



**CUSTOMER ORDER SCHEDULING WITH LOT STREAMING TO
MINIMIZE THE TOTAL ORDER COMPLETION TIME IN A TWO-
MACHINE FLOW SHOP**



GÜNCE YOZGAT

FEBRUARY 2018

**CUSTOMER ORDER SCHEDULING WITH LOT STREAMING TO
MINIMIZE THE TOTAL COMPLETION TIME IN
A TWO-MACHINE FLOW SHOP**

**A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ÇANKAYA UNIVERSITY**

**BY
GÜNCE YOZGAT**

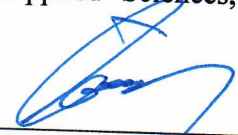
**IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
FOR THE DEGREE OF MASTER OF SCIENCE
IN
THE DEPARTMENT OF INDUSTRIAL ENGINEERING**

FEBRUARY 2018

Title of the Thesis : **Customer Order Scheduling with Lot Streaming to Minimize the Total Completion Time in a Two-Machine Flow Shop**

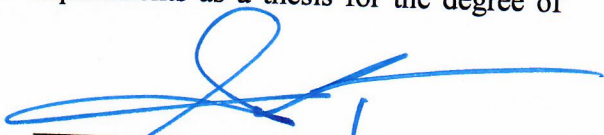
Submitted by **Günce YOZGAT**

Approval of the Graduate School of Natural and Applied Sciences, Çankaya University



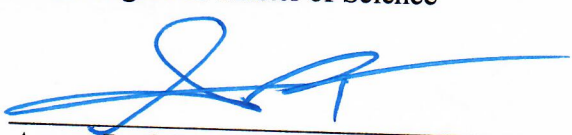
Prof. Dr. Can ÇOGUN
Director

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



Assoc. Prof. Dr. Ferda Can ÇETİNKAYA
Head of Department

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



Assoc. Prof. Dr. Ferda Can ÇETİNKAYA
Supervisor

Examination Date: 05.02.2018

Examining Committee Members

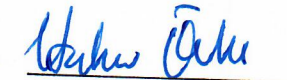
Assoc. Prof. Dr. Sedef MERAL (METU)



Assoc. Prof. Dr. Ferda Can ÇETİNKAYA (Çankaya Univ.)




Asst. Prof. Dr. Hakan ÖZAKTAŞ (Çankaya Univ.)



STATEMENT OF NON PLAGIARISM

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : Günce YÖZGAT

Signature : 

Date : 05.03.2018

ABSTRACT

CUSTOMER ORDER SCHEDULING WITH LOT STREAMING TO MINIMIZE THE TOTAL COMPLETION TIME IN A TWO-MACHINE FLOW SHOP

YOZGAT, Günce

M.Sc., Department of Industrial Engineering

Supervisor: Assoc. Prof. Dr. Ferda Can ÇETİNKAYA

February 2018, 55pages

In this study, we consider a customer order scheduling problem in which each customer can request a variety of products (also called jobs) in an order. All products are processed on a two-machine flow shop in which each product has one operation on each machine, and all products are first processed by machine 1 and then by machine 2. Each customer order is delivered to the customer when the processing of all products in the customer order is completed. Thus, the completion time of the job subplot processed as the last product in a customer order defines the completion time of the customer order. Our goal is to find a sequence of the job lots as well as the sequences of the sublots in each job so that the total completion time, which is the sum of the completion times of the customer orders, is minimized. We develop a mixed integer linear programming model capable of solving small-sized problem instances optimally, and propose a tabu-search based heuristic algorithm that obtains optimal and near-optimal solutions for medium and large-sized problem instances. The results of our computational experiments performed to evaluate the performance of our solution approaches in terms of both quality and time show that the proposed heuristic algorithm finds optimal or near-optimal solutions in very short time.

Keywords: Customer order scheduling; lot streaming; two-machine flow shop; total completion time; mixed integer linear programming; tabu-search

ÖZ

İKİ MAKİNALI AKIŞ TİPİ ÜRETİM HATTINDA MÜŞTERİ SİPARİŞLERİNİN TAMAMLANMA ZAMANLARI TOPLAMININ ENKÜÇÜKLENEREK KAFİLE KAYDIRMALI OLARAK ÇİZELGELENMESİ

YOZGAT, Günce

Yüksek Lisans, Endüstri Mühendisliği

Anabilim Dalı Tez Yöneticisi: Doç. Dr. Ferda Can ÇETİNKAYA

Şubat 2018, 55sayfa

Bu çalışmada, çeşitli ürünleri (işleri) içerebilen müşteri siparişlerini çizelgeleme problemi ele alınacaktır. Bir ürün işlenirken, o ürüne ait her müşteri siparişi alt kabileleri (bir ürünün özdeş grupları) olarak işlenir ve aynı ürünün tüm alt kabileleri aynı makinada aralıksız olarak işlenir ve aynı ürünün diğer alt kabileleri makina 1' de işlenirken, işlenen alt kabileler makina 1'den makina 2'ye aktarılır. Bu durum, aynı ürünün alt kabilelerinin birbirine karışmasına izin vermeksizin iki operasyonun örtüşmesi anlamına gelir (yani bir ürünün ilk alt kabileleri bir makinaya ulaştığında, o ürüne ait tüm alt kabileler tamamlanana kadar başka ürünlerin alt kabileleri bu makinaya atanamaz). Müşteri siparişinde yer alan tüm ürünlerin üretimi tamamlandıktan sonra müşterinin siparişi teslim edilir. Bir müşteri siparişinde son ürün olarak işlem gören son alt kabilenin tamamlanma zamanı, müşteri siparişinin tamamlanma zamanıdır. Amacımız, müşteri siparişlerinin tamamlanma zamanlarının toplamını enküçükleyen iş kabilelerinin sırasını ve her iş kabilesindeki alt kabilelerin sırasını bulmaktır. Küçük ölçekli problemleri optimal olarak çözebilen bir karışık tamsayı doğrusal programlama modeli ile büyük ve orta ölçekli problemler için optimal veya optimale yakın sonuçlar verebilen tabu arama esaslı sezgisel bir algoritma geliştirdik. Çözüm yöntemlerinin süre ve kalite açısından değerlendirilmesi

için yapılan deneylerin sonuçları, önerilen sezgisel algoritmanın çok kısa sürede optimal ya da optimale yakın sonuçlar bulduğunu göstermektedir.

Anahtar Kelimeler: Müşteri siparişi çizelgeleme; kabile kaydırma; iki makinalı akış tipli atölye; toplam tamamlanma zamanı; karışık tamsayı doğrusal programlama; tabu arama



ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor Assoc. Prof. Dr. Ferda Can ÇETİNKAYA for his endless support, suggestions, guidance and encouragement during this thesis.

I would like to express my appreciation to my family for their sincere love, support and positiveness.

Lastly, I would like to give special thanks to my husband for his endless inexpressible support and motivation.



TABLE OF CONTENTS

STATEMENT OF NON PLAGIARISM.....	iii
ABSTRACT.....	iv
ÖZ.....	v
ACKNOWLEDGEMENTS.....	vii
TABLE OF CONTENTS.....	viii
LIST OF FIGURES.....	x
LIST OF TABLES.....	xi
LIST OF ABBREVIATIONS.....	xii
CHAPTERS :	
1. INTRODUCTION.....	1
2. PROBLEM DEFINITION, COMPLEXITY AND PRELIMINARIES.....	4
2.1 Problem Definition.....	4
2.2 Problem Complexity.....	5
2.3 Preliminaries.....	6
3. LITERATURE REVIEW.....	10
3.1 Literature Review for the Customer Order Scheduling Problems.....	10
3.1.1 Single Machine Problems.....	10
3.1.2 Parallel Machine Problems.....	11
3.1.3 Open Shop and Job Shop Problems.....	13
3.2 Literature Review for Multi-Job Lot Streaming Problems.....	13
4. SOLUTION APPROACHES: MATHEMATICAL PROGRAMMING MODEL AND A TABU-SEARCH BASED PROPOSED HEURISTIC ALGORITHM.....	17
4.1 Mathematical Programming Model.....	17
4.2 Proposed Tabu-search Based Heuristic Algorithm.....	21

4.3 Numerical Example.....	24
5. COMPUTATIONAL EXPERIMENTS	36
5.1 Computational Settings for the Test Problems.....	36
5.2 Performance Measures	37
5.3 Discussions of the Results.....	38
5.3.1 Setup Case.....	38
5.3.1.1 Performance of the MILP model	38
5.3.1.2 Performance of the heuristic algorithm.....	40
5.3.2 No Setup Case	42
5.3.2.1 Performance of the MILP model	42
5.3.2.2 Performance of the heuristic algorithm.....	43
5.4 Comparison of the Setup and No Setup Cases	46
5.4.1 Comparison of the Setup and No Setup Cases for the MILP Performance	46
5.4.2 Comparison of the Setup and No Setup Cases for the Heuristic Algorithm Performance.....	48
6. CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS	50
REFERENCES.....	52

LIST OF FIGURES

Figure 1	Gantt Chart for the Example Problem with Three Customers and Two Products	3
Figure 2	Run-in Times of the Jobs.....	9
Figure 3	Gantt Chart for the Customer Order O_1	26
Figure 4	Tabu-search Iteration-1.....	34
Figure 5	Tabu-search Iteration-2.....	34
Figure 6	Gantt Chart of the Schedule Obtained for the Numerical Example	35

LIST OF TABLES

Table 1 Data Set for the Run-in Time Numerical Example Illustrating the Run-in Times.....	8
Table 2 Calculation of the V -Value.....	21
Table 3 Data Set for the Numerical Example.....	25
Table 4 Total Order Completion Times for Two Partial Sequences	28
Table 5 Total Order Completion Times for Three Partial Sequences	29
Table 6 Total Order Completion Time for Four Partial Sequences	30
Table 7 Total Order Completion Time for Five Partial Sequences.....	32
Table 8 Performance of the MILP Model for the Setup Case.....	39
Table 9 Average Percent Deviations of the Heuristic Algorithm from the Optimal and Best Integer Solution for the Setup Case	40
Table 10 Performance of the MILP Model for the No Setup Case	43
Table 11 Average Percent Deviations of the Heuristic Algorithm from the Optimal and Best Integer Solution for the No Setup Case	46
Table 12 Comparison of the Setup and No Setup Cases for MILP Performance for 5 orders	47
Table 13 Comparison of the Setup and No Setup Cases for MILP Performance for 10 orders	47
Table 14 Average and Maximum Percent Deviations for Setup and No Setup Cases	49

LIST OF ABBREVIATIONS

COS	Customer Order Scheduling
GAMS	General Algebraic Modelling System
MILP	Mixed Integer Linear Programming
NP	Non-Deterministic Polynomial
LP	Linear Programming
LS	Lot Streaming



CHAPTER 1

INTRODUCTION

In today's world customer satisfaction has great importance for all of the companies. This situation promotes the companies to select make-to-order (order-based) manufacturing strategy. In make-to-order manufacturing environments, scheduling is usually referred to as *customer order scheduling* (COS). Manufacturing operations start after a customer order is placed. To provide customer satisfaction, the questions “when to deliver orders”, “which quantities should be produced” and “which sequence of jobs should be produced” must be answered. Although answering these questions are difficult enough, product variety and constraints like order lead times increase the complexity of customer order scheduling problem.

Customer orders can have multiple product types with different quantities. Also, each product type can be produced by using different operations and different number of operations. These operations can be conducted on different machines. Moreover, each product can be manufactured by using different machines which may need setups, release times or other operations before starting the manufacturing of the products. In the flow shop environment, the operations follow each other on different types of machines. If the flow shop environment allows changes in the product (job) sequence on all machines, it is called *non-permutation flow shop* environment. Otherwise, the product (job) sequence is same on all machines, and this type of flow shop is called as *permutation flow shop*. In this study, we deal with two-machine permutation flow shop environment.

Job lots having different orders are split in to sublots. This technique of splitting the jobs into sublots and processing these different sublots simultaneously over different machines is called *lot streaming* (LS). Lot streaming has a lot of advantages in make-to-order manufacturing environment to improve delivery times if setup times are

significantly large. Lot streaming problems are divided into two groups that are single-lot and multi-lot problems. While single-lot problems determine the number of sublots and their sizes, multi-lot problems determine the number of sublots, subplot sizes and sequence of the sublots. In this study, we deal with the multi-lot problems with customer order scheduling.

The problem considered in this thesis is the scheduling of K customer orders in which each customer gives order for several types of products (or jobs). N jobs are processed firstly on the first machine and then on the second machine in a two-machine flow shop environment. A customer order can only be delivered when all jobs of that customer order are completed on the second machine.

In this problem defined above, decisions are made as to determine the sequence of the products (jobs) as well as the sequence of customer orders in each job to minimize the total completion time of the customer orders. Throughout this study, we will call the sequences of jobs and customer orders as *job sequence* and *subplot sequence*, respectively. Furthermore, a schedule in which both job and subplot sequences are specified will be called a *schedule*.

Before we proceed with our analysis, it seems appropriate to illustrate the problem by a numerical example. Consider a simple instance of the problem in which there are three customer orders and two products (jobs). Customers 1 and 2 order 100 units of product 1. Customers 1, 2 and 3 have ordered 100, 50 and 25 units of product 2, respectively. Setup and unit processing times for operations 1 and 2 of Job 1 are (10; 1) and (10; 2), respectively. Similarly, Setup and unit processing times for operations 1 and 2 of Job 2 are (10; 2) and (10; 1), respectively. As it is illustrated in Fig. 1, the optimal job (product) sequence with the optimal subplot (customer order) sequences in each job is $J_2(O_3 - O_2 - O_1) - J_1(O_1 - O_2)$. The completion times of the orders 1, 2 and 3 are 680, 880 and 95 time units, respectively, and the total completion time of the orders is $680 + 880 + 95 = 1655$ time units.

