these sublots on the machines ; and (b) the "optimal" schedule, for a fixed number of sublots and fixed sublot sizes. In (a), the optimal sublot sizes are obtained by solving the LP formulation (i.e., the sublots are consistent.) In (b), the task of obtaining the optimal schedule, even for fixed sublot sizes, in a job-shop system is NP-hard. Thus, the shifting bottleneck heuristic is used to solve for a near optimal schedule. They noted that, setup times may be considered via their model as well, simply by adding them in the LP formulation. This, however, turns the LP formulation into Mixed-IP formulation, since binary variables (representing whether setup is or is not performed) need to be introduced. They conclude that the makespan obtained by their procedure approaches a lower bound in a few iterations, thereby indicating that the procedure is efficient and "seems to yield a nearly global optimum". Another observation that they make is that the makespan is very close to the lower bound with only 3 to 4 sublots. That is, a small number of sublots is sufficient to gain most of the lot streaming benefits. This result coincides with similar results of Baker and Jia (1993) and Kropp and Smunt (1990) for the single batch models.

As mentioned earlier, models of lot-sizing which simultaneously consider lot-streaming, have typically used cost-based objective functions. However, the models for lot-streaming have typically used the makespan as the objective. One exception to this classification, is the work of Doutriaux and Sarin (1996), who suggest a cost-based objective for the lot streaming problem, to relax the need to pre-specify the number of transfers (or sublots), *n*. They propose an objective function which accounts for the production costs (proportional to the makespan), and the handling costs. The objective function used is as follows:

$$\min_{n}\left\{Z(n) = h \cdot n + u \cdot M(n)\right\}$$

where:

h	- handling cost per transfer
n	- number of transfer sublots
u	- marginal operational cost per unit time
M	- makespan time (function of n)

They investigate the case of two-machine-flow-shop, where they show that for the above objective function, the batches are processed continuously (i.e., the sublots are not intermingled) in the optimal solution. The sublot sizes are consistent and can be obtained by considering each batch independently. That is, the sequence which contains the sublot sizes that minimize their individual makespans separately in an unbroken string is the optimal sequence. The same theorem is also proved by Cetinkaya (1994).

Hence, by using an economical objective function, the need to pre-specify the number of sublots is no longer necessary. Its optimal value can be derived from the model. This type of analysis is similar to approaches used in another area of research namely, the Single Assembly Line Balancing (SALB) problem. The SALB problem can be described as follows: given a certain number of manual workstations, what is the optimal assignment of tasks to these stations, so as to minimize the cycle time (or to maximize the production rate) (Ghosh and Gagnon, 1989). Rosenblatt and Carlson (1985) have shown that, when considering an economical objective function, the optimal solution may differ from the one that minimizes the cycle time. They have used an objective function of the form:

$$\max_{n,c}\left\{ Z(n) = p \cdot \left(\frac{1}{c}\right) - \sum_{k=1}^{n} f(k) \right\}$$

where:

n	- number of workstations (variable)
c	- cycle time of the line (variable)
p	- contribution from unit product
f(k)	- cost associated with constructing the k-th workstation; k=1..n

The objective function accounts for the income from finished goods that the line produces, less the accumulated operational cost of the line workstations. The optimization is done on both the number of stations, *n*, and the cycle time, *c*, simultaneously. By using an economical objective function, the need to pre-specify one of them is no longer necessary.

2.5 Summary of Existing Research and Directions for Future Research

There exists a consensus among the researchers about the possible benefits that can be obtained via lot streaming. However, it appears that its theoretical properties are still poorly understood (Vickson, 1995). In this Section, a link is established between the different components of this literature survey, namely, existing PPC methods, flow shop scheduling, and lot streaming. The purpose of this is to justify and to further motivate the research presented throughout the remainder of this work. This is done by establishing the link in three steps. First, issues that are left unanswered by existing PPC methods with respect to lot streaming are summarized. Based on the literature presented on lot streaming, questions that remain open are emphasized. Next, the relationships between flow shop sequencing and scheduling and lot streaming are described, and the usefulness of the former to the latter is discussed. Lastly, a summary of the justifications for the use of equal sublots is provided.

2.5.1 PPC methods and lot streaming

Modern production planning and control methods call for the streaming of lots in production. This is evident in the Kanban method of Just-In-Time (JIT), the generalization of Kanban in the form of constant work-in-process (CONWIP), and in the optimized production technology (OPT) method which supports the theory of constraints (TOC). However, of the three methods, only OPT does not set a-priori conditions, that must be met, in order for the method to work. Kanban is the most restrictive of the three. For Kanban to work efficiently, setup times must be drastically reduced to a level in which they are practically negligible relative to the processing times. Process variability must also be eliminated. Balanced and repetitive manufacturing production layouts must be designed. If any of these conditions is not satisfied, Kanban would not perform as efficiently as might be expected. No solution is suggested within the framework of JIT to cases which do not meet the prerequisites. In that sense, lot streaming is a more general approach towards the shop floor problem, acknowledging the fact that it may not be possible to satisfy all the prerequisites of Kanban.

In CONWIP, the Kanban prerequisites are relaxed at the expense of higher WIP levels, and thus, higher inventory costs. To preserve the qualities of pull systems, CONWIP calls for keeping a strict level of WIP throughout the production system at all times. This is achieved by operating the system as a closed network. However, CONWIP does not address the issue of breaking the lots (batches) to sublots throughout the production. In addition, the question of how to sequence the lots (or the sublots) remains open. Moreover, since the WIP is predetermined, average WIP level is not considered an objective in CONWIP. Thus, the primary objective utilized by this method is throughput. It may, however, be possible to vary the WIP level around its constant level, proposed by CONWIP procedures, to further improve upon system performance, with respect to the average WIP as well.

OPT is therefore the only current PPC method which recommends using transfer batches and also provides some hints as to how these transfer batches should be determined. Nonetheless, no systematic approach is provided in OPT to determine the optimal sizes of the transfer batches or to determine the sequence of these transfer batches. The OPT software appears to be using transfer batches of equal sizes. Further, the concept of transfer batches in OPT is merely utilized to save time on the bottleneck and does not extend throughout of the system. Existing research on lot streaming indicates that further improvements are possible by optimizing the sizes of transfer batches and streaming them throughout the system and not merely on the bottleneck. As Russell and Fry (1997) put it: "The impact of lot splitting is yet to be studied."

The research into lot streaming has shed some light toward answering the important question of how the process of streaming lots throughout the production system should be done. Most of the work has focused on flow shops consisting of two machines, with the objective of minimizing makespan (i.e., the total flow time of the batch within the production system). In some cases, the procedures developed for two machines have been extended to three machines. Discrete solution procedures have been developed only for a small subset of special cases of two and three machines. Trietsch and Baker (1993) conclude that the next step for research is to develop practical solutions for problems where $m>3$. They also suggest that different objectives (such as minimizing WIP) should be considered as well. Vickson and Alfredsson (1992) also stress the need to consider relationships between various schedule-related quantities (such relationships have been established only in the absence of transfer batches). This implies that there is a need to consider sublots from an economical standpoint, and not merely from an operational standpoint. These issues give rise to many different scenarios that are yet to be investigated. Such scenarios may be generated by relaxing at least one of the following limitations of existing research on single batch models:

- The number of machines considered
- The number of sublots considered (either known in advance or restricted by the model)
- The objective function considered (makespan, in the vast majority of the cases)
- The assumption that setup times and/or transfer times are negligible or can be absorbed in the processing times
- The integrality of the solution (continuous and not discrete)

Cases of multiple batch models are limited by the same reasons. In this work, we extend existing results, for both single batch models as well as multiple batch models, by relaxing one or more of the above limitations while assuming that the sublots are of equal size.

2.5.2 Flow shop scheduling and lot streaming

A primary objective of multiple batch models for lot streaming in flow shops is to determine an optimal sequence of the sublots on the machines. This problem can be viewed as a larger scale problem of lot sequencing in flow shops, by considering each sublot as an independent lot. Viewing the lot streaming problem in that manner suggests that results obtained in the area of flow shop sequencing and scheduling can be applied to the lot streaming problem. Some results available in the flow shop sequencing and scheduling literature have already been applied to the lot streaming problem of multiple batches. However, such applications are rather limited since even the classic flow shop sequencing problem is difficult-to-solve when more than three machines are considered. For such cases, several heuristics have been presented.

When trying to apply these heuristics on the lot streaming problem, one issue that comes to mind is that most of the heuristics construct the sequence by comparing lot-based measures, e.g. the slope index, thus if a certain lot should be placed before another lot based on that measure then, every sublot of that lot should precede every sublot of the other lots. In other words, such measures imply that the best sequence is constructed of an unbroken string of sublots of the same lot. However, this is clearly sub-optimal sequence since it may be beneficial to intermingle the sublots. The CDS heuristic which is also presented in section 2.3.1 does not fall into the category of lot-based heuristics but, when it is applied to a lot streaming problem, it too would result in a sequence of an unbroken string of sublots of the same lot. The only exception is the fast insertion heuristic (FIH/NEH), which does not necessarily result in non-intermingled sequence since when to a lot streaming problem, it would consider every sublot separately for the best position in the partial sequence. The fast insertion heuristic, therefore, possesses two important qualities:

- It is the current champion in solving flow shop sequencing problems
- It is not limited to non-intermingled sequences

These qualities make it a good candidate for further implementation on the multiple batch lot streaming problem. In the proposed research, the FIH/NEH heuristic is extended to the lot streaming and compared with another heuristic. It is also utilized to provide useful insights into the impact of setup times on the sequence.

2.5.3 Justification for the use of equal sublots in lot streaming

The literature on lot streaming can be divided into two classes, based on the assumption made with respect to the sublot sizes. One class of models deals with equal sublots while the other class deals with unequal sublots. There is strong evidence to support that, in many cases, equal sublots do result in a near-optimal solution. The assumption of equal sublots is usually made to simplify the resultant model and to provide additional insights into the problem characteristics, but a practical reason for making that assumption is also commonly mentioned. The reason is that equal sublots are easier to manage, track, and control. In the proposed research, we investigate the lot streaming problem, consisting of n lots, m machines, unknown number of sublots and undetermined sequence, under various objective functions. To do so, we make the simplifying assumption of equal sublots. Therefore, it is appropriate to provide reasonable support for making that assumption. A list of arguments given in the literature for the use of equal sublots in lot streaming is summarized below:

- Kropp and Smunt (1990) provide two important practical reasons to support the use of equal sublots:
    1. A standardized container size is more convenient, and the container is always fully utilized.
    2. The use of equal sublots would likely reduce the number of implementation errors.
- Eynan and Li-Chung (1997) consider the objective of maximizing the net present value of the total revenue collected at the delivery of the sublots. They show that the “convenient” operational approach of equal sublot sizes is nearly optimal.
- Kropp and Smunt (1990) observe that, as the setup times increase and *n*>2, sublots of equal sizes (except the first sublot) tend to be optimal.
- Baker and Jia (1993) present results concerning the impact of equal and consistent sublots in a three-machine flow shop problem. Their results indicate the following: As the number of sublots increases, the ratio $\frac{M^E}{M^C}$ approaches 1.00, justifying the use of equal sublots.
- Potts and Baker (1989) argue that a justification for using equal sublots, in the case of multiple lots, is that a hierarchical approach of sequencing lots and then streaming them does not guarantee optimality.
- Wagner and Ragatz (1994) investigate the impact of equal sublots, via simulation, on due date performance in a five-machine open job shop problem. Their conclusion is that lot splitting in the presence of equal sublots provides improvements that are essentially independent of the setup time, as long as the utilization (i.e., the ratio: $\frac{\text{Setup Time + Processing Time}}{\text{Available Time}}$) is constant.

Chapter 3: Single Batch Models With Equal Sublots

3.1 Scope

In this Chapter, the impact of dividing a single batch into equal sublots on various objective functions, in a general m-machine flow shop, is investigated. The key issue is the determination of the number of sublots. Extreme cases are analyzed to demonstrate the extent of the potential benefits of using lot streaming with equal sublots. The effects of non-negligible transfer and setup times on these benefits are also analyzed. Transfer and setup times are primarily differentiated by the fact that a setup time is always incurred on a machine when a sublot is to be processed while a transfer time, from one machine to another, may or may not be incurred on a machine, depending upon whether the machine can be operational during transfer or not. Note that both the transfer times and the setup times cannot be added to the processing times of a sublot as the sublot size is yet to be determined.

For various cases of non-negligible transfer times, closed-form formulae are obtained to compute the optimal sublot sizes with respect to both operational and cost-based objective functions. For the case of non-negligible setup times, an algorithm of complexity $O(m)$ computations is developed to obtain the optimal number of sublots for the makespan criterion. Modifications of this algorithm are made to adapt it for obtaining optimal solutions with respect to the other objective functions as well. These include the objectives of minimizing mean flow time and minimizing production cost. Finally, a unified Goal Programming (GP) approach is utilized to solve for the optimal sublot size with respect to a general objective function that is comprised of the following terms: (a) makespan, (b) mean flow time, (c) average work-in-process, (d) setup, and (e) transfer time.

3.2 Preliminary Analysis of Objective Functions and Potential Benefits

In this Section, three operational objective functions are considered. These include makespan, mean flow time, and average WIP. An expression for the determination of optimal sublot size (for the case of equal sublots) is derived and is shown to be optimal, under negligible setup and transfer times.

The following notation will be utilized in this Chapter:

Q	- The lot size
n	- The number of sublots streamed through the flow shop
m	- The number of machines in the flow shop ; $j = 1,..,m$
p_j	- The processing time of a single item on machine j ; $p_{\max} \equiv \max_{1 \le j \le m}\left\{p_j\right\}$
L	- The sublot size
TT	- The transfer time per sublot
su_j	- The setup time prior to the processing of a sublot on machine j
M	- The makespan
$M_j(n)$	- The workload on machine j (defined separately.)
MFT	- The mean flow time
WIP	- The average WIP level
c_k	- The weight, or the cost per unit time, associated with component k

A superscript '*LS*' is added to an objective, such as the makespan, to denote that the analysis applies to the lot streaming case. A superscript '*' is added to denote optimality.

All the results developed in this Chapter are based on the following assumptions:

- The items within the lot are identical
- The machines are available for continuous processing.
- Preemption is not allowed, i.e., once the processing of a sublot has begun it cannot be stopped.
- The setup times and the unit processing times are deterministic and known in advance.
- The machines cannot process more than one sublot at any time.

3.2.1 Makespan objective

In the simplest case of equal sublots, when both setup times and transfer times are assumed to be negligible, we have: (see section 2.4.2.3-1)

$$M = \frac{Q}{n}\left[\sum_{j=1}^{m} p_j + \left((n-1)p_{\max}\right)\right] = Q \cdot p_{\max} + \frac{Q \cdot \left(\sum_{j=1}^{m} p_j - p_{\max}\right)}{n} \tag{3.2.1-1}$$

Therefore, the optimal continuous solution is:

$$n \to \infty$$
$$M = Q \cdot p_{\max}$$

This solution is clearly infeasible since the lot is not infinitely divisible. The discrete (and feasible) solution is, therefore, to have the maximum amount of integer sublots possible, i.e., Q sublots of unit size:

$$\begin{aligned} n^* &= \max\{n\} = Q \\ M^* &= (Q-1) \cdot p_{\max} + \sum_{j=1}^{m} p_j \end{aligned} \tag{3.2.1-2}$$

Hence, as Kanban suggests, it is best to use unit-sized sublots if setup and transfer times are negligible. To assess the possible reduction in the makespan via lot streaming, note that without the use of lot streaming the makespan for the lot would be:

$$M = Q \cdot \sum_{j=1}^{m} p_j \tag{3.2.1-3}$$

Denoting by M^{LS} the makespan under lot streaming, given by (3.2.1-2), the ratio of the two makespans is as follows:

$$\frac{M^{LS}}{M} = \frac{(Q-1) \cdot p_{\max} + \sum_{j=1}^{m} p_j}{Q \cdot \sum_{j=1}^{m} p_j} \tag{3.2.1-4}$$

Two extreme cases are of interest. First, consider the (less realistic) case of a small batch size, processed on a large number of machines, such that the following holds:

$$(Q-1) \cdot p_{\max} << \sum_{j=1}^{m} p_j \tag{3.2.1-5a}$$

i.e., the total processing time per item is significantly larger than the product of the number of items and the maximum processing time per item. In that case, the ratio approaches the value $1/Q$. On the other hand, if Q is large, such that:

$$(Q-1) \cdot p_{\max} >> \sum_{j=1}^{m} p_j \tag{3.2.1-5b}$$

then the ratio approaches the value $\frac{p_{\max}}{\sum_{j=1}^{m} p_j}$, which will be denoted by *P*. Note that if

$p_j = p(= p_{\max}) \ \ \forall j$, then *P* admits its lowest possible value, which is: $P = 1/m$, i.e., the maximal makespan reduction (MMR) is as follows:

$$MMR = 1 - 1/m = \frac{m-1}{m} \tag{3.2.1-6}$$

Hence, the maximal makespan reduction is attained when all the processing times are equal and Q is sufficiently large. Under these circumstances, the makespan reduction depends purely on the number of machines according to eq. (3.2.1-6). Note that even for a relatively small number of machines, the reduction is substantial (e.g. for $m=5$, 80% reduction.)

We now develop similar results for two other operational measures, namely, mean flow time (MFT) and average work-in-process (WIP).

3.2.2 Mean flow time (MFT) objective

Since portions of the lot become available to the downstream machines before the last portion, the mean flow time of the entire lot under lot streaming is expected to be smaller than what it would be without the use of lot streaming. Let FT_k be the flow time of the k-th sublot ($k = 1,..,n$, where: $n=Q/L$, and L be the sublot size.) Assume that n is integer. Then,

$$FT_k = (k-1)\cdot L \cdot p_{\max} + L \cdot \sum_{j=1}^{m} p_j \quad \forall k = 1,..,n \tag{3.2.2-1}$$

Hence, for the mean flow time of the lot, we have:

$$MFT^{LS} = \frac{\sum_{k=1}^{n} FT_k}{n} \tag{3.2.2-2}$$

For the numerator we have:

$$FT \quad L \cdot \sum_{=} \left((\quad - \quad) \; p_{\max} + \sum_{=} p \right) = \left(p_{\max} \quad (k-1) + \sum_{=} p \right) = \\ Q \cdot \left(\frac{1}{\quad} \cdot p_{\max} + \sum_{=} p \right) \tag{3.2.2-3}$$

Substituting (3.2.2-3) into (3.2.2-2), we get:

$$MFT^{LS} = \frac{Q \cdot \left(\frac{n-1}{2} \cdot p_{\max} + \sum_{j=1}^{m} p_j \right)}{n} = \frac{Q \cdot p_{\max}}{2} + \frac{Q \cdot \left(\sum_{j=1}^{m} p_j - {p_{\max}}/{2} \right)}{n} \qquad (3.2.2\text{-}4)$$

Although, in this analysis, it was assumed that *n* is integer, exp. (3.2.2-4) is, in fact, valid for every *n* value (i.e., continuous as well as integer.) As in the makespan case, the optimal mean flow time is obtained for the largest *n* possible, i.e., when sublots of unit size are utilized. Thus, for optimality we have: $n^* = Q$. Substituting this in (3.2.2-4), we get:

$$MFT^{LS} = \frac{Q-1}{2} \cdot p_{\max} + \sum_{j=1}^{m} p_j \qquad (3.2.2\text{-}5)$$

On the other hand, when lot streaming is not utilized, the mean flow time is simply $Q \cdot \sum_{j=1}^{m} p_j$ and therefore, the ratio of the two is as follows:

$$\frac{MFT^{LS}}{MFT} = \frac{\frac{Q-1}{2} \cdot p_{\max} + \sum_{j=1}^{m} p_j}{Q \cdot \sum_{j=1}^{m} p_j} \qquad (3.2.2\text{-}6)$$

For the same two extreme cases considered earlier for the makespan criterion (3.2.1-5a, b), the MFT ratio results in the same value for the case when lot size is relatively small while the number of machines is relatively large. For the case when lot size is very large, the ratio approaches the value *P/2* (half the value obtained under the makespan criterion), which, under perfect balance, becomes $\frac{1}{2 \cdot m}$.

3.2.3 Average work-in-process (WIP) objective

Similar to the case of mean flow time, it is expected that the average WIP would be smaller under lot streaming since portions of the original lot leave the floor earlier and the WIP keeps on decreasing after the completion of each sublot.

Obviously, the optimal WIP is met, in this case as well, when the sublots are of unit size. To see this, note that, the average WIP under lot streaming reduces in time. During the time $\left[0, L \cdot \sum_{j=1}^{m} p_j\right]$, the WIP is of Q units because the entire lot is yet to be completed. At time $L \cdot \sum_{j=1}^{m} p_j$, the first sublot is completed (and leaves the shop), thereby reducing the WIP level to (Q-L). From this point on, the WIP is reduced by one sublot every $L \cdot p_{\max}$ time units. Therefore, the average WIP under lot streaming is as follows:

$$WIP^{LS} = \frac{Q \cdot L \cdot \sum_{j=1}^{m} p_j + L \cdot p_{\max}\left[(Q-L) + (Q - 2 \cdot L) + \ldots + L\right]}{L \cdot \sum_{j=1}^{m} p_j + (n-1) \cdot L \cdot p_{\max}} \qquad (3.2.3\text{-}1)$$

The numerator in exp. (3.2.3-1) represents the area lying under the WIP level as a function of time. The denominator in exp. (3.2.3-1) represents the total time period over which the WIP level changes. Exp. (3.2.3-1) can be simplified as follows:

$$WIP^{LS} = \frac{L \cdot \left\{Q \cdot \sum_{j=1}^{m} p_j + p_{\max}\left[L \cdot \left(\left(\frac{Q}{L} - 1\right) + \left(\frac{Q}{L} - 2\right) + \ldots + 1\right)\right]\right\}}{L \cdot \left[\sum_{j=1}^{m} p_j + (n-1) \cdot p_{\max}\right]} =$$

$$= \frac{Q \cdot \sum_{j=1}^{m} p_j + L \cdot p_{\max} \cdot \left((n-1) + (n-2) + \ldots + 1\right)}{\sum_{j=1}^{m} p_j + (n-1) \cdot p_{\max}} = \frac{Q \cdot \sum_{j=1}^{m} p_j + L \cdot p_{\max} \cdot \frac{n \cdot (n-1)}{2}}{\sum_{j=1}^{m} p_j + (n-1) \cdot p_{\max}}$$

which leads to:

$$WIP^{LS} = Q \cdot \frac{\left(\sum_{j=1}^{m} p_j + \frac{(n-1)}{2} \cdot p_{\max}\right)}{\sum_{j=1}^{m} p_j + (n-1) \cdot p_{\max}} \qquad (3.2.3\text{-}2)$$

From exp. (3.2.3-2), it is clear that the minimum WIP is attained for the largest possible n, i.e., $n^* = Q$. The average WIP without lot streaming is simply Q units (since the entire lot stays in the system.) Hence, the ratio of the two is as follows:

$$\frac{WIP^{LS}}{WIP} = \frac{\sum_{j=1}^{m} p_j + \frac{Q-1}{2} \cdot p_{\max}}{\sum_{j=1}^{m} p_j + (Q-1) \cdot p_{\max}} \tag{3.2.3-3}$$

Note that the above ratio approaches 1.00 under the extreme condition of (3.2.1-5a), thereby indicating that no reduction in the average WIP is expected via lot streaming. This is indeed anticipated. If the lot size is relatively small while the processing times on the machines are relatively large, the time difference between the completion of the first sublot and the completion of the last sublot becomes insignificant with respect to reducing the average WIP. Thus, streaming sublots throughout the system would not have a significant impact on the average WIP. In other words, in this case, the sublots spend almost the same amount of time in the shop as if they were produced as a single unbroken lot.

For the condition specified by (3.2.1-5b), the ratio reduces to *1/2*. This implies that the maximal reduction in the average WIP level under lot streaming is 50%.

3.2.4 Summary of the benefits via lot streaming

Table 3.1 presents a summary of the above findings. These pertain to the extent of the benefits via lot streaming. The limits on the benefits with respect to each of the three performance measures, namely, the makespan, the mean flow time, and the average work-in-process are provided in terms of the ratio of the values of the performance measures obtained with lot streaming to these obtained without lot streaming.

Table 3.1. Upper bounds on the benefits via lot streaming

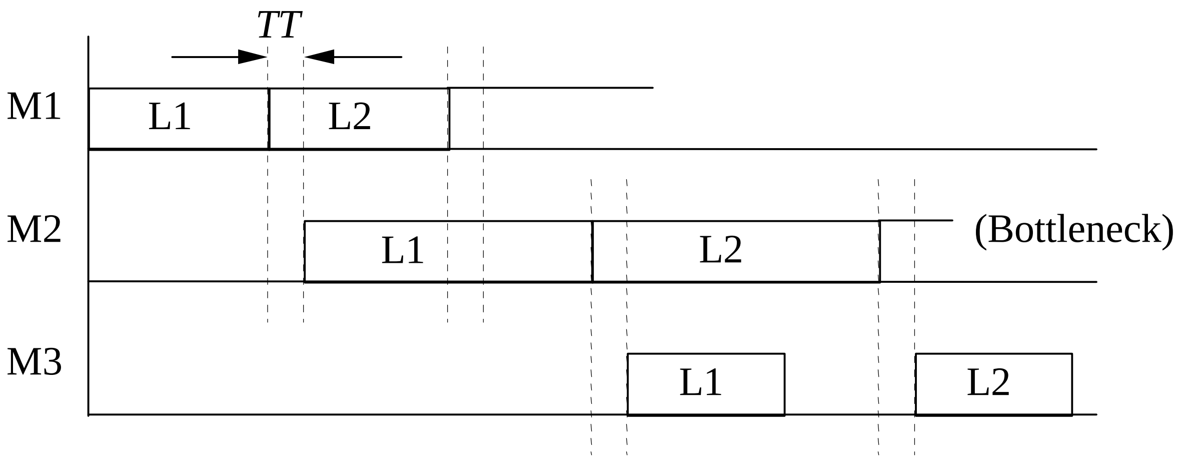
Measure	General ratio	If $(Q-1)\cdot p_{\max} >> \sum_{j=1}^{m} p_j$	If $(Q-1)\cdot p_{\max} << \sum_{j=1}^{m} p_j$
Makespan	$\dfrac{(Q-1)\cdot p_{\max} + \sum_{j=1}^{m} p_j}{Q\cdot \sum_{j=1}^{m} p_j}$	$\dfrac{p_{\max}}{\sum_{j=1}^{m} p_j}$	$1/Q$
Mean flow time	$\dfrac{\frac{Q-1}{2}\cdot p_{\max} + \sum_{j=1}^{m} p_j}{Q\cdot \sum_{j=1}^{m} p_j}$	$\dfrac{p_{\max}}{2\cdot \sum_{j=1}^{m} p_j}$	$1/Q$
Work-in-process	$\dfrac{\sum_{j=1}^{m} p_j + \frac{Q-1}{2}\cdot p_{\max}}{\sum_{j=1}^{m} p_j + (Q-1)\cdot p_{\max}}$	$1/2$	1

3.3 The impact of Transfer on the Objective Function

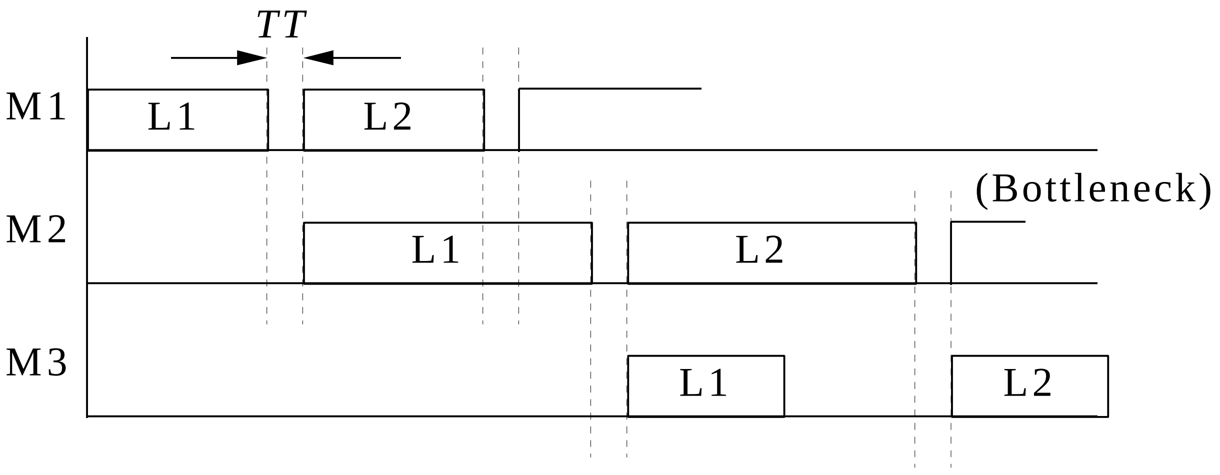
It was assumed in the above analysis that the transfer time is either negligible or can be absorbed in the processing time. In this Section, we relax this assumption and determine optimal number of sublots for the cases of minimizing makespan and a cost-based objective. Several variations of transfer are considered:

- Sublot size dependent transfer versus sublot size independent transfer
- Linear transfer dependency versus stepwise transfer dependency
- Operator-based transfer (i.e., production is halted while transfer is performed) versus non-operator-based transfer (i.e., transfer is performed along with production)

In **Fig. 3.1**, the last two forms of transfer are illustrated using a three-machine flow shop.



(a) Non-operator-based transfer (production not halted during transfer)



(b) Operator-based transfer (production halted during transfer)

Fig. 3.1. Non-operator-based and operator-based transfer scenarios

Fig. 3.1(a) depicts the case in which the machines continue processing during transfers whereas Fig. 3.1(b) depicts the case in which production is halted upon completion of a sublot to allow for the transfer of that sublot to the next machine. The assumption that the resources are operational during transfer is appropriate in environments where dedicated resources are assigned to transferring parts between stations (Trietsch, 1989). In manual flow shops, on the other hand, the transfer is typically done by the operators.

3.3.1 Independent non-operator-based transfer

We begin with the case in which the transfer is independent of the sublot size, and it is non- operator-based. Then, for the makespan objective, we have:

$$M = \frac{Q}{n} \cdot \sum p_j + (n-1) \cdot \frac{Q}{n} \cdot p_{\max} + (m-1) \cdot TT \qquad (3.3.1\text{-}1)$$

The expression states that the transfer time has no effect on the makespan except for a constant contribution due to the transfer of the first sublot throughout the system. It is assumed in exp. (3.3.1-1) that the transfer operation is not the bottleneck operation, i.e., that the following holds:

$$\frac{Q}{n} \cdot p_{\max} \geq TT$$

Under this condition, the optimal sublots are of unit size, similar to the case of negligible transfer time (eq. 3.2.1-2).

3.3.2 Independent operator-based transfer

Assuming the production is halted during the transfer of completed sublots to successive machines, we have:

$$M = \frac{Q}{n} \cdot \left(\sum p_j + (n-1) \cdot p_{\max}\right) + (m-1) \cdot TT + (n-1) \cdot TT \qquad (3.3.2\text{-}1)$$

The first and the second components of this expression are the same as in eq. (3.3.1-1). The last component, however, is proportional to the number of sublots (with the exception of the first sublot which has already been accounted for in the second component of the expression.)

Letting: $\frac{dM}{dn} = -\frac{Q}{n^2} \cdot \left(\sum p_j - p_{\max}\right) + TT = 0$, leads to:

$$n^* = \sqrt{\frac{Q \cdot \left(\sum p_j - p_{\max}\right)}{TT}} \qquad (3.3.2\text{-}2)$$

Since:

$$\frac{d^2M}{dn^2} > 0$$

M is a strictly convex function of *n*, and hence, the value of n^*, given by (3.3.2-2), is minimum. Moreover, the optimal discrete solution is either the rounded up value or the rounded down value of n^*. The general case of operator-based transfer times, that are different for any two consecutive machines, is similar to the case of sublot-attached setup times, which is analyzed thoroughly in Section 3.4.

3.3.3 Dependent (non-operator-based) transfer

Relaxing the assumption that the transfer time is independent of the sublot size, two alternative assumptions on the type of transfer time dependency are examined:

1. TT is a linear function of the sublot size $\left(\frac{Q}{n}\right)$. That is, $TT = \frac{Q}{n} \cdot TT_1$, where TT_1 is the transfer time of a single item.
2. TT is a step function, i.e., transfer of *r* parts or less takes TT_2, transfer of more than *r* parts but no more than *2r* parts takes *2TT*₂, and so forth. Thus, $TT = \left[\frac{Q}{n \cdot r}\right]^+ \cdot TT_2$, where $[\bullet]^+$ is the rounded up integer of the expression within.

Note that, case (2) is the discrete version of case (1), where TT_1 replaces $\frac{TT_2}{r}$. Case (1) can be easily solved to obtain the optimal number of sublots following the previous development. Substitute $TT = \frac{Q}{n} \cdot TT_1$ in expression (3.3.1-1), to get:

$$M = \frac{Q}{n} \cdot \left[\left(\sum p_j + (n-1) \cdot p_{\max} + (m-1) \cdot TT_1\right)\right] \qquad (3.3.3\text{-}1)$$

The optimal solution is specified by expression (3.3.1-2) i.e., sublots of unit size. This is also the optimal solution to case (2), since it is an integer solution. This finding should be intuitive since, in this case, the transfer time does not affect the makespan except for a constant contribution due to the first sublot. Thus, a sublot of unit size would, obviously, result in minimum transfer time as it is the smallest possible. In addition, as shown earlier, unit-sized sublots are also optimal with respect to the makespan when the transfer time is negligible. Combining these two observations leads to the conclusion that unit-sized sublots must be optimal in this case.

3.3.4 The impact of transfer time on a cost-based objective

Results identical to the above can be obtained when considering an economical (cost-based) objective function. Consider the following objective, used by Doutriaux and Sarin (1996):

$$Z = u \cdot M + h \cdot T \tag{3.3.4-1}$$

where:

M is the makespan, *T* is the total transfer time, and *u*, *h* are the respective costs per unit makespan time and per unit transfer time. In this case, an assumption on the operationality of the machines during transfer is not required since the expression for the total transfer time, *T*, is the same regardless of the machines' status during transfers. Therefore, we have:

$$T = n \cdot (m-1) \cdot TT \tag{3.3.4-2}$$

Using exp. (3.3.1-1) for the makespan, in the case of independent non-operator-based transfer, we get:

$$Z = u \cdot \left[\frac{Q}{n} \cdot \left(\sum p_j + (n-1) \cdot p_{\max} \right) + (m-1) \cdot TT \right] + h \cdot \left[n \cdot (m-1) \cdot TT \right] \tag{3.3.4-3}$$

Which leads to:

$$n^* = \sqrt{\frac{u \cdot Q \cdot \left(\sum p_j - p_{\max} \right)}{h \cdot TT \cdot (m-1)}} \tag{3.3.4-4}$$

The second derivative is strictly positive for every *n*, therefore, the rounded down solution or the rounded up solution, whichever results in a smaller Z-value, is the optimal discrete solution. In the same manner, results can be obtained for the other cases as well, for the economical objective function considered in (3.3.4-1).

3.4 The impact of Setup on the Makespan Function

3.4.1 Analysis of the problem

In this Section, the case of a single batch split into sublots with attached setups is considered. Setups can be attached or detached, and they can be sequence dependent or sequence independent. When the setup operation must be performed immediately prior to the beginning of processing, it is said that the setup is "attached." Conversely, when the setup operation can be performed, regardless of the presence of work, it is said that the setup is "detached." We analyze the more complicated (and realistic) case of attached setups. Note that for the case of detached setups, or alternatively, the case of lot-attached setups, the optimal solution is simply to have unit-sized sublots, as in the case of no setup, since the setup is merely a constant (unavoidable) contribution to the makespan. Therefore, we consider the more general case of sublot-attached setup.

When the setup time of an operation to be performed is dependent on the previous operation performed, it is said that the setup is sequence dependent. In the case of a single batch the previous operation is the same as the next one. Therefore, it is not relevant to discuss sequence dependency. Moreover, in most of the analytical work on the multiple batch lot streaming problem in the literature, it has been assumed that the setups are sequence independent. Sequence dependency leads to an embedded traveling salesman sub-problem and is therefore not considered herein.

Analysis of the impact of sublot-attached setup times in a single batch environment can help gain insight into the impact of lot splitting in a multiple batch environment. Algorithms developed here can later be used hierarchically, or in another fashion, for the multiple batch problem with setups. In addition, setups that are attached to every sublot occur in many practical situations. For example, in semi-conductor coating and polishing processes there is a fixed setup prior to the processing of the next sublot. Also, cleaning operations between machining and other metal cutting processes can be considered as fixed setup, performed before the next transfer batch is processed. Lastly, as mentioned in the previous section, operator-based and machine-based transfer times can also be handled as sublot-attached setup times. Thus, the problem is relevant even without considering its role in the multiple batch problem.

Before we analyze the problem of determining the optimal number of sublots under sublot-attached setup, it is noted that, in this case, there exists a trade-off between the time spent on setup (i.e., unproductive time) and the saved by splitting lots. The larger the number of sublots, the smaller is the makespan value, but, the greater becomes the time spent in setups which increases makespan. Hence, there must exist an optimal number of sublots which minimize the makespan.

Fig. 3.2 depicts a three-machine flow shop in which the machines require different setup times before the start of the processing of each sublot. As can be observed, the bottleneck machine is not necessarily the machine with the largest processing time, $p_{\max}$ (machine 2 in Fig. 3.2) but rather the machine which maximizes the expression:

$$\frac{Q}{n} \cdot p_j + su_j \tag{3.4-1}$$

where su_j is the setup time on machine j. In Fig. 3.2, it is machine 1 that maximizes the above expression.

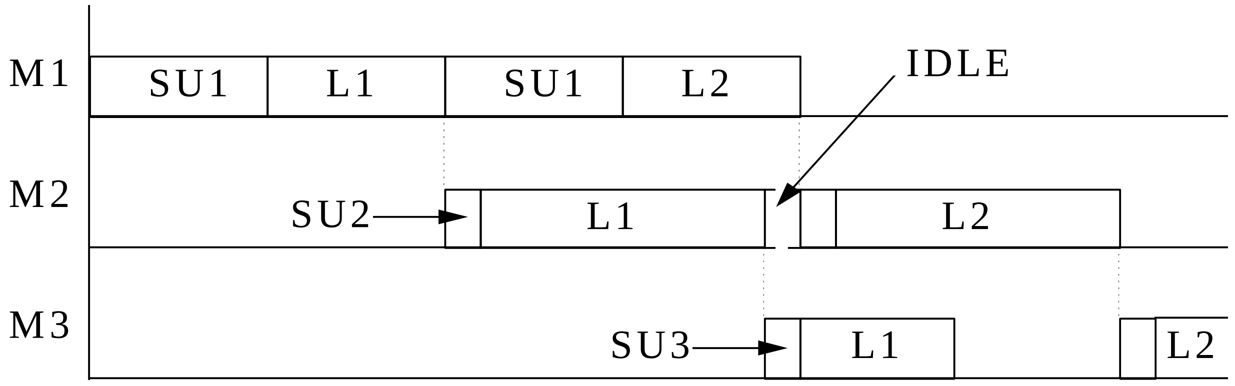


Fig. 3.2. Attached setups in a three-machine flow shop

Note that exp. (3.4-1) is dependent on the value of *n*, which is unknown. Thus, for the makespan we have:

$$M(n)=\left[\frac{Q}{n}\cdot\sum_{k=1}^{m}p_k+\sum_{k=1}^{m}su_k\right]+(n-1)\cdot\max_{1\le j\le m}\left\{\frac{Q}{n}\cdot p_j+su_j\right\} \tag{3.4-2}$$

The second part of each term of exp. (3.4-2) represents addition to the makespan value due to setup which affects the optimal solution. Note that, since the first part of the expression is independent of the machine type (i.e., same for all j=1,..,m), the maximization operand can be taken on the entire expression as follows:

$$M(n)=\max_{1\le j\le m}\left\{\left[\frac{Q}{n}\cdot\sum_{k=1}^{m}p_k+\sum_{k=1}^{m}su_k\right]+(n-1)\cdot\left(\frac{Q}{n}\cdot p_j+su_j\right)\right\} \tag{3.4-3}$$

Another way to view exp. (3.4-3) is as follows. The first term accounts for the processing of a single sublot across all the machines while the second term accounts for the total processing time of the lot (except the single sublot that had already been accounted for) on the bottleneck machine. The maximum of the expression, considering each machine as a candidate for being bottleneck, gives the makespan. Let us define the expression within the brackets as $M_j(n)$, being the workload of machine j. Thus, we have:

$$M_j(n)=\left[\frac{Q}{n}\cdot\sum_{k=1}^{m}p_k+\sum_{k=1}^{m}su_k\right]+(n-1)\cdot\left(\frac{Q}{n}\cdot p_j+su_j\right) \tag{3.4-4}$$

$$M(n)=\max_{1\le j\le m}\left\{M_j(n)\right\} \tag{3.4-5}$$

Thus, the problem can be formulated as a non-linear integer programming problem:

$$\begin{aligned}&\min\{M\}\\ &s.t.\quad M\ge M_j(n)\\ &\qquad\quad \mathrm{Q}\ge\ n\ge 1,\ \text{int.}\end{aligned}$$

Note that the linear programming (LP) formulation, given in the literature survey (2.4.2.4-11) cannot be utilized to solve the single batch m-machine flow shop problem with sublot-attached setups. The reasoning is as follows. To utilize the LP formulation, the number of sublots must be given in advance which is not true in our case and, in fact, is part of the decision process.

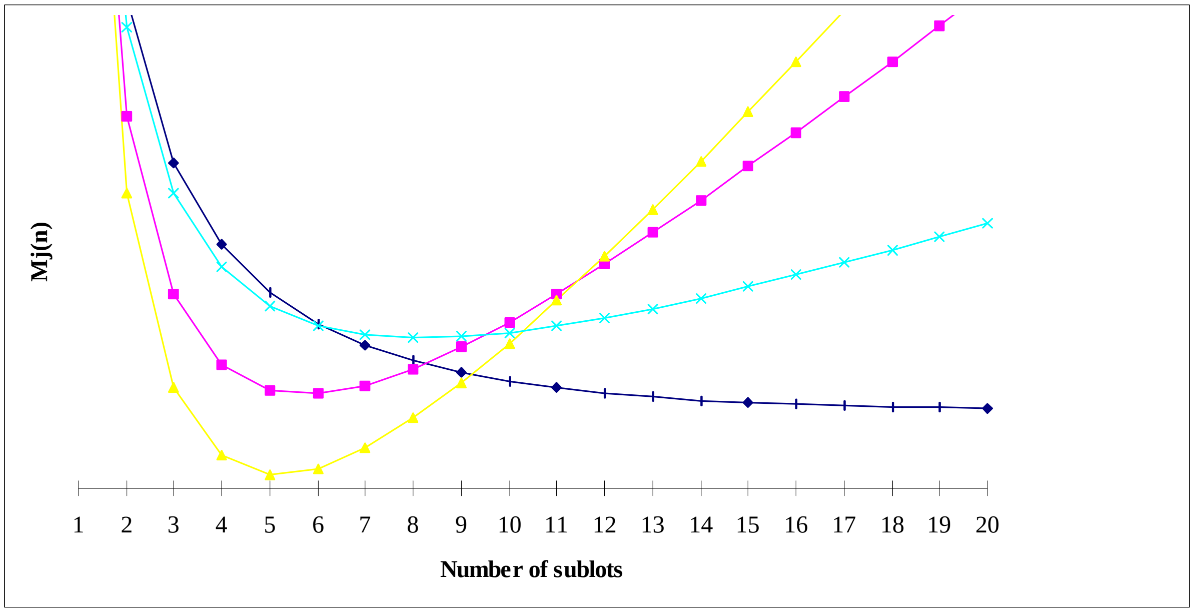


Fig. 3.3. Typical $M_j(n)$ functions as a function of the number of sublots.

Note that, for *n*=1, we have:

$$M(1) = M_j(1) = Q \cdot \sum_{k=1}^{m} p_k + \sum_{k=1}^{m} su_k \quad \forall 1 \le j \le m$$

A plot of $M_j(n)$ for different j is shown in **Fig. 3.3**. Note that $M_j(1)$ is the same for $\forall j$. As will be shown later, $M_j(n)$ are strictly convex functions of *n*. However, they differ in their curvatures, as shown in Fig. 3.3. The upper envelope formed by the $M_j(n)$ functions is the $M(n)$, which we wish to minimize.

Next, we derive certain properties of the makespan objective that are utilized later to develop an efficient algorithm for obtaining the optimal sublot sizes for the general *m*-machine single batch flow shop case with equal sublots.

Theorem 3.1. $M_j(n),\ \forall j,$ *are strictly convex functions.*

Proof.

$M_j(n)$ is twice differentiable, and the second derivative, with respect to *n*, is always positive:

$$\frac{d^2 M_j(n)}{dn^2} = \frac{2 \cdot Q \cdot \left(\sum p_j - p_j\right)}{n^3} > 0 \qquad \forall n$$

Therefore, $M_j(n),\ \forall j,$ are strictly convex functions. <u>QED.</u>

Theorem 3.2. M(n) is a strictly convex function.

Proof.

The maximum function of strictly convex functions is a strictly convex function as well (for proof see Bazaraa et al., 1993). Since $M_j(n)$ are all strictly convex, it follows that *M(n)* is also strictly convex since: $M(n) = \max\limits_{1 \le j \le m} \left\{ M_j(n) \right\}$. <u>QED.</u>

Corollary 3.1. M(n) has a unique global optimal solution.

Corollary 3.2. At points for which M(n) is differentiable, there exists at most one point, n^*, *at which the first derivative satisfies:*

$$\frac{dM(n^*)}{dn} = 0\,.$$

Proof.

More than one point contradicts the fact that *M(n)* is <u>strictly</u> convex. <u>QED.</u>

The next theorem helps identify the optimal solution. The optimal solution can either coincide with the local minima of some $M_q(n)$ or with an intersection point formed by some two $M_j(n)$ functions. Theorem 3.3 gives the necessary conditions under which the optimal solution coincides with the local minima of $M_q(n)$ or the intersection point of two $M_j(n)$ functions.

Theorem 3.3. For any q=1,..,m, let n_q^* *be the solution to the equation* $\frac{dM_q(n_q^*)}{dn} = 0$. *If there exists a q, such that q is the maximizing index* of $\left\{\frac{Q}{n_q^*} \cdot p_j + su_j\right\}_{j=1}^{m}$, *then* n_q^* *is the global optimum of M(n).*

Otherwise, if no such q exists, the optimum must occur at an intersection point, n_{kl}, *for which* $M_k(n_{kl}) = M_l(n_{kl})$

Proof

If n_q^* maximizes $\left\{\frac{Q}{n_q^*} \cdot p_j + su_j\right\}_{j=1}^{m}$ for some *q*, then from (3.4-2) and (3.4-4) we have:

$$M(n_q^*) = \max_{1 \le j \le m} \left\{M_j(n^*)\right\} = M_q(n_q^*).$$

Since:

$$\frac{dM_q(n_q^*)}{dn} = 0$$

It follows that $M_q(n) \ge M_q(n_q^*)\ \ \forall n$.

By definition we know that: $M(n) \ge M_q(n) \ge M_q(n_q^*) = M(n_q^*)\ \ \forall n$, i.e., n_q^* is the optimal solution of $M(n)$.

This completes the proof of the first part of the theorem. Next, we show by contradiction that, if no such *q* exists, the optimum must lie at an intersection point.

Suppose the optimum occurs at a non-intersection point, say n_1. The derivative of *M(n)* is defined at all points other than the intersection points. Therefore it is defined at n_1. Since the first part of theorem 3.3 is not satisfied, there is no point among these non-intersection points at which the derivative of *M(n)* equals zero. In particular, $\frac{dM(n_1)}{dn} \ne 0$. Since *M(n)* is strictly convex, either to the right or to the left of n_1, there exists a better solution. Thus, the optimum could not have occurred at the non-intersection point n_1. QED.

The next two corollaries provide the values of n_q^* and n_{kl}.

Corollary 3.3. n_q^* can be obtained using the following expression:

$$n_q^* = \sqrt{\frac{Q \cdot \left(\sum_{j=1}^{m} p_j - p_k \right)}{su_q}} \tag{3.4-6}$$

(This is simply the result of setting the first derivative of $M_q(n)$, given by (3.4-4), to zero.)

Corollary 3.4. The intersection point, n_{kl}, can be obtained using the following expression:

$$n_{kl} = \frac{Q \cdot (p_k - p_l)}{su_l - su_k} \tag{3.4-7}$$

Proof

$$M_k(n_{kl}) = M_l(n_{kl})$$
$$\Leftrightarrow$$
$$\frac{Q}{n_{kl}} \cdot p_k + su_k = \frac{Q}{n_{kl}} \cdot p_l + su_l$$
$$n_{kl} = \frac{Q \cdot (p_k - p_l)}{su_l - su_k}$$

QED.

Theorem 3.4. "The Dominance Property":

If, for any machine pair {k, l}, the following holds:

$$su_k \geq su_l \quad \text{and} \quad p_k \geq p_l$$

then machine type l need not be considered for optimality.

Proof

If, for any *n*, we have: $su_k \geq su_l$ and $p_k \geq p_l$, then it follows that:

$$\frac{Q}{n} \cdot p_k + su_k \geq \frac{Q}{n} \cdot p_l + su_l$$

and therefore:

$$M_k(n) \geq M_l(n)$$

Thus, it suffices to consider only machine *k*. QED.

Let *S* denote the set of all machines and let $\bar{S}$ denote the reduced set of dominating machines.

Remark 3.1. For any (dominating) machine pair $\{k, l\} \in \bar{S}$*, it must be that:*

$$\{su_k > su_l \text{ and } p_k < p_l\} \text{ or: } \{su_k < su_l \text{ and } p_k > p_l\}$$

(The emphasis is on the strict inequality.)

Lemma 3.1. $M_j(n)$ *has a linear asymptote with a slope of* su_j *as n reaches infinity. That is:*

$$M_j(n) \xrightarrow[n\to\infty]{} su_j \cdot n$$

(An immediate result from exp. (3.4-4))

Theorem 3.5. If, for any (dominating) machine pair $\{k, l\} \in \bar{S}$ *, we have:* $su_l > su_k$*, then the following holds:*

$$M_k(n) > M_l(n) \quad \textit{for } 1 \leq n < n_{kl}$$

Proof

Both $M_l(n)$ and $M_k(n)$ are strict convex functions (Theorem 3.1). They intersect at $n = 1$ and at $n = n_{kl}$. These are the only two intersection points since there is no other solution to the equation:

$$M_l(n) = M_k(n)$$

Based on Lemma 3.1, we know that, for *n* sufficiently large, the function that has the higher su_j dominates the other. Thus, it must also be the case for all $n > n_{kl}$. Therefore, if $su_l > su_k$, we have: $M_l(n) > M_k(n)$ for all $n > n_{kl}$. It follows from the strict convexity of the functions that, for $n < n_{kl}$ (and $n \geq 1$), the inequality sign flips, and we have: $M_k(n) > M_l(n)$. QED.

Corollary 3.5. For the machine with the least setup time ($k = \arg\min_{j \in \overline{S}} \{su_j\}$ *), the following holds:*

$$M(n) = M_k(n) \text{ until its first intersection point.}$$

(Follows from theorem 3.5)

Remark 3.2. The machine with the least setup is also the machine that maximizes:

$$\left\{ \frac{Q}{n} \cdot p_j + su_j \right\}_{j=1}^{m}$$

from $n = 1$*, until its first intersection.*

(Follows from corollary 3.5, applied to the machine with the least setup.)

Corollary 3.6. Let $k = \arg\min_{j \in \overline{S}} \{su_j\}$. *The first intersection point,* n_{kl}*, for this k is such that:*

$$l = \arg\min_{j \in \overline{S}} \left\{ \frac{p_k - p_j}{su_j - su_k} \right\}$$

Proof.

Recall that: $n_{kl} = \dfrac{Q \cdot (p_k - p_l)}{su_l - su_k}$. Therefore, the smallest n_{kl} is attained for:

$$l = \arg\min_{j \in \overline{S}} \left\{ \frac{p_k - p_j}{su_j - su_k} \right\}.$$

QED.

Based on the above analysis, next, we propose an optimal solution algorithm for the problem on hand.

3.4.2 Optimal solution algorithm

Following is an algorithm which solves the continuous problem optimally. The algorithm searches for the optimal solution along the $M(n)$ function, starting from $n=1$, and checking for optimality along the first dominant $M_j(n)$ function, its first intersection point, the second dominant $M_j(n)$ function and its first intersection point, and so forth.

Algorithm (A1): Optimal solution algorithm for the makespan criterion

Step 1. Set $n_1 = 1$

Apply the Dominance Property ($su_k \geq su_l$ and $p_k \geq p_l$), to eliminate some of the candidate machines. Denote by $\overline{S}$ the candidate machine indices left after the elimination. Determine the least setup machine: $k = \arg\min\limits_{j \in \overline{S}} \{su_j\}$.

Step 2. Compute n_k^* by using the expression: $n_k^* = \sqrt{\dfrac{Q \cdot \left(\sum p_j - p_k\right)}{su_k}}$

Use $l = \arg\min\limits_{j \in \overline{S}} \left\{ \dfrac{p_k - p_j}{su_j - su_k} \right\}$ to find its first intersection point: $n_{kl} = \dfrac{Q \cdot (p_k - p_l)}{su_l - su_k}$

If $n_1 \leq n_k^* \leq n_{kl}$ then n_k^* optimal. STOP.

If $M(n_1) \leq M(n_{kl})$ then n_1 optimal. STOP.

If $\overline{S}$ is empty, STOP. Otherwise, continue to step 3.

Step 3. Set $n_1 = \min\{n_{kl}, Q\}$.

Remove k from $\overline{S}$.

Set $k \leftarrow l$.

Go to step 2.

3.4.3 Complexity of the algorithm

The complexity of the above algorithm is *O(m)* computations and $O\left(m^2\right)$ comparisons, a low polynomial order which does not depend on the number of sublots. Therefore, it is expected to find the optimal solution relatively fast. To realize the algorithm's order of complexity note that in the worst case, no machine is dominated in step 1, and steps 2, and 3 must be carried out fully. Step 2 checks a single point for optimality and makes at most (*m*-1) comparisons to find the intersection point. Then, it may also evaluate the objective function at the intersection point. Therefore, it is of order *O(1)* computations (or evaluations) and of order *O(m)* comparisons. Step 2 is repeated at most (*m*-1) times, thus the entire algorithm is of order *O(m)* computations and $O\left(m^2\right)$ comparisons.

3.4.4 Numerical example

Next, we illustrate the application of the above algorithm on a simple numerical example. Consider the following data: *m*=4 ; *Q*=100, and:

Table 3.2. Data for numerical example 3.1

j	1	2	3	4
p_j	2	3	1	2
su_j	12	5	10	10

Step 1. Machines 3, 4 are dominated by machine 1. Thus, *S*={1, 2}, with {2} having the least setup.

Step 2. We get: $n_2^* = 10.00$; $n_{12} = 14.28$.

