YAŞAR UNIVERSITY
GRADUATE SCHOOL

MASTER THESIS

PARALLEL MACHINE SCHEDULING:

AN APPLICATION IN

APPAREL INDUSTRY

GÜLCE ÇİNİ

THESIS ADVISOR: PROF. (PHD) AYHAN ÖZGÜR TOY
SECOND ADVISOR: ASSOC. PROF. (PHD) ÖNDER BULUT

BUSINESS ENGINEERING

PRESENTATION DATE: 19.06.2023

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

Jury Members:	**Signature:**
Prof. (PhD) Ayhan Özgür TOY Yasar University	.....
Prof. (PhD) Yücel ÖZTÜRKOĞLU Yasar University	.....
Assoc. Prof. (PhD) Önder BULUT Yasar University	.....
Asst. Prof. (PhD) Adalet ÖNER Yasar University	.....
Asst. Prof. (PhD) Zehra DÜZGİT Istanbul Bilgi University	.....

Prof. (PhD) Yucel Ozturkoglu
Director of the Graduate School

ABSTRACT

PARALLEL MACHINE SCHEDULING: AN APPLICATION IN

APPAREL INDUSTRY

Çini, Gülce
MSc in Business Engineering with Thesis
Advisor: Prof. (PhD) Ayhan Özgür TOY
Second Advisor: Assoc. Prof. (PhD) Önder BULUT
July 2023

In this study, the sewing stage of a company from the apparel industry was examined, and six MILP scheduling models were developed. Since the machines are identical in terms of speed and other features, it has been determined that the system is appropriate for identical parallel machine scheduling problems. The first model is given as a simple maximum completion time (makespan) minimization. Next, the problem was extended by adding constraints such as sequence-dependent setup times, job splitting, job-ready dates, and machine eligibility. Finally, after introducing the job due dates, the objective function of the problem was defined as total earliness and tardiness. We proposed a Genetic Algorithm to solve MILP models. We solved the small-size problems via OPL Cplex and assessed the performance of GA by comparing the optimal solutions with GA solutions. We also designed a numerical study to test the algorithm for large problems. In real-life problems, the processing time of a job can vary due to factors such as operator learning curve, fatigue, and machine maintenance requirements. Likewise, setup times may vary depending on human and technical factors. Due to these factors, we also developed a model with fuzzy processing times and fuzzy setup times. The fuzzy optimization model was solved via a random search algorithm, which is a kind of Monte Carlo simulation.

keywords: parallel machine scheduling, makespan, earliness, tardiness, fuzzy logic, genetic algorithm, sequence dependent setup times, job splitting, ready date, machine eligibility, apparel industry

ÖZ

PARALEL MAKINE ÇIZELGELEME: KONFEKSIYON ENDÜSTRISINDE BIR UYGULAMA

Çini, Gülce
İşletme Mühendisliği Tezli Yüksek Lisans
Danışman: Prof. Dr. Ayhan Özgür TOY
İkinci Danışman: Doç. Dr. Önder BULUT
Temmuz 2023

Bu çalışmada, hazır giyim sektörüne ait bir firmanın üretim süreci incelenmiş olup, firma için altı farklı tam sayılı çizelgeleme modeli geliştirilmiştir. Sistemdeki makinelerin hız ve diğer özellikleri bakımından aynı olması nedeniyle, sistemin özdeş paralel makine çizelgeleme problemine uygun olduğu tespit edilmiştir. İlk model, basit bir maksimum tamamlanma süresi (makespan) minimizasyonu olarak verilmiştir. Ardından, sıraya bağlı kurulum süreleri, işlerin üretime hazır olma tarihleri, makine uygunluğu ve iş bölme gibi kısıtlar eklenerek problem genişletilmiştir. Son modelde ise, her işin teslim tarihi verilerek amaç fonksiyonu toplam erkencilik ve gecikme olarak belirlenmiştir. Geliştirilen son modeli çözmek için Genetik Algoritma önerilmiştir. Küçük boyutlu problemleri OPL Cplex ile çözdürdük ve optimum çözümleri GA çözümleri ile karşılaştırarak GA'nın performansını değerlendirdik. Algoritmayı büyük problemlerde test etmek için nümerik bir çalışma da tasarladık. Gerçek hayattaki problemlerde, bir işin işlem süresi, operatör öğrenme eğrisi, yorgunluğu ve makine bakım gereksinimleri gibi faktörlere bağlı olarak değişebilir. Aynı şekilde makine kurulum süreleri de insan ve teknik faktörlere bağlı olarak değişebilir. Bu faktörler nedeniyle, bulanık işlem süreleri ve bulanık kurulum süreleri olan bir model daha geliştirdik. Bulanık optimizasyon modeli, bir tür Monte Carlo simülasyonu olan rastgele arama algoritması ile çözdürülmüştür.

Anahtar Kelimeler: özdeş paralel makine çizelgeleme, makespan, erkenlik, geçlik, bulanık mantık, genetik algoritma, sıraya bağlı kurulum süresi, iş bölünmesi, hazırlık tarihi, makine uygunluğu, konfeksiyon endüstrisi

ACKNOWLEDGEMENTS

I would like to thank my thesis advisor Prof. (PhD) Ayhan Özgür Toy and Assoc. Prof. (PhD) Önder Bulut for their knowledge, guidance, support, and patience during this study. They constantly encouraged me to continue my thesis and believed that I would complete my thesis successfully.

I would especially like to thank Asst. Prof. (PhD) Hande Öztop and Asst. Prof. (PhD) Adalet Öner, whenever I wanted to ask a question, they always answered me. Thank you for supporting me.

I appreciate my jury members Prof. (PhD) Ayhan Özgür Toy, Assoc. Prof. (PhD) Önder Bulut, Asst. Prof. (PhD) Adalet Öner, Prof. Dr. (PhD) Yücel Öztürk, and Asst. Prof. (PhD) Zehra Düzgit, and for accepting to be a jury in my thesis, their many insightful comments, suggestions, and contributions.

Finally, I would like to thank my mother Selda, my father Nevzat, and my friend Berke, who supported and motivated me throughout the years I wrote my thesis. Special thanks to my dear grandmother Sevim, who supported me with her prayers and encouraged me to finish my thesis. I also would like to thank my cat Güneş, who positively influenced me and motivated me with her energy.

I certainly would not have been able to do this without the unfailing support and constant encouragement of everyone I thank.

Gülce Çini

İzmir, 2023

TEXT OF OATH

I declare and honestly confirm that my study, titled “Parallel Machine Scheduling: An Application in Apparel Industry” and presented as a master’s Thesis, has been written without applying to any assistance inconsistent with scientific ethics and traditions. I declare, to the best of my knowledge and belief, that all content and ideas drawn directly or indirectly from external sources are indicated in the text and listed in the list of references.

Gülce Çini

11.07.2023

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

SYMBOLS AND ABBREVIATIONS

ABBREVIATIONS:

GA Genetic Algorithm

MILP Mixed Integer Linear Programming

P Identical Machines

R Unrelated Machines

Q Uniform Machines

SYMBOLS:

i, j, k Jobs indices.

m Machine index.

f Facility index.

J Set of job indices.

J' Set of job indices including the dummy job 0.

M Set of machine indices.

F Set of facility indices.

M_i Index set of machines that can process job i.

M_f Index set of machines at facility f, $M_f \subset M$.

M_{if} Set of machine indices that are eligible to process job $i \in J$ at facility f, $M_{if} \subset M_f$

P_i Processing time of one unit of job i.

S_i Setup time of job i.

S_{ij} Setup time of job j when immediately precedes job i.

d_i The due date of job i.

Q_i The number of units in job i.

q_{im} The number of unit jobs of job i processed on machine m.

C_{im} The completion time of job i on machine m.

C_{max} Makespan, the maximum completion time of all jobs.

T_i Tardiness of job i.

E_i Earliness of job i.

ps Population size.

n Number of iterations.

L A very large number.

r_i The ready date of job i.

CHAPTER 1
INTRODUCTION

In the current competitive age, the performance of companies is measured by their success in achieving targets. Companies set their targets as profitability, efficiency, on-time delivery, and most importantly customer satisfaction. An effective and efficient planning is the first and the most important step for setting targets and tracking performances. Customers, suppliers, infrastructure, manpower, materials, and third-party service providers are the major components of the system. The management of these components is the determinant for the company's success. For their competitive advantage, companies need to be aware of the issues such as production capacities, lead times, and the quality of their output along with the reflection of these issues on their costs. Companies focus mostly on cost reduction techniques for achieving price advantage. The key approach for cost reduction is waste elimination. Time is a resource which cannot be stored or recovered. The tool for elimination of time waste is to make an efficient scheduling based on a sustainable system. As defined by Pinedo (2012), scheduling is an approach to making choices that are used in both the manufacturing and service sectors. The purpose of scheduling is to assign resources to tasks within specific time frames in order to optimize one or multiple objectives. The scheduling literature is quite diverse and covers both deterministic and uncertain cases, single-machine and multi-machine problems, and static and dynamic problems. According to the classification made by According to Nagar et al. (1995), the problem is able to be described as a deterministic schedule problem when parameters such as the processing times of jobs are able to be described as certain; otherwise, it is classified as stochastic. The parameters in stochastic problems are described as random variables with a probability distribution. Generally speaking, stochastic models are more accurate in simulating real-life scenarios than deterministic models. Additionally, according to the number of machines in the system, various structures are obtainable: a single-machine problem is when one machine processes all the jobs, whereas a multi-machine problem is when jobs need to be processed by more than one machine and

can be categorized as a single machine, parallel machine, flow shop, or job shop. Wojakowski et al. (2014), following Nagar et al. (1995), depicts the classification of scheduling problems as in Figure 1.1.

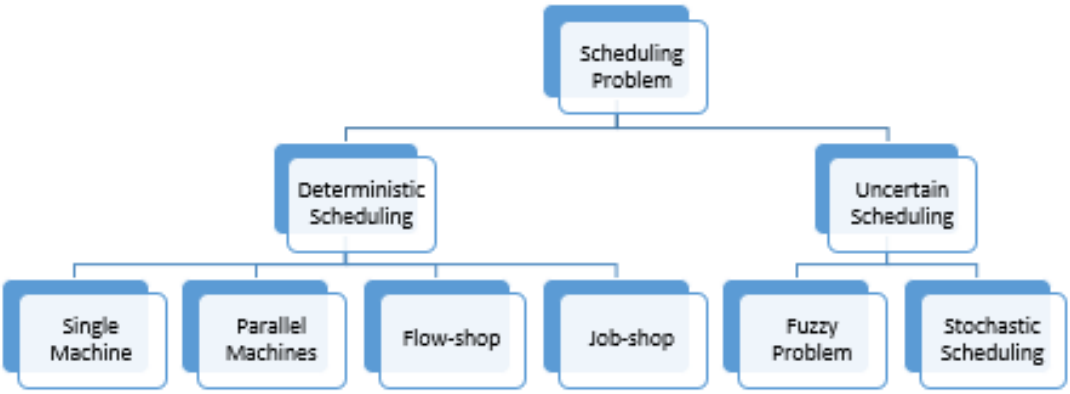

Figure 1.1. The Classification of Scheduling Problems

Research questions in this thesis stem from a real-life problem, specifically from the needs of the production planning department of a textile company. With the desired effective and efficient scheduling, it is aimed to convey the demands of the customers on time. In this direction, the production of the existing system has been examined and it has been seen that the production is suitable for parallel machine scheduling, as it will be clear in the rest of the thesis. However, three different kinds of parallel machine problems *identical machines, uniform machines,* and *unrelated machines* are recognized in the literature. For parallel machines, Berthier et al. (2022) stated the differences between these categories as follows: Each job's processing time is the same across all machines in the system for identical parallel machines. For uniform machines, there are differences between the processing times of the same job on different machines and are affected by the machine speed factor. Machines that are unrelated to one another have many different processing times, even though they perform the same jobs. In these types of problems, machines may be set up for the upcoming job which requires a certain time called machine setup times. Depending on the sequence of the jobs, machine setup times may or may not change. In this study, we assume that the times required to set up the machine depends on the jobs that are finished on the machine and the job in order, hence sequence dependent.

An illustration of parallel machine scheduling including setup times is shown in Figure 1.2. in this example there are 5 different jobs and 3 machines (lines). Job A and B are processed on machine 1. First Job B produced, then Job A produced, and a setup time

occurred between 2 jobs. In machine 2, Job C and D are processed. First Job D produced, then Job C produced. There is a setup time between them. In Machine 3, Job E is processed and there is no setup.

Jobs: A (5.000 pcs.), B (10.000 pcs.), C (7.500 pcs.), D (15.000 pcs.), E (15.000 pcs.)