The contribution of our study in this thesis is threefold. First, to the best of our knowledge, there is no study that considers the flow shops with both customer order scheduling and lot streaming. Second, we formulate the total completion time minimization problem as a mixed integer linear programming (MILP) model to solve

the problem under consideration optimally. Third, our proposed tabu-search based heuristic algorithm for solving the problem is straightforward and easy to implement for finding optimal and near-optimal solutions for medium and large-sized problem instances in which a solution cannot be obtained by solving the MILP model.

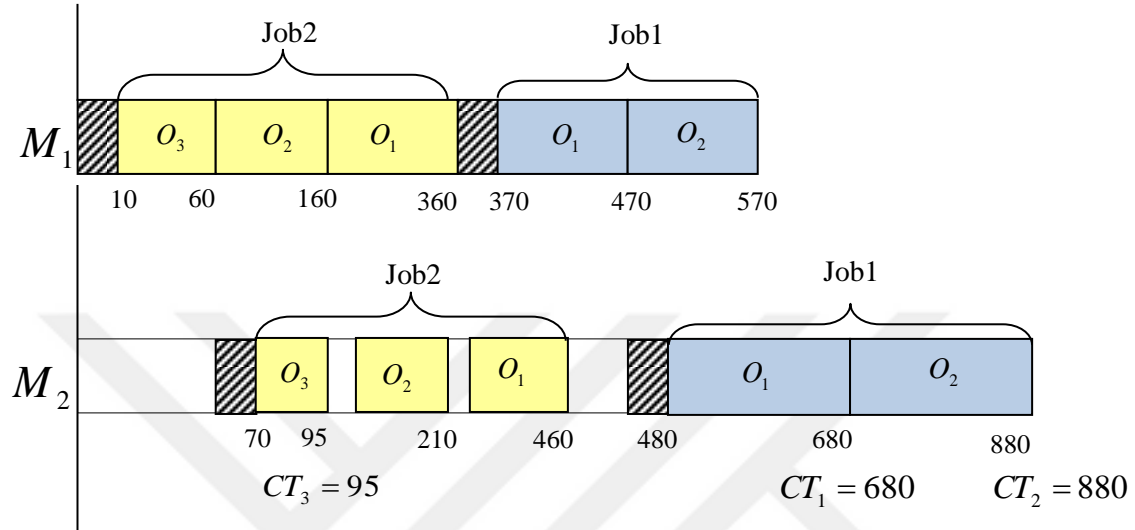


Figure 1 Gantt Chart for the Example Problem with Three Customers and Two Products

The remainder of this report is organized as follows. Chapter 2 defines the problem and provides the problem complexity and some structural properties of the optimal schedule for the problem under consideration. Chapter 3 provides a review of the most relevant works to our study on customer order scheduling and lot streaming. Chapter 4 proposes a mixed-integer linear programming model and a tabu-search based heuristic algorithm. We also provide a numerical example for better understanding of the proposed heuristic algorithm. The computational tests to evaluate the performance of the mathematical model and the proposed heuristic algorithm are given in Chapter 5. Finally, our main findings and several directions for future research are discussed in Chapter 6.

CHAPTER 2

PROBLEM DEFINITION, COMPLEXITY AND PRELIMINARIES

In this chapter, the problem under consideration is defined, its complexity is discussed, and some properties of the optimal schedule, which will be used in the development of the heuristic algorithm, are provided.

2.1. Problem Definition

Consider a scheduling problem of K customer orders ($k = 1, 2, \dots, K$) in which each customer can request a variety of products (also called jobs) in an order. There are N products ($j = 1, 2, \dots, N$) to be processed on a two-machine flow shop in which each product has one operation on each machine and all products are first processed by machine 1 and then by machine 2.

Each customer order k has $D_{k,j}$ units of identical items of product j , where this quantity is called the product subplot (or job subplot) size. While processing a product, each customer order for that product is processed as a subplot (a batch of identical items of a product), and all sublots of the same product must be processed continuously by the same machine, and each processed subplot of the same product is transferred from machine 1 to machine 2 for the second operation, while other sublots of the same product are processed on machine 1. This means that overlapping of the two operations on the same product through the use of sublots (i.e., lot streaming) is allowed without intermingling the sublots of other products (i.e., once the first subplot of a product arrives at a machine, the other sublots of different products cannot be assigned to this machine until all of the sublots are processed). Unit processing time of the job j on machine m ($m = 1, 2$) is $p_{j,m}$, and a sequence independent attached setup time $t_{j,m}$ is needed to set up the tools, jigs, fixtures, etc.

before processing the first subplot of job j on machine m . In the case of *attached setups*, the setup on machine 2 cannot start until the first subplot is available on machine 2. Besides, the following assumptions are considered:

- Preemption of the sublots is not allowed; i.e., any subplot cannot be interrupted until the completion of its operation.
- No setup is necessary between successive sublots of the same product.
- All customer orders are available for processing at the same time, say time 0.
- Each machine is available at time zero and remains continuously available.
- Each machine can process at most one subplot at a time and each subplot can be processed on only one machine at any given time.
- Sufficient storage space exists to stock the processed sublots on machine 1.
- Transportation times between machines are considered to be negligible.

Each customer order is delivered to the customer when the processing of all products in the customer order is completed. Thus, the completion time of the job subplot processed as the last product in a customer order defines the completion time of the customer order. Our goal is to find a sequence of the jobs as well as the sequences of the sublots in each job so that the total completion time, which is the sum of the completion times of the customer orders, is minimized to increase the customer satisfaction.

2.2. Problem Complexity

Theorem 1 *Customer order scheduling problem with lot streaming to minimize the total completion time in a two-machine flow shop is NP-hard in the strong sense.*

Proof Consider a special case of the problem, where (a) the number of customer orders is equivalent to the number of products (jobs), i.e., $K = N$, (b) each customer gives an order with exactly one product different from the products in the other customer orders, i.e., $D_{j,k} = D_j = 1$ for $\forall j$, and (c) the setup times are omitted, i.e., $t_{j,m} = 0$ for $\forall j, m$. This special case is equivalent to the total completion time minimization problem in the classical two-machine flow shop without lot streaming $F2 // \sum C_j$, which has been proven to be NP-hard in the strong sense by Gonzalez

and Sahni (1978). Hence, the problem under consideration is also NP-hard in the strong sense. \square

2.3. Preliminaries

Given the complexity of the problem under consideration, it is desirable to develop a heuristic algorithm to obtain optimal and/or near-optimal solutions for large scale problem instances in reasonable CPU times. Thus, we now give some definitions and theorems to derive the structural properties, which will be used in the development of a heuristic algorithm, of the optimal solution for the problem under consideration.

Definition 1 (Smith et al., 1975) The M -machine N -job flowshop is called an *ordered flowshop* if the following two properties are satisfied:

- (i) If a particular job has a smaller processing time on any machine than does a second job on the same machine, this implies that the processing time of this first job is less than or equal to the processing time of the second job on all corresponding machines.
- (ii) If a job has its r th smallest processing time on some machine m , $m = 1, 2, \dots, M$, this implies that every other job will have its r th smallest processing time on the same machine m where $r = 1, 2, \dots, M$.

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

Lemma 1 (Panwalker and Khan, 1979) *In the optimal solution of the ordered flow shop problem to minimize the total completion time, jobs are arranged in nondecreasing order of their processing times.*

Çetinkaya and Gupta (1994) gives the following result for the single-job lot streaming problem in the M -machine flowshop.

Lemma 2 (Çetinkaya and Gupta, 1994) *The single-job lot streaming problem with consistent sublots satisfies the characteristics of the ordered flowshops.*

Proof Proof is obvious from two facts. First, if a subplot has the k th smallest processing time on a machine m across all sublots, then it has the k th smallest processing time on every other machine due to consistency. Second, if a subplot has

its k th smallest processing time on machine m , then every other subplot has the k th smallest processing time on machine m due to proportionality of processing times with subplot sizes. \square

The following theorem describes the optimal schedule of the problem under consideration for the single-job case.

Theorem 1 *For the problem under consideration, there exists an optimal schedule in which the customer orders (sublots) are processed in non-decreasing order of their subplot sizes if each customer gives an order with exactly one product that is the same product (job) in all customer orders.*

Proof If each customer gives an order with exactly one product which is the same product (job) in all customer orders, then the problem reduces to the single-job lot streaming problem to minimize the total completion time of the customer orders. From Lemmas 1 and 2, it is clear that the customer orders (sublots) are processed in non-decreasing order of their subplot sizes. \square

The following theorem describes the schedule of the sublots of the last job in the last position of the optimal schedule for the problem under consideration.

Theorem 2 *For the problem under consideration, there exists an optimal schedule in which all sublots of the last job in the job sequence are processed in non-decreasing order of their subplot sizes.*

Proof Note that the sum of the completion times for the customer orders (sublots) having no demand for the product (job) processed in the last position of the job sequence does not depend on the sequence of the sublots of the job processed as the last in the job sequence. Thus, the problem of finding the sequence of the sublots of the job in the last position of the job sequence can be considered as the single-job case as given in Theorem 1, and all sublots of the last job in the job sequence are sequenced in non-decreasing order of their subplot sizes. \square

Definition 2 (Run-in Time) Run-in time RI_j of the job j is the time that elapses between the starts of the setups for job j on machines 1 and 2 in the flow shop environment, and is calculated as:

$$RI_j = t_{j,1} + p_{j,1} s_{[1],j},$$

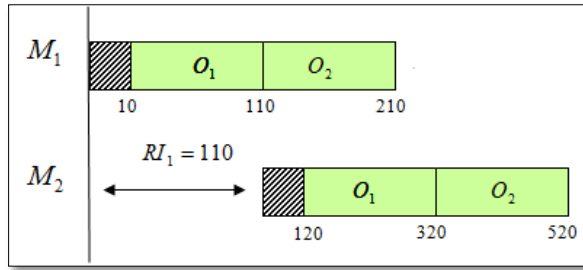
where $S_{[1],j}$ is the size of the first subplot (customer order) in job j .

To illustrate the run-in times, we consider a problem instance in which there are four products (jobs) and five orders (sublots). Unit processing times ($p_{j,m}$), sequence independent setup time ($t_{j,m}$) and requirement of product j for customer order k ($D_{k,j}$) are given in Table 1.

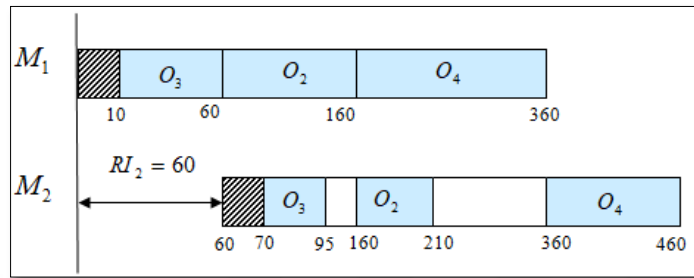
Table 1 Data Set for the Run-in Time Numerical Example Illustrating the Run-in Times

		$D_{k,j}$					$p_{j,m}$		$t_{j,m}$	
Orders (O_k)	Products (j)	O_1	O_2	O_3	O_4	O_5	m_1	m_2	m_1	m_2
		J_1	100	100	-	-	-	1	2	10
J_2	-	50	25	100	-	2	1	10	10	
J_3	25	-	-	50	100	1	3	10	10	
J_4	50	50	-	100	-	4	1	10	10	

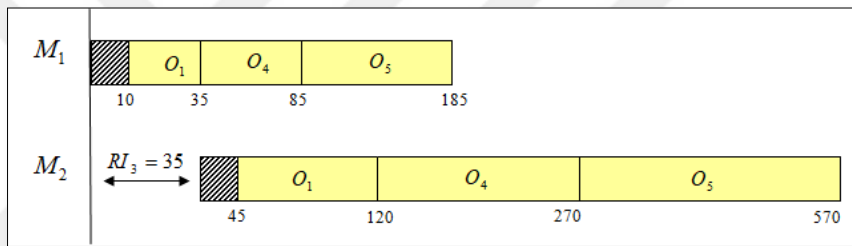
Run-in times for all jobs are illustrated by the Gantt charts in Figure 2.



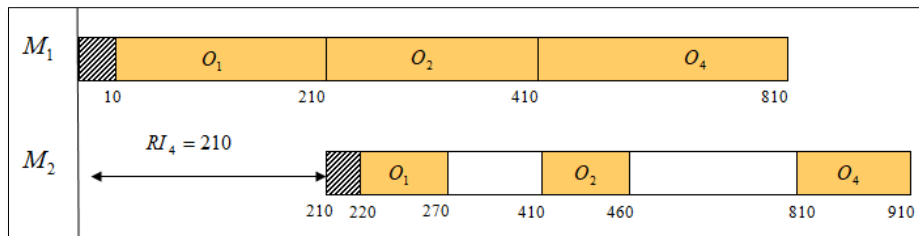
(a) Job J_1



(b) Job J_2



(c) J_3



(d) Job J_4

Figure 2 Run-in Times of the Jobs

CHAPTER 3

LITERATURE REVIEW

In this chapter, the most relevant works to our study on customer order scheduling and lot streaming are reviewed in detail.

3.1. Literature Review for the Customer Order Scheduling Problems

In the literature of machine scheduling, our problem to be studied in this thesis falls in the intersection of two main areas of research: “Customer order scheduling” and “Lot streaming”. In an order-based manufacturing environment, scheduling is usually referred to a customer order scheduling (COS) problem in which there are several customer orders consisting of one or more individual products. The composition of products in each customer order is pre-specified by the customer. Furthermore, different customers may give orders having the same products, and all products in each order are shipped as a group at the same time to the customer (Julien and Magazine (1990) and Ahmadi and Bagchi (1990)). Julien and Magazine (1990) have introduced the COS problem and have used polynomial time algorithm for a given order processing sequence. Also, they have examined the structure of optimal schedules. Ahmadi and Bagchi (1990) have studied the order scheduling problem to minimize the total weighted completion time.