Since $n_2^* < n_{12}$ $(n_2^* > 1)$ it is optimal, and the corresponding makespan is $M_2\left(n_2^*\right) = 432.0$

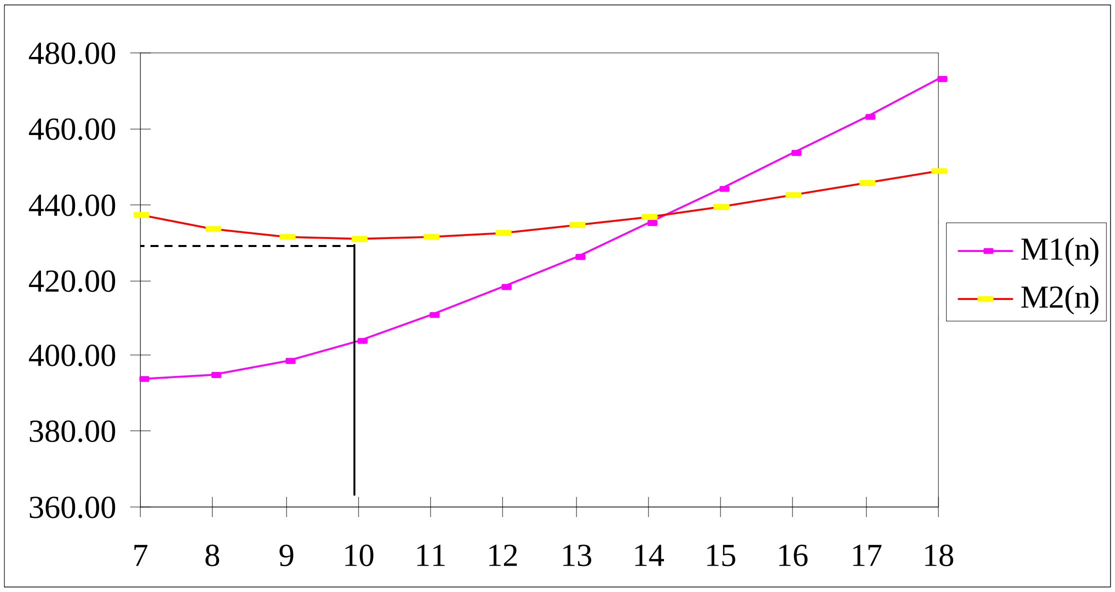


Fig. 3.4. The solution for the numerical example via step 2 of the algorithm.

The solution is depicted in **Fig. 3.4**. It is identified at the second step, and therefore, there is no need to go through step 3. To demonstrate what happens if an optimal solution does occur at an intersection point, suppose that the setup time of the first machine changes to: $su_1 = 20$, instead of 12 (any number above 15 will do.) In that case, $n_{12} = 6.67$, and we continue to step 3 as follows:

Step 3. Set $n_1 = 6.67$.

Set k={1}.

Remove {2} from $\overline{S}$.

Go to step 2.

Step 2. We get: $n_1^* = 5.47$;

Since $n_1^* < n_1$ it is infeasible.

Since $\overline{S}$ is empty, the last intersection point, $n_{12} = 6.67$, is optimal.

This solution is illustrated in **Fig. 3.5**.

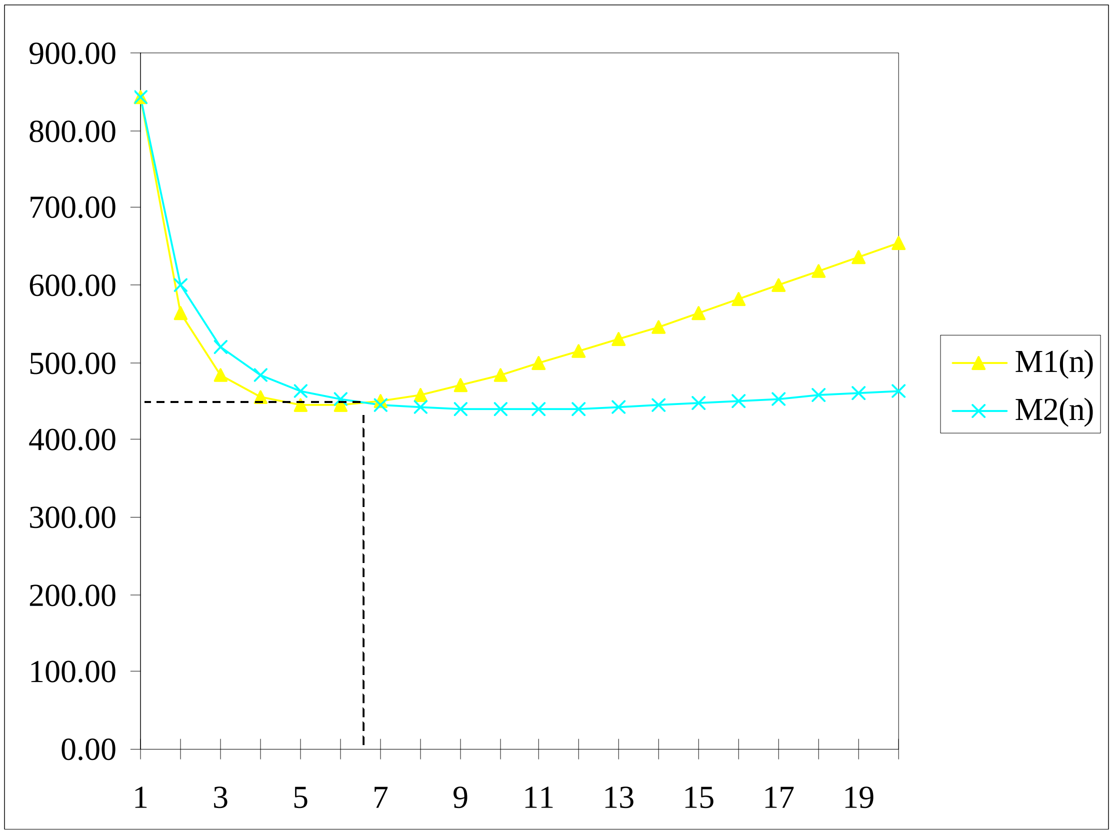


Fig. 3.5. The solution for the numerical example via step 3 of the algorithm.

3.4.5 Optimal integer solutions

Generally, the solution obtained via step 2 or 3 of the algorithm need not be integer, although it happens to be integer in the above example. Nevertheless, it is an easy task to obtain the integer optimal solution. Recall that the function is strictly convex and, therefore, the integer solution must either be the rounded down value or the rounded up value of the optimal continuous solution.

If the optimal solution is obtained via step 2 of the algorithm (i.e., n_k^*), and there are no intersection points in between this value and its rounded down value (i.e., $n_{kl} \notin \left(\lfloor n_k^* \rfloor, n_k^*\right)$) and between this value and its rounded up value (i.e., $n_{kl} \notin \left(n_k^*, \lceil n_k^* \rceil\right)$, then the optimal makespan is:

$$\min\left\{ M_k\left(\lfloor n_k^* \rfloor\right), M_k\left(\lceil n_k^* \rceil\right)\right\} \tag{3.4-8}$$

If, on the other hand, there exist intersection points in between, the rounded down and the rounded up values must be checked for each $M_l(n)$ (i.e., the makespan functions of the machines which constitute the intersection points.) The minimum of the maximum makespan values corresponding to the rounded down value and the maximum makespan values corresponding to the rounded up value is optimal. That is, the optimal makespan in this case is:

$$\min\left\{ \max_{\forall l:\, n_{kl} \in \left(\lfloor n_k^* \rfloor, n_k^*\right)} \left\{ M_l\left(\lfloor n_k^* \rfloor\right)\right\}, \max_{\forall l:\, n_{kl} \in \left(n_k^*, \lceil n_k^* \rceil\right)} \left\{ M_l\left(\lceil n_k^* \rceil\right)\right\}\right\} \tag{3.4-9}$$

If the optimal solution is obtained via step 3 of the algorithm (i.e., n_{kl}^*), then, if there are no intersection points in between this value and its rounded down value and between this value and its rounded up value, the optimal makespan is:

$$\min\left\{ M_k\left(\lfloor n_{kl}^* \rfloor\right), M_l\left(\lceil n_{kl}^* \rceil\right)\right\} \tag{3.4-10}$$

If, however, there exist intersection points in between, the rounded down and the rounded up values must be checked for each $M_g(n)$ similar to that in (3.4-9). The optimal makespan can thus be obtained via the following:

$$\min\left\{ \max_{\forall g:\, n_{kg} \in \left(\lfloor n_{kl}^* \rfloor, n_{kl}^*\right)} \left\{ M_g\left(\lfloor n_{kl}^* \rfloor\right)\right\}, \max_{\forall g:\, n_{gl} \in \left(n_{kl}^*, \lceil n_{kl}^* \rceil\right)} \left\{ M_g\left(\lceil n_{kl}^* \rceil\right)\right\}\right\} \tag{3.4-11}$$

Finally, we make a comment with regard to the optimal integer sublot size (as opposed to the number of sublots discussed thus far.) Let:

$\tilde{n}^*$ - The optimal integer number of sublots

$\tilde{L}^*$ - The optimal integer sublot size

Expressions (3.4-8) to (3.4-11) have provided the means to obtain the optimal makespan using the integer number of sublots, $\tilde{n}^*$. To obtain the optimal integer sublot size, $\tilde{L}^*$, the following procedure can be utilized:

1. Compute the optimal (continuous) sublot size as follows: $L^* = \dfrac{Q}{n^*}$
2. Obtain the integer sublot sizes of L^* (i.e., $\lfloor L^* \rfloor$, $\lceil L^* \rceil$)
3. Compute the *n*-values that correspond to the integer *L*-values, i.e., $n_1 = \dfrac{Q}{\lfloor L^* \rfloor}; n_2 = \dfrac{Q}{\lceil L^* \rceil}$
4. Evaluate these two *n*-values to determine which results in minimum makespan. The respective *L*-value for which the *n*-value results in minimum makespan is the optimal $\tilde{L}^*$.

The equation in step 1 of the procedure is true because of the following result.

Theorem 3.6 The optimal number of sublots, n^*, and the optimal sublot size, L^*, satisfy:

$$L^* = \frac{Q}{n^*}$$

Proof.

To see that this is true, re-write exp. (3.4-3) using *L* instead of *n*.

$$M(L) = \max_{1 \le j \le m} \left\{ \left[L \cdot \sum_{j=1}^{m} p_j + \sum_{j=1}^{m} su_j \right] + \left(\frac{Q}{L} - 1 \right) \cdot \left(L \cdot p_j + su_j \right) \right\} \tag{3.4-12}$$

For exp. (3.4-3) we have $n^* = \arg\min\{M(n)\}$, while for exp. (3.4-12) we have $L^* = \arg\min\{M(L)\}$. By definition we have: $M(n) = M(L)$ for $L = \dfrac{Q}{n}$ for every *n* and *L* values. In particular, the following holds: $M(n^*) = M(L^*)$. Thus, it follows that: $L^* = \dfrac{Q}{n^*}$.

QED.

To illustrate, we use the example given in section 3.4.4. The continuous optimal solution for the number of sublots, that has been identified there, lies on an intersection point and its values is: $n^* = 6.67$. The integer optimal solution is, therefore, either 6.00 or 7.00. Since there is no intersection along $M(n)$ within the intervals (6.00,6.67) and (6.67,7.00), the integer optimal solution is obtained via exp. (3.4-10) as follows:

$$\min\left\{ M_k\left(\lfloor n_{kl}^* \rfloor\right), M_l\left(\lceil n_{kl}^* \rceil\right)\right\} = \min\left\{ M_2(6.00), M_1(7.00)\right\} = \min\left\{452.33, 449.71\right\}$$
$$\Rightarrow\ \tilde{n}^* = 7.00$$

Note that, in this case, the continuous optimal number of sublots corresponds to an integer sublot size of $\tilde{L}^* = \dfrac{100}{6.67} = 16$. Therefore, evaluation of the optimal integer sublot size is not required. Otherwise, if it were required, the procedure above would have been utilized. For example, suppose the sublot size was computed to be $L^* = 16.5$, instead of 16.0.
Then, $\lfloor L^* \rfloor = 16.0,\ \lceil L^* \rceil = 17.0$. To determine which of the two values is optimal, convert these values to the *n*-dimension as follows: $n_1 = \dfrac{Q}{L_1} = \dfrac{100}{16} = 6.67$; $n_2 = \dfrac{Q}{L_2} = \dfrac{100}{17} = 5.88$. The makespans of these two are {447.36 ; 453.43} respectively. Thus, $\tilde{L}^* = 16.0$ is the optimal integer sublot size.

3.5 The Impact of Setup and Transfer on a Cost-Based Objective

In this section, we combine the effects of both setup and transfer on the optimal sublot size. The cost-based objective function presented earlier (see exp. (3.3.4-1)) is considered, with the addition of a setup cost:

$$Z = u \cdot M + h \cdot T + c \cdot SU \tag{3.5-1}$$

where:
c is the setup cost per unit time, and *SU* is as follows:

$$SU = n \cdot \sum_{j=1}^{m} SU_j \tag{3.5-2}$$

where:

SU_j - setup cost per sublot on machine j.

Substituting exps. (3.3.4-2), (3.4-3) and (3.5-2) into (3.5-1), we get:

$$Z(n) = \max_{1 \le j \le m} \left\{ \begin{array}{l} \left[u \cdot \left(\dfrac{Q}{n} \cdot \sum p_j + \sum_{j=1}^{m} su_j \right) + (n-1) \cdot \left(\dfrac{Q}{n} \cdot p_j + su_j \right) \right] + \\ + n \cdot h \cdot (m-1) \cdot TT + n \cdot c \cdot \sum_{j=1}^{m} SU_j \end{array} \right\} \tag{3.5-3}$$

Note that the contribution of the handling and setup cost terms to the second derivative is zero. Therefore, the second derivative remains positive and $Z(n)$ is still strictly convex for every value of *n*. Moreover, all the theorems developed in the previous section for the makespan objective hold. Hence, the algorithm given in Section 3.4.2 for the makespan criterion (algorithm (A1)) can also be used to minimize the cost objective in (3.5-3). A single modification is required to make the algorithm suitable for this case. The modification is in the expression for n_k^*, checked in step 2 of the algorithm. This should be:

$$n_k^* = \sqrt{\frac{u \cdot Q \cdot \left(\sum p_j - p_k \right)}{u \cdot su_k + h \cdot (m-1) \cdot TT + c \cdot \sum_{j=1}^{m} SU_j}} \tag{3.5-4}$$

The expression is obtained by setting the first derivative of $Z(n)$ to zero (ignoring the max operand.) The remainder of the algorithm is the same.

Next, we examine the impact of setup on the other two operational measures that have been considered in the beginning of this chapter, namely the mean flow time (MFT) and the average work-in-process (WIP) level. The results are presented in the next two sections.

3.6 The Impact of Setup on the Mean Flow Time (MFT)

Recall that the expression developed in section 3.2.2, for the MFT as a function of the number of sublots, *n*, did not account for setups. However, it can be generalized for the setup case as follows. Exp. (3.2.2-1) becomes:

$$FT_k = \left[L \cdot \sum_{j=1}^{m} p_j + \sum_{j=1}^{m} su_j \right] + (k-1) \cdot \max_j \left\{ L \cdot p_j + su_j \right\} \quad \forall k = 1,..,n \qquad (3.6\text{-}1)$$

The term in the square brackets represents the flow time of the first sublot throughout the system, while the second component represents the additional flow time of each sublot thereafter. Thus, for the MFT we get:

$$MFT = \frac{\sum_{k=1}^{n} FT_k}{n} = \frac{n \cdot \left[L \cdot \sum_{j=1}^{m} p_j + \sum_{j=1}^{m} su_j \right] + \max_j \left\{ L \cdot p_j + su_j \right\} \cdot \sum_{k=1}^{n} (k-1)}{n}$$

Simplifying and taking the max operand out, we get the following:

$$MFT = \max_j \left\{ \left[\frac{Q}{n} \cdot \sum_{k=1}^{m} p_k + \sum_{k=1}^{m} su_k \right] + \frac{n-1}{2} \cdot \left(\frac{Q}{n} \cdot p_j + su_j \right) \right\} \qquad (3.6\text{-}2)$$

Note that the only difference between exp. (3.6-2), for the MFT objective, and exp. (3.4-3) for the makespan objective is in the multiplier of the second term. This multiplier is $\frac{n-1}{2}$ in exp. (3.6-2) compared to $(n-1)$ in exp. (3.4-3). This implies that the analysis carried out for the makespan applies here as well. Thus, the MFT function can be minimized via the algorithm proposed for the makespan objective in Section 3.4.2 (algorithm (A1)), with the following adjustment (resulting from the difference in the multiplier mentioned above.) In step 2 of the algorithm, n_k^* should be computed via the following:

$$n_k^* = \sqrt{\frac{Q \cdot \left(2 \cdot \sum p_j - p_k\right)}{su_k}} \tag{3.6-3}$$

The remainder of the algorithm is the same as algorithm (A1).

3.7 The Impact of Setup on the Average WIP

Similar to the extension derived for the MFT function, we first extend our previous results for the average WIP to account for setup. Thus, exp. (3.2.3-2) becomes:

$$WIP = Q \cdot \frac{\left[\frac{Q}{n} \cdot \sum_{j=1}^{m} p_j + \sum_{j=1}^{m} su_j\right] + \frac{n-1}{2} \cdot \max_j \left\{\frac{Q}{n} \cdot p_j + su_j\right\}}{\left[\frac{Q}{n} \cdot \sum_{j=1}^{m} p_j + \sum_{j=1}^{m} su_j\right] + (n-1) \cdot \max_j \left\{\frac{Q}{n} \cdot p_j + su_j\right\}} \tag{3.7-1}$$

Note that the numerator is exactly the mean flow time, *MFT*, while the denominator is exactly the makespan, *M*. Thus we have:

$$WIP = \frac{MFT}{\left({}^{M}\!/_{Q} \right)}$$

This should come as no surprise, since it is exactly the Little's law. According to Little's law, the queue size (*WIP*) is equal to the flow time of each unit through the system (*MFT*) divided by the cycle time, which is represented here as the makespan time (*M*, the total time for all units) divided by the number of units, *Q*.

Exp. (3.7-1) suggests, as expected that, regardless of the setup, the optimal solution is to have the maximum number of sublots possible, i.e., $n^* = Q$. This result is identical to the result in the case of negligible setup. The rationale is that it is always better to use the smallest sublots, with respect to WIP objective, since by doing so the sublots leave the system as soon as they possibly can. This, in turn, minimizes the average WIP in the system.

Note that exp. (3.7-1) admits a lower bound when $n \to \infty$. The lower bound's value is $Q/2$, as in the case of negligible setup times. This suggests that setup times only affect the average WIP level by slowing down the reduction in WIP when smaller sublots are utilized but if the number of sublots is sufficiently large (i.e., Q is sufficiently large) the WIP level would reach the same level as if setup times were negligible. However, the maximal value for n is Q, and not infinity, and therefore, for the limiting value to approach $Q/2$ an additional condition is required. The condition is as follows:

$$\sum_{k=1}^{m} su_k << \frac{Q}{2} \cdot su_j \quad \text{where j is the maximizing index}$$

Another important observation that can be made based on exp. (3.7-1) is that the function $WIP(n)$ is <u>not</u> a convex function, unlike the previous two functions of makespan and MFT. To show that it is not convex, we use the data of example 3.4.4 as a counter example. Define:

$$WIP_j(n) = Q \cdot \frac{\left[\frac{Q}{n} \cdot \sum_{j=1}^{m} p_j + \sum_{j=1}^{m} su_j\right] + \frac{n-1}{2} \cdot \left\{\frac{Q}{n} \cdot p_j + su_j\right\}}{\left[\frac{Q}{n} \cdot \sum_{j=1}^{m} p_j + \sum_{j=1}^{m} su_j\right] + (n-1) \cdot \left\{\frac{Q}{n} \cdot p_j + su_j\right\}} \tag{3.7-2}$$

$WIP_j(n)$ represents the average WIP level dictated by machine *j*, i.e., if *j* is the maximizing index of:

$$\left\{\frac{Q}{n} \cdot p_k + su_k\right\}_{k=1}^{m}$$

Thus, $WIP(n)$ is comprised of segments of $WIP_j(n)$. In each segment $WIP(n) = WIP_j(n)$, where *j* is the maximizing index over the segment. For the data of example 3.4.4, **Fig. 3.6** depicts the $WIP_j(n)$ functions and the resultant $WIP(n)$ function. From Fig. 3.6, it appears that $WIP_j(n)$, $\forall$j, are strict convex functions. When $WIP_j(n)$ is strictly convex, it makes $WIP(n)$ a segmental strict convex function (i.e., a strict convex function over segments.) However, over the entire set of *n*-

values, $WIP(n)$ is clearly not convex since, as indicated in Fig. 3.6, it is comprised of the smallest (and not the largest) $WIP_j(n)$ function in every segment.

Although it appears from Fig. 3.6 that $WIP_j(n)$ are strict convex functions, this may not always be the case in general. Derivation of the conditions for the strict convexity of $WIP_j(n)$ is lengthy and tedious and is therefore moved to **Appendix A**. Here, we merely summarize that, in the vast majority of real situations, it is expected that $WIP_j(n)$ will be strictly convex.

Another property of the $WIP(n)$ function that is apparent from Fig. 3.6 is the following. $WIP(n)$ is comprised of the $WIP_j(n)$ functions that take the smallest values in each segment (i.e., the "bottom" functions.) This property is proven next.

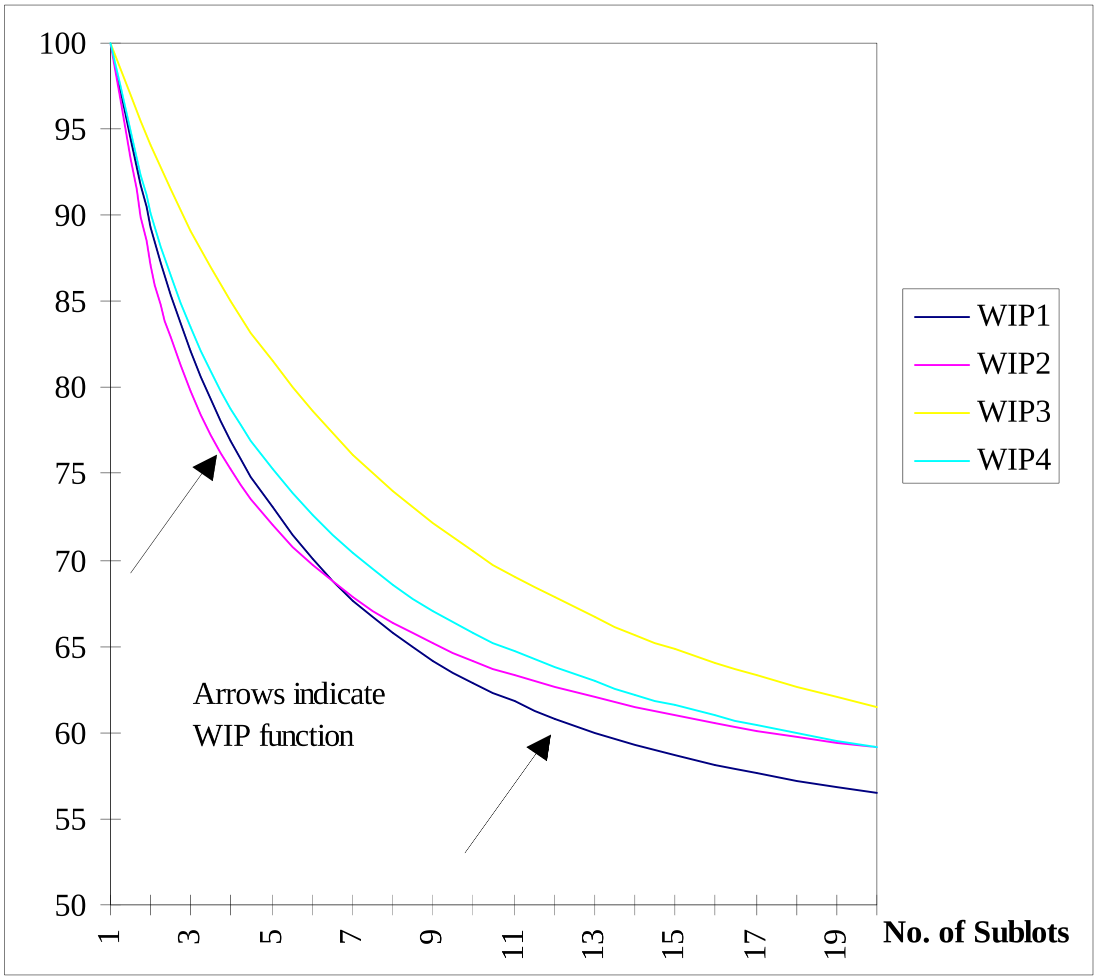


Fig. 3.6. The non-convex function *WIP(n)*.

Theorem 3.7 For a given n, $WIP(n) = WIP_j(n)$ *iff (if and only if) j is the maximizing index of*

$$\left\{ \frac{Q}{n} \cdot p_k + su_k \right\}_{k=1}^{m}$$

Proof.

Define: $a(n) \equiv \left[\frac{Q}{n} \cdot \sum_{j=1}^{m} p_j + \sum_{j=1}^{m} su_j \right]$; $b_j(n) \equiv \frac{Q}{n} \cdot p_j + su_j$, and we have: $a(n), b_j(n) > 0$.

Thus, we have:

$$WIP_j(n) = \frac{a(n) + \frac{n-1}{2} \cdot b_j(n)}{a(n) + (n-1) \cdot b_j(n)}$$

Using this notation, the result stated in the theorem reduces to showing that:

$$b_j(n) \geq b_k(n) \;\; \forall k \neq j \;\; \Leftrightarrow \quad WIP(n) = WIP_j(n) \leq WIP_k(n) \quad \forall k \neq j$$

To that end, consider a pair of machine indices (*j*, *k*). It suffices to show that, for a given n:

$$b_j(n) \geq b_k(n) \quad \Leftrightarrow \quad WIP_j(n) \leq WIP_k(n) \qquad (*)$$

To show that (*) is true, note that the following reductions are equivalent:

$$\begin{aligned}
& WIP_j(n) \leq WIP_k(n) \\
& \Leftrightarrow \\
& \frac{a(n) + \frac{n-1}{2} \cdot b_j(n)}{a(n) + (n-1) \cdot b_j(n)} \leq \frac{a(n) + \frac{n-1}{2} \cdot b_k(n)}{a(n) + (n-1) \cdot b_k(n)} \\
& \Leftrightarrow \\
& a(n)^2 + \frac{n-1}{2} \cdot a(n) \cdot b_j(n) + (n-1) \cdot a(n) \cdot b_k(n) + \frac{n-1}{2}(n-1) \cdot b_k(n) \cdot b_j(n) \leq \\
& \leq a(n)^2 + \frac{n-1}{2} \cdot a(n) \cdot b_k(n) + (n-1) \cdot a(n) \cdot b_j(n) + \frac{n-1}{2}(n-1) \cdot b_k(n) \cdot b_j(n) \\
& \Leftrightarrow \\
& b_j(n) + 2 \cdot b_k(n) \leq b_k(n) + 2 \cdot b_j(n) \\
& \Leftrightarrow \\
& b_k(n) \leq b_j(n)
\end{aligned}$$

QED.

We have shown that $WIP(n)$ is (almost always) a segmental strict convex function, comprised of the $WIP_j(n)$ functions that take the smallest values in each segment. To further determine the path of the function $WIP(n)$, we note that at $n = 1$ the function is equal to $WIP_j(n)$, where j is the machine which maximizes the index, $b_j(n)$. Recall that the machine that has the least setup also maximizes the index (Remark 3.3). Thus, the machine that determines the makespan (and the MFT) also determines the WIP function. That is the machine with the least setup. Then, at the first intersection point, it switches to some $WIP_l(n)$, where l can be obtained from:

$$l = \arg\min_{k \in \overline{S}} \left\{ \frac{p_k - p_j}{su_j - su_k} \right\}$$

See Corollary 3.6 for a proof (follows from the same condition that is used to find the intersection point.) That index, *l*, is again, the index of the machine which determines the makespan (and the MFT), and the intersection point above is identical to the intersection point along the makespan (and the MFT) functions as well.

Note that there are, at most, $(m-1)$ intersection points. This number is further reduced if there exist dominated machines, for which there would not be an intersection point.

3.8 A Unified Cost-Based Model Using a Goal Programming Approach

3.8.1 Problem description and analysis

So far in this Chapter, the effects of non-negligible transfer times and setup times on three operational performance measures, namely, the makespan, the MFT, and the average WIP, have been discussed. An optimal solution algorithm has been proposed for finding the minimizing solution for the makespan criterion, which can also be utilized, with minor changes, to find the optimal solution for the MFT objective and for a cost-based objective that considers the makespan cost, the setup cost, and the handling cost.

In this Section, we present a unified model that attempts to find the optimal solution with respect to a unified objective function. This is done by utilizing the approach of Goal Programming (GP). In GP, weights are assigned to the various terms of the objective function. The weights reflect the degree of importance of each term relative to the other terms. They may also reflect the actual cost per unit time for each term. In addition, they can be used to scale the different terms (e.g., between [0,1].) The following five terms are considered:

- Makespan
- Mean flow time (MFT)
- Average work-in-process (WIP)
- Setup
- Handling (or transfer)

Let $c_1, \ldots, c_5$ be the associated weights assigned to each of the above terms.

Thus, our objective is as follows:

$$\min\left\{Z(n) = c_1 \cdot M(n) + c_2 \cdot MFT(n) + c_3 \cdot WIP(n) + c_4 \cdot SU(n) + c_5 \cdot T(n)\right\} \tag{3.8-1}$$

The only constraint that is imposed is that the number of sublots, *n*, has to satisfy:

$1 \le n \le Q,$ and that *n* is integer

The expressions for the functions $M(n), MFT(n), WIP(n), SU(n),$ and $T(n)$ have been prescribed in the previous sections (see exps. (3.4-3), (3.6-2), (3.7-1), (3.5-2), and (3.3.4-2) respectively.)

To find the optimal solution, we note that as shown earlier, $M(n)$ and $MFT(n)$ are strictly convex functions, while $SU(n)$, and $T(n)$ are linear functions (and therefore convex as well.) The only exception is the function $WIP(n)$, which is considered as a segmental strict convex function. With that in mind, the following Lemma is helpful for our analysis.

Lemma 3.2. The weighted sum of twice-differentiable strict convex functions, $\sum_{i=1}^{n} c_i \cdot f_i(x)$*, is a strict convex function as well, for* $\forall c_i \geq 0$.

Proof.

The second derivative is always positive, since it is always positive for each of the terms. QED.

Corollary 3.7. The objective function, $Z(n)$*, is a segmental strict convex function.*

Proof.

Follows from *Lemma 3.3* , applied to each segment separately. QED.

It is possible to further strengthen Corollary 3.7, and to prove that the objective function, $Z(n)$, has a unique global solution, when considering the effects of either the makespan or the mean flow time (or both).

Theorem 3.8 The objective function, $Z(n)$*,has a unique global minimum* n^**, if* $c_1 > 0$ or $c_2 > 0$.

Proof.

It is trivial for $c_3 = 0$, so consider $c_3 > 0$.

From the analysis of *WIP(n)* in sections 3.4, 3.6, and 3.7, it follows that intersection points along *M(n), MFT(n),* and *WIP(n)* are common because they are determined by the common index expression.

Note that the rate of change in the makespan function at an intersection point, n_{jk}, is:

$$\Delta M_{jk}(n) = M_j(n) - M_k(n) =$$
$$= \left[a(n) + (n-1)\cdot b_j(n)\right] - \left[a(n) + (n-1)\cdot b_k(n)\right] = (n-1)\cdot\left[b_j(n) - b_k(n)\right]$$

Similarly, for the mean flow time function we have:

$$\Delta MFT_{jk}(n) = \frac{(n-1)}{2}\cdot\left[b_j(n) - b_k(n)\right]$$

and for *WIP(n):*

$$\Delta WIP_{jk}(n) = \frac{a(n) + \frac{(n-1)}{2}\cdot b_j(n)}{a(n) + (n-1)\cdot b_j(n)} - \frac{a(n) + \frac{(n-1)}{2}\cdot b_k(n)}{a(n) + (n-1)\cdot b_k(n)} =$$
$$= \frac{\frac{(n-1)}{2}\cdot a(n)}{\left[a(n) + (n-1)\cdot b_j(n)\right]\cdot\left[a(n) + (n-1)\cdot b_k(n)\right]}\cdot\left[b_j(n) - b_k(n)\right]$$

From the above expressions, the following holds:

$$\left|\Delta M_{jk}(n)\right| > \left|\Delta MFT_{jk}(n)\right| > \left|\Delta WIP_{jk}(n)\right| \quad \forall n,\ (j,k) \qquad (1)$$

Moreover, since *M(n)* and *MFT(n)* are both strictly convex, we know that the rates of change of $\Delta M, \Delta MFT$ are increasing at an increasing rate.

By definition we have:

$$\Delta Z(n) = c_1\cdot\Delta M(n) + c_2\cdot\Delta MFT(n) + c_3\cdot\Delta WIP(n) + \Delta C$$

where ΔC is the fixed positive contribution of *SU(n)* and *T(n)*.

Consider the following three cases:

(i) $\Delta Z(n) < 0$ for every *n*

In this case, the optimal number of sublots is attained for the maximum possible *n*.

(ii) $\Delta Z(n) > 0$ for every *n*

In this case, the optimal number of sublots is attained for the minimum possible *n*.

(iii) $\Delta Z(n) < 0$ for some *n* and $\Delta Z(n) > 0$ for some *n*

Let n^* be the point at which the function $\Delta Z(n)$ switches signs (i.e., becomes positive.) For this point we have:

$$\begin{aligned}&\Delta Z\left(n^* + d\right) > 0\\&\Rightarrow\\&c_1 \cdot \Delta M\left(n^* + d\right) + c_2 \cdot \Delta MFT\left(n^* + d\right) + c_3 \cdot \Delta WIP\left(n^* + d\right) + \Delta C > 0\end{aligned} \tag{2}$$

Inequality (2) implies that the rate of change of the first two terms (with the fixed addition of the last) outweigh the rate of change of the third term. Inequality (2) will also hold for any $n > n^*$ since from (1), the rate of change of the first two terms at every intersection point, $n_{jk} (> n^*)$, is greater than the rate of change of the third term. Thus, it can be concluded that $\Delta Z(n)$ will also be increasing at an increasing rate. Therefore, n^* is a unique global minimum. <u>QED</u>.

Before we introduce the algorithm, we discuss one more issue, the first derivative test, which is utilized by the algorithm. Here, as opposed to the previous cases, we include the function $WIP(n)$ in our objective function. Therefore, its' first derivative is a part of the first derivative of the objective function. While it has been a simple task to obtain the first derivative for the other terms, it is certainly not the case for $WIP(n)$. The derivation of its first derivative can be found in **Appendix A**. It is as follows:

$$\frac{dWIP(n)}{dn} = \frac{dWIP_j(n)}{dn} = -Q \cdot \frac{n^2 \cdot k_{12} + n \cdot (2k_{14}) + (k_{23} + k_{34} - k_{14})}{\left[n^2 \cdot 2k_1 + n \cdot (k_2 + 2k_3 - 2k_1) + (k_4 - 2k_3)\right]^2} \tag{3.8-2}$$

where:

j - The maximizing index over the segment

and:

$$k_1 = \frac{su_j}{2}, k_2 = \sum_{j=1}^{m} su_j, k_3 = \frac{Q \cdot p_j}{2}, k_4 = Q \cdot \sum_{j=1}^{m} p_j \text{ and } k_{ij} = k_i \cdot k_j$$

Thus, for the objective function, we have (for each segment in which *j* is the maximizing index):

$$\frac{dZ_j(n)}{dn} = c_1 \cdot \frac{dM_j(n)}{dn} + c_2 \cdot \frac{dMFT_j(n)}{dn} + c_3 \cdot \frac{dWIP_j(n)}{dn} + c_4 \cdot \frac{dSU(n)}{dn} + c_5 \cdot \frac{dT(n)}{dn} \tag{3.8-3}$$

Which leads to:

$$\frac{dZ_j(n)}{dn} = -\frac{1}{n^2}\left(c_1 \cdot Q \cdot \left(\sum_{k=1}^{m} p_k - p_j\right) + c_2 \cdot Q \cdot \left(\sum_{k=1}^{m} p_k - {p_j}\big/{2}\right)\right) + \\ + \left(su_j \cdot \left(c_1 + {c_2}\big/{2}\right) + c_4 \cdot \sum_{k=1}^{m} su_k + c_5 \cdot (m-1) \cdot TT\right) + c_3 \cdot \frac{dWIP_j(n)}{dn} \tag{3.8-4}$$

Exp. (3.8-4) has three terms. The first two terms are similar to what we had in the previous cases while the third term represents the addition due to the function $WIP(n)$. Substituting exp. (3.8-2) in exp. (3.8-4), and setting the objective function derivative to zero, leads to an equation of the fourth power for which we can obtain a solution only by applying numerical techniques. Instead, we propose a quick approximation for exp. (3.8-2) that will enable us to obtain a closed form solution when setting exp. (3.8-4) equal to zero. In the approximation, we only consider the highest power terms of exp. (3.8-2). The approximation is given by exp. (3.8-5).

$$\frac{dWIP_j(n)}{dn} = -Q \cdot \frac{n^2 \cdot k_{12} + n \cdot (2k_{14}) + (k_{23} + k_{34} - k_{14})}{\left[n^2 \cdot 2k_1 + n \cdot (k_2 + 2k_3 - 2k_1) + (k_4 - 2k_3)\right]^2} \approx -Q \cdot \frac{n^2 \cdot k_{12}}{n^4 \cdot 4k_1^2} \tag{3.8-5}$$

Clearly, the approximation is more accurate for large values of *n*. Exp. (3.8-5) can be further simplified, by returning to the original coefficients:

$$\frac{dWIP_j(n)}{dn} = -\frac{1}{n^2} \cdot Q \cdot \frac{\sum_{k=1}^{m} su_k}{su_j} \tag{3.8-6}$$

Substituting (3.8-6) in (3.8-4) and setting it equal to zero leads to:

$$\frac{dZ_j(n)}{dn} = -\frac{1}{n^2}\left(c_1 \cdot Q \cdot \left(\sum_{k=1}^{m} p_k - p_j \right) + c_2 \cdot Q \cdot \left(\sum_{k=1}^{m} p_k - {p_j}/{2} \right) + c_3 \cdot Q \cdot \frac{\sum_{k=1}^{m} su_k}{su_j} \right) + \\ + \left(su_j \cdot \left(c_1 + {c_2}/{2} \right) + c_4 \cdot \sum_{k=1}^{m} su_k + c_5 \cdot (m-1) \cdot TT \right) = 0 \tag{3.8-7}$$

From exp. (3.8-7), we can obtain a closed form solution for the desired value of n_j^*:

$$n_j^* = \sqrt{\frac{c_1 \cdot Q \cdot \left(\sum_{k=1}^{m} p_k - p_j \right) + c_2 \cdot Q \cdot \left(\sum_{k=1}^{m} p_k - {p_j}/{2} \right) + c_3 \cdot Q \cdot \frac{\sum_{k=1}^{m} su_k}{su_j}}{su_j \cdot \left(c_1 + {c_2}/{2} \right) + c_4 \cdot \sum_{k=1}^{m} su_k + c_5 \cdot (m-1) \cdot TT}} \tag{3.8-8}$$

We are now ready to introduce the solution algorithm for this case. The algorithm works as follows. It searches for the minimizing solution over each segment of the function $WIP(n)$, keeping the best solution so far, as it moves from one segment to the next. Within each segment, the algorithm checks for an optimal solution using the first derivative test. If the first derivative test is not satisfied within the segment, the end-point of the segment (i.e., the intersection point) is checked. The algorithm is then repeated until an increase in the objective function value is detected. At that point the algorithm stops, and declares the best solution found thus far as optimal.

We note that the algorithm finds the optimal solution if it occurs at an intersection point. If, however, it does not occur at an intersection point, the algorithm finds a quick approximation to the optimal solution, based on the above analysis (exp. (3.8-8)). The approximation is more accurate when the optimum occurs for large values of *n*. To solve the problem optimally, one can replace the equation given in the second step of the algorithm with a numerical technique for solving equation (3.8-4).

3.8.2 Proposed solution algorithm

Algorithm (A2) for the weighted objective function

Step 1. Set $n_0 = 1$; $Z_0 = Z(n = 1)$.

Apply the Dominance Property ($su_k \geq su_l$ and $p_k \geq p_l$), to eliminate some of the candidate machines. Denote by $\overline{S}$ the candidate machine indices left after the elimination.

Locate the least setup machine: $j = \arg\min_{k \in \overline{S}} \{ su_j \}$.

Step 2. Compute n_j^* using the following:

$$n_j^* = \sqrt{\frac{c_1 \cdot Q \cdot \left(\sum_{k=1}^{m} p_k - p_j \right) + c_2 \cdot Q \cdot \left(\sum_{k=1}^{m} p_k - {p_j}/{2} \right) + c_3 \cdot Q \cdot \frac{\sum_{k=1}^{m} su_k}{su_j}}{su_j \cdot \left(c_1 + {c_2}/{2} \right) + c_4 \cdot \sum_{k=1}^{m} su_k + c_5 \cdot (m-1) \cdot TT}}$$

Use $l = \arg\min_{k \in \overline{S}} \left\{ \frac{p_k - p_j}{su_j - su_k} \right\}$ to find the next intersection point: $n_{jl} = \frac{Q \cdot (p_j - p_l)}{su_l - su_j}$

If $n_0 \leq n_j^* \leq n_{jl}$ then n_j^* is current best ; Set $Z_0 = Z(n_j^*)$.

If $Z(n_{jl}) \leq Z_0$, then n_{jl} is current best ; Set $Z_0 = Z(n_{jl})$

Else (i.e., $Z(n_{jl}) > Z_0$), STOP

If $\overline{S}$ is empty, STOP. Otherwise, continue to step 3.

Step 3. Set $n_0 = n_{jl}$.

Remove j from $\overline{S}$.

Set $j \leftarrow l$.

Go to step 2.

Note that the equation for computing n_k^* in step 2 is an approximation. It is accurate only if the *WIP* term, which has been approximated, is not considered (i.e., if $c_3 = 0$). In that case, the algorithm results in the optimal solution. The algorithm also results in the optimal solution when $c_3 \neq 0$, but the optimal solution occurs at an intersection point. Otherwise, the algorithm provides a near-optimal solution. For the exact optimal solution, one can solve exp. (3.8-4) via numerical methods.

3.8.3 Complexity of the algorithm

The changes made in the algorithm (A2) with respect to algorithm (A1) (presented earlier for the makespan criterion) do not affect the complexity of the algorithm, and the arguments provided there remain true in this case as well. Therefore, the complexity is still of order *O(m)* for arithmetic operations and $O\left(m^2\right)$ for comparisons.

3.8.4 A comprehensive numerical example

Consider a six-machine flow shop. Setup and processing times are given in **Table 3.3**. The weights for the components (i.e., the makespan, the MFT, the WIP, the setup, and the transfer) are provided as well. The transfer time per sublot is *TT*=10 time units (and non-operator-based.)

Table 3.3. Data for example 3.2

Q=	10000						
Machines	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**	**Total**
Setup Time	10	100	180	210	100	200	800
Proc. Time	1.20	1.10	0.90	0.80	0.70	0.50	5.2

C_1	**C_2**	**C_3**	**C_4**	**C_5**	Total
1.00	1.00	1.00	1.00	1.00	5.00

The solution via the algorithm given in section 3.8.2 is as follows:

Step 1:

Set $n_0 = 1,\ \mathrm{Z}_0 = Z(n_0) = 59{,}450$.

Exclude machines M5, M6 from set $\overline{S}$ (dominated by M2, M4 respectively.)

$\overline{S} = \{1,2,3,4\}$.

Set k=1, the machine with the least setup.

Step 2:

$n_1^* = 22.63,\ l = 2,\ n_{12} = 5.56$. Set n_{12} current best, $Z_0 = Z(n_{12}) = 27{,}110$.

Step 3:

$n_0 = 5.56,\ \overline{S} = \{2,3,4\},\ k = 2$. Go to step 2.

Step 2:

$n_2^* = 8.15,\ l = 3,\ n_{23} = 12.50$. Set n_2^* current best, $Z_0 = Z(n_2^*) = 26{,}774$.

$Z(n_{23}) > Z_0$, STOP.

The various terms of the objective function are depicted in **Fig. 3.7**. The optimal solution found by the algorithm is $n^* = 8.15$ with $Z_0 = 26{,}774$. Note that the optimal solution does not occur at an intersection point, and is therefore approximated. The optimal solution found via numerical analysis is: $n^* = 7.15$ with $Z^* = 26{,}591$. Thus, the objective function value of the approximated solution is only 0.69% above the objective function value of the optimal solution. (For larger n-optimal values, it is expected that the difference would be reduced even further.) However, the discrete solution, under the approximation scheme as well as the optimal scheme, is attained for $n^* = 8$ with $Z^* = 26{,}729$.

Next, we utilize the above example for sensitivity analysis. As stated earlier, the algorithm's solution is optimal either if c_3 is negligible or if the optimal solution occurs at an intersection point. To illustrate both cases, two parameters are varied, namely c_3 and Q, and the solutions obtained by the algorithm are compared with the optimal solution. The results are summarized in **Table 3.4**.

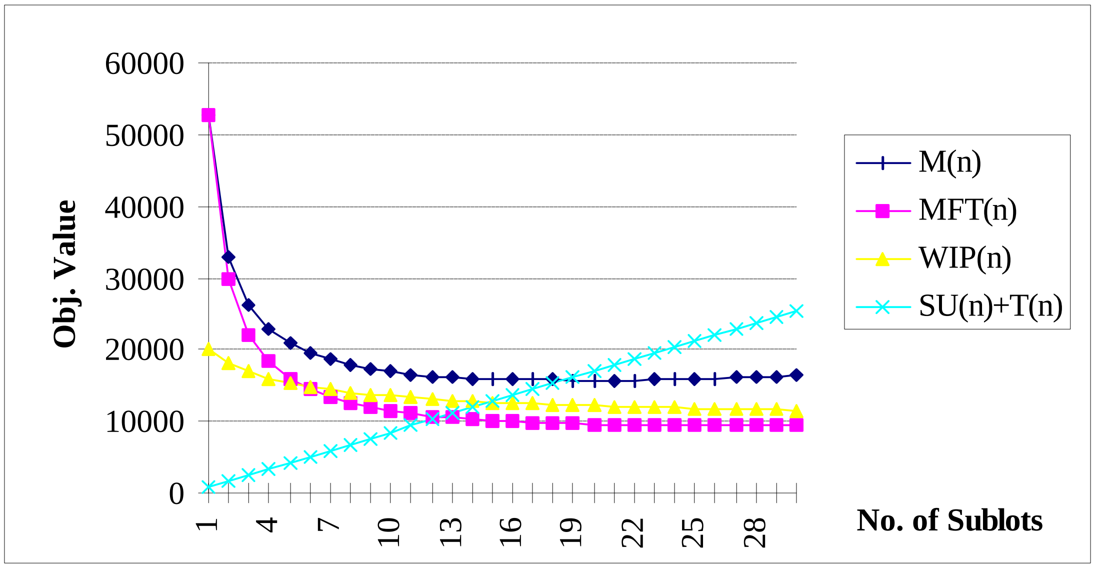


Fig. 3.7. The values of the terms of the objective function as a function of the number of sublots for Example 3.2

Table 3.4. Sensitivity analysis for Example 3.2

		Algorithm's Solution		**Optimal Solution**		**Difference**
Q	**C_3**	**n***	**Z***	**n***	**Z***	**in Z* (%)**
2,500	0.00	4.99	14,950	4.99	14,950	0.00
2,500	0.50	5.65	16,054	5.15	15,919	0.85
2,500	1.00	5.47	17,002	4.75	16,901	0.60
2,500	2.00	6.25	19,084	5.14	18,867	1.15
5,000	0.00	6.61	22,928	6.61	22,928	0.00
5,000	0.50	7.68	24,869	6.80	24,767	0.41
5,000	1.00	7.76	26,676	7.15	26,602	0.28
5,000	2.00	8.95	30,512	7.75	30,254	0.85
7,500	0.00	8.33	30,033	8.33	30,033	0.00
7,500	0.50	8.33	32,671	8.33	32,671	0.00
7,500	1.00	8.33	35,309	8.71	35,301	0.02
7,500	2.00	8.33	40,584	9.12	40,517	0.17
10,000	0.00	9.97	36,835	9.97	36,835	0.00
10,000	0.50	11.11	40,266	10.27	40,215	0.13
10,000	1.00	11.11	43,596	11.11	43,596	0.00
10,000	2.00	11.11	50,257	11.11	50,257	0.00

It is evident from Table 3.4 that even when the algorithm does not find the optimal solution, the difference in the objective function values is within 1%. This finding suggests that in the neighborhood of the optimum, the objective function is less sensitive to changes in the independent variable, as expected based on the shape of the objective function (strictly convex with a global minimum.) Moreover, the algorithm finds the optimal solution in many cases, because it occurs at an intersection point. The results suggest that, in many cases, the optimal solution occurs at an intersection point.

3.9 Summary of Results

In this Chapter, the single-lot equal-sublot case has been analyzed. The possible benefits from lot streaming have been evaluated under various operating conditions and for various performance measures. It has been shown that lot streaming can result in substantial improvements in many practical situations.

Various non-negligible forms of transfer times have been considered, and their impact on the optimal number of sublots to be used, with respect to various operational and cost-based objective functions, has been analyzed.

Similarly, the effect of sublot-attached setup on various operational and cost-based objective functions has been studied. An algorithm has been proposed for finding the optimal number of sublots (and, consequently, the sublot size) for the most general objective function and an *m*-machine flow shop. Utilizing numerical analysis methods, one can solve the problem optimally via the algorithm. For faster results, a quick approximation equation has been developed, which is utilized by the algorithm to avoid the need for numerical analysis. Experiments with the approximation reveal that the algorithm finds the optimal solution in many situations. When the algorithm does not find the optimal solution, the difference in the objective function values is minor (i.e., the solution is near-optimal.) This algorithm can easily be utilized to solve for the optimal number of sublots, with respect to any of the specific objective functions that have been considered, by simply setting the other coefficients to zero.

Chapter 4: Multiple Batch Models With Equal Sublots

4.1 Introduction

In this Chapter, the results obtained in Chapter 3 for a single lot are extended to the case of multiple lots. The multiple-lot problem is more complicated since, in addition to the determination of the sublot-size, a sequence among lots must be determined. Such a sequencing problem on multiple machines is known to be NP-hard (Pinedo, 1996). Thus, the multiple batch problem has a NP-hard problem embedded in it. This problem is addressed in two steps. First, the sublot size is assumed known. In this case, the problem reduces to that of sequencing lots (or sublots, if intermingling is possible), given that they are **streamed** throughout the production system. This problem will be referred to as the Lot Streaming Sequencing Problem (LSSP). Then, the more general problem of simultaneously determining the sublot size and the sequence among lots is considered. This problem will be referred to as the Flow Shop Lot Streaming (FSLS) problem. In line with their applications, lot-attached setups are considered for the former problem (LSSP) whereas sublot-attached setups are considered for the latter problem (FSLS). Due to the complexity of the analysis in this case, only the makespan objective is considered. The mean flow time and the average WIP objectives will be considered separately in Chapter 5, for the dynamic version of the problem.

In the analysis and the development of efficient solution procedures for the above problems, several issues related to the solution characteristics are addressed. These further enrich the discussion throughout this Chapter. These issues give rise to the following questions: (a) How is the solution (i.e., the makespan reduction) affected by the number of lots, N? ; (b) How is the solution affected by the lot sizes, Q_i ? ; (c) How is the solution affected by the equal sublot size, *L*, and is it possible to obtain near-optimal performance with a sublot size that is significantly greater than one? ; (d) How is the solution affected by setup times (as a percentage of the processing times)? ; and (e) How is the sequence affected by setup times?

The development of solution procedures for the LSSP and the FSLS problems, and the search for answers to the above questions, motivate the discussion in this Chapter and the subsequent Chapter (Chapter 5).

In this Chapter, an efficient near-optimal heuristic, referred to as the Bottleneck Minimal Idleness (BMI) heuristic for the Lot-Streaming Sequencing Problem (LSSP), is developed and examined. Several numerical examples are provided to demonstrate how the BMI heuristic works. A comprehensive experimental study is given to show that the non-intermingled BMI heuristic produces near-optimal solutions. The same study shows that the BMI heuristic outperforms the FIHLS heuristic. The FIHLS heuristic is an extension of the Fast Insertion Heuristic (FIH), the best known heuristic for traditional flow-shop sequencing problems, to lot-streaming. The impact of setup times on the level of intermingling in the schedule, from a theoretical as well as from a practical standpoint, is also discussed in this Chapter. Lower bounds on the number of intermingled sublots in an intermingled schedule are derived. A numerical example is used to show that, practically, even when relatively small setup times are introduced, it becomes undesirable to intermingle sublots, due to the significant negative impact of setup on the makespan measure.

4.1.1 Classification of multiple-batch lot streaming problems

The general problem that is addressed in this Chapter can be denoted as $N/m/F_T/C_{max}$ (i.e., N sublots, m machines, Flow-shop with Transfer batches, and makespan objective.) This notation is borrowed from Vickson and Alfredson (1992). To further clarify the specific types of problems that arise, a classification of the problems according to two system characteristics is provided in **Fig. 4.1**. These two characteristics are:

- Intermingling of sublots: This pertains to the flexibility of the production system to process different sublots one after another. This is a production system constraint. For example, in SMT lines for PCB manufacturing, the printed-circuit-boards cannot intermingle. The machines are set-up to process lots of PCBs of the same type continuously. On the other hand, in Semi-Conductor manufacturing processes, sublots of different types can intermingle and, in fact, it is very common.

- <u>Nature of setup:</u> A setup can either be negligible or performed off-line (zero setup), or it can also be lot-attached (i.e., incurred prior to the entire lot) or sublot-attached (i.e., incurred prior to every sublot, regardless of the type.) Note that lot-attached setup implies no-intermingling of the sublots.

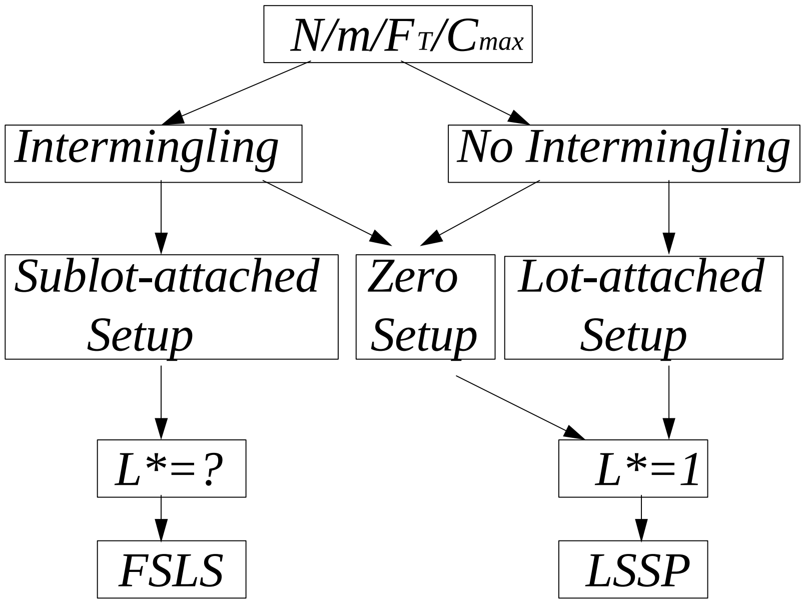


Fig. 4.1. Classification of multiple-batch flow shop lot streaming problems

As we shall see in this Chapter, as long as the setup is not sublot-attached, the optimal sublot size is one ($L^*=1$). Thus, in these cases, the problem reduces to the LSSP, where the purpose of sequencing lots is to minimize the makespan, given that the lots are streamed one-at-a-time. On the other hand, under sublot-attached setup, the sublot sizes need to be determined as well. This, in turn, leads to the FSLS problem.

4.1.2 Notation and problem formulation

In contrast to the single batch model discussed in Chapter 3, where only the determination of sublot size is required, in multiple-batch lot-streaming, the sequence among lots need also be determined. Thus, two questions must be addressed by the model:

- What is (are) the optimal sublot size(s) ?
- What is the optimal sequence of the sublots ?

These questions dictate the decision variables of the model. The issue of formulating a sequence is addressed next. A sequence can be viewed as a chain of pairs, where the first operation in each pair is the same as the second operation of the previous pair in the chain. Viewing the sequence in this fashion allows to impose constraints on the pairs in order to establish a valid sequence. For any sublot, its processing on any two consecutive machines cannot overlap. In other words, the completion times of the sublot on these machines must differ by at least the processing time of the sublot on the first machine of the pair. Similarly, for any two sublots, on any machine, either the first sublot of the pair precedes the other or vice versa. In either case, the corresponding completion times of the sublots on the machine must differ by at least the processing time of the first sublot of the pair on that machine.

Before we formulate a mathematical model, the notation which is utilized throughout this Chapter is introduced next. It is mostly an extension of the notation given in Chapter 3, with the additional subscript *i*, used to represent the lot type.