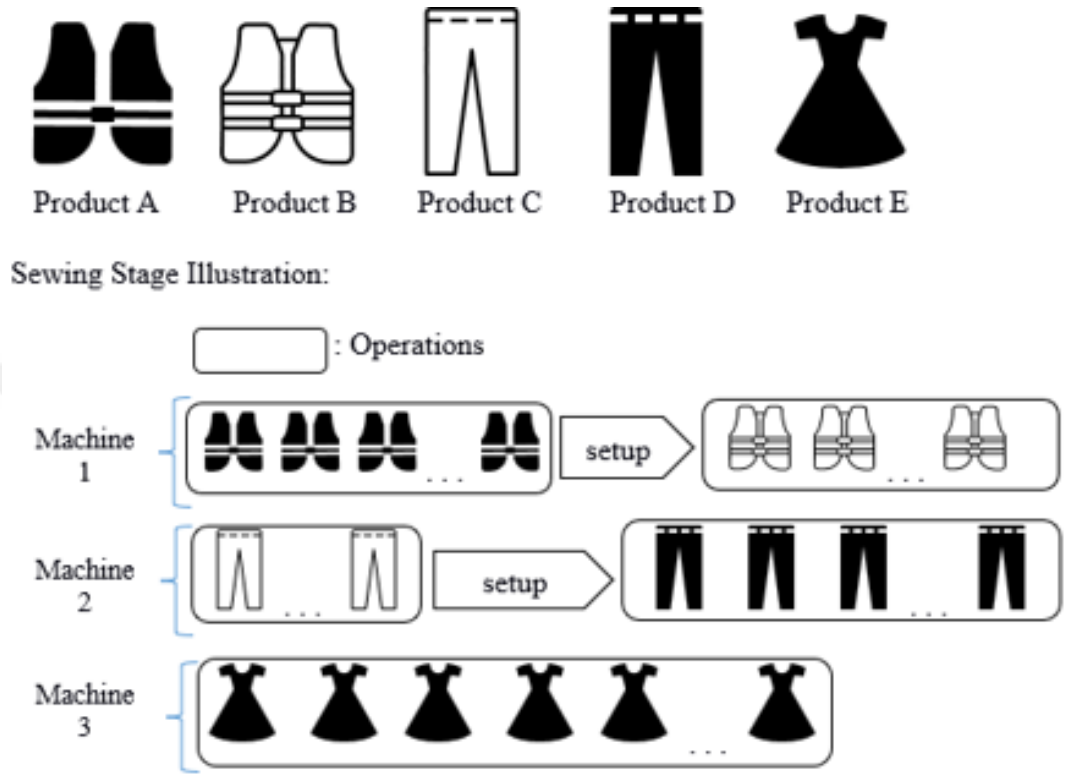

Figure 1.2. An Illustration of Parallel Machine Scheduling with Setup Time

Unlike most studies of known models, approaches, or techniques, the emphasis here is on gaining a multifaceted approach by adding problem-specific constraints. A lot of research was done before finally developing the ideas presented here. Most importantly, it was necessary to understand the real dynamics of a textile company, examine the existing system, and determine how to handle the problem according to the scheduling literature. In the following parts of the article, for a better comprehension of the system, we talk about the company structure.

The company we consider has a total of ten production facilities. It also has sales offices in USA, England, and the Netherlands. The factories of the company are divided into two, sewing and cutting. Currently, there are seven sewing factories and three cutting factories. Jobs are assigned to the cutting factories first, and then transferred to the factories for sewing. There are also factories abroad in Moldova and Bulgaria. However, those factories make their own production. It is not included

in our problem. They have four product groups which are changing the fabric qualities such as single jersey, fleece, underwear fabrics and others. In these groups, most of the production is provided by single jersey and fleece fabrics. While fabrics called single jersey are used mostly for t-shirt sewing, fabrics called fleece are mostly used in products such as sweatshirts, sweatpants, and hoodie. You can see the details of the product groups in Table 1.1. Product Group and Order Type.

Table 1.1. Product Group and Order Type

Product Group	**Order Type**
Single Jersey	Short or Long Sleeve T-shirt
Fleece	Sweatshirts, Hoodie, Sweatpants, Sweatshorts, Dress
Underwear	Trunk, Shorty, Bralet, Tight
Others	Skirt, Dress, Short, Zipthrough, Polo

Firstly, customers place their orders by specifying the colors and products along with their name and the sizes. Customers share a spreadsheet with the information on how many pieces of each color and size they have with the due dates. We refer to each color of each product as a *job*. The shipping date of different colors of the same product may differ. Table 1.2. is an example of a customer order, where the first and the second column are the product name and colors requested.

Table 1.2. A Sample of Customer Order Table

Product Name	Colorways	Total Qty.	Confirmed Date
Product A	Smooth Stone	7.350	3.May.21
Product A	Black	18.000	3.May.21
Product A	Liberty Blue	4.750	2.Apr.21
Product A	Twilight Navy	13.100	2.Apr.21
Product A	White	18.050	2.Apr.21
Product A	Pollen	7.250	2.Apr.21
Product B	Frigid Blue	2.000	3.Mar.21
Product B	White	10.000	3.Mar.21

In this example Product A has 6 colorways. The shipping date of the smooth stone and black color of this product is 3rd of May while the shipping dates for the remaining colors are 2nd of April. As you can see, the shipping dates may change according to the colorways of a product. Although some jobs have a common shipping time, the arrival time of each fabric of job may vary. Therefore, it may not be possible to process jobs that use the same product but a different color of fabric at the same time. In addition, arrival times of orders may vary within themselves. While the orders of some brands (customers) are periodically and regularly, the arrival time of the orders of some other brands may be sporadic. For seasonal orders, customers share the information on their order forecasts on a table for material pre-preparation. This table includes the information on the size of the order for each product and their required shipping dates. There are some products that the company makes every season, which they call carry-over products. The patterns and fabric quality of these products are the same every season. Upon receiving the customer order forecasts the company orders fabric and trim for the carry-over products first. Then the customers share their second order forecast table which initiates the fabric and trim orders for other products (without carry-over products). The lead-time for the fabric and trims delivery is between 6 and 8 weeks. Therefore, orders for fabrics and trims are initiated early. Before the jobs are assigned to the sewing line, all trims and fabrics must be ready.

A line is the group of operations needed to sew a job as given in Figure 1.3. In some cases, even if the fabrics arrive on time their quality may be substandard so that the supplier needs to repair or refurbish which takes another 1 or 2 weeks. In this case, the plans are updated again. Then the exact shipping dates, product names, and their order quantities are shared clearly in the final order tables. When the final order table arrives, planners create a sewing and cutting plan for each job.

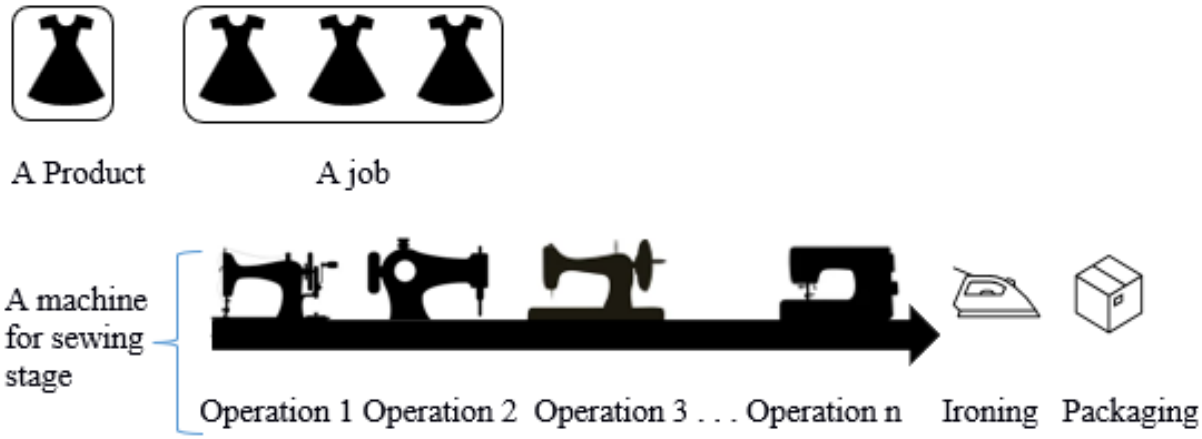

Figure 1.3. Illustration of a Product, Job, Machine, and Operation

Customer orders go through several major production stages, such as cutting, printing, embroidering, washing, sewing, ironing, packaging. In terms of planning activities sewing is the dominant stage. Hence, in this study we focus on sewing. Upon their arrival, fabrics, i.e., one of the raw materials, are first checked for any defects, which is done on cutting tables. If fabrics are approved the products are cut according to their patterns on cutting tables. The sequence of the stages may be as follows based on the design:

1) Cutting, printing & embroidering, sewing, ironing, packaging
2) Cutting, sewing, washing, printing & embroidering, ironing, packaging
3) Cutting, printing & embroidering, sewing, washing, ironing, packaging

The second type of raw materials, the trims, are distributed to the sewing machines depending on the product assignment of the sewing machines. When the sewing of an item is finished, it is taken to the ironing and packing stages. Right after the last stage the quality control is performed before they are shipped to the customers.

Several jobs can be treated as a single job if they are the same products but with different colors, and their raw material (fabrics and trims) are ready simultaneously since the sequence of their production stages are the same and no setup is required when switching. The time required for processing one unit of a job is calculated beforehand based on an expert opinion at the sampling phase for customer confirmation. Operation types and numbers to be used in the sewing machines are also identified at that phase. The processing time estimated by the expert does not incorporate downtimes due to factors such as operation breakdown, bobbin change or

needle breakage, therefore a margin for the tolerance is added. The company standard for the tolerance margin is 15%. A job's daily production quantity is determined by the yield rate. The yield rate is a function of the number of independent operations that a single item goes through. For example, suppose that the processing time, p, for a single t-shirt takes 10 minutes; the company has a total working time, w, of 520 minutes per day; and there are 10 operations, r, in a sewing machine, then the daily yield rate is $(w/p)r = (520/10)10 = 520$. If the size of a job contains 18,000 pcs., the time required for this job is 18,000 pcs. /520 days. Note that the company does not allow a bottleneck to arise in a line by adjusting the speed of each operation by assigning more workers. Hence, the product flow over the machine is smooth.

On the same machine, different products can be sewn one after the other, and there is a setup for machines for switching jobs. The reason for this is that different operation types are used in each job. If there is an unnecessary operation for the next job in the current machine layout or if there is a need to add an operation, the machine is reinstalled accordingly. In addition, settings for new operations, thread checks, and needle adjustments are made for the next job. Therefore, setup time can vary from job to job. However, the "product groups" and the production sequence of the "order types" in the machines are what affect how long this setup time can take (see details of the product group and order type in Table 1.1.). For instance, when the product group is single jersey & the order type is "short sleeve t-shirt" and the product group is fleece & the order type is a "sweatshirt" are sewn consecutively in the same machine, need to add extra operation to the machine. Another factor affecting the set-up time is operator competence. Operators should also be assigned to machines according to their competencies. Operating an operation requires expertise, and not all operators can use all operation types. For example, while an “overlock” operation will be used in a job, a “flatlock” operation can be used in another job. Therefore, operators who can use these operations should be assigned to those machines. For this reason, the types of operations to be used in the workmanship of each job and the number of operators required are determined in advance. Accordingly, machine set-ups are made according to the number of operations and operators that should be used in each job. However, if a different product t-shirt is given after the order is completed on a t-shirt sewing machine, it is almost like the operation types to be used the same, so no setup is required.

Not every job can be produced on every machine. Jobs can be produced on eligible identical machines. Therefore, machine eligibility is available in the production system. In addition, the company has more than one factory; each job must be processed in only one factory and processed in the eligible machine where it can be produced in the factory where it assigns. We mentioned that the fabric and trim orders are started at the suppliers, before starting the production stage. During the planning phase, scheduling is made according to the ready date of the raw materials of the jobs at the sewing machines. The "ready date" of the jobs means the production phase can be started and the raw materials are ready in front of the relevant machine on this date. After the production is started, the next job is not taken to the machine until the current job is finished. Therefore, the problem is not suitable for pre-emptive production. If the size of the job is high and the due date is very close, the job can be produced on different machines to send it on time. This case can also be used to prevent production from stopping on the machines. If the capacity is not well managed, production stops on some machines. To prevent this, they can produce on these machines by dividing large-size jobs. In this case, "job splitting" can also be performed to prevent production stoppages. Job splitting has been investigated, according to Xing and Zhang (2000), on identical parallel machines. The research presented in this work stems from the requirement to "split" specific jobs due to factors including high levels of priority, large order sizes, and lengthy completion times.

Customers have many different suppliers around the world apart from this company. They make performance analysis for each supplier by evaluating the number of orders shipped each season, the number of days delayed in the orders, and their ability to send the requested quantities. Depending on this performance, they increase and decrease the number of orders to be given each season. If it has a poorly performing supplier in a season (not shipped on time, production under-order quantity, etc.), they expect to improve the supplier's performance by decreasing the order numbers the following season. The firm’s main goal is to succeed in on-time performance, take more orders, use its capacity efficiently, and thus make the firm a leader among other suppliers of the customers. Until this part, we have handled the structure of the production system and how the company can be affected if there is no on-time production. If we mention the early production phase of jobs, in some cases, production can be made months

before the shipment. The reason for this is that the fabrics and trims are ready long beforehand. If the machine is also available, the production of the job can be started. After the completion of the production of the job, permission is obtained from the customer for early shipment. Sometimes the customer can accept the early shipment and keep the products in their stock. Sometimes customers may request that it be kept in the firm's warehouse and shipped on time. However, if the sewn jobs are kept for a long time, there may be shrinkage on the fabric. To prevent this situation, after the jobs are ready, the quality approval of the customer is taken and waiting for the warehouse.