The research on COS problems are scarce in the literature. COS problem has been studied for single-machine, parallel machines, open shops and job shops with various scheduling criteria such as makespan, total completion time, and maximum lateness. In most of these studies, the focus has been on the parallel-machine environments.

3.1.1. Single Machine Problems

Gupta *et al.* (1997) study on non-preemptive single machine bi-criteria scheduling

problems with m customer orders having n varied job types. The initial objective of the study is minimizing the makespan. The objective has the same aim as the objective of minimizing the total set-up time between job classes. Another objective is minimizing the total carrying costs of the customer orders. Carrying cost is thought as length of time interval between completion time of the first and last job in a customer order. Polynomial time algorithm is developed for the two objectives above.

Gerodimos *et al.* (2000) study on the scheduling of the jobs that consist of standard and specific components. Minimizing the number of late jobs is the objective of this study. Because of the NP-hardness, they develop dynamic programming algorithm with pseudo-polynomial time.

Erel and Ghosh (2007) examine COS problem with orders that have varied products from varied product families (when the product families switch, setup time is needed). Minimizing the total order lead time (order completion time) is the aim of this study while finding a production schedule. A dynamic programming is developed based on exact solution algorithm by them.

Hazir *et al.* (2008) study on customer order scheduling problem to minimize average customer order flow time on single machine. Because of the NP hardness problem of COS, four major metaheuristics: simulated annealing, genetic algorithm, ant colony optimization and tabu-search are developed. They observe that tabu-search algorithm and ant colony optimization methods perform better for large problems, while simulated annealing performs better for smaller problems.

3.1.2. Parallel Machines Problems

For parallel machines, Yang and Postner (2005) examine the scheduling of the job batches (customer orders). Their objectives are to minimize the sum of completion times of batches and to minimize the total work-in-process in production shop environment where jobs are dispatched in batches. Two heuristic algorithms are constructed for one and two parallel machines.

Yang (2005) examines the scheduling of a set of jobs (called as customer orders) on two parallel machine environments that is shipped concurrently. Minimizing the

completion time of orders is the aim of this study. He study on several objective functions like; minimizing the last batch completion time (makespan), the maximum batch lateness, the total batch completion time and so on. He analyses the computational complexity with different types of objectives. However, this study is constructed for minimizing the sum of the batch completion times on two parallel identical machines.

Leung *et al.* (2006) consider the orders having various due dates and study on minimizing the maximum lateness objective and the total number of late orders on m non-identical parallel machine environment. To minimize the total number of late orders is the objective of this study and they propose an exact algorithm based on constraint propagation and bounding strategy. Also, Leung *et al.* (2007) study for minimizing the total weighted completion time of the orders on m dedicated parallel machine that can produce only one type of product. This study concentrates on both design and analysis of the efficient heuristics for the cases with and without release dates. Finally, performance analyses are conducted to make comparative analysis and control the heuristic bounds.

Lin and Kononov (2007) examine to minimize the number of late jobs on non-identical parallel machine. A fully polynomial time approximation scheme (FPTAS) is applied for due date case and a heuristic algorithm is designed. The performance of the heuristic algorithm is analyzed to measure the performance for the unweighted case. Also, LP based approximation algorithm is applied for multi-cover problems.

Wang and Cheng (2007) study on the scheduling of customer orders consisting of several different types of jobs. It is assumed that m different facilities and each job can be produced on one type of facility. All jobs are scheduled to minimize the total weighted order completion time that is the aim of this study. A heuristic algorithm is developed since the problem is the NP-hard.

Su *et al.* (2013) examine also COS problem and orders are assumed to be dispatched in batches. This problem is studied on parallel machines that can produce only one job at a time and process simultaneously the jobs from a batch. Minimizing the maximum lateness is the aim of this study. Three heuristic algorithms based on simple scheduling rules are developed and algorithms are compared based on their

effectiveness.

Xu *et al.* (2015) examine COS problem with product type splitting property on unrelated parallel machines. The objective of this study is minimizing the total completion time of the orders. Three heuristic algorithms are developed and compared based on their effectiveness of the lower bound. Also, the effectiveness of the heuristic algorithms is determined with the help of numerical studies.

3.1.3. Open Shop and Job Shop Problems

Wagneur and Sriskandarajah (1993) study scheduling on two-machine open shop environments where each job must be processed on both machines and a job may overlap. The objective is to minimize the total completion time.

Liu (2010) studies on coordinated scheduling of customer orders system that is thought for order based production systems with job shop environments. Also, releasing and dispatching the jobs at station level are thought for that system where several jobs are produced for a determined customer order. The aim of this study is minimizing the customer order flow time that is time between the release of the first job and completion time of the last job of the order.

3.2. Literature Review for Multi-Job Lot Streaming Problems

The concept of lot streaming (LS) was first introduced by Reither (1966) and rediscovered in the late 1980s to early 1990s. In the past three decades, with the increasing interest in just-in-time and optimized production technology philosophies in manufacturing systems, the application of the lot streaming idea in scheduling problems has received considerable attention. Considering the number of job lots, the literature on lot streaming problems can easily be divided into two main categories, one category dealing with a single job lot and the other category addressing the multi-job case, given various job and shop characteristics.

In this study, we provide a brief overview of the lot streaming studies on the multi-job case with an aim to facilitate the proper positioning of our study in the literature. A comprehensive review of scheduling problems with lot streaming concept for both single job lot and multi-job cases can be found in Chang and Chiu (2005) and in

Sarin and Jaiprakash (2007).

Vickson and Alfredsson (1992) consider effects of transfer batches in two and three machine flow shop environment. For the two-machine flow shop makespan minimization problem and the special case of the three-machine flow shop problem, the modified Johnson's Algorithm (1954) is used.

Vickson (1995) examines multiple product lot streaming problem in two machine flow shop environment to minimize the makespan. Job setup times and subplot transfer times are considered in this study. Optimal solution can be obtained with continuous sublots, while a polynomial algorithm is developed for integer valued lot sizes.

Kalir (1999) studies both single and multi-job streaming problem. Equal subplot sizes are used for both single and multi-job problems. Goal programming approach is used for solving the single job problem with multiple objectives that include makespan, the mean flow time, average work in process, and the setup and handling related costs. A near optimal heuristic is developed for the multi-job problem.

Kalir and Sarin (2001) study on sequencing the set of batches with equal sublots in flow shop environment. The objective of this study is to minimize the makespan. A new heuristic method called *bottleneck minimal idleness heuristic* is developed. The solutions that are obtained from the heuristic method are close to the optimal solutions.

Defersha and Chen (2010) consider lot streaming problems with variable sublots to minimize makespan. Although the problems with variable sublots are difficult to solve, they provide an improvement on the makespan. Efficient solution methods, which are mathematical model and hybrid genetic algorithm, are developed to solve n -job m -machine lot streaming problems with variable sublots and setup times.

Feldmann and Biskup (2008) examine the multi-job streaming problem in permutation flow shop environment. Their aim is to find optimal subplot sizes and sequence the sublots optimally. To split the order quantities of different products into sublots, a mixed integer programming model is developed. This model gives the optimal solution for small and medium sized problem instances.

Glass and Possani (2011) study on multi-job streaming problem as well. The aim of this study is to minimize the makespan. A polynomial time algorithm is developed to solve this problem that also contains attached setups on the first machine and transportation times between machines.

Mortezaei and Zulkifli (2013) have developed a mathematical model to integrate lot sizing and flow shop scheduling with lot streaming. A mixed integer linear programming model is developed to determine optimal production quantities, optimal inventory levels, optimal subplot sizes, and an optimal sequence. The mathematical model is developed for eight different types of cases, which are consistent sublots with intermingling, consistent sublots and no intermingling between sublots of the products (without intermingling), equal sublots with intermingling, equal sublots without intermingling, no-wait consistent sublots with intermingling, no-wait equal sublots with intermingling, no-wait consistent sublots without intermingling, and no-wait equal sublots without intermingling. The best makespan is obtained when consistent sublots with intermingling case are considered.

Kumar *et al.* (2000) consider an m -machine flow shop environment for multiple products lot streaming. The aim of this study is minimizing the makespan for multi-product problem with continuous-sized sublots. Optimal sequence is obtained by solving a traveling salesman problem. Also, a genetic algorithm is developed for this lot streaming and sequencing problem.

Hall *et al.* (2003) study the lot streaming problem with multiple products and attached setup times in a no-wait flow shop environment. A dynamic programming algorithm is developed. They show that this problem is equivalent to a classical traveling salesman problem with a pseudo-polynomial number of cities.

Çetinkaya (1994) considers two-machine multi-product flow shop problem with lot-detached setups and removal times and shows that the subplot sizing and subplot sequencing problems are independent and optimal sequences of lots are determined by using modification of Johnson's Algorithm (1954) to minimize maximum flow time (makespan).

Çetinkaya and Kayaligil (1992) study the scheduling of multiple job lots with unit sized transfer batches on a two machine flow line. The aim of this study is

minimizing the makespan. Optimal solution procedure, which is similar to Johnson's Rule (1954), is developed.

Çetinkaya and Duman (2010) consider the lot streaming problem of multiple jobs in a two-machine mixed shop with two different job types. An optimal solution method is developed for the mixed shop scheduling problem with flow shop and open shop jobs.

Liu (2009) studies on both COS and LS problems for a job shop environment. To solve this problem a mixed integer mathematical model is constructed using minimization of makespan, maximum lateness and finished goods. Because of the complexity of the problem, Genetic Algorithm is applied to determine lot streaming conditions. Lot Streaming – Genetic Algorithm heuristic is applied to solve the COS problem with lot streaming.

There are many studies in the scheduling literature that deal with the customer order scheduling and the multi-product lot streaming. However, to the best of our knowledge, both customer order scheduling and lot streaming for flow shops are studied first in this thesis.

CHAPTER 4

SOLUTION APPROACHES: MATHEMATICAL PROGRAMMING MODEL AND A PROPOSED TABU- SEARCH BASED HEURISTIC ALGORITHM

In this chapter, two solution approaches, which are the mathematical programming model and the proposed tabu-search based heuristic algorithm, are explained in detail.

4.1. Mathematical Programming Model

In this section, we develop a mixed integer linear programming (MILP) model, which is an extension of the generic model for the classical two-machine multi-job lot streaming problem by Sarin and Jaiprakash (2007), to determine the *job sequence* (i.e., sequence of the products) as well as the *sublot sequence* (i.e., sequence of the customer orders) in each job to minimize the total completion time of customer orders. The following indices, sets, parameters and decision variables are used in our model.

Parameters, indices and sets:

- K Number of customer orders
- k Index for customer orders ($k = 1, 2, \dots, K$)
- N Number of jobs
- j Index for jobs ($j = 1, 2, \dots, N$)
- O_k Set of jobs in customer order k
- J_j Set of customer orders having demand for job j
- n_j Number of customer orders having demand for job j
- $D_{j,k}$ Demand (number of identical items) for job j in customer order k
- L_j Lot size (total demand) for j , where $L_j = \sum_{k \in J_j} D_{j,k}$

- i Index for sublots
 m Index for machines ($m = 1, 2$)
 $p_{j,m}$ Unit processing time for job j on machine m
 $t_{j,m}$ Attached setup time for job j on machine m
 V Sufficiently large positive number

Decision variables:

$$X_{i,j,k} = \begin{cases} 1 & \text{if } i\text{th sublot of job } j \text{ belongs to customer order } k \\ 0 & \text{otherwise} \end{cases}$$

$$Y_{h,j} = \begin{cases} 1 & \text{if job } h \text{ precedes } j \\ 0 & \text{otherwise} \end{cases}$$