Q_i	- The lot size of product type i ; $i = 1,..,N$
n_i	- The number of sublots streamed through the flow shop for lot type i ; $l = \sum_{i=1}^{N} n_i$
m	- The number of machines in the flow shop ; $j = 1,..,m$
p_{ij}	- The processing time of a single item of type i on machine j

BN	- The bottleneck machine ; $BN \equiv \arg \max_{1 \le j \le m} \left\{ \sum_{i=1}^{N} Q_i \cdot p_{ij} \right\}$
L_{ik}	- The size of the *k*-th sublot of product type *i* ; $k = 1, \dots, n_i$
L	- The (equal) sublot size
su_{ij}	- The setup time required prior to the processing of sublot type *i* on machine *j*
$C_{[i],j}$	- Completion time of the i-th lot in the sequence on machine j.
C_{ikj}	- The completion time of the *k*-th sublot of product type *i* on machine *j*
M	- The makespan
MFT	- The mean flow time
WIP	- The average WIP level
SN	- Set of all non-intermingled permutation sequences.
SI	- Set of all intermingled permutation sequences.
S	- Set of all permutation sequences. $S = SN \bigcup SI$.

[*i*] will be used to denote the type of *i*-th lot in a (non-intermingled) sequence. As before, a superscript '*LS*' is added to a measure, such as the makespan, to denote that the analysis applies to the lot streaming case. A superscript '*' is added to denote optimality.

Under the restriction of equal sublots, with *L* denoting the sublot size in the model, it follows that the number of sublots of each lot, n_i, is:

$$n_i = \left[\frac{Q_i}{L} \right]^+$$

Let the last sublot consist of the remainder of the lot, i.e., be equal to: $Q_i - (n_i - 1) \cdot L < L$.

The problem of determining the optimal sublot size and the optimal sequence among the sublots can be formulated as follows:

$$Min\ \ M$$

$$s.t \quad \begin{aligned} & L_{ik} - L = 0 \quad \forall \mathrm{i} = 1,..,\mathrm{N} \ ; \ \forall \mathrm{k} = 1,..,n_i - 1 \\ & L_{i,n_i} + (n_i - 1)\cdot L - Q_i = 0 \quad \forall \mathrm{i} = 1,..,\mathrm{N} \end{aligned} \tag{4.1-1}$$

$$C_{i,k,j+1} - C_{ikj} - L_{ik}\, p_{i,j+1} \ge 0 \quad \forall \mathrm{i} = 1,..,\mathrm{N} \ ; \ \forall \mathrm{k} = 1,..,n_i \ ; \ \forall \mathrm{j} = 1,..,\mathrm{m} \tag{4.1-2}$$

$$\left\langle \begin{aligned} & C_{i'k'j} - C_{ikj} - L_{i'k'}\, p_{i'j} \ge 0 \ ; \quad (\mathrm{i},\mathrm{k})\,\text{before}\,(\mathrm{i}',\mathrm{k}')\,\text{on machine j} \\ & \text{or} \\ & C_{ikj} - C_{i'k'j} - L_{ik}\, p_{ij} \ge 0 \ ; \ (\mathrm{i}',\mathrm{k}')\,\text{before}\,(\mathrm{i},\mathrm{k})\,\text{on machine j} \end{aligned} \right\rangle \tag{4.1-3}$$

$$For: \forall (\mathrm{i},\mathrm{k}) \ne (\mathrm{i}',\mathrm{k}') \ ; \ \forall \mathrm{i},\mathrm{i}' = 1,..,\mathrm{N} \ ; \ \forall \mathrm{k},\mathrm{k}' = 1,..,n_i \ \forall \mathrm{j} = 1,..,\mathrm{m}$$

$$M - C_{i,n_i,m} \ge 0 \quad \forall \mathrm{i} = 1,..,\mathrm{N} \tag{4.1-4}$$

$$L \ge 1, \ \text{int.} \tag{4.1-5}$$

Constraints (4.1-1) ensure that all the sublots are of equal size, and that the last sublot of each lot consists of the remaining items. Constraints (4.1-2) guarantee the feasibility of the schedule by not permitting overlap of a sublot on every two consecutive machines. Constraints (4.1-3) are responsible for ensuring the validity of the sequence. For every pair of sublots, either one has to precede the other on the same machine. Constraints (4.1-4) determine the resulting makespan, which must be at least as large as the completion time of the last sublot of every lot on the last machine. Lastly, constraint (4.1-5) makes sure that the sublot size is valid and integer. Note that non-negativity constraints on the completion times, C_{ijk}, are not necessary here since *L* is positive. Also, note that the formulation does not enforce a permutation schedule. On the other hand, note that, since *L* is unknown, the n_i's are also unknown. Therefore, the above formulation only holds for a given *L*. However, if necessary, the optimal *L* can be determined by enumeration over all possible *L*-values, using the above formulation. Fortunately, as we shall demonstrate in Theorem 4.1, for cases of zero setup, as well as for cases with lot-attached setup, only unit-sized sublots need to be considered (i.e., $L = 1$). Therefore, in the above formulation, it suffices to consider $n_i = Q_i$, and constraint (4.1-5) becomes redundant.

The above formulation can also be modified to accommodate consistent sublots and job-shop routing (see Dauzere-Peres and Lasserre, 1997.) It is obvious from this formulation that the third set of constraints is problematic and makes the problem difficult-to-solve. In order to model the disjunctive ("or") constraints, a binary variable may be used to determine which of the two constraints is binding for a given pair of sublots. Let y_{ik} be the corresponding binary variable for a certain pair (*i*,*k*). Using y_{ik}, constraints (4.1-3) can be re-written as follows:

$$\left\langle \begin{array}{l} A \cdot (1 - y_{ik}) \geq C_{i'k'j} - C_{ikj} - L_{ik}\, p_{ij} \geq -A \cdot y_{ik} \\ A \cdot y_{ik} \geq C_{ikj} - C_{i'k'j} - L_{i'k'}\, p_{i'j} \geq -A \cdot (1 - y_{ik}) \end{array} \right\rangle$$

$$For: \forall(\mathrm{i,k}) \neq (\mathrm{i}', \mathrm{k}')\ ;\ \forall \mathrm{i}, \mathrm{i}' = 1,.., \mathrm{N}\ ;\ \forall \mathrm{k}, \mathrm{k}' = 1,.., n_i\ \forall \mathrm{j} = 1,.., \mathrm{m}$$

where *A* is a sufficiently large number. However, the introduction of binary constraints immediately makes the problem a mixed-linear integer programming problem, that is difficult-to-solve.

In the above formulation, setup times can be incorporated by including them in constraints (4.1-2) and (4.1-3). For the case of lot-attached setups, setup times are added whenever $i \neq i'$, i.e., when sublots of different lots follow one another. For the case of sublot-attached setups, setup times are also added whenever $i = i', k \neq k'$, i.e., when sublots of the same lot follow one another.

4.2 Problem Complexity

As mentioned in Section 2.3 of the literature survey, the sequencing problem in a flow-shop that consists of more than two machines is known to be NP-hard (with special exceptions for *m*=3, when one machine dominates the others.) Under lot streaming, the fact that each sublot can be sequenced separately, and independently of the other sublots of the same type, further complicates the sequencing problem. It becomes necessary to consider sequences of intermingled sublots as well. To that end, we wish to differentiate between two types of sequences. A sequence in which sublots of different lots are intermingled is referred to as an 'intermingled (permutation) sequence' (SI). A sequence in which all the sublots of the same lot are consecutive is referred to as a 'non-intermingled (permutation) sequence' (SN).

In scheduling theory of general flow-shops, a permutation schedule need not necessarily be optimal. Rarely as it may be, sequences on downstream machines may differ from the initial sequence (i.e., the sequence on M_1) in an optimal schedule. Nevertheless, it is well accepted to assume identical sequences (i.e., permutation schedules) and to find the best among them (Lawler et al., 1993). We consider this to be the case under lot-streaming as well.

To fully appreciate the complications associated with lot-streaming sequencing problems, the expression for the number of possible sequences is derived next. For a given number of lots, *N*, the number of possible sequences in the set SN is $N!$. The number of possible sequences in the set SI, on the other hand, is by far larger. If the number of items in each lot *i* is Q_i, the total number of sequences, *K*, of (only) unit-sized sublots, is:

$$K = \frac{\left(\sum_{i=1}^{N} Q_i\right)!}{\prod_{i=1}^{N} (Q_i\,!)} \tag{4.2-1}$$

Exp. (4.2-1) reduces to $N!$ in the case of $Q_i = 1\ \forall i$, i.e., when the lots cannot be decomposed and lot-streaming is not considered. Based on Exp. (4.2-1), the number of *SI* sequences is the difference, *K*-*N*!. To illustrate the complexity implied by Exp. (4.2-1), consider two lots of only 10 items each. The number of possible (different) sequences, computed via Exp. (4.2-1), i.e., using *K*-*N*!, is 184,754. This number is of the same order of magnitude as the number of possible sequences in a regular flow-shop problem (i.e., when lot streaming is not considered) of <u>nine</u> jobs. Note that an increase in Q_i to a realistic order of 100's or 1000's of items makes the number of possible sequences enormously large. Thus, the lot-streaming sequencing problem appears to be a notoriously hard combinatorial optimization problem, in which the computation of an optimal solution, even for small-size problems, requires too much computation time and makes explicit and/or implicit (branch-and-bound based) solution methods impractical.

4.3 Potential Benefits from Lot Streaming in a Multiple-Batch Flow-Shop

Similar to the analysis carried out for the single batch case in Section 3.2, we wish to start with evaluating the extent of the potential benefits via lot-streaming in multiple-batch flow-shops, assuming negligible setup and transfer times. The potential benefits are evaluated with respect to the same three operational measures considered in Chapter 3, namely the makespan, the mean flow time, and the average WIP. Extreme cases are analyzed in order to realize the extent of the benefits possible under certain operating conditions.

4.3.1 The makespan objective

For simplicity, we assume that the lots have a common bottleneck and that the bottleneck machine is dominant for all the lots, i.e., the following holds:

$$p_{i,BN} \geq p_{ij} \quad \forall j \neq BN \;;\; \forall i = 1,\ldots,N \tag{4.3.1-1}$$

Under condition (4.3.1-1), the makespan of a given (non-intermingled) sequence, which utilizes lot streaming of unit-sized sublots, is as follows:

$$M^{LS} = \sum_{j=1}^{BN-1} p_{[1],j} + \sum_{i=1}^{N} Q_{[i]} \cdot p_{[i],BN} + \sum_{j=BN+1}^{m} p_{[N],j} + I^{LS} \tag{4.3.1-2}$$

where:

I^{LS} - The total idle time between the **lots** on the bottleneck machine, under lot streaming.

Similarly, the makespan without lot streaming is as follows:

$$M = \sum_{j=1}^{BN-1} Q_{[1]} \cdot p_{[1],j} + \sum_{i=1}^{N} Q_{[i]} \cdot p_{[i],BN} + \sum_{j=BN+1}^{m} Q_{[N]} \cdot p_{[N],j} + I \tag{4.3.1-3}$$

where:

I - The total idle time between the **lots** on the bottleneck machine (without lot streaming.)

Therefore, the ratio of the two is:

$$\frac{M^{LS}}{M} = \frac{\sum_{j=1}^{BN-1} p_{[1],j} + \sum_{i=1}^{N} Q_{[i]} \cdot p_{[i],BN} + \sum_{j=BN+1}^{m} p_{[N],j} + I^{LS}}{\sum_{j=1}^{BN-1} Q_{[1]} \cdot p_{[1],j} + \sum_{i=1}^{N} Q_{[i]} \cdot p_{[i],BN} + \sum_{j=BN+1}^{m} Q_{[N]} \cdot p_{[N],j} + I} \tag{4.3.1-4}$$

Consider the extreme case in which no idle time on the bottleneck exists between the lots or the sublots in both cases, i.e., assume that the following condition holds:

$$p_{[i],j} \geq p_{[i+1],j-1} \quad \forall \text{i, j} \tag{4.3.1-5}$$

This condition means that $I^{LS} = I = 0$. If, in addition, we use equal lot sizes, i.e., $Q_i = Q$; exp. (4.3.1-4) becomes the following:

$$\frac{M^{LS}}{M} = \frac{Q \cdot \sum_{i=1}^{N} p_{[i],BN} + \sum_{j=1}^{BN-1} p_{[1],j} + \sum_{j=BN+1}^{m} p_{[N],j}}{Q \cdot \left(\sum_{j=1}^{BN-1} p_{[1],j} + \sum_{i=1}^{N} p_{[i],BN} + \sum_{j=BN+1}^{m} p_{[N],j} \right)} \tag{4.3.1-6}$$

Thus, for large lot sizes, such that: $Q \cdot \sum_{i=1}^{N} p_{[i],BN} >> \sum_{j=1}^{BN-1} p_{[1],j} + \sum_{j=BN+1}^{m} p_{[N],j}$, we get:

$$\frac{M^{LS}}{M} = \frac{\sum_{i=1}^{N} p_{[i],BN}}{\sum_{j=1}^{BN-1} p_{[1],j} + \sum_{i=1}^{N} p_{[i],BN} + \sum_{j=BN+1}^{m} p_{[N],j}} \tag{4.3.1-7}$$

Under the case of perfect balance, i.e., $p_{ij} = p \quad \forall \text{i, j}$ (which still satisfies conditions (4.3.1-1), and (4.3.1-5)), the above expression reduces to the following:

$$\frac{M^{LS}}{M} = \frac{N}{\left[N + (m-1)\right]} \tag{4.3.1-8}$$

Note that exp. (4.3.1-8) is purely dependent upon the problem parameters. For a relatively small number of lots, and a large number of machines, lot streaming is expected to be very beneficial. For example, for N=5, and m=10, the reduction in the makespan is: $1 - {}^{5}\!/_{14} = 64.3\%$. Also note that the above analysis assumes no idle time on the bottleneck machine, even without lot streaming. However, it is expected that in many instances $I >> I^{LS}$ which further enhances the gains via lot streaming.

We also note that based on exp. (4.3.1-8), the benefits are likely to decrease as the number of lots considered, N, increases. This is true in general (i.e., without the assumptions made in the above analysis.) The proof is provided in **Appendix E**.

The other extreme case, that is considered, is the case in which the number of machines is large while the lot sizes and the number of lots are relatively small, such that:

$$Q \cdot \sum_{i=1}^{N} p_{[i],BN} << \sum_{j=1}^{BN-1} p_{[1],j} + \sum_{j=BN+1}^{m} p_{[N],j} .$$

In this case, from exp. (4.3.1-6), we get:

$$\frac{M^{LS}}{M} = \frac{\sum_{j=1}^{BN-1} p_{[1],j} + \sum_{j=BN+1}^{m} p_{[N],j}}{Q \cdot \left(\sum_{j=1}^{BN-1} p_{[1],j} + \sum_{i=1}^{N} p_{[i],BN} + \sum_{j=BN+1}^{m} p_{[N],j} \right)} \tag{4.3.1-9}$$

which, under perfect balance, reduces to:

$$\frac{M^{LS}}{M} = \frac{m-1}{Q \cdot \left[N + (m-1) \right]} \tag{4.3.1-10}$$

Exp. (4.3.1-10) suggests even greater improvements to the makespan, via lot streaming, than the previous case. However, *N* must be sufficiently small in this case. Note that when *N*=1, i.e., a single lot is considered, both exp. (4.3.1-8) and exp. (4.3.1-10) reduce to their respective ratios in the single lot case (i.e., to $\frac{1}{m}$ and $\frac{1}{Q}$).

4.3.2 The MFT objective

The mean flow time (MFT) for multiple lots, in a non-intermingled sequence, where sublots of the same lot follow one another, can be obtained by dividing the sum of the MFT's of the individual lots by the number of lots. Thus, we have:

$$MFT^{LS} = \frac{\sum_{i=1}^{N} MFT_i^{LS}}{N} \tag{4.3.2-1}$$

Exp. (4.3.2-1) represents the lot-MFT, not to be confused with the sublot-MFT. Using exp. (3.2.2-5), for the first lot in the sequence, we have:

$$MFT_{[1]}^{LS} = \sum_{j=1}^{m} p_{[1],j} + \frac{Q_{[1]} - 1}{2} \cdot p_{[1],BN} \tag{4.3.2-2}$$

Assuming that conditions (4.3.1-1) and (4.3.1-5) are in force, the mean flow time, for the i-th lot in the sequence, can be computed as follows. If it started from time zero, and had no interference of flow due to the schedule of previous lots (because conditions (4.3.1-1) and (4.3.1-5) are in force), then it would have been:

$$\sum_{j=1}^{m} p_{[i],j} + \frac{Q_{[i]} - 1}{2} \cdot p_{[i],BN}$$

However, it actually starts not at time zero, but at time $\sum_{r=1}^{i-1} Q_{[r]} \cdot p_{[r],1}$. Therefore, the MFT of the i-th lot is:

$$MFT_{[i]}^{LS} = \sum_{r=1}^{i-1} Q_{[r]} \cdot p_{[r],1} + \sum_{j=1}^{m} p_{[i],j} + \frac{Q_{[i]} - 1}{2} \cdot p_{[i],BN} \tag{4.3.2-3}$$

Substituting (4.3.2-3) in (4.3.2-1) we get:

$$MFT^{LS} = \frac{\sum_{i=2}^{N} \sum_{r=1}^{i-1} Q_{[r]} \cdot p_{[r],1} + \left(\sum_{i=1}^{N} \sum_{j=1}^{m} p_{[i],j} \right) + \left(\sum_{i=1}^{N} \frac{Q_{[i]} - 1}{2} \cdot p_{[i],BN} \right)}{N} \tag{4.3.2-4}$$

Utilizing the same approach for the case without lot streaming, we have:

$$MFT_{[1]} = \sum_{j=1}^{m} Q_{[1]} \cdot p_{[1],j}$$

$$MFT_{[i]} = \sum_{r=1}^{i-1} Q_{[r]} \cdot p_{[r],1} + \sum_{j=1}^{m} Q_{[i]} \cdot p_{[i],j} \tag{4.3.2-5}$$

Therefore:

$$MFT = \frac{\sum_{i=1}^{N} MFT_i}{N} = \frac{\sum_{i=2}^{N}\sum_{r=1}^{i-1} Q_{[r]} \cdot p_{[r],1} + \sum_{i=1}^{N}\sum_{j=1}^{m} Q_{[i]} \cdot p_{[i],j}}{N} \tag{4.3.2-6}$$

From Exps. (4.3.2-4) and (4.3.2-6), the ratio is as follows:

$$\frac{MFT^{LS}}{MFT} = \frac{\sum_{i=2}^{N}\sum_{r=1}^{i-1} Q_{[r]} \cdot p_{[r],1} + \left(\sum_{i=1}^{N}\sum_{j=1}^{m} p_{[i],j}\right) + \left(\sum_{i=1}^{N} \frac{Q_{[i]}-1}{2} \cdot p_{[i],BN}\right)}{\sum_{i=2}^{N}\sum_{r=1}^{i-1} Q_{[r]} \cdot p_{[r],1} + \sum_{i=1}^{N}\sum_{j=1}^{m} Q_{[i]} \cdot p_{[i],j}} \tag{4.3.2-7}$$

For the extreme case of equal lots and a perfect balance ($Q_i = Q, p_{ij} = p \quad \forall \text{i, j}$), Exp. (4.3.2-7) reduces to the following:

$$\frac{MFT^{LS}}{MFT} = \frac{\frac{N \cdot (N-1)}{2} \cdot Q \cdot p + N \cdot m \cdot p + N \cdot \frac{Q-1}{2} \cdot p}{\frac{N \cdot (N-1)}{2} \cdot Q \cdot p + N \cdot m \cdot Q \cdot p} = \frac{Q \cdot N + 2 \cdot m - 1}{Q \cdot (N + 2 \cdot m - 1)} \tag{4.3.2-7a}$$

From Exp. (4.3.2-7a), if the lot size is sufficiently large, the ratio approaches the value $\frac{N}{N + 2 \cdot m - 1}$, which is significant as long as *N* is relatively small. Note that for *N*=1 (i.e., a single lot), the above value reduces to $\frac{1}{2 \cdot m}$, in agreement with the results for the single lot case (Section 3.2.2). Also note that if the number of lots becomes very large, the value approaches 1.00, as in the case of the makespan.

Next, consider the MFT resulting from excluding the waiting time of the lots on the first machine, before the start of their processing, i.e., accounting only for the flow time of the lots through the system from the beginning of their processing on machine 1. In that case, exp. (4.3.2-3) becomes:

$$MFT_{[i]}^{LS} = \sum_{j=1}^{m} p_{[i],j} + \frac{Q_{[i]}-1}{2} \cdot p_{[i],BN} \tag{4.3.2-8}$$

Note that this time we are only considering the flow time of each sublot (excluding any waiting time prior to the beginning of the processing of the entire lot.) Substituting in (4.3.2-1) we get:

$$MFT^{LS} = \frac{\left(\sum_{i=1}^{N}\sum_{j=1}^{m} p_{[i],j}\right) + \left(\sum_{i=1}^{N} \frac{Q_{[i]}-1}{2} \cdot p_{[i],BN}\right)}{N} \tag{4.3.2-9}$$

Similarly, for the regular MFT we have:

$$MFT = \frac{\sum_{i=1}^{N}\sum_{j=1}^{m} Q_{[i]} \cdot p_{[i],j}}{N} \tag{4.3.2-10}$$

Thus, the ratio is:

$$\frac{MFT^{LS}}{MFT} = \frac{\left(\sum_{i=1}^{N}\sum_{j=1}^{m} p_{[i],j}\right) + \left(\sum_{i=1}^{N} \frac{Q_{[i]}-1}{2} \cdot p_{[i],BN}\right)}{\sum_{i=1}^{N}\sum_{j=1}^{m} Q_{[i]} \cdot p_{[i],j}} \tag{4.3.2-11}$$

And under the simplification of equal lots and a perfect balance ($Q_i = Q, p_{ij} = p \;\; \forall \text{i, j}$), it reduces to the following:

$$\frac{MFT^{LS}}{MFT} = \frac{N \cdot m \cdot p + N \cdot \frac{Q-1}{2} \cdot p}{N \cdot m \cdot Q \cdot p} = \frac{Q + 2 \cdot m - 1}{Q \cdot (2 \cdot m)} \tag{4.3.2-12}$$

In this case, as can be expected, the number of lots has no effect on the ratio. Moreover, the ratio for a large lot size is still as in the previous case, i.e., $\frac{1}{2 \cdot m}$. This implies that for dynamic arrival times of lots to the system, such that the waiting time of each lot before the first machine is negligible, the gains via lot streaming can, in fact, be decoupled and associated with the individual lots.

4.3.3 The WIP objective

To realize the benefits of lot-streaming with respect to the WIP objective, note that, as in the analysis of a single lot, at time zero, all the lots are awaiting processing. Thus, the WIP level is $\sum_{i=1}^{N} Q_{[i]}$. The first sublot of the first lot leaves the shop after $\sum_{j=1}^{m} p_{[1],j}$ time units. The next $\left(Q_{[1]} - 1\right)$ unit-sized sublots of the first lot each leave the shop after an additional $p_{[1],BN}$ time units. For each unit-sized sublot of the i-th lot, the time between inter-departures is $p_{[i],BN}$. If, for simplicity, we assume at this stage that the lots are of equal size and that the processing times are equal (i.e., perfect balance), the average WIP level under lot streaming is as follows:

$$WIP^{LS} = \frac{(N \cdot Q) \cdot (m \cdot p) + p \cdot \sum_{i=1}^{N \cdot Q-1} (N \cdot Q - i)}{m \cdot p + (N \cdot Q - 1) \cdot p} = \frac{N \cdot Q \cdot \left(m + \frac{N \cdot Q - 1}{2} \right)}{m + N \cdot Q - 1} \tag{4.3.3-1}$$

On the other hand, for the regular case (i.e., without lot streaming), we have:

$$WIP = \frac{(N \cdot Q) \cdot (Q \cdot m \cdot p) + (Q \cdot p) \cdot \sum_{i=1}^{N-1} (N - i)}{Q \cdot m \cdot p + (N - 1) \cdot Q \cdot p} = \frac{N \cdot Q \cdot \left(m + \frac{N - 1}{2} \right)}{m + N - 1} \tag{4.3.3-2}$$

Dividing the two, we obtain the following ratio:

$$\frac{WIP^{LS}}{WIP} = \frac{m + \frac{N \cdot Q - 1}{2}}{m + \frac{N - 1}{2}} \cdot \frac{m + N - 1}{m + N \cdot Q - 1} \tag{4.3.3-3}$$

As expected, Exp. (4.3.3-3) suggests that for large *N* (such that *N*>>*m*), the value approaches 1.00, i.e., no benefits via lot streaming. However, consider the following instance: $N = 10, Q = 10, m = 5$. The resulting ratio is 77.2%, i.e., a significant reduction of 22.8% in the average WIP level. Moreover, for a reasonable *N*, and a large *Q*, the ratio becomes:

$$\frac{1}{2} \cdot \frac{m + N - 1}{m + \frac{N - 1}{2}}$$

This ratio admits its lower bound, of $\frac{1}{2}$, when m>>N. This lower bound is identical to the one obtained for the single lot case, and indicates a huge opportunity for WIP reduction.

4.3.4 Summary of benefits

Table 4.1 provides a summary of the extent of the potential benefits expected, when utilizing lot streaming in multiple-batch flow-shop systems. The ratios shown in Table 4.1 have been obtained using the simplifying assumptions of equal lot sizes, equal processing times (i.e., perfect balance), and negligible transfer and setup times. Nevertheless, the bottom line is that, in the multiple-lot case, as in the single-lot case, lot-streaming is expected to significantly improve system performance.

Table 4.1. Benefits via lot streaming in a multiple batch flow shop, under equal lot sizes, equal processing times, and negligible transfer and setup times

Measure	Ratio	If: Large lot sizes, Small number of machines ($Q>>m$)	If: Large number of machines, Small lot sizes ($m>>Q$)
Makespan	$\dfrac{Q\cdot\sum_{i=1}^{N}p_{[i],BN}+\sum_{j=1}^{BN-1}p_{[1],j}+\sum_{j=BN+1}^{m}p_{[N],j}}{Q\cdot\left(\sum_{j=1}^{BN-1}p_{[1],j}+\sum_{i=1}^{N}p_{[i],BN}+\sum_{j=BN+1}^{m}p_{[N],j}\right)}$	$\dfrac{N}{\left[N+(m-1)\right]}$	$\dfrac{m-1}{Q\cdot\left[N+(m-1)\right]}$
Mean flow time	$\dfrac{\sum_{i=2}^{N}\sum_{r=1}^{i-1}Q_{[r]}\cdot p_{[r],1}+\left(\sum_{i=1}^{N}\sum_{j=1}^{m}p_{[i],j}\right)+\left(\sum_{i=1}^{N}\frac{Q_{[i]}-1}{2}\cdot p_{[i],BN}\right)}{\sum_{i=2}^{N}\sum_{r=1}^{i-1}Q_{[r]}\cdot p_{[r],1}+\sum_{i=1}^{N}\sum_{j=1}^{m}Q_{[i]}\cdot p_{[i],j}}$	$\dfrac{N}{N+2\cdot m-1}$	$1/Q$
Work-in-process	$\dfrac{m+\frac{N\cdot Q-1}{2}}{m+\frac{N-1}{2}}\cdot\dfrac{m+N-1}{m+N\cdot Q-1}$ (perfect balance)	$\dfrac{1}{2}\cdot\dfrac{m+N-1}{m+\frac{N-1}{2}}$	1

Table 4.2. Processing time data for the case study

Lot	**M_1**	**M_2**	**M_3 (BN)**	**M_4**	**M_5**	Q_i
1	4	2	6	4	5	7
2	1	6	1	6	8	15
3	5	8	9	8	2	11
4	10	5	10	4	2	11

Table 4.3. Summary of results for the case study

		Makespan [min]	**MFT [min]**	**Avg WIP [lots]**
With Lot-Streaming	I	318	224	2.81
Without Lot-Streaming	II	590	476	3.23
Improvement (%)	III	46	53	13
Actual Ratio	IV=I/II	.54	.47	.87
Theoretical Ratio (*)	V	.50	.31	.62

(*) assuming large lot-sizes

4.3.5 Case study

To demonstrate the significance of Table 4.1, a case study is presented next. The Surface Mount Technology (SMT) process, which was discussed in Chapter 1, is a common process used for the manufacture of Printed Circuit Boards (PCBs), utilizing several Pick and Place (P&P) machines. This process has been implemented in two ways. The traditional way has been to organize the machines, as stand-alone, in a flow-shop configuration. The more recent way has been to organize the machines in serial, connected via conveyors, in a production line configuration. In the former configuration, lots are transferred from one machine to the next upon completion of the entire lot while in the latter configuration, the PCBs of each lot are streamed one by one through the machines, i.e., unit-sized sublots are implemented.

Consider the instance of four PCB-lots, to be processed on five P&P machines, for which the unit processing times and the lot-sizes are provided in **Table 4.2**. The lots are prioritized as follows: {3-1-2-4}, and are processed in this sequence. **Table 4.3** presents a summary of the results with respect to the three measures examined earlier in this Chapter, namely makespan, MFT, and average WIP, and the percentage improvement made by the configuration that supports lot-streaming over the configuration that does not support lot-streaming. As indicated in Table 4.3, substantial improvements, in all the performance measures considered, are attained by utilizing lot-streaming. The makespan is cut down by 46%, approaching the theoretical ratio of 50%, which is obtained from Table 4.1, by assuming large lot sizes. The MFT is also reduced significantly, by 53%, while the WIP is reduced by 13%.

This case study demonstrates that, by utilizing lot streaming, two major benefits, which are of utmost importance to managers, can be accomplished. Firstly, better response to market demands is possible, if it is advantageous to have smaller portions of the lots available earlier. Secondly, production costs can be cut down. While meeting the same demand, less work-in-process is managed, at any time, under lot streaming, throughout the production system. However, to be able to fully take advantage of the benefits of lot streaming, setup and transfer times must be kept significantly small compared to processing times.

In the case of the SMT line, it is appropriate to assume that setup and transfer times are negligible. This is true since the P&P machines can be setup for the next lot, in parallel to the processing of a current lot, and the transfer is done while the machines are working. Transfer does not limit any of the machines since it is negligible compared to the machines processing times. In different cases, setup and transfer times may be incurred on the machines due to lot streaming. Their impact can roughly be estimated by adding the sublot setup and transfer times to the sublot processing times, for the case of lot streaming. Note, however, that, in these cases, while it may not be beneficial to stream the lots one-by-one, it may still be beneficial to stream them in larger portions (i.e., larger than unit-sized sublots) which are still smaller than the lot sizes.

4.4 The Lot Streaming Sequencing Problem (LSSP)

Having shown that substantial improvements can be made via lot-streaming in the multiple-lot case, we now wish to focus on the first problem, namely the lot streaming sequencing problem (LSSP). Our objective in the this Section is to develop efficient heuristics for this complicated sequencing problem. In the LSSP, the sublot size is pre-determined, and so the problem reduces to the following:

Given the sublot size, and the fact that lots are streamed through the m-machine flow-shop, what is the optimal sequence of the N lots with respect to the makespan objective?

Note that this is purely a sequencing problem. However, the additional complication here, compared to the traditional flow-shop sequencing problem, results from the fact that the lots are **streamed** through the system, therefore they do not keep their composition in every stage, as in the traditional sequencing problem.

The LSSP arises, for example, in the manufacture of PCBs along an SMT production line (see **Fig. 1.1** for illustration.) Lots of unprocessed PCBs are loaded onto the input buffer of the line according to some sequence, and are then streamed one by one through the production line. Thus, unit-sized sublots are utilized. Note that in the above application, only sequencing of the lots is required, i.e., the sublots are not intermingled. However, in general, this may not always be the case. Thus, we consider the less restrictive case of intermingling as well. To further illustrate the LSSP, consider the following numerical example. Four lots are to be processed on a five-machine flow-shop. The lot sizes and the unit-processing times of the lots on the machines are given in **Table 4.4**.

Table 4.4. Data for numerical example 4.1

Lot	M_1	M_2	M_3	M_4	M_5	Lot Size
1	1	6	8	7	4	15
2	10	10	7	2	6	14
3	10	4	2	4	8	6
4	4	2	9	6	4	5

The question is what should the sequence of the lots be, in order to minimize the makespan, given that they are to be streamed item-by-item through the flow-shop. Enumerating all the 24 feasible sequences and computing the corresponding makespans reveals that the optimal sequence is {1-3-4-2} with a makespan of 290, while the worst sequence is {3-2-1-4}, with a mekspan of 393 (35.3% above the optimum.) The average sequence in this instance results in a makespan which is 20.5% above the optimum. Thus, it is apparent that significant improvements to the makespan can be achieved by utilizing an effective solution approach to the problem of sequencing the lots.

Note that, in this example, only unit-sized-sublot sequences have been considered. Next, we show that, indeed, only unit-sized sublots need to be considered, as they are optimal for the cases of negligible setup and lot-attached setup. We do so by generalizing the argument of Vickson and Alfredsson (1992), for the case of no setup.

Theorem 4.1. There exists an optimal schedule (w.r.t. makespan) to the $N/m/F_T/C_{max}$ problem with lot-attached setups, in which all the sublots are of unit size, i.e., $L^* = 1$.

Proof (By contradiction)

Suppose the optimal schedule, S, has (at least) one sublot of L units (L>1). Construct another schedule, S', which is similar to S, except that instead of the L-unit sublot, unit-sized sublots are used (and transferred independently to the next machine upon completion.) Clearly, all the units of the original L-unit sublot are available at the next machine earlier, or at least at the same time, under schedule S'. Thus, S' will perform at least as good as S with respect to any regular measure of performance, in particular for C_{max}. QED.

4.4.1 The two-machine case

The two-machine case can easily be solved using Theorem 4.1 and Johnson's sequencing rule for the two-machine flow-shop. For the case of zero setup, the resulting optimal solution would be a non-intermingled sequence of unit-sized sublots, where for every two sublot types, *g* and *h*, sublots of type *g* precede sublots of type *h* iff:

$$\min\left\{p_{g,1}, p_{h,2}\right\} \leq \min\left\{p_{g,2}, p_{h,1}\right\} \qquad (4.4.1\text{-}1)$$

Note that this sequence would not necessarily be identical to the sequence obtained by utilizing Johnson's rule on the lot-processing times, i.e., by sequencing sublots of type *g* before sublots of type *h* iff:

$$\min\left\{Q_g \cdot p_{g,1}, Q_h \cdot p_{h,2}\right\} \leq \min\left\{Q_g \cdot p_{g,2}, Q_h \cdot p_{h,1}\right\} \qquad (4.4.1\text{-}2)$$

For example, consider two lot types, *g* and *h*, such that: $p_{g,1} \leq p_{g,2}$; $p_{g,1} \leq p_{h,1}$ $p_{h,1} \leq p_{h,2}$; and : $Q_h < \frac{Q_g \cdot p_{g,1}}{p_{h,1}}$. Under condition (4.4.1-1), lot type *g* would precede lot type *h* in the sequence. On the other hand, it can easily be verified that under condition (4.4.1-2), lot type *h* would precede lot type *g* in the sequence.

For the case of lot-attached setup, note that the lots are processed continuously, i.e., without intermingling. Therefore, only a slight modification to Johnson's rule, to account for the setup times, is required in this case to obtain the optimal (unit-sized sublot-) sequence. The optimal sequence is given in Theorem 4.2.

Theorem 4.2 For the two-machine flow shop with lot-attached setup times, lot type g would precede lot type h in the optimal sequence iff:

$$\min\left\{E_{g,1}, E_{h,2}\right\} \leq \min\left\{E_{g,2}, E_{h,1}\right\} \qquad (4.4.1\text{-}3)$$

where:

$E_{g,1}$ - The time for lot type g to be started on machine 2.

$E_{g,2}$ - The additional time required for lot type g to be completed on machine 2, after machine 1 has finished working.

Proof.

For a complete proof, see Baker (1995).

4.4.2 The general m-machine case

As indicated by Baker (1995), the result for the two-machine case does not, in general, extend to three machines or more, unless there are only two non-dominated machines. He summarizes that further research is needed to study heuristic solutions for the general case. In the next two Sections, two such heuristics are proposed.

Following Theorem 4.1, we are mainly interested in the optimality of sequences which utilize unit-sized sublots. Although this reduces the search space, the problem is still, as explained in Section 4.2, extremely difficult and time-consuming even for small-size problems, if explicit enumeration is to be used. Moreover, when considering the more general problem of sublot-intermingling, the solution space grows even further. Therefore, the heuristic approach is indeed the only practical approach. In the next two Sections, for each of the two heuristics that are developed and presented, the following is provided: (a) The motivation for the heuristic ; (b) The heuristic procedure ; (c) A comparison of the heuristic with available lower bounds, to evaluate its performance ; and (d) A utilization of the heuristic to evaluate the impact of changes in the problem parameters on the solution and other problem characteristics.

The first heuristic is based on the Fast Insertion Heuristic (FIH/NEH) which was introduced in detail in the literature as the best known heuristic for the regular flow-shop sequencing problem (Section 2.3). As mentioned there, it is also the only heuristic from among the several heuristics discussed there for regular flow-shop sequencing problems, which can efficiently be extended to lot-streaming.

The second heuristic, referred to as the Bottleneck Minimal Idleness (BMI) heuristic, introduces a new approach to the LSSP. The BMI heuristic is developed to specifically address cases in which lot-streaming is possible. It takes advantage of the special characteristics of the LSSP, in particular bottleneck-related characteristics.

4.5 The Modified FIH Heuristic for LSSP

4.5.1 The Fast Insertion Heuristic for LSSP (FIHLS)

The fast insertion heuristic was proposed by Nawaz et al. (1983) for the regular *n*-job *m*-machine flow-shop sequencing problem. It was described in detail in Section 2.3. This is the only heuristic that can efficiently be extended to account for lot-streaming since, applying this heuristic does not necessarily result in non-intermingled sequences (which are obviously sub-optimal.) However, a direct implementation of the heuristic to the case of lot-streaming is not possible and some modifications are necessary. The modified Fast Insertion Heuristic for Lot-Streaming (FIHLS) procedure is described next. There exists more than one way to modify the fast insertion heuristic to the case of lot-streaming. The modification that is adapted follows the intuition of the original heuristic, to consider the sublots that are candidates for insertion, in non-increasing order of the lots' total processing times. However, an alternative way would be to consider the sublots in non-increasing order of the sublots' total processing times. Obviously, the two need not result in the same sequence. In the latter case, the total processing time for a sublot would be:

$$T_i = \sum_{j=1}^{m} \left(su_{ij} + L \cdot p_{ij} \right) \tag{4.5-1}$$

This can be used, instead of exp. (4.5-2) given in step 2 of the modified FIHLS. Note, however, that in the latter, the original lot size, Q_i, has no effect on the priority given to the sublot, as opposed to the proposed modification. A special case in which the two alternatives coincide (i.e., result in the same sequence) is the case of equal lot sizes.

The Fast Insertion Heuristic for LSSP (FIHLS)

Step 1. For a given sublot size *L*, obtain the number of sublots of each lot type:

$$n_i = \left\lceil \frac{Q_i}{L} \right\rceil^+$$

and the size of the last sublot (if positive):

$$0 < R_i = Q_i - (n_i - 1) \cdot L < L$$

Step 2. For each lot i , compute the total processing time (including setup times):

$$T_i = \sum_{j=1}^{m} \left(su_{ij} + Q_i \cdot p_{ij} \right) \qquad (4.5\text{-}2)$$

Arrange the lots in a LIST, in non-increasing order of their T_i.

Step 3. Pick the first lot on the LIST and sequence its sublots in serial.

Step 4. Pick the next lot on the LIST. Sequence its sublots one-at-a-time using the fast insertion method for each sublot. That is, start with its first sublot. Check for the best position for this sublot along the partial sequence that had been established so far. Place the sublot in the best position and do the same for the other sublots of this lot.

Step 5. Repeat step 4, until all the lots have been sequenced.

Implementation of the FIHLS for Non-intermingled Sequences

Note that in order for the FIHLS algorithm to result in a non-intermingled sequence, Step 4 of the algorithm needs to be modified as follows:

Step 4. Pick the next lot on the LIST. Sequence its sublots one-at-a-time, in an unbroken string, using the fast insertion method. Check for the best position for this lot along the partial sequence that had been established so far. Place the lot in the best position.

4.5.2 The complexity of the FIHLS

While the complexity of the original heuristic is $O\left(N^2\right)$ (see Section 2.3.1), where N is the number of lots, the complexity of the modified heuristic is as follows. At each iteration of step 4, at most $\sum_{i=1}^{N} Q_i$ positions are considered and the step is repeated $O\left(\sum_{i=1}^{N} Q_i\right)$ times. Therefore, the complexity of the modified heuristic is $O\left(\left(\sum_{i=1}^{N} Q_i\right)^2\right)$. Considering lot sizes of the same order, the order of complexity can be simplified to the following: $O\left(N^2 \cdot Q^2\right)$. This implies that the modified heuristic can only qualify as a pseudo-polynomial heuristic since it also depends on the magnitude of the input data. However, for the case of no intermingling, the order of complexity remains $O\left(N^2\right)$, as in the original procedure.

4.5.3 Optimality of the FIHLS heuristic

A computer program in BASIC was written to test the FIHLS heuristic. The program is given in **Appendix B**. Before the results are presented we note that, in order to test the quality of the solutions generated by the FIHLS heuristic, we compare them with a lower bound on the optimal solutions. The two lower bounds discussed in Section 2.3.1 of the literature survey and summarized in Eqs. (2.3.1-4) and (2.3.1-5) can easily be extended for the flow-shop lot-streaming problem and are therefore used to test the quality of the FIHLS heuristic. The two bounds were also computed by the computer program and the maximum of the two was then utilized for comparison with the heuristic's solutions.

A six-machine flow shop was considered (m=6) to test the heuristic. The lot sizes were generated from a uniform distribution: $U[50,100]$. The processing times were generated from a uniform distribution as well: $U[5,15]$. Setup times were set as a percentage of the processing times of the lots. The following parameters were varied:

- The number of lots: N=3, 6, 10
- The setup time: 0% (no setup) and 5%
- The sublot size: L=100 (i.e., no lot streaming), 50, 25, 12, 6, 3, and 1.

A full factorial experiment was carried out. For each combination of the above factors, eight repetitions were created. The standard deviation of the results indicate that eight repetitions were indeed sufficient for the averages to be accurate to 0.1%. The following output was recorded for each repetition:

- The best sequence
- The resultant makespan of the sequence
- The ratio of the resultant makespan to the lower bound
- The CPU time

An example of the output format is provided in **Appendix C**. The results are summarized in **Appendix D**. Next, we wish to test the quality of the results obtained by the FIHLS heuristic. **Table 4.5** presents the average ratios of the best makespan to the lower bound for different numbers of lots and different sublot sizes, when setup times are ignored (i.e., 0%). **Table 4.6** presents similar results for the case of 5% setup times.

Table 4.5. Average makespan to lower bound ratios for different numbers of lots and different sublot sizes, when setup times are ignored (i.e., 0%).

Sublot Size	N=3	N=6	N=10
100	2.448	1.757	1.391
50	1.776	1.424	1.197
25	1.360	1.194	1.091
12	1.163	1.100	1.043
6	1.080	1.052	1.025
3	1.041	1.030	1.013
1	1.016	1.012	1.005

Table 4.6. Average makespan to lower bound ratios for different numbers of lots and different sublot sizes, for the case of 5% setup times.

#Sublots	N=3	N=6	N=10
100	2.493	1.757	1.521
50	1.804	1.451	1.298
25	1.402	1.241	1.159
12	1.211	1.155	1.111
6	1.130	1.124	1.103
3	1.104	1.118	1.082
1	1.058	1.066	1.046

The average computation times for obtaining a single result, for sublots of unit-size, for the case of *N*=6 lots and *N*=10 lots, in **Tables 4.5 and 4.6**, are 4.5 and 18.0 hours respectively, using a PC-P167 Mhz processor. This clearly indicates that the FIHLS heuristic can only be used to solve instances for which the lot sizes are relatively small. Medium to large lot-sizes make the heuristic impractical in terms of computation time.

Next, the effects of the sublot size, the number of lots, and the setup times on the makespan are discussed. The quality of the solutions obtained by FIHLS is examined as well. It can be seen from **Table 4.5** that as the sublot size decreases, the ratio of the makespan obtained by the FIHLS, to the lower bound, decreases as well. In fact, the ratio is only 1.6% above 1.000

for three lots with unit-sized sublots, and is (expected to be) less than that for a larger number of lots. This result indicates that:

- The FIHLS is near-optimal if sufficiently small sublot sizes are used.
- The machine-based lower bound, used for comparison with the FIHLS, is very tight for small sublot sizes.

For large sublot sizes, the ratio is by far larger than 1.000 and it is, therefore, impossible to draw conclusions about the optimality of the FIHLS in these cases. However, it does not suggest that the FIHLS is inefficient in these cases. It merely suggests that it is more beneficial to stream smaller portions of the lots than larger portions, because by further breaking the lots down, substantial additional reductions in the makespan are possible.

With regard to the effect of the number of lots, note that as the number of lots in the system increases, the sublot size required to achieve a near optimal solution (i.e., a ratio sufficiently close to 1.000) increases as well. For example, for 10 lots, even a sublot size of 6 items is sufficient in order to be within a 2.5% range from the lower bound. Note that there exists a general trend with respect to the number of lots in the system. As the number of lots increases, regardless of the sublot size, the makespan ratio to the lower bound decreases. This result is indeed true in general (see **Appendix E** for a proof.)

It is harder to conclude on optimality with regard to the case of setup times. This is due to the fact that the ratios are slightly worse than in the case of no setup, as can be seen in **Table 4.6**. While this result does not imply anything about the optimality of the FIHLS heuristic, as explained earlier, it does suggest that as the setup times increase, it becomes less attractive to use lot streaming since, the potential reductions in the makespan are smaller than in the case of negligible setup times. Experiments with larger portions of setup times (i.e., 10% and 20%) further confirm this conclusion.

To learn more about the effects of the number of lots and the setup times on the ratios of the makespan to the lower bound, **Fig. 4.2** depicts these ratios as a function of the number of sublots, for N=3, 6, and 10 lots in the system and under negligible (0%) and non- negligible (5%) setup times.

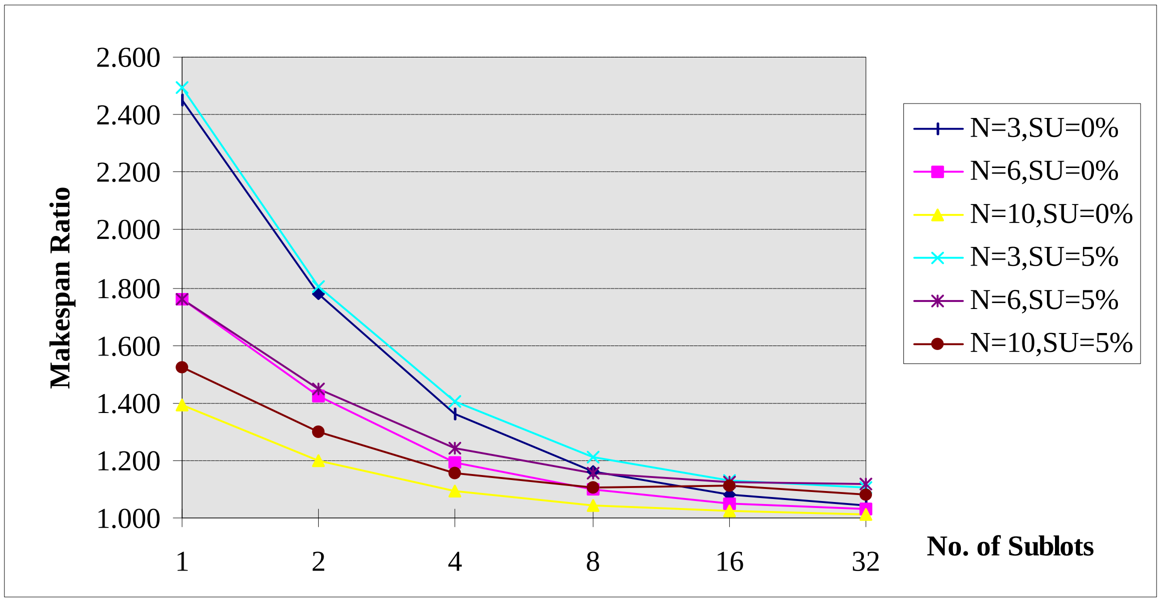


Fig. 4.2. The makespan ratios as a function of the number of sublots and the setup times.

From **Fig. 4.2**, it can be observed that the impact of setup on the makespan ratio becomes more noticeable as the number of lots in the system increases. While for N=3 and N=6, there is hardly any impact, in particular for large sublot sizes (or small number of sublots), the impact becomes noticeable for N=10 for all sublot sizes. Another important observation is that, although setup times tend to lessen the reduction in the makespan, still, for sufficiently small sublot sizes, the ratios tend to converge, in all cases, to the lower bound.

Next, we address the question of intermingling of different sublot types in the resultant sequences of the FIHLS solution. **Table 4.7** shows three typical sequences, corresponding to cases of 0% setup, 5% setup and 10% setup, for a sublot size of 3. It can be seen from Table 4.7 that, while the sublots are intermingled in the case of 0% setup, they become non-intermingled even when a relatively small setup is introduced. Experiments with setup times as high as 20% of the processing times strongly support this conclusion. Note that for the case of 5% setup, the lots are almost completely non-intermingled (with the exception of lot #2, which has been broken down to two non-intermingled strings of sublots.) However, for most of the cases of 5% setup, the lots appeared completely non-intermingled, as for all the cases of 10% setup.

Table 4.7. The impact of setup time on the amount of intermingling in the FIHLS solution

SUBLOT SIZE= 3   SETUP PERCENTAGE: 0 %
7 2 7 2 6 2 7 1 5 10 10 10 10 10 10 8 10 8 8 5 10 8 8 5 10 8 5 8 8 5 10 8 8 5 10 8 5 8 8 5 10 8 5 8 10 8 5 8 8 5 10 8 5 8 10 8 5 8 5 8 10 8 5 8 5 10 8 8 5 8 1 1 5 1 1 10 8 5 4 1 8 4 5 4 1 1 1 1 1 5 1 1 1 1 5 1 1 1 1 1 5 1 10 8 4 1 1 5 4 1 6 6 5 6 6 6 5 6 6 6 5 6 6 6 5 6 6 6 5 6 6 6 5 6 6 6 5 6 6 6 5 6 6 6 5 6 6 3 3 4 3 4 4 3 7 4 7 4 7 4 3 2 6 3 3 4 7 3 7 4 3 2 3 4 3 2 6 3 3 4 3 4 7 3 2 3 3 2 6 3 3 3 2 7 7 1 7 3 7 9 3 9 3 9 9 2 3 9 9 9 2 3 9 9 3 9 9 9 2 3 9 9 3 9 9 9 9 2 9 9 2 9 2 2 10 6 1 2 4 3 7 3 2 7 4 7 7 7 7 7 7 7 7 7 7 7
SUBLOT SIZE= 3   SETUP PERCENTAGE: 5 %
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 2 2 2 2 2 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2
SUBLOT SIZE= 3   SETUP PERCENTAGE: 10 %
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

4.5.4 Limitations and drawbacks of the FIHLS

A couple of important observations regarding the FIHLS, as applied to the LSSP, can be made based on the above analysis. First, and foremost, the modified heuristic essentially becomes impractical to use for the cases of intermingling and of large lot-sizes (i.e., 100's or 1000's of items.) The computational requirements for such cases are so highly time-consuming that they cannot be obtained in a reasonable time under the current computer capabilities. Second, the heuristic attempts to schedule the larger lots first regardless of bottleneck considerations, i.e., regardless of whether the resulting schedule fully utilizes the bottleneck machine. It appears, however, that, as opposed to traditional lot sequencing, the bottleneck machine does play a key role under lot streaming. This is further demonstrated in the following discussion of the second heuristic proposed for the LSSP, namely the Bottleneck Minimal Idleness (BMI) heuristic. The BMI heuristic is specifically designed to resolve the limitations of the FIHLS. It is developed as a fast single-pass heuristic which strives for constructing a schedule that fully utilizes the bottleneck machine.

4.6 The Bottleneck Minimal Idleness (BMI) Heuristic for the LSSP

The Bottleneck Minimal Idleness (BMI) heuristic, that is presented next, is based solely on properties related to the bottleneck machine. To that end, it exploits special features of the LSSP, that are not dominant in regular flow-shop problems. The BMI heuristic constructs a schedule which attempts to minimize the idle time on the bottleneck machine. It is motivated by the fact that in the LSSP, when small sublot sizes are utilized, the time it would take for the first sublot to get to the bottleneck machine, and the time it would take for the last sublot on the bottleneck machine to leave the shop, are negligible relative to the total processing time of the entire set of lots on the bottleneck machine. Thus, minimizing the idle time on the bottleneck machine must necessarily result in a near-optimal schedule. In the sequel, we first develop the BMI heuristic for the case of no intermingling. Modifications are then proposed to account for intermingling.

4.6.1 Preliminaries

We wish to start with some preliminaries that shed light on the motivation for the BMI heuristic. Let *BN* denote the bottleneck machine:

$$BN = \arg\max_{j} \left\{ \sum_{i=1}^{N} \left(su_{ij} + Q_i \cdot p_{ij} \right) \right\}$$

Then, any actual makespan of a given schedule for the LSSP can be prescribed as follows:

$$M^{LS} = \sum_{i=1}^{N} \left(su_{i,BN} + Q_i \cdot p_{i,BN} \right) + IT + \sum_{j=1}^{BN-1} \left(su_{[1],j} + L \cdot p_{[1],j} \right) + \sum_{j=BN+1}^{m} L \cdot p_{[l[,j} \qquad (4.6\text{-}1)$$

where:

IT	- The resultant idle time on the bottleneck machine, possibly because of delaying sublots.
L	- The sublot size considered.
$p_{[1],j}$	- The processing time per unit of the first sublot in the sequence on machine j.

$p_{[l],j}$ - The processing time per unit of the last sublot in the sequence, l, on machine j ; $l = \sum_{i=1}^{N} n_i$

The first term of exp. (4.6-1) represents the processing time of all the lots on the bottleneck machine. The second term, *I,* represents the idle time on the bottleneck machine (counted from the moment it had begun processing), possibly due to delaying some sublots, as will be explained later. The third term represents the time it takes for the first sublot to get to the bottleneck machine while the fourth term represents the time it takes for the last sublot on the bottleneck machine to leave the system.

Note that, if the actual makespan is obtained by a certain critical path which does not include the entire set of sublots on the bottleneck machine, then one (or more) sublot(s) on the bottleneck machine can arbitrarily be delayed, by creating bottleneck idle times in between them, so as to create an additional (artificial) critical path, equal in its length to the original one, but includes the entire set of sublots on the bottleneck machine. By doing so, the makespan is not worsened but, *IT* increases, to account for the additional delay on the bottleneck machine that has been added to it.

Based on expression (4.6-1), a lower bound on the makespan of any LSSP would be as follows:

$$LB^{LS} = \sum_{i=1}^{N} \left(su_{i,BN} + Q_i \cdot p_{i,BN} \right) + \min_{S} \{IT\} + \min_{i} \left\{ \sum_{j=1}^{BN-1} \left(su_{ij} + L \cdot p_{ij} \right) \right\} + \min_{i} \left\{ \sum_{j=BN+1}^{m} L \cdot p_{ij} \right\} \qquad (4.6\text{-}2)$$

where:

S - The set of all feasible sequences.

The first term in exp. (4.6-2) is identical to its parallel in exp. (4.6-1). However, the second term represents the minimal idle time (that cannot be avoided) on the bottleneck machine from among the set of all feasible schedules, *S*. Since it is a difficult task to estimate the value of this term, it is typically estimated simply as zero (which is clearly an underestimate.) The third term represents the minimal time it would take the first sublot to get to the bottleneck machine while the fourth term represents the minimal time it would take the last sublot on the bottleneck machine to leave the system. Exp. (4.6-2) admits a lower bound on the makespan since its terms are all less than or equal to their respective components in exp. (4.6-1) for every feasible schedule.