When the system was examined, each job can be completed on any machine that can be completed at the same time. The problem is broadened with different constraints and is appropriate for scheduling identical parallel machines. Various mathematical models have been developed by considering the problem in different scenarios. In total, six different models were developed. The details of the models are given in Chapter 3. The models were created by adding different objective functions and constraints, starting from the simplest version of the problem. The first model was created to minimize makespan, while the last model was created to minimize total earliness and tardiness. Since we examined the problem in real life, the model was also examined using the fuzzy logic approach as a problem extension. Some factors affect the processing times of jobs in real life such as operator learning curve and fatigue, and machine maintenance requirements. In the fuzzy approach, modeling is developed by assuming fuzzy processing times and fuzzy setup times. Due to the scheduling problem's inherent nature of growing in difficulty with the number of jobs and machines, the large-scale identical parallel machine scheduling problem shows several difficulties. In order to address challenging optimization issues, particularly those that fall under the purview of NP-hard, a variety of heuristic search techniques have been developed over time. Since the problem is in the NP-hard class, we approached the problem intuitively and coded it with a genetic algorithm and a randomized search algorithm. Thus, we gained the ability to solve big problems. For the MILP model, a genetic algorithm was applied. The randomized search algorithm, Monte Carlo simulation, was applied for fuzzy MILP. Then the consistency of the resulting values was compared. The randomized search algorithm's and genetic algorithm's performances were analyzed. Because of their high efficiency and widespread use,

these methods have shown to be beneficial in tackling the combinatorial optimization problem.

This thesis' remaining sections are structured as follows: The scheduling problem literature study is presented in Chapter 2, and the problem models are defined in Chapter 3. The solution methods, including genetic algorithms and randomized search algorithm approaches, are presented in Chapter 4. Chapter 5 presents the findings of computational experiments and comparisons. The conclusion and potential outcomes are presented in the last chapter, Chapter 6.

CHAPTER 2
LITERATURE REVIEW

Before discussing the identical parallel machine scheduling problem in this thesis, a thorough literature review was undertaken. Inspired by similar studies in literature, a new mathematical model was created by adding constraints specific to the problem. When creating this model, different model approaches were created by adding company-specific constraints such as sequence-dependent set-up times, job splitting, and machine eligibility. After researching the literature for each feature, mathematical models were created in accordance with scheduling terminology. Two different objective functions were used which are minimizing makespan and total earliness and tardiness. The problem also was analyzed with fuzzy logic approach and the objective function was determined as minimizing the fuzzy makespan. Since the problem is in the NP-hard class, what was done in the literature as a solution algorithm was examined. An algorithm based on genetics and randomized search was then used to solve the problem.

Graham et al. (1979) show triplet notations for scheduling problems as alpha(α), beta(β), and gamma(γ). (α) relates to machine environments, (β) is for the problem features, and (γ) defines the main function. These notations are given below Table 2.1., and these notations are used in the paper. You can find the summary of the papers inspired by the literature below and in Table 2.2.

Table 2.1. The Description of Notations of Used in Scheduling Problem

α		**γ**	
Notation	**Description**	**Notation**	**Description**
1	Single machine	C_{max}	Makespan
P	Parallel machines (identical)	E_{max}	Maximum earliness
Q	Parallel machines (uniform)	L_{max}	Maximum lateness
R	Parallel machines (unrelated)	T_{max}	Maximum tardiness
F_m	m-stage flow shop	$\sum C_i$	Total completion time
J	Job shop	$\sum E_i$	Total earliness
FJ	Flexible job shop	$\sum T_i$	Total tardiness
O	Open shop	$\sum U_i$	Number of tardy(late) jobs
β		$\sum w_i C_i$	Total weighted completion time
Notation	**Description**	$\sum w_i U_i$	Weighted of tardy(late) jobs
ST_{si}	Sequence-independent setup time	$\sum w_i E_i$	Total weighted earliness
ST_{sd}	Sequence-dependent setup time	$\sum w_i T_i$	Total weighted tardiness
Prec	Precedence constraints		
r_i	Non-zero release date (ready times)		
d_i	Due date		
split	Job splitting		
M_i	Machine eligibility		

2.1 MILP Literature Review

Fanjul-Peyro et al. (2019) and Rocha et al. (2008) addressed an unrelated parallel machine schedule. The most essential constraint is sequence-dependent setup times. Fanjul-Peyro et al. (2019) also suggested a mathematical approach in order to reduce makespan. They proposed a new algorithm. Then, solutions were compared with Cplex. Rocha et al. (2008) investigated minimizing the weighted tardiness and makespan as the objective function. In order to compare the results, they proposed a branch and bound approach and used a mixed integer programming model.

With identical parallel machine scheduling, Unlu and Mason (2010) developed four formulas for mixed integer programming to determine the total weighted tardiness, total weighted completion time, makespan, maximum lateness, and the total number of tardy jobs.

To determine the makespan that is smallest, Mokotoff (2002) studied the standard deterministic identical parallel machine problem with fixed processing time and without preemption. To obtain the best results, they suggested branch and bound (B&B) strategies.

For parallel machine scheduling with machine eligibility, Edis and Ozkarahan (2011) integrated integer and constraint programming techniques. They created an analytical framework.

Tahar et al. (2006) addressed an identical parallel machine problem to find the makespan by using sequence-dependent setup times and job splitting. They make a new heuristic algorithm recommendation and then compare it with the optimal solutions.

In order to shorten the manufacturing timeline for pistons, Kim and Lee (2020) considered problems with uniform parallel machine scheduling involving sequence-dependent setup times, machine eligibility, and job splitting constraints. They used certain heuristic techniques after figuring out the mathematical programming model. Then, they experimented with various machine setups and job counts. They want to approach the problem using meta-heuristic algorithms such as SA, GA, and tabu search in future investigations.

Omar and Teo (2006) studied identical parallel machine scheduling problems by using distinct due dates, and process time to find the minimizing sum of earliness and tardiness. modeled on a set of identical parallel machines, considering the Early Due Date (EDD). Another feature addressed by this research is that early penalties and delay penalties were applied to each job. In further research, they want to apply a heuristic method for solving large instances.

Su (2007) looked at an alternate paper on reducing early and late arrival using a parallel strategy. On m identical parallel machines, a set of n jobs must be scheduled on the assumption that they are all available at time zero. There are two distinct objective functions. One of them is to identify ways to minimize overall flow time, and another is to minimize the sum of the jobs' early and late arrivals, $P||(E+T)/\sum C_i$. They have tested the model with many different samples and this model has proven to be efficient and robust.

Ekici et al. (2019) investigation of the real-world production scheduling issue. After receiving a customer order, the company plans its manufacturing. Each order has a unique due date, release date, and order size. On one of the suitable machines, each order is processed. The goal of unequal release timings, sequence-dependent setups, machine-job compatibility limitations, and workload balancing requirements in unrelated parallel machine scheduling is to minimize the earliness and tardiness of jobs. This is because customer happiness is a top priority. They put forth a few sequential heuristic techniques that are quite understandable, such as the random set partitioning method, the tabu search algorithm, and the tabu search-based metaheuristic.

A MILP model was created by Hamzadayi and Yildiz (2017) to solve the challenge of scheduling m identical parallel computers. For the huge, scaled issues, Hamzadayi and Yildiz (2017) suggested genetic and simulated annealing techniques depending on the NP-hardness class and assessed performance using dispatching criteria, shortest processing time first (SPT) and longest processing time first (LPT). Consequently, the GA performance offers a more potent performance.

Using batch splitting for non-related parallel machine scheduling, Bastos and Redendo (2020) explored sequence-dependent setup times and created a mixed integer linear programming model (MILP).

To minimize total tardiness, Mensendiek et al. (2015) examined identical parallel machines with defined delivery dates jobs. To determine the best route for a solution, they used the branch and bound method. They suggested an assignment-oriented tabu search and a hybrid genetic algorithm to address the NP-hardness.

Biskup et al. (2008) studied identical parallel machines to find minimizing total tardiness, $P/\sum T_i$. Heuristic approaches were proposed for the solution of large instances. According to Biskup et al. (2008), preemption, division, or cancellations are not permitted, and all jobs are presumed to be ready at time zero. Then, Biskup et al. (2008) examined heuristics, meta-heuristics, and optimization techniques.

2.2 Fuzzy MILP Literature Review

Ahmadizar and Hosseini (2011) studied single-machine scheduling with a position-based learning effect, and they use fuzzy processing times. Processing times are

regarded as triangular fuzzy numbers in the problem. The problem, whose objective function is to minimize the total completion time, is given a polynomial time algorithm.

Yeh et al. (2014) analyzed the parallel machine scheduling problem with learning effects to minimize the makespan. They used fuzzy processing times. They suggest heuristic algorithms which are simulated annealing algorithms and genetic algorithms.

Parallel machine scheduling with fuzzy due dates and fuzzy processing times was researched by Peng and Liu in 2004. The fuzzy maximum completion time, the fuzzy maximum tardiness, and the fuzzy maximum idleness are the three different goal functions that they developed.

Li et al. (2019) proposed the uniform parallel machines problem. The objective function is to minimize makespan. For adapting to the real world, they use triangle fuzzy numbers. Since the problem NP was hard, they worked on a fuzzy simplified swarm optimization (SSO) algorithm. They compared the performance of the algorithm with the genetic algorithm.

The formulation of the expected amount for a function of fuzzy variables was taken into consideration by Xue et al. (2008). They demonstrated how the outcome matches the value expected for the function of a random variable.

In a single machine scheduling problem, Wang et al. (2002) explored the ready-time scheduling problem with the fuzzy processing time that is represented by a triangular fuzzy number. There is no idle time between jobs, the timetable is non-preemptive, and all jobs must be scheduled.

2.3 Genetic Algorithm Literature Review

Vallada and Ruiz (2011) developed an evolutionary approach to the problem of scheduling unrelated parallel machines with sequence-dependent setup durations to lower the maximum completion time between jobs. For each machine, they display the issue as a variety of tasks. A competition is used as the selection procedure. The crossover technique is an adaptation of one point crossover to the problem of scheduling parallel machines. Local search techniques are used as the metaheuristic way for enhancing the solutions.

Yu et al. (2018) employed a genetic algorithm for hybrid flow shop scheduling with unrelated machines and using machine eligibility. Even if the capacity of the machines

is not limited, each machine only processes one task at a time, every task is only processed on one machine at a time, and setup times are considered and added to processing times. A series of genes serve as the depiction of the answer. Scheduling jobs is represented by a chromosome. As a result, compared to previous heuristic methods, GA addressed the multi-objective issues with more realistic presumptions.

To reduce early arrival and late arrival penalties, Haghnevis et al. (2006) looked at the issue of multi-criteria unrelated parallel machine scheduling difficulties. For this issue, preemption is prohibited. Each job has distinct due dates that are given by customers. Haghnevis et al. (2006) developed the MILP model and solved the problem for small sizes data. Then, a genetic algorithm was used, and binary coding representations are used for chromosomes. As a mutation, they used two different operations such as randomly selected jobs and machine-based ones. As a crossover, they used two different operations as simple and partial crossover methods.

Joo and Kim (2012) investigated non-identical parallel machine scheduling using sequence-dependent set-up times. In this study, a genetic algorithm known as a heuristic algorithm was presented. They used an extremely popular solution representation which included machine assignments and job sequences. The used "*" character is for separating the machines. Crossover is applied by one point approach, while mutation is applied by swapping that randomly selected between two different jobs. This paper's genetic algorithm approach is close to our problem.

Fığlalı and Engin (2011) presented the general approach to genetic algorithms in flow shop scheduling. They explained in detail the parameters used in the genetic algorithm and the operators that can be used for crossover. helped us choose the suitable crossover method for our problem.

Körez (2005) gave detailed information about genetic algorithms and explained mutation and crossover methods through examples. Körez (2005) published detailed work to understand the logic of genetic algorithms.

For the non-identical parallel machine scheduling using setup times, ready times, and due dates, Balin (2011) focused on minimizing the makespan problem. A novel crossover operator and a new GA optimality criterion were suggested by Balin (2011). Then, Balin (2011) solved the problem by using different examples and concluded that the algorithm for the problem works correctly and efficiently.

To determine how to reduce the makespan, Ruiz and Vallada (2011) looked at the scheduling problem for unrelated parallel machines. A genetic algorithm approach was proposed in this work. A Mixed Integer Programming (MIP) mathematical model with sequence-dependent set-up times was created by them. The new crossover in the genetic algorithm makes use of a local search method.