$s_{i,j}$ = Size of the i th sublot of job j

$C_{i,j,m}$ = Completion time of i th sublot of job j on machine m

CT_k = Completion time of the customer order k

MILP model:

$$\text{Minimize } \sum_{k=1}^K CT_k \quad (1)$$

Subject to;

$$\sum_{i=1}^{n_j} s_{i,j} = L_j \quad \text{for } j = 1, 2, \dots, N \quad (2)$$

$$\sum_{i=1}^{n_j} X_{i,j,k} = 1 \quad \text{for } k = 1, 2, \dots, K ; j \in O_k \quad (3)$$

$$\sum_{k \in J_j} X_{i,j,k} = 1 \quad \text{for } j = 1, 2, \dots, N ; i = 1, 2, \dots, n_j \quad (4)$$

$$s_{i,j} = \sum_{k \in J_j} D_{j,k} X_{i,j,k} \quad \text{for } j = 1, 2, \dots, N ; i = 1, 2, \dots, n_j \quad (5)$$

$$C_{1,j,1} - p_{j,1} s_{1,j} \geq t_{j,1} \quad \text{for } j = 1, 2, \dots, N \quad (6)$$

$$C_{i+1,j,1} - p_{j,1} s_{i+1,j} = C_{i,j,1} \quad \text{for } j = 1, 2, \dots, N ; i = 1, 2, \dots, n_j - 1 \quad (7)$$

$$C_{1,j,2} - p_{j,2} s_{1,j} \geq C_{1,j,1} + t_{j,2} \quad \text{for } j = 1, 2, \dots, N \quad (8)$$

$$C_{i+1,j,2} - p_{j,2} s_{i+1,j} \geq C_{i+1,j,1} \quad \text{for } j = 1, 2, \dots, N ; i = 1, 2, \dots, n_j - 1 \quad (9)$$

$$C_{i+1,j,2} - p_{j,2} s_{i+1,j} \geq C_{i,j,2} \quad \text{for } j = 1, 2, \dots, N ; i = 1, 2, \dots, n_j - 1 \quad (10)$$

$$\begin{aligned}
& (C_{i,j,m} - p_{j,m} s_{i,j}) - (C_{e,h,m} - p_{h,m} s_{e,h}) \\
& + V(1 - Y_{h,j}) \geq \left(L_h - \sum_{r=1}^{e-1} s_{r,h} \right) p_{h,m} + t_{j,m} \\
& + p_{j,m} \sum_{r=1}^{i-1} s_{r,j} \qquad \text{for } h \neq j \quad e = 1, 2, \dots, n_h \quad h = 1, 2, \dots, N \\
& \qquad \qquad \qquad i = 1, 2, \dots, n_j; j = 1, 2, \dots, N; m = 1, 2 \quad (11)
\end{aligned}$$

$$\begin{aligned}
& (C_{e,h,m} - p_{h,m} s_{e,h}) - (C_{i,j,m} - p_{j,m} s_{i,j}) \\
& + V Y_{h,j} \geq \left(L_j - \sum_{r=1}^{i-1} s_{r,j} \right) p_{j,m} + t_{h,m} \\
& + p_{h,m} \sum_{r=1}^{e-1} s_{r,h} \qquad \text{for } h \neq j; \quad e = 1, 2, \dots, n_h; \quad h = 1, 2, \dots, N \\
& \qquad \qquad \qquad i = 1, 2, \dots, n_j; j = 1, 2, \dots, N; m = 1, 2 \quad (12)
\end{aligned}$$

$$CT_k \geq C_{i,j,2} - V(1 - X_{i,j,k}) \quad \text{for } k = 1, 2, \dots, K; j \in O_k; i = 1, 2, \dots, n_j \quad (13)$$

$$X_{i,j,k}, Y_{h,j} \in \{0, 1\} \quad \text{for } \forall h, i, j, k \quad (14)$$

$$s_{i,j}, C_{i,j,m}, CT_k \geq 0 \quad \text{for } \forall i, j, k, m \quad (15)$$

In the above MILP model, the objective in (1) is to minimize the total completion time of customer orders. Constraint set (2) ensures that the sum of the items in the sublots of a job must be equal to the total number of items in that job. That is, the sum of the subplot sizes of a job must be equal to the lot size for that job. Constraint set (3) guarantees that each job of a customer order is assigned to only one subplot of that job. Constraint set (4) ensures that each subplot of a job can be assigned to only one customer order. Constraint set (5) guarantees that the sum of the items in a subplot of a job must be equal to the demand for that job in the customer order assigned to subplot. Constraint set (6) ensures that the processing of the first subplot, of any job appearing first in the sequence of the jobs, on machine 1 begins after the setup on the same machine has been completed. Constraint set (7) guarantees that a subplot, except the first subplot, of a job begins processing on machine 1 after the previous subplot is completed on the same machine. Constraint set (8) ensures that the first subplot of a job begins processing on machine 2 after it is completed on machine 1 and the setup on machine 2 has been completed. Constraint set (9) guarantees that all the sublots, excluding the first subplot, of a job begin processing on machine 2 after they are completed on machine 1. Constraint set (10) ensures that a subplot, except the first subplot, of a job begins processing on machine 2 after the previous subplot is

completed on the same machine. The terms on the right hand side of the constraint set (11) ensures that the difference between the start times of sublots e and i is at least equal to the sum of the processing times of the sublots e to n_h of job h and 1 to $i-1$ of job j and the setup time for job. Note that either constraints set (11) or (12) is valid for an optimal solution. Constraint set (13) ensures that the completion time of a customer order is the maximum of completion times of the jobs in that customer order. Constraint sets (14) and (15) impose binary and non-negativity restrictions on the decision variables, respectively.

Selecting the value of parameter V in the constraint sets (11), (12) and (13) affects the computational burden of the model, since it defines the feasible region. Our mathematical model is solved with a 3-hour time limit for every problem instance. Also, the 3-hour time limit increases the importance of the V -value. When V -value is estimated, total demand of the customer orders, processing times and setup times of machines are considered and we take the V -value as:

$$\sum_{m=1}^2 \sum_{j=1}^N [(L_j P_{j,m}) + t_{j,m}]$$

Note that V -value must be bigger than one of the maximum order's completion time. While V -value is estimated, the solution of the formulation below is rounding up. The numerical example is explained below to facilitate to understand.

When we consider the data in Table 2 below, the V -value is calculated as 1833. But we are rounding up 1833 to 2000 to prevent the infeasibility problem. On the other hand, this V -value affects the solution performance of GAMS. The example below is considered to compare the performance. When we take the V -value as 10 times bigger than the calculated one, the solution increases by nearly 54.5%. Also the iteration number increases nearly by 57%.

Table 2 Calculation of the V-Value

$D_{j,k}$						L_j	$P_{j,m}$		$t_{j,m}$		V	2000
$D_{j,k}$	O_1	O_2	O_3	O_4	O_5		m_1	m_2	m_1	m_2		
J_1	-	6	10	-	-	16	1	6	89	87	288	
J_2	3	4	10	2	8	27	5	9	73	55	506	
J_3	-	-	-	9	5	14	10	8	2	60	314	
J_4	9	8	8	4	7	36	4	9	38	80	586	
J_5	4	-	-	-	-	4	3	7	68	31	139	

4.2. Proposed Tabu-search Based Heuristic Algorithm

The size of the MILP model discussed in Section 4.1 increases drastically as the number of products (jobs) and the number of customer orders (sublots) increase. Therefore, the optimal solution for large-sized problems by solving the mathematical model is unlikely to be obtained within a reasonable amount of computation time. Moreover, suboptimal solutions are quite satisfactory for most real life problems. This reason motivated us to develop a fast tabu-search algorithm that provides optimal or near-optimal solutions.

Our proposed tabu-search based heuristic algorithm consists of four main phases: *Finding an Initial Job Sequence*, *Improving the Initial Job Sequence by the Insertion Algorithm*, *Improving the Job Sequence Obtained in the Second Phase by Pairwise Exchanges of the Sublots in Each Job*, and *Finding a Better Solution by the Tabu-search Algorithm*.

Phase 1: Finding an Initial Job Sequence

In the first phase of our proposed algorithm, we find an initial job sequence. The stepwise description of Phase 1 is given below.

Step 1. Construct the customer order and job list by arranging the customer orders (sublots) in each job in ascending order.

Step 2. To obtain the order list, sort the customer orders in ascending order of their number of jobs. If there is more than one customer order having the same number of jobs, then calculate the completion time for these orders and sort the customer orders in ascending order of their completion times.

Step-3. Consider the first customer order in the sorted order list, and check whether

this customer order has only one job or several jobs. If the customer order has more than one job, then calculate the run-in times of these jobs and sort the jobs in ascending order of their run-in times.

- Step 4. Select the first job of this customer order as the first job of the initial job sequence. Remove the selected job in Step 3 from the order list and go back to Step 2 until all jobs are sequenced in the *initial job sequence*. At this step, the completion times of orders are calculated according to run-in time job sequence in ascending order.
- Step 5. Calculate the total order completion time of the initial sequence by considering the subplot sequence from Step 1.

Phase 2: Improving the Initial Job Sequence by the Insertion Algorithm

Insertion algorithm is a kind of neighborhood algorithm used first by Nawaz *et al.* (1983) which solves the m -machine flow shop makespan minimization problem. In this phase of our proposed algorithm we adapt the insertion algorithm to improve the initial job sequence obtained in Phase 1. The stepwise description of Phase 2 is given below.

- Step 1. Consider the initial job sequence obtained in Phase 1, and select the first two jobs from this sequence. Form two partial sequences such that the first selected job is in the first and second positions in these partial sequences, respectively. For each partial sequence, compute the total order completion time, and select the best partial sequence.
- Step 2. Pick the job that is in the next position of the initial job sequence obtained in Phase 1. Generate all possible partial sequences by placing the new job in all possible positions (beginning, between and ending) in the partial sequence developed so far. Compute the total order completion times of all partial sequences, and select the best partial sequence giving the minimum total order completion time.
- Step 3. If all jobs of the initial sequence obtained in Phase 1 are considered, then stop; otherwise, go to Step 2.

Phase 3: Improving the Job Sequence Obtained in Phase 2 by Pairwise Exchanges of the Sublots in Each Job

In the first two phases of our proposed algorithm, we assume that the customer orders (sublots) in each job are sequenced in ascending order of their sizes (demands). However, the total order completion time may be improved by pairwise exchanges of the sublots in each job. The stepwise description of this phase of our algorithm is given below.

- Step 1. Consider the first job of the job sequence obtained in Phase 2 as the current job.
- Step 2.
 - (i) Check whether there is a subplot (customer order) in the current job such that this customer order does not appear in the following jobs of the job sequence obtained in Phase 2.
 - (ii) If there is no such customer order then
 - (a) Consider the next job of the sequence as the current job.
 - (b) If the current job is the last job in the sequence then stop; otherwise, go to Step 2(i).
- Step 3.
 - (i) Consider the first customer order which does not appear in the following jobs as the current customer order.
 - (ii) Temporarily pairwise exchange the positions of the current customer order and the customer order which immediately precedes the current customer order.
 - (iii) Check whether the pairwise exchange in Step 3(ii) improves the total completion time of all customer orders.
 - (iv) If the pairwise interchange does not improve the total completion time, then do not make this exchange and go to Step 3(v); otherwise,
 - (a) Make this pairwise exchange.
 - (b) If the new position of the current customer order is the first position in the current job, then go to Step 3 (v); otherwise go to Step 3 (ii).
 - (v) Check for the next customer order which does not appear in the following jobs and go to Step 2(ii).

Phase 4: Finding a Better Solution by the Tabu-Search Algorithm

Tabu-search is a higher level heuristic method to find a local optima. Principle of the tabu-search algorithm is to obtain improvement by pairwise interchanging the jobs. After determining the tabu list size and number of iterations, tabu iterations are conducted when the tabu list size and number of tabu iterations are reached to pre-determined values.

The job sequence obtained in Phase 3 will be the initial sequence of the tabu-search algorithm. By pairwise interchanging of the jobs, the total order completion time is improved. In the classical application of the tabu-search algorithm, all possible children are determined and the total order completion times of all possible children are calculated. The child with the minimum total order completion time is selected, and the tabu list is updated. The next tabu-search iteration continues with the new job sequence. However, we change the application of tabu-search by inserting a new step before selecting the best child. The new step is attached to the tabu-search algorithm to provide improvement on the total order completion time. Phase 3 is inserted as the new step of the tabu-search, and implemented before selecting the best child and after producing the sequences of the children. If the total order completion time is not improved after applying Phase 3, then we continue with the previous total order completion time and job sequence.

There are two stopping conditions for the tabu-search algorithm. Firstly, if all possible children are worse than the root, the algorithm automatically terminates. If the iteration count exceeds the pre-determined tabu iteration size, which we select as 5 then the algorithm stops. For the tabu-search algorithm, we specify the tabu list size as 3.

4.3. Numerical Example

In this section, a numerical example is provided to demonstrate the Proposed Heuristic Algorithm. We consider a problem instance in which there are five products (jobs) and five orders (sublots). Also, the unit processing time ($p_{j,m}$), sequence independent setup time ($t_{j,m}$) and demand for product j in customer order k ($D_{k,j}$) are given in Table 3.

Table 3 Data Set for the Numerical Example

Orders (O_k) Products (j)	$D_{k,j}$					$P_{j,m}$		$t_{j,m}$	
	O_1	O_2	O_3	O_4	O_5	m_1	m_2	m_1	m_2
J_1	-	6	10	-	-	1	6	89	87
J_2	3	4	10	2	8	5	9	73	55
J_3	-	-	-	9	5	10	8	2	60
J_4	9	8	8	4	7	4	9	38	80
J_5	4	-	-	-	-	3	7	68	31

Phase 1: Finding an Initial Job Sequence

Step 1: Using the data given in Table 3 the order and job lists are constructed as follows:

Order list:

$$O_1 = \{ J_2, J_4, J_5 \}, O_2 = \{ J_1, J_2, J_4 \}, O_3 = \{ J_1, J_2, J_4 \}, O_4 = \{ J_2, J_3, J_4 \} \text{ and}$$

$$O_5 = \{ J_2, J_3, J_4 \}$$

Job list:

$$J_1 = \{ O_2[6], O_3[10] \}$$

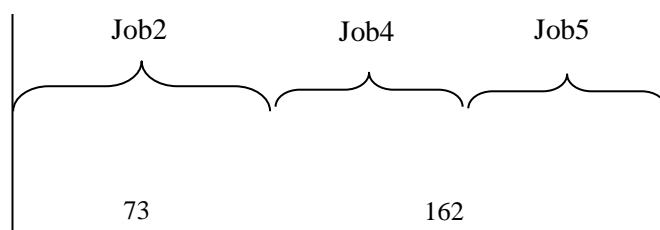
$$J_2 = \{ O_4[2], O_1[3], O_2[4], O_5[8], O_3[10] \}$$

$$J_3 = \{ O_5[5], O_4[9] \}$$

$$J_4 = \{ O_4[4], O_5[7], O_2[8], O_3[8], O_1[9] \}$$

$$J_5 = \{ O_1[4] \}$$

Step 2: From the order list in Step 1, it is clear that all customer orders have the same number of jobs, which is three. Thus, the order completion time for each customer order is independently calculated to sort the customer orders. As an illustration, the Gantt chart for order O_1 is given in Figure 3 below. From this figure, it is clear that order O_1 is completed in 390 time units. Similarly, the completion times for the other customer orders can be determined as $CT(O_2) = 461$, $CT(O_3) = 543$, $CT(O_4) = 423$, $CT(O_5) = 483$. Finally, the sorted list becomes $O_1 - O_4 - O_2 - O_5 - O_3$.