Viewing expressions (4.6-1) and (4.6-2), of the actual makespan and the lower bound on the makespan, respectively, we wish to make the following comment. Obviously, a schedule can be constructed, such that its first sublot would be the one with the least total processing time to reach the bottleneck while its last sublot would be the one with the least total processing time remaining after being completed on the bottleneck. However, this may alter the schedule for which the bottleneck idleness is minimized, thereby creating some additional idle time on the bottleneck machine. A question regarding the trade off then arises - Is it better to alter the schedule and absorb additional idle time on the bottleneck in order to save time at the beginning and at the end of the schedule? Note that this question remains in tact for any sublot size because for smaller sublot sizes, both the idle time and the time savings at the beginning and at the end of the schedule are expected to decrease.

The above discussion stresses the possible use of the bottleneck machine in developing a lower bound on the makespan. It can be seen from the bound in Exp. (4.6-2) that a schedule which achieves minimal bottleneck idleness may perform very well under lot streaming as sublots decrease in size, i.e., as L decreases. This is summarized in the following theorem.

Theorem 4.3. For any given sequence (in which some idle time exists on the bottleneck machine), bottleneck idle time decreases monotonically with decrement in the sublot size, L (the idle time is measured from time 0.)

Proof.

The proof is straightforward. Since the makespan (as any other regular performance measure) is a decreasing function of the sublot processing times (and consequently, the sublot sizes), and we have the following equivalence holding:

$$\min\{M_{BN}\} \Leftrightarrow \min\left\{\sum_{i=1}^{n} Q_{[i]} \cdot p_{[i],BN} + I\right\} \Leftrightarrow \min\{I\}$$

Then, it follows that I also decreases with decrement in the sublot size. QED.

To further enhance the result in theorem 4.3, let us focus on the *i*-th lot in a given sequence, for which there exists some idle time on the bottleneck machine, i.e., $I_{[i]} > 0$. Let $Q_{[i]}$ be the *i*-th lot size and $p_{[i],j}$ $(1 \le j \le BN)$ be the unit processing times of that lot on the machines. Thus, we have:

$$I_{[i]} = C_{[i],BN-1} - C_{[i-1],BN} > 0 \tag{4.6-3}$$

Assuming there is no idleness prior to processing the i-th lot on the (*BN-1*)-th machine, we have:

$$C_{[i],BN-1} = C_{[i-1],BN-1} + Q_{[i]} \cdot p_{[i],BN-1} \tag{4.6-4}$$

Substituting (4.5-4) in (4.5-3), we get:

$$I_{[i]} = \left(C_{[i-1],BN-1} - C_{[i-1],BN}\right) + Q_{[i]} \cdot p_{[i],BN-1} > 0 \tag{4.6-5}$$

Note that the first term in (4.6-5) (in parentheses) is strictly negative. It is the processing time of the entire *i*-th lot on the machine prior to the bottleneck that creates idle time on the bottleneck.. If instead, the portion of the *i*-th lot that had been completed on the machine prior to the bottleneck until time $C_{[i-1],BN}$ (from which idle time starts) would have been immediately transferred to the bottleneck machine then only a smaller idle time, if at all, would have been created on the bottleneck machine, resulting from the new (smaller than the original) portion of the *i*-th lot. Moreover, for a sublot size of:

$$L = \frac{C_{[i-1],BN} - C_{[i-1],BN-1}}{p_{[i],BN-1}}$$

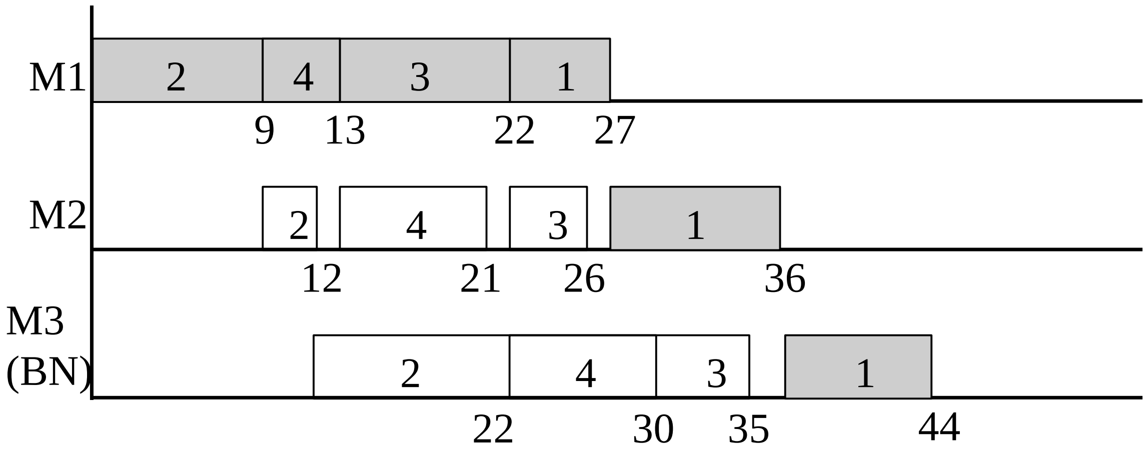
we would have had: $I_{[i]} = 0$, and idle time, if at all, would have been only due to a smaller portion of the original lot, $Q_{[i]} - L$. Therefore, by splitting the original lot to smaller portions, of size *L*, expression (4.6-5) suggests that the idle time could be reduced and, possibly even be completely removed. It can easily be shown that any idle time created on the bottleneck in between the sublots of the i-th lot is strictly less than the idle time originally created by the entire i-th lot, since sublots overlap in their processing while the original lot is not.

To further illustrate how the idle time decreases on the bottleneck machine under lot streaming, consider the following numerical example. The processing time data is given in **Table 4.8**.

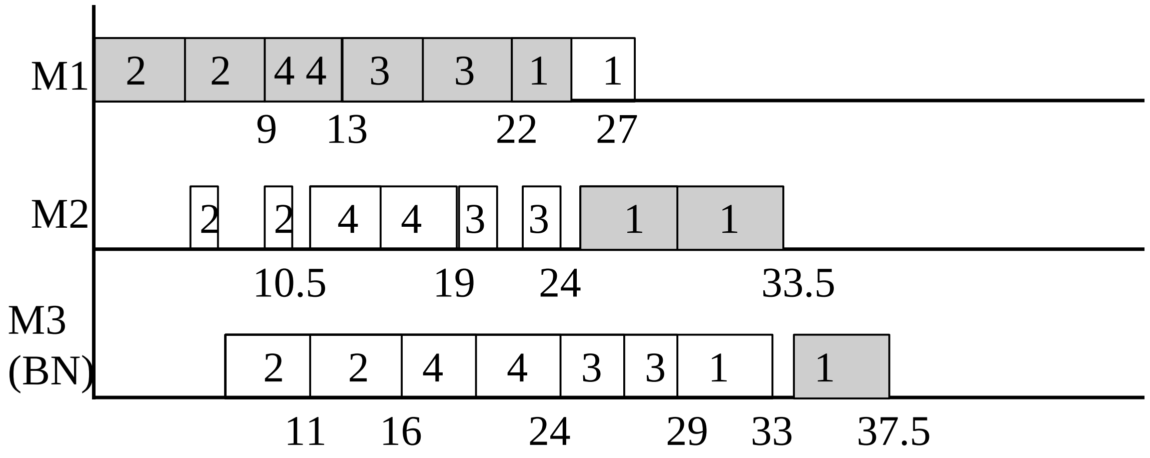
Table 4.8. Data for numerical example 4.2

	Machines					
Lots	**1**	**2**	**3 (BN)**	**4**	**5**	**Total**
1	5	9	8	10	1	33
2	9	3	10	1	8	31
3	9	4	5	8	6	32
4	4	8	8	7	2	29
Total	27	24	31	26	17	

Figure 4.3 depicts the schedules for (a) the original problem, and (b) the problem when lot streaming is considered, assuming each lot is comprised of two items. The critical path which determines the makespan is highlighted in both cases.



(a) The original problem



(b) the problem when lot streaming is considered

Fig. 4.3. The effect of lot streaming on bottleneck idle time

From **Fig. 4.3**, it can be seen that while the original schedule has an idle time interval of 1 time unit on the bottleneck machine, between [35,36], the LS-schedule has an idle time interval of 0.5 time units on the bottleneck machine, between [33,33.5]. Furthermore, observe that while in the original schedule processing on the bottleneck machine begins only at time 12, in the LS-schedule it begins at time 6. Therefore, the total time saved on the bottleneck machine is 6.5 time units, which is also the difference in the makespans of the two schedules. Another observation that can be made is that, the critical path (CP) has changed under lot streaming. While in the original schedule lot type 1, in its entirety, is a part of the CP, under lot streaming the second sublot of lot type 1 is no longer on the CP.

Next, a numerical illustration is provided to show that if a zero bottleneck idleness schedule can be constructed, it must be near-optimal. Then, in the Sections to follow a thorough analysis of bottleneck properties is carried out in an attempt to produce a heuristic which constructs a minimal (possibly zero) bottleneck idleness schedule.

4.6.2 Numerical illustration

A numerical illustration is provided next to clarify what is meant by 'sufficiently small' sublots and 'near-optimal' schedules alluded to in the discussion thus far. Suppose that by using sublots of size L, a schedule can be constructed for which there does not exist any bottleneck idle time, i.e., $I = \min_S \{IT\} = 0$. Let $\overline{Q}, \overline{p}, \overline{p}_{BN}$ denote the average lot size, the average unit processing time, and the average unit processing time on the bottleneck machine $\left(\overline{p} < \overline{p}_{BN}\right)$, respectively. For simplicity, setup times are ignored in the following analysis. Then, expression (4.6-1), updated for the average actual makespan, becomes:

$$M^{LS} = \sum_{i=1}^{N}\left(Q_i \cdot p_{i,BN}\right) + \sum_{j=1}^{BN-1} L \cdot p_{[1],j} + \sum_{j=BN+1}^{m} L \cdot p_{[l[,j} \approx N \cdot \overline{Q} \cdot \overline{p}_{BN} + (m-1) \cdot L \cdot \overline{p} \qquad (4.6\text{-}6)$$

Similarly, expression (4.6-2), for the lower bound on the makespan, becomes:

$$LB = \sum_{i=1}^{N}\left(Q_i \cdot p_{i,BN}\right) + \min_i\left\{\sum_{j=1}^{BN-1} L \cdot p_{ij}\right\} + \min_i\left\{\sum_{j=BN+1}^{m} L \cdot p_{ij}\right\} \geq N \cdot \overline{Q} \cdot \overline{p}_{BN} + (m-1) \cdot L \cdot p_{\min} \qquad (4.6\text{-}7)$$

From (4.6-6) and (4.6-7), the ratio of the makespan of the average LS-schedule (which does not have bottleneck idle time), to the lower bound, is as follows:

$$\frac{M^{LS}}{LB} \leq \frac{N \cdot \overline{Q} \cdot \overline{p}_{BN} + (m-1) \cdot L \cdot \overline{p}}{N \cdot \overline{Q} \cdot \overline{p}_{BN} + (m-1) \cdot L \cdot p_{\min}} \qquad (4.6\text{-}8)$$

The ratio that is given by Exp. (4.6-8) enables to determine how 'sufficiently' small the sublot size should be in order for the difference to become 'negligible' and, consequently, for the schedule to be considered 'near-optimal'. For example, consider the following numerical data:

$$\begin{aligned} &N = 3,\ 6,\ \text{and 10 lots}, \\ &m = 6 \text{ machines},\ Q_i \sim [50{,}100],\ p_{ij} \sim [5{,}15] \end{aligned} \qquad (4.6\text{-}9)$$

From the data, we have: $\overline{Q} = 75$, $\overline{p} = 10$, $p_{\min} = 5$. For simplicity, let $\overline{p}_{BN} = \overline{p}$ (higher value for $\overline{p}_{BN}$ would result in even lower values for the ratio.) **Table 4.9** summarizes the ratios, obtained for the different numbers of lots and for the different sublot sizes.

Table 4.9. The effect of sublot size on the optimality ratio of a zero bottleneck idleness schedule

Sublot Size	100	50	25	12	6	3	1
Ratio, N=3	1.5263	1.3571	1.2174	1.1176	1.0625	1.0323	1.0110
Difference (%)	52.63	35.71	21.74	11.76	6.25	3.23	1.10
Ratio, N=6	1.3571	1.2174	1.1220	1.0625	1.0323	1.0164	1.0055
Difference (%)	35.71	21.74	12.20	6.25	3.23	1.64	0.55
Ratio, N=10	1.2500	1.1429	1.0769	1.0385	1.0196	1.0099	1.0033
Difference (%)	25.00	14.29	7.69	3.85	1.96	0.99	0.33

Note that the difference prescribed in **Table 4.9** is the maximal difference (i.e., an upper limit on the difference), since we have used an underestimate for the lower bound. It could have been in fact higher, which would have, in turn, made the optimality ratio even smaller. From the Table it can be seen that, for N=10, while the schedule which uses the original lot sizes may end up (in the best case) with a makespan which is 25% higher than the lower bound, the schedule which would use sublots of size 6 would end up with a makespan which is less than 2% from the lower bound. In that respect, one might consider $L=6$ to be sufficiently small to make the difference negligible.

Note that it has been assumed in the numerical example that, for any sublot size, schedules which achieve zero idle time on the bottleneck are readily available. While this task is easier to accomplish when small sublot sizes are in use, it is almost an impossible task to accomplish for the original problem (i.e., without lot-streaming.)

Lastly, recall that the same data which has been utilized in the numerical example has also been used in the previous Section to test the FIHLS. Therefore, the difference between the two ratios, obtained here and there, can serve as a good indicator of the estimated idle time on the bottleneck machine in each case. The differences are provided in **Table 4.10**.

Table 4.10. Estimated idle time (%) in LS-schedules for numerical data 4.3

Sublot Size	100	50	25	12	6	3	1
N=3 (*)	52.63	35.71	21.74	11.77	6.25	3.23	1.10
(**)	144.83	77.56	35.95	16.28	8.02	4.13	1.60
(***)	92.19	41.85	14.21	4.51	1.77	0.90	0.50
N=6 (*)	35.71	21.74	12.20	6.25	3.23	1.64	0.55
(**)	75.74	42.38	19.42	9.98	5.16	2.96	1.20
(***)	40.02	20.65	7.22	3.73	1.93	1.32	0.65
N=10 (*)	25.00	14.29	7.69	3.85	1.96	0.99	0.33
(**)	39.05	19.69	9.14	4.34	2.49	1.31	0.50
(***)	14.05	5.40	1.45	0.49	0.53	0.32	0.17

(*) Theoretical value, based on exp. (4.6-8)

(**) Actual value, based on the FIHLS heuristic

(***) Expected idle time on the bottleneck machine under the FIHLS heuristic

As evident from **Table 4.10**, bottleneck idle time occurs in all cases without exception. However, it is drastically reduced with decrements in the sublot size and/or with increments in the number of lots. For example, bottleneck idle time in the case of no lot-streaming (i.e., L=100) reduces from 92% for 3 lots to as low as 14% for 10 lots. As the sublot size decreases, the bottleneck idle time is further reduced, and is less than 1% for cases of L=3, and 1.

Note that another benefit of the derivation of the bottleneck-based bound is that it acts as an even tighter bound than the machine-based bound previously utilized to evaluate the FIHLS. Actually, it is an extension of the machine-based bound to the bottleneck machine in which, in addition to accounting for the total processing time of all the lots on the bottleneck, the processing times of the first sublot to the bottleneck and the last sublot from the bottleneck are considered as well.

From the discussion thus far, it should now be apparent that the sequence which minimizes the idle time on the bottleneck machine is a near-optimal sequence. As mentioned in Chapter 2, the Optimized Production Technology (OPT) method postulates that the best schedule would be obtained by constructing it from the bottleneck machine backwards, i.e., by sequencing the sublots such that the bottleneck machine would not idle in between any two consecutive sublots. Hence, OPT strives for a schedule that would minimize bottleneck idleness but no explicit algorithm has been reported in the public domain that explains how to accomplish this task. As was shown earlier, sequencing of lots at the bottleneck is a difficult problem to solve. In the following section, a bottleneck-based analysis under lot streaming of unit-sized sublots is carried out in an attempt to produce a heuristic which would construct minimal bottleneck idleness schedules.

4.6.3 Bottleneck-based analysis

We first show that the problem of finding the schedule which minimizes the bottleneck idleness is difficult-to-solve and can, in fact, be reduced to the problem of finding the minimizing makespan schedule, which is known to be NP-hard for *m*>2.

Lemma 4.1. The problem of finding the minimal bottleneck idleness schedule is at least as hard as the problem of finding the minimal makespan schedule.

Proof.

The proof follows from the definitions of the bottleneck idleness and the makespan. Let [*i*] denote the *i*-th position in the sequence. Also, let:

$I_{[i]}$ - Idle time on the bottleneck machine preceding the i-th lot in the sequence.

$C_{[i],j}$ - Completion time of the i-th lot in the sequence on machine j.

Let $BN \geq 3$. Then, to minimize the makespan over *BN* machines, we require:

$$\min\{M\} = \min\left\{\max\left\{C_{[n],BN-1}, C_{[n-1],BN}\right\} + p_{[n],BN}\right\} \tag{4.6-10}$$

Since $p_{[n],BN}$ is a constant, exp. (4.5-10) is equivalent to the following:

$$\min\left\{\max\left\{C_{[n],BN-1} - C_{[n-1],BN}, 0\right\} + C_{[n-1],BN}\right\} \tag{4.6-11}$$

On the other hand, to minimize the bottleneck idleness, we require:

$$\min\{IT\} = \min\left\{\sum_{i=2}^{n} I_{[i]}\right\} = \min\left\{\sum_{i=2}^{n} \max\left\{C_{[i],BN-1} - C_{[i-1],BN}, 0\right\}\right\} \tag{4.6-12}$$

The similarity of the two problems is clear from the last two expressions. While the makespan problem involves minimizing the maximum of an expression, the bottleneck problem involves minimizing the summation of the maximum of the same expression. Therefore, it can be reduced to the makespan problem by assuming that all the lots in any sequence, except the last one, do not incur any idleness on the bottleneck. QED.

Although in general, it is a hard problem identifying the minimizing bottleneck idleness schedule, it becomes easier if certain conditions apply. These special conditions are discussed next. The following theorem plays a key role in the BMI heuristic. It addresses the issue of creating a schedule which would also minimize the bottleneck idle time under lot streaming.

Theorem 4.4. (“Bottleneck Dominance Property”) *If, for some lot i, the difference:* $p_{i,BN} - \max_{1 \le j < BN}\{p_{i,j}\}$ *is nonnegative then, under lot streaming, there would be no idle time created between the sublots of lot i on the bottleneck machine.*

Proof.

Consider the first sublot of lot *i*, with a size *L*. The processing time of that sublot on the bottleneck is: $L \cdot p_{i,BN}$. The latest that the second sublot can be started on the machine preceding the bottleneck is equal to the time the first sublot is completed on that machine (and subsequently started on the bottleneck.), say time *T*. Therefore, the latest the second sublot is completed on the machine preceding the bottleneck is:

$$T + L \cdot \max_{1 \le j < BN}\{p_{i,j}\}.$$

However, the earliest the first sublot is completed on the bottleneck is at time:

$$T + L \cdot p_{i,BN} \ge T + L \cdot \max_{1 \le j < BN}\{p_{i,j}\}$$

Thus, the second sublot becomes available at the bottleneck before or immediately when the first sublot is completed, hence no idleness is created in between the sublots. The same argument applies for the rest of the sublots of lot *i*. QED.

Remark 4.1. If the bottleneck dominance property holds for all the lots, then any sequence results in zero bottleneck idle time between the sublots, under lot streaming.

Remark 4.2. For every lot which satisfies the bottleneck dominance property, bottleneck idle time can occur only before or after the entire lot (but not in between its sublots.)

Remark 4.3. If in addition to the bottleneck dominance property, the regular schedule (i.e., without lot streaming) results in zero bottleneck idle time between the lots, then any LS-schedule would result in zero bottleneck idle time.
(The proof is trivial.)

Theorem 4.5. If for any two consecutive lots, the first lot is "bottleneck dominant" and the following holds:

$$\left(Q_{[1]} - L\right) \cdot \left(p_{[1],BN} - p_{[1],k}\right) \geq L \cdot \left(\sum_{j=k}^{BN} p_{[2],j} - \sum_{j=k+1}^{BN} p_{[1],j} \right) + \Delta_k$$

where $k = \underset{1 \leq j < BN}{\arg\max} \left\{ p_{i,j} \right\}$ *i.e., k is the machine with the second largest processing time for the first lot, and* Δ_k *is the idle time (if any) before starting the first sublot of the second lot on machine k, then there would be no idle time created on the bottleneck machine between the two lots.*
Proof.
The completion time of the last sublot of the first lot on the bottleneck machine would be:

$$C_{[1],BN} = L \cdot \sum_{j=1}^{BN} p_{[1],j} + \left(Q_{[1]} - L\right) \cdot p_{[1],BN}$$

Similarly, the completion time of that sublot on machine *k* would be:

$$C_{[1],k} = L \cdot \sum_{j=1}^{k} p_{[1],j} + \left(Q_{[1]} - L\right) \cdot p_{[1],k}$$

Therefore, the time difference in the completion of the first lot on machines *k* and *BN* is:

$$L \cdot \sum_{j=k+1}^{BN} p_{[1],j} + \left(Q_{[1]} - L\right) \cdot \left(p_{[1],BN} - p_{[1],k}\right)$$

The largest completion times of the first sublot of the second lot on the machines occur when the largest processing time for this lot occurs before machine *k*, causing a possible idle time of Δ_k on machine *k*. Accounting for the entire processing time of that sublot from thereon (i.e., assuming the sublot is critical on all the machines from machine *k* on), we have a total time of: $L \cdot \sum_{j=k}^{BN} p_{[2],j}$ for the sublot to be completed on the bottleneck machine. If that time, in addition to the idle time on machine *k* (Δ_k) is still less than the time difference in the completion of the first lot, then there would be no idle time encountered on the bottleneck machine. That is to say that the following should hold:

$$L \cdot \sum_{j=k+1}^{BN} p_{[1],j} + \left(Q_{[1]} - L\right) \cdot \left(p_{[1],BN} - p_{[1],k}\right) \geq L \cdot \sum_{j=k}^{BN} p_{[2],j} + \Delta_k$$
$$\Leftrightarrow$$
$$\left(Q_{[1]} - L\right) \cdot \left(p_{[1],BN} - p_{[1],k}\right) \geq L \cdot \left(\sum_{j=k}^{BN} p_{[2],j} - \sum_{j=k+1}^{BN} p_{[1],j}\right) + \Delta_k$$

<u>QED</u>.

Several comments should be made in regard to the above theorem. First, note that the same condition can be applied recursively for the second bottleneck machine, the third, and so forth, until machine 1 has been reached, for which we know that: $\Delta_1 = 0$. Using the theorem in a recursive manner is equivalent to verifying whether there exists an idle time between the two lots on the bottleneck machine.

Second, we stress the importance of positive dominance. Note that if, for the bottleneck dominant lot, we have: $p_{[1],BN} - p_{[1],k} = 0$, then Q_i has no effect whatsoever on whether or not an idle time would be created between the two lots on the bottleneck machine. However, as long as the difference is strictly positive (≥ 1, for integer processing times) the larger the lot size, the less likely it is for an idle time to occur between the two lots.

Lastly, note that, regardless of the processing times, the largest value of the left-hand-side and the smallest value for the right-hand-side of the inequality are both attained when $L = 1$. This indicates that, in order to minimize the bottleneck idle time, it is always best to consider unit-sized sublots (in agreement with Theorem 4.1)

Next, the issue of sequencing bottleneck-dominant lots is considered. Recall that the purpose is to obtain a sequence which minimizes bottleneck idleness. To do so, note that this can be accomplished by maintaining as large a time difference (or “time buffer”) as possible between the bottleneck machine and the one preceding it while constructing the sequence since, the larger the time difference, the less likely it is for the next lot to cause bottleneck idleness. In order to achieve a large time buffer, it may be very beneficial to sequence lots in decreasing order of closeness of the secondary bottleneck machine to the primary bottleneck machine. By ‘secondary bottleneck machine’ we refer to the upstream machine with the next largest processing time after the bottleneck machine (which is referred to as the ‘primary bottleneck machine’.) A lexicographic rule that fully utilizes the machines from the bottleneck backwards is proposed in order to maximize the so called time buffer between the bottleneck machine and the one preceding it. To illustrate how the proposed lexicographic rule works, consider the following example. Four lots, with their associated processing times and lot sizes on five machines (with the last machine being the bottleneck machine) are given in **Table 4.11**.

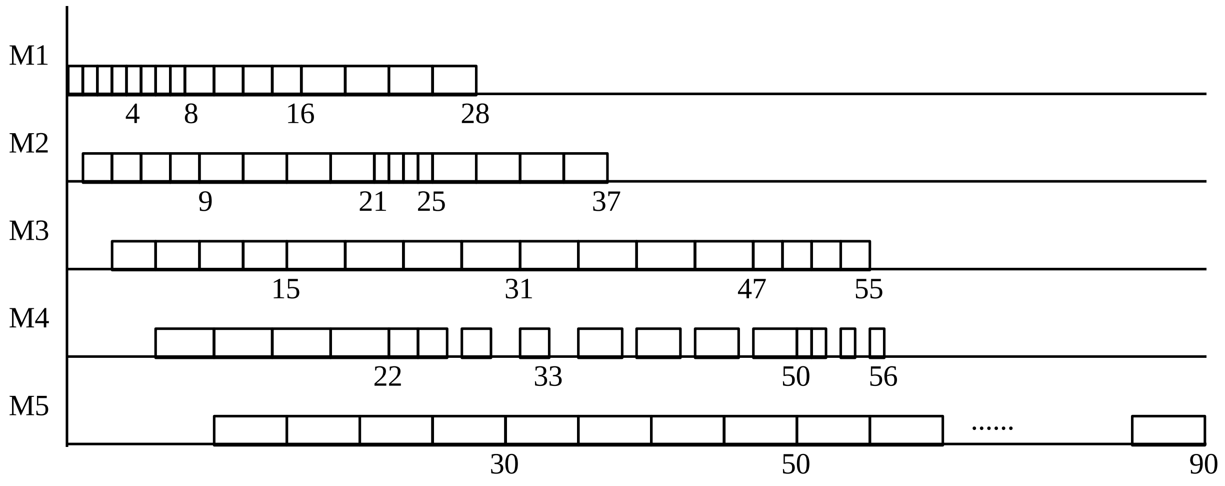
For simplicity, we consider equal-sized lots ($Q_i = 4 \ \forall i$). Note that all the lots are bottleneck dominant (i.e., machine 5 has the largest processing time.) The question is how should the lots be sequenced in order to maximize the time buffer between M_{BN} and M_{BN-1}(or, alternatively, minimize the idle time on M_{BN-1}.) Unit-sized sublots shall be used in this example.

Table 4.11. Data for numerical example 4.4

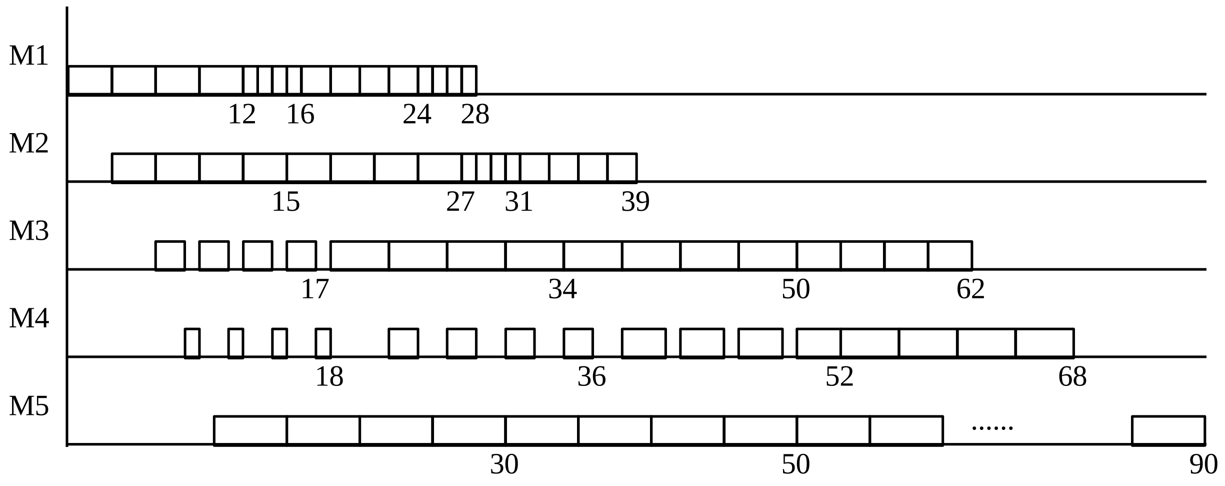
Lot	**M$_1$**	**M$_2$**	**M$_3$**	**M$_4$**	**M$_5$ (BN)**	Q_i
1	1	2	3	4	5	4
2	1	3	4	2	5	4
3	2	1	4	3	5	4
4	3	3	2	1	5	4

To solve using the lexicographic rule, note that secondary bottleneck machines are machines 4, 3, 3, and 2 for lots 1, 2, 3, and 4, respectively. The lexicographic rule would place lot 1 first in the sequence, because its secondary bottleneck machine is the closest to the primary bottleneck machine. Next, there is a tie between lots 2 and 3 (both having machine 3 as their secondary bottleneck machine.) Therefore, to determine which of these two lots is sequenced next, their subsequent bottleneck machines are compared. It can be seen that, for lot 2, machine 2 is the next bottleneck machine while, for lot 4, it is machine 1. Hence, the sequence continues with lot 2 followed by lot 3. To complete the sequence, lot 4 is sequenced last. The resultant sequence is thus {1-2-3-4}. The corresponding schedule of unit-sized-sublots is given in **Fig. 4.4(a)**.

This schedule achieves a time buffer of: 90-56=34 time units. Note how the first two sublots of the second lot on machine 4 take only two time units each (instead of three time units, as the last two sublots of this lot) due to their earlier availability. In other words, the idle time was absorbed for the first two sublots. This availability opportunity was created by scheduling a lot (in this example, lot 1) which has M_4 as its secondary bottleneck beforehand, to create a queue of work before the machine. Similarly, observe that idle time is also avoided for the first two sublots of lot 4 on M_4.



(a) The schedule corresponding to the lexicographic rule-based sequence {1-2-3-4}.



(b) The schedule corresponding to the arbitrary sequence {4-2-3-1}.

Fig. 4.4. A lexicographic rule-based sequence versus an arbitrary sequence

To appreciate the effect of the lexicographic rule, consider the following arbitrary sequence, obtained by exchanging lots 1 and 4 (thereby not sequencing them according to the lexicographic rule.) The resultant schedule is presented in **Fig. 4.4(b)**. Note that by exchanging lots 1 and 4, the idle time in between the sublots and lots on machine 4 has increased from 10 in the original schedule to 20 in the alternative schedule (2 time units of the difference in the completion time of the lots on machine 4 are due to a later start.) As a consequence, the completion time on machine 4 in the revised schedule has increased to 68, reducing the time buffer to 90-68=22 time units. Obviously, if non-bottleneck-dominant lots were to follow the current sequence, the first sequence would have been more desirable.

The example above demonstrates how the lexicographic rule works. The lots are sequenced in a manner that fully utilizes the machines preceding the bottleneck. For lots that are sufficiently large relative to the processing times of the sublots, this rule ensures the maximal utilization of the machines in a decreasing order of importance, from the bottleneck machine backwards.

Up to this point, we have derived special properties which enable us to sequence bottleneck-dominant lots in a manner that minimizes bottleneck idle time and maximizes bottleneck time buffer. This task has been accomplished by considering the effect of the bottleneck-dominant lots on the upstream machines. Two additional issues need to be addressed. First, how should the remaining bottleneck non-dominant lots be sequenced? and second, what would be the effect of the resultant sequence on the downstream machines and the makespan? To keep consistent with our initial strategy, the set of the bottleneck dominant lots would be sequenced first, followed by the set of the bottleneck non-dominant lots, in hope that the bottleneck dominant lots would build up a sufficient time buffer for the bottleneck non-dominant lots to not cause any bottleneck idleness. The sequence within the first set (i.e., the bottleneck dominant lots) should be determined via the lexicographic rule. But how should the sequence of the second set be determined? One way is to attempt to schedule lots (in their entirety) from the second set in between lots from the first set, if this does not create any idle time on the bottleneck machine and is in agreement with the lexicographic rule (i.e., the lots from the second set have a closer secondary bottleneck machine to the primary bottleneck machine than lots from the first set.) However, these two conditions, in particular the first one, may not, in many cases, hold. Even if they do hold for some lots of the second set, the remaining lots would still need to be scheduled using some other method.

Seemingly, it may appear as if the sequence of the second set should not have a major impact on the makespan since its effect on the bottleneck is almost unaffected by its sequence- it would either cause bottleneck idleness or it won’t - under both cases, the sequence itself may not have much of an impact on the result. Although this is a reasonable argument to make, the sequence of the second set can have a major impact on the makespan, as a consequence of a completely different cause. It is in fact a consequence of the impact of the sequence on the downstream machines, rather than on the bottleneck machine. For lots that are bottleneck non-dominant, there exists a different machine, other than the bottleneck machine, possibly a downstream machine, that may require a (substantially) larger processing time. Therefore, if the bottleneck machine does not have enough time buffer to ensure zero bottleneck idleness, this can result in a downstream machine becoming the bottleneck thereafter. In that event, it would have been best if the lots with the larger remaining processing time after the bottleneck machine would have been scheduled earlier in the sequence ahead of time. This would have enabled to absorb these large processing times, at least partially, while the bottleneck machine was still continuously working thus, better utilizing the future downstream bottleneck machine early in the process. In addition, this would have resulted in lesser workload for the downstream bottleneck machine once it has become the bottleneck. Another way to reach the above conclusion is by viewing the critical path (CP) of the resulting schedule. Knowing that the work on the bottleneck machine may not all belong to the CP, the above argument attempts to stretch the work on the bottleneck machine to the maximum extent. This is done by shifting lots that have large processing times on downstream machines forward in the set of non-dominant lots. There is more than one way to implement the "pushing-forward" of lots that have large processing times on downstream machines. We utilize the "tail" rule. This rule is applied as follows. For each lot, the largest unit processing time from among the machines immediately following the bottleneck machine to the last machine is identified. This is defined as the tail of lot and is identified for every lot. In the process of sequencing the lots, a lot with a larger tail would be sequenced before a lot with a smaller tail whenever possible.

One last issue has to be addressed before we present the BMI heuristic. This issue pertains to the dominance of the bottleneck machine. In the case of strong dominance (i.e., when the total processing time on the bottleneck machine is by far larger than the total processing time on any other machine), it may suffice to consider solely the bottleneck machine in the implementation of the heuristic. However, if the machines are relatively balanced (i.e., no clear dominance exists), it is obviously necessary to consider several machines, instead of a single machine, as potential primary bottlenecks. To that end, a statistical measure is utilized to determine the number of machines to be considered. First, the average total processing time and the standard deviation of the average total processing time are computed as follows:

$$\overline{T} = \frac{\sum_{j=1}^{m} T_j}{m} \quad ; \quad s_{\overline{T}} = \frac{S}{\sqrt{m}} \quad \text{where: } S = \sqrt{\frac{\sum_{j=1}^{m} \left(T_j - \overline{T}\right)^2}{m-1}}$$

where:

T_j - Total processing time on machine j.

Then, the set of machines to be considered as primary bottleneck machines is defined as follows. If, for any j, T_j satisfies: $T_j \geq \overline{T} + k \cdot s_{\overline{T}}$ (where k is some pre-determined value), then machine j should be included in the set of potential bottleneck machines. Next, the BMI heuristic is summarized.

In Step 1, the set (D) of potential primary bottleneck machines is identified. In Step 2, the first machine in the set (D) is considered to be the primary bottleneck machine and the heuristic starts. In Step 3, the tail of each lot is identified for later use. The sets of bottleneck-dominant lots and bottleneck non-dominant lots are defined in Step 4. In Step 5, the recursive chain of secondary bottleneck machines for each lot is identified while in Step 6 it is used to arrange the lots via the lexicographic rule. Then, in Step 7, an array is constructed which contains the sequence of the lots according to the lexicographic rule and the lot-type (“1” for bottleneck-dominant lots and “2” for bottleneck non-dominant lots). Step 8 contains the heart of the heuristic as it is the schedule-construction step. It constructs the schedule following the lexicographic sequence with the exception of pushing-back “2”-type lots after the next “1”-type lot in case they create bottleneck idleness when taken in their current location. This essentially achieves the same objective of pushing “2”-type lots forward when they are considered as a separate set which follows the set of “1”-type lots. Then, in Step 9, after all the “1”-type lots have been scheduled, the remaining “2”-type lots are added to the schedule in a non-increasing order of their *tail(i)*. The entire process is then repeated for other machines in the set D.

4.6.4 The BMI heuristic

The BMI heuristic for the LSSP without intermingling

Step 1. Initialize k ; Let $S \equiv \{1,2,...,N\}$ be the set of lots ; Compute:

$$T_j = \sum_{i=1}^{N} Q_i \cdot p_{ij} \ \forall \mathrm{j} \ ; \ \overline{T} = \frac{\sum_{j=1}^{m} T_j}{m} \ ; \ s_{\overline{T}} = \frac{s}{\sqrt{m}} \ \text{where}: \ s = \sqrt{\frac{\sum_{j=1}^{m} \left(T_j - \overline{T}\right)^2}{m-1}}$$

Determine the ordered set of candidate machines: $D = \left\{u : T_u \geq \overline{T} + k \cdot s_{\overline{T}}\right\}$

Step 2. Let j be the index of the first machine in set D. Set $D \leftarrow D - \{j\}$

Step 3. Let $BN = j$. If $j \neq m$, compute:

$$tail(i) = \max_{BN+1 \leq r \leq m} \{p_{ir}\} \quad \forall i$$

Step 4. Let S_1 be the set of bottleneck-dominant lots and S_2 be the complementary set (i.e., the set of bottleneck dominated lots.)

$$S_1 = \left\{ i : p_{i,BN} - \max_{1 \leq r < BN} \{p_{i,r}\} \geq 0 \right\} \ ; \ S_2 = \left\{ i : p_{i,BN} - \max_{1 \leq r < BN} \{p_{i,r}\} < 0 \right\}$$

Step 5. For each lot in the two sets, determine its secondary bottleneck machines starting with machine *BN*-1, throughout the upstream machines, until machine 1.

Step 6. Determine an ordered set S by arranging the lots in the decreasing order of the closeness of the secondary bottleneck machine to *BN*. Break ties in order of:

- The closeness of the subsequent secondary bottleneck machines to *BN* ;
- The value of *tail(i)* (larger value first) ;
- Arbitrarily (smaller sequential number)

Step 7. Construct a bi-dimensional array with the ordering as one of its dimensions and the set-type as its other dimension ("1" for S_1 and "2" for S_2)

Step 8. Construct the sequence as follows:

Sub-step 8.1 Schedule the first "1"-type lot in the array first. Push back any "2"-type lots that precede it in the array to follow the "1"-type lot in the array

```
Sub-step 8.2    If the next lot in the array is a “1”-type, schedule it.
                Else, if the next lot is a “2”-type, and, if scheduling this lot (in its entirety)
                does not result in bottleneck idleness, schedule the “2”-type lot next
                Else, if the “2”-type lot does result in bottleneck idleness, push it back in the
                array to follow the next “1”-type lot in the array.  If there are no more “1”-type
                lots to be scheduled, go to 9.
Sub-step 8.3    Repeat Step 8.2 until all the “1”-type lots are scheduled.
Step 9.  If  all the “2”-type lots have been scheduled, then go to Step 10 ;
         Otherwise, add the remaining “2”-type lots in non-increasing order of their tail(i)
Step 10. Save the best solution so far:  $j, S,$ and $M(S)$
Step 11. If  $D = \varnothing$  then STOP & PRINT SOLUTION ; Otherwise go to Step 2.
```

Modification of the BMI heuristic for the LSSP with intermingling

The BMI heuristic can also be adjusted to accommodate intermingling of sublots. Note that the original heuristic divides the lots into two sets, of bottleneck-dominant lots and bottleneck non-dominant lots, that, for the most part, follow one another sequentially. ‘Pushing-forward’ of lots from the second set in between lots from the first set occurs only when an <u>entire</u> lot from the second set can be inserted without causing idleness in between lots from the first set. However, if intermingling of sublots is allowed, then it is possible to forward sublots of lots that belong to the second set in between sublots of the first set, if this is desired according to the lexicographic rule, i.e., if sublots from the second set have a closer secondary bottleneck machine to the primary bottleneck machine than sublots from the first set. In this case, the amount of sublots from the second set that would be pushed-forward in between sublots of the first set is determined by the time buffer created on the primary bottleneck machine by the partial schedule. When the scheduling of the next sublot from the second set results in bottleneck idle time (i.e., no time buffer left), then we immediately switch back to scheduling the next sublot from the first set. Following the scheduling of every sublot from the first set, it can easily be checked whether it is possible to schedule a sublot from the second set next. Clearly, this modification enables more flexibility in scheduling the bottleneck machine, as it is no longer necessary to restrict the schedule to unbroken strings of sublots.

4.6.5 Numerical examples of the BMI heuristic

In this Section, several examples are provided to demonstrate how the BMI heuristic works under different operating conditions (i.e., with and without intermingling.) Consider the following numerical example. Four lots are to be processed on a five-machine flow-shop. The unit processing times as well as the lot-sizes are provided in **Table 4.12**. The question is how should the lots be sequenced to minimize makespan, given that intermingling is not permitted.

Table 4.12. Data for numerical example 4.5

Lot	**M_1**	**M_2**	**M_3 (BN)**	**M_4**	**M_5**	Q_i
1	2	3	7	10	10	8
2	5	3	5	1	1	11
3	6	4	10	6	4	12
4	10	1	7	1	5	14

The optimal sequence for this problem is {1-3-4-2} with a makespan of 336, and it is unique. The FIHLS heuristic, which was introduced in the previous Section, produces the sequence {3-4-1-2} with a makespan of 341 which is the second best. The worst makespan for this instance is 422 (26% above optimum) and is attained by three different sequences.

The BMI heuristic, applied to this example, works as follows. First, the total processing time on each machine is computed and machine 3 is identified as the primary bottleneck machine (Steps 1 and 2). Next, the sequence {1-3-4-2} is generated based on the lexicographic rule and the "tail" rule for breaking ties. Lot 1 is sequenced first because it has machine 2 as its secondary bottleneck machine while the other lots all have machine 1 as their secondary bottleneck machine. Lot 3 precedes lots 4 and 2 because it has a larger tail (6 compared to 5 and 1). Lots 1, 2, and 3 are bottleneck-dominant and are therefore "1"-type lots while lot 4 is a "2"-type lot. This completes Steps 3-7. The schedule is constructed via Step 8 as follows. Lot 1 is scheduled first because it is a "1"-type and next in the sequence. Lot 3 is scheduled next for the same reason. Lot 4, which is next in the sequence, is a “2”-type and must therefore be tried before being scheduled to ensure that it does not cause any bottleneck idleness. It is easy to verify that this is indeed the case. Thus, lot 4 is scheduled next. Lot 2 is sequenced last to complete the schedule. Step 9 is skipped in this case and the algorithm either terminates or repeats with a different machine considered as the primary bottleneck machine. The BMI heuristic has resulted in the sequence {1-3-4-2} which is the optimal solution in this case (and it was unique.)

Next, we take another example to show that, sometimes, the algorithm may not perform this well. This, however, happens due to special circumstances which, under intermingling and many common operating conditions (namely larger lot sizes, strong bottleneck, and/or regular processing time distributions with no number fixing) fade away and do not affect the efficiency of the BMI heuristic. In this example we again consider a four-lot five-machine flow-shop. The unit processing times as well as the lot-sizes are provided in **Table 4.13**.

Table 4.13. Processing time data for numerical example 4.6

Lot	**M_1**	**M_2**	**M_3 (BN)**	**M_4**	**M_5**	Q_i
1	4	2	6	4	5	7
2	1	6	1	6	8	15
3	5	8	9	8	2	11
4	10	5	10	4	2	11

The optimal sequence in this case is {3-1-2-4}, with a makespan of 318, and it is unique. The worst makespan is 400 (26% above optimum). The FIHLS heuristic produces the sequence {1-3-4-2} which has a makespan of 387 and is, therefore, one of the worst sequences. The BMI heuristic works as follows in this case. The primary bottleneck machine is identified as machine 3. Using the lexicographic rule and the tail-based rule, the sequence {3-2-1-4} is generated. Lot 3 is first because it is a “1”-type and has the closest secondary bottleneck machine. Lot 2 is next, although it is a “2”-type lot, because it also has machine 2 as its’ secondary bottleneck while lots 1 and 4 have machine 1 as their secondary bottleneck. Lot 1 precedes lot 4 because it has a larger tail. When constructing the schedule, it is apparent that lot 2, which is a "2"-type, would cause bottleneck idleness if it is to immediately follow lot 3 and is therefore pushed-back after the next "1"-type lot which is lot 1. It can easily be verified that, in that location as well, lot 2 would create some bottleneck idleness. It is therefore pushed further back after lot 4. The resulting schedule is {3-1-4-2} with a makespan of 394 which is one of the worst. Better results are obtained when the BMI procedure is repeated with the next two bottleneck candidates ({2-3-1-4} with a makespan of 356 and {3-2-4-1} with a makespan of 349). Nevertheless, the point to emphasize in this example is that, occasionally, it may not be possible to insert a "2"-type lot in between "1"-type lots early in the sequence and it would have to be delayed a few times. This delay may result in worsening the quality of the BMI solution because it is clear that according to the lexicographic rule (and the tail-based rule in this case), it is advantageous to schedule it early but, on the other hand, it must be delayed to avoid any idleness on the bottleneck. This problem would have easily been resolved if intermingling was allowed. Under intermingling, the "2"-type lot would have been partially scheduled, up to the exact portion for which an additional sublot creates bottleneck idleness. Then, only the (smaller) remainder of the lot would have been delayed. This is illustrated in **Fig. 4.5**. Eight items of lot 2 are scheduled in between items of lot 1. Thus, only seven of the fifteen items in lot 2 are delayed. The seven delayed items of lot 2 are then scheduled in between the first eight items of lot 4. The remaining three items of lot 4 are scheduled last. The resulting makespan for the intermingled schedule is reduced substantially to 285 (i.e., much better than any non-intermingled schedule). Note that in the schedule depicted in Fig. 4.5, there exists no idle time on the bottleneck machine. Using exp. (4.6-2), the lower bound on the optimal (intermingled) schedule in this case is:

$$LB^{LS} = \sum_{i=1}^{N}\left(Q_i \cdot p_{i,BN}\right) + \min_{S}\{IT\} + \min_{i}\left\{\sum_{j=1}^{BN-1}\left(p_{ij}\right)\right\} + \min_{i}\left\{\sum_{j=BN+1}^{m} p_{ij}\right\} = 266 + 0 + 6 + 6 = 278$$

Thus, the BMI heuristic has definitely produced a near-optimal schedule for the case of intermingling. Of course, an even better schedule would have had a single item of lot 1 as its first sublot (it is the fastest sublot to reach the bottleneck machine), followed by the BMI schedule. However, as mentioned earlier, the differences in the makespan due to the first and last sublot are practically negligible and are of theoretical importance only.

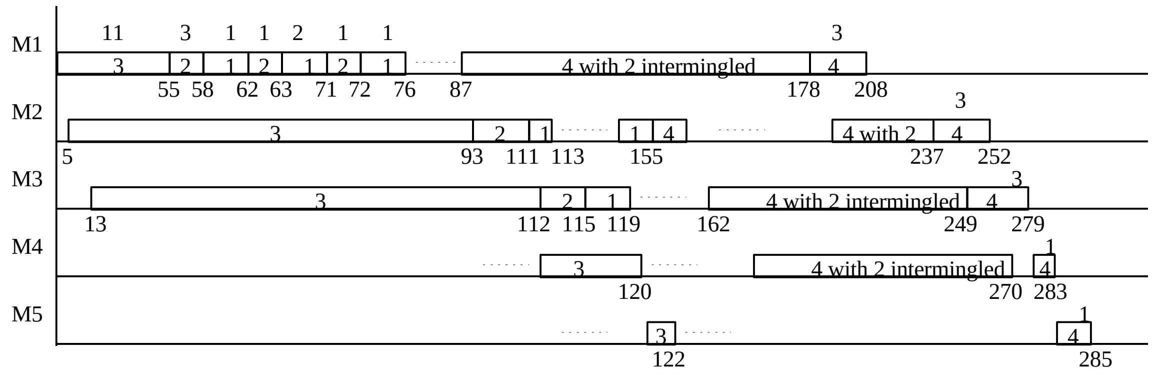


Fig. 4.5. The intermingled BMI schedule for example 4.6

4.6.6 Optimality of the BMI heuristic

It has already been established in the previous Sections that the BMI heuristic produces near-optimal schedules when a strong bottleneck machine exists. In such cases, the BMI heuristic finds a near-optimal schedule by constructing a schedule which minimizes the idle time on the bottleneck machine and, consequently, minimizes the makespan. Clearly, the task of finding a schedule which results in zero bottleneck idleness becomes easier when a strong bottleneck machine exists in the system.

In this Section, it is further shown, via experimentation, that, under other operating conditions, namely weak bottleneck dominance and even no bottleneck dominance (i.e., balance between the machines) the BMI heuristic still performs very well. Since the level of bottleneck dominance is a clear factor in determining the optimality of the BMI heuristic, the various problems that are tested are classified into three groups according to bottleneck dominance. The first group consists of problems with a strong bottleneck dominance. The second group includes problems with a weak bottleneck dominance and the third group includes problems with no bottleneck dominance.

To classify a certain problem into a group, two measures are used, that are computed based on the input data of the given (generated) problem. The first measure, R_1 , indicates the level of bottleneck dominance. It is defined as the ratio of the largest total processing time on the machines to the second largest total processing time on the machines. The second measure, R_2, indicates the level of overall dominance within the entire set of machines. It is defined as the ratio of the largest total processing time on the machines to the smallest total processing time on the machines. Mathematically, we have:

$$R_1 = \frac{\left\{\sum_{i=1}^{N} Q_i \cdot p_{i,BN}\right\}}{\max_{j \neq BN}\left\{\sum_{i=1}^{N} Q_i \cdot p_{i,j}\right\}} \quad ; \quad R_2 = \frac{\left\{\sum_{i=1}^{N} Q_i \cdot p_{i,BN}\right\}}{\min_{j}\left\{\sum_{i=1}^{N} Q_i \cdot p_{i,j}\right\}} \qquad (4.6.6\text{-}1)$$

The classification of each problem is made with respect to its values of the above two measures, and using **Table 4.14** as a guideline.

Table 4.14. Values for classification of problems into groups according to level of dominance

Group	*R_1 value*	*R_2 value*
Strong Dominance	Greater than 1.30	Greater than 2.00
Weak Dominance	Between [1.10,1.30]	Between [1.50,2.00]
No Dominance	Between [1.00,1.10]	Between [1.00,1.50]

A second factor that may have an effect on the solution quality of the BMI heuristic is the location of the bottleneck machine along the line. Three levels are considered for this factor as well namely, beginning of line (B), center (C) and end of line (E). For each of the seven combinations of level of dominance and bottleneck location (only one combination for the case of no dominance), both the non-intermingled FIHLS heuristic and the non-intermingled BMI heuristic have been tested and compared with the optimal solution. A four-lot five-machine flow-shop has been utilized. The processing times and lot-sizes have been randomly generated from a uniform distribution $U[5,15]$. The problems have been generated until eight results have been collected for each combination. Eight results are sufficient for the averages to be accurate to 0.5%. The ratios of the BMI and the FIHLS resultant makespans to the optimal makespan, determined via complete enumeration, have been recorded. The BMI heuristic has been utilized with the cardinality of the set D being set to 1 (i.e., a single machine considered as the primary bottleneck machine at Step 2 of the algorithm.) The entire set of results is given in **Appendix F**. These results are summarized in **Table 4.15**.

Recall that in Section 4.5 the optimality of the intermingled version of the FIHLS heuristic has been examined. Here, conversely, the non-intermingled version of the FIHLS heuristic is tested and compared to the BMI heuristic. The results shown in Table 4.15 indicate that the BMI heuristic is near-optimal not only in cases for which there exists a clear bottleneck in the system but in the other cases as well. In fact, it appears that the BMI heuristic is insensitive to either the level of dominance of the bottleneck machine or its location, and produces near-optimal solutions regardless of the level of dominance of the bottleneck machine or its location. On the average, the BMI heuristic produces sequences with makespan values which are 1.1% above the optimum. The average is slightly larger for weak bottleneck dominance (1.3%) and no bottleneck dominance (1.3%) than for strong dominance (0.8%) but these differences appear to be within the random range of 0.5%. It is also apparent from the results that the BMI heuristic outperforms the FIHLS heuristic in this case of no intermingling. On the average, the FIHLS heuristic produces sequences which are 3.5% above the optimum. Furthermore, unlike the BMI heuristic, the FIHLS heuristic appears to be sensitive to both the level of dominance and the location of the bottleneck. The FIHLS heuristic performs better when there is more dominance in the system (3.1% above the optimum for strong dominance, 3.4% for weak dominance, and 5.1% for no dominance). The FIHLS heuristic also performs better when the location of the bottleneck machine is at the beginning of the line (2.4% above the optimum).

Table 4.15. Optimality ratios for the BMI and FIHLS heuristics

Level of Dominance	Bottleneck Location		
	B	C	E
Strong Dominance	1.006 1.022	1.013 1.047	1.007 1.025
Weak Dominance	1.016 1.026	1.013 1.037	1.011 1.038
No Dominance	1.013 1.051		

4.6.7 The impact of setup on the performance of the BMI heuristic

In the discussion thus far, setup has been ignored in evaluating the BMI heuristic. As has been shown earlier, with respect to the FIHLS heuristic, the impact of even a small level of setup, in the magnitude of 5% of the lots' processing times, on the sequence, is significant. It essentially makes intermingled sequences become undesirable since there is more setup associated with such sequences. Hence, non-intermingled sequences result when applying the FIHLS heuristic on problems where setup times are 5% (or more) of the lots' processing times. It is expected that a similar behavior would be encountered using the BMI heuristic.

The BMI heuristic attempts to construct a schedule which minimizes bottleneck idleness. This task is clearly easier to accomplish when setup is not considered. However, when setup is considered, there exists a trade-off between the saved time from avoiding bottleneck idleness and the saved time from avoiding intermingling of sublots which implies larger setup times (on the bottleneck machine as well as on the other machines.) Obviously, as setup time in the system increases, less intermingling is anticipated.

In this Section, we first examine the theoretical impact of setup on the bottleneck and the critical path. In the analysis, a lower bound on the makespan is derived which provides means of determining the number of intermingled sublots of each lot-type that would still need to be considered by an enumerative search technique. Then, in the second part of this Section, the practical numerical example 4.5.5(2) is utilized to demonstrate that, as expected, even a small percentage of setup is sufficient to make intermingling undesirable and, consequently, for a non-intermingled sequence to result from the BMI heuristic.

4.6.7.1 Theoretical impact of intermingled sequences on optimality

Next, a lower bound on the makespan is proposed for intermingled sequences. This lower bound can be utilized to reduce the search space in any explicit or implicit enumeration scheme of the possible sequences. Let *LB* denote a Lower Bound on the optimal makespan. The following lower bound is obtained by taking the minimal time it takes for the first sublot to reach the last machine, plus the minimal total time for all the sublots to be processed on the last machine:

$$LB = \min_{i} \left\{ \sum_{j=1}^{m-1} \left(su_{ij} + p_{ij} \right) \right\} + \sum_{i=1}^{n} \left(su_{im} + Q_i \cdot p_{im} \right) \tag{4.6.7.1-1}$$

The above bound is prescribed based on the critical path which is comprised of a single sublot over the first *m-1* machines and all the sublots on the last machine.