Table 2.2. Summary of Literature Review

References	Problem	Approach
Fanjul-Peyro et al. (2019)	$R/ST_{sd}/C_{max}$	Unrelated parallel machine scheduling problem; MILP model, new algorithm
Rocha et al. (2008)	$P/C_{max}, \sum w_i\ T_i,$	Identical parallel machine scheduling problem; MIP model, B&B
Unlu and Mason (2010)	$P/\sum w_i\ C_i, \sum w_i\ T_i,$ $L_{max}, \sum U_i$	Identical parallel machine scheduling problem; MIP models
Mokotoff (2002)	P/C_{max}	Identical parallel machine scheduling problem; MIP model, B&B
Edis and Ozkarahan (2011)	$P/M_i/C_{max}$	Identical parallel machine scheduling problem; MIP model
Tahar et al. (2006)	$P/ST_{sd}, split/C_{max}$	Identical parallel machine scheduling problem; MIP model, new heuristic approach
Kim and Lee (2020)	$Q/ST_{sd}/C_{max}$	Uniform parallel machine scheduling problem; MIP model, heuristic algorithm
Omar and Teo (2006)	$P/\sum(E_i + T_i)$	Identical parallel machine scheduling problem; MILP model
Su (2007)	$P\|\|\sum(E_i + T_i)/$ $\sum C_i$ & $P\|\|C_{max}/\sum c_i$	Identical parallel machine scheduling problem; MILP model
Ekici et al. (2019)	$R/ST_{sd}/\sum(E_i + T_i)$	Unrelated parallel machine scheduling problem; MILP model- tabu search, matheuristic
Hamzadayi and Yildiz (2017)	$R/\sum C_i$	Non-related parallel machine scheduling problem; MILP model, proposed heuristic algorithms
Mensendiek et al. (2015)	$P/\sum T_i$	Identical parallel machine scheduling problem; MILP model, branch and bound, metaheuristics
Biskup et al. (2008)	$P/\sum T_i$	Identical parallel machine scheduling problem; MILP model, proposed heuristic algorithms

Vallada and Ruiz (2011)	$R/ST_{sd}/C_{max}$	Non-identical parallel machine scheduling problem; MIP model, genetic algorithm, local search algorithm
Yu et al. (2018)	$HF/M_j/\sum T_i$	Hybrid flow shop scheduling; MILP model, GA
Haghnevis et al. (2006)	$P/ST_{sd}/\sum(E_i + T_i)$	Multi-criteria parallel machine scheduling; MILP model, GA
Joo and Kim (2012)	$R/ST_{sd}/M_j/C_{max}$	Non-identical parallel machine scheduling problem; MIP model, meta-heuristic algorithms (GA)
Fığlalı and Engin (2011)	P/C_{max}	Flowshop scheduling- GA
Körez (2005)	P/C_{max}	Flowshop scheduling-tabu search, GA, simulated annealing
Balin (2011)	R/C_{max}	Non-identical parallel machine scheduling problem; GA
Ruiz and Vallada (2011)	R/C_{max}	Unrelated parallel machine scheduling problem; MIP model- GA
Ahmadizar and Hosseini (2011)	$1/\sum(E_i + T_i)$	single-machine scheduling problem; fuzzy MIP model, fuzzy triangular processing times, the shortest processing time rule
Yeh et al. (2014)	P/C_{max}	Parallel machine scheduling problem; fuzzy MIP model, fuzzy processing times
Peng and Liu (2004)	$P/C_{max}, L_{max}, \sum T_i$	Parallel machine scheduling problem; fuzzy MIP model, fuzzy processing times, fuzzy due dates
Li et al. (2019)	Q/C_{max}	Uniform parallel machine scheduling problem; fuzzy MIP model, fuzzy processing times, SSO
Xue et al. (2008)	Fuzzy modelling	The formulation for a function of a fuzzy variable
Wang et al. (2002)	Fuzzy modelling	a ready time scheduling problem with fuzzy job processing time

CHAPTER 3
MODELS

In the general structure of the production system, there are m identical lines in parallel which are called machines. There are many different products. Customers place high volume orders of them. We called each color of each product a job. J refers to the set of all job indices while, M refers to the set of all machine indices in this problem. Every job has a date when it can start production. We consider this date when the raw material (trim and fabric) of each i job arrives in production. We call this a ready date, and the notation is r_i. Any job i can be processed on any eligible machine. The problem has machine eligibility constraints by defining the eligible machine set for any job i as M_i. Also, the structure of the system is therefore suitable for parallel identical machines. There is a setup between two jobs which are i and j processed consecutively. This problem has sequence dependent setup time, and the notation is S_{ij}. While creating the models, we define the units in job i with Q_i. In addition, we give the processing time of one unit of job i with P_i. While creating the models, we define the units in job i with Q_i. In addition, we give the processing time of one unit of job i with P_i. Also, each job i has a due date and the notation is d_i. When developing the model, we use the due date of each job to find out how tardy or how early each job is finished. We define the early completion of a job i with E_i, and the tardy completion with T_i.

The problem's assumptions are given below.

Assumption 1: The company has production lines which are called "machines" and a high-volume order of each product and color which is called a "job" to comply with the scheduling terminology.

Assumption 2: Preemption is not allowed.

Assumption 3: We assume that all machines are ready to be processed at time zero.

Various mathematical models have been developed by considering the problem in different scenarios. In total, six different models have been developed. The models were created by adding different objective functions and constraints, starting from the

simplest version of the problem. The developed MILP models are given in section 3.1. The details of the models are as follows; the objective function of Model 1-A was determined as makespan minimization, by handling the problem in the simplest way with sequence independent job setup times. Then, the objective function changed to minimizing completion time and given as Model 1-B. In Model 2-A, the setup time is presented as sequence dependent and objective function is minimizing makespan. In Model 2-B, the objective function is determined as minimization completion time using sequence-dependent setup time. In Model 3-A, a job-splitting constraint has been added. Then, Model 3-B was developed by changing the objective function of Model 3-A to minimize earliness tardiness. Then, Model 4 was created by adding machine eligibility constraints and ready time. In Model 5, the factory eligibility constraint has been added. Finally, Model 6 was created by removing job splitting and factory eligibility constraints from the problem. "Model 6" is a mathematical model in which the problem is verified, and the comparison is made by solving it with both Cplex OPL and genetic algorithms.

One of the models has been developed as fuzzy MILP as an extension of the problem which is given in Section 3.2. The solution approach relies on the comparison of solutions through a randomized search algorithm, a Monte Carlo simulation, with improvement routines. We evaluate the performance of the suggested approach for various problem situations using a comprehensive numerical investigation.

3.1 MILP Models

Using the standard scheduling terminology, the problem of the company can be defined as a parallel machine scheduling problem. While developing the mathematical problems, we first approached the problem as a simple makespan minimization. We then improved the model by adding problem-specific constraints such as sequence-dependent set-up times, job splitting, facility eligibility and machine eligibility constraints. When we improve the minimizing makespan model by adding due dates and ready dates for the jobs, the objective function became to minimize total earliness and tardiness. The details of all models we have developed are given below.

Model 1-A (P/C_{max}): For identical parallel machines with job setup times, we offer the classic mixed integer linear programming formulation. Finding the makespan that

can be minimized is the goal. The model satisfies all the assumptions 1, 2 and 3 given above. The additional assumptions given are as follows.

- There is sequence independent setup times.
- Each job can be produced on every machine.

Sets and Indices;

$J = \{1,2, \ldots., |J|\}$: set of job indices

i, j, k: job indices, $i, j, k \in J$

$M = \{1,2, \ldots., |M|\}$: set of machine indices

m: machine index, $m \in M$

Parameters;

Q_i: The number of units in job i.

P_i: The processing time of one unit of job i

S_i: Setup time of job i

Decision Variables;

C_{max}: Makespan, the maximum completion time of all jobs.

$$x_{im} = \begin{cases} 1, & \text{if job } i \text{ is assigned to machine } m \\ 0, & \text{otherwise.} \end{cases}$$

Objective Function:

Minimize C_{max}

s.t.

$$\sum_{m \in M} x_{im} = 1 \qquad \forall i \in J \ (1)$$

$$C_{max} - \sum_{i \in J} (S_i + Q_i P_i) x_{im} \geq 0 \qquad \forall m \in M \ (2)$$

$$x_{im,} \in \{1,0\} \qquad \forall i \in J, \forall m \in M \ (3)$$

The objective is to minimize the makespan. The constraint set (1) allows each job to be assigned to any position on any machine. The constraint set (2) total completion time of jobs on each machine cannot be more than the maximum completion time. The constraint set (3) defines binary decision variables.

Model 1-B ($P/\sum C_i$): We provide the classic mixed integer linear programming formulation for identical parallel machines with job setup times. The difference from Model 1-A is the objective function which is to find minimizing total completion time.

The model satisfies all the assumptions 1, 2 and 3 given above. The additional assumptions given are as follows.

- There is sequence independent setup times.
- Each job can be produced on every machine.

Sets and Indices;

$J = \{1,2, \ldots., |J|\}$: set of all job indices

$J' = \{0,1,2, \ldots., |J|\}$: set of all job indices including the dummy job 0

i, j, k: job indices, $i, j, k \in J$ (job index 0 is for dummy job)

$M = \{1,2, \ldots., |M|\}$: set of machine indices

m: machine index, $m \in M$

Parameters;

P_i: The processing time of one unit of job i

Q_i: The number of units in job i.

S_i: Setup time of job i

L: a very large number

Decision Variables;

C_i: The completion time of job i

$$x_{ijm} = \begin{cases} 1, & \text{if job } i \text{ immediately precedes of job j on machine } m \\ 0, & \text{otherwise.} \end{cases}$$

$$x_{0jm} = \begin{cases} 1, & \text{if job } j \text{ is the first job on machine } m \\ 0, & \text{otherwise.} \end{cases}$$

Objective Function:

Minimize $\sum_{i \in J'} C_i$

s.t.

$$\sum_{m \in M} \sum_{i \in J', i \neq j} x_{\text{ijm}} = 1 \qquad \forall j \in J \ (1)$$

$$\sum_{j \in J} x_{0\text{jm}} = 1 \qquad \forall m \in M \ (2)$$

$$\sum_{j \in J} x_{\text{ijm}} \leq \sum_{k \in J} x_{\text{kim}} \qquad \forall m \in M; \forall i \in J' (3)$$

$$C_j - C_i \geq S_j + Q_j P_j - L(1 - x_{\text{ijm}}) \qquad \forall i \in J', j \in J, i \neq j, \forall m \in M \ (4)$$

$$x_{ijm,} \in \{1,0\} \qquad \forall i \in J', j \in J, \forall m \in M \ (5)$$

$$C_i \geq 0 \qquad \forall i \in J' \ (6)$$

The objective is to minimize total completion time. The constraint set (1) allows each job to be assigned to any position on any machine. The constraint set (2) any job must be assigned to the first position of any machine. The constraint set (3) indicates that the order of jobs assigned to each machine is kept. The constraint set (4) means that if two jobs are processed immediately sequence on the same machine, the difference between their completion times is greater than or equal to the summation of the processing time of the immediately succeeding job plus the setup time of this job. (5) defines binary decision variables and constraint set (6) non-negative decision variable.

Model 2-A ($P/ST_{sd}/C_{max}$): We provide the mixed integer linear programming formulation for identical parallel machines. This model is the version of Model 1-A with added sequence-dependent setup times constraint. The model satisfies all the assumptions 1, 2 and 3 given above. The additional assumption given is as follows.

- Each job can be produced on every machine.
- Setup times depend on job sequences.

Sets and Indices;

$J = \{1,2, \dots, |J|\}$: set of job indices

$J' = \{0,1,2, \dots, |J|\}$: set of job indices including the dummy job 0

i, j, k: job indices, $i \in J'; j, k \in J$.

$M = \{1,2, \dots, |M|\}$: set of machine indices

m: machine index, $m \in M$

Parameters;

Q_i: The number of units in job i.

P_i: The processing time of one unit of job i

S_{ij}: Setup time of job i when immediately precedes job j on the same machine.

Decision Variables;

C_{max}: Makespan, the maximum completion time of all jobs.

$$x_{ijm} = \begin{cases} 1, & \text{if job } i \text{ immediately precedes of job } j \text{ on machine } m \\ 0, & \text{otherwise.} \end{cases}$$

$$x_{0jm} = \begin{cases} 1, & \text{if job } j \text{ is the first job on machine } m \\ 0, & \text{otherwise.} \end{cases}$$

Objective Function:

Minimize C_{max}

s.t.

$$\sum_{m\in M}\sum_{i\in J\prime, i\neq j} x_{\mathrm{ijm}} = 1 \qquad \forall j \in J \ (1)$$

$$\sum_{j\in J} x_{0\mathrm{jm}} = 1 \qquad \forall m \in M \ (2)$$

$$\sum_{j\in J} x_{\mathrm{ijm}} \leq \sum_{k\in J} x_{\mathrm{kim}} \qquad \forall m \in M; \forall i \in J' \ (3)$$

$$C_{max} - \sum_{j\in J, j\neq 0}\sum_{i\in J\prime} (S_{ij} + Q_i P_i) x_{\mathrm{ijm}} \geq 0 \qquad \forall m \in M \ (4)$$

$$x_{ijm,} \in \{1,0\} \qquad \forall i \in J', j \in J, m \in M \ (5)$$

The objective is to minimize makespan. The constraint set (1) allows each job to be assigned to any position on any machine. The constraint set (2) any job must be assigned to the first position of any machine. The constraint set (3) indicates that the order of jobs assigned to each machine is kept. The constraint set (4) detects the makespan of each machine can be less than or equal to the makespan. Finally, the constraint set (5) defines binary decision variables.

Model 2-B ($P/ST_{sd}/\sum C_j$): We provide the mixed integer linear programming formulation for identical parallel machines. This model is the version of Model 1-B with added sequence-dependent setup times constraint. The difference from Model 2-A is the objective function which is to find minimizing total completion times. The model satisfies all the assumptions 1, 2 and 3 given above. The additional assumption given is as follows.

- Each job can be produced on every machine.
- Setup times depend on job sequences.

Sets and Indices;

$J = \{1,2,\ldots.,|J|\}$: set of job indices

$J' = \{0,1,2,\ldots.,|J|\}$: set of job indices including the dummy job 0

i, j, k: job indices, $i \in J'; j, k \in J$

$M = \{1,2,\ldots.,|M|\}$: set of machine indices

m: machine index, $m \in M$

Parameters;

P_i: The processing time of one unit of job i

Q_i: The number of units in job i.

S_{ij}: Setup time of job i when immediately precedes job j on the same machine.