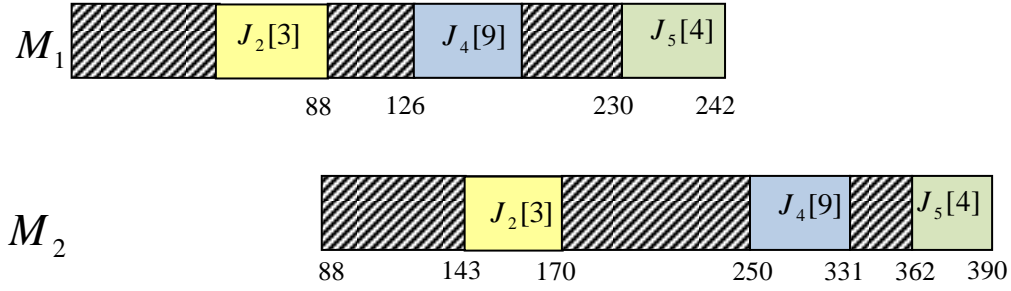


Figure 3 Gantt Chart for the Customer Order O_1

Step 3: The first customer in the sorted order list is order O_1 , which has more than one job. To select the first job of the initial sequence, run-in times for the jobs J_2 , J_4 and J_5 in the customer order O_1 must be calculated. The run-in times of J_2 , J_4 and J_5 equals to 83, 54, and 80, respectively. Thus, job J_4 will be the first job of the initial sequence since it has the minimum run-in time.

Step 4: We remove job J_4 from all orders having this job, and obtain the updated order list below:

Updated order list:

$$O_1 = \{J_2, J_5\}, O_2 = \{J_1, J_2\}, O_3 = \{J_1, J_2\}, O_4 = \{J_2, J_3\} \text{ and } O_5 = \{J_2, J_3\}$$

Step 2: From the updated order list in Step 4, it is clear that all customer orders have the same number of jobs which is two. The completion times for the customer orders are independently re-calculated according to jobs run-in time sequence and obtained as $CT(O_1) = 250$, $CT(O_2) = 309$, $CT(O_3) = 391$, $CT(O_4) = 307$, $CT(O_5) = 340$. Finally, the new sorted list becomes $O_1 - O_4 - O_2 - O_5 - O_3$.

Step 3: The first customer order in the new sorted order list is order O_1 , which has two jobs J_2 and J_5 . To select the second job of the initial sequence, the run-in times for the jobs J_2 and J_5 in the customer order O_1 must be calculated. The run-in times of jobs J_2 and J_5 equals to 83 and 80, respectively. Thus, job J_5 becomes the second job of the initial sequence.

Step 4: The second job of the initial sequence is selected at Step 3. So, the job J_5 is removed from the order list and the list is updated. Steps 2, 3 and 4 are repeated until all jobs are scheduled at the initial sequence, and we obtain the updated order list below:

Updated Order list:

$$O_1 = \{ J_2 \}, O_2 = \{ J_1, J_2 \}, O_3 = \{ J_1, J_2 \}, O_4 = \{ J_2, J_3 \} \text{ and } O_5 = \{ J_2, J_3 \}$$

Step 2: From the updated order list in Step 4, it is clear that order O_1 has only one job. Thus, there is no need for any completion time calculation.

Step 3: The first customer order in the new sorted order list is order O_1 which has only one job J_2 . Thus, there is no need for any run-in time calculation to select the candidate job. Job J_2 becomes the third job of the initial sequence.

Step 4: We remove job J_2 from all orders having this job and obtain the updated order list below:

Updated Order list:

$$O_2 = \{ J_1 \}, O_3 = \{ J_1 \}, O_4 = \{ J_3 \} \text{ and } O_5 = \{ J_3 \}$$

Step 2: From the updated order list in Step 4, it is clear that all customer orders have the same number of jobs which is one. The completion times for the customer orders are independently re-calculated according to jobs run-in time sequence and obtained as $CT(O_2) = 218$, $CT(O_3) = 246$, $CT(O_4) = 224$, $CT(O_5) = 152$. Finally, the new sorted list becomes $O_5 - O_2 - O_4 - O_3$.

Step 3: The first customer order in the new sorted order list is order O_5 which has only job J_3 . Thus, there is no need for any run-in time calculation to select the candidate job. Job J_3 becomes the fourth job of the initial sequence.

Step 4: We remove the job J_3 from all orders having this job and obtain the updated order list below:

Updated Order list:

$$O_2 = \{ J_1 \} \text{ and } O_3 = \{ J_1 \}$$

Step 2: From the updated order list in Step 4, it is clear that remaining two orders O_2 and O_3 have the same number of jobs which is one. Thus, the completion times for orders O_2 and O_3 are independently re-calculated according to jobs run-in time sequence, and obtained as $CT(O_2) = 218$ and $CT(O_3) = 246$. Finally, the new sorted order list becomes $O_2 - O_3$.

Step 3: The first customer order in the new sorted order list is order O_2 , which has only job J_1 . Thus there is no need for any run-in time calculation to select the candidate job. Job J_1 becomes the last job of the initial sequence.

Step 4: All jobs are considered, $J_4 - J_5 - J_2 - J_3 - J_1$ is the initial sequence and the jobs and orders schedule is as follows:

$$J_4(O_4[4], O_5[7], O_2[8], O_3[8], O_1[9]) - J_5(O_1[4]) - \\ J_2(O_4[2], O_1[3], O_2[4], O_5[8], O_3[10]) - J_3(O_5[5], O_4[9]) - J_1(O_2[6], O_3[10])$$

Step 5: The associated total order completion for the initial schedule is 4799 time units.

Phase 2: Improving the Initial Job Sequence by the Insertion Algorithm

Step 1: From the initial sequence ($J_4 - J_5 - J_2 - J_3 - J_1$) obtained in Phase 1, we select the first two jobs J_4 and J_5 . We form two partial sequences $J_4 - J_5$ and $J_5 - J_4$, and obtain the total order completion times for these partial sequences as given in Table 4 below.

Table 4 Total Order Completion Times for Two Partial Sequences

Job and Order Sequence	Total Order Completion Time
$J_4(O_4[4], O_5[7], O_2[8], O_3[8], O_1[9]) - J_5(O_1[4])$	1602
$J_5(O_1[4]) - J_4(O_4[4], O_5[7], O_2[8], O_3[8], O_1[9])$	1968

The selected partial sequence is $J_4 - J_5$ since its total completion time is smaller than

that of the partial sequence $J_5 - J_4$.

Step 2: We select the next job, which is job J_2 , from the initial job sequence obtained in Phase 1, and form three partial sequences $J_2 - J_4 - J_5$, $J_4 - J_2 - J_5$ and $J_4 - J_5 - J_2$, and obtain the total order completion time of each partial sequence as given in Table 5.

Table 5 Total Order Completion Times for Three Partial Sequences

Job and Order Sequence	Total Order Completion Time
$J_2(O_4[2], O_1[3], O_2[4], O_5[8], O_3[10]) -$ $J_4(O_4[4], O_5[7], O_2[8], O_3[8], O_1[9]) -$ $J_5(O_1[4])$	3237
$J_4(O_4[4], O_5[7], O_2[8], O_3[8], O_1[9]) -$ $J_2(O_4[2], O_1[3], O_2[4], O_5[8], O_3[10]) -$ $J_5(O_1[4])$	3362
$J_4(O_4[4], O_5[7], O_2[8], O_3[8], O_1[9]) -$ $J_5(O_1[4]) -$ $J_2(O_4[2], O_1[3], O_2[4], O_5[8], O_3[10])$	3400

The selected partial sequence is $J_2 - J_4 - J_5$ since its total completion time is the smallest one among the total completion times of the three partial sequences above.

Step 2: We select the next job J_3 from the initial job sequence obtained in Phase 1, and form four partial sequences $J_3 - J_2 - J_4 - J_5$, $J_2 - J_3 - J_4 - J_5$, $J_2 - J_4 - J_3 - J_5$, and $J_2 - J_4 - J_5 - J_3$, and compute the total order completion times of each partial sequence as given in Table 6.

Table 6 Total Order Completion Time for Four Partial Sequences

Job and Order Sequence	Total Order Completion Time
$J_3(O_5[5], O_4[9]) -$ $J_2(O_4[2], O_1[3], O_2[4], O_3[8], O_3[10]) -$ $J_4(O_4[4], O_5[7], O_2[8], O_3[8], O_1[9]) -$ $J_5(O_1[4])$	3947
$J_2(O_4[2], O_1[3], O_2[4], O_3[8], O_3[10]) -$ $J_3(O_5[5], O_4[9]) -$ $J_4(O_4[4], O_5[7], O_2[8], O_3[8], O_1[9]) -$ $J_5(O_1[4])$	4097
$J_2(O_4[2], O_1[3], O_2[4], O_3[8], O_3[10]) -$ $J_4(O_4[4], O_5[7], O_2[8], O_3[8], O_1[9]) -$ $J_3(O_5[5], O_4[9]) -$ $J_5(O_1[4])$	4194
$J_2(O_4[2], O_1[3], O_2[4], O_3[8], O_3[10]) -$ $J_4(O_4[4], O_5[7], O_2[8], O_3[8], O_1[9]) -$ $J_5(O_1[4]) -$ $J_3(O_5[5], O_4[9])$	4140

The selected partial sequence is $J_3 - J_2 - J_4 - J_5$ since its total order completion time is the smallest one among the total order completion times of the above four partial sequences.

Step 2: We select the next job J_1 from the initial job sequence obtained in Phase 1, and form five partial sequences $J_1 - J_3 - J_2 - J_4 - J_5$, $J_3 - J_1 - J_2 - J_4 - J_5$, $J_3 - J_2 - J_1 - J_4 - J_5$, $J_3 - J_2 - J_4 - J_1 - J_5$ and $J_3 - J_2 - J_4 - J_5 - J_1$, and obtain the total order completion time of each partial sequence as given in Table 7.

The selected partial sequence is $J_3 - J_2 - J_4 - J_5 - J_1$ since it has the smallest total order completion time.

Phase 3: Improving the Job Sequence Obtained in Phase 2 by Pairwise Interchanging the Sublots of the Jobs

Step 1: The first job of the sequence obtained in Phase 2 is $J_3(O_5[5], O_4[9])$, which is considered as the current job.

Step 2: The sublots of job J_3 are order O_5 and order O_4 , and these orders exist in the following jobs J_2 , J_4 and J_5 of the job sequence obtained in Phase 2. Thus, the next job, i.e., job J_2 has all types of sublots and these sublots exist in the following jobs J_4 and J_5 . Thus, no subplot swapping is possible. Therefore, we must consider the next job of the sequence obtained in Phase 2, which is job J_4 . Some of the sublots cannot be seen in the following jobs. So, we can pass to the next step.

Step 3: Order O_5 is the second customer order in job J_4 . Orders O_4 and O_5 can be pairwise interchanged, and the total order completion time increases to 4632. Thus, this pairwise interchange does not improve the total order completion time, and we don't make this interchange. We check the remaining sublots for possible improvement. Orders O_1 , O_2 and O_3 exist in the following jobs J_4 and J_1 of the sequence obtained in Phase 2. Therefore, we go to Step 2 again.

Table 7 Total Order Completion Time for Five Partial Sequences

Job and Order Sequence	Total Order Completion Time
$J_1(O_2[6], O_3[10]) -$ $J_3(O_5[5], O_4[9]) -$ $J_2(O_4[2], O_1[3], O_2[4], O_5[8], O_3[10]) -$ $J_4(O_4[4], O_5[7], O_2[8], O_3[8], O_1[9]) -$ $J_5(O_1[4])$	5072
$J_3(O_5[5], O_4[9]) -$ $J_1(O_2[6], O_3[10]) -$ $J_2(O_4[2], O_1[3], O_2[4], O_5[8], O_3[10]) -$ $J_4(O_4[4], O_5[7], O_2[8], O_3[8], O_1[9]) -$ $J_5(O_1[4])$	4922
$J_3(O_5[5], O_4[9]) -$ $J_2(O_4[2], O_1[3], O_2[4], O_5[8], O_3[10]) -$ $J_1(O_2[6], O_3[10]) -$ $J_4(O_4[4], O_5[7], O_2[8], O_3[8], O_1[9]) -$ $J_5(O_1[4])$	4862
$J_3(O_5[5], O_4[9]) -$ $J_2(O_4[2], O_1[3], O_2[4], O_5[8], O_3[10]) -$ $J_4(O_4[4], O_5[7], O_2[8], O_3[8], O_1[9]) -$ $J_1(O_2[6], O_3[10]) -$ $J_5(O_1[4])$	4670
$J_3(O_5[5], O_4[9]) -$ $J_2(O_4[2], O_1[3], O_2[4], O_5[8], O_3[10]) -$ $J_4(O_4[4], O_5[7], O_2[8], O_3[8], O_1[9]) -$ $J_5(O_1[4]) -$ $J_1(O_2[6], O_3[10])$	4605

Step 2: The next job J_5 has only one customer order, so we could not make any

pairwise interchange. Now, our current job is job J_1 , which is the last job of the sequence obtained in Phase 2. The algorithm stops here without customer order (sublot) swapping.

Phase 4: Finding a Better Solution by the Tabu-Search Algorithm

The first iteration of the tabu-search algorithm is given in Figure 4. The initial sequence and the associated total order completion time are taken from the solution obtained in Phase 3. When we apply Phase 3 to all job sequences generated from these initial sequences, we observe that no improvement is obtained for some job sequences generated.