Corollary 4.1. Every sublot (or a group of similar sublots) and its associated setup is critical at least on one machine.
Proof.
(By contradiction)
Suppose there exists a sublot which does not belong to the critical path on any machine. Let the vector $\left(t_1, t_2, .., t_m\right)$ denote the times of the critical path on machines 1,2,..m respectively (with $t_1 = 0$). Obviously, $t_1 \le t_2 \le .. \le t_m$. Since the sublot is not on the critical path of machine 1, it must have been processed after time t_2. Thus, it could not have been processed before time t_2 on machine 2. In addition, it could not have been processed before time t_3 as well because that would have made it part of the critical path. Hence, it must have been processed after time t_3 on machine 2. Repeating the argument for machines 3, 4,.., *m-1* leads us to the conclusion that the sublot must have been processed after time t_m on machine *m-1*. Thus, it must have also been

processed after time t_m on machine *m*, but that would have made it part of the critical path, leading to contradiction under the assumption that its not critical. *QED*.

Note that the above holds with respect to the sublot and its associated setup, as the setup is attached to the sublot. The next theorem helps in identifying an optimal non-intermingled sequence, if it exists.

Theorem 4.6. There exists an optimal SN *sequence,* SN^**, having a makespan* M^**, if it satisfies the following:*

$$M^* \leq LB + \min_{i,j}\{su_{ij}\} \tag{4.6.7.1-2}$$

Proof

Consider the best *SI* sequence. This sequence can be obtained by a series of (one or more) shifts of sublots to SN^*. Based on Corollary 4.1, the least that these shifts add to the makespan is the minimal additional setup associated with the sublots that have been shifted, i.e., $\min_{i,j}\{su_{ij}\}$, in case of a single shift of sublot of type *i*. Therefore, the *SI* makespan would be at least: $LB + \min_{j}\{su_{ij}\}$. Hence, if the above holds, it guarantees that no *SI* sequence can do better than SN^*. QED.

Consider the following numerical example which demonstrates how the above theorem can be utilized to declare the optimality of a non-intermingled sequence. Two lots are to be processed by a two-machine flow-shop. The processing times and the setup times are provided in **Table 4.16**.

Table 4.16. Data for numerical example 4.7

	Proc. times		Setup times		Lot size	# Items
i	p_{i1}	p_{i2}	su_{i1}	su_{i2}	Q_i	n_i
1	1	3	2	1	3	3
2	2	1	2	3	2	2

Since *N=2*, there are only two (*2*!) sequences in the set *SN*: $SN = \{(1,1,1,2,2);(2,2,1,1,1)\}$. The set of *SI* sequences, on the other hand, consists of $\frac{5!}{3!\cdot 2!} - 2 = 8$ sequences. Constructing the corresponding schedules of *SN* sequences, we find that $SN^* = (1,1,1,2,2)$ with a makespan of 18. The lower bound on the makespan for this problem, obtained via (4.6.7.1-1), is:

$$LB = (2+1) + [(1+3) + (3\cdot 3 + 2\cdot 1)] = 18$$

Thus, we can immediately conclude that $SN^* = (1,1,1,2,2)$ is the optimal sequence (note that this is a non-intermingled sequence.) We can also conclude that this optimum is unique, since $M^* = 18$ is strictly less than $LB + \min_{i,j}\{su_{ij}\} = 19$. A generalization of Theorem 4.7, to exclude *SI* sequences with more than n_i sublots of type *i* intermingled, is also possible and is given next.

Theorem 4.7. There exists an SN sequence, SN^*, having a makespan M^*, which is better than an SI sequence having $\{n_i\}_{i=1}^{N}$ groups of sublots of type i intermingled, if the following holds:

$$M^* \leq LB + \sum_{i=1}^{N}(n_i - 1)\cdot \min_j\{su_{ij}\} \qquad (4.6.7.1\text{-}3)$$

Proof. Similar to Theorem 4.6.

Note that in the case of a single sublot of type *i*, shifted from its position in an *SN* sequence (i.e., $n_i = 2$), the right-hand-side of the above expression reduces to $\min_{i,j}\{su_{ij}\}$, in agreement with (4.6.7.1-2). Theorem 4.7 provides a boundary for the maximal number of sublots of each lot type that is needed to be considered for optimality and it can, therefore, be used in order to limit the search space in any enumeration scheme.

4.6.7.2 Numerical example

Next, it is shown, utilizing the practical numerical example 4.6, that, as expected, even a small percentage of setup is sufficient to make intermingling undesirable and, consequently, for a non-intermingled sequence to result from the BMI heuristic. Consider the setup time data provided in **Table 4.17**. The integer number given in Table 4.17 is the rounded-up/rounded-down number of the 5% setup time applied to the lot-processing time on the respective machine, as it is given in Table 4.11.

Table 4.17. Setup time data for numerical example 4.6

Lot	M_1	M_2	M_3 (BN)	M_4	M_5
1	1	1	2	1	2
2	1	4	1	4	6
3	3	4	5	4	1
4	5	3	5	2	1

Setup time must be added prior to a sublot or a series of sublots of the same type whenever there is a change in the schedule of the lot-type to be processed next (i.e., attached-setup.) Incorporating the setup times in the schedule obtained previously, which is illustrated in Fig. 4.5, results in a very large makespan (357 based on adding setup times to the sublots on the CP), as well as idle times on the bottleneck machine which further worsen the makespan. Hence, an introduction of 5% setup time causes an increase of more than 24% in the makespan. This result is due to the high level of intermingling in the original BMI schedule for the case of no setup. In the absence of setup, there is no real motivation to reduce the changeover, and a high level of intermingling results, as it significantly improves the makespan. However, even the introduction of a small setup immediately results in less intermingling, as it becomes necessary to reduce changeover time, in particular on the bottleneck machine.

An interesting question arises with regard to the setup percentage that would make an intermingled BMI sequence as desirable as a non-intermingled sequence. This breakeven point may be estimated as follows. Let:

$M(S)$	- Makespan of a given sequence *S*, where $S = \{SI \text{ or } SN\}$.
$f_{ij}(S)$	- Number of times that sublots of type i require setup on machine j in sequence S ; $f_{ij}(SN) = 1 \ \forall i, j$ (sublots not intermingled)
$su_{BN}(S)$	- Total setup time associated with sequence *S* on the bottleneck machine (*BN*):
K	- Breakeven setup percentage.

Assume that the entire increase in the makespan is due to the setup attached to the sublots on the bottleneck machine only. That is, setup times elsewhere do not create idleness on the bottleneck machine. The accuracy of this assumption relies on the level of dominance of the bottleneck. Obviously, the stronger the bottleneck, the more accurate the assumption is. However, note that in the case of a weak (or no) bottleneck dominance, the impact of setup in intermingled sequences would be even larger. Therefore, the following estimate can serve as an upper-bound estimate on the breakeven setup percentage. In order to estimate this breakeven point, *K*, the following can be prescribed:

$$M(SN) + su_{BN}(SN) = M(SI) + su_{BN}(SI) \qquad (4.6.7.2\text{-}1)$$

The Left-Hand-Side (LHS) of equation (4.6.7.2-1) represents the makespan of the *SN* sequence with the setup on the bottleneck machine only added to it. Similarly, the RHS of the equation represents the makespan of the *SI* sequence with the setup on the bottleneck machine only added to it. By definition, for the setup time to be K% of the lot-processing time, on the bottleneck machine, we have:

$$\begin{aligned} su_{BN}(SI) &= \sum_{i=1}^{N} f_{i,BN}(SI) \cdot \left(K \cdot Q_i \cdot p_{i,BN}\right) = \\ &= K \cdot \sum_{i=1}^{N} f_{i,BN}(SI) \cdot \left(Q_i \cdot p_{i,BN}\right) \end{aligned} \qquad (4.6.7.2\text{-}2)$$

For the *SN* sequence,

$$su_{BN}(SN) = K \cdot \sum_{i=1}^{N} \left(Q_i \cdot p_{i,BN}\right) \qquad (4.6.7.2\text{-}3)$$

Substituting (4.6.7.2-2) and (4.6.7.2-3) in (4.6.7.2-1), and re-organizing, we get:

$$K = \frac{M(SN) - M(SI)}{\sum_{i=1}^{N} \left(f_{i,BN}(SI) - 1\right) \cdot \left(Q_i \cdot p_{i,BN}\right)} \qquad (4.6.7.2\text{-}4)$$

In our numerical example, for the intermingled BMI sequence, we have:

$$f_{1,3} = 6;\ f_{2,3} = 13;\ f_{3,3} = 1;\ f_{4,3} = 8$$

Utilizing exp. (4.6.7.2-4), we get:

$$K = \frac{318 - 285}{1{,}160} = 0.028 = 2.8\%$$

That is, if a setup of more than 2.8% of the lot-processing time is introduced, it is most likely that the intermingled schedule would result in a higher makespan than the non-intermingled schedule. As has been demonstrated earlier, indeed, the BMI schedule, that was near-optimal without setup, had become totally undesirable with the introduction of 5% setup.

4.6.8 Summary of the BMI heuristic for the LSSP

In this Section, an efficient near-optimal heuristic for the Lot-Streaming Sequencing Problem (LSSP) has been developed and examined. The proposed heuristic is based on a thorough bottleneck analysis, and on the special properties derived from this analysis for the problem at hand. Algorithms have been proposed for the construction of both intermingled and non-intermingled schedules. Several numerical examples have been provided to demonstrate how the heuristic works. A comprehensive experimental study has been conducted to show that the non-intermingled BMI heuristic produces near-optimal solutions. The same study has also shown that the BMI heuristic outperforms the FIHLS heuristic, and that it is insensitive to either the level of dominance of the bottleneck machine or its location. Finally, the impact of setup times on the level of intermingling in the schedule has been exploited, from a theoretical as well as from a practical standpoint (through example). Lower bounds on the number of intermingled sublots in an intermingled schedule were derived. A numerical example was utilized to show that, practically, even relatively small setup times (with respect to lot-processing times) make it almost immediately undesirable to intermingle sublots, due to the significant negative impact on the makespan.

4.7 The Flow Shop Lot Streaming (FSLS) Problem

In Sections 4.4 to 4.6, the lot-streaming sequencing problem (LSSP) was considered. In the LSSP, the sublot size is pre-determined. In contrast, in the remainder of Chapter 4, the more general Flow-Shop Lot-Streaming (FSLS) problem, in which the sublot size and the sequence must both be determined, is considered.

The FSLS problem arises, for example, in semi-conductor processes. The production processes are all designed to support equal sublots, transferred between the machines using transfer devices known as cassettes (or boxes, or boats.) Prior to the start of processing a sublot, certain operations (e.g., re-calibration, material tracking, quality control, inspection) are required, which can simply be regarded as sublot-attached setups. Under these operating conditions, the problem can be stated as follows:

Given N lots, to be streamed through an m-machine flow shop, determine the optimal sublot size(s) and the optimal sequence of the N lots, so as to minimize the makespan objective.

The single-lot version of this problem has been dealt with, in length, in Chapter 3. An optimal solution algorithm has been developed for the determination of the optimal sublot size, which minimizes the makespan.

Unfortunately, the multiple-lot version of this problem is by far more complicated than the single-lot version since, in the multiple-lot version, the sequence must be determined in addition to the sublot size and, therefore, a sequencing problem is embedded in this case. The essence of the problem, however, remains the same as in the single-lot version of the problem. There exists a trade-off between the setup time and the idle time on the bottleneck machine (as well as on the other machines.) This trade-off can be stated as follows. The larger the number of sublots that the multiple lots are split into, the higher is the amount of overlapping on the machines (and the lower is the idle time on the bottleneck), leading to possible reduction in the makespan. On the other hand, the larger the number of sublots that the multiple lots are split into, the larger is the overall (unproductive) setup time associated with the sublots, leading to a possible increase in the makespan.

The resultant problem is difficult-to-solve simply by virtue of its nature as a sequencing problem. However, as we shall see in the sequel, optimal solutions can be obtained for a two-machine flow-shop. Extensions are proposed for the *m*-machine flow-shop. In these extensions, the BMI heuristic, which has been developed earlier in this Chapter for the LSSP, is utilized for the solution of the embedded sequencing problem.

4.7.1 Global sublot-size two-machine FSLS problem

We start this analysis of the FSLS problem with the case of an identical ("global") equal sublot-size and a two-machine flow-shop. Recall that the *N*-job two-machine flow-shop sequencing problem can be solved optimally via Johnson's rule (see Section 2.3.1 of the Literature.)

However, in this case, the jobs cannot be determined without determining the sublot-size first. It should therefore be apparent that an iterative procedure can be utilized to solve the global sublot-size two-machine FSLS problem optimally. In this iterative procedure, iterations on the sublot-size would be carried-out, starting at some minimal value and ending at some maximal value for the sublot-size. At each iteration, given the sublot-size, the processing times of the sublots would then be determined. Johnson's rule would then be applied on these processing times to determine the optimal sequence of the sublots. The resultant makespan would be computed and recorded for the identification of the optimal sublot size and sequence.

Clearly, the sublot-size and sequence, derived from the application of the procedure described above, form the optimal solution to the global sublot-size two-machine FSLS problem. This is simply due to the fact that the procedure searches over the entire set of feasible solutions. However, a significant amount of computation is saved since the optimal sequence is identified for each sublot-size via Johnson's rule (as opposed to complete enumeration.) The optimal solution algorithm is presented next.

4.7.1.1 Optimal solution algorithm for the global sublot-size two-machine FSLS problem

<u>Optimal solution algorithm for the global sublot-size two-machine FSLS problem</u>

Step 1. Input: $su_{i,1}, p_{i,1}, su_{i,2}, p_{i,2}, Q_i \quad \forall i$, and *UB* (Use $UB = \max_i \{Q_i\}$ in the absence of a tighter value)

Initialize sublot-size, *L*=*LB* (Use *LB* =1 in the absence of a tighter value)

Step 2. For each lot i, compute the number of "jobs" to be considered as follows: $n_i = \left\lfloor \frac{Q_i}{L} \right\rfloor$

Consider the last job to be of size $R_i = Q_i - n_i \cdot L$ (the "remainder" sublot).

With each of these jobs, associate the following "job" processing times on the two machines:

$$A_i = su_{i,1} + L \cdot p_{i,1} \quad ; \quad B_i = su_{i,2} + L \cdot p_{i,2}$$

Step 3. Apply Johnson's rule on the entire set of jobs formed in Step 2 to obtain a sequence, *S*(L)*. Compute the resultant makespan of the this sequence, *M(S*(L))*.

Save the best sequence so far, *S*(L)*.

Step 4. If *L*=*UB* then go to 5 ; Otherwise, set *L*=*L*+1, and go to Step 2.

Step 5. Print the optimal solution: *L*, S*(L*), M*(S*(L*))* ; STOP

4.7.1.2 Complexity of the algorithm

The complexity of the algorithm is as follows. In Step 3, Johnson's algorithm is applied on a list of length $(2 \cdot N)$ instead of *N* due to the application of the rule to sublots instead of lots. The length is doubled because there are two sublot-types associated with each lot (the full sublot and the remainder sublot). Note that the full sublots of each lot type would all be ordered continuously in an unbroken string since they are identical. It is therefore not required to consider each full sublot separately. Thus, the order of complexity of the application of Johnson's rule remains the same, i.e. as if it was applied on a list of *N* jobs. The complexity of Johnson's rule in that case is known to be $N \cdot \log(N)$. Then, in Step 4, the application of Johnson's rule is repeated for a maximum of $Q_{\max}$ times. Hence, the overall complexity of the algorithm is of order $O(N \cdot \log(N) \cdot Q_{\max})$. The algorithm, therefore, qualifies only as a pseudo-polynomial algorithm. Nevertheless, it is expected to be very fast for reasonable lot-sizes, as a reasonable *Q* is in the order of 100's to 1,000's of items while a reasonable number of computations per second is in the order of 1,000,000's operations.

4.7.1.3 Numerical example

To demonstrate how the above algorithm works, consider the following numerical example. Three lots are to be processed on a two-machine flow-shop. The unit processing times, the setup times, and the lot-sizes are given in **Table 4.18**.

Initially, *L*, the sublot size, is set to 1. For this sublot size, the number of full sublots of each lot-type, n_i, is computed and the size of the remainder sublot, R_i, is determined. Next, the job processing times, A_i and B_i, are computed for these sublots, and the optimal sequence, *S*(L)*, is established using Johnson's rule. Lastly, the corresponding makespan is realized. These computations, for every value of *L*, are summarized in **Table 4.19**. Note that in specifying a sequence, an underscore has been used to differentiate between remainder sublots and full (regular) sublots. Remainder sublots have been underscored. It is clear from Table 4.16 that the optimal solution for the above problem is to have a sublot-size of four (*L**=4). Note that no lot is split in that solution. Lot type 1 is considered a full sublot while lots 2 and 3 are considered remainder sublots in their entirety since they contain a quantity of items which is less than what is needed for a full sublot. The optimal makespan is 37.

To show how the above solution changes into a solution in which lots are split into sublots, consider the following change in the input data. The setup times for the lots decrease by three time units all across while the unit processing times increase by two time units all across. The revised input data is given in **Table 4.20**, and a summary of the new results is presented in **Table 4.21**. It can be seen that, after the change, a sublot-size of two becomes optimal. Due to the increased processing time and the decreased setup time, it has become desirable to split the lots into smaller sublots. Note that, in the optimal solution, lot type 1 is split into exactly two full sublots. Lot type 2, on the other hand, is split to a full sublot and a remainder of 1 item. Lot type 3 is taken, in its entirety, as a full sublot. The optimal makespan is 47, and either a decrease or an increase in the sublot-size would cause a strict increase in the makespan. Hence, $L=2$ with the corresponding makespan is the unique optimal solution for the revised problem.

The two examples point to an expected trend. As processing time in the system increases relative to setup time, it becomes more attractive to split lots in order for more overlapping to occur. Therefore, it can be concluded that the optimal sublot-size would tend to decrease (i.e., approach 1) with increments in the amount of processing in the system.

Table 4.18. Data for numerical example 4.8

Lot	Proc. times		Setup times		Lot size
i	p_{i1}	p_{i2}	SU_{i1}	SU_{i2}	Q_i
1	1	2	5	4	4
2	2	1	3	6	3
3	2	2	5	3	2

Table 4.19. Solution summary for numerical example 4.8

Sublot-Size, *L*	**Lot #**	n_i	A_i	B_i	R_i	A_i	B_i	**Sequence $S^*(L)$**	**Makespan $M(S^*(L))$**
1	1	4	6	6	0				
	2	3	5	7	0			{2,2,2,1,1, 1,1,3,3}	60
	3	2	7	5	0				
2	1	2	7	8	0				
	2	1	7	8	1	5	7	{$\underline{2}$,2,1,1,3}	43
	3	1	9	7	0				
3	1	1	8	10	1	6	6		
	2	1	9	9	0			{$\underline{1}$,1,2,$\underline{3}$}	40
	3	0			1	9	7		
4	1	1	9	12	0				
	2	0			1	9	9	{1,$\underline{2}$,$\underline{3}$}	37
	3	0			1	9	7		

Table 4.20. Revised data for numerical example 4.8

Lot	Proc. times		Setup times		Lot size
i	p_{i1}	p_{i2}	SU_{i1}	SU_{i2}	Q_i
1	3	4	2	1	4
2	4	3	0	3	3
3	4	4	2	0	2

Table 4.21. Solution summary for the <u>revised</u> numerical example 4.8

Sublot-Size, *L*	**Lot #**	n_i	A_i	B_i	R_i	A_i	B_i	**Sequence $S^*(L)$**	**Makespan $M(S^*(L))$**
1	1	4	5	5	0				
	2	3	4	6	0			{2,2,2,1,1, 1,1,3,3}	50
	3	2	6	4	0				
2	1	2	8	9	0				
	2	1	8	9	1	4	6	{<u>2</u>,2,1,1,3}	47
	3	1	10	8	0				
3	1	1	11	13	1	5	5		
	2	1	12	12	0			{<u>1</u>,1,2,<u>3</u>}	49
	3	0			1	10	8		
4	1	1	14	17	0				
	2	0			1	12	12	{<u>1</u>,2,3}	51
	3	0			1	10	8		

4.7.1.4 The makespan function – useful insights

One question that is of practical importance with regards to the shape of the makespan function is the following: Is the makespan function convex in (the continuous) *L*? Because if it is, the linear search for the optimal solution, over all the possible (discrete) *L*-values, can be replaced by an exponential (or binary) search over the (discrete) *L*-values. In terms of complexity, this would result in $\log_2(Q_{\max})$ replacing $Q_{\max}$ in the expression for the order of complexity of the optimal solution algorithm presented in Section 4.7.1.1.

The numerical example in the previous Section can be used to gain useful insights regarding the shape of the makespan function. Note that, for different sublot-sizes, different sequences correspond to the optimal makespan. From **Table 4.21**, it can be seen that for sublot-sizes of *L=1,2,* the optimal (non-intermingled/unbroken) sequence is {2-1-3} while for sublot-sizes of *L=3,4*, the optimal (non-intermingled/unbroken) sequence is {1-2-3}. Let $S_1 \equiv$ {2-1-3} and $S_2 \equiv$ {1-2-3}. The makespan value, for *L*=1,..,4, for each of the two sequences is plotted and presented in **Fig. 4.6**.

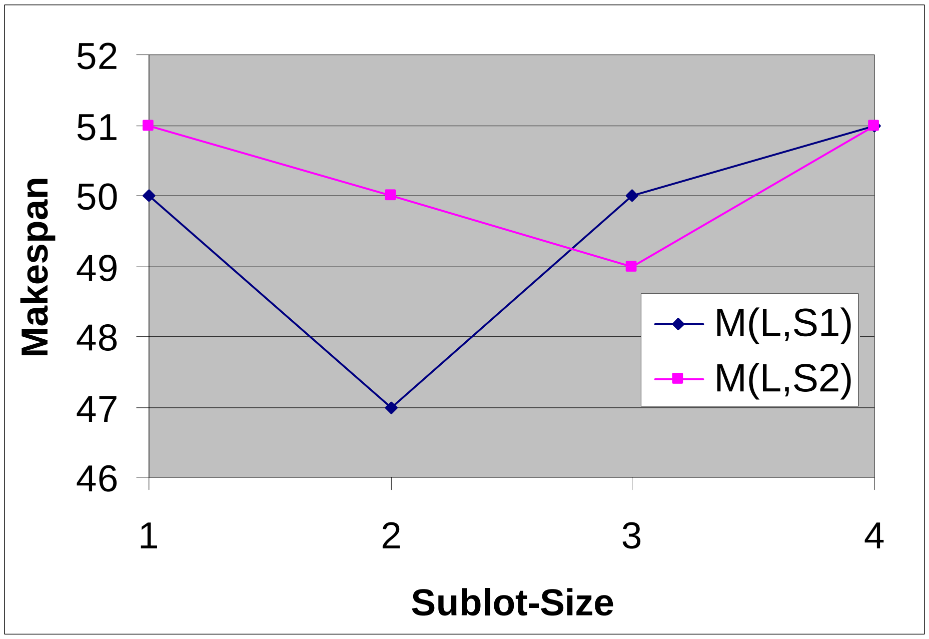


Fig. 4.6. Makespan as a function of the sublot-size

It can be observed that, the makespan function for a given sequence need not be convex, as in the case of sequence S_1. Consequently, the makespan function, that is comprised of the (lower) envelope formed by the two sequences, need not be convex (or even quasi-convex). Nevertheless, experiments with other problem scenarios indicate that the makespan function for a given sequence, *M*(*L/ S* given), is almost always quasi-convex. In those cases, the makespan function, *M*(*L*), is quasi-convex as well. Therefore, substitution of the linear search for the optimal solution, with an exponential (or binary) search over the (discrete) *L*-values appears like a promising practical approach.

4.7.2 Lot-based sublot size two-machine FSLS problem

The case of lot-based sublot sizes for a two-machine flow-shop problem is clearly more complicated than the case discussed above, of a global sublot-size. However, an optimal solution, which is pseudo-polynomial, can easily be derived for this case, in a manner similar to that for the case of a global sublot-size. The only modification required in the algorithm presented in Section 4.7.1.1, to adjust it for the lot-based sublot size two-machine FSLS problem, is that, the iterations over the sublot sizes would be carried out with respect to the individual lot-sizes. Thus, the optimal solution would be identified from among all possible combinations of the various sublot sizes of the lots. The revised algorithm is as follows.

4.7.2.1 Optimal solution algorithm for the lot-based sublot-size two-machine FSLS problem

Optimal solution algorithm for the lot-based sublot-size two-machine FSLS problem

Step 1. Input: *N, m,* $su_{i,1}, p_{i,1}, su_{i,2}, p_{i,2}, Q_i \quad \forall i$

Step 2. Initialize sublot-sizes:

$L_i = LB_i$ ($LB_i = 1; UB_i = Q_i \quad \forall$i, in the absence of tighter values)

Step 3. FOR $\underline{L}=\underline{1}$ to $\underline{UB}$ (vector notation)

Step 4. For each lot, $\forall i$, compute the number of "jobs" to be considered.

Consider the last job to be of size $R_i = Q_i - n_i \cdot L_i$ (the "remainder" sublot).

With each of these jobs, associate the following "job" processing times on the two machines:

$$A_i = su_{i,1} + L_i \cdot p_{i,1} \quad ; \quad B_i = su_{i,2} + L_i \cdot p_{i,2}$$

Step 5. Apply Johnson's rule on the entire set of jobs formed in Step 2 to obtain a sequence, *S*(L)*. Compute the resultant makespan of the this sequence, *M(S*(L))*.

Save the best sequence so far, *S*(L)*.

Step 6. If *L<UB* then NEXT *L* (Back to Step 3)

Otherwise, Go to Step 7.

Step 7. Print the optimal solution: *L*, S*(L*), M*(S*(L*))* ; STOP

The above algorithm extends the search, compared to the algorithm developed for the global sublot-size case, from a search over a single-dimensional variable (namely, the global sublot-size) to a search over a multi-dimensional variable (namely, the lot-based sublot-sizes.) The iterative process is repeated for every combination of values of the sublot sizes, L_i.

4.7.2.2 Complexity of the algorithm

Although optimal, the above algorithm may become cumbersome and practically inefficient when the number of lots, and the lot-sizes, are both relatively large. It can easily be observed that the algorithm iterates $\prod_{i=1}^{N} Q_i$ times, using Johnson's rule each time, to identify the optimal combination of sublot sizes for the various corresponding lots. Therefore, its complexity is of order $O\left(N \cdot \log(N) \cdot \prod_{i=1}^{N} Q_i\right)$. The expression $\prod_{i=1}^{N} Q_i$ is problematic, complexity-wise, since for lot-sizes of the same order of magnitude, it is of order $O\left(Q^N\right)$, which is exponential in the number of lots.

4.7.2.3 Fast heuristic for the lot-based sublot-size two-machine FSLS problem

The order of complexity associated with the optimal solution algorithm motivates the development of a quick and efficient heuristic. A two-phase heuristic is proposed. This heuristic first attempts to find a near-optimal sequence for the lots. Then, at the second phase, given the sequence, the lot-based sublot-sizes of each of the lots are optimized. Hence, the proposed heuristic includes a construction phase (phase 1) and an improvement phase (phase 2).

The sequence of the lots in phase 1 is obtained as follows. For each lot, the single-lot two-machine problem is solved optimally. The optimal solution, based on the analysis in Chapter 3, can be obtained from exp. (4.7.2.1). Johnson's rule is then applied on the sublot processing times of the lots on machines 1 and 2 in order to obtain the sequence. In all likelihood, the first lot (denoted as *p*) in this sequence is indeed the best lot to sequence first. This is due to two reasons. First, the processing time of the first sublot of this lot on machine 1 must be the smallest from among all the lots (recall that Johnson's rule is an SPT(1) rule), therefore minimizing the start time (or the lag time) on machine 2. Second, it is most likely that this lot (i.e., lot *p*) would not cause any idle time on machine 2. This is due to the fact that Johnson's rule schedules jobs that are $A_p \le B_p$ first, i.e. the sublot processing time on machine 1 is less than or equal to the sublot processing time on machine 2. That, in turn, implies no idle time on machine 2. The only case for which the first lot would, in fact, cause idle time on machine 2 is the case in which, for all the lots, $A_i > B_i$. In this special case, however, every lot would cause some unavoidable idle time on machine 2.

Once the sequence is established, with the first lot as the most likely best lot to be sequenced first, phase 2 of the heuristic is carried out. At this phase, re-optimization of the sublot sizes of subsequent lots is performed.

To determine the lot ordering for the re-optimization process, note that minimizing the makespan can be done by reducing the (unproductive) setup time (on both machines), without increasing the idle time on machine 2 by more than the time saved by the setup time reduction. In phase 2, the proposed heuristic does exactly that. It first prioritizes the lots from the lot with the most potential setup time savings to the lot with the least potential setup time savings. Then it iteratively increments the sublot size of the first lot on the prioritized list such that setup time is saved and the makespan is reduced to its minimum. This re-optimization of the sublot size is then repeated for all the lots according to the prioritized list.

While iterations are carried out to reduce the makespan with respect to a lot, the heuristic accepts a new (larger) sublot size even if that sublot size does not result in a strict reduction in the makespan, rather it leads to the same makespan. This is because a larger sublot size, although not necessarily strictly improving the makespan, always implies that the run-time on machine 1 is strictly smaller (since the setup time is smaller.) Thus, between the two choices that lead to the same makespan, the one with the larger sublot size is always preferable since it would always have a smaller run-time on machine 1.

In the process of ordering the lots by total setup time, the first lot in the sequence is not considered. This is due to the following reason. Since it is the first lot in the sequence, it is not constrained by any previous lots. Consequently, its sublot size (determined in phase 1 by solving the single-lot unconstrained problem) is already optimal and, thus, there is no need to re-optimize. However, after the re-optimization process of the other lots is completed, it may still be beneficial to re-optimize the sublot size of the first lot. Therefore, the first lot is added as the last lot in the prioritized list, implying that its sublot size would be re-optimized last.

It should be apparent that the re-optimized sublot size of the next lot on the prioritized list would not necessarily be equal to the sublot size which was computed in phase 1 for the (unconstrained) single-lot problem. The re-optimized sublot size would either be the same or larger (which means that some setup time is saved.) Therefore, iterations are made over the sublot size, starting with the single-lot sublot size, and with increments of 1, to obtain the re-optimized sublot size. The proposed heuristic is given next.

Heuristic for the lot-based sublot-size two-machine FSLS problem

Phase 1

Step 1. Input: N, (m=2), $su_{i,1}, p_{i,1}, su_{i,2}, p_{i,2}, Q_i \quad \forall i = 1,.., N$

Step 2. For each lot i, solve the single-lot two-machine problem. The optimal solution, based on the analysis in Chapter 3, is:

$$L_i^* = \arg\min_{L_i} \left\{ \begin{array}{l} M(L_i) = \left(\frac{Q_i}{L_i} - 1 \right) \cdot \max_{j=1,2} \{ su_{ij} + L_i \cdot p_{ij} \} + \sum_{j=1}^{2} \left(su_{ij} + L_i \cdot p_{ij} \right) \\ \left| L_i : \left[\sqrt{\frac{Q_i \cdot su_{i1}}{p_{i2}}} \right]^{\pm}, \left[\sqrt{\frac{Q_i \cdot su_{i2}}{p_{i1}}} \right]^{\pm}, \left[\frac{su_{i2} - su_{i1}}{p_{i1} - p_{i2}} \right]^{\pm}, 1, Q_i \right. \end{array} \right\} \quad (4.7.2\text{-}1)$$

Consider each lot to be a job.

With each of these jobs, associate the following “job” processing times on the two machines:

$$A_i = su_{i,1} + L_i \cdot p_{i,1} \quad ; \quad B_i = su_{i,2} + L_i \cdot p_{i,2}$$

Step 3. Apply Johnson’s rule on the entire set of jobs formed in Step 2 to obtain a sequence, *S*. This sequence remains unchanged in subsequent steps.

Step 4. Schedule the lots according to the sequence (with the sublot size computed in Step 2.) Compute the makespan, $M(S, \underline{L})$; Set $M^*(S, \underline{L}) = M(S, \underline{L})$

Phase 2

Step 5. Compute the total setup time associated with each lot on machine 2. Form a list, LIST, with the non-increasing total setup time ordering of the lots (except the first lot.) Place the first lot last in LIST.

Step 6. For the next lot in LIST, say lot *k*, re-optimize its sublot-size as follows:

6.1 $L_k = L_k + 1$; Compute the resulting makespan, $M^t\left(S, \underline{L^{new}}\right)$

6.2 If $M^t\left(S, \underline{L^{new}}\right) \le M^*\left(S, \underline{L}\right)$ then set $M^*\left(S, \underline{L}\right) = M^t\left(S, \underline{L^{new}}\right)$; Repeat 6.1. Otherwise, go to Step 7.

Step 7. Remove lot *k* from LIST. If LIST = empty, go to Step 8. Otherwise, repeat Step 6.

Step 8. Print the revised solution, $M^*(S, \underline{L})$; STOP.

Exp. (4.7.2-1) states that, in the case of two machines, the optimal sublot size, for a single lot, is either the rounded-down or the rounded-up value of one of the following four possibilities:

(a) Machine 1 dictates the makespan. In that case, the optimal sublot size is: $\sqrt{\frac{Q \cdot su_1}{p_2}}$.

(b) Machine 2 dictates the makespan. In that case, the optimal sublot size is: $\sqrt{\frac{Q \cdot su_2}{p_1}}$.

(c) Both machines dictate the makespan (i.e., the CP is comprised of some sublots on machine 1 and some on machine 2.) The intersection point corresponds to the optimal solution, i.e., the optimal continuous sublot size is: $\frac{su_2 - su_1}{p_1 - p_2}$.

(d) All the above result in a sublot size that is either greater than the lot size, *Q*, or smaller than the minimum discrete sublot size of 1. In that case, the solution will be either *Q* or 1 respectively.

The proposed heuristic overcomes the shortcoming of the optimal solution algorithm, by fixing the sequence and iterating, according to a prioritized list, to optimize the solution at each step, given the pre-determined sequence. This approach has been used by Dauzere and Lasserre (1997) for the job-shop problem with lot-attached setups, and has shown promising results. In the proposed heuristic, the order of complexity $O\left(Q^N\right)$ of the original optimal solution algorithm reduces to $O(Q \cdot N)$, due to the replacement of the exponential search, in the optimal solution algorithm, with a sequential search in the proposed heuristic. The overall order of complexity of the heuristic is $O\left(N^2 \cdot \log(N) \cdot Q\right)$. Although still pseudo-polynomial, it is not exponential in *Q*, and expected to perform reasonably when applied on practical problems.

Next, a numerical example (4-8) is utilized to demonstrate how the heuristic works. Consider a five-lot (two-machine) flow-shop problem. The data is provided in **Table 4.22**. The lot sizes are all set to 4 and, for simplicity, cases of fractional sublots are not considered (i.e., L=3 is excluded.) Thus, sublot-sizes of 1, 2, and 4 are evaluated. The number of combinations that would have to be evaluated by a total enumeration scheme is $3^5 = 243$.

Table 4.22. Data for numerical example 4.9

Lot	Proc. times		Setup times		Lot size
i	p_{i1}	p_{i2}	SU_{i1}	SU_{i2}	Q_i
1	4	2	2	1	4
2	2	4	1	2	4
3	3	3	2	1	4
4	3	3	1	2	4
5	4	3	1	2	4

The computations of Step 2 of the heuristic are summarized in **Table 4.23**. For each lot, *i*, the values of A_i, B_i, and the resultant single-lot makespan are shown. The optimal (unconstrained) sublot-size for the single-lot problem is indicated by '*'. The resultant sequence, determined in Step 3, is {2-5-4-3-1}. The corresponding schedule is depicted in **Fig. 4.7**. The overall makespan is 85. This concludes phase 1 of the heuristic.

Table 4.23. Computations performed in Step 2 for example 4.9

Lot #	Sublot size	***A_i*** $su_{i1} + L_i \cdot p_{i1}$	***B_i*** $su_{i2} + L_i \cdot p_{i2}$	***Makespan M(L)***
1	1	2+4=6	1+2=3	27
	2	2+8=10	1+4=5	25 (*)
	4	2+16=18	1+8=9	27
2	1	3	6	27
	2	5	10	25 (*)
	4	9	18	27
3	1	5	4	24
	2	8	7	23 (*)
	4	14	13	27
4	1	4	5	24
	2	7	8	23 (*)
	4	13	14	27
5	1	5	5	20 (*)
	2	9	8	26
	4	17	14	31

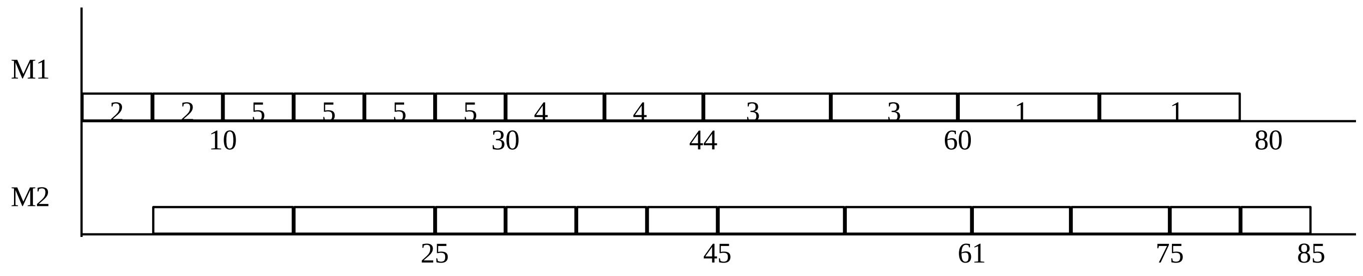


Fig. 4.7. Phase 1 schedule for example 4.9

Phase 2 of the heuristic begins with computing the total setup time of the lots on machine 2. This is done for each lot except the first lot in the sequence (lot 2) which is added to the end of the list. The total setup times associated with the sublots of lots 1, 3, 4, and 5, are 2, 2, 4, and 8 respectively. Consequently, the lot ordering is determined to be: LIST={5,4,3,1, and 2} (from the most setup to the least.)

Starting the re-optimization process of the sublot-sizes with lot #5, note that $L_5 = 1$. Therefore, $L_5 = 2$, is evaluated. This indeed results in a makespan reduction, from 85 to 83, despite the creation of a slot of 2 idle time units on machine 2. The resultant schedule is depicted in **Fig. 4.8(a)**. Incrementing L_5 to 4 leads to the further makespan reduction, to 82. This schedule is depicted in **Fig. 4.8(b)**. Similarly, for lot #4, which is next on LIST, incrementing L_4 from 2 to 4 results in further improvement with respect to the makespan. Although the idle time slots on machine 2 increase to 4, the makespan reduces from 82 to 81. This is depicted in **Fig. 4.8(c)**. Repeating the process with lot #3, which is next on LIST, incrementing L_3 from 2 to 4 also results in a makespan reduction (to 79). The schedule is depicted in **Fig. 4.8(d)**. The sublot size of lot #1, which is next on LIST, does not change, as incrementing it from 2 to 4 results in a makespan increase of 2 time units (to 81). Lastly, the first lot in the sequence, lot #2, is evaluated. Its sublot size is incremented from 2 to 4, to result in further makespan reduction, to 78. This is depicted in **Fig. 4.8(e)**. This is also the final solution of the heuristic. The final solution can be summarized as follows:

$$L_1 = 1; L_2 = 2; L_3 = 1; L_4 = 1; L_5 = 1; S = \{2\text{-}5\text{-}4\text{-}3\text{-}1\}; M(S, \underline{L}) = 78$$

As indicated earlier, the number of combinations that would have to be evaluated by a total enumeration scheme, in order to determine the optimal solution, is $3^5 = 243$. Performing such total enumeration on the given problem shows that the solution found by the heuristic is indeed optimal.

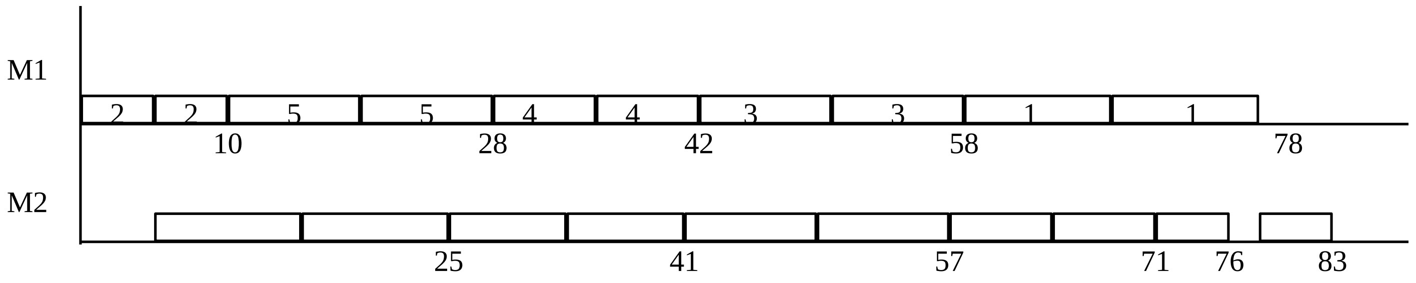


Fig. 4.8(a). Improved schedule for example 4.9: first iteration

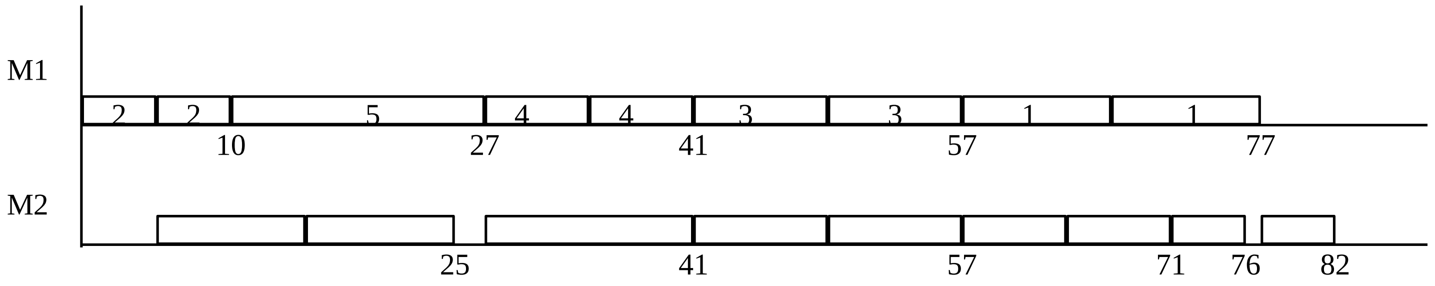


Fig. 4.8(b). Improved schedule for example 4.9: second iteration

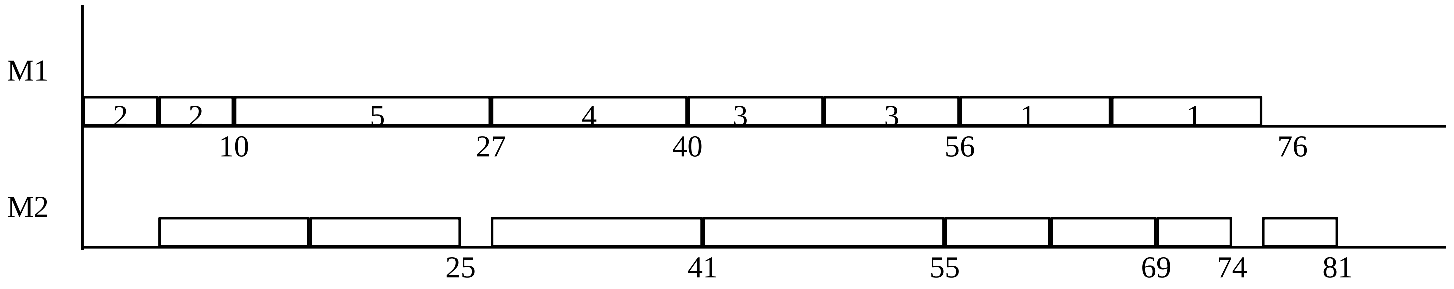


Fig. 4.8(c). Improved schedule for example 4.9: third iteration

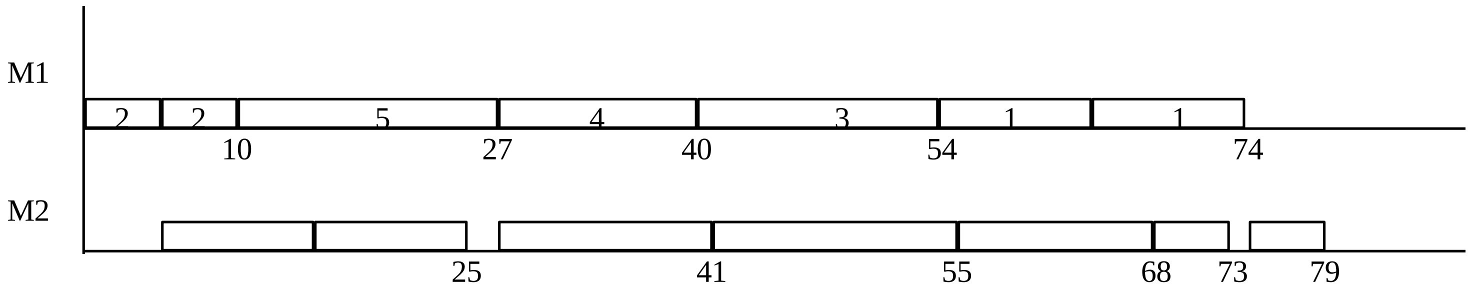


Fig. 4.8(d). Improved schedule for example 4.9 (fourth iteration**)**

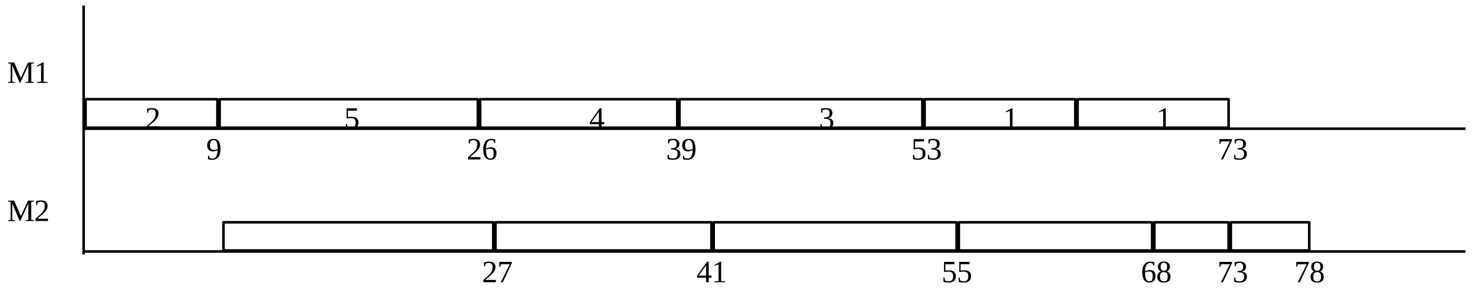


Fig. 4.8(e). Final schedule for example 4.9

The optimality of the heuristic was tested on 100 problem instances. The lot sizes and the unit processing times were generated from a uniform distribution, *U*[1, 5]. The setup times were generated from a uniform distribution, *U*[1, 3]. Small lot sizes were used to enable to solve the problems optimally, via the optimal solution algorithm, in a reasonable time. The results of this experiment were as follows. The heuristic found the optimal solutions in 87 cases (87%). In those cases where the heuristic resulted in a non-optimal solution, the gap from the optimal solution was only 2.43%, on the average, with 3.84% being the worst gap. Therefore, it appears that the heuristic produces satisfiable results, at least for small-scale problems.

4.7.3 Extensions to m-machine flow-shops

Sections 4.7.1 and 4.7.2 have dealt with the two-machine FSLS problem. Next, we propose a methodology for extending these results to *m*-machine flow-shops.

4.7.3.1 Methodology of solution

The *m*-machine FSLS problem is, as explained in the introduction of this Chapter, a very complicated problem, much more complicated than the two-machine problem discussed thus far. Consequently, the emphasis in the sequel is on developing efficient (near-optimal) heuristic solutions. These heuristics are built upon those introduced in the development of optimal solutions for the two-machine problems. The extension from two-machine FSLS problems to *m*-machine FSLS problems is illustrated in **Fig. 4.9**. As depicted in Fig. 4.9, the iterative process over the sublot-size values, which is utilized by the optimal solution algorithm for the two-machine FSLS problem, is preserved in the solution scheme for the *m*-machine FSLS problem. However, the sequencing procedure is replaced. Johnson's rule, which produces optimal sequences of the sublots for two machines, cannot unfortunately be extended for more than two machines. It is therefore replaced by the very efficient BMI heuristic, which was shown to be near-optimal for sequencing sublots on *m* machines (Section 4.6). Recall that, in the BMI heuristic, the unit processing time is used to determine the sequence. The exp. $su_{ij} + L \cdot p_{ij}$, which represents the sublot processing time in this case, is therefore utilized as the unit processing time. Note that, since the sublot size is varied over its entire range, all possible processing times for the sublots are considered. Larger than unit-sized sublot processing times must be considered as well since sublot-attached setups are considered here, while only lot-attached setups were considered for the LSSP, for which the BMI heuristic was originally developed.

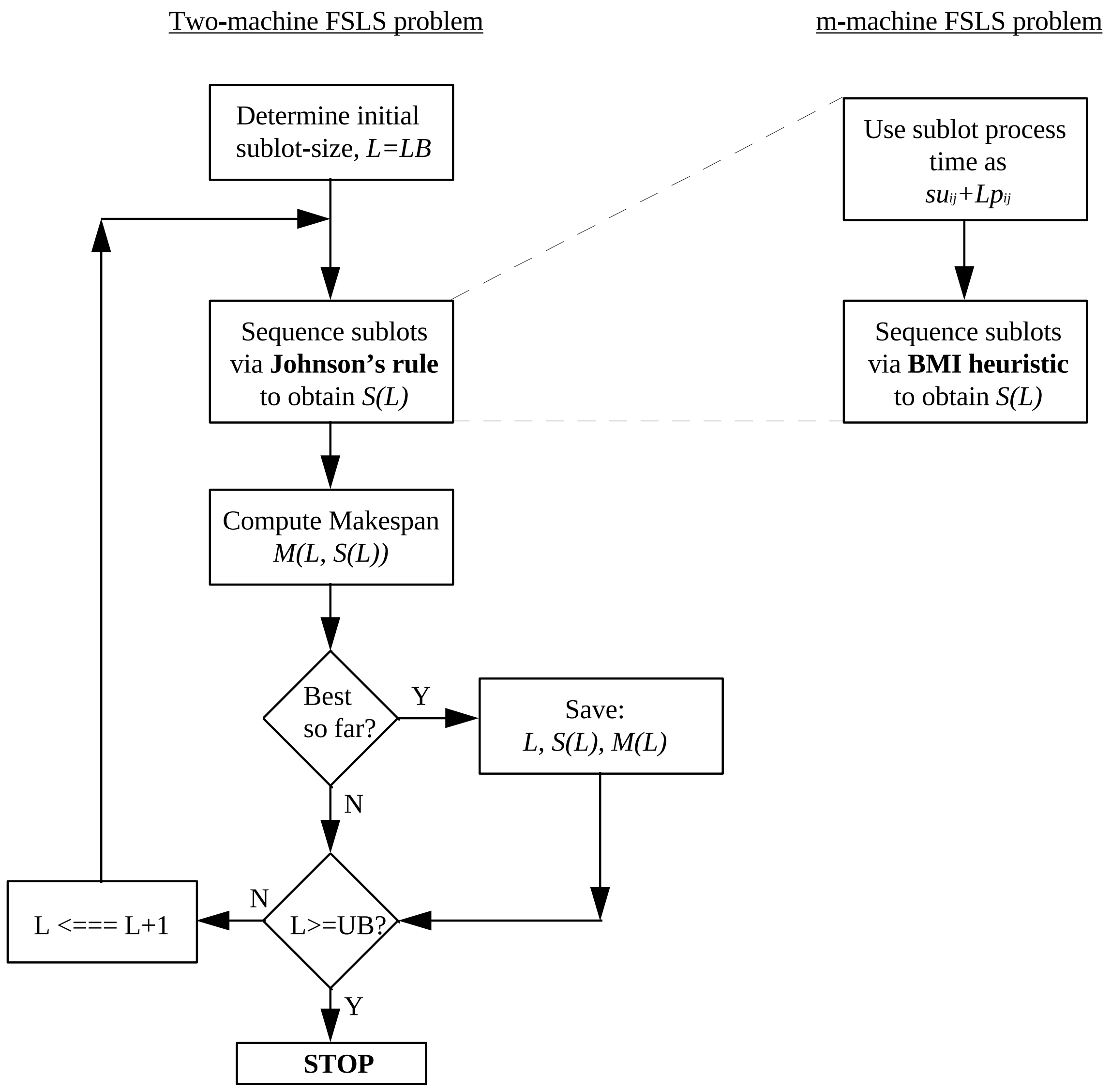

Two-machine FSLS problem
m-machine FSLS problem
Determine initial sublot-size, L=LB
Use sublot process time as su_ij+Lp_ij
Sequence sublots via Johnson's rule to obtain S(L)
Sequence sublots via BMI heuristic to obtain S(L)
Compute Makespan M(L, S(L))
Best so far?
Y
Save: L, S(L), M(L)
N
L>=UB?
N
L <=== L+1
Y
STOP

Fig. 4.9. Solution scheme for the two-machine and m-machine FSLS problems

4.7.3.2 Heuristic solution for the global sublot-size m-machine FSLS problem

The BMI-based heuristic for the global sublot-size m-machine FSLS problem

Step 1. Input: $su_{ij}, p_{ij}, Q_i \quad \forall i = 1,..,m$, and *UB* (Use $UB = \max_i \{Q_i\}$ in the absence of a tighter value)

Initialize sublot-size, *L*=*LB* (Use *LB* =1 in the absence of a tighter value)

Step 2. For each lot i, compute the number of "jobs" to be considered as follows: $n_i = \left\lfloor \frac{Q_i}{L} \right\rfloor$

Consider the last job to be of size $R_i = Q_i - n_i \cdot L$ (the "remainder" sublot).

Let: $p_{ir} \leftarrow su_{ir} + L \cdot p_{ir} \quad \forall r = 1,..,m$

Step 3. Apply the BMI heuristic (the version with intermingling in Section 4.6.4) on the set of job types formed in Step 2 to obtain a sequence, *S(L)*.

Compute the resultant makespan of the this sequence, *M(S(L))*.

Save the best sequence so far, *S*(L)*.

Step 4. If *L*=*UB* then go to 5 ; Otherwise, set *L*=*L*+1, and go to Step 2.

Step 5. Print the optimal solution: *L*, S*(L*), M*(S*(L*))* ; STOP

Given the enormous number of feasible solutions to the problem, even for small scale instances, the only argument to justify the near-optimality of the proposed heuristic is based on logical derivation. We have shown that the BMI heuristic solves the sequencing problem, under both intermingling and no intermingling, near-optimally. It should also be apparent that, by searching over the entire feasible space of sublot-size(s), the near-optimal sublot-size(s) would be identified. Hence, the proposed heuristic, which is comprised of these two pieces, should result in near-optimal solutions. This will further be demonstrated in Chapter 5.

4.8 Summary of Results

In this Chapter, the Lot Streaming Sequencing Problem (LSSP) and the Flow Shop Lot Streaming (FSLS) problem have been dealt with. An efficient near-optimal heuristic, referred to as the Bottleneck Minimal Idleness (BMI) heuristic, has been developed for the LSSP. A comprehensive experimental study has been conducted to demonstrate that the non-intermingled BMI heuristic produces near-optimal solutions. This study has also shown that the BMI heuristic outperforms the FIHLS heuristic. The FIHLS heuristic, which has also been developed in this Chapter, is an extension of the Fast Insertion Heuristic (FIH), the best known heuristic for traditional flow-shop sequencing problems.

For the FSLS problem, several algorithms have been developed. For the two-machine FSLS problem with an identical sublot-size for all lots, an optimal pseudo-polynomial solution algorithm, of complexity $O(N \cdot \log(N) \cdot Q_{\max})$, has been proposed. This algorithm is expected to be fairly fast for reasonable lot-sizes. For the case in which the sublot-sizes are lot-based, optimal and heuristic procedures have been developed. The heuristic procedure has been developed to, primarily, reduce the complexity of the optimal solution algorithm. It indeed reduces the complexity to $O\left(N^2 \cdot \log(N) \cdot Q_{\max}\right)$. Although still pseudo-polynomial, it is not exponential in *Q*, as is the optimal solution algorithm. The heuristic procedure includes a construction phase and an improvement phase. It first attempts to find a near-optimal sequence for the lots and then, at the improvement phase, given the sequence, it attempts to optimize the lot-based sublot-sizes of each of the lots. Extensions of the solution algorithms proposed for the two-machine FSLS problem have been presented. In these extensions, the only modification is in the mechanism that is utilized to generate sequences. Johnson's rule, which yields optimal sequences in two-machine flow-shops, is replaced with the BMI heuristic, which has been shown to result in near-optimal sequences for general *m*-machine flow-shops under lot streaming.