L: a very large number

Decision Variables;

C_i: The completion time of job i

$$x_{ijm} = \begin{cases} 1, & \text{if job } i \text{ immediately precedes of job } j \text{ on machine } m \\ 0, & \text{otherwise.} \end{cases}$$

$$x_{0jm} = \begin{cases} 1, & \text{if job } j \text{ is the first job on machine } m \\ 0, & \text{otherwise.} \end{cases}$$

Objective Function:

Minimize $\sum_{i \in J'} C_i$

s.t.

$$\sum_{m \in M} \sum_{i \in J', i \neq j} x_{\text{ijm}} = 1 \qquad \forall j \in J \ (1)$$

$$\sum_{j \in J} x_{\text{0jm}} = 1 \qquad \forall m \in M \ (2)$$

$$\sum_{j \in J} x_{\text{ijm}} \leq \sum_{k \in J} x_{\text{kim}} \qquad \forall m \in M; \forall i \in J' (3)$$

$$C_j - C_i \geq S_{ij} + Q_j P_j - L(1 - x_{\text{ijm}}) \qquad \forall i \in J', j \in J, i \neq j, \forall m \in M \ (4)$$

$$x_{ijm,} \in \{1,0\} \qquad \forall i \in J', j \in J, \forall m \in M \ (5)$$

$$C_i \geq 0 \qquad \forall i \in J' \ (6)$$

The objective is to minimize total completion time. The constraint set (1) allows each job to be assigned to any position on any machine. The constraint set (2) any job must be assigned to the first position of any machine. The constraint set (3) indicates that the order of jobs assigned to each machine is kept. The constraint set (4) means that if two jobs are processed immediately sequence on the same machine, the difference between their completion times is greater than or equal to the summation of the processing time of the immediately succeeding job plus the setup time of these jobs. (5) defines binary decision variables and constraint set (6) non-negative decision variable.

Model 3-A ($P/ST_{sd}, split/C_{max}$): We provide the mixed integer linear programming formulation for identical parallel machines. This model has sequence-dependent setup times, and job splitting. This model is the version of Model 2-A with added job splitting

constraint. The model satisfies all the assumptions 1, 2 and 3 given above. The additional assumptions given are as follows.

- Each job can be produced on every machine.
- Setup times depend on job sequences.
- Job splitting is available.

Sets and Indices;

$J = \{1,2, \ldots., |J|\}$: set of job indices

$J' = \{0,1,2, \ldots., |J|\}$: set of job indices including the dummy job 0

i, j, k: job indices, $i \in J'; j, k \in J$

$M = \{1,2, \ldots., |M|\}$: set of machine indices

m: machine index, $m \in M$

Parameters;

P_i: The processing time of one unit of job i

S_{ij}: Setup time of job i when immediately precedes job j on the same machine.

Q_i: The number of units in job i.

L: a very large number

Decision Variables;

C_{max}: Makespan, the maximum completion time of all jobs.

$$x_{ijm} = \begin{cases} 1, & \text{if job } i \text{ immediately precedes of job } j \text{ on machine } m \\ 0, & \text{otherwise.} \end{cases}$$

C_{im}: The completion time of job i on machine m

q_{im}: The number of unit jobs of job i processed on machine m

C_i: The completion time of job i

Objective Function:

Minimize C_{max}

s.t.

$$\sum_{m \in M} q_{\text{im}} = Q_i \qquad \forall i \in J' \ (1)$$

$$q_{\text{jm}} \leq L \sum_{i \in J', i \neq j} x_{\text{ijm}} \qquad \forall j \in J, \forall m \in M (2)$$

$$x_{\text{jkm}} \leq \sum_{i \in J', i \neq j} x_{\text{ijm}} \qquad \forall j, k \in J, \forall m \in M, j \neq k \ (3)$$

$$C_{jm} - C_{im} \geq S_{ij} + \left(q_{\text{jm}} P_j\right) - L\left(1 - x_{ijm}\right) \qquad \forall i \in J', j \in J, \forall m \in M, i \neq j \ (4)$$

$$\sum_{j \in J, j \neq i} x_{ijm} \leq 1 \qquad \forall i \in J', \forall m \in M \ (5)$$

$$C_i \geq C_{im} \qquad \forall i \in J', \forall m \in M \ (6)$$

$$C_{max} \geq C_i \qquad \forall i \in J' \ (7)$$

$$x_{ijm} \in \{1,0\} \qquad \forall i \in J', j \in J, \forall m \in M \ (8)$$

$$C_{im,}\, C_{i,}\, q_{im} \geq 0 \qquad \forall i \in J', \forall m \in M \ (9)$$

The objective function is to minimize makespan. The first constraint (1) makes sure that the total number of unit jobs of each job processed on various machines is equal to the job's size. If a job is not processed on a machine, constraint (2) ensures that the number of unit jobs of that job on the machine is zero. Constraint (3) means that jobs are correctly sequenced in a machine by requiring that anytime a job is handled on a machine, its predecessor must be handled as well on the same machine. As a result, job i must have previously been sequenced if job k is to be performed after job j. According to constraint (4), a job's completion time on each machine must be at least more than or equal to the total of its predecessor's completion time, the setup time between the two jobs, and the job's processing time. There can only be one successor for a job on a machine, thanks to constraint (5). Completion time and C_{max} are determined by constraints (6) and (7). Binary decision variables are defined by constraint set (8), and non-negative decision variables are defined by constraint set (9).

Model 3-B ($P/ST_{sd}, split/\ \sum(E_i + T_i)$) : We provide the mixed integer linear programming formulation for identical parallel machines. This model has sequence-dependent setup times, and job splitting. The difference from Model 3-A is the objective function which is to find minimizing total earliness and tardiness. The model satisfies all the assumptions 1, 2 and 3 given above. The additional assumptions given are as follows.

- Each job can be produced on every machine.
- We supposed the earliness penalty and tardiness penalty are "1". Therefore, these are not included in the problem.
- Setup times depend on job sequences.
- Job splitting is available.

Sets and Indices;

$J = \{1,2, \ldots., |J|\}$: set of job indices

$J' = \{0,1,2, \ldots., |J|\}$: set of job indices including the dummy job 0

i, j, k: job indices, $i \in J'; j, k \in J$

$M = \{1,2, \ldots., |M|\}$: set of machine indices

m: machine index, $m \in M$

Parameters;

P_i: The processing time of one unit of job i

S_{ij}: Setup time of job i when immediately precedes job j on the same machine.

Q_i: The number of units in job i.

L: a very large number

d_i: The due date of job i.

Decision Variables;

$x_{ijm} = \begin{cases} 1, & \text{if job } i \text{ immediately precedes of job } j \text{ on machine } m \\ 0, & \text{otherwise} \end{cases}$

$y_{im} = \begin{cases} 1, & \text{if job } i \text{ process on machine } m \\ 0, & \text{otherwise} \end{cases}$

C_{im}: The completion time of job i on machine m

q_{im}: The number of unit jobs of job i processed on machine m

C_i: The completion time of job i

T_i: Tardiness of job i

E_i: Earliness of job i

Objective Function

$Minimize \sum_{\forall i \in J'} (T_i + E_i)$

s.t.

$$\sum_{m \in M} q_{im} = Q_i \qquad \forall i \in J' \ (1)$$

$$q_{jm} \leq L \sum_{i \in J', i \neq j} x_{ijm} \qquad \forall j \in J, \forall m \in M \ (2)$$

$$x_{jkm} \leq \sum_{i \in J', i \neq j} x_{ijm} \qquad \forall j, k \in J, \forall m \in M, j \neq k \ (3)$$

$$C_{jm} - C_{im} \geq S_{ij} + (q_{jm} P_j) - L(1 - x_{ijm}) \qquad \forall i \in J', j \in J, \forall m \in M, i \neq j \ (4)$$

$$C_{jm} \leq L y_{jm} \qquad \forall j \in J, \forall m \in M \ (5)$$

$$C_i \leq C_{im} + L(1 - y_{im}) \qquad \forall i \in J', \forall m \in M \ (6)$$

$$\sum_{m \in M} y_{im} \geq 1 \qquad \forall i \in J' \ (7)$$

$$\sum_{j\in J, j\neq i} x_{\text{ijm}} \leq 1 \qquad \forall i \in J', \forall m \in M \ (8)$$

$$C_i \geq C_{im} \qquad \forall i \in J', \forall m \in M \ (9)$$

$$C_i - d_i = T_i - E_i \qquad \forall i \in J' \ (10)$$

$$x_{ijm} \in \{1,0\} \qquad \forall i \in J', j \in J, \forall m \in M \ (11)$$

$$C_{im}, T_i, E_i, C_{i,} q_{im} \geq 0 \qquad \forall i \in J', j \in J, \forall m \in M \ (12)$$

The objective is to minimize total earliness and tardiness. The first constraint (1) makes sure that the total number of unit jobs of each job processed on various machines is equal to the job's size. If a job is not processed on a machine, constraint (2) ensures that the number of unit jobs of that job on the machine is zero. Constraint (3) means that jobs are correctly sequenced in a machine by requiring that anytime a job is handled on a machine, its predecessor must be handled as well on the same machine. As a result, job i must have previously been sequenced if job k is to be performed after job j. According to constraints (4), (5), (6), and (7), a job's completion time on each machine must be at least greater than or equal to the total of its predecessor's completion time, setup time between them, and processing time. There can only be one successor for a job on a machine, thanks to constraint (8). Constraints (9) and (10) aid in determining each job's tardiness and earliness. Binary decision variables are defined by constraint set (11) while non-negative decision variables are defined by constraint set (12).

Model 4 ($P/ST_{sd}, split, M_i/\sum(E_i + T_i)$): We provide the mixed integer linear programming formulation for identical parallel machines. This model has sequence-dependent setup times, job splitting, job ready dates and machine eligibility. The difference from Model 3-B is adding machine eligibility and job ready date parameters. The objective function is to find minimizing total earliness and tardiness, the same as the Model 3-B. The model satisfies all the assumptions 1, 2 and 3 given above. The additional assumptions given are as follows.

- We supposed the earliness penalty and tardiness penalty are "1". Therefore, these are not included in the problem.
- Each job has a ready date.
- Setup times depend on job sequences.
- Job splitting is available.

- Machine eligibility is available.

Sets and Indices;

$J = \{1,2, \ldots., |J|\}$: set of job indices

$J' = \{0,1,2, \ldots., |J|\}$: set of job indices including the dummy job 0

i, j, k: job indices, $i \in J'; j, k \in J$

$M = \{1,2, \ldots., |M|\}$: set of machine indices

m: machine index, $m \in M$

Parameters;

P_i: The processing time of one unit of job i

S_{ij}: Setup time of job i when immediately precedes job j on the same machine.

Q_i: The number of units in job i.

L: a very large number

d_i: The due date of job i.

r_i: The ready date of job i.

M_i: Eligible machine set of job i.

Decision Variables;

$x_{ijm} = \begin{cases} 1, & \text{if job } i \text{ immediately precedes of job } j \text{ on machine } m \\ 0, & \text{otherwise} \end{cases}$

$y_{im} = \begin{cases} 1, & \text{if job } i \text{ process on machine } m \\ 0, & \text{otherwise} \end{cases}$

C_{im}: The completion time of job i on machine m

q_{im}: The number of unit jobs of job i processed on machine m

C_i: The completion time of job i

T_i: Tardiness of job i

E_i: Earliness of job i

Objective Function

$Minimize \sum_{\forall i \in J} (T_i + E_i)$

s.t.

$$\sum_{m \in M_i} q_{im} = Q_i \qquad \forall i, m \in M_i \ (1)$$

$$q_{jm} \leq L \sum_{\forall i \in J'; i \neq j} x_{ijm} \qquad \forall j, m \in M_j \ (2)$$

$$\sum_{i \in J, i=0; i \neq j} x_{ijm} \leq q_{jm} \qquad \forall j, m \in M_j \ (3)$$

$$x_{jkm} \leq \sum_{i \in J', i \neq j} x_{ijm} \qquad \forall j, k, m \in \{M_i \cap M_k\}, j \neq k \ (4)$$

$$C_{jm} - C_{im} \geq S_{ij} + q_{\mathrm{jm}}P_j - L(1 - x_{ijm}) \qquad \forall i,j,m \in \{M_i \cap M_j\}, j \neq i \quad (5)$$

$$C_{jm} - S_{ij}x_{ijm} - q_{\mathrm{jm}}P_j \geq r_j - L(1 - x_{ijm}) \qquad \forall i,j,m \in \{M_i \cap M_j\} \quad j \neq i \quad (6)$$

$$C_{jm} \leq Ly_{\mathrm{jm}} \qquad \forall j,m \in M_j \quad (7)$$

$$C_i \geq C_{im} \qquad \forall i,m \in M_i \quad (8)$$

$$C_i \leq C_{im} + L(1 - y_{im}) \qquad \forall i,m \in M_i \quad (9)$$

$$\sum_{m \epsilon M_i} y_{\mathrm{im}} \geq 1 \qquad \forall i \in J' \quad (10)$$

$$\sum_{j \in J, j \neq i} x_{ijm} \leq 1 \qquad \forall i,m \in M_i \quad (11)$$

$$T_i \geq C_{im} - d_i \qquad \forall i\,,m \in M_i \quad (12)$$

$$E_i \geq d_i - C_{im} \qquad \forall i\,,m \in M_i \quad (13)$$

$$x_{ijm} \in \{1,0\} \qquad \forall i,j,m \in \{M_i \cap M_j\} \quad (14)$$

$$y_{im} \in \{1,0\} \qquad \forall i,m \in M_i \quad (15)$$

$$C_{im}, T_i, E_i, C_{i,}\, q_{im} \geq 0 \qquad \forall i\,,m \in M_i \quad (16)$$

The objective is to minimize total earliness and tardiness. The first constraint (1) makes sure that the total number of unit jobs of each job processed on various machines is equal to the job's size. If a job is not processed on a machine, constraints (2), and (3) ensure that the number of unit jobs of that job on the machine is zero. Constraint (4) means that jobs are correctly sequenced in a machine by requiring that anytime a job is handled on a machine, its predecessor must be handled as well on the same machine. As a result, job i must have previously been sequenced if job k is to be performed after job j. According to constraint (5), a job's completion time on each machine must be at least greater than or equal to the total of its predecessor's completion time, setup time between them, and processing time. Constraints (6), and (7) ensure that the completion time of a new job after a job on a machine is equal to the sum of that job's ready time, job completion time, and setup time between two jobs. Constraint (8), (9) and (10) work together. Constraints (8), and (9) ensure that if a job is split and processed on different machines, the completion time of that job is equal to the maximum completion time of the completion times between the machines into which it is split. Constraint (10) is an intermediate decision variable. There can only be one successor for a job on a machine, thanks to constraint (11). Constraints (12), and (13) find tardiness and earliness of each job. Constraint sets (14), and (15) define binary decision variables and constraint set (16) non-negative decision variable.