However, the total order completion time is decreased from 4605 to 4579 as illustrated in Figure 4. The tabu list is updated with pair of (J_4, J_5) . The detail of the second iteration is given in Figure 5. It is clear that no further improvement on the total order completion time is achieved in this iteration. Thus, tabu-search algorithm terminates before reaching the tabu-search iteration size of 5. Moreover, the updated tabu list does not change.

The Gantt chart of the schedule obtained for the example is illustrated in Figure 6, where the sum of the completion time of the customer orders is 4579 ($=842+1109+1169+698+761$). This completion time is equivalent to that of the optimal solution.

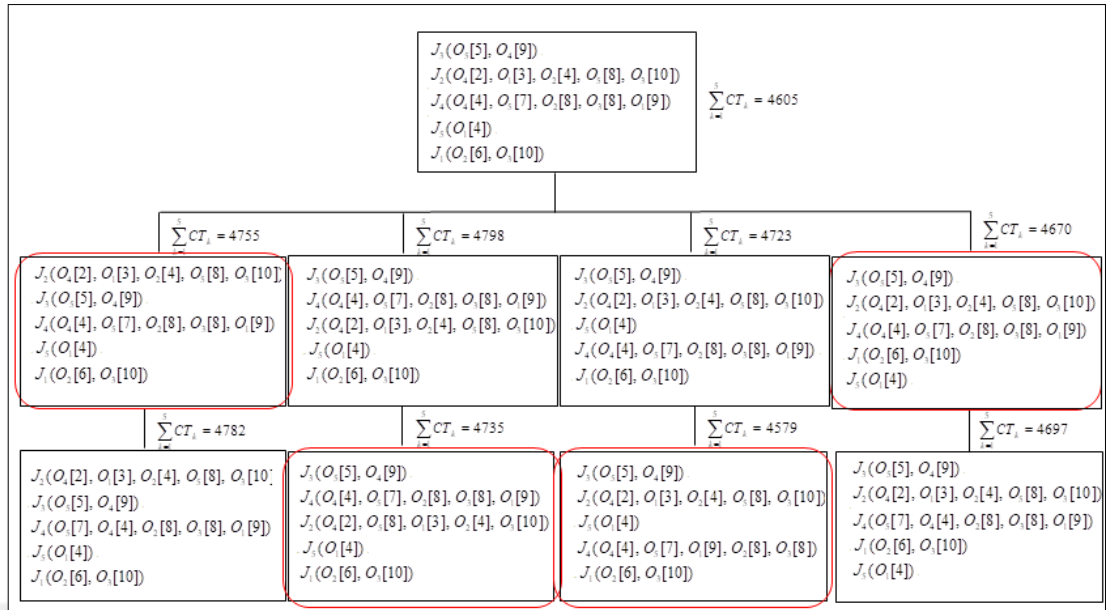


Figure 4 Tabu-search Iteration -1

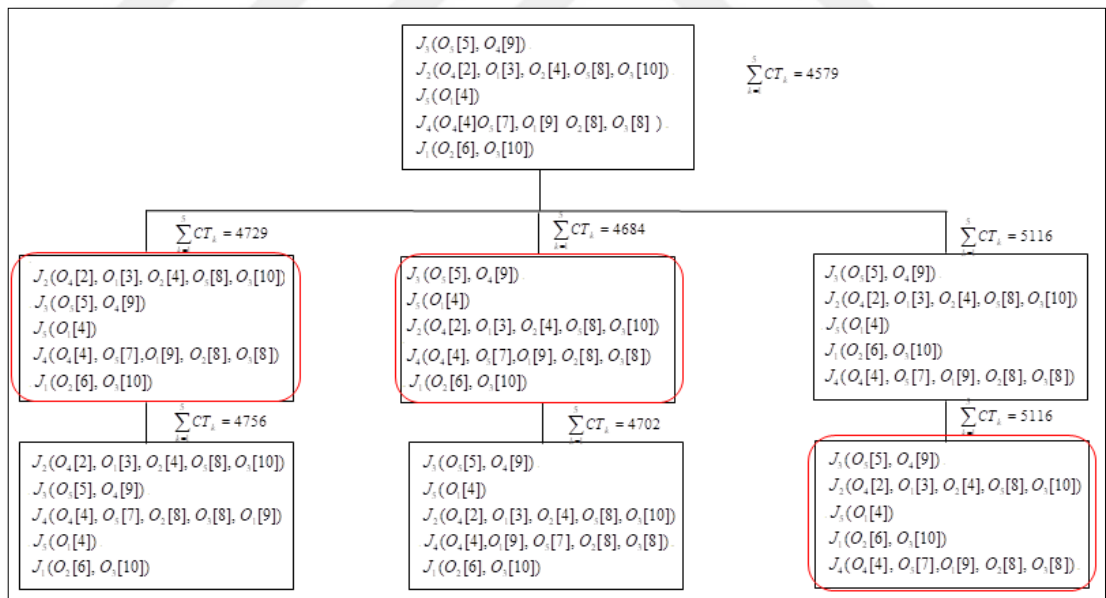


Figure 5 Tabu-search Iteration-2

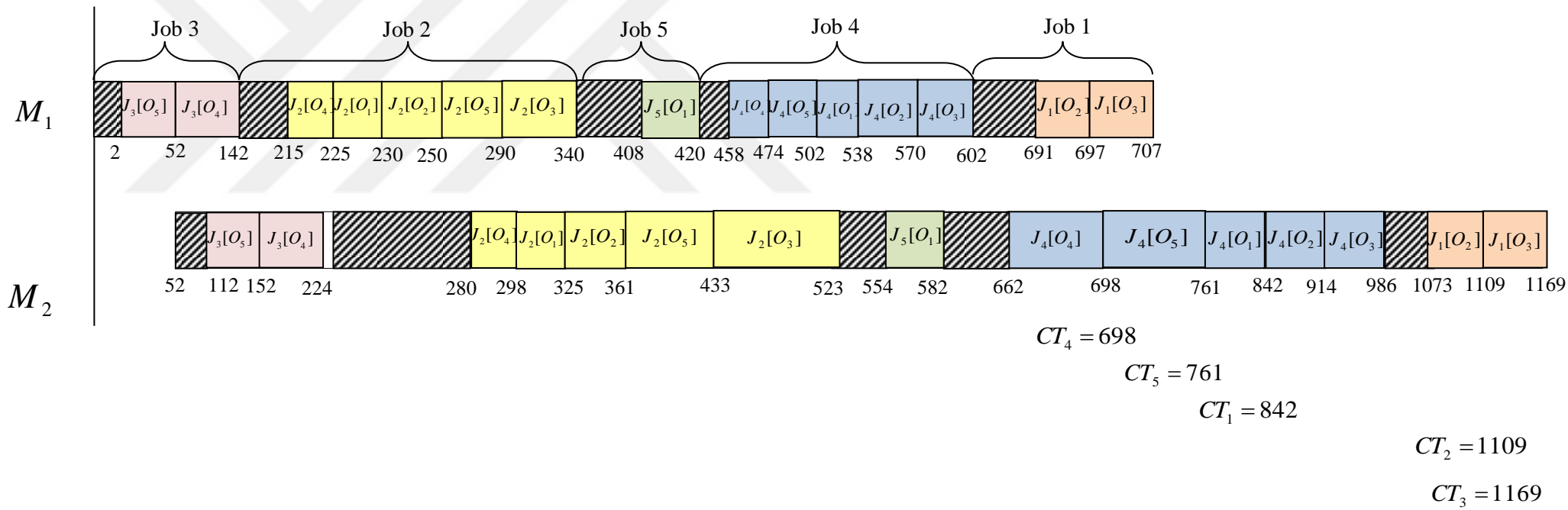


Figure 6 Gantt Chart of the Schedule Obtained for the Numerical Example

CHAPTER 5

COMPUTATIONAL EXPERIMENTS

In this chapter, we describe our computational tests with randomly generated problems to evaluate the effectiveness and efficiency of the MILP model and the proposed heuristic algorithm in finding optimal or near optimal schedules. The mathematical model is solved by using GAMS 24.1 and all computations are conducted on a computer with Intel (R) Xeon (R) CPU E52650 0@ 2.00GHz 2.00 GHz (2 Processors) and 128 GB RAM under Windows 10 operating system. Also the proposed heuristic algorithm is coded in Java programming language. All computational experiments are conducted on a personal computer with Intel Core i7 Dual-Core 2.00 GHz CPU and 8 GB RAM under Windows 10 operating system.

5.1. Computational Settings for the Test Problems

The values of the parameters used in our experiments will be generated as follows:

1. *Number of customer orders (K)*: They are taken as 5 and 10.
2. *Number of jobs (N)*: They are taken as 5, 10, 15 and 20.
3. *Number of customer orders having demand for each job (n_j)*: They are randomly generated from a $DU[1, K]$.
4. *Demand (number of identical items) for each job in each customer order ($D_{j,k}$)*: They are randomly generated from a discrete uniform distribution $DU[1, 10]$.
5. *Processing times ($p_{j,m}$)*: They are randomly generated from a discrete uniform distribution $DU[1, 10]$.
6. *Setup times ($t_{j,m}$)*: They are randomly generated from a discrete uniform distribution $DU[0, 100f]$, where f is taken as 0.0, 0.5, 1.0, 1.5, and 2.0.

For each possible combination of the above parameters, 5 replicates (problem

instances) are generated, and totally 200 problem instances are tested for the setup case. Also, 5 replicates are generated for each possible combination of the above parameters excluding setup times, and totally 200 problem instances are tested for no setup case. Hence, totally 400 problem instances are tested.

5.2. Performance Measures

The solver GAMS gives two types of solutions for the MILP models. One of the solutions is the best integer solution which is the desired one; other solution is the best non-integer solution in which some of the variables are not integers. If the best non-integer solution obtained is equal to the best integer solution, then we conclude that the optimal solution is achieved by the MILP model. Otherwise we are uncertain about optimality of the solution. For the problems with optimal solutions, we compare the total completion time obtained by our heuristic algorithms with the total completion time of the optimal solution. However, for the problems with the best integer solutions, we compare the total completion time obtained by our heuristic algorithms with the total completion time of the best integer solution.

To measure the effectiveness of the heuristic algorithms for the cases in which an optimal solution is obtained by the MILP model, we calculate the percent deviation of the total completion time obtained by the heuristic algorithm from the total completion time of the optimal solution. The following notation is used to measure the effectiveness of the Proposed Tabu-search Based Heuristic Algorithm.

PD^O : Percent deviation from the optimal solution.

TC^H : Total completion time solution obtained by heuristic algorithm.

TC^O : Total completion time solution obtained by the MILP model

$$PD^O = 100 \times (TC^H - TC^O) / TC^O$$

Similarly, for the cases in which an optimal solution is not guaranteed (but a best-integer solution exists) by the MILP model, we calculate the percent deviation of the total completion time obtained by each heuristic algorithm from the total completion time of the best-integer solution. Also, the following extra notation is used to evaluate the effectiveness of the proposed heuristic algorithm.

PD^B : Percent deviation from the best integer solution

TC^B : Total completion time of the best integer solution obtained by the MILP model

$$PD^B = 100 \times (TC^H - TC^B) / TC^B$$

The efficiency measure of the MILP model and the heuristic algorithm is the computational time required to solve the problem. In our experiments, we limit the run time of the GAMS for obtaining the optimal solution of each problem instance to 10,800 seconds (3 hours). The computational time for the proposed heuristic algorithm is relatively very small, less than 45 seconds, for all problem instances. Also note that the computational time required for solving a problem instance increases as the number of jobs and the number of customer orders increase. But the computational time is again very small, which is less than 45 seconds generally.

5.3. Discussions of the Results

In this section the performance of the solution approaches is discussed. We first examine the performance of the MILP model, and then discuss the performance of the heuristic algorithm for the setup and no setup cases. The solution approaches are examined with respect to the number of jobs and the number of customer orders.

5.3.1. Setup Case

5.3.1.1. Performance of the MILP model

In this section, performance of the MILP is discussed. Performance of the MILP is impacted by the complexity of the problem. We deal with both COS and lot streaming problem. Also, the number of machine increases the complexity of the problem. These mentioned situations affects the problem solution duration seriously. We limit the runtime of the MILP to 3 hours.

Depending on the number of jobs and number of customer orders, problem complexity increases directly. As shown in Table 8, when the number of jobs and the number of customer orders are 5, all problem instances give the optimal solution. When the number of jobs is 10 and the number of customer orders is 5, the MILP found only one optimal solution within three hours' time limit. Finally, four optimal solutions are found when the number of jobs is 5 and the number of customer orders

is 10. For the remaining problem instances, best integer solutions are obtained and there is no unsolved problem among the experiment set.

Table 8 Performance of the MILP Model for the Setup Case

Number of customer orders (K)	5				10			
Number of jobs (N)	5	10	15	20	5	10	15	20
Total number of problem instances considered	25	25	25	25	25	25	25	25
Number of optimum integer solutions obtained in 3 hours	25	1	0	0	4	0	0	0
Number of best integer solutions obtained	0	24	25	25	21	25	25	25
Average gap (%)	0	45	78	87	38	85	91	93
Total Average gap (%)	64.7							

To emphasize the performance of MILP model, we should investigate the quality of solutions which are not optimal. It is a common phenomenon that MILP model ends up with a gap between solution found and the best possible. Therefore, gap values are examined in order to indicate the percentage difference of integer solution from the theoretical optimum. Gap values are analyzed for 170 non-optimally solved problem instances under three circumstances; best case, worst case, and average case. For some of the problem instances, so many iterations are done and integer solutions found become closer to the theoretical optimum after each iteration. However, GAMS is terminated because of time limitation before reaching the optimum solution. But, this case is the best case since until 3-hour time limitation is completed, gap values are very close to zero.