Chapter 5: Dynamic Lot Streaming With Equal Sublots

5.1 Introduction

The previous two Chapters have alluded to the analysis of lot-streaming in static environments, in which the lots are all available at time zero, and once production decisions have been made, they are not changed until production terminates. In reality, however, there are numerous situations in which this is simply not true. For example, when the production system is driven by unexpected market demands. In such cases, orders keep coming in for production and the production never terminates. Moreover, the lots for which the orders are produced are not all available at time zero.

Similar questions are pertinent in these situations as well. Can lot streaming still be beneficial in such cases? If so, under what conditions? And if these conditions are indeed satisfied, what is the best way to utilize lot-streaming? These questions are addressed in this Chapter. To address these questions, a simulation-based comparative study is carried out. The performance measures that are utilized for operational performance evaluation, and discussed in more length in Section 5.3, are similar to those used by the industry. The primary objective of the simulation study is to demonstrate that a dynamic version of the BMI heuristic, developed in this Chapter, outperforms existing WIP management rules that are commonly used today in flow-shop facilities.

5.2 Conditions for Lot Streaming to be Beneficial

Chapters 3 and 4 have both demonstrated the many potential benefits of lot streaming in static flow-shop production systems. Various measures have been utilized to show the effectiveness of lot streaming. The makespan measure was utilized in Chapter 4 to evaluate the performance of a couple of heuristics in static multi-product multi-machine flow-shop production systems. In Chapter 3, on the other hand, the assumption of a single product made it possible to use other performance measures as well, such as the MFT, average WIP, and cost-related. Lot streaming was found to be very beneficial with respect to all these measures. However, the analyses in Chapters 3 and 4 rely on the assumption that all the lots are available at time zero and that, the number of lots considered is finite (i.e., the static case). In this Section we set the stage for the dynamic-case analysis by showing that the following is true:

- Certain conditions, which essentially imply system stability, must be met for lot-streaming to be beneficial with respect to dynamic measures.
- The makespan measure (i.e., the completion time of all jobs measured from the start of processing of the first job) is not a useful measure since it is static in nature.

We begin with the latter argument. It has been observed by several researchers that makespan improvement rates decrease as N increases (Vickson and Alfredsson, 1992 ; Kropp et al, 1988). The next two theorems prove that the makespan measure is not a useful measure to be utilized for evaluating lot streaming in dynamic multi-product multi-machine flow shop production systems, since makespan improvement rates do indeed diminish to zero as N increases. These theorems show that as long as an infinite number of lots is considered, lot streaming would not result in any savings with respect to the makespan.

Theorem 5.1. In the two-machine case, makespan reduction due to lot streaming decreases to zero with the number of lots.

Proof.

See Appendix E.

The above theorem is easily extended to m machines in Theorem 5.2.

Theorem 5.2. (Generalization of Theorem 5.1 to the m-machine case)
In the general m-machine case, makespan reduction due to lot streaming decreases to zero with the number of lots.
Proof.
See Appendix E.

Theorem 5.2 should not come as a surprise. Recall that, by definition, the makespan is the total completion time of all lots (measured from time 0.) Clearly, as the number of lots increases, the makespan, which is the total completion time of the lots, increases as well. Since one (or more) of the machines would necessarily be a bottleneck, it would dictate the makespan in the presence or absence of lot streaming. However, since it is a bottleneck, it would not idle in both cases (for a sufficiently large N) and, therefore, the makespan would be approaching the same value under both cases. This static nature of the makespan is summarized in the following remark.

Remark 5.1. Lot streaming is beneficial, with respect to a makespan objective, only if the number of lots considered is finite (i.e., in static problems.)

Note that remark 5.1 is not limited only to the case of negligible setup times. In general, setup times tend to actually make lot streaming less attractive since, if the sublots are sequenced in an intermingled manner, there is more setup time associated with the LS-schedule than there is with a non-LS-schedule. Therefore, Remark 5.1 remains in tact for the case of setup as well.

Next, we show that benefits from lot steaming in a dynamic flow-shop production system are possible with respect to other performance measures, provided that certain operating conditions are satisfied, namely that the system is stable.

Theorem 5.3. In the general m-machine flow-shop, if the rate at which lots arrive to the first machine is higher than the bottleneck processing rate (i.e., the bottleneck utilization is 100% or *more), mean flow time (MFT) and the average WIP level reductions due to lot streaming decrease to zero with the number of lots.*

Proof.

As long as there exists a workload that is higher than the system processing level, queue(s) would build up within the system before the bottleneck machine(s). If in fact, a bottleneck machine does exist, which builds up an increasing queue of lots, then, in the long term, most of every sublot's flow time in the shop would be spent awaiting processing on the bottleneck machine. This, in turn, would make any time savings, if at all, insignificant. Similarly, the average WIP level would mainly be dictated by the WIP levels accumulated before the bottleneck machine(s). By utilizing lot streaming, this WIP level would not decrease since it would not matter whether the sublots reach the queue independently or together - in both cases, they would simply be added to the queue and wait a substantial amount of time for further processing. The above argument is in agreement with Little's law which suggests that, if there is no benefits with respect to the MFT, there cannot be any benefits with respect to the WIP, and vice versa. QED.

Note that, over the long term, if the system experiences a lower than 100% utilization, the queues within the system should never increase to infinity. In fact, if a certain utilization level is kept fixed, then the queues should reach a steady state. At these levels, it may still be beneficial to stream arriving lots through the system, since this may result in some of the sublots being processed faster.

5.3 The Impact of WIP Policies

Our primary objective in this Chapter is to show that a dynamic version of the BMI heuristic outperforms existing WIP rules commonly in use, in a lot streaming environment. To understand how different WIP scheduling policies can result in different performance measures, the impact of WIP on system performance is explored in this Section.

We start with Little's famous law. This law states that, in a steady-state system, there exists a relationship between the WIP inventory level, the production rate (PR) of the system (measured in products per time unit), and the mean flow time (MFT), or as it is more commonly referred to, the cycle time (CT). Little's law can be prescribed as follows:

$$PR = \frac{WIP}{CT} \tag{5.3-1}$$

It should be intuitive that, as long as WIP progresses through the manufacturing system, and is not intentionally stopped or delayed, the production rate of the system would be the same in the long run. This is true simply because the workload (or the utilization) on each machine remains the same and, therefore, its output rate remains the same. If the output rate of each system component (machine) remains the same, then the output rate of the entire system must be the same.

If this is indeed the case, then what could we possibly gain by varying the WIP policy? The answer is not intuitive. It turns out that different WIP policies remove WIP differently out of the system. Some of these policies remove WIP consistently faster than others. This, in turn, decreases waiting times for the machines which leads to incoming products spending less time in the system. In terms of Little's law, this means that, under some **effective** WIP policies, WIP would be lower than under others and, respectively, cycle time would be lower as well. Note that the ratio of the two, however, would not change, to comply with Little's law.

To further understand how a WIP policy can affect system performance, consider the following numerical example. Three lot-types are to be produced repeatedly in a two-machine flow-shop. The release times of the lots to the system, the lot-sizes, and the unit processing times (in minutes) are given in **Table 5.1**. Each of the three lots is released to the system repeatedly every 100 minutes (this makes the problem dynamic in nature.)

Table 5.1. Data for numerical example 5.1

Lot-type	Release times	Processing times		Lot size
i	R_i	p_{i1}	p_{i2}	Q_i
1	0	5	5	4
2	10	10	10	4
3	20	5	5	4
Total proc. time:		80	80	

Assuming a sublot size of *L=1*, consider the following two schedules. The first is obtained by sequencing the lots in FIFO/FCFS order, i.e., by their release times to the system. The FIFO sequence in this case is {1-2-3}. The second sequence that is considered is, just for the sake of comparison, a LIFO-based sequence. Since at time 0, only lot-type 1 is available, it is the first lot in the sequence. However, by the time lot-type 1 is completed on machine 1, lot-types 2 and 3 are both in the queue. Since lot-type 3 enters the system after lot-type 2, the resultant sequence is {1-3-2}. **Fig. 5.1** depicts the corresponding schedules of these two sequences.

Note that the completion times of the lots 1, 2, and 3 under sequence {1-2-3} are 25, 70, and 85, respectively, while the completion times under sequence {1-3-2} are 25, 45, and 90, respectively. Thus, the lot-based cycle times in the two cases are as follows:

$$CT(FIFO) = \frac{\sum_{i=1}^{N}(C_i - R_i)}{N} = \frac{(25-0)+(70-10)+(90-20)}{3} = 51.7 \qquad (5.3\text{-}2)$$

$$CT(LIFO) = \frac{(25-0)+(45-20)+(90-10)}{3} = 43.3$$

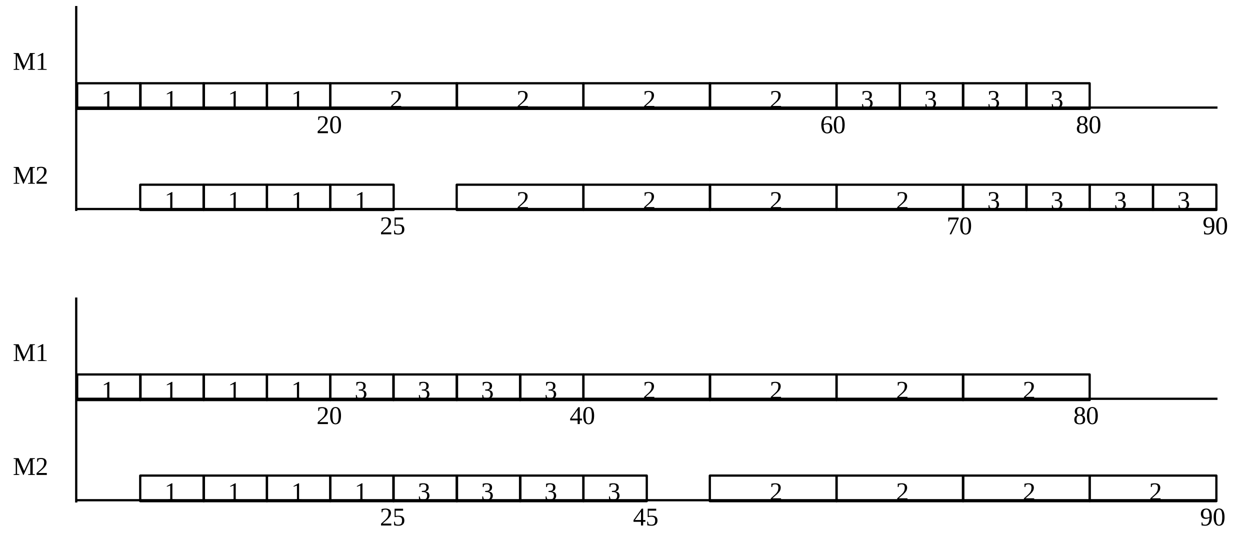


Fig. 5.1. FIFO and LIFO schedules, example 5.1

Similarly, the average lot-based WIP inventory in the system, under the two WIP rules, can be derived from **Table 5.2**, which depicts the number of lots in the system throughout the 100 minutes cycle. The <u>lot-based</u> WIP averages are as follows:

$$WIP(FIFO) = \frac{\int_{t=0}^{T} WIP(t)\cdot dt}{T} = \frac{1\cdot 10 + 2\cdot 10 + 3\cdot 5 + 2\cdot 45 + 1\cdot 20}{100} = 1.55\,\text{lots} \qquad (5.3\text{-}3)$$

$$WIP(LIFO) = \frac{1\cdot 10 + 2\cdot 10 + 3\cdot 5 + 2\cdot 20 + 1\cdot 25 + 1\cdot 20}{100} = 1.30\,\text{lots}$$

Table 5.2. Lot-based WIP inventory levels over time, example 5.1

Sequence \ Time	[0,10]	[10,20]	[20,25]	[25,45]	[45,70]	[70,90]	[90,100]
FIFO {1-2-3}	1	2	3	2	2	1	0
LIFO {1-3-2}	1	2	3	2	1	1	0

Note that, under both the FIFO rule and the LIFO rule, the production rate of the system is identical, that is 3 lots per 100 minutes, or 33.3 minutes per lot on the average. Also note that the utilization of the machines is the same in both cases (80% on the average.) Nevertheless, the LIFO rule outperforms the FIFO rule with respect to both the average cycle time and the average WIP inventory in the system. The average improvement that the LIFO rule makes relative to the FIFO rule is 19.2%, a significant improvement indeed.

Note that the computations above have all been made with respect to lot-based measures. These computations could have been performed in the same manner with respect to sublot-based measures. The results would have been similar.

5.4 Performance Measures

In the previous Section, it has been shown that different WIP policies can lead to different results with respect to system performance measures. In this Section, these performance measures, as well as others, are formally introduced. These performance measures are, in many cases, equally important, and are in constant use in the industry, to evaluate different possible WIP management policies. They can all be obtained directly from simulation statistics.

Before we introduce these performance measures, we note that there exists a general trade-off between two key measures, namely the output rate and the cycle time. In order to maximize revenues, maximum output rate is desirable, which leads to maximal utilization of the machines and large WIP within the system, to ensure constant feeding of the bottlenecks by, essentially, protecting them from upstream machine failures and variable processing times. On the other hand, in order to deliver specific orders on-time, quick delivery to customers is desirable. This, in turn, leads to trying to keep minimum WIP since higher WIP means higher waiting times for new lots and new orders. In this context, effective WIP management rules attempt to reduce WIP within the system, thereby reducing cycle time, while maintaining the same output rate.

5.4.1 Cycle time

Cycle time, or as it is more often referred to, Through-Put Time (TPT), is a measure of the Mean Flow Time (MFT) of a lot, or a sub-lot, whichever is the basic entity that flows independently throughout the system. This is the most commonly used indicator for the pace at which the flow-shop works. It directly affects the time it would take to respond to incoming orders. The average cycle time is the mean flow time across all the lot/sub-lot (part) types that are produced by the flow-shop. The cycle time of a certain part type may obviously be higher or lower than the average cycle time, depending upon the priority which it receives. This priority, which eventually determines the cycle time per part type, is determined by the WIP management rule that is applied.

In the simulation, the cycle time of each sublot is recorded as soon as it is completed. It is then used in the calculation of the average cycle time within each simulation time period (or snap). Let ct_{pr} be the cycle time of the *p*-th sublot (*p*=1,..,P) at the *r*-th simulation snap (*r*=1,..,R). The average cycle time over each snap is as follows:

$$CT_r = \frac{\sum_{p=1}^{P} ct_{pr}}{P} \tag{5.4-1}$$

The overall cycle time average is:

$$CT = \frac{\sum_{r=1}^{R} CT_r}{R} \tag{5.4-2}$$

5.4.2 Cycle time variability

Cycle time variability is measured through the standard deviation of the cycle time. This can be done for both the average cycle time as well as for the cycle time for each part type. Cycle time variability indicates what the level of certainty is, that the next part would be produced in the same cycle time as the average cycle time. This measure is important since, the lower the cycle time variability, the higher the capability is to anticipate whether due-dates would be met. A similar measure that could alternatively be utilized is the 95% (confidence level) cycle time, which represents the cycle time for which 95% of the individual flow-times of the parts fall under. If the individual flow-times of the parts follow the normal distribution, it can be prescribed that:

$$CT_r(95\%) = CT_r + 1.645 \cdot S_r \tag{5.4-3}$$

where:

$$S_r = \sqrt{\frac{\sum_{p=1}^{P}\left(CT_{pr} - CT_r\right)^2}{P-1}} \tag{5.4-4}$$

Hence, there exists a relationship between the 95% cycle time, the average cycle time, and the cycle time variability. The average cycle time is tracked explicitly and, therefore, one more measure should be used. In this study, we shall use S_r, the standard deviation of the cycle time.

Note that each simulation snap may result in a different cycle time standard deviation, just like it may result in a different cycle time average. Thus, similar to the overall average cycle time, the overall average of the standard deviation of the cycle time, S, can be obtained as follows:

$$S = \frac{\sum_{r=1}^{R} S_r}{R} \tag{5.4-5}$$

5.4.3 Output variability

The output variability, similar to the cycle time variability, is measured through the standard deviation, but of the output instead of the cycle time. This is a measure of the stability of the system to produce consistent and predictable output. Higher variability in the output can ultimately lead to increased production costs (Kalir and Sarin, 1998). It is therefore desirable to have minimum output variability. The output variability should not be confused with cycle time variability. While the former measures the variability in the times between sublot departures the latter measures the variability in the times that sublots spend in the system. Note that, although the output volume is the same under different WIP management rules, the output variability is not necessarily identical.

In the simulation, the completion times of the sublots and the corresponding times between departures are recorded for all completed sublots. They can then be used to compute the standard deviation in the output.

5.4.4 Average WIP

The average WIP inventory is a measure of the work that accumulates in the queues within the system. It is completely correlated with the average cycle time. The relationship between the two is given by Little's law, which also indicates that, for the same output, one is a multiplication by a constant of the other. Clearly, the larger the WIP, the more it takes, on the average, to complete a part, since there is more waiting time in the queues. The focus would therefore be on the cycle time, keeping in mind that the behavior of the average WIP is identical.

5.4.5 WIP-Turns

WIP-Turns is a measure for the speed at which certain marked operations are performed. In semi-conductor processes, operations within the process flow of the different products can be divided into two types: operations that change the nature and characteristics of the product, called 'activities', and operations that do not change the nature and characteristics of the product, called 'moves'. WIP-Turns is a complicated function that measures the speed at which the activities of the different parts in the system are performed. In steady state, this function reduces to the following formula:

$$WT_i = \frac{Act_i}{CT_i} \tag{5.4-6}$$

where:

WT_i - WIP-Turns per product type *i*.

Act_i - Activities per product type *i*.

CT_i - Cycle time for product type *i*.

Since, in steady state, as can be seen from exp. (5.4-2), the WIP-Turns performance in product type *i* behaves essentially as the inverse of the cycle time for product type *i*, we shall only utilize the latter in our performance metric.

To summarize, three performance measures are of primary interest and focus. These are the cycle time, the standard deviation of the cycle time, and the standard deviation of the output. The other three measures, average WIP, 95% cycle time, and WIP-Turns follow the pattern and the behavior of the measures under focus.

5.5 The Dynamic BMI Heuristic

The BMI heuristic was developed and presented in length in Chapter 4, to address the problem of minimizing the makespan of the **static** version of the lot-streaming sequencing problem (LSSP). In the static LSSP, all the information about the work to be performed is known in advance. The dynamic version of the problem, which is discussed in this Chapter, differs from the static problem in that respect. The information is not known in advance, and is even, often times, not deterministic (e.g., the arrival of new orders.) The decisions must, therefore, be re-visited and revised as new information is coming in.

The revised decisions, that are made based on the new information may be applied to new work as it is introduced to the system or, better yet, to “old” work that already awaits further processing in the system. The main decisions to be made in a flow-shop pertain to the ordering of work in the queues between the machines. In the static problem, the only decision to be made pertains to the sequencing of the lots, or the sublots. Since all the information is known in advance, the sequence can be determined up-front. The resultant sequence is a permutation sequence, since no deviation is made from the original sequence, as work progresses through the system. In contrast, more than a single decision is required in the dynamic problem. In fact, decisions need to be made with regard to the ordering of each of the queues in the system. Moreover, as the work content that these queues store changes over time, the decisions may have to be revised, depending upon the change in the state of the queues and the entire system.

It should be clear from the above discussion that the BMI heuristic must be adjusted accordingly, to account for the dynamic nature, prior to its implementation on the dynamic problem. Fortunately, this task of adjusting the BMI heuristic to the dynamic problem is a simple one. Recall that the BMI heuristic for the static problem essentially works as follows:

- It is applied on the queue of the first machine (permutation sequence.)
- It identifies the downstream bottleneck by evaluating the total processing time per machine. The machine with the highest total processing time is the bottleneck.
- It schedules the next sublot according to lot-dominance and the lexicographic rule.
    - It performs this scheduling activity for all the sublots up front.
- It schedules the machines downstream of the bottleneck in the same sequence (since only a permutation sequence is considered.)

The adjustments for the dynamic problem are, respectively, as follows:

- Apply at any queue (i.e., not necessarily a permutation sequence.)
- Identify the downstream bottleneck by evaluating the utilization per machine. The machine with the highest utilization is the bottleneck.
- Schedule the next sublot according to lot-dominance and the lexicographic rule.
    - Re-perform this scheduling activity as a machine becomes available.
- Schedule machines which are downstream of the bottleneck, to remove WIP as fast as possible, using the least 'tail' first rule

A summary of the similarities and the differences between the static problem characteristics and the static BMI heuristic, to the dynamic problem characteristics and the dynamic BMI heuristic, are provided in **Table 5.3**.

Table 5.3. Problem characteristics and the BMI heuristic:
comparison for the static and the dynamic lot streaming problems

Problem Characteristics	**Static**	**Dynamic**
Availability of work	Time zero	During run
Variability of data	Deterministic	Stochastic
BMI Heuristic	**Static**	**Dynamic**
Sequencing decision	First queue	All queues
Decision timing	At time zero (up front)	Throughout run (during)
Sequence type	Permutation	Non-permutation
Bottleneck identification	Max total processing time	Max utilization
Bottleneck sequencing rule	Lot-dominance and the lexicographic rule	Lot-dominance and the lexicographic rule
Sequencing downstream from bottleneck	Same sequence (permutation)	Least 'tail' rule
Nature of sequence	Fixed	Adjusted over time

Note that the dynamic BMI heuristic is a relatively simple rule to apply in a manufacturing environment. If the proportions of the different products are known, it is possible to identify the bottleneck machine(s) in advance and, consequently, to determine the priorities of the sublots at each queue according to the lexicographic rule and lot-dominance up-front. Once the system reaches steady-state, and the bottleneck machine establishes a sufficient amount of work in queue, the priorities on the upstream machines become static. There is no reason to deviate from them since the bottleneck is not going to switch to a different machine, and a sufficient amount of work would be established before it. For the same reason, the priorities on downstream machines are not going to change either. Thus, although the problem is dynamic, as long as the proportions of the different products to be produced are known in advance, in steady state, the BMI heuristic reduces to a static (lexicographic) rule, in which the lots can be prioritized regardless of the changes in the system's state over time.

5.6 Factors in the Experimental Study

Our primary objective in the simulation study, that will be presented in the following Sections, is to test the effectiveness of the BMI rule. To that end, the values of the performance measures discussed in Section 5.3, under various operating conditions, will be utilized to evaluate the performance of the BMI rule against the performance of other commonly used WIP management rules. In this Section, a description of these rules is provided. A secondary objective in the simulation study is to evaluate the potential impact of the operating conditions on the results, and the performance of the various rules. Two factors that dictate the operating conditions are the level of randomness that is experienced by the system and the level of workload that the system is introduced with. These two factors are discussed in this Section as well.

5.6.1 WIP management rules

Several commonly used WIP management rules are utilized as a benchmark to evaluate the effectiveness of the BMI rule. These rules are described next.

- FIFO (First-In-First-Out): parts are processed on each of the machines in the order they enter the queue.
- ESD (Earliest Start Date): parts are processed on each of the machines in the order they enter the system.
- SPT (Shortest Processing Time): parts are processed on each of the machines in the non-decreasing order of their processing times on the machine.
- X-Theoretical: parts are processed on each of the machines in the non-increasing order of their ratio of actual processing time to the current operation to the theoretical processing time to the current operation. A sublot with a higher value on this measure has encountered more delays than a sublot with a lower value and is, therefore, given higher priority. This measure is commonly used in the semi-conductor industry to identify work that is behind. Parts for which it should, theoretically, take a relatively short time to get to a certain operation (i.e.,

without considering waiting time) but it actually takes a relatively long time to get to that operation, due to the actual waiting time, receive higher priority at the machine. The motivation is to try and reduce the remainder of their time in the system, thus reducing the overall cycle time.

The first two rules (FIFO and ESD) are considered because, traditionally, they have been used as a common benchmark for the evaluation of new rules. It can easily be shown that, for regular flow-shops (i.e., without re-entry), these two rules would give identical results and, therefore, only FIFO is depicted in the sequel.

The third rule, SPT, is considered here since it is known to be optimal with respect to the MFT measure for the static proportionate flow-shop sequencing problem (for proof, see Pinedo, 1995.) Although proportionate flow-shops are special cases of regular flow-shops, in which the processing times of each job on all the machines is identical ($p_{ij} = p_i \ \forall i$), there is still numerous studies to show that SPT performs well on regular flow-shops and flexible flow-shops in general (see: Gupta et al, 1989 ; Montazeri and Van-Wassenhove, 1990 ; Chan and Bedworth, 1991.)

The fourth rule, denoted as X-Theoretical, also known in the literature as SDT or SPT/TOT (Gupta et al, 1989), is considered an efficient rule commonly used in the semi-conductor industry, where flow-shop processes are highly re-entrant. The X-Theoretical rule is similar to the Critical Ratio rule, described in McCutchen and Lee (1995), which performs best in a semi-conductor fabrication environment when compared against FIFO, EDD, and SLACK-based rules. According to Montazeri and Van-Wassenhove (1990), this rule performs best, out of 16 rules that were examined, with respect to the MFT measure in a flexible flow-shop. However, it also introduces a high variability to the system in terms of machines' imbalance and output.

Three more WIP rules are considered, in addition to the above, for a subset of the simulated cases. These three are as follows:

- LPTR (Least Processing Time Remaining): parts are sequenced on each of the machines in the non-decreasing order of their remaining processing time. The remaining processing time is the sum of processing times at all the remaining steps.

- LBA (Least Balance Ahead): the part that has the fewest sublots of the same type at its next machine is selected. This rule attempts to balance the mixture of part types at the various machines throughout the flow-shop.
- LLA (Least sub-Lots Ahead): the sublot that has the fewest sublots (of any type) at its next machine is selected. This rule attempts to ensure that all the machines are constantly fed with work.

Although these rules do not explicitly attempt to minimize cycle time, it is interesting to compare their performance to the performance of the others.

5.6.2 Operating conditions: factors of randomness

A secondary objective in the simulation study is to evaluate the potential impact of certain operating conditions on the performance of the various rules. One aspect of the operating conditions pertains to the level of randomness within the system. There are essentially three factors that introduce randomness into manufacturing systems. These three factors are as follows:

- Order release patterns: the orders, or the parts, may be introduced to the system in a non-deterministic manner. This represents the case of a produce-to-order environment, where the system produces varying proportions of each product type over time and, cannot respond to future demand in advance. In a simulation model, this can be reflected by assigning exponential distributions to the arrival times of new parts to the system.
- Processing time variability: the various operations on the different machines may not necessarily be deterministic. In fact, they may have various distributions with various degrees of variability. In the simulation model, this can be represented accurately by assigning distributions with the desired degree of variability to the processing times of each operation.
- Machine breakdown: it is a well known fact that machines are not totally reliable. They are prone to failures. Associated with machine failures are parameters that can characterize the nature of the failures. Typically, the Mean Time Between Failures (MTBF) and the Mean Time To Repair (MTTR) are utilized to characterize the repetitious cycle of the failure and, consequently, the machine's expected availability. Each of these two parameters, the MTBF

and the MTTR, may have a distribution associated with it. This can be represented accurately in the simulation model as well.

In the simulation study, the effects of the first two are considered. Experiments are conducted once without the consideration of the above factors, and once with the consideration of the first two factors.

5.6.3 Operating conditions: system loading and bottleneck effects

A second aspect of the operating conditions, that may impact the performance of the various WIP rules, pertains to the level of system workload and balance. In this case, we consider three possible factors as follows:

- Bottleneck idleness: The idleness percentage on the bottleneck machine indicates how constrained the system is. Clearly, the smaller the bottleneck idleness, the larger the WIP and waiting times are in the queues upstream of the bottleneck machine. While this gives more opportunities in selecting work from the queues, it may reduce the impact of different WIP rules due to the increased amount of waiting times in the system and, specifically, prior to the bottleneck.
- Bottleneck location: This may clearly impact the effectiveness of certain rules since, in the process of prioritizing work for the machines, some rules are affected by the amount of time that parts have already spent in the system. Thus, they prioritize work differently, dependant upon whether the majority of the time spent in the system has yet to come, because the bottleneck machine is downstream, or has already been encountered, because the bottleneck machine is an upstream machine.
- System balance or the number of similar bottleneck machines: There is obviously an amplified impact on system and WIP rules' performance in cases where there exists more than a single bottleneck machine. Not only would the waiting times increase, but also the work would be prioritized differently by rules that estimate the amount of work to be performed or that has already been performed.

In the simulation study, effects of all of these factors shall be considered. Experiments will be conducted, that consider two levels of bottleneck idleness, two locations for the bottleneck machine, and cases where there exists more than one bottleneck machine within the system.

5.6.4 Experimental design

A full factorial experimental design has been carried out. The experiments are summarized in **Table 5.4**. There are a total of 10 different experiments, resulting from the different levels of the factors that dictate the operating conditions, each of which has been carried out with the five different WIP rules. Thus, the total number of experiments is 50. Each experiment includes a number of simulation replications, with each of the various WIP rules considered, to ensure that the simulation results are statistically valid.

Table 5.4. Experimental design for simulation study

#	Bottleneck Utilization	Bottleneck Location	Randomness	WIP rules
1	High	Beg-Center	Deterministic	All WIP rules
2	High	Beg-Center	Random	All WIP rules
3	High	Center-End	Deterministic	All WIP rules
4	High	Center-End	Random	All WIP rules
5	Medium-High	Beg-Center	Deterministic	All WIP rules
6	Medium-High	Beg-Center	Random	All WIP rules
7	Medium-High	Center-End	Deterministic	All WIP rules
8	Medium-High	Center-End	Random	All WIP rules
9	Medium-High	Both locations	Deterministic	All WIP rules
10	Medium-High	Both locations	Random	All WIP rules

5.7 The Simulation Model

The simulation model that is described next was developed using the AutoSched AP software© by Auto-Simulations, inc. It is a C++ object oriented based application. The inputs are presented to the simulation model in tabular format, using a Microsoft EXCEL© interface, by Microsoft corp. There are four main worksheets, which are used to define the four necessary manufacturing system components in the simulation model. These are:

- The Orders worksheet
- The Stations worksheet
- The Parts worksheet, and
- The Routes worksheet.

Additional optional worksheets are used to define non-standard WIP management rules (such as the BMI rule), the format and the statistics collected and presented in the output reports, the setups and the layout locations of the machines, the preventive maintenance schedules and the expected unscheduled downtimes on the machines, and operator-related input, such as operator certifications required to support certain machines, the periods of availability and unavailability of operators, etc.

Next, we briefly describe the type of information that is specified in each of the four mandatory input worksheets. An example of each of the worksheets can be found in **Appendix G**. For more detail, refer to the AutoSched AP software© user manuals.

5.7.1 The Orders worksheet

This worksheet contains information on the orders, such as the product types listed on each of the orders ; the amount of pieces in each of the lots, or the sublots, of the various product types ; the starting date, or the date of arrival of the order, and each of its specific sublots, in the system ; the due date of the order, and each of its specific sublots (optional) ; the mean time between arrivals (as a constant or a distribution) of identical orders, lots, or sublots ; and, the number of times these orders, lots, or sublots, repeat throughout the simulation.

5.7.2 The Stations worksheet

This worksheet contains information on the workstations that perform the processing of the orders. It contains the names of the station families, and the names of the stations, or the machines, within each station family. A station family may consist of several machines of different capabilities ; the quantity of the machines of each type ; and the dispatching rules to be used when machines select work from their queues.

The RULE field in this worksheet indicates the type of rule to be used once a machine has become available, i.e., when retrieving the next lot or sublot from the queue. The FWLRANK (Family Work List RANK) field specifies the method by which lots in the queue are to be ranked. Re-ranking of the lots in the queue is done each time that new work is added to the queue if the FWLRERANK field is set to ‘Yes’.

5.7.3 The Parts worksheet

This worksheet contains the names of the products ; the route files in which the product routings are stored, and ; the route names. It serves as a link between product types and their respective route file(s).

5.7.4 The Routes worksheet

The routes worksheet includes information pertaining to the routings of the different product types. It lists the product names ; the identifying numbers for the steps of each ; the station family at which each step is performed ; the processing times and their respective distributions ; the entity for which the processing time applies (e.g., lot, piece), and ; the setup times required prior to the processing, and whether it should always be performed or just in case that the next entity is different than the previous one.

5.7.5 The simulated flow-shop

The flow-shop that was utilized to simulate the different experiments consists of 10 stations in series. Each station consists of a single machine. As indicated earlier, it is assumed that the machines are not subject to failure. Five product types, that have different routes, are processed simultaneously in the flow-shop.

The introduction of sub-lots of the various product types to the system is discussed in more detail in the following Section but, both deterministic and variable arrival times have been considered. In the latter case, arrival times follow an exponential distribution. The sublot size is 25 and it is the equal for all sublots.

The processing times of the various sub-lots on the machines were randomly generated from the range [1,15] minutes per sub-lot. The setup times for the sub-lots, which are always performed prior to the processing of a sublot, i.e., they are sublot-attached, are in the range [1,3] minutes per sublot. Note that since, in this study, we do not vary the sublot size, the sublot-attached setup can be regarded as simply part of the sublot processing time.

The routings of the various product types may include operations on all or part of the machines. In the simulation, this is managed by assigning a zero processing time to a sublot on a machine that is not included on its routing.

5.7.6 The experiments

For each combination of the 50 experiments described in **Table 5.4**, a simulation run of 1 year, or 52 weeks, was performed. The first 10 weeks were considered to be the warm-up period, and were discarded. Weekly statistics were collected for the remainder period of 42 weeks.

The bottleneck utilization, or the workload of the system, was managed through the determination of the frequency of arrival of new work to the system. To increase system workload, the frequency of arrival was changed from 50 minutes, for every sublot type, on the average, to 46 minutes. The two resulting levels of bottleneck utilization, that were considered, are 88% (medium-high) and 95% (high), respectively.

The two locations that were considered for the bottleneck machine are: machine 3, at the front of the flow-shop, and machine 7, at the back of the flow-shop. An identical bottleneck at both locations was also considered. This was done by duplicating the processing and setup times, across all product types, on both machines 3 and 7.

To study the effects of randomness in the system, distributions were assigned to the arrival times of new sublots to the system and to the processing times of the sublots on the machines. Arrival times were assigned an exponential distribution. Processing times were assigned a uniform distribution, with the deterministic processing time as the mean, and a $\pm 5\%$ interval.

5.8 Simulation Results Analysis

In this Section, the simulation results are discussed to ascertain the main effects of the various factors described in Section 5.6. The system performance measures that are utilized for the evaluation are the mean cycle time, the standard deviation of the cycle time, and the standard deviation of the output. These measures were described in detail in Section 5.4. Primary concern is given to the effects of the following factors on system performance measures:

- WIP rules
- Randomness
- System loading
- Bottleneck considerations

The results of the simulation runs, that are utilized for the analysis in this Section, can be found in **Appendix H**.

5.8.1 The impact of WIP rules on system performance

In the sequel, for brevity and clarity, the results are shown for four WIP rules. These rules are: FIFO, X-Theoretical, SPT, and BMI. The other rules that were considered (ESD, LPTR, LBA, and LLA) all perform similar to FIFO. **Figures 5.2 and 5.3** depict the overall average cycle time performance of each of the four WIP rules - FIFO, X-Theoretical, SPT, and BMI - across all the deterministic and stochastic cases, respectively. The X-axis in these figures pertains to the experiments in Table 5.4. Experiments 1, 3, 5, 7, and 9 represent the various deterministic cases whereas experiments 2, 4, 6, 8, and 10 represent the various stochastic cases. The trends in both cases are similar. The results point to a clear ordering of the rules based on their performance. The BMI outperforms the other rules in **all** of the cases. The second best is the SPT rule, followed by the X-Theoretical and FIFO rules. Compared to FIFO, which demonstrates the worst performance amongst the rules, the improvement that the BMI makes is significant - 9.0% and 6.9% for the deterministic and stochastic cases, respectively. Even though, overall, the improvement decreases in the stochastic cases, the improvement margin from the second best, the SPT rule, actually increases from 1.9% in the deterministic cases to 2.4% in the stochastic cases.

The cycle time standard deviation under each of the rules is depicted in **Figures 5.4 and 5.5** for the same set of deterministic and stochastic cases, respectively. Based on these figures, it can be concluded that the FIFO rule is the best to use in order to minimize cycle time variability. The BMI rule appears to not perform as well on this measure, outperforming only the X-Theoretical rule. Compared to FIFO, the cycle time standard deviation of the BMI is higher by 65.3% and 49.1% in the deterministic and stochastic cases, respectively.

For more detail, results of the cycle time by experiment are provided in **Appendix H**. With regard to the output variability, **Figures 5.6 and 5.7** provide the standard deviation of output across the same set of experiments. These figures clearly indicate that, on this measure, the BMI outperforms all the other WIP rules, in all the cases, by a significant margin, especially in the deterministic cases. It reduces the standard deviation of the output by 74.2%, compared to the second best, in the deterministic cases and by 17.1% in the stochastic cases.

To illustrate how such a significant reduction can be attained, **Fig. 5.8** depicts a series of observations of the times between departures, under the various WIP rules, for the deterministic case that is denoted as experiment #3 in Table 5.4. Although this is based on a small subset of the entire set of observations throughout the simulation run, yet it gives insight as to what is actually happening. This case is also representative of the other cases. It can be seen from Fig 5.8 that the range of the times between departures under the BMI rule is much smaller than the ranges resulting from the implementation of the other rules.

Note that the mean times between departures (i.e., the averages of the times between departures), under all the rules, are almost identical, in agreement with Little's law which states that, in steady state, the average output rate (which is the inverse of the mean time between departures) should be the same. Nevertheless, the variability of the output need not be identical, as is evident in our experiments. The reason that the BMI rule has smaller output variability than rules such as FIFO, SPT and X-Theoretical can, at least partially, be attributed to the inflexibility of the latter rules to switch from one part type to another. SPT would always pick the part type with the shortest processing time on the machine, as long as this part is available in the queue. Thus, the ordering of the part types on the machines is **absolute** under SPT (the same argument is true for FIFO and X-Theoretical.) On the other hand, the BMI rule is **relative**. It may result in intermingling of different part types, as demonstrated in Chapter 4.

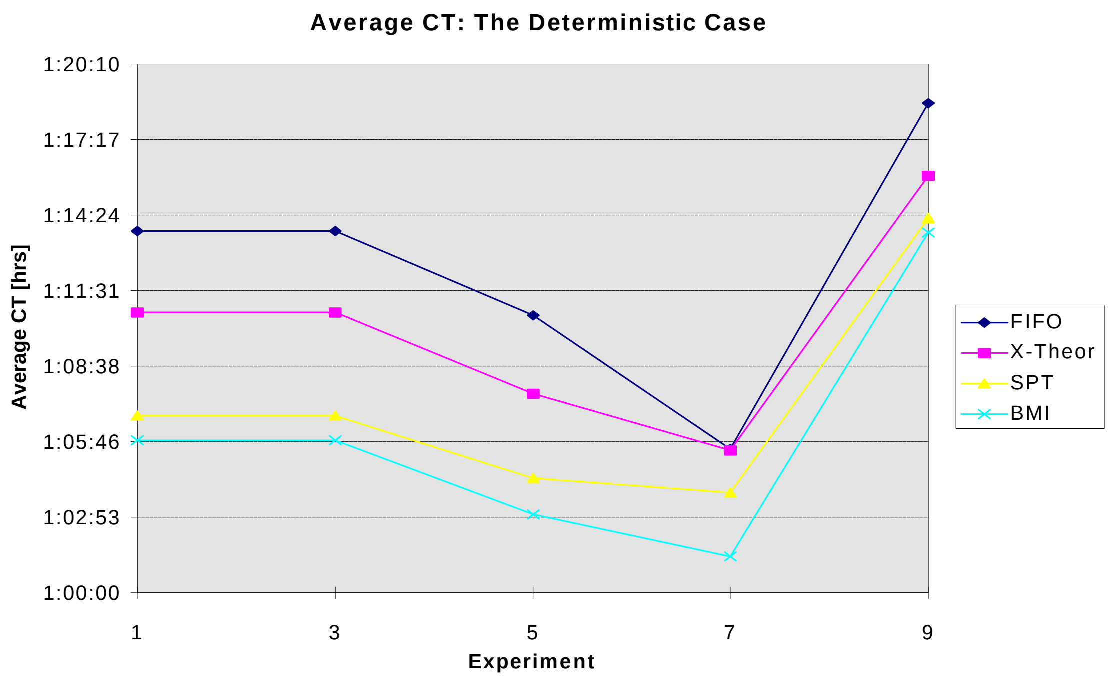


Fig. 5.2. Average cycle time for the deterministic set of experiments

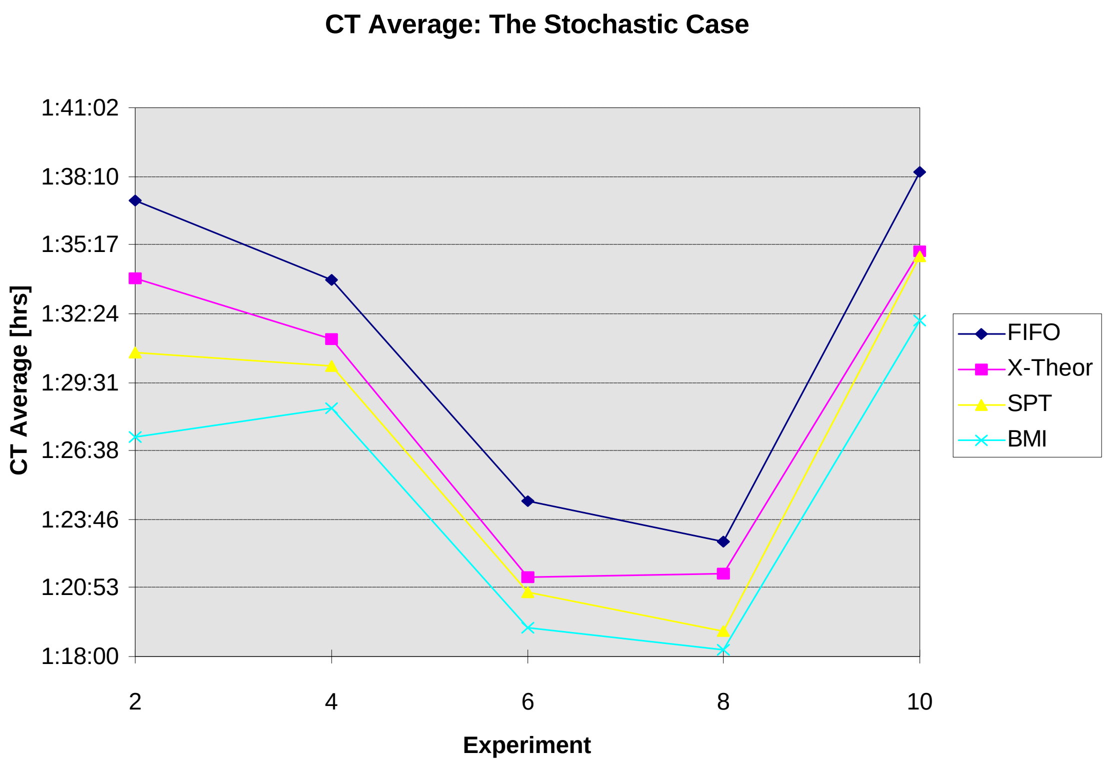


Fig. 5.3. Average cycle time for the stochastic set of experiments

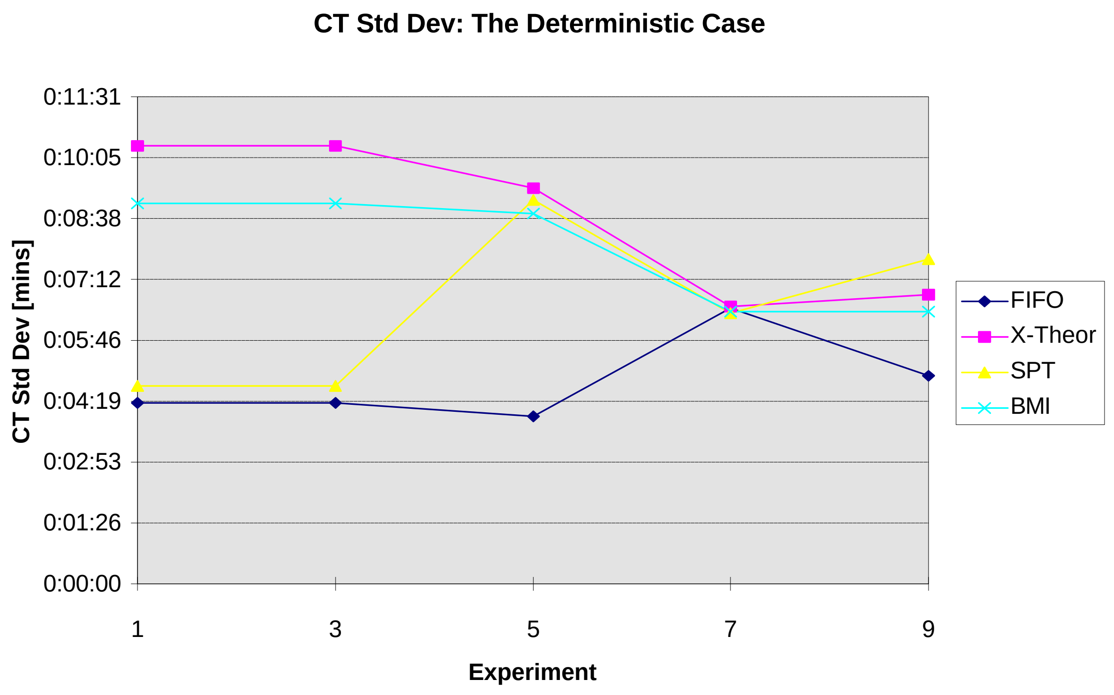


Fig. 5.4. Standard deviation of cycle time for the deterministic set of experiments

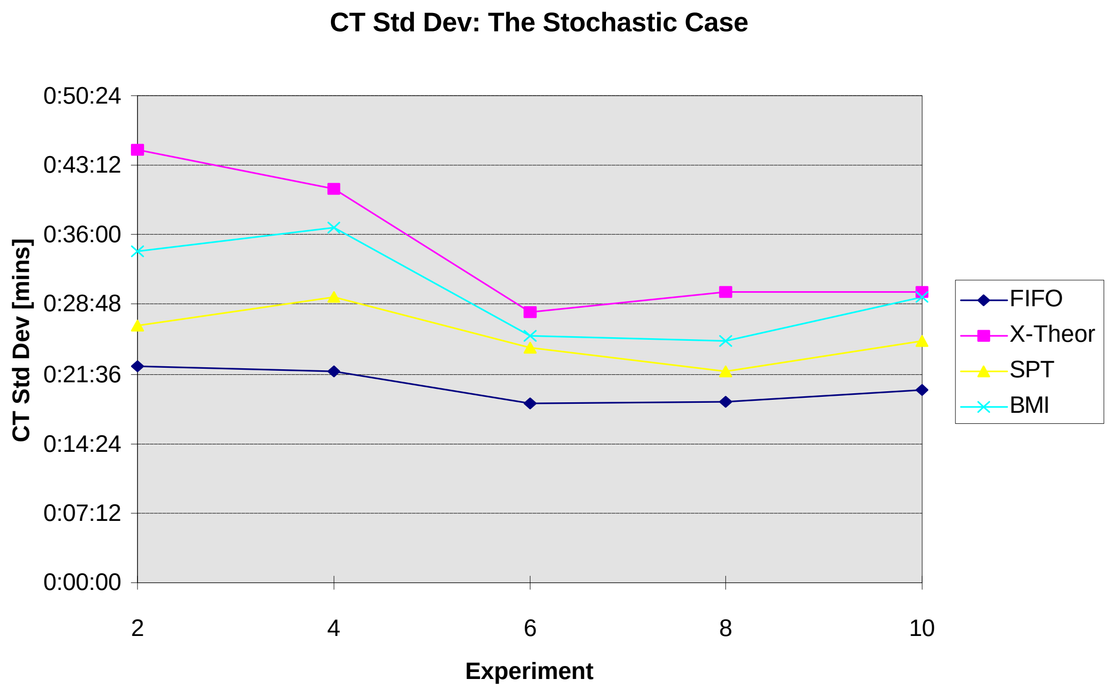


Fig. 5.5. Standard deviation of cycle time for the stochastic set of experiments

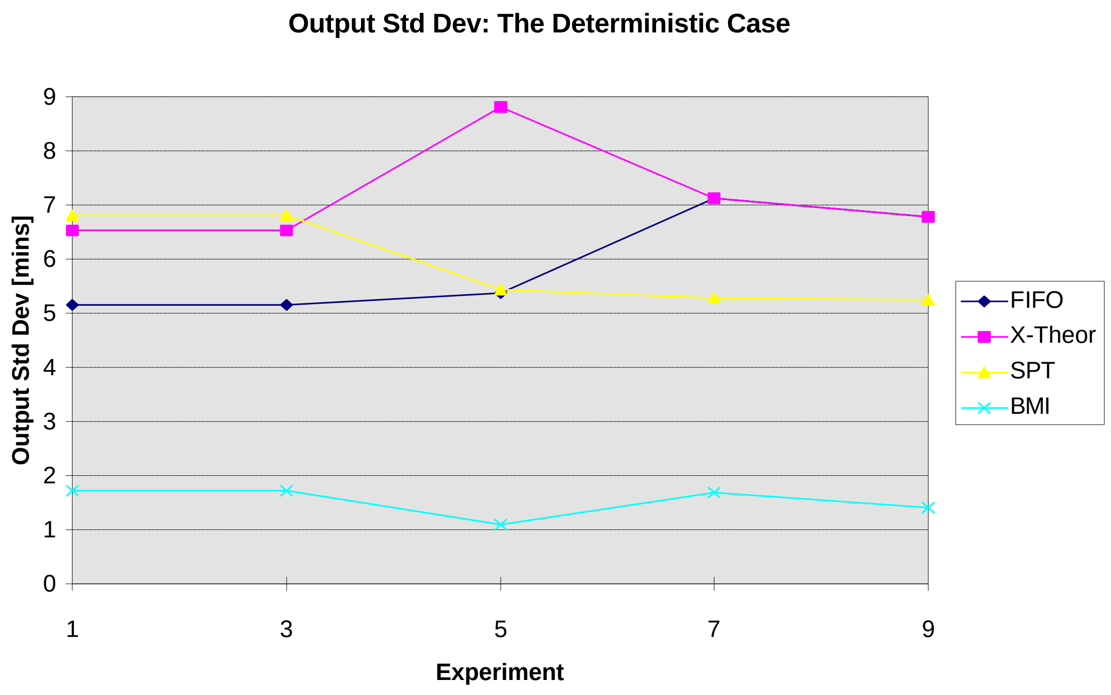


Fig. 5.6. Standard deviation of output for the deterministic set of experiments

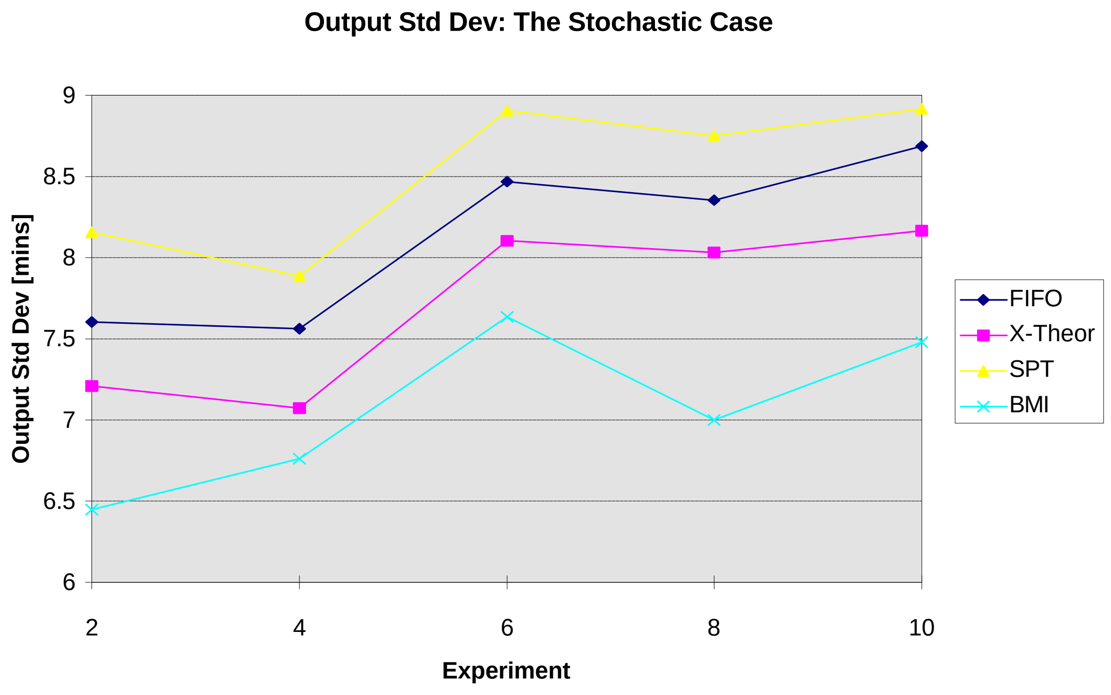


Fig. 5.7. Standard deviation of output for the stochastic set of experiments

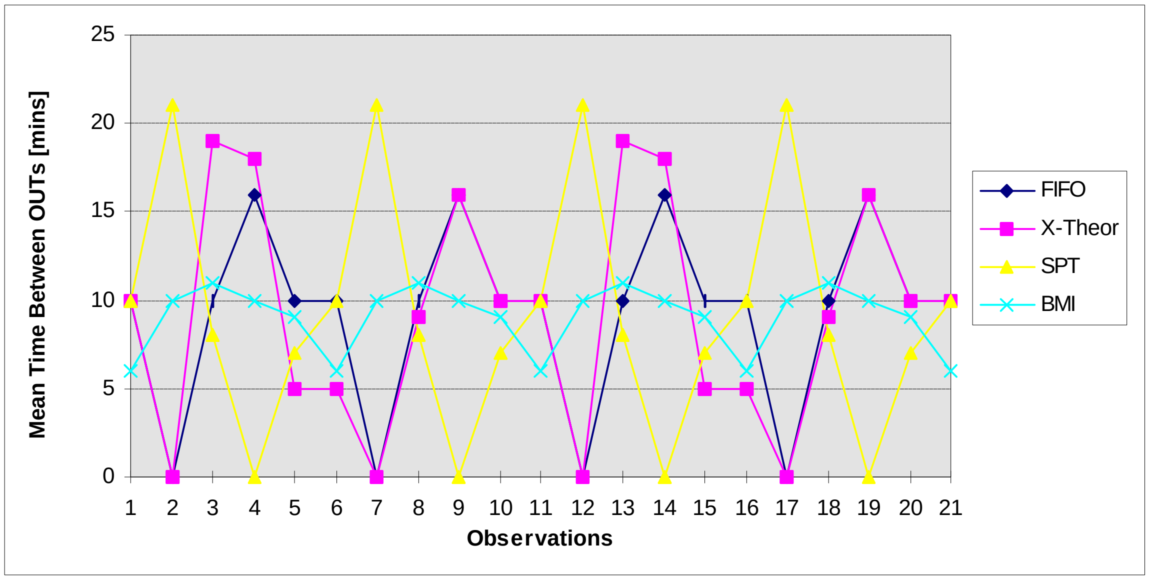


Fig. 5.8. Mean time between departures for the deterministic case (experiment #3)

5.8.2 The impact of randomness on system performance

In this Section, the sensitivity of the WIP rules to the level of system randomness is examined. As mentioned earlier, the odd experiments, that are depicted in Figures 5.2, 5.4, and 5.6, pertain to the deterministic cases whereas the even experiments, that are depicted in Figures 5.3, 5.5, and 5.7, pertain to the stochastic cases. As evident from Figures 5.2 to 5.7, randomness clearly affects the performance of different WIP rules. On the average, the randomness that has been created through the introduction of exponential arrival times of new sublots and uniform processing times has increased the cycle time by 27.7%, the standard deviation of the cycle time by almost 300%, and the standard deviation of the output by 53.2%.