Model 5 ($P/ST_{sd}, split, M_f/\sum(E_i + T_i)$): We provide the mixed integer linear programming formulation of the consider scheduling problem for identical parallel machines. In this approach, we supposed there is more than one facility with machine eligibility where M_f is the index set of machines that are capable of processing job i on a facility bases. This model has sequence-dependent setup times, job splitting, job ready dates and machine eligibility. The difference from Model 4 is adding set of facility indices and machine eligibility is differ due to the facility index. The objective function is to find minimizing total earliness and tardiness, the same as the Model 4. The model satisfies all the assumptions 1, 2 and 3 given above. The additional assumptions given are as follows.

- We supposed the earliness penalty and tardiness penalty are "1". Therefore, these are not included in the problem.
- Each job has a ready date.
- Setup times depend on job sequences.
- Job splitting is available.
- Each job must be produced in only one facility.
- Each job must be processed on the eligible machine of the facility to which it is assigned.

Sets and Indices:

$J = \{1,2, \dots., |J|\}$: set of job indices

$J' = \{0,1,2, \dots., |J|\}$: set of job indices including the dummy job 0

i, j, k: job indices, $i \in J'; j, k \in J$

$M = \{1,2, \dots., |M|\}$: set of machine indices

m: machine index, $m \in M$

$F = \{1,2, \dots., |F|\}$: set of facility indices

f: facility index, $f \in F$

M_f: set of machine indices at facility f, $M_f \subset M$

M_{if}: set of machine indices that are eligible to process job $i \in J'$ at facility f, $M_{if} \subset M_f$

Parameters;

P_i: The processing time of one unit of job i

S_{ij}: Setup time of job i when immediately precedes job j on the same machine.

Q_i: The number of units in job i.

L: a very large number

d_i: The due date of job i

r_i: The ready date of job i

Decision Variables;

$$x_{ijm} = \begin{cases} 1, & \text{if job } i \text{ immediately precedes of job } j \text{ on machine } m \\ 0, & \text{otherwise} \end{cases}$$

$$Z_{if} = \begin{cases} \geq 1, & \text{if job } i \text{ processed on facility } f \\ 0, & \text{if not processed on a facility } f, \text{otherwise} \end{cases}$$

y_{im}: 1st intermediate binary decision variable

α_{if}: 2nd intermediate binary decision variable (which job i is process on which facility f)

C_{im}: The completion time of job i on machine m

q_{im}: The number of unit jobs of job i processed on machine m

C_i: The completion time of job i

T_i: Tardiness of job i

E_i: Earliness of job i

Objective Function

Minimize $\sum_{j\in J} (T_{\mathrm{j}} + E_{\mathrm{j}})$

s.t.

$$\sum_{m\in \bigcup_f M_{if}} q_{im} = Q_i \qquad \forall i \in J' \ (1)$$

$$q_{\mathrm{jm}} \leq L \sum_{i\in J', i\neq j} x_{\mathrm{ijm}} \qquad \forall j \in J, m \in \bigcup_f M_{jf} \ (2)$$

$$\sum_{i\in J', i\neq j} x_{\mathrm{ijm}} \leq q_{\mathrm{jm}} \qquad \forall j \in J, m \in \bigcup_f M_{jf} \ (3)$$

$$x_{\mathrm{jkm}} \leq \sum_{i\in J', i\neq j} x_{\mathrm{ijm}} \qquad \forall \mathrm{j,k} \in J, j \neq k, \ m \in \left(\bigcup_f M_{jf} \cap \bigcup_f M_{kf}\right) \ (4)$$

$$Z_{\mathrm{jf}} = \sum_{i\in J', i\neq j} \sum_{m\in M_{if}\cap M_{jf}} x_{\mathrm{ijm}} \qquad \forall j \in J, f \in F \ (5)$$

$$\sum_{f\epsilon F} Z_{\mathrm{jf}} \geq 1 \qquad \forall j \in J \ (6)$$

$$Z_{\mathrm{jf}} \leq L\alpha_{jf} \qquad \forall j, f \in F \ (7)$$

$$\sum_{f\epsilon F} \alpha_{jf} = 1 \qquad \forall j \in J \ (8)$$

$$C_{jm} - S_{ij} - q_{\mathrm{jm}} p_j \geq C_{im} - L(1 - x_{ijm}) \quad \forall i \in J', j \in J, j \neq i \ ; \ m \in (\bigcup_f M_{if} \cap \bigcup_f M_{jf}) \ (9)$$

$C_{jm} - S_{ij} - q_{\mathrm{jm}} p_j \geq r_j - L(1 - x_{ijm}) \quad \forall i \in J', j \in J, j \neq i;\ m \in \left(\bigcup_f M_{if} \cap \bigcup_f M_{jf}\right)$ (10)

$C_{jm} \leq L y_{\mathrm{jm}} \quad \forall j \in J, m \in \bigcup_f M_{jf}$ (11)

$C_i \leq C_{\mathrm{im}} \quad \forall i \in J', m \in \bigcup_f M_{if}$ (12)

$C_i \leq C_{im} + L(1 - y_{im}) \quad \forall i \in J', m \in \bigcup_f M_{if}$ (13)

$\sum_{m \epsilon \bigcup_f M_{if}} y_{\mathrm{im}} \geq 1 \quad \forall i \in J'$ (14)

$\sum_{j \in J, j \neq i} x_{\mathrm{ijm}} \leq 1 \quad \forall i \in J', m \in \bigcup_f M_{if}$ (15)

$T_i \geq C_{im} - d_i \quad \forall i \in J', m \in \bigcup_f M_{if}$ (16)

$E_i \geq d_i - C_{im} \quad \forall i \in J', m \in \bigcup_f M_{if}$ (17)

$x_{ijm} \in \{1,0\} \quad \forall i \in J', j \in J, \mathrm{m} \in \left(\bigcup_f M_{if} \bigcup_f M_{jf}\right)$ (18)

$y_{im} \in \{1,0\} \quad \forall i \in J', \mathrm{m} \in \bigcup_f M_{jf}$ (19)

$\alpha_f \in \{1,0\} \quad \forall f \in F$ (20)

$C_{im}, T_i, E_i, C_{i,}\, q_{im}, Z_{if} \geq 0 \quad \forall i \in J', j \in J, \mathrm{m} \in \left(\bigcup_f M_{if} \cap \bigcup_f M_{jf}\right)$ (21)

The objective is to minimize total earliness and tardiness. The first constraint (1) makes sure that the total number of unit jobs of each job processed on various machines is equal to the job's size. If a job is not processed on a machine, constraints (2), and (3) ensure that the number of unit jobs of that job on the machine is zero. Constraint (4) means that jobs are correctly sequenced in a machine by requiring that anytime a job is handled on a machine, its predecessor must be handled as well on the same machine. As a result, job i must have previously been sequenced if job k is to be performed after job j. Constraints (5), (6), (7), and (8) work together. Each facility has different machine sets. With these constraints, it is ensured that each job is assigned a facility. At the same time, it is processed in the machine sets of that facility. Constraints (7), and (8) allow each job to be assigned to a facility using an intermediate variable. Z_{if} is a positive integer value if jobs assign to machines at a facility and this value gives a total assigning machine number. If there is not an assigning job to any facility that value gives zero. can be positive integer value if jobs are assigned to machines at a facility. According to constraint (9), a job's completion time on each machine must be at least greater than or equal to the total of its predecessor's completion time, setup time between them, and processing time. Constraints (10), and (11) ensure that the

completion time of a new job after a job on a machine is equal to the sum of that job's ready time, job completion time, and setup time between two jobs. Constraint (11) forces a job completion time to be zero when not split to a machine. Constraints (12), (13) and (14) work together. Constraints (12), and (13) ensure that if a job is split and processed on different machines, the completion time of that job is equal to the maximum completion time of the completion times between the machines into which it is split. Constraint (14) is an intermediate decision variable. Whichever machine the last job is completed on, makes that machine's value 1. There can only be one successor for a job on a machine, thanks to constraint (15). Constraints (16), and (17) find tardiness and earliness of each job. Constraints (18), (19), and (20) define the binary decision variables. Constraints (21) is to define non-negative decision variables.

Model 6 ($P/M_i, ST_{sd}/ \sum(E_i + T_i)$) : We provide the mixed integer linear programming formulation of the consider scheduling problem for identical parallel machines with sequence dependent set-up times, job ready dates, and machine eligibility to find minimize total earliness tardiness and machine. We created this model by simplifying it compared to Model 5. Differences from Model 5 are that job splitting, and facility eligibility are unavailable in this model. The model satisfies all the assumptions 1, 2 and 3 given above. The additional assumptions given are as follows.

- We supposed the earliness penalty and tardiness penalty are "1". Therefore, these are not included in the problem.
- Each job has a ready date.
- Setup times depend on job sequences.
- Machine eligibility is available.

Sets and Indices:

$J = \{1,2, \dots., |J|\}$: set of job indices

$J' = \{0,1,2, \dots., |J|\}$: set of job indices including the dummy job 0

i, j, k: job indices, $i \in J'; j, k \in J$

$M = \{1,2, \dots., |M|\}$: set of machine indices

m: machine index, $m \in M$

M_i: set of machine indices that are eligible to process job $i \in J'$

Parameters;

P_i: The processing time of one unit of job i

S_{ij}: Setup time of job i when immediately precedes job j on the same machine.

Q_i: The number of units in job i.

L: a very large number

d_i: The due date of job i

r_i: The ready date of job i

Decision Variables;

$x_{ijm} = \begin{cases} 1, & \text{if job } i \text{ immediately precedes of job } j \text{ on machine } m \\ 0, & \text{otherwise} \end{cases}$

C_{im}: The completion time of job i on machine m

T_i: Tardiness of job i

E_i: Earliness of job i

Objective Function

Minimize $\sum_{j\in J,}(T_{\mathrm{j}} + E_{\mathrm{j}})$

s.t.

$$\sum_{i\in J',i\neq j} \sum_{m\in M_i\cap M_j} x_{\mathrm{ijm}} = 1 \qquad \forall \mathrm{j} \in J \ (1)$$

$$x_{\mathrm{jkm}} \leq \sum_{i\in J',i\neq j} x_{\mathrm{ijm}} \qquad \forall \mathrm{j,k} \in J, j \neq k, m \in (M_j \cap M_k) \ (2)$$

$$C_{jm} - S_{ij} - Q_{\mathrm{j}}P_j \geq C_{im} - L(1 - x_{ijm}) \qquad \forall i \in J', j \in J, j \neq i \ ; \ m \in (M_i \cap M_j) \ (3)$$

$$C_{jm} - S_{ij} - Q_{\mathrm{j}}P_j \geq r_j - L(1 - x_{ijm}) \qquad \forall i \in J', j \in J, j \neq i; \ m \in (M_i \cap M_j) \ (4)$$

$$C_{jm} \leq LQ_j \qquad \forall j \in J, m \in M_j \ (5)$$

$$\sum_{j\in J,j\neq i} x_{\mathrm{ijm}} \leq 1 \qquad \forall i \in J', m \in M_i \ (6)$$

$$T_i \geq C_{im} - d_i \qquad \forall i \in J', m \in M_i \ (7)$$

$$E_i \geq d_i - C_{im} \qquad \forall i \in J', m \in M_i \ (8)$$

$$x_{ijm} \in \{1,0\} \qquad \forall i \in J', j \in J, \mathrm{m} \in (M_i \cap M_j) \ (9)$$

$$C_{im}, T_i, E_i \geq 0 \qquad \forall i \in J', j \in J, \mathrm{m} \in (M_i \cap M_j) \ (10)$$

The objective is to minimize total earliness and tardiness. The first constraint (1) makes sure that the total number of unit jobs of each job processed on various machines is equal to the job's size. Constraint (2) means that jobs are correctly sequenced in a machine by requiring that anytime a job is handled on a machine, its predecessor must be handled as well on the same machine. As a result, job i must have previously been sequenced if job k is to be performed after job j. According to constraint (3), a job's

completion time on each machine must be at least greater than or equal to the total of its predecessor's completion time, setup time between them, and processing time. Constraints (4), and (5) ensure that the completion time of a new job after a job on a machine is equal to the sum of that job's ready time, job completion time, and setup time between two jobs. There can only be one successor for a job on a machine, thanks to constraint (6). Constraints (7), and (8) find tardiness and earliness of each job. Constraint (9) defines the binary decision variables. Constraint (10) is to define non-negative decision variables.