On the other hand, for some problem instances branching becomes very difficult and time consuming. When branching is slow, the number of iterations is moderate which leads to higher gap values than the best case and lower gap values than the worst case. Experiments that are conducted according to generated data from Section 5.1, has gap values as in Table 8. When the number of jobs and the number orders are 5, all of the gap values equal to zero which means that MILP can solve all experiment set optimally.

When the number of orders is five, the average gap value for 100 problem instances

solved equal to 52.7%. Also the average gap value for 100 problem instances solved equals to 77 % when the number of orders is 10. The details of the average gap values can be seen in Table 8.

5.3.1.2. Performance of the proposed heuristic algorithm

In this section, we discuss the effects of changes in the problem parameters on the performance of the proposed heuristic algorithm. Table 9 is prepared to show the average percent deviations of the heuristic algorithm. Also, the maximum deviations can be seen in this table. These analyses are made based on the optimal solution and the best integer solution that are analyzed in Section 5.3 For all problem instances, average percent deviation of the heuristic algorithm is founded 0.57% as you can see in Table 9.

Table 9 Average Percent Deviations of the Heuristic Algorithm from the Optimal and Best Integer Solution for the Setup Case

<i>N</i>	<i>K</i>	Number of Problem Instances Give Optimal Solution	Average Percent Deviation of Heuristic Algorithm from Optimal Solution	Number of Problem Instances Give Best Integer Solution	Average Percent Deviation of Heuristic Algorithm from Best Integer Solution	Overall AVG.	MAX. DEV.
5	5	25	0.36%	-	-	0.36%	2.59%
	10	4	0.00%	21	1.51%	1.27%	7.66%
Grand Avg_5		0.81%					
10	5	1	0.61%	24	1.11%	1.09%	9.34%
	10	-	-	25	0.82%	0.82%	4.54%
Grand Avg_10		0.96 %					
15	5	-	-	25	0.52%	0.52%	3.52%
	10	-	-	25	0.44%	0.44%	7.11%
Grand Avg_15		0.48%					
20	5	-	-	25	0.08%	0.08%	1.62%
	10	-	-	25	0.00%	0.00%	0.00%
Grand Avg_20		0.04%					
AVG_TOTAL		0.57%					

When the number of jobs is 5, 29 problem instances give optimal solution. Also, the grand average of the heuristic algorithm is obtained as 0.81%. When the number of orders is 5, all of the problem instances are solved optimally and the average percent deviation of the heuristic algorithm is 0.36%. Also, maximum deviation of these

problem instances is derived as 2.59%. This value is smaller than the 7.66% that is the maximum deviation when the number of orders is 10. Moreover, 4 problem instances whose average percent deviation is obtained as 0.00% are solved optimally. The average percent deviation of the remaining problem instances is 1.51% and overall average percent deviation is derived as 1.27% while 25 problem instances are being considered. On the other hand, when the maximum deviation value (7.66%) is not considered, the grand average percent deviation decreases from 0.81% to 0.66%.

When the number of jobs is 10, only 1 problem instance gives optimal solution. Also, the grand average of the heuristic algorithm is obtained as 0.96%. The average percent deviation for the optimal problem instance is derived as 0.61% when the number of orders is 5. Also, average percent deviation of the remaining problem instances are obtained as 1.11% and average percent deviation of overall problem instances is obtained as 1.09%. MILP cannot find any optimal solution under time limitation when the number of orders is 10. The average percent deviation is obtained as 0.82% for these problem instances. Also, the maximum deviations are obtained as 9.34% and 4.54% when the number of orders is 5 and 10 respectively. If the maximum deviation (9.34%) is omitted, grand average percent deviation decreases from 0.96% to 0.77%.

Optimal solution could not be obtained under 3-hour time limitation when the number of jobs is 15. While the complexity of the problem is increasing, the number of problem instances having better heuristic algorithm solution than best integer solution also increases. For this experiment set, 34 instances are observed with the same situation, and negative deviations for these problem instances are taken as 0.00%. Also, the grand average percent deviation is 0.48%. When we take into account these negative deviations, the grand average percent deviation is obtained as -1.85%. The average percent deviations are obtained as 0.52% and 0.44% when the number of orders is 5 and 10, respectively. On the other hand, the maximum deviations are obtained as 3.52% and 7.11% for these experiments. If the maximum deviation is omitted, the grand average percent deviation decreases from 0.48% to 0.34%.

Finally, none of the problem instances are solved optimally under less than 3-hour time limitation when the number of jobs is 20. For the 48 instances, the heuristic algorithm solutions are closer to the optimal solution than the best integer solution,

and the negative percent deviation for these instances is taken as 0.00%. As it is understood, the heuristic algorithm gives better solutions than the best integer solutions while complexity of the problem increases. Also, the grand average percent deviation is obtained as 0.04%. When we take into account the negative percent deviations, the average percent deviation is obtained as -5.99%. The average percent deviations are obtained as 0.08% and 0.00% when the numbers of orders are 5 and 10, respectively. On the other hand, the maximum deviations are derived as 1.62% and 0.00% for these experiments.

Consequently, the total average percent deviation is 0.57% including the problem instances which have maximum deviations. If the maximum deviations are omitted, the total average percent deviation decreases from 0.57% to 0.45%. Furthermore, when the negative deviations are considered, the total average deviation decreases to -1.53%. On the other hand, 47 problem instances are solved with 0.00% deviation among the 200 problem instances. While 21 of them can be solved optimally, 26 of them are best integer with 59% average gap value. Among 21 problem instances 17 of them are obtained when the numbers of jobs and orders are considered as 5. All the remaining 4 of them are obtained when the number of jobs is considered as 5 and the number of orders is considered as 10. Among 26 problem instances, 13 of them are obtained when the number of jobs is 10. 5 of them are obtained when number of jobs is 15 and all remaining 8 of them are obtained when the number of jobs is 5.

5.3.2. No Setup Case

5.3.2.1. Performance of the MILP model

In this section, performance of the MILP is discussed for the no setup case. Much as setup times affect the performance of the MILP, problem complexity that is why COS and LS problems are both considered at the same time could not be ignored. Also, the 3-hour run time limit is still valid for no setup case that is same as with setup case. While the problem instances for the no setup case is generated, the f value which is described in Section 5.1 is taken as 0.0.

As it is mentioned before, the complexity is impressed by the number of jobs and number of orders. As shown in Table 10, when the number of jobs and the number of

customer orders are 5, all problem instances give the optimal solution. When the number of jobs is 10 and the number of customer orders is 5, the MILP gives 2 optimal solutions within 3-hour time limit. Also, 4 optimal solutions are obtained when the number of jobs is 5 and the number of customer orders is 10. The remaining solutions among the whole problem instances are obtained as best integer solution by using GAMS.

Table 10 Performance of the MILP Model for the No Setup Case

Number of customer orders (K)	5				10			
Number of jobs (N)	5	10	15	20	5	10	15	20
Total number of problem instances considered	25	25	25	25	25	25	25	25
Number of optimum integer solutions obtained in 3 hours	25	2	0	0	4	0	0	0
Number of best integer solutions obtained	0	23	25	25	21	25	25	25
Average GAP (%)	0	42	80	89	38	90	94	96
Total Average GAP (%)	66.4							

The gap values are also analyzed for the no setup case. As mentioned before the gap values are examined in order to indicate the percentage difference of integer solution from the theoretical optimum. The gap values are analyzed for 169 non-optimal problems for the no setup case. It is expected that the gap values for no setup case is better than the gap values for setup case. However, committed branching operations affect the gap values seriously. The details of the gap values can be seen in Table 10. Totally, the average percent of the gap value is obtained 66.4% for the no setup case.

5.3.2.2. Performance of the proposed heuristic algorithm

In this section, we discuss the effects of changes in the problem parameters on the performance of the proposed heuristic algorithm for the no setup case. Average percent deviations of the proposed heuristic algorithm are analyzed for the no setup case. These analyses are made based on the optimal and best integer solutions that are analyzed in Section 5. For all problem instances, average percent deviation of the heuristic algorithm is 1.24% as it can be seen in Table 11.

When the number of jobs is 5, 29 problem instances give optimal solution. Also the grand average of the heuristic algorithm is obtained as 1.61%. All of the problem instances are solved optimally when the number of orders is 5 and the average percent deviation of the heuristic algorithm from the optimal is 2.26%. The reason that is why the high average percent deviation is obtained is the number of problem instances having high deviations and also the maximum deviation. When the maximum deviation is omitted, the average percent deviation decreases from 2.26% to 1.75%. Also the maximum deviation is 7.93% when we consider the problem instances having 10 orders. Among 25 problem instances, 4 of them are solved optimally, and the heuristic algorithm solutions are same as the optimal solutions. The average percent deviation is 1.14% for the remaining problem instances, and the overall deviation is 0.96% when the number of orders is 10. When the maximum deviations are omitted, the grand average percent deviation decreases from 1.61% to 1.20%.

When the number of jobs is 10, 2 problem instances give the optimal solution. The average percent deviation is 8.52% for the problem instances solved optimally when the number of orders is 5. Also, the grand average percentage of the heuristic algorithm is obtained as 2.00%. Also, the average percent deviation of all remaining problem instances and the overall average percent deviation are 2.52% and 3.00% respectively. The maximum deviation in this experiment set is 14.73% while this value is 3.98% when the number of orders is 10. None of the problem instances in this experiment set can be solved optimally and the average percent deviation is 1.00%. If the maximum deviation is omitted, the grand average percent deviation decreases from 2.00% to 1.45%. On the other hand, the proposed heuristic algorithm obtains better solutions than the best integer solutions provided by the MILP.

Optimal solution could not be obtained under 3-hour limitation, when the number of jobs is 15. While the complexity of the problem is increasing, the number of problem instances having better heuristic algorithm solution than the best integer solution also increase. For this experiment set, 24 instances are observed with the same situation, and negative deviations are taken as 0.00%. The grand average percent deviation is 1.03%. When we take into account this negative deviations, the grand average percent deviation is obtained as -0.94%. The average percent deviations are obtained as 1.55% and 0.50% when the number of orders is 5 and 10 respectively. Also, the

maximum deviations are 8.26% and 3.71% for this experiment set.

Finally, none of the problem instances are solved optimally under 3-hour time limitation when the number of jobs is 20. For the 44 instances, the heuristic algorithm solutions are closer to the optimal solution than the best integer solution, and the negative percent deviation for these instances is taken as 0.00%. Also the grand average percent deviation is obtained as 0.32%. When we take into account the negative percent deviations, the grand average percent deviation decreases from 0.32% to -0.94%. The average percent deviations are obtained as 0.18% and 0.46% when the number of orders is 5 and 10, respectively. Also, the maximum deviations are 1.76% and 6.57% for these experiments.

Consequently, the total average percent deviation is 1.24% including the problem instances which the maximum deviations. If the maximum deviations are omitted, the total average percent deviation decreases from 1.24 % to 0.99%. Furthermore, when the negative deviations are considered, the total average deviation decreases to -0.91%. On the other hand, 27 problem instances are solved with 0.00% deviation among the 200 problem instances. While 17 one of them can be solved optimally, 10 of them are best integer with 42% average gap value. Among 17 problem instances 17 of them are obtained when the number of jobs is considered as 5. Among 10 problem instances, 6 of them are obtained when the number of jobs is 5. 4 of them are obtained when number of jobs is 10.

Table 11 Average Percent Deviations of the Heuristic Algorithm from the Optimal and Best Integer Solution for the No Setup Case

N	K	Number of Problem Instances Give Optimal Solution	Average Percent Deviation of Heuristic Algorithm from Optimal Solution	Number of Problem Instances Give Best Integer Solution	Average Percent Deviation of Heuristic Algorithm from Best Integer Solution	Overall AVG.	MAX. DEV.
5	5	25	2.26%	-	-	2.26%	12.64%
	10	4	0.00%	21	1.14%	0.96%	7.93%
Grand Avg_5		1.61%					
10	5	2	8.52%	23	2.52%	3.00%	14.73%
	10	-	-	25	1.00%	1.00%	3.98%
Grand Avg_10		2.00 %					
15	5	-	-	25	1.55%	1.55%	8.26%
	10	-	-	25	0.50%	0.50%	3.71%
Grand Avg_15		1.03%					
20	5	-	-	25	0.18%	0.18%	1.76%
	10	-	-	25	0.46%	0.46%	6.57%
Grand Avg_20		0.32%					
AVG_TOTAL		1.24%					

5.4. Comparison of the Setup and No Setup Cases

All of the performance details of the heuristic algorithm are discussed for both setup and no setup cases. Totally, 400 problem instances are analyzed. 200 of them are analyzed for setup case and the remaining 200 problem instances are analyzed for no setup case. In this section, we compare the setup and no setup cases for both MILP and proposed heuristic algorithm performances.

5.4.1. Comparison of the Setup and No Setup Cases for the MILP Performance

Table 12 illustrates the summary for both setup and no setup cases when the number of customer orders is 5. For both cases optimal solutions are obtained when the number of jobs is 5. When the number of jobs is 10, among 25 problem instances solved 1 optimal solution is obtained for the setup case and 2 optimal solutions are obtained for the no setup case. Impact of the setup time can be seen on the Gap values. While the average Gap values for the setup case and no setup case are 45%, and 42%, respectively. Totally all remaining 147 problem instances cannot be solved optimally for both cases. Also, the Gap values can be seen in Table 11 when the

numbers of jobs are 15 and 20. From Table 12, it is clear that the performance of the MILP model is almost same for setup and no setup cases when the number of customer order is 5.