The significant increase in the standard deviation of the cycle time should not come as a surprise. The variability in cycle time is comprised of two components – variability in cycle time due to different part types (between) and variability in cycle time due to individual part types (within). Under the deterministic conditions, there is hardly any variability in the cycle time **within** the different part types. They all experience a similar cycle time once the system reaches steady state. Therefore, it is much lower than it is under stochastic conditions.

As implied in Section 5.8.1, it appears that the higher the amount of randomness in the system, the smaller are the differences in the performance of the different rules. The ranges of improvement between the best and the worst rules on cycle time, standard deviation of cycle time, and standard deviation of output, decrease from 9.0% to 6.9%, 65.3% to 49.1%, and from 74.2% to 17.1%, respectively. Nevertheless, as can be observed, the improvements are still substantial even at high levels of randomness.

5.8.3 The impact of system loading on system performance

In this Section, the sensitivity of the WIP rules to the level of system loading is examined. Experiments 1-4 in Table 5.4 pertain to high system loading (95% bottleneck utilization), attained by increased frequency of arrival of new sublots to the system. Experiments 5-8, on the other hand, pertain to a lower level of system loading, denoted as medium-high (88% bottleneck utilization). **Fig. 5.9** depicts the impact of system loading on the average cycle time performance for the BMI rule. Similar trends were seen with regard to the other WIP rules that were considered. As anticipated, under higher system loading, the average cycle time resulting from the application of the rule is higher. Furthermore, note that, for the latter two data points, which represent the stochastic cases, the increase in the cycle time is more significant than it is for the first two data points, which represent the deterministic cases. This is indeed anticipated as well. For the latter two data points, both system loading and randomness contribute together to the (expected) increase in the average cycle time. Although the BMI rule has been chosen to show the impact of system loading on the average cycle time, it is possible to conclude, based on a comparison of the means, that the other rules are affected in the same manner by this factor (i.e., experience the same order of magnitude of increase in the average cycle time across the different experiments.)

Fig. 5.10 depicts the impact of system loading on the standard deviation of the cycle time for the BMI rule. The trend of the impact is very similar to the one shown for the average cycle time measure. However, with respect to the standard deviation of the cycle time, it appears that the effect, unlike in the case of the average cycle time measure, is rule-sensitive. In other words, the magnitude of the impact is significantly different between the rules. While for the BMI and the X-Theoretical rules, higher system loading leads to increases of 35.6% and 43.6%, respectively, in the standard deviation of the cycle time, it has less of an effect on the FIFO and SPT rules, for which the increases in the standard deviation of the cycle time are only by 10.4% and 6.7%, respectively. Thus, FIFO and SPT are more robust to changes in system loading in terms of cycle time variability.

The impact of system loading on the variability of the output is somewhat surprising. It could be speculated that it should increase, just like the other measures, but, in fact, it slightly decreases. The slight decrease is apparent for all the WIP rules. This implies that the higher the system loading, the more predictable is the output. This, actually, should not come as a surprise since, as system loading increases, the bottleneck machine becomes a more dominant factor in determining the rate at which sublots depart the system.

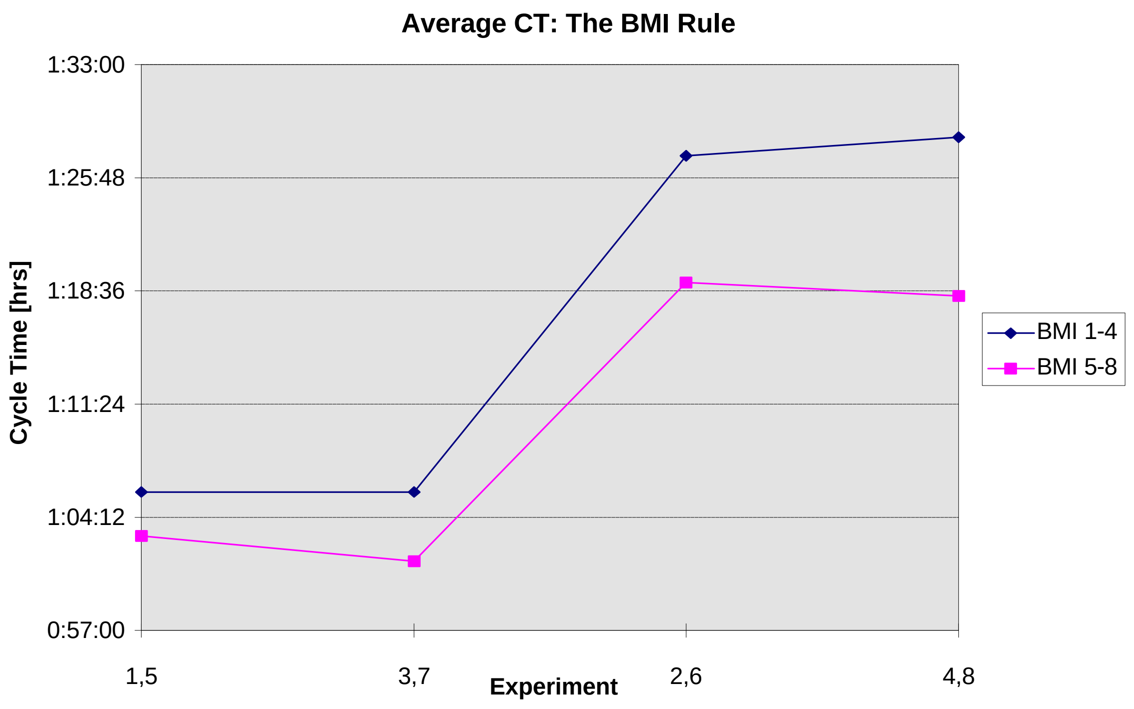


Fig. 5.9. Impact of system loading on the average cycle time

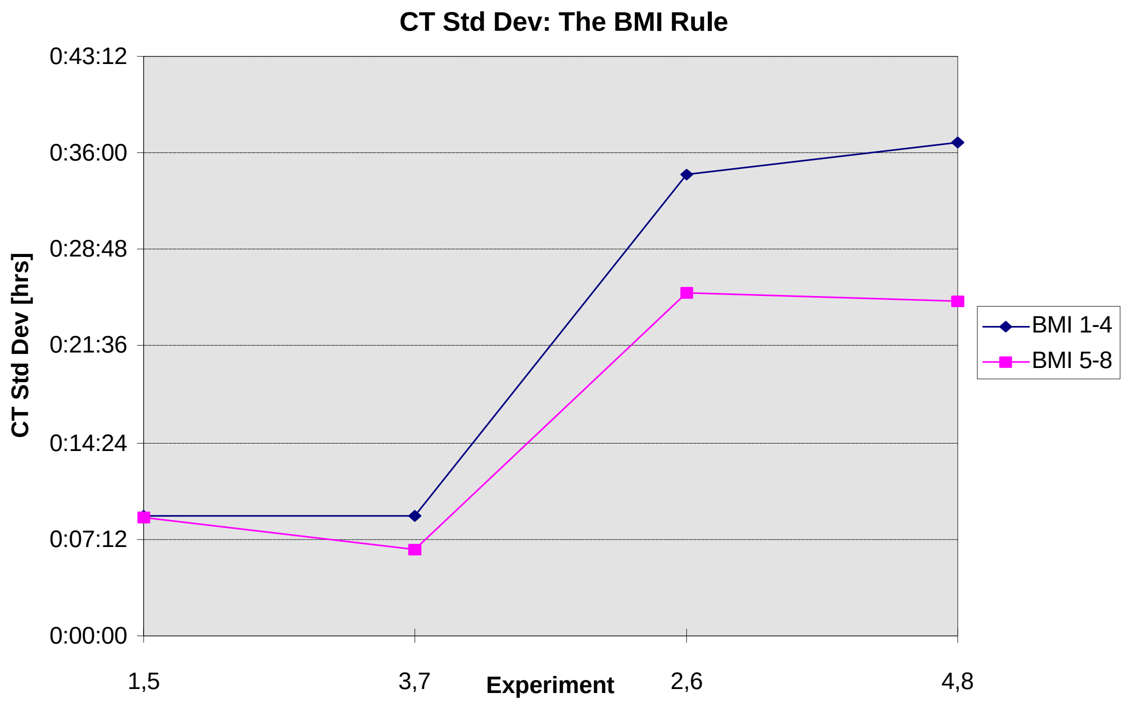


Fig. 5.10. Impact of system loading on the cycle time standard deviation

5.8.4 Bottleneck effects on system performance

In this Section, the effects of bottleneck location(s) on the WIP rules and on the system performance are examined. Experiments 5 (deterministic) and 6 (stochastic) pertain to cases in which the bottleneck location is at the front of the line (the third machine out of ten.) Experiments 7 and 8 pertain to similar cases in which the bottleneck location is at the back of the line (the seventh machine.) Experiments 9 and 10 pertain to similar cases in which the bottleneck is at both locations, i.e., at the front and at the back of the line. The system loading is identical in all these experiments.

Fig. 5.11 depicts the impact of the bottleneck location on the average cycle time performance for the BMI rule. Similar trends were observed with regard to the other WIP rules that were considered. The data points on the left hand side of the chart represent the deterministic cases while the data points on the right hand side of the chart represent the stochastic cases. As anticipated, when more than a single bottleneck exists in the system, the average cycle time increases. In fact, it increases significantly (18.7%). Almost identical increases were recorded for the other WIP rules. In comparing the effect of the location of a single bottleneck on the average cycle time, it appears that, under all the WIP rules, there is no difference between having the bottleneck machine located at the front or at the end of the line.

The behavior of the standard deviation of the cycle time, depicted in **Fig. 5.12**, follows the pattern of the average cycle – increase due to stochasticity and dual bottlenecks. However, the two measures differ in one respect. The standard deviation of the cycle time appears to be identical, in the deterministic cases, regardless of the number of bottlenecks or their location, whereas the average cycle time has increased, even in the deterministic cases, due to the introduction of a second bottleneck.

With regard to the output variability, the standard deviation of the output appears to be insensitive to either the bottleneck location or the number of bottlenecks. Only minor changes of 2% either way have been recorded, across all the WIP rules that were considered, for the standard deviation of the output measure, under different bottleneck locations or for the case of a second bottleneck.

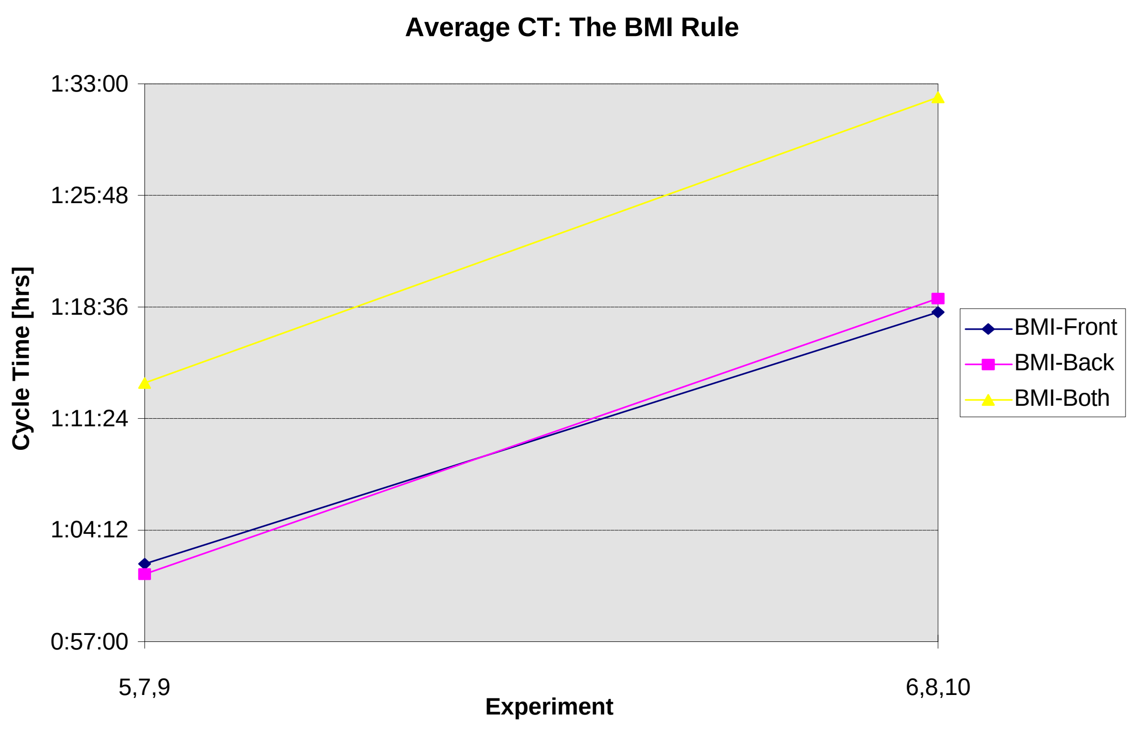


Fig. 5.11. Impact of bottleneck location(s) on the average cycle time

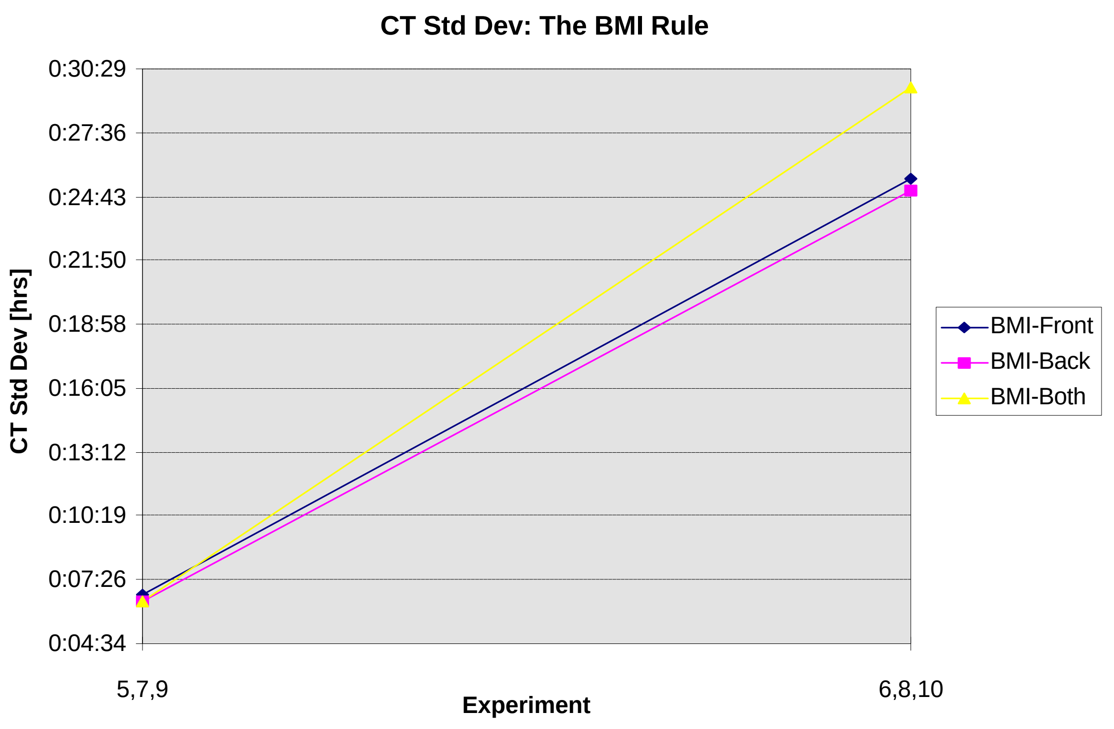


Fig. 5.12. Impact of bottleneck location(s) on the cycle time standard deviation

5.9 Summary of Results

Chapter 5 has primarily provided additional validity to the results obtained in the previous two Chapters, which were limited to static problems, by showing, via discrete event simulation of a flow shop system under lot streaming, that these results can be applied, with minor adjustments, to practical dynamic situations.

A dynamic version of the BMI heuristic has easily been derived from the static BMI heuristic that was developed in Chapter 4. Based on the results of an extensive simulation study, it has been shown that the dynamic BMI rule outperforms existing WIP rules, commonly in use by the industry, in scheduling flow-shops that experience lot streaming. With respect to the primary performance measure – cycle time (or MFT) – the BMI rule has demonstrated a clear improvement over the other WIP rules that were considered in **all** the cases examined. It has been further shown that the dynamic BMI rule also outperforms the other WIP rules with respect to another important measure, the output variability, measured through the standard deviation in the times between sublot departures.

The impact of several other factors, namely system randomness, system loading, and bottleneck-related (location and number), has also been studied. As anticipated, higher levels of system randomness and system loading contribute significantly, under all the WIP rules, to increases in the average cycle time. Similarly, the introduction of a second bottleneck in the system also contributes to increases in the average cycle time. However, the WIP rules and the performance measures appear to not be affected by the location of the bottleneck, i.e. whether it is located at the front or at the end of the line. The only measure that has been sensitive to the WIP rule is the standard deviation in the cycle time measure. The other two measures tend to follow the same pattern of behavior across all the WIP rules.

Chapter 6: Summary and Conclusions

This research work has been motivated by the need to develop efficient practical solutions to various problems that arise in flow-shop environments which utilize lot-streaming. Both JIT and OPT, the revolutionary production control theories of the 80's, have concluded that significant improvements can be attained by breaking down production batches into smaller transfer batches, i.e., by utilizing lot-streaming. Somewhat surprisingly, the critical problems, pertaining to the splitting of the production batches into transfer batches and the sequencing (or streaming) of these transfer batches throughout the production system have remained open. Neither JIT nor OPT have addressed these problems explicitly (Jacobs, 1984). In this research we have successfully addressed these problems for various flow-shop environments.

In this research, we have presented, for the first time, analytical results pertaining to the extent of the potential benefits of lot streaming in flow-shop systems. The benefits have been evaluated with respect to three commonly used performance measures, namely, the makespan, the mean flow time, and the average WIP. For each, an expression of the ratio of the measure under lot streaming to the measure without lot streaming has been provided. These expressions can be used to evaluate the extent of the benefits of lot streaming under certain operating conditions. It has been further shown that, in special extreme cases, these expressions purely depend upon the problem parameters (i.e., the number of machines, the number of lots, the lot-sizes, etc.)

The different lot-streaming problems that arise in various flow-shop environments have been described in the literature review. The research on flow shop lot streaming can roughly be divided into two: single-lot problems and multiple-lot problems. Single-lot problems are well classified and the literature offers a relatively extensive body of knowledge on these problems. In contrast, very little attention has been given to the more complicated but more practical collection of multiple-lot problems. A classification of multiple-lot problems has been presented in this research. At the highest level, these problems are differentiated by whether the sublot-size is pre-determined or not. When it is pre-determined, the lot streaming sequencing problem (LSSP) results, in which the sequence of the sublots needs to be determined. On the other hand, if the sublot-size is not pre-determined and it, therefore, must be determined in parallel with the determination of the sequence, the flow-shop lot-streaming (FSLS) problem results. Further classification of each of these problems is made in this work, based upon whether intermingling of sublots is considered and whether the setup is lot-attached or sublot-attached. In the FSLS problem even further classification is made, depending upon whether the sublot-sizes are lot-specific or not. In this research these multiple-lot problems, emerging from the above classification, have been addressed. In agreement with numerous practical applications consideration has been restricted to sublots of equal sizes. Justifications for this assumption were provided in the literature review. To approach the multiple-lot problems more effectively the single-lot problem has been addressed first, in Chapter 3.

Although the literature on single-lot problems is extensive, solution schemes that have been proposed thus far suffer from one or more of the following limitations: (a) the number of sublots must somehow be determined independently (b) only the makespan criterion is utilized to measure the optimality of the solution (c) it is assumed that the transfer time does not affect the makespan (d) the setup is assumed negligible or, at most, lot-attached but not sublot-attached ; and (e) the discrete solution (which is the practical solution as far as implementation is concerned) is difficult to obtain. In this research, the above limitations have been relaxed. First, the number of sublots has been considered to be a decision. Second, the objective function to be minimized has incorporated not only the makespan but also the mean flow time, the average work-in-process (WIP), and the setup and handling costs. Third, the effects of transfer time and sublot-attached setup on the makespan, and on other criteria, have been considered and, fourth, discrete solutions can be obtained in polynomial time. A goal programming (GP) approach has been utilized to prescribe a unified expression for the objective function in the model. An algorithm has been proposed for finding the optimal number of sublots (and, consequently, the sublot size) for this unified objective function and for an *m*-machine flow shop. The complexity of the proposed algorithm is *O(m)* and it is, therefore, a very fast algorithm. However, it involves solving an equation of the fourth order. Although this can easily be done, a quick approximation equation has also been developed, that can be used to avoid solving the fourth order equation. Experiments with the approximation have revealed that the algorithm with the approximation finds the optimal solution in many situations. When the algorithm does not find the optimal solution the difference in the objective function values is minor (i.e., the solution is near-optimal.)

In Chapter 4 an attempt has been made to extend the results obtained in Chapter 3, for a single lot, to the case of multiple lots. The multiple-lot problem, however, is by far more complicated than the single-lot problem since it requires determining a sequence of the sublots (or the lots) in addition to the determination of the sublot-size. Thus, a (large scale) flow-shop sequencing problem, which is known to be NP-hard, is embedded. The focus has, thus, shifted to the sequencing problem, referred to as the LSSP. A near-optimal heuristic for the solution of the LSSP in a multi-batch multi-machine flow-shop has been developed. This proposed heuristic procedure, referred to as the Bottleneck Minimal Idleness (BMI) heuristic, identifies and employs certain properties of the problem that are irregular in traditional flow-shop problems. The BMI heuristic attempts to maximize the time buffer prior to the bottleneck machine, thereby minimizing potential bottleneck idleness, while also looking-ahead to sequence lots with large remaining process time earlier in the schedule. It has been further shown in this research, through an experimental study, that the BMI heuristic outperforms a modification of the Fast Insertion Heuristic for Lot Streaming (FIHLS), which is based on the best known heuristic for flow-shop scheduling so far, when applied to the problem on hand.

For the FSLS problem, several algorithms have been developed. For the two-machine FSLS problem with an identical sublot-size for all lots, an optimal pseudo-polynomial solution algorithm, of complexity $O(N \cdot \log(N) \cdot Q_{\max})$, has been proposed. This algorithm is expected to be fairly fast for reasonable lot-sizes. For the case in which the sublot-sizes are lot-based, optimal and heuristic procedures have been developed. The heuristic procedure has been developed to, primarily, reduce the complexity of the optimal solution algorithm. It indeed reduces the complexity to $O(N^2 \cdot \log(N) \cdot Q_{\max})$. Although still pseudo-polynomial, it is not exponential in Q, as is the optimal solution algorithm. The heuristic procedure includes a construction phase and an improvement phase. In the construction phase, it first attempts to find a near-optimal sequence for the lots and then, at the improvement phase, given the sequence, it attempts to optimize the lot-based sublot-sizes of each of the lots. Extensions of the solution algorithms proposed for the two-machine FSLS problem have been presented to deal with the general *m*-machine FSLS problem. In these extensions, the only modification is in the mechanism that is utilized to generate the sequences. Johnson's rule, which yields optimal sequences in two-machine flow-shops, has been replaced with the BMI heuristic, which has been shown to result in near-optimal sequences for general *m*-machine flow-shops under lot-streaming.

Chapter 5 in this research has primarily provided additional validity to the results obtained in Chapters 3 and 4, which are limited to static problems, by showing, via discrete event simulation of a flow shop system under lot streaming, that the results obtained therein can also be implemented, with minor adjustments, in practical dynamic situations. In Chapter 5, a dynamic version of the BMI heuristic has easily been derived from the static BMI heuristic that was developed in Chapter 4. Based on the results of an extensive simulation study, it has been shown that the dynamic BMI rule outperforms existing WIP rules, commonly in use by the industry, in scheduling a flow-shop that experiences lot streaming. With respect to the primary performance measure – cycle time (or MFT) – the BMI rule has demonstrated a clear improvement over the other WIP rules that were considered in **all** the cases examined. It has been further shown that the dynamic BMI rule also outperforms the other WIP rules with respect to another important measure, the output variability, measured through the standard deviation in the times between sublot departures.

The impact of several other factors, namely system randomness, system loading, and bottleneck-related (location and number), has also been studied. As anticipated, higher levels of system randomness and system loading contribute significantly, under all the WIP rules, to increases in the average cycle time. Similarly, the introduction of a second bottleneck in the system also contributes to increases in the average cycle time. However, the WIP rules and the performance measures appear to not be affected by the location of the bottleneck, i.e. whether it is located at the front or at the end of the line. The only measure that has been sensitive to the

WIP rule is the standard deviation in the cycle time measure. The other two measures, cycle time and output variability, tend to follow the same pattern of behavior across all WIP rules.

Bibliography

Anthony, R.N., 1965, Planning and Control Systems: A Framework for Analysis, Harvard University Press, Cambridge, MA

AutoSched AP software, User Manuals I-IV. Auto-Simulations, Bountiful, Utah.

Baker, K.R., 1974, Introduction to Sequencing and Scheduling. John Wiley and Sons, NY.

Baker, K.R., 1995, Lot Streaming in the Two-Machine Flow Shop with Setup Times, Annals Oper. Res., V-57, pp. 1-11

Baker, K.R., 1988, Lot Streaming to Reduce Cycle Time in a Flow Shop. Working Paper #203, The Amos Tuck School of Business Administration, Dartmouth College, Hanover NH

Baker, K.R., and Pyke, D.F., 1990, Solution Procedures for the Lot-Streaming Problem. Dec. Sci., V-21, pp. 475-491.

Baker, K.R., and Jia, D., 1993, A Comparative Study of Lot Streaming Procedures, OMEGA Int. J. Mgmt. Sci., V-21, N-5, pp. 561-566

Benjaafar, S., 1996, On Production Batches, Transfer Batches, and Lead Times. IIE Trans., V-28, pp. 357-362

Bazaraa, M.S., Sherali, H.D., and Shetty, C.M., 1993, Nonlinear Programming Theory and Algorithms, 2nd. Ed., Wiley.

Campbell, H.G., Dudek, R.A., and Smith, M.L, 1970, A Heuristic Algorithm for the n-job, m-machine Sequencing Problem, Mgmt. Sci., V-16B, pp. 630-637

Cetinkaya, F.C., 1994, Lot Streaming in a Two-Stage Flow Shop with Set-up, Processing and Removal Times Separated, J. Oper. Res. Soc., V-45, N-12, pp. 1445-1455

Chan, D.Y., and Bedworth, D.D., 1991, Design of Scheduling Systems for FMS, Int. J. Prod. Res., 27, 8, 1257-1267.

Chen, J. and Steiner, G., 1997, Lot Streaming With Detached Setups in Three-Machine Flow Shops, Europ. J. Oper. Res., V-96, N-3, pp. 591-611.

Cheng, T.C.E., and Podolsky, S., 1996 (2 ed.), Just-In-Time Manufacturing: An Introduction, Chapman & Hall, London, UK.

Conway, R.W., Maxwell, W.L., and Miller, L.W., 1967, Theory of Scheduling, Addison-Wesley, MA, USA

Dannenbring, D.G., 1977, An Evaluation of Flow Shop Sequencing Heuristics, Proc. 3rd Annual ACM Symp. Theory of Computing, pp. 1174-1182

Dauzere, P.S., and Lasserre, J.B., 1993, An Iterative Procedure for Lot Streaming in Job-Shop Scheduling. Computers & Ind. Eng., V-25, pp. 231-4

Dauzere, P.S., and Lasserre, J.B., 1997, Lot Streaming in Job-Shop Scheduling. Oper. Res., V-45, N-4, 584-595

Demeulemeester, E.L., and Herroelen, W.S., Modelling Setup Times, Process Batches and Transfer Batches Using Activity Network Logic. European J. Oper. Res., V-89, N-2, pp. 355-365

Dobson, G., Yano, C.A., 1994, Cyclic Scheduling to Minimize Inventory in a Batch Flow Line. European J. Oper. Res., V-75, N-2, pp. 441-461.

Doutriaux, F.W., Sarin, S.C., and Kalir, A.A., 1996, Optimal Sublot Determination in a Multiple Batch, Two Machine Production System, IIE Trans, In press.

El-Najdawi, M.K., 1994, A Job-splitting Heuristic for Lot-size Scheduling in Multi-stage, Multi-product Production Processes, Euro. J. Oper. Res., V-75, pp. 365-377

El-Najdawi, M.K., and Kleindorfer, P.R., 1993, Common Cycle Lot-size Scheduling for Multi-stage Multi-product Production, Mgmt. Sci., V-39, N-7, pp. 872-885

Eynan, A., and Li-Chung, L., 1997, Lot-Splitting Decisions and Learning Effects, IIE Trans., V-29, N-2, pp. 139-146

Fisher, D., 1995, The Just-In-Time Self Test: Success Through Assessment and Implementation, Irwin Publishing, Boston, USA.

Fox, R.E., 1983, “OPT - An Answer for America (Part IV)”, Inventories and Production Magazine, V-3, N-2.

Fukukawa, T., and Hong, S.C., 1993, The Determination of the Optimal Number of Kanbans in a Just-In-Time Production System. Comput. & Ind. Eng., V-24, N-4, pp. 551-559

Glass, C.A., Gupta, J.N.D., and Potts, C.N., 1994, Lot Streaming in Three-stage Production Processes, Euro. J. Oper. Res., V-75, pp. 378-394

Ghosh, S., and Gagnon, R.J., 1989, A Comprehensive Literature Review and Analysis of the Design, Balancing and Scheduling of Assembly Systems, Int. J. Prod. Res., V-27, pp. 637-670

Goldratt, E., 1990, What is this Thing Called the THOERY OF CONSTRAINTS and How Should It Be Implemented? , North River Press, Crotonon-Hudson, NY

Goldratt, E. and Cox, J., 1984, The Goal: A Process of Ongoing Improvement, North River Press, Crotonon-Hudson, NY

Goldratt, E. and Fox, R.E., 1986, The Race. North River Press, Crotonon-Hudson, NY

Goyal, S.K., 1976, Note on Manufacturing Cycle Time Determination for a Multi-Stage Economic Production Quantity Model, Mgmt. Sci., V-23, pp. 332-333

Greco, M.P, 1996, Sequencing Policy for a CONWIP Production System, M.Sc. Thesis, Virginia Polytechnic Institute and State University, VA, USA

Gupta, G.N.D., 1971, A Functional Heuristic Algorithm for the Flow-shop Scheduling Problem, Oper. Res. Qtrly., V-22, N-1, pp. 39-47

Gupta, Y.P., Gupta, M.C., and Bector, C.R., 1989, A Review of Scheduling Rules in FMS, Int. J. CIM, V-1, N-6, pp. 356-377.

Hall, E.H., 1989, Just-In-Time Management: A Critical Assessment, Acad. Mgmt. Exec., V-3, N-4, pp. 315-318.

Hancock, T.M., 1991, Effects of Lot-Splitting Under Various Routing Strategies. Int. J. Oper. & Prod. Mgmt., V-11, N-1, pp. 68-74

Hax, A.C., and Candea, D., 1984, Production and Inventory Management, Prentice Hall Inc., Ch.1 The Nature of Production Inventory Systems, pp. 1-7

Herer, Y.T., and Masin, M., 1997, Mathematical Programming Formulation of CONWIP Based Production Lines and Relationships to MRP, Int. J. Prod. Res., V-34, N-4, pp. 1067-1076

Hernandez, A., 1989, Just-In-Time Manufacturing: A Practical Approach, Prentice Hall Inc., NJ, USA.

Hey, E.G., 1988, The Just-In-Time Breakthrough, John Wiley and Sons.

Hopp, W.J., and Spearman, M.L., 1991, Throughput of a CONWIP Manufacturing Line Subject to Failures, Int. J. Prod. Res., V-29, N-3, pp. 635-655

Huang, P.Y., Rees, L.P., and Taylor, III, 1983, A Simulation Analysis of the Japanese Just-in-time Technique With Kanbans for a Multiline, Nultistage Production System. Dec. Sci., V-14, pp. 326-344

Jacobs, F.R., 1984, OPT Uncovered: Many Production Planning and Scheduling Concepts Can be Applied With or Without the Software, Industrial Eng., V-16, pp. 330-334

Jacobs, F.R., and Bragg, D.J., 1988, Repetitive Lots: Flow-Time Reductions Through Sequencing and Dynamic Batch Sizing. Dec. Sci., V-19, pp. 281-294

Johnson, S.M., 1954, Optimal Two- and Three-Stage Production Schedules With Setup Time Included. Naval Res. Log. Qtrly., V-1, pp. 61-68

Kalir, A.A, and Sarin S.C., 1998, A method for reducing the inter-departure time variability in serial production lines. Dec. Sci., In press.

Klein, J.A., 1989, The Human Costs of Manufacturing Reform, Harvard Bus. Rev., V-67, N-2, pp. 62-66.

Kropp, D.H., Smunt, T.L., and Buss, A.H., 1988, Lot-Splitting Performance Under Varying Flow Dominance Conditions. presented at ORSA/TIMS, denver, CO, USA.

Kropp, D.H., and Smunt, T.L., 1990, Optimal and Heuristic Models for Lot Splitting in a Flow Shop. Dec. Sci., V-21, N-4, pp. 691-709

Logendran, R., and Ramakrishna, P., 1995, Manufacturing Cell Formation in the Presence of Lot Splitting and Multiple Units of the Same Machine. Int. J. Prod. Res., V-33, N-3, pp. 675-693

Lawler, E.G., Lenstra, J.K., Rinnooy Kan, A.H.G., and Shmoys, D.B., 1993, Sequencing and Scheduling: Algorithms and Complexity, in: S.C. Graves, Rinnooy Kan, A.H.G., and Zipkin, P.H. (eds.), Handbooks in OR & MS, North-Holland, Amsterdam, V-4, pp. 445-522

Mather, H., 1986, MRP-II Won't Schedule the Factory of the Future. Annual International Conference Proceedings - American Production and Inventory Control Society 29th. Publ by APICS, Falls Church, VA, USA pp. 495-497.

McCutchen, T., and Lee, C.Y., 1995, An Analysis of Dispatching Rules in a Semi-Conductor Wafer Fabrication Environment. J. Electronics Mfg., V-5, N-3, pp. 165-174.

Moily, J.P., 1986, Optimal and Heuristic Procedures for Component Lot-Splitting in Multi-Stage Manufacturing Systems, Mgmt. Sci., V-32, N-1, 113-125.

Monden, Y., 1983, <u>Toyota Production System</u>, IE and Mgmt. Press, GA, USA.

Montazeri, M., and Van-Wassenhove, L.N., 1990, Analysis of Scheduling Rules for an FMS. Int. J. FMS, 3, 121-147

Nawaz, M., Enscore Jr, E.E., and Ham, I., 1983, A Heuristic Algorithm for the n-job, m-machine Flow Shop Sequencing Problem, Omega, V-11, pp. 91-95

Osman, I.H., and Potts, C.N, 1989, Simulated Annealing for Permutation Flow Shop Scheduling, Omega, V-17, 551-557

Page, E.S., 1961, An Approach to Scheduling of Jobs on the Machines, J. Royal Stat. Soc., V-23, pp. 484-492

Palmer, D.S., 1965, Sequencing Jobs through a Multi-Stage Process in the Minimum Total Time - A Quick Method, Oper. Res. Qtrly., pp. 101-107

Park, Y.B., 1981, A Simulation Study and an Analysis for Evaluation of Performance – Effectiveness of Flow Shop Sequencing Heuristics: A Static and a Dynamic Flow Shop Model. M.Sc. Thesis, Pennsylvania State University, PA, USA

Pinedo, M., 1995, Scheduling Theory: Algorithms and Systems, Perntice Hall Inc.

Pinto, P.A., and Rao, B.M., 1992, Joint Lot-Sizing and Scheduling for Multi-Stage Multi-Product Flow Shops. Int. J. Prod. Res., V-30, N-5, pp. 1137-1152

Potts, C.N., Baker, K.R., 1989, Flow Shop Scheduling With Lot Streaming. Oper. Res. Lett., V-8, N-6, pp. 297-303

Pyreddy, V.R., 1996, Solution Procedures for the Multiple Product Lot Streaming Problem in Multiple Machine Flow Shops, Ph.D. thesis, Drexler University, PA, USA.

Rees, L.P., Philipoom, P.R., Taylor, III, B.W., and Huang, P.Y., 1987, Dynamically Adjusting the Number of Kanbans in a Just-In-Time Production System Using Estimated Values of Leadtime, IIE Trans., V-19, pp. 199-207

Reiter, S., 1966, A System for Managing Job-Shop Production, J. Business, V-39, pp. 371-393

Roderick, L.M., Toland, J., and Rodriguez, F.P., 1994, Simulation Study of CONWIP versus MRP at Westinghouse, Comput. & Ind. Eng., V-26, N-2, pp. 237-242

Rosenblatt M.J., and Carlson R.C., 1985, Designing a Production Line to Maximize Profit, IIE Trans., pp. 117-122

Russell, G.R., and Fry, T.D., 1997, Order Review/Release and Lot Splitting in Drum-Buffer-Rope, Int. J. Prod. Res., V-35, N-3, pp. 827-845

Santos, C., and Magazine, M., 1985, Batching in Single Operation Manufacturing Systems. Oper. Res. Letters, V-4, N-3, pp. 99-103

Santos, D.L., Hunsucker, J.L., and Deal, D.E., 1995, Global lower bounds for flow shops with multiple processors. Europ. J. Oper. Res., 80, 1, 112-120.

Sarin, S.C., and Greco, M., 1997, Job Sequencing and Material Flow Control in a Closed Production System, IIE Trans., In press.

Smunt, T.L., Buss, A.H., and Kropp, D.H., 1996, Lot Splitting in Stochastic Flow Shop and Job Shop Environments. Dec. Sci., V-27, N-2, pp. 215-238

Sohlenius, G., Arnstrom, A. and Grondahl., P., 1989, Solution to Flexible, Productive Assembly as Part of the Total Manufacturing System, Publ-by-ASME, p 223-230, Proceedings of Mfg. Int. '88, V-1, Atlanta, GA, USA

Spearman M.L., and Zazanis, M.A., 1988, Push and Pull Production Systems: Issues and Comparisons, Technical Report 88-24, Dept. of IE&MS, Northwestern University, IL,USA

Spearman M.L., Woodruff, D.L., and Hopp, W.J., 1990, CONWIP: A Pull Alternative to Kanban, Int. J. Prod. Res., V-25, N-5, pp. 879-894

Sriskandarajah, C., and Wagneur, E., 1995, Lot Streaming of a Single Product in Two-Machine No-Wait Flowshops, IEEE Symp. Emerging Technologies & Factory Automation, V-3, NJ, USA. pp. 427-436

Suzaki, K., 1987, The New Manufacturing Challenge, The Free Press, NY.

Szendrovits, A.Z., 1975, Manufacturing Cycle Time Determination for a Multistage Economic Production Quantity Model, Mgmt. Sci., V-22, N-3, pp. 298-308

Thompson, M., 1996, Simulation-Based Scheduling: Meeting the Semiconductor Wafer Fabrication Challenge, IIE Solutions, V-28, N-5, pp. 30-34.

Trietsch, D., and Baker, K.R., 1993, Basic Techniques for Lot Streaming, Oper. Res., V-41, N-6, pp. 1065-1076

Trietsch, D., 1987, Optimal Transfer Lots for Batch Manufacturing: a Basic Case and Extensions, Technical Report NPS-54-89-010, Naval Postgraduate School, Monterey CA

Trietsch, D., 1989, Polynomial Transfer Lot Sizing Techniques or Batch Processing on Consecutive Machines, Technical Report NPS-54-89-011 Naval Postgraduate School, Monterey CA

Truscott, W., 1986, Production Scheduling with Capacity Constrained Transportation Activities, J. Oper. Mgmt., V-6, pp. 333-348

Turner, S., and Booth, D., 1987, Comparison of Heuristics for Flow Shop Sequencing, Omega, V-15, pp.75-78

Vembu, S., and Srinivasan, G., 1995, Heuristics for Operator Allocation and Sequencing in Just-In-Time Flow Line Manufacturing Cell. Computers and Ind. Eng., V-29, N-1-4, pp. 309-313

Vickson, R.G., and Alfredson, B.E., 1992, Two and Three Machine Flow Shop Scheduling Problems With Equal Sized Transfer Batches, Int. J. Prod. Res., V-30, pp. 1551-1574

Vickson, R.G., 1995, Optimal Lot Streaming for Multiple Products in a Two-Machine Flow Shop. Europ. J. Oper. Res., V-85, pp. 556-575

Vollmann, T.E., 1986, OPT as an Enhancement to MRP-II, Prod. and Inv. Mgmt., Q-2, pp. 38-46

Wagner, B.J. and Ragats, G.L., 1994, The Impact of Lot Splitting on Due Date Performance, J. Oper Mgmt., V-12, pp.13-25

Wu, C.T., Egbelu, P.J., 1994, Control of Material Flow in Serial Production Systems. Proceedings of the 1994 Int. Mechanical Eng. Congress, pp. 1-8, Chicago, IL, USA

Appendices

Appendix A: Terms for the strict convexity of $WIP_j(n)$

From exp. (3.7-2) we have:

$$WIP_j(n) = Q \cdot \frac{\left[\frac{Q}{n} \cdot \sum_{j=1}^{m} p_j + \sum_{j=1}^{m} su_j\right] + \frac{n-1}{2} \cdot \left\{\frac{Q}{n} \cdot p_j + su_j\right\}}{\left[\frac{Q}{n} \cdot \sum_{j=1}^{m} p_j + \sum_{j=1}^{m} su_j\right] + (n-1) \cdot \left\{\frac{Q}{n} \cdot p_j + su_j\right\}}$$

Multiplying the numerator and the denominator by *n*, and re-organizing, we get:

$$\frac{1}{Q} \cdot WIP_j(n) = \frac{n^2 \cdot \frac{su_j}{2} + n \cdot \left(\sum_{j=1}^{m} su_j + \frac{Q \cdot p_j}{2} - \frac{su_j}{2}\right) + \left(Q \cdot \sum_{j=1}^{m} p_j - \frac{Q \cdot p_j}{2}\right)}{n^2 \cdot su_j + n \cdot \left(\sum_{j=1}^{m} su_j + Q \cdot p_j - su_j\right) + \left(Q \cdot \sum_{j=1}^{m} p_j - Q \cdot p_j\right)}$$

For simplicity, define:

$$k_1 = \frac{su_j}{2}, k_2 = \sum_{j=1}^{m} su_j, k_3 = \frac{Q \cdot p_j}{2}, k_4 = Q \cdot \sum_{j=1}^{m} p_j$$

Note that $k_1, k_2 \geq 0, \quad k_3, k_4 > 0$. Thus, we have:

$$\frac{1}{Q} \cdot WIP_j(n) = \frac{n^2 \cdot k_1 + n \cdot (k_2 + k_3 - k_1) + (k_4 - k_3)}{n^2 \cdot 2k_1 + n \cdot (k_2 + 2k_3 - 2k_1) + (k_4 - 2k_3)}$$

Taking the first derivative, we get:

$$\frac{1}{Q}\cdot\frac{dWIP_j(n)}{dn}=\frac{\left[\left(n\cdot 2k_1+\left(k_2+k_3-k_1\right)\right)\cdot\left(n^2\cdot 2k_1+n\cdot\left(k_2+2k_3-2k_1\right)+\left(k_4-2k_3\right)\right)\right]}{\left[n^2\cdot 2k_1+n\cdot\left(k_2+2k_3-2k_1\right)+\left(k_4-2k_3\right)\right]^2}-$$
$$-\frac{\left[\left(n\cdot 4k_1+\left(k_2+2k_3-2k_1\right)\right)\cdot\left(n^2\cdot k_1+n\cdot\left(k_2+k_3-k_1\right)+\left(k_4-k_3\right)\right)\right]}{\left[n^2\cdot 2k_1+n\cdot\left(k_2+2k_3-2k_1\right)+\left(k_4-2k_3\right)\right]^2}$$

For simplicity, let us define the following terminology: $k_{ij}=k_i\cdot k_j$.

After simplifying the numerator and utilizing the above notation, we are left with:

$$\frac{1}{Q}\cdot\frac{dWIP_j(n)}{dn}=-\frac{n^2\cdot k_{12}+n\cdot\left(2k_{14}\right)+\left(k_{23}+k_{34}-k_{14}\right)}{\left[n^2\cdot 2k_1+n\cdot\left(k_2+2k_3-2k_1\right)+\left(k_4-2k_3\right)\right]^2}$$

Note that as the number of sublots, *n*, increases - it must be that that the *WIP* is strictly decreasing because portions of the lot are completed earlier than what they would have otherwise. Therefore, the first derivative must satisfy the negative slope condition:

$$\begin{aligned}&\frac{1}{Q}\cdot\frac{dWIP_j(n)}{dn}=-\frac{n^2\cdot k_{12}+n\cdot\left(2k_{14}\right)+\left(k_{23}+k_{34}-k_{14}\right)}{\left[n^2\cdot 2k_1+n\cdot\left(k_2+2k_3-2k_1\right)+\left(k_4-2k_3\right)\right]^2}<0\\&\Leftrightarrow\\&n^2\cdot k_{12}+n\cdot\left(2k_{14}\right)+\left(k_{23}+k_{34}-k_{14}\right)>0\\&\Rightarrow\\&\left(k_{23}+k_{34}-k_{14}\right)>0\end{aligned}$$

This inequality will be useful next, when we take the second derivative in an attempt to show that it is always positive. To further simplify, define:

$$C_1=k_{23}+k_{34}-k_{14},C_2=k_2+2k_3-2k_1$$

Note that $C_1,C_2>0$.

$$\frac{1}{Q}\cdot\frac{dWIP_j^2(n)}{dn^2}=\frac{2\cdot\left[n^2\cdot 2k_1+n\cdot C_2+(k_4-2k_3)\right]\cdot\left(n\cdot(4k_1)+C_2\right)\cdot\left(n^2\cdot k_{12}+n\cdot(2k_{14})+C_1\right)}{\left[n^2\cdot 2k_1+n\cdot C_2+(k_4-2k_3)\right]^4}-$$
$$-\frac{\left(n\cdot(2k_{12})+2k_{14}\right)\cdot\left[n^2\cdot(2k_1)+n\cdot C_2+(k_4-2k_3)\right]^2}{\left[n^2\cdot 2k_1+n\cdot C_2+(k_4-2k_3)\right]^4}$$

The sign of the second derivative is dictated by the sign of the numerator, since the denominator is always positive. Thus, we need only be concerned with the sign of the numerator. After simplification and gathering of common elements, the numerator reduces to the following:

$$\begin{aligned}&n^5\cdot\left(8k_1^2\cdot k_{12}\right)+n^4\cdot\left(16k_1^2\cdot k_{14}\right)+n^3\cdot\left(16k_1\cdot\left(k_1\cdot C_1+k_{14}\cdot C_2\right)\right)+\\&+n^2\cdot\left(8k_1\cdot k_{14}\cdot(k_4-2k_3)+12k_1\cdot C_1\cdot C_2-2k_{12}\cdot C_2\cdot(k_4-2k_3)\right)+\\&+n\cdot\left(2\cdot C_1\cdot C_2^2+2\cdot(k_4-2k_3)\cdot\left(4k_1\cdot C_1-k_{12}\cdot(k_4-2k_3)\right)\right)+\\&+2\cdot(k_4-2k_3)\cdot\left(C_1\cdot C_2-k_{14}\cdot(k_4-2k_3)\right)\end{aligned}$$

Note that the coefficients of n^5, n^4, and n^3 are all positive. However, this need not necessarily be the case for smaller powers of *n*. The following conditions suffice for the non-negitivity of the other coefficients:

$$\begin{aligned}&[I]\quad &&(4\cdot k_{14}-k_2\cdot C_2)\cdot(k_4-2k_3)+6\cdot C_1\cdot C_2\geq 0\\&[II]\quad &&\left(4\cdot k_1\cdot C_1-k_{12}(k_4-2k_3)\right)\cdot(k_4-2k_3)+C_1\cdot C_2^2\geq 0\\&[III]\quad &&C_1\cdot C_2-k_{14}(k_4-2k_3)\geq 0\end{aligned}$$

Unfortunately, these conditions cannot be reduced any further by returning to the original coefficients. Therefore, in general it appears that $WIP_j(n)$ is not necessarily strictly convex. Nevertheless, note that there is a strong reason to believe that it does behave so in the vast majority of real situations. To support this, we note that the higher powers of the second derivative are all positive and obviously, even for small *n* values, they strongly dominate the lower powers and dictate the sign of the derivative.