3.2 Fuzzy MILP Model

Most research on scheduling problems assumes deterministic processing and setup times (Pinedo, 1999). However, in the company, processing times are not deterministic due to various factors such as operator learning curve and fatigue, environmental and material related factors, and machine maintenance requirements (Biskup, 1999). Likewise, setup times also vary depending on human and technical factors. A preliminary study indicates that these times can be interpreted as fuzzy numbers. Hence, we also developed a fuzzy model for these uncertainties by assuming fuzzy processing times and setup times are fuzzy to minimize makespan which is given in section 3.2.2.

3.2.1. Formulation Fuzzy Processing and Sequence Dependent Set-up Times

In this study, we represent processing times and sequence-dependent setup times as triangular fuzzy numbers (TFN). A triangular fuzzy number $\tilde{A}$ can be defined by a triplet $(a_1, a_2, a_3) \in \mathbb{R}^3$ such that $a_1 < a_2 < a_3$. The membership function of A is defined as follows (Ahmadizar, Fardin & Leila, 2011; Mokotoff, 2004; Tahar, Yalaoui, Chu & Amodeo, 2006; Vallada & Ruiz, 2011; Fanjul-Peyro, Ruiz & Perea, 2019; Unlu & Mason, 2010; Low & Wu, 2016; Edis & Ozkarahan, 2011; Rocha, Ravetti, Mateus & Pardalos, 2008).

$$\mu_{\tilde{A}}(x) = \begin{cases} \dfrac{x - a_1}{a_2 - a_1}, & a_1 \leq x \leq a_2 \\ \dfrac{a_3 - x}{a_3 - a_2}, & a_2 < x \leq a_3 \\ 0, & otherwise \end{cases}$$

$\tilde{P}_i$ is the fuzzy processing time of job i denoted by triplet (P_{1i} (optimistic value), P_{2i} (most plausible value), P_{3i} (pessimistic value)). The fuzzy sequence-dependent setup time between job i and i' is $\tilde{S}_{ii'}$ that is denoted by ($S_{1ii'}$ (optimistic value), $S_{2ii'}$ (most plausible value), $S_{3ii'}$ (pessimistic value)). For two fuzzy numbers $\tilde{A} = (a_1, a_2, a_3)$ and $\tilde{B} = (b_1, b_2, b_3)$ addition operation is defined as $\tilde{A} + \tilde{B} = (a_1 + b_1, a_2 + b_2, a_3 + b_3)$ (Ahmadizar, Fardin & Leila, 2011). Fuzzy addition operation is required for our study since completion time of any job i ($\tilde{c}_i$) is calculated as summation of fuzzy processing and setup times of all jobs that are processed before job i on the same machine. Our objective function makespan is denoted by $\tilde{C}_{max}$, which is the maximum of all $\tilde{c}_i$ values. For the defuzzification we employ the centroid method and hence defuzzified times are represented by $\frac{a_1+a_2+a_3}{3}$.

3.2.2. Fuzzy MILP Formulation

In this section, we provide the fuzzy mixed integer linear programming formulation of the considered $P/M_i, ST_{sd}/C_{max}$ scheduling problem with identical parallel machines, fuzzy sequence-dependent set-up times, fuzzy processing times and machine eligibility restrictions where M_i is the index set of machines that are capable of processing job i. It is assumed that all jobs are ready, and all machines are available at time zero. In short, the problem is a fuzzy $P/M_i, ST_{sd}/C_{max}$ scheduling problem. The assumptions and notation used in the fuzzy MILP model is given below:

- Setup times depend on job sequences.
- Machine eligibility is available.

Sets and Indices;

$J = \{1,2, \dots, |J|\}$: set of job indices

i, j, k: job indices, $i, j, k \in J$ (job index 0 is for dummy job)

$M = \{1,2, \dots, |M|\}$: set of machine indices

m: machine index, $m \in M$

M_i: index set of machines that can process job i, $M_i \subset M$

J_m: index set of jobs that can be processed by machine m, $J_m \subset J$

Fuzzy Parameters;

$\tilde{P}_i$: fuzzy processing time of one unit of job i

$\tilde{S}_{ij}$: fuzzy setup time when job j is the immediate successor of job i on the same machine.

Q_i: The number of units in job i.

Decision Variables;

$$x_{ijm} = \begin{cases} 1, & \text{if job } i \text{ immediately precedes of job } j \text{ on machine } m \\ 0, & \text{otherwise.} \end{cases}$$

$$x_{0jm} = \begin{cases} 1, & \text{if job } j \text{ is the first job on machine } m \\ 0, & \text{otherwise.} \end{cases}$$

$\tilde{C}_{max}$ = fuzzy makespan, the maximum completion time of all jobs

The fuzzy MILP formulation is as follows*:*

Objective Function

minimize $\tilde{C}_{max}$

s.t.

$$\sum_{i \in J, i \neq \mathrm{j}} \sum_{m \in M_i \cap M_\mathrm{j}} x_{ijm} = 1 \qquad \forall j \in J \ (1)$$

$$\sum_{j \in J_m} x_{0jm} = 1 \qquad \forall m \in M \ (2)$$

$$\sum_{\mathrm{j} \in J_m, j \neq i} x_{ijm} \leq \sum_{\mathrm{k} \in J_m \cup \{0\}, k \neq i} x_{kim} \qquad \forall i \in J, \forall m \in M_i \ (3)$$

$$\tilde{C}_{max} \geq \sum_{j \in J_m} \sum_{i \in J_m \cup \{0\}, i \neq j} (\tilde{S}_{\mathrm{ij}} + \tilde{P}_i Q_i) x_{\mathrm{ijm}} \qquad \forall m \in M_i \ (4)$$

$$x_{ijm} \in \{0,1\} \qquad \forall m \in M_i, i \in J_m \cup \{0\}, \mathrm{j} \in J_m, i \neq \mathrm{j} \ (5)$$

The objective is to minimize the makespan. The constraint set (1) allows each job to be assigned to any position on any machine. Constraint (2) is to determine the first job of each machine. Constraint (3) indicates that for any job that is processed on any machine, there might not be a successor (if the job is the last one, then there is no successor) but there should always be a predecessor (if the job is the first one, then dummy job 0 is the predecessor). Constraint set (4) specifies that makespan is greater than equal to the completion time of all jobs assigned to any machine which is the sum of fuzzy setup and processing times of all assigned jobs. Finally, constraint (5) is to say that decision variables are binary.

CHAPTER 4
SOLUTION ALGORITHMS

MILP models as given in Chapter 3 were verified by solving them in Cplex OPL using toy data. However, optimal solution methods are generally effective for small-sized problems, depending on the complexity of the problem. Factors such as insufficient processor speeds and therefore long solution times and memory limitations make it impossible to use optimal solution methods for real-scale large problems. Heuristic methods that require less memory and provide consistent and near-optimal solutions in a shorter time are preferred. As solution algorithm approaches, we used the genetic algorithm and the randomized search algorithm. The models that we use heuristic approaches are determined as model-6 given in section 3.1 and the fuzzy MILP model provided in section 3.2. Model 6 is the mathematical model in which the problem is verified, and the comparison is made by solving it with genetic algorithms. The fuzzy MILP was compared with the randomized search algorithm. The detail of the genetic algorithm is given in section 4.1 while the randomized search algorithm is given in section 4.2.

4.1 Genetic Algorithm (GA)

Since the problem is of the NP-hard class, there is a need for an algorithm that gives a faster and more suitable solution. The basic logic of the genetic algorithm, which dates back to the 1970s, is survival of the fittest and evolution depending on the principles of natural selection. Three key elements form the foundation of the algorithm. The first is the chromosome structure, in which solution alternatives are defined. The second is the genes that contain each chromosome, and they are the smallest part of the solution. All these compositions form the third component, the population, which is the sample area of the model. In summary, the combination of genes constitutes the chromosomes, and the combination of the chromosomes constitutes the population. When we apply the specific constraints of the problem to the genetic algorithm, the general structure is formed as follows:

Step 1: The first step is to assign values to the population's chromosomes.

Step 2: Assess the chromosomes to determine each chromosome's fitness value.

Step 3: Establishing a new population and applying a selection technique.

Step 4: Apply the crossover procedure to the existing population in step four.

Step 5: Apply the mutation procedure to the existing population.

Step 6: Return to the best schedule in step six.

Step 1. Chromosome Representation:

Establishing the starting population is an important process, as GA performance is dependent on chromosome diversity. In our approach, the initial population is formed by randomly selecting 1, 50, 100, 200, and 300 individuals of different population sizes with different permutations.

In the proposed structure, genes contain assignment and permutation information. Joo and Kim (2012) described that the chromosome structure is designed as a multi-component system, and this notation expresses the order of jobs, machine selection, and how many machines the jobs will be processed on. The "*" character is used to divide machine numbers. We selected this representation for the problem. As shown in Figure 4.1., according to the example of 5 jobs and 3 machines, the 3rd job is produced on machine 1. On machine 2, jobs 1, 5, and 2 are produced in sequence. In machine 3, the 4th job is produced.

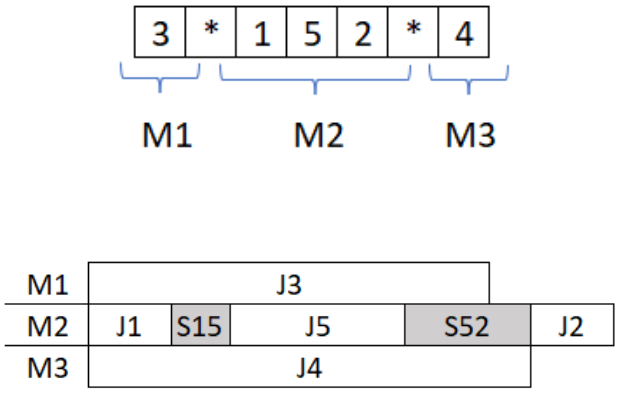

Figure 4.1. Chromosome Representation

Since there is a machine eligibility constraint in the problem, an additional control mechanism was developed. After the jobs are randomly assigned to the machines,

each job is checked to see if it is on its own eligible machine. If it is not located on an eligible machine, it is assigned at random to any machine and any location within the group of machines.

Step 2. Fitness Function:

GA is a maximization algorithm. However, the objective function of the problem is to minimize the sum of the total tardy and early finish times. In order to use the GA in the minimization structure, the transfer function and the fitness function were created by using the total tardiness and earliness value. The fitness function is calculated by 1/ "objective function value". To calculate the objective function, the completion times of each job were calculated as specified in the model. Then, the completion times of each job are subtracted from the due date and find how much it deviates from the due date. The deviation value of all jobs is summed up and the value of objective function is found. The smaller this value, the better the result.

Step 3. Selection:

The mechanism by which individuals are determined to create a new generation is defined as the reproduction operator. The roulette wheel mechanism is employed in this study. In the roulette wheel mechanism, individuals have a chance to determine their fitness value. It is "sorted" according to the fitness value in the reproduction operator used. Thus, the basis for matching the appropriate values with each other is prepared.

Step 4. Crossover:

The one-point crossover was used to increase variations. A job is randomly selected from the list of jobs. Suppose the number of this job coincides with the jth index in the first parent and the jth index in the second parent. The part up to the first place of the first parent is transferred directly to the first child. The transferred parts are deleted from the second parent. The remaining machinery and jobs in the second parent are transferred to the remaining part of the first child. This process takes place in the same way as the second parent for the formation of the second child. (As shown in Figure 4.2.). For the crossover probability, we follow Min and Cheng's (1999) work, which selects 0.92. However, since there is a machine eligibility constraint in the problem after the crossover occurs, the jobs are checked to see if they are in the eligible machine set. In the absence of this, it is given a random location on one of the machines in the list of machines to which it may be assigned.

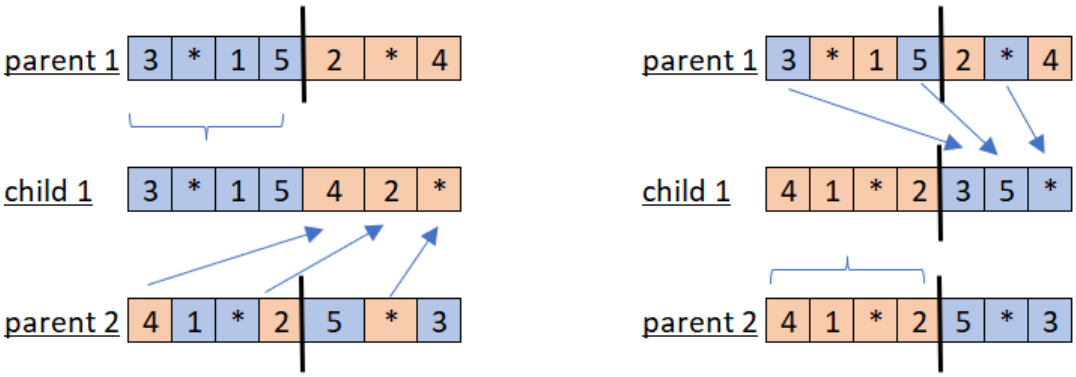

Figure 4.2. An Example of One-point Crossover Representation

Step 5. Mutation:

Two different mutations were proposed. If there is a chromosome that will mutate according to the mutation rate, one of the two mutations with a 50% chance can be performed. For the mutation probability, we follow Min and Cheng's (1999) work, which selected 0.05. The mutations are given in Figure 4.3.