Table 12 Comparison of the Setup and No Setup Cases for MILP Performance for 5 orders

Case	Setup Case				No Setup Case			
Number of customer orders (K)	5				5			
Number of jobs (N)	5	10	15	20	5	10	15	20
Total number of problem instances	25	25	25	25	25	25	25	25
Number of optimum integer solutions	25	1	0	0	25	2	0	0
Number of best integer solutions	0	24	25	25	0	23	25	25
Average GAP (%)	0	45	78	87	0	42	80	89
Total Average GAP (%)	52.7				53.2			

Table 13 gives the summary for both setup and no setup cases when the number of customer orders is 10. For both cases 4 optimal solutions are obtained when number of jobs is 5. Totally all remaining 192 problem instances cannot be solved optimally for both cases. Also, the average gap values can be seen in Table 13. Again the performance of the MILP model is almost same for both setup and no setup cases when the number of customer order is 10, as it can be observed from Table 13.

Table 13 Comparison of the Setup and No Setup Cases for MILP Performance for 10 orders

Case	Setup Case				No Setup Case			
Number of customer orders (K)	10				10			
Number of jobs (N)	5	10	15	20	5	10	15	20
Total number of problem instances	25	25	25	25	25	25	25	25
Number of optimum integer solutions	4	0	0	0	4	0	0	0
Number of best integer solutions	21	25	25	25	21	25	25	25
Average GAP (%)	38	85	91	93	38	90	94	96
Total Average GAP (%)	77				80			

5.4.2. Comparison of the Setup and No Setup Cases for the Heuristic Algorithm Performance

Table 14 illustrates the performance of the heuristic algorithm for both setup and no setup cases. When the solutions are analyzed, it is seen that the heuristic algorithm is more successful for the setup case. Average percent deviation for the setup case is 0.57% and this value is more than two times smaller than the average percent deviation of the no setup case.

Heuristic algorithm is seriously more successful for the setup case when the number of jobs is considered as 5. While the grand average percent deviation is obtained as 0.81% for the setup case, the grand average percent deviation is obtained as 1.61% for the no setup case. Also, the maximum deviations can be seen from Table 9.

When the number of jobs is considered as 10, the heuristic algorithm is again more successful for the setup case. The grand average percent deviations are 0.96% and 2.00% for the setup and no setup cases, respectively. However, the grand maximum percent deviations are 6.94% and 9.35 % for the setup and no setup cases, respectively.

When the number of jobs is considered as 15, the grand average percent deviations are 0.48% and 1.03% for the setup and no setup cases, respectively. Although the difference between the grand average percent deviations is so different, it is seen that the grand maximum percent deviations are so close to each other.

The grand average percent deviations are closer to each other when the number of jobs is considered as 20. The grand average percent deviations are 0.04% and 0.32% for the setup and no setup cases, respectively. Among all grand average percent deviations, the smallest deviation is for the large-sized problems. This situation is also valid for the grand maximum deviations.

As a conclusion, the proposed heuristic algorithm is better perform for more large-sized problem sets, and is more successful for the setup case than the no setup case. Furthermore, the heuristic algorithm solves the problem in very short time for large-sized problems.

Table 14 Average and Maximum Percent Deviations for Setup and No Setup Cases

<i>N</i>	<i>K</i>	Number of Problem Instances	Setup Case		No Setup Case	
			Average Percent Deviation	Maximum Percent Deviation	Average Percent Deviation	Maximum Percent Deviation
5	5	25	0.36%	2.59%	2.26%	12.64%
	10	25	1.27%	7.66%	0.96%	7.93%
Grand Avg / Total		50	0.81%	5.12%	1.61%	10.28%
10	5	25	1.09%	9.34%	3.00%	14.73%
	10	25	0.82%	4.54%	1.00%	3.98%
Grand Avg / Total		50	0.96%	6.94%	2.00%	9.35%
15	5	25	0.52%	3.52%	1.55%	8.26%
	10	25	0.44%	7.11%	0.50%	3.71%
Grand Avg / Total		50	0.48%	5.31%	1.03%	5.98%
20	5	25	0.08%	1.62%	0.18%	1.76%
	10	25	0.00%	0.00%	0.46%	6.57%
Grand Avg / Total		50	0.04%	0.81%	0.32%	4.16%
Total Avg / Total		200	0.57%	4.55%	1.24%	7.45%

CHAPTER 6

CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

In this study, we consider a scheduling problem of several customer orders in which each customer can request a variety of products (also called jobs) in an order. All products are processed on a two-machine flow shop in which each product has one operation on each machine, and all products are first processed by machine 1 and then by machine 2. While processing a product, each customer order for a product is processed as a subplot (a batch of identical items of a product), all sublots of the same product must be processed continuously by the same machine, and each processed subplot of the same product is transferred from machine 1 to machine 2, while other sublots of the same product are processed on machine 1. This means that overlapping of the two operations on the same product through the creation of sublots (i.e., lot streaming) is allowed without intermingling the sublots of other products (i.e., once the first subplot of a product arrives at a machine, the other sublots of different products cannot be assigned to this machine until all of the sublots are processed).

Each customer order is delivered to the customer when the processing of all the products in the customer order is completed. Thus, the completion time of the job subplot processed as the last product in a customer order defines the completion time of the customer order. Our goal is to find a sequence of the jobs as well as the sequences of the sublots in each job so that the total completion time, which is the sum of the completion times of the customer orders, is minimized and equivalent to minimizing total work-in-process inventory focusing on increasing the customer satisfaction.

We have shown that the problem is NP-hard in the strong sense, and developed a mathematical programming model and a tabu-search heuristic algorithm to determine optimal and near-optimal solutions, respectively. We prepared two types of experiment set for the setup and no setup cases. From our experiments it is observed

that solving the problem by a standard MILP solver seems not to be useful alternative for both cases, especially for large-sized problem instances. While our proposed heuristic algorithm cannot give the successful results, it finds promising results for setup case. Moreover, the proposed heuristic algorithm can solve small- and medium-sized problem instances optimally and heuristic algorithm can find near-optimal solutions for large-sized problem instances in very short computational times.

There is a considerable number of issues remaining open for future research. Several extensions of our study can be investigated. In our study, we do not allow for the intermingling the sublots of different products (jobs). However, this assumption can be relaxed. Our problem studied in this study can also be extended for more complex machining environments such as flow shops having more than two machines, job shops, and open shops. Study of the same problem for different performance measures such as total or maximum lateness, total tardiness, and the number of tardy customer orders would be some other extensions.

REFERENCES

Ahmadi, R.H., and Bagchi, U. (1990), “Scheduling of multi-job customer orders in multi-machine environments”, ORSA/TIMS, Philadelphia.

Chang, J.H., and Chiu, H.N. (2005), “A comprehensive review of lot streaming”, *International Journal of Production Research*, 2005. 43: 1515-1536.

Çetinkaya, F.C., and Kayaligil M.S., (1992), “Unit sized transfer batch scheduling with setup times”, *Computers and Industrial Engineering*, 22(2): 177-182.

Çetinkaya, F.C. (1994), “Lot streaming in a two-stage flow shop with set-up, processing and removal times separated”, *Journal of Operational Research Society*, 45(12): 1445-1455.

Çetinkaya, F.C., and Gupta, J.N.D. (1994), Flowshop lot streaming to minimize total weighted flow time. Research Memorandum No. 94-24, School of Industrial Engineering, Purdue University, West Lafayette, Indiana.

Çetinkaya, F.C., and Duman, M. (2010), “Lot streaming in a two-machine mixed shop”, *International Journal of Advanced Manufacturing Technology*, 49: 1161-1173.

Defersha, F. M., and Chen, M. (2010), “A hybrid genetic algorithm for flowshop lot streaming with setups and variable sublots”, *International Journal of Production Research*, 48(6): 1705-1726.

Erel, E., and Ghosh, J.B. (2007), “Customer order scheduling on a single machine with family setup times: complexity and algorithms”, *Applied Mathematics and Computation*, 185: 11-18.

Feldmann, M., and Biskup, D. (2008), “Lot streaming in a multiple product permutation flow shop with intermingling”, *International Journal of Production Research*, 46(1): 197-216.

Gerodimos, A.E., Glass, C.A., and Potts, C.N. (2000), “Scheduling the production of two-component jobs on a single machine”, *European Journal of Operational Research*, 120: 250-259.

Glass, C.A., and Possani, E. (2011), “Lot streaming multiple jobs in a flow shop”, *International Journal of Production Research*, 49(9): 2669-2681.

Gonzales, T., and Sahni, S. (1978), “Flow shop and job shop scheduling: complexity and approximation”, *Operations Research*, 26: 36-52.

Gupta, J.N.D., Ho, J.C., and van der Veen, J.A.A. (1997), “Single machine hierarchical scheduling with customer orders and multiple job classes”, *Annals of Operations Research*, 70: 127-143.

Hall, N.G., Laporte, G., Selvarajah, E., and Sriskandarajah, C. (2003), “Scheduling and lot streaming in flowshops with no-wait in process”, *Journal of Scheduling*, 6: 339-354.

Hazır, Ö., Günalay, Y., and Erel, E. (2008), “Customer order scheduling problem: a comparative metaheuristics study”, *International Journal of the Advanced Manufacturing Technology*, 37: 589-598.

Johnson, S.M. (1954), “Optimal two- and three-stage production schedules with setup times included”, *Naval Research Logistics Quarterly*, 1: 61-67.

Jullien, F.M., and Magazine, M.J. (1990), ‘Scheduling customer orders: an alternative production scheduling approach’, *Journal of Manufacturing and Operations Management*, 3: 177-199.

Kalir, A.A. (1999), “Optimal and heuristic solutions for the single and multiple batch flow shop lot streaming problems with equal sublots”, PhD thesis, State University, Virginia.

Kalir, A.A., and Sarin, S.C. (2001), “A near-optimal heuristic for the sequencing problem in multiple-batch flow shops with small equal sublots”, *Omega*, 29(6): 577-584.

Kumar, S., Bagchi, T.P. and Sriskandarajah, C. (2000), “Lot streaming and

scheduling heuristics for m-machine no-wait flowshops”, *Computers and Industrial Engineering*, 38: 149-172.

Lee, I.S. (2013), “Scheduling on parallel machines to minimize maximum lateness for the customer order problem”, *International Journal of Production Economics*, 144: 128-134.

Leung, J.Y.T., Li, H., and Pinedo, M. (2005), “Order Scheduling in an Environment with Dedicated Resources in Parallel”, *Journal of Scheduling*, 8: 355-386.

Leung, J.Y.T., Li, H., and Pinedo, M. (2006), “Scheduling orders with multiple product types with due date related objectives”, *European Journal of Operational Research*, 168: 370-389.

Leung, J.Y.T., Li, H., and Pinedo, M. (2007), “Scheduling orders for multiple product types to minimize total weighed completion time”, *Discrete Applied Mathematics*, 155: 945-970.

Lin, B.M.T., and Kononov, A.V. (2007), “Customer order scheduling to minimize the number of late jobs”, *European Journal of Operational Research*, 183: 944-948.

Liu, C.H. (2009), “Lot streaming for customer order scheduling problem in job shop environments”, *International Journal of Computer Integrated Manufacturing*, 22: 890-907.

Liu, C.H. (2010), “A coordinated scheduling system for customer orders scheduling problem in job shop environments”, *Expert Systems with Applications*, 37: 7831-7837.

Mortezaei, N., and Zulkifli, N. (2013), “Integration of lot sizing and flow shop scheduling with lot streaming”, *Journal of Applied Mathematics*, vol. 2013, Article ID 216595, 9 pages, 2013.doi:10.1155/2013/216595.

Nawaz, M., Ensore, E., and Ham, I. (1983), “A heuristic for the m-machine, n-job flow shop sequencing problem”, *Omega*, 5: 11: 91.

Panwalker, S.S., and Khan, A.W. (1976), “An ordered flow-shop sequencing

problem with mean completion time criterion”, *International Journal of Production Research*, 14: 631-635.

Reiter, S. (1966), “A system for managing job-shop production”, *The Journal of Business*, 39: 371-393.

Sarin, S.C., and Jaiprakash, P. (2007), Flow shop lot streaming, Springer.

Smith, M.L., Panwalker, S.S., and Dudek, R.A. (1975), “Flowshop sequencing problem with ordered processing time matrices: A general case”, *Management Science*, 21:544-549.

Su, L.H., Chen, P.S., and Chen, S.Y. (2013), “Scheduling on parallel machines to minimize maximum lateness for the customer order problem”, *International Journal of Systems Science*, 44: 926-936.

Vickson, R.G. (1995), “Optimal lot streaming for multiple products in a two-machine flow shop”, *European Journal of Operational Research*, 85: 556-575.

Vickson, R.G. and Alfredsson, B.E.(1992), “Two- and three-machine flow shop scheduling problems with equal sized transfer batches”, *International Journal of Production Research*, 30: 1551-1574.

Wagneur, E., and Sriskandarajah, C. (1993), “Open shops with jobs overlap”, *European Journal of Operational Research*, 71: 366-378.

Wang, G., and Cheng, T.C.E. (2007), “Customer order scheduling to minimize total weighted completion time”, *Omega*, 35: 623-626.

Xu, X., Ma, Y., Zhou, Z., and Zhao, Y. (2015), “Customer order scheduling on unrelated parallel machines to minimize total completion time”, *IEEE Transactions on Automation Science and Engineering*, 12: 244-257.

Yang, J., and Posner, M.E. (2005), “Scheduling parallel machines for the customer order problem”, *Journal of Scheduling*, 8: 49-74.

Yang, J. (2005), “The complexity of customer order scheduling problems on parallel machines”, *Computers and Operations Research*, 32: 1921-1939.