Appendix B: BASIC program for the FIHLS heuristic

```
REM ************************************************************
REM ***          LOT STREAMING NEH-BASED ALGORITHM           ***
REM ************************************************************

REM *** OUTPUT FILE
OPEN "TRY5.DAT" FOR OUTPUT AS #1
REM *** DEFINITIONS
DIM TOTAL(25), ASSIGN(25), TMP(25), LST(25)
DIM SU(25, 25), P(25, 25), NSUBLOT(25), R(25), COUNTER(25)
DIM PT(400, 25), C(400, 25), SEQ(400), TRY(400), SOFAR(400)

REM *** INPUT DATA
REM * #LOTS, #MACHINES
REM * SUBLOT SIZE: 100,50,25,12,6,3,1 AND PERCENTAGE OF SETUP:0,.05,.10,.20
REM * QUANTITIES FOR ITEMS PER EACH LOT [50-100]
REM * SETUP TIME FOR EACH LOT ON ALL MACHINES
REM * UNIT PROCESSING TIME FOR EACH LOT ON ALL MACHINES  [5-15]

CLS
RANDOMIZE TIMER
N = 3: M = 6
QMIN = 50: QMAX = 100
PMIN = 5: PMAX = 15
EXPMT = 1
10
FOR I = 1 TO N
  Q(I) = INT(QMIN + (QMAX - QMIN + 1) * RND(1))
  FOR J = 1 TO M
    P(I, J) = INT(PMIN + (PMAX - PMIN + 1) * RND(1))
  NEXT J
NEXT I

PERCENT = 0
20
SIZE = 100
FOR I = 1 TO N
  FOR J = 1 TO M
     SU(I, J) = PERCENT * Q(I) * P(I, J)
  NEXT J
NEXT I

30
REM *** STEP 1
NSUBLOTS = 0
FOR I = 1 TO N
   REM READ Q(I)
   NSUBLOT(I) = INT(Q(I) / SIZE)
   R(I) = Q(I) - SIZE * NSUBLOT(I)
   IF R(I) > 0 THEN NSUBLOT(I) = NSUBLOT(I) + 1
   NSUBLOTS = NSUBLOTS + NSUBLOT(I)
NEXT I

START = TIMER

REM *** STEP 2

FOR I = 1 TO N: TOTAL(I) = 0: ASSIGN(I) = 0: NEXT I

FOR I = 1 TO N: TOTAL(I) = 0
  FOR J = 1 TO M
    REM READ P(I, J)
    PT(I, J) = SU(I, J) + SIZE * P(I, J): REM * RULE FOR ORDERING:Q[I] OR SIZE
    TOTAL(I) = TOTAL(I) + PT(I, J)
  NEXT J
NEXT I

REM * ARRANGE LOTS IN DESCENDING ORDER
LST(0) = 0
FOR I = 1 TO N: TMP(I) = 0
  FOR K = 1 TO N
    IF TOTAL(K) >= TMP(I) AND ASSIGN(K) = 0 THEN TMP(I) = TOTAL(K): INDEX = K
  NEXT K
  LST(I) = INDEX: ASSIGN(INDEX) = 1
NEXT I

REM * SEQUENCE THE SUBLOTS OF THE LARGEST LOT FIRST
FOR I = 1 TO NSUBLOT(LST(1))
   SEQ(I) = LST(1)
NEXT I
SEQLENGTH = NSUBLOT(LST(1))

REM *** STEP 3
LOT = 2
100
NSUB = 1
200
SEQLENGTH = SEQLENGTH + 1: REM * CURRENT LENGTH OF PARTIAL SEQUENCE
POSITION = 1: REM * CURRENT POSITION FOR THE NEXT SUBLOT WITHIN THE SEQUENCE
BESTSOFAR = 1000000

300
LOCN = 1: REM * LOCATION POINTER
400
  IF LOCN < POSITION THEN TRY(LOCN) = SEQ(LOCN)
  IF LOCN = POSITION THEN TRY(LOCN) = LST(LOT)
  IF LOCN > POSITION THEN TRY(LOCN) = SEQ(LOCN - 1)
  LOCN = LOCN + 1
  IF LOCN <= SEQLENGTH THEN 400

REM * EVALUATE CURRENT PARTIAL SEQUENCE
  FOR I = 0 TO SEQLENGTH
    FOR J = 0 TO M
      C(I, J) = 0
    NEXT J
  NEXT I
  FOR J = 1 TO M
    FOR I = 1 TO N: COUNTER(I) = 0: NEXT I
    FOR I = 1 TO SEQLENGTH
      COUNTER(TRY(I)) = COUNTER(TRY(I)) + 1
      IF COUNTER(TRY(I)) = NSUBLOT(TRY(I)) AND R(TRY(I)) > 0 THEN SUBSIZE = R(TRY(I)) ELSE
SUBSIZE = SIZE
      IF TRY(I) = TRY(I - 1) THEN PT(I, J) = SUBSIZE * P(TRY(I), J) ELSE PT(I, J) = SU(TRY(I), J) + SUBSIZE
* P(TRY(I), J)
      IF C(I - 1, J) > C(I, J - 1) THEN C(I, J) = C(I - 1, J) + PT(I, J) ELSE C(I, J) = C(I, J - 1) + PT(I, J)
      REM PRINT "C["; I; J; "]="; C(I, J);
    NEXT I
  NEXT J

REM * COMPARE AND SAVE THE BEST SEQUENCE
500
  IF C(SEQLENGTH, M) > BESTSOFAR THEN 600
  FOR I = 1 TO SEQLENGTH
    SOFAR(I) = TRY(I)
  NEXT I
  BESTSOFAR = C(SEQLENGTH, M)

600
  POSITION = POSITION + 1
  IF POSITION <= SEQLENGTH THEN 300

REM * THE BEST PARTIAL SEQUENCE FOR GIVEN SEQ_LENGTH IS NOW ESTABLISHED
  FOR I = 1 TO SEQLENGTH
    SEQ(I) = SOFAR(I)
  NEXT I
  BEST = BESTSOFAR

NSUB = NSUB + 1
IF NSUB <= NSUBLOT(LST(LOT)) THEN 200

700
LOT = LOT + 1
IF LOT <= N THEN 100

FINISH = TIMER - START

REM *** LOWER BOUND ON THE OPTIMAL SOLUTION
LB = 0
REM * JOB BASED BOUND
FOR I = 1 TO N: LB1(I) = 0
  FOR J = 1 TO M
    LB1(I) = LB1(I) + SU(I, J) + P(I, J)
  NEXT J
  IF LB1(I) > LB THEN LB = LB1(I)
NEXT I

REM * MACHINE BASED BOUND
FOR J = 1 TO M: LB2(J) = 0
  FOR I = 1 TO N
    LB2(J) = LB2(J) + SU(I, J) + Q(I) * P(I, J)
  NEXT I
  IF LB2(J) > LB THEN LB = LB2(J)
NEXT J

REM * A COMPLETE SEQUENCE IS NOW OBTAINED
PRINT "SUBLOT SIZE="; SIZE; "  SETUP PERCENTAGE:"; PERCENT * 100; "%"
PRINT "BEST SEQUENCE:";
FOR I = 1 TO SEQLENGTH: PRINT SEQ(I); : NEXT I: PRINT
PRINT "RESULTANT MAKESPAN = "; BEST
PRINT "CPU TIME:"; FINISH; " SEC."
PRINT "RATIO TO LOWER BOUND ON THE OPTIMAL SOLUTION: "; BEST / LB
PRINT #1, "SUBLOT SIZE="; SIZE; "  SETUP PERCENTAGE:"; PERCENT * 100; "%"
PRINT #1, "BEST SEQUENCE:";
FOR I = 1 TO SEQLENGTH: PRINT #1, SEQ(I); : NEXT I: PRINT
PRINT #1, "RESULTANT MAKESPAN = "; BEST
PRINT #1, "CPU TIME:"; FINISH; " SEC."
PRINT #1, "RATIO TO LOWER BOUND ON THE OPTIMAL SOLUTION: "; BEST / LB

SIZE = INT(SIZE / 2)
IF SIZE >= 3 THEN 30
IF PERCENT = 0 THEN PERCENT = .05: GOTO 20
REM IF PERCENT > 0 AND PERCENT < .2 THEN PERCENT = PERCENT * 2: GOTO 20
EXPMT = EXPMT + 1
IF EXPMT <= 8 THEN 10
```

Appendix C: Example of file output of the FIHLS heuristic program

```
SUBLOT SIZE= 100   SETUP PERCENTAGE: 0 %
BEST SEQUENCE: 7  5  6  8  1  9  3  10  2  4 RESULTANT MAKESPAN =  11923
CPU TIME: 1.367188  SEC.
RATIO TO LOWER BOUND ON THE OPTIMAL SOLUTION:  1.556527
SUBLOT SIZE= 50   SETUP PERCENTAGE: 0 %
BEST SEQUENCE: 5  6  1  5  8  8  6  3  9  2  4  10  1  9  7  3  10  2  4  7 RESULTANT MAKESPAN =  10030
CPU TIME: 9.179688  SEC.
RATIO TO LOWER BOUND ON THE OPTIMAL SOLUTION:  1.309399
SUBLOT SIZE= 25   SETUP PERCENTAGE: 0 %
BEST SEQUENCE: 5  6  1  2  10  7  8  8  8  1  1  5  6  6  3  9  9  5  4  10  4  10  8  9  1  6  3  2  3  5  7  7  3  2  4  7
RESULTANT MAKESPAN =  9005
CPU TIME: 49.71094  SEC.
RATIO TO LOWER BOUND ON THE OPTIMAL SOLUTION:  1.175587
SUBLOT SIZE= 12   SETUP PERCENTAGE: 0 %
BEST SEQUENCE: 7  6  5  1  10  8  2  7  8  5  10  8  5  8  8  5  10  8  5  8  1  5  4  10  8  4  1  1  1  6  5  6  6  6  5  6
6  1  3  3  3  3  4  3  7  9  3  9  9  7  9  7  9  2  2  4  4  10  7  2  1  6  9  3  2  3  7 RESULTANT MAKESPAN =  8322
CPU TIME: 305.5469  SEC.
RATIO TO LOWER BOUND ON THE OPTIMAL SOLUTION:  1.086423
SUBLOT SIZE= 6   SETUP PERCENTAGE: 0 %
BEST SEQUENCE: 7  6  2  1  5  10  10  8  10  8  5  8  10  8  5  8  5  10  8  8  5  8  5  10  8  8  5  10  8  8  5  8  5  10
8  8  5  1  1  1  1  5  1  1  1  1  1  5  4  10  8  4  1  6  6  5  6  6  6  5  6  6  6  5  6  6  6  5  6  6  3  7  1  3  3  2  3  6  1  3
3  7  4  7  3  7  4  3  2  3  4  3  7  7  4  3  7  9  3  7  9  4  7  4  7  4  3  7  9  9  2  3  7  9  9  9  9  9  2  9  2  2  10  9  2  2
3 RESULTANT MAKESPAN =  8058
CPU TIME: 2070.25  SEC.
RATIO TO LOWER BOUND ON THE OPTIMAL SOLUTION:  1.051958
SUBLOT SIZE= 3   SETUP PERCENTAGE: 0 %
BEST SEQUENCE: 7  2  7  2  6  2  7  1  5  10  10  10  10  10  10  8  10  8  8  5  10  8  8  5  10  8  5  8  8  5  10  8  8
5  10  8  5  8  8  5  10  8  5  8  10  8  5  8  8  5  10  8  5  8  10  8  5  8  5  8  10  8  5  8  5  10  8  8  5  8  1  1  5  1  1  10
8  5  4  1  8  4  5  4  1  1  1  1  1  5  1  1  1  1  5  1  1  1  1  1  5  1  10  8  4  1  1  5  4  1  6  6  5  6  6  6  5  6  6  6  5  6
6  6  5  6  6  6  5  6  6  6  5  6  6  6  5  6  6  6  5  6  6  3  3  4  3  4  4  3  7  4  7  4  7  4  3  2  6  3  3  4  7  3  7  4  3  2  3
4  3  2  6  3  3  4  3  4  7  3  2  3  3  2  6  3  3  3  2  7  7  1  7  3  7  9  3  9  3  9  9  2  3  9  9  9  2  3  9  9  3  9  9  9  2  3
9  9  3  9  9  9  9  2  9  9  2  9  2  2  10  6  1  2  4  3  7  3  2  7  4  7  7  7  7  7  7  7  7  7  7  7 RESULTANT
MAKESPAN =  7953
CPU TIME:-70744.86  SEC.
RATIO TO LOWER BOUND ON THE OPTIMAL SOLUTION:  1.038251
SUBLOT SIZE= 100   SETUP PERCENTAGE: 5 %
BEST SEQUENCE: 7  9  5  6  4  8  1  3  10  2 RESULTANT MAKESPAN =  12102.3
CPU TIME: 1.370117  SEC.
RATIO TO LOWER BOUND ON THE OPTIMAL SOLUTION:  1.5047
SUBLOT SIZE= 50   SETUP PERCENTAGE: 5 %
BEST SEQUENCE: 7  7  6  6  5  9  1  5  8  8  2  2  3  4  4  10  9  1  3  10 RESULTANT MAKESPAN =  10500.35
CPU TIME: 9.339844  SEC.
RATIO TO LOWER BOUND ON THE OPTIMAL SOLUTION:  1.305527
SUBLOT SIZE= 25   SETUP PERCENTAGE: 5 %
BEST SEQUENCE: 7  7  7  7  6  6  6  6  2  4  5  9  9  9  1  1  1  1  5  5  5  8  8  8  8  3  3  3  2  2  4  4  10  10  10  3
RESULTANT MAKESPAN =  9529.051
CPU TIME: 48.77051  SEC.
RATIO TO LOWER BOUND ON THE OPTIMAL SOLUTION:  1.184763
SUBLOT SIZE= 12   SETUP PERCENTAGE: 5 %
```

```
BEST SEQUENCE: 5  7  7  7  7  7  7  7  6  6  6  6  6  6  6  6  2  2  2  10  5  5  5  8  8  8  8  8  8  8  8  5  5  5  5  9  9  2
2  4  4  4  4  4  3  3  3  3  3  3  3  9  9  9  9  1  1  1  1  1  1  1  10  10  10  10  3 RESULTANT MAKESPAN =  9183.5
CPU TIME: 297.0898  SEC.
RATIO TO LOWER BOUND ON THE OPTIMAL SOLUTION:  1.1418
SUBLOT SIZE= 6   SETUP PERCENTAGE: 5 %
BEST SEQUENCE: 5  5  5  5  5  5  5  5  5  5  5  7  7  7  7  7  7  7  7  7  7  7  7  7  8  8  8  8  8  8  8  8  8  8  8  8  8  8
8  8  5  5  5  5  5  1  1  1  1  1  1  1  1  1  1  1  1  1  4  4  4  4  4  4  4  4  4  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  2  2  2
2  10  10  10  10  10  10  10  10  10  10  2  2  2  9  9  9  9  9  9  9  9  9  9  9  2  2  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3
RESULTANT MAKESPAN =  8915.851
CPU TIME: 1956.88  SEC.
RATIO TO LOWER BOUND ON THE OPTIMAL SOLUTION:  1.108523
SUBLOT SIZE= 3   SETUP PERCENTAGE: 5 %
BEST SEQUENCE: 7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  5  5  5  5  5  5  5  5  5  5  5  5
5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8
8  8  8  8  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6
6  6  6  6  6  6  6  6  6  6  6  6  6  10  10  10  10  10  10  10  10  10  10  10  10  10  10  10  10  10  10  10  10  9  9  9  9
9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  2  2  2  2  2  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  3  3  3  3  3  3  3
3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  2  2  2  2  2  2  2  2  2  2  2  2  2 RESULTANT
MAKESPAN =  8852.35
CPU TIME: 14988.23  SEC.
RATIO TO LOWER BOUND ON THE OPTIMAL SOLUTION:  1.100628
SUBLOT SIZE= 100   SETUP PERCENTAGE: 10 %
BEST SEQUENCE: 7  9  5  6  8  1  3  10  2  4 RESULTANT MAKESPAN =  12678.6
CPU TIME: 1.429688  SEC.
RATIO TO LOWER BOUND ON THE OPTIMAL SOLUTION:  1.5047
SUBLOT SIZE= 50   SETUP PERCENTAGE: 10 %
BEST SEQUENCE: 5  6  6  1  1  5  7  7  8  8  3  9  9  2  2  10  4  4  3  10 RESULTANT MAKESPAN =  11429
CPU TIME: 9.169922  SEC.
RATIO TO LOWER BOUND ON THE OPTIMAL SOLUTION:  1.356397
SUBLOT SIZE= 25   SETUP PERCENTAGE: 10 %
BEST SEQUENCE: 7  5  6  6  6  6  7  7  7  1  1  5  5  5  8  8  8  8  1  1  4  4  4  10  10  10  9  9  9  2  2  2  3  3  3  3
RESULTANT MAKESPAN =  10328.5
CPU TIME: 48.6582  SEC.
RATIO TO LOWER BOUND ON THE OPTIMAL SOLUTION:  1.225789
SUBLOT SIZE= 12   SETUP PERCENTAGE: 10 %
BEST SEQUENCE: 5  5  5  5  5  5  5  5  6  6  6  6  6  6  6  6  9  9  9  9  9  9  4  4  4  4  4  10  10  10  10  10  8  8  8  8
8  8  8  8  2  1  1  1  1  1  1  1  2  2  2  2  7  7  7  7  7  7  7  3  3  3  3  3  3  3  3 RESULTANT MAKESPAN =
9707.299
CPU TIME: 296.0508  SEC.
RATIO TO LOWER BOUND ON THE OPTIMAL SOLUTION:  1.152065
SUBLOT SIZE= 6   SETUP PERCENTAGE: 10 %
BEST SEQUENCE: 7  7  7  7  7  7  7  7  7  7  7  7  7  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  8  8  8  8  8  8  8  8  8
8  8  8  8  8  8  8  1  1  1  1  1  1  1  1  1  1  1  1  1  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  10  10  10  10  10  10  10  10
10  10  9  9  9  9  9  9  9  9  9  9  9  4  4  4  4  4  4  4  4  4  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  2  2  2  2  2  2  2  2  2
RESULTANT MAKESPAN =  9539.5
CPU TIME: 1953.26  SEC.
RATIO TO LOWER BOUND ON THE OPTIMAL SOLUTION:  1.132151
SUBLOT SIZE= 3   SETUP PERCENTAGE: 10 %
BEST SEQUENCE: 7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  5  5  5  5  5  5  5  5  5  5  5  5
5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8
8  8  8  8  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6
6  6  6  6  6  6  6  6  6  6  6  6  6  10  10  10  10  10  10  10  10  10  10  10  10  10  10  10  10  10  10  10  10  9  9  9  9
9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  3  3  3  3  3  3  3  3  3  3  3  3
3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2 RESULTANT
MAKESPAN =  9437.5
CPU TIME: 16229  SEC.
```

```
RATIO TO LOWER BOUND ON THE OPTIMAL SOLUTION:  1.120045
SUBLOT SIZE= 100   SETUP PERCENTAGE: 0 %
BEST SEQUENCE: 6  2  3  4  5  1  7  9  10  8 RESULTANT MAKESPAN =  11157
CPU TIME: 1.371094  SEC.
RATIO TO LOWER BOUND ON THE OPTIMAL SOLUTION:  1.443338
SUBLOT SIZE= 50   SETUP PERCENTAGE: 0 %
BEST SEQUENCE: 3  5  4  5  2  6  3  1  1  7  7  2  4  10  9  8  9  10  8  6 RESULTANT MAKESPAN =  9566
CPU TIME: 9.230469  SEC.
RATIO TO LOWER BOUND ON THE OPTIMAL SOLUTION:  1.237516
SUBLOT SIZE= 25   SETUP PERCENTAGE: 0 %
BEST SEQUENCE: 3  3  5  6  6  3  1  1  1  5  10  7  2  7  6  7  4  2  5  4  10  7  3  4  8  2  9  9  8  9  8  9  10
RESULTANT MAKESPAN =  8604
CPU TIME: 38.76953  SEC.
RATIO TO LOWER BOUND ON THE OPTIMAL SOLUTION:  1.113066
SUBLOT SIZE= 12   SETUP PERCENTAGE: 0 %
BEST SEQUENCE: 3  3  2  6  3  5  1  1  1  1  1  1  1  10  9  9  3  10  6  10  2  6  6  10  7  7  2  7  2  6  7  4  3  5  7  7
5  7  4  5  7  9  9  9  4  5  4  8  5  2  9  4  3  3  2  10  9  3  8  6  9  8  8  8  8  8  8 RESULTANT MAKESPAN =  8200
CPU TIME: 294.5117  SEC.
RATIO TO LOWER BOUND ON THE OPTIMAL SOLUTION:  1.060802
SUBLOT SIZE= 6   SETUP PERCENTAGE: 0 %
BEST SEQUENCE: 3  3  2  6  3  3  3  1  1  1  1  1  1  1  1  1  1  1  1  1  9  10  9  10  9  9  10  6  9  10  4  2  9  3  10  4
6  10  4  4  6  2  10  2  6  6  10  6  6  10  7  7  2  6  7  7  7  7  6  7  7  7  7  6  5  7  7  7  5  7  7  2  5  7  5  2  5  2  8  8  3
2  9  9  9  4  2  8  4  5  4  3  9  9  9  9  3  8  3  4
SUBLOT SIZE= 6   SETUP PERCENTAGE: 0 %
BEST SEQUENCE: 3  3  2  6  3  3  3  1  1  1  1  1  1  1  1  1  1  1  1  1  9  10  9  10  9  9  10  6  9  10  4  2  9  3  10  4
6  10  4  4  6  2  10  2  6  6  10  6  6  10  7  7  2  6  7  7  7  7  6  7  7  7  7  6  5  7  7  7  5  7  7  2  5  7  5  2  5  2  8  8  3
2  9  9  9  4  2  8  4  5  4  3  9  9  9  9  3  8  3  4
RESULTANT MAKESPAN =  7789
CPU TIME: 17788  SEC.
RATIO TO LOWER BOUND ON THE OPTIMAL SOLUTION:  1.020103
```

Appendix D: Summary of output of the FIHLS

N=10						SU=0%					
	EXEPMT	1	2	3	4	5	6	7	8	#Sublots	AVG.
SUBLOT	100	1.556	1.443	1.353	1.344	1.46	1.271	1.353	1.344	1	1.391
SIZE	50	1.309	1.238	1.166	1.182	1.203	1.129	1.166	1.182	2	1.197
	25	1.117	1.113	1.087	1.087	1.096	1.057	1.087	1.087	4	1.091
	12	1.051	1.06	1.037	1.045	1.045	1.027	1.037	1.045	8	1.043
	6	1.038	1.038	1.019	1.021	1.029	1.014	1.019	1.021	16	1.025
	3	1.019	1.02	1.011	1.01	1.015	1.009	1.011	1.01	32	1.013

N=10						SU=5%					
	EXEPMT	1	2	3	4	5	6	7	8	#Sublots	
SUBLOT	100	1.505	1.436	1.53	1.624	1.505	1.518	1.443	1.611	1	1.521
SIZE	50	1.306	1.237	1.297	1.357	1.306	1.287	1.246	1.346	2	1.298
	25	1.185	1.134	1.152	1.17	1.185	1.143	1.139	1.161	4	1.159
	12	1.142	1.078	1.091	1.104	1.142	1.082	1.088	1.095	8	1.103
	6	1.109	1.068	1.115	1.162	1.109	1.106	1.07	1.153	16	1.111
	3	1.1	1.079	1.077	1.075	1.1	1.068	1.075	1.08	32	1.082

N=6						SU=0%					
	EXEPMT	1	2	3	4	5	6	7	8	#Sublots	AVG.
SUBLOT	100	1.913	1.77	1.613	1.691	1.765	1.929	1.781	1.597	1	1.757
SIZE	50	1.455	1.328	1.324	1.449	1.582	1.492	1.45	1.311	2	1.424
	25	1.192	1.154	1.139	1.223	1.273	1.232	1.213	1.128	4	1.194
	12	1.092	1.066	1.087	1.105	1.177	1.096	1.099	1.076	8	1.100
	6	1.039	1.034	1.048	1.053	1.106	1.047	1.048	1.038	16	1.052
	3	1.016	1.018	1.03	1.025	1.083	1.022	1.023	1.02	32	1.030

N=6						SU=5%					
	EXEPMT	1	2	3	4	5	6	7	8	#Sublots	
SUBLOT	100	1.913	1.77	1.613	1.691	1.761	1.929	1.781	1.597	1	1.757
SIZE	50	1.455	1.371	1.376	1.456	1.611	1.515	1.462	1.362	2	1.451
	25	1.224	1.179	1.207	1.253	1.337	1.252	1.279	1.195	4	1.241
	12	1.147	1.095	1.133	1.172	1.221	1.154	1.198	1.122	8	1.155
	6	1.101	1.077	1.143	1.125	1.147	1.086	1.18	1.132	16	1.124
	3	1.075	1.069	1.133	1.104	1.176	1.095	1.168	1.122	32	1.118

N=3						SU=0%					
	EXEPMT	1	2	3	4	5	6	7	8	#Sublots	AVG.
SUBLOT	100	2.569	2.269	2.452	2.262	2.355	2.561	2.454	2.664	1	2.448
SIZE	50	1.955	1.612	1.719	1.882	1.694	1.902	1.62	1.821	2	1.776
	25	1.417	1.281	1.367	1.402	1.351	1.403	1.293	1.362	4	1.360
	12	1.173	1.126	1.172	1.186	1.163	1.172	1.143	1.167	8	1.163
	6	1.095	1.069	1.083	1.082	1.078	1.084	1.068	1.083	16	1.080
	3	1.047	1.034	1.044	1.043	1.046	1.042	1.034	1.04	32	1.041

N=3						SU=5%					
	EXEPMT	1	2	3	4	5	6	7	8	#Sublots	
SUBLOT	100	2.569	2.269	2.452	2.622	2.355	2.561	2.454	2.664	1	2.493
SIZE	50	1.984	1.617	1.748	1.917	1.724	1.934	1.651	1.859	2	1.804
	25	1.475	1.288	1.397	1.41	1.414	1.444	1.362	1.429	4	1.402
	12	1.243	1.141	1.259	1.215	1.226	1.202	1.187	1.211	8	1.211
	6	1.2	1.087	1.136	1.121	1.14	1.116	1.104	1.134	16	1.130
	3	1.18	1.054	1.103	1.078	1.102	1.159	1.069	1.084	32	1.104

Appendix E: A Proof for the decreasing effect of the number of lots on makespan reduction

In the sequel, it is shown that, as the number of lots considered at time zero increases, the makespan reduction via lot-streaming decreases. We start with the two-machine case which is then generalized to the m-machine case. In the two-machine case, it will be shown that, as long as there is an infinite queue of lots available at the first machine (i.e., *N* approaches infinity), lot-streaming does not result in makespan reduction. The following notation is used:

$C_{[i],j}$ - The completion time of lot [i] in the sequence on machine j. $i = 1,..,N$

$C^{LS}_{[i],j}$ - The completion time of lot [i] in the sequence on machine j, under Lot Streaming (LS).

Theorem 1. In the two-machine case, makespan reduction due to lot streaming decreases to zero with the number of lots.

Proof.

For *m*=2, in the absence of lot streaming, we have:

$$C_{[N],1} = C^{LS}_{[N],1} = \sum_{i=1}^{N} Q_{[i]} \cdot p_{[i],1} \tag{1-1}$$

$$\begin{aligned} C_{[N],2} &= \max\{C_{Nn],1}, C_{N-1],2}\} + Q_{[N]} \cdot p_{[N],2} = \\ &= \max\{C_{[N],1} + Q_{[N]} \cdot p_{[N],2}, C_{[N-1],2} + Q_{[N]} \cdot p_{[N],2}\} = \\ &= \max\{C_{[N-1],1} + Q_{[N]} \cdot p_{[N],1} + Q_{[N]} \cdot p_{[N],2}, C_{[N-1],2} + Q_{[N]} \cdot p_{[N],2}\} \end{aligned} \tag{1-2}$$

Note that $C_{[N],2}$ in this case represents the makespan, i.e., the completion time of all the lots up to the *N*-th lot on the second machine. Under lot streaming of unit-sized sublots, the expression becomes:

$$C^{LS}_{[N],2} = \max\{C^{LS}_{[N-1],2} + Q_{[N]} \cdot p_{[N],2}, C^{LS}_{[N-1],1} + p_{[N],1} + Q_{[N]} \cdot p_{[N],2}, C^{LS}_{[N-1],1} + Q_{[N]} \cdot p_{[N],1} + p_{[N],2}\} \tag{1-3}$$

Since: $C^{LS}_{[i],1} = C_{[i],1} \quad \forall i$, and: $C^{LS}_{[N-1],2} > C_{[N-1],1}$, substituting in (1-2) we get:

$$C^{LS}_{[N],2} \le \max\left\{C_{[N-1],1} + Q_{[N]} \cdot p_{[N],2}, C_{[N-1],1} + p_{[N],1} + Q_{[N]} \cdot p_{[N],2}, C_{[N-1],1} + Q_{[N]} \cdot p_{[N],1} + p_{[N],2}\right\} =$$
$$= C_{[N-1],1} + \max\left\{Q_{[N]} \cdot p_{[N],2}, p_{[N],1} + Q_{[N]} \cdot p_{[N],2}, Q_{[N]} \cdot p_{[N],1} + p_{[N],2}\right\} \tag{1-4}$$

Dividing (1-4) by (1-2), we have:

$$\frac{C^{LS}_{[N],2}}{C_{[N],2}} \ge \frac{C_{[N-1],1} + \max\left\{Q_{[N]} \cdot p_{[N],2}, p_{[N],1} + Q_{[N]} \cdot p_{[N],2}, Q_{[N]} \cdot p_{[N],1} + p_{[N],2}\right\}}{\min\left\{C_{[N-1],1} + Q_{[N]} \cdot p_{[N],1} + Q_{[N]} \cdot p_{[N],2}, C_{[N-1],2} + Q_{[N]} \cdot p_{[N],2}\right\}} \ge$$
$$\ge \frac{C_{[N-1],1} + \max\left\{Q_{[N]} \cdot p_{[N],2}, p_{[N],1} + Q_{[N]} \cdot p_{[N],2}, Q_{[N]} \cdot p_{[N],1} + p_{[N],2}\right\}}{C_{[N-1],1} + Q_{[N]} \cdot p_{[N],1} + Q_{[N]} \cdot p_{[N],2}} \xrightarrow[N \to \infty]{} \frac{C_{[N-1],1}}{C_{[N-1],1}} = 1 \tag{1-5}$$

QED.

Hence, regardless of the lot sizes and the processing times, in a two-machine flow-shop, in which the number of lots approaches infinity (or, alternatively, the next lot is always available at the first machine before completing the previous lot), lot streaming would not result in makespan reduction. Next, it is shown that the above theorem holds for *m*>2 as well.

Theorem 2. (Generalization of Theorem 1 to the m-machine case)
In the general m-machine case, makespan reduction due to lot streaming decreases to zero with the number of lots.
Proof.
Consider two cases:
Case (i): All the machines have equal workload.
Since M_1 works continuously and no other machine has a higher workload, there exists some large *N* such that the makespan (without lot streaming) is as follows:

$$C_{[N],m} = C_{[N],1} + \sum_{j=2}^{m} Q_{[N]} \cdot p_j \tag{1-6}$$

On the other hand, under lot streaming, we have the following inequality holding for any *N*.

$$C^{LS}_{[N],m} \geq C^{LS}_{[N],1} + \sum_{j=2}^{m} p_j \tag{1-7}$$

Since: $C^{LS}_{[n],1} = C_{[n],1} \quad \forall n$, dividing (1-7) by (1-6) we get:

$$\frac{C^{LS}_{[N],m}}{C_{[N],m}} \geq \frac{C_{[N],1} + \sum_{j=2}^{m} p_j}{C_{[N],1} + \sum_{j=2}^{m} Q_{[N]} \cdot p_j} \xrightarrow[N \to \infty]{} \frac{C_{[N],1}}{C_{[N],1}} = 1 \tag{1-8}$$

Case (ii): One machine (or more) has a higher workload than the other machines.
Let M_{BN} denote the machine with the highest workload. Then, for some sufficiently large n *(<<N)* in the sequence, M_{BN} works continuously. To see that this is indeed the case, suppose that this is not true, i.e., suppose there exist periodical idle times on M_{BN}. This implies that one of the machines preceding M_{BN}, say M_k $\left(1 \leq k < BN\right)$, does not make the next lot available to M_{BN} upon completion of the present lot. Thus, it must be that processing the lots on M_k takes more time than processing them on M_{BN}. However, this contradicts the fact that M_{BN} has a higher workload. The remainder of the proof is similar to case (i) above with M_{BN} replacing M_1.
QED.

To conclude, we make the following comment. Note that the difference in the makespans (with and without lot streaming) is merely a constant. Although this constant might make a substantial difference when dealing with a few (or more generally, finite number of) lots, it is negligible if an infinite queue of lots is considered and all the lots are readily available at the first machine. Moreover, since the difference in the makespans is a constant, obviously the makespan reduction decreases with the number of lots, as the above theorem asserts.

Appendix F: A Comparative study of the optimality of the BMI and FIHLS heuristics

	Bottleneck Location								
Level of Dominance	B			C			E		
Strong Dominance	**1.006**	1.000	1.016	**1.013**	1.000	1.013	**1.007**	1.000	1.000
	1.022	1.000	1.088	**1.047**	1.061	1.035	**1.025**	1.014	1.000
	1.000	1.013	1.000	1.000	1.031	1.000	1.012	1.000	1.000
	1.000	1.013	1.053	1.010	1.012	1.000	1.000	1.089	1.049
	1.008	1.000	1.010	1.023	1.006	1.028	1.042	1.000	1.000
	1.000	1.002	1.023	1.127	1.015	1.113	1.022	1.000	1.028
Weak Dominance	**1.016**	1.027	1.001	**1.013**	1.023	1.000	**1.011**	1.000	1.009
	1.026	1.106	1.000	**1.037**	1.032	1.000	**1.038**	1.011	1.000
	1.000	1.021	1.041	1.021	1.006	1.017	1.000	1.022	1.019
	1.057	1.038	1.004	1.089	1.014	1.041	1.090	1.043	1.087
	1.000	1.016	1.022	1.000	1.027	1.011	1.000	1.031	1.010
	1.000	1.000	1.000	1.016	1.027	1.078	1.000	1.045	1.031
No Dominance	**1.013**	1.016	1.000	1.001	1.000	1.002	1.052	1.021	
	1.051	1.088	1.120	1.000	1.012	1.000	1.126	1.011	

Legend:
The numbers are the makespan ratios of the BMI and FIHLS heuristics to the optimal makespan
The upper left pair is the average of the others for the given combination

Appendix G: Format of input data of the simulation model

The ORDER worksheet

IGNORE	LOT	PART	PIECES	START	REPEAT	RUNITS	RDIST	RPT#	PRIOR
Order1	L1	Part1	25	06/01/98 08:00:00	46	min	exponential	1000	1
	L2	Part2	25	06/01/98 08:10:00	46	min	exponential	1000	2
	L3	Part3	25	06/01/98 08:20:00	46	min	exponential	1000	1
	L4	Part4	25	06/01/98 08:30:00	46	min	exponential	1000	2
	L5	Part5	25	06/01/98 08:40:00	46	min	exponential	1000	2
Order2	L1								
	...								

The STATIONS worksheet

STNFAM	STN	RULE	FWLRANK	TRACE	FWLRERANK	STNQTY
WS1	M11	rule_FIRST	rank_BMI	1	yes	1
WS2	M21	rule_FIRST	rank_BMI	1	yes	1
WS3	M31	rule_FIRST	rank_BMI	1	yes	1
WS4	M41	rule_FIRST	rank_BMI	1	yes	1
WS5	M51	rule_FIRST	rank_BMI	1	yes	1
WS6	M61	rule_FIRST	rank_BMI	1	yes	1
WS7	M71	rule_FIRST	rank_BMI	1	yes	1
WS8	M81	rule_FIRST	rank_BMI	1	yes	1
WS9	M91	rule_FIRST	rank_BMI	1	yes	1
WS10	M101	rule_FIRST	rank_BMI	1	yes	1

The PARTS worksheet

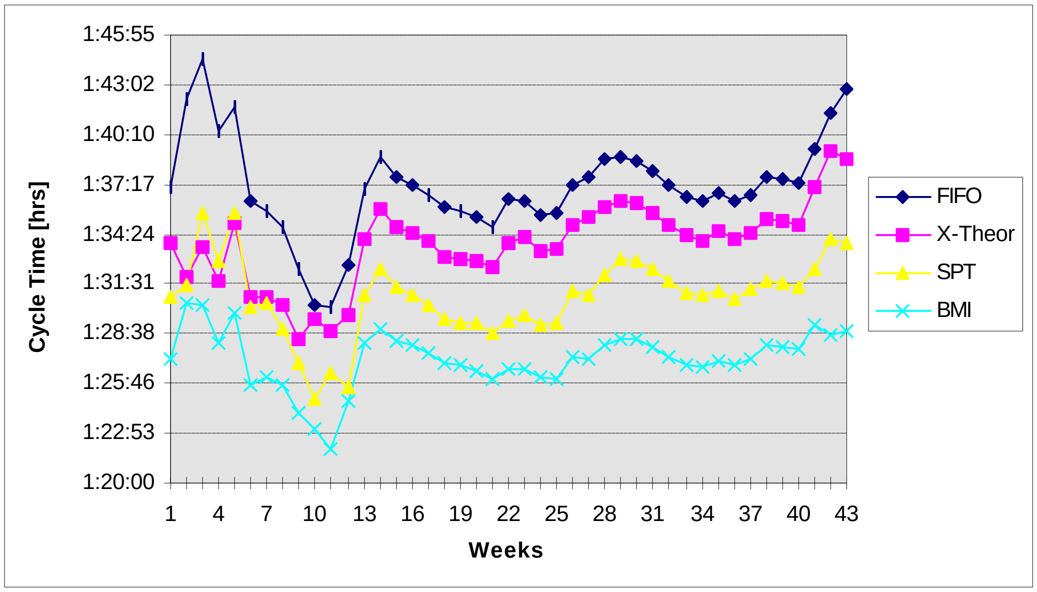
PART	ROUTEFILE	ROUTE
Part1	route.txt	P1
Part2	route.txt	P2
Part3	route.txt	P3
Part4	route.txt	P4
Part5	route.txt	P5

The ROUTE worksheet

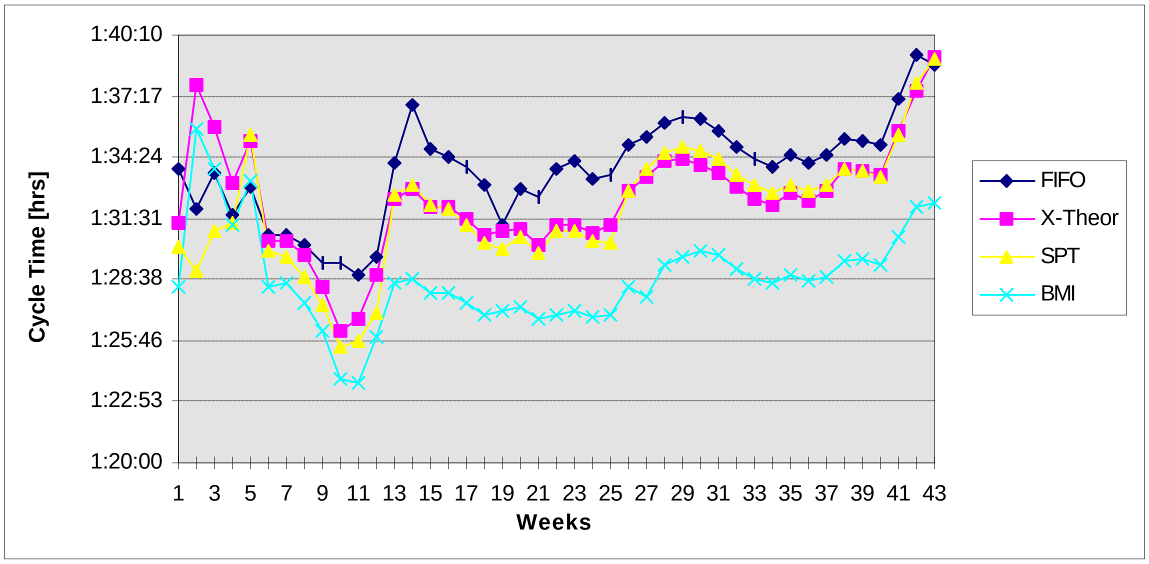
ROUTE	STEP	STNFAM	PTIME	PTUNITS	PDIST	PTIME2	PTPER	STIME	STUNITS	WHEN
P1	1	WS1	4	min	uniform	0.2	lot	1	min	always
	2	WS2	2	min	uniform	0.1	lot	2	min	always
	3	WS3	2	min	uniform	0.1	lot	1	min	always
	4	WS4	4	min	uniform	0.2	lot	1	min	always
	5	WS5	5	min	uniform	0.25	lot	2	min	always
	6	WS6	0	min	uniform	0	lot			
	7	WS7	6	min	uniform	0.3	lot	3	min	always
	8	WS8	0	min	uniform	0	lot			
	9	WS9	5	min	uniform	0.25	lot	3	min	always
	10	WS10	4	min	uniform	0.2	lot	1	min	always
P2	...									

Appendix H: Simulation model results

Experiment #2				
Snap	**FIFO**	**X-Theor**	**SPT**	**BMI**
Average	**1:37:08**	**1:33:51**	**1:30:46**	**1:27:11**
1	1:42:15	1:31:58	1:31:30	1:30:27
2	1:44:35	1:33:41	1:35:36	1:30:15
3	1:40:19	1:31:39	1:32:50	1:28:03
4	1:41:46	1:35:02	1:35:36	1:29:51
5	1:36:17	1:30:44	1:30:08	1:25:37
6	1:35:45	1:30:43	1:30:22	1:26:08
7	1:34:47	1:30:17	1:28:53	1:25:43
8	1:32:26	1:28:23	1:26:58	1:24:02
9	1:30:15	1:29:27	1:24:50	1:23:07
10	1:30:08	1:28:49	1:26:25	1:22:00
11	1:32:36	1:29:45	1:25:34	1:24:45
12	1:37:00	1:34:07	1:30:54	1:28:09
13	1:38:52	1:35:52	1:32:24	1:28:53
14	1:37:43	1:34:49	1:31:20	1:28:13
15	1:37:12	1:34:26	1:30:51	1:27:56
16	1:36:38	1:33:59	1:30:16	1:27:33
17	1:36:01	1:33:06	1:29:26	1:27:00
18	1:35:46	1:33:00	1:29:15	1:26:47
19	1:35:20	1:32:53	1:29:13	1:26:29
20	1:34:52	1:32:31	1:28:44	1:26:04
21	1:36:28	1:33:51	1:29:22	1:26:38
22	1:36:22	1:34:14	1:29:44	1:26:39
23	1:35:33	1:33:24	1:29:10	1:26:09
24	1:35:40	1:33:34	1:29:17	1:26:03
25	1:37:15	1:34:56	1:31:06	1:27:20
26	1:37:44	1:35:21	1:30:51	1:27:08
27	1:38:45	1:35:59	1:32:04	1:27:58
28	1:38:50	1:36:17	1:32:57	1:28:22
29	1:38:39	1:36:15	1:32:53	1:28:22
30	1:38:01	1:35:40	1:32:23	1:27:53
31	1:37:14	1:34:54	1:31:39	1:27:18
32	1:36:35	1:34:18	1:31:02	1:26:49
33	1:36:19	1:33:59	1:30:55	1:26:41
34	1:36:44	1:34:32	1:31:05	1:27:04
35	1:36:17	1:34:06	1:30:41	1:26:48
36	1:36:37	1:34:28	1:31:12	1:27:12
37	1:37:39	1:35:18	1:31:38	1:28:00
38	1:37:37	1:35:12	1:31:34	1:27:54
39	1:37:21	1:34:59	1:31:18	1:27:46
40	1:39:19	1:37:06	1:32:24	1:29:07
41	1:41:26	1:39:15	1:34:04	1:28:37
42	1:42:45	1:38:43	1:33:56	1:28:48

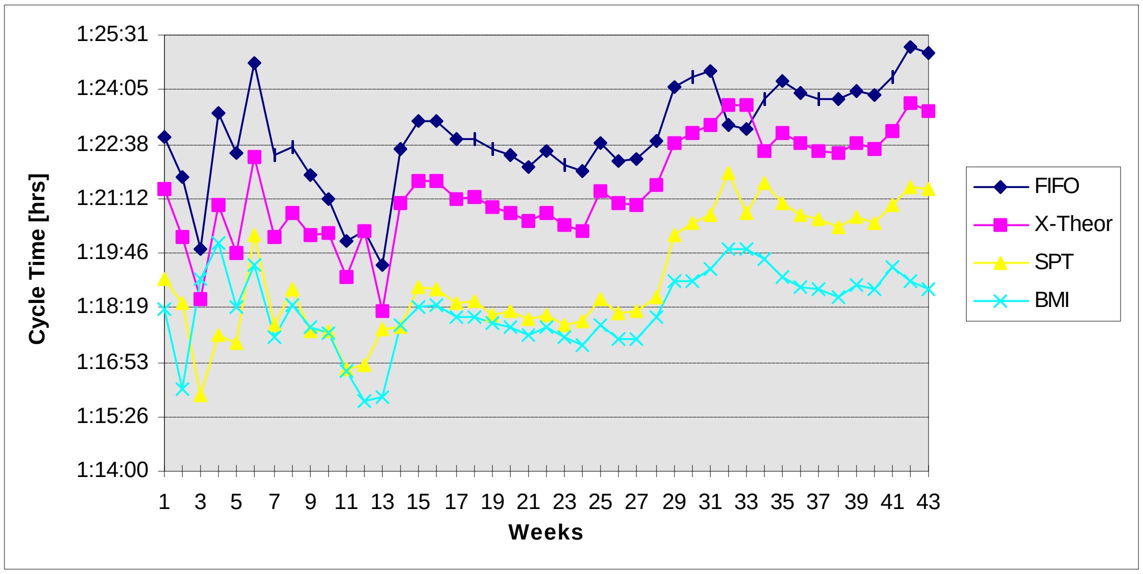

1:45:55
1:43:02
1:40:10
1:37:17
1:34:24
1:31:31
1:28:38
1:25:46
1:22:53
1:20:00
Cycle Time [hrs]
1
4
7
10
13
16
19
22
25
28
31
34
37
40
43
Weeks
FIFO
X-Theor
SPT
BMI

Experiment #4				
Snap	**FIFO**	**X-Theor**	**SPT**	**BMI**
Average	**1:33:48**	**1:31:21**	**1:30:10**	**1:28:18**
1	1:31:58	1:37:46	1:29:04	1:35:43
2	1:33:41	1:35:47	1:30:56	1:33:53
3	1:31:39	1:33:13	1:31:16	1:31:13
4	1:33:02	1:35:11	1:35:26	1:33:16
5	1:30:44	1:30:29	1:29:59	1:28:20
6	1:30:43	1:30:30	1:29:41	1:28:31
7	1:30:17	1:29:49	1:28:48	1:27:32
8	1:29:23	1:28:20	1:27:25	1:26:11
9	1:29:27	1:26:15	1:25:27	1:23:59
10	1:28:49	1:26:46	1:25:43	1:23:44
11	1:29:45	1:28:53	1:27:03	1:25:54
12	1:34:07	1:32:25	1:32:35	1:28:28
13	1:36:52	1:32:57	1:33:04	1:28:39
14	1:34:49	1:32:05	1:32:09	1:28:00
15	1:34:26	1:32:05	1:31:58	1:27:58
16	1:33:59	1:31:30	1:31:13	1:27:34
17	1:33:06	1:30:44	1:30:24	1:27:00
18	1:31:10	1:30:55	1:30:04	1:27:12
19	1:32:53	1:31:02	1:30:36	1:27:23
20	1:32:31	1:30:18	1:29:53	1:26:48
21	1:33:51	1:31:12	1:30:54	1:26:57
22	1:34:14	1:31:14	1:30:53	1:27:08
23	1:33:24	1:30:50	1:30:29	1:26:54
24	1:33:34	1:31:11	1:30:21	1:26:57
25	1:34:56	1:32:47	1:32:49	1:28:15
26	1:35:21	1:33:29	1:33:49	1:27:48
27	1:35:59	1:34:11	1:34:36	1:29:18
28	1:36:17	1:34:17	1:34:52	1:29:40
29	1:36:15	1:34:03	1:34:39	1:29:58
30	1:35:40	1:33:40	1:34:20	1:29:49
31	1:34:54	1:32:58	1:33:33	1:29:10
32	1:34:18	1:32:27	1:33:04	1:28:40
33	1:33:59	1:32:09	1:32:42	1:28:26
34	1:34:32	1:32:44	1:33:06	1:28:49
35	1:34:06	1:32:23	1:32:46	1:28:32
36	1:34:28	1:32:48	1:33:06	1:28:47
37	1:35:18	1:33:52	1:33:49	1:29:30
38	1:35:12	1:33:44	1:33:44	1:29:36
39	1:34:59	1:33:35	1:33:26	1:29:22
40	1:37:06	1:35:38	1:35:29	1:30:40
41	1:39:15	1:37:33	1:37:52	1:32:01
42	1:38:43	1:39:05	1:38:59	1:32:13

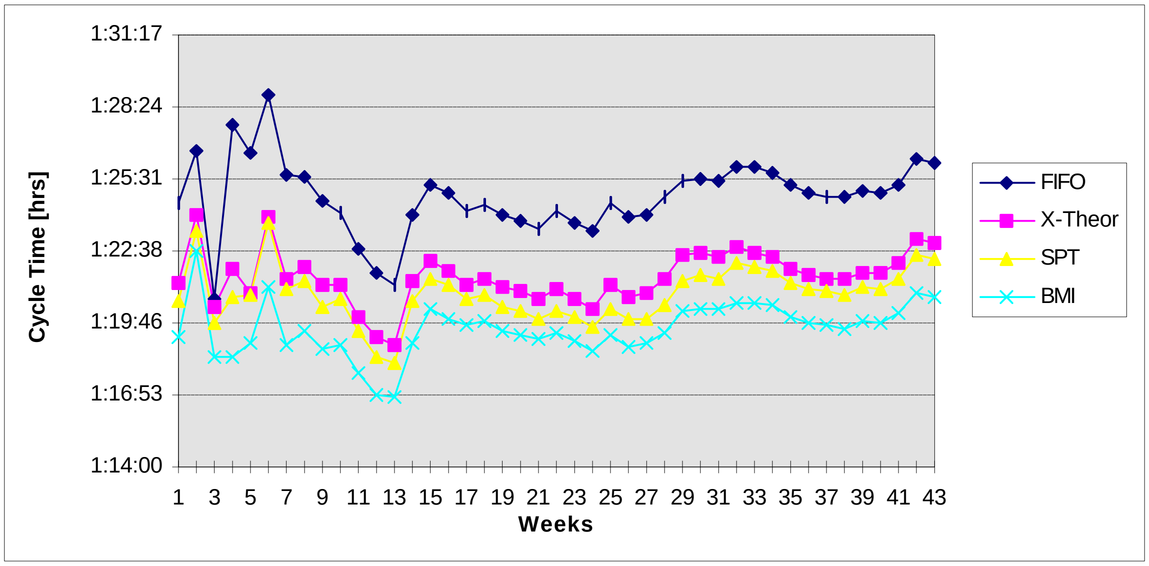


1:40:10
1:37:17
1:34:24
1:31:31
1:28:38
1:25:46
1:22:53
1:20:00
Cycle Time [hrs]
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41
43
Weeks
FIFO
X-Theor
SPT
BMI

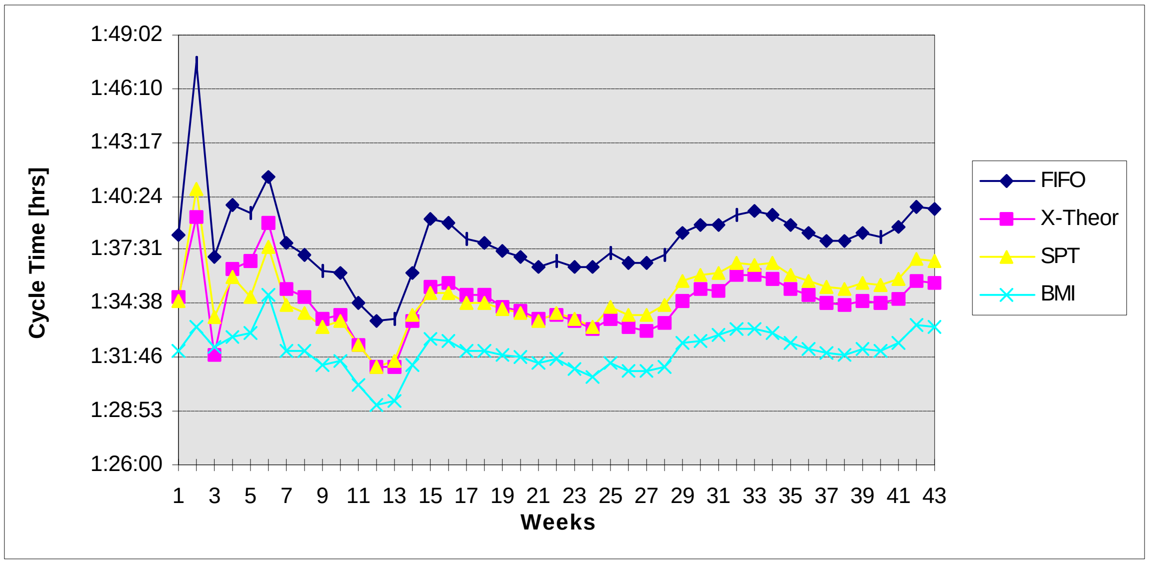
Experiment #6				
Snap	**FIFO**	**X-Theor**	**SPT**	**BMI**
Average	**1:22:51**	**1:21:28**	**1:19:05**	**1:18:16**
1	1:21:47	1:20:11	1:18:26	1:16:11
2	1:19:52	1:18:34	1:16:00	1:19:05
3	1:23:26	1:21:02	1:17:35	1:20:00
4	1:22:24	1:19:45	1:17:23	1:18:19
5	1:24:48	1:22:18	1:20:13	1:19:28
6	1:22:22	1:20:12	1:17:53	1:17:33
7	1:22:35	1:20:48	1:18:49	1:18:24
8	1:21:49	1:20:15	1:17:43	1:17:47
9	1:21:11	1:20:17	1:17:41	1:17:39
10	1:20:04	1:19:06	1:16:41	1:16:39
11	1:20:19	1:20:21	1:16:49	1:15:51
12	1:19:26	1:18:13	1:17:45	1:15:56
13	1:22:31	1:21:04	1:17:48	1:17:50
14	1:23:16	1:21:39	1:18:52	1:18:20
15	1:23:16	1:21:40	1:18:47	1:18:22
16	1:22:45	1:21:12	1:18:26	1:18:03
17	1:22:45	1:21:15	1:18:29	1:18:03
18	1:22:29	1:20:58	1:18:08	1:17:54
19	1:22:22	1:20:49	1:18:13	1:17:48
20	1:22:01	1:20:35	1:18:00	1:17:35
21	1:22:27	1:20:50	1:18:06	1:17:49
22	1:22:06	1:20:29	1:17:51	1:17:32
23	1:21:55	1:20:21	1:17:59	1:17:19
24	1:22:41	1:21:23	1:18:32	1:17:52
25	1:22:12	1:21:04	1:18:11	1:17:28
26	1:22:15	1:21:01	1:18:14	1:17:30
27	1:22:44	1:21:32	1:18:35	1:18:03
28	1:24:09	1:22:41	1:20:14	1:19:02
29	1:24:24	1:22:56	1:20:32	1:19:01
30	1:24:34	1:23:10	1:20:46	1:19:21
31	1:23:09	1:23:40	1:21:53	1:19:51
32	1:23:03	1:23:39	1:20:50	1:19:51
33	1:23:49	1:22:28	1:21:35	1:19:36
34	1:24:17	1:22:56	1:21:06	1:19:09
35	1:23:59	1:22:39	1:20:47	1:18:53
36	1:23:49	1:22:27	1:20:38	1:18:49
37	1:23:50	1:22:24	1:20:27	1:18:36
38	1:24:04	1:22:39	1:20:44	1:18:55
39	1:23:55	1:22:32	1:20:34	1:18:49
40	1:24:26	1:22:59	1:21:02	1:19:23
41	1:25:13	1:23:42	1:21:31	1:19:00
42	1:25:02	1:23:31	1:21:27	1:18:49


1:25:31
1:24:05
1:22:38
1:21:12
1:19:46
1:18:19
1:16:53
1:15:26
1:14:00
Cycle Time [hrs]
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41
43
Weeks
FIFO
X-Theor
SPT
BMI

Experiment #8				
Snap	**FIFO**	**X-Theor**	**SPT**	**BMI**
Average	**1:24:31**	**1:21:20**	**1:20:41**	**1:19:11**
1	1:26:37	1:24:04	1:23:28	1:22:37
2	1:20:42	1:20:23	1:19:46	1:18:25
3	1:27:40	1:21:57	1:20:48	1:18:23
4	1:26:33	1:21:00	1:20:54	1:18:57
5	1:28:52	1:24:02	1:23:44	1:21:11
6	1:25:41	1:21:31	1:21:06	1:18:52
7	1:25:37	1:22:01	1:21:27	1:19:27
8	1:24:40	1:21:16	1:20:24	1:18:45
9	1:24:12	1:21:19	1:20:43	1:18:53
10	1:22:42	1:20:02	1:19:26	1:17:47
11	1:21:44	1:19:10	1:18:23	1:16:54
12	1:21:19	1:18:55	1:18:11	1:16:47
13	1:24:05	1:21:28	1:20:40	1:18:58
14	1:25:18	1:22:15	1:21:33	1:20:19
15	1:24:59	1:21:50	1:21:15	1:19:57
16	1:24:16	1:21:17	1:20:44	1:19:40
17	1:24:29	1:21:31	1:20:53	1:19:50
18	1:24:03	1:21:12	1:20:23	1:19:26
19	1:23:49	1:21:02	1:20:13	1:19:18
20	1:23:30	1:20:45	1:19:56	1:19:05
21	1:24:13	1:21:07	1:20:15	1:19:24
22	1:23:46	1:20:42	1:19:58	1:19:04
23	1:23:26	1:20:21	1:19:37	1:18:40
24	1:24:33	1:21:18	1:20:21	1:19:15
25	1:24:00	1:20:50	1:19:53	1:18:46
26	1:24:06	1:20:57	1:19:53	1:18:58
27	1:24:46	1:21:32	1:20:30	1:19:23
28	1:25:26	1:22:31	1:21:26	1:20:15
29	1:25:29	1:22:32	1:21:39	1:20:17
30	1:25:26	1:22:26	1:21:30	1:20:18
31	1:26:02	1:22:47	1:22:10	1:20:33
32	1:26:02	1:22:33	1:22:00	1:20:36
33	1:25:47	1:22:25	1:21:51	1:20:28
34	1:25:16	1:21:57	1:21:23	1:20:02
35	1:24:57	1:21:39	1:21:08	1:19:45
36	1:24:46	1:21:33	1:21:01	1:19:39
37	1:24:46	1:21:32	1:20:52	1:19:33
38	1:25:02	1:21:48	1:21:11	1:19:50
39	1:24:59	1:21:44	1:21:05	1:19:47
40	1:25:18	1:22:12	1:21:33	1:20:10
41	1:26:21	1:23:08	1:22:29	1:20:56
42	1:26:10	1:22:57	1:22:21	1:20:47


1:31:17
1:28:24
1:25:31
1:22:38
1:19:46
1:16:53
1:14:00
Cycle Time [hrs]
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41
43
Weeks
FIFO
X-Theor
SPT
BMI

Experiment #10				
Snap	**FIFO**	**X-Theor**	**SPT**	**BMI**
Average	**1:38:21**	**1:35:03**	**1:34:49**	**1:32:08**
1	1:47:34	1:39:17	1:40:47	1:33:23
2	1:37:08	1:31:56	1:33:53	1:32:19
3	1:39:57	1:36:30	1:36:06	1:32:50
4	1:39:33	1:36:58	1:35:02	1:33:07
5	1:41:26	1:38:55	1:37:39	1:35:08
6	1:37:51	1:35:23	1:34:32	1:32:07
7	1:37:17	1:35:00	1:34:09	1:32:06
8	1:36:23	1:33:47	1:33:23	1:31:20
9	1:36:20	1:34:05	1:33:42	1:31:33
10	1:34:43	1:32:27	1:32:24	1:30:20
11	1:33:43	1:31:12	1:31:17	1:29:14
12	1:33:50	1:31:12	1:31:32	1:29:24
13	1:36:19	1:33:42	1:34:02	1:31:19
14	1:39:10	1:35:32	1:35:14	1:32:42
15	1:38:55	1:35:45	1:35:12	1:32:41
16	1:38:04	1:35:09	1:34:38	1:32:08
17	1:37:54	1:35:04	1:34:39	1:32:08
18	1:37:29	1:34:30	1:34:21	1:31:55
19	1:37:10	1:34:16	1:34:08	1:31:49
20	1:36:37	1:33:50	1:33:43	1:31:27
21	1:36:57	1:34:01	1:34:08	1:31:40
22	1:36:38	1:33:44	1:33:48	1:31:11
23	1:36:34	1:33:16	1:33:26	1:30:45
24	1:37:20	1:33:49	1:34:26	1:31:31
25	1:36:48	1:33:21	1:34:03	1:31:03
26	1:36:48	1:33:10	1:34:01	1:30:59
27	1:37:12	1:33:36	1:34:37	1:31:17
28	1:38:23	1:34:45	1:35:50	1:32:32
29	1:38:50	1:35:24	1:36:12	1:32:39
30	1:38:49	1:35:22	1:36:19	1:32:56
31	1:39:25	1:36:14	1:36:47	1:33:16
32	1:39:34	1:36:09	1:36:42	1:33:18
33	1:39:26	1:35:58	1:36:52	1:33:06
34	1:38:49	1:35:23	1:36:14	1:32:34
35	1:38:27	1:35:04	1:35:53	1:32:15
36	1:38:03	1:34:42	1:35:30	1:31:58
37	1:38:03	1:34:34	1:35:27	1:31:54
38	1:38:24	1:34:48	1:35:48	1:32:13
39	1:38:13	1:34:38	1:35:38	1:32:04
40	1:38:44	1:34:55	1:36:01	1:32:33
41	1:39:50	1:35:49	1:37:04	1:33:29
42	1:39:43	1:35:45	1:36:56	1:33:25


Cycle Time [hrs]
1:49:02
1:46:10
1:43:17
1:40:24
1:37:31
1:34:38
1:31:46
1:28:53
1:26:00
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41
43
Weeks
FIFO
X-Theor
SPT
BMI

VITA

Adar Kalir received his B.Sc. (1990) and M.Sc. (1995) degrees in Industrial Engineering and Management from Tel-Aviv University, Israel. His Master's Thesis has been recognized as an outstanding research in 1995 by the Institute of Industrial Engineers in Israel. He has held a Production Manager position with Telrad Telecommunications and served as a Project Manager in an IE consulting firm specializing in semiconductor manufacturing, prior to pursuing his Ph.D. at Virginia Tech. As part of his course of study, he has also been on a coop program with Intel. He is a member of the Phi-Kappa-Phi honorary society, and has won awards for excellence in research, the most recent one being a first place award for research in 1998, awarded by the Decision Sciences Institute. He has represented Virginia Tech in the prestigious National Doctoral Colloquium for Industrial Engineering and Operations Research, held in 1998 in Seattle, Washington. He has published in various journals, such as the IIE transactions and the International Journal of Production Research. His research interests are in production management and control, capacity planning, design and control of flexible manufacturing systems, and in the applications of sequencing and scheduling theory.