Mutation 1: If there is more than one job in a machine with the first mutation, one of these machines is chosen randomly. Then two of the jobs on the selected machine are randomly selected, and their sequence changes.

Mutation 2: If a job can be assigned to more than one machine, one of those jobs is chosen randomly. A random machine is selected from among the machines it can assign to, except for its current machine, and this job is assigned to a random order on its new machine. For example, in Figure 4.3. (Mutation 2), we assume that job 1 can be assigned to all machines. Currently, it is assigned to machine 2. After mutation, it can be assigned machine 1 or 3. After randomly choosing we assume machine 3 was selected. This job should be assigned to machine 3 and to the position selected randomly.

Termination Criteria: The algorithm terminates if the best solution converges in given different scenarios (details are given in section 5.2).

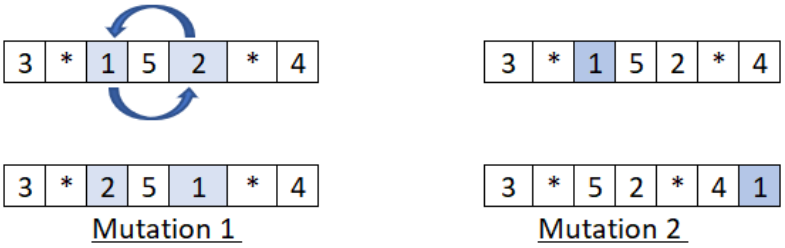

Figure 4.3. An Example of Mutation Representation

4.2 Randomized Search Algorithm

Since the considered $P/M_i/C_{max}$ scheduling problem is in the NP-hard class, heuristic approaches that provide near-optimal solutions in a reasonable amount of time are needed. In this study, we propose a randomized search algorithm enriched with an improvement routine. At each replication, the algorithm first randomly assigns jobs to the eligible machines and for each machine randomly sequences the assigned jobs. Afterwards, it identifies the machine with the longest completion time. The last job of that machine is transferred to a randomly selected position of the machine with the shortest completion time among the machines that can process the job to be transferred. The current replication's best solution is saved as the subroutine's best solution, and this transfer process is repeated until the makespan does not improve. This procedure is repeated until the replication counter is reached to a predetermined number, and among all replications, the solution that has the minimum makespan is reported as the best solution.

Example: Suppose that there are five jobs and three machines in the production system. As can be seen in Table 4.1. and 4.2., jobs cannot be processed on every machine, and the process times change according to the jobs. There are also different setup times in the transition from job to job. If we assume that these jobs are randomly assigned to the machines that they can assign to, the first order is as in Figure 4.4. In iteration 1, the completion time of the last job is 61 and is on machine 1. The last job found on this machine is the 5th job. If we assign this job to any other machine it can assign to and calculate the completion time of last job again (2nd iteration), and the new completion time becomes 49. This sequence is kept because this solution is better than the first solution (given in Table 4.3.). Then the job that finished the latest in the new schedule is on machine 1 and job 2. The 2nd job can be randomly assigned to the 1st and 3rd machines. However, if it is assigned to the 3rd machine, the iteration stops in the current sequence because the completion time will be longer than the current one.

.

Table 4.1. Eligible Machines and Processing Times for the Example Problem

Jobs	Eligible Machine Set	Processing Time
1	1,2	15
2	3,1	30
3	3	20
4	2,3	25
5	1,2,3	10

Table 4.2. Setup Time Matrix for the Example Problem

jxj	**1**	**2**	**3**	**4**	**5**
1	0	4	4	1	6
2	2	0	1	4	2
3	4	3	0	2	1
4	3	2	2	0	3
5	1	2	5	1	0

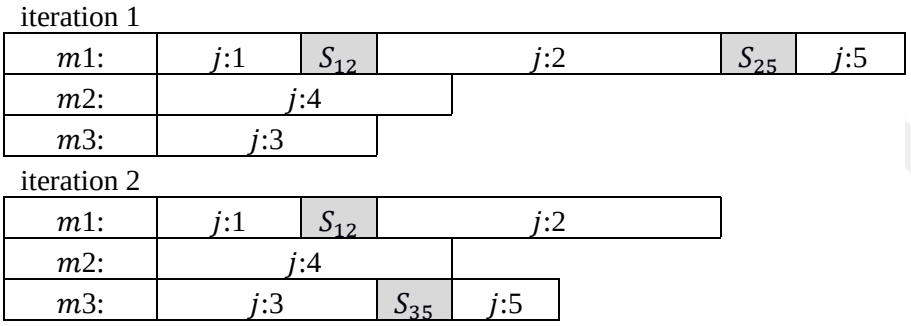

Figure 4.4. Schedule in Identical Parallel Machines for the Example Problem

Table 4.3. Completion Times of the Last Jobs for Each Machines

Machine #	Completion Times of The Last Jobs (Iteration 1)	Completion Times of The Last Jobs (Iteration 2)
***m*1**	61	49
***m*2**	25	25
***m*3**	20	31

CHAPTER 5
COMPUTATIONAL STUDY

The section is related to analyzing the experiments carried out with two different approaches. The one of them is for the developed Model 6 with the genetic algorithm. The other one is for the fuzzy MILP model with the developed randomized search algorithm. We compared heuristic solutions with the MILP model solutions by solving IBM ILOG CPLEX 20.1.0.0 on a computer Intel Core i5-3317U CPU 1.70 GHz.

5.1 The Genetic Algorithm

By changing the makespan scheduling model to Model 6, some extra information such as job ready dates, sequence-dependent setup durations, machine eligibility, and job due dates were included in the model. The function's objective was set to minimize earliness and tardiness. First, to confirm the developed mathematical model's accuracy, the problem was solved in OPL Cplex (the code is given in Appendix 1), and the accuracy of the model was determined. Then the problem was coded for the genetic algorithm in VBA-Macro. The performance of the genetic algorithm was measured by comparing different data (data are given in Appendix 2) with Cplex solutions. The number of jobs in the data examined in this problem is 10 and 20. The number of machines is 3 and 8. Genetic algorithm solutions were handled with five different GA parameter sets. The details are given below.

1. In the first parameter set, the population size was determined as 100 and the number of iterations was given as 200 as the stopping criterion.
2. In the second parameter set, the population size was increased to 200 and iteration number was not changed. However, a solution could not be obtained in the combination of a population number of 200 and 20 jobs and 8 machines.
3. Afterward in the third parameter set, the population size was increased to 300 and the number of iterations was given as 20. Here, the resulting population

was given as input to the next solution and the iteration number was given as 20. This process continued until the fitness function result was the same in the whole population. In this approach, the result of 10 jobs x 3 machine combination was repeated 3 times. It was repeated 5 times in 20 jobs x 3 machine combination. It was repeated 3 times on 10 jobs x 8 machines. It was repeated 4 times in 20 jobs x 8 machines. The results of this approach were plotted (see Figure 5.1., Figure 5.2., Figure 5.3., and Figure 5.4.)

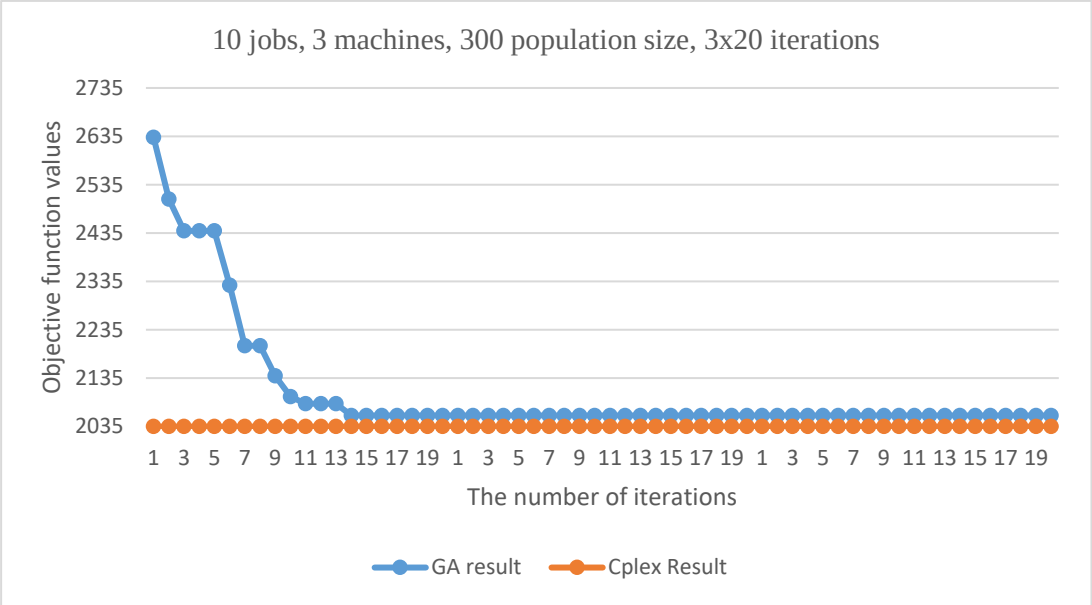

Figure 5.1. The Plot of Comparison of GA and Optimal Solution for 10$\boldsymbol{j}$ and 3$\boldsymbol{m}$

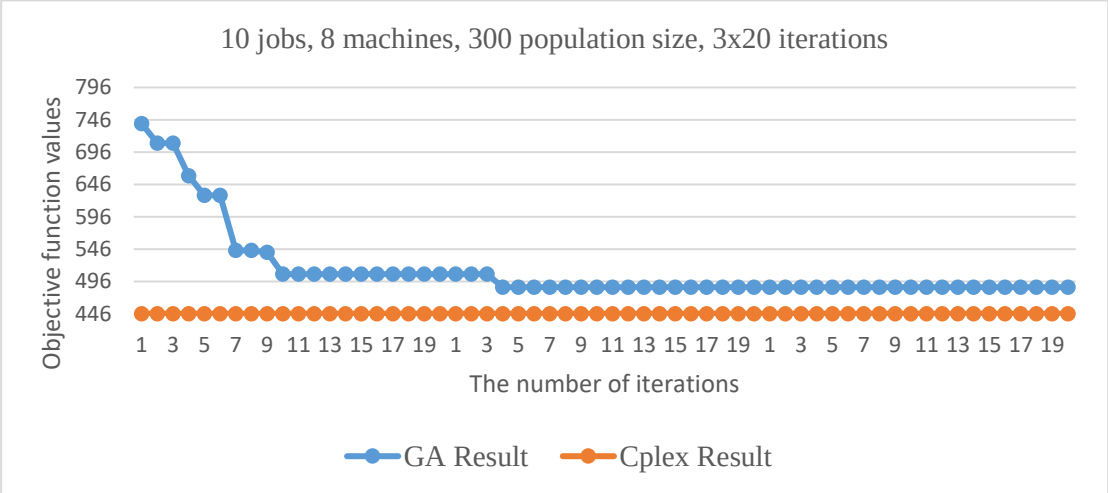

Figure 5.2. The Plot of Comparison of GA and Optimal Solution for 10$\boldsymbol{j}$ and 8$\boldsymbol{m}$

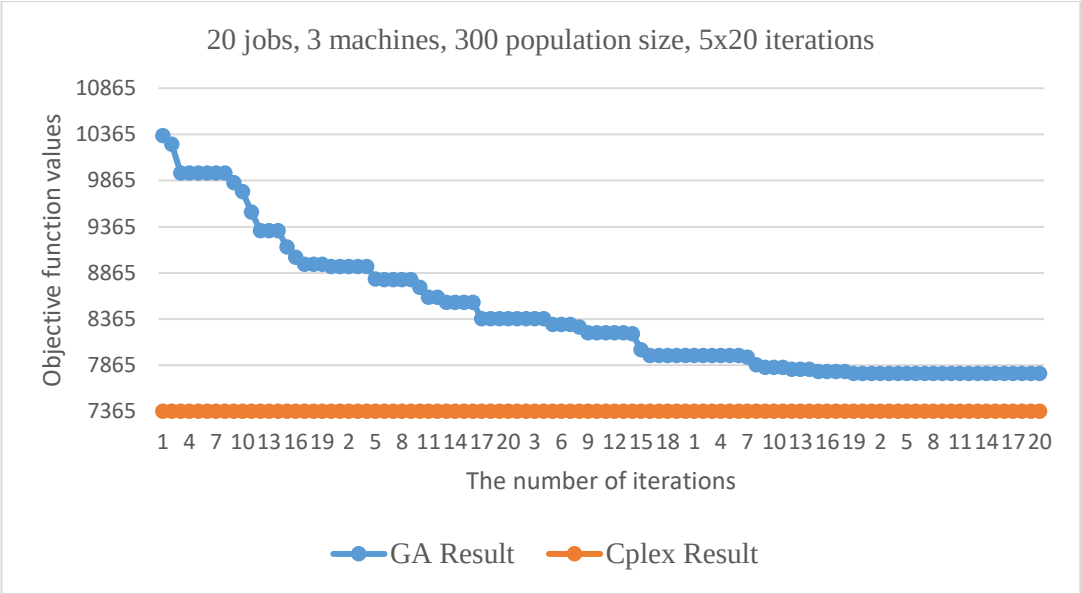

Figure 5.3. The Plot of Comparison of GA and Optimal Solution for 20$\boldsymbol{j}$ and 3$\boldsymbol{m}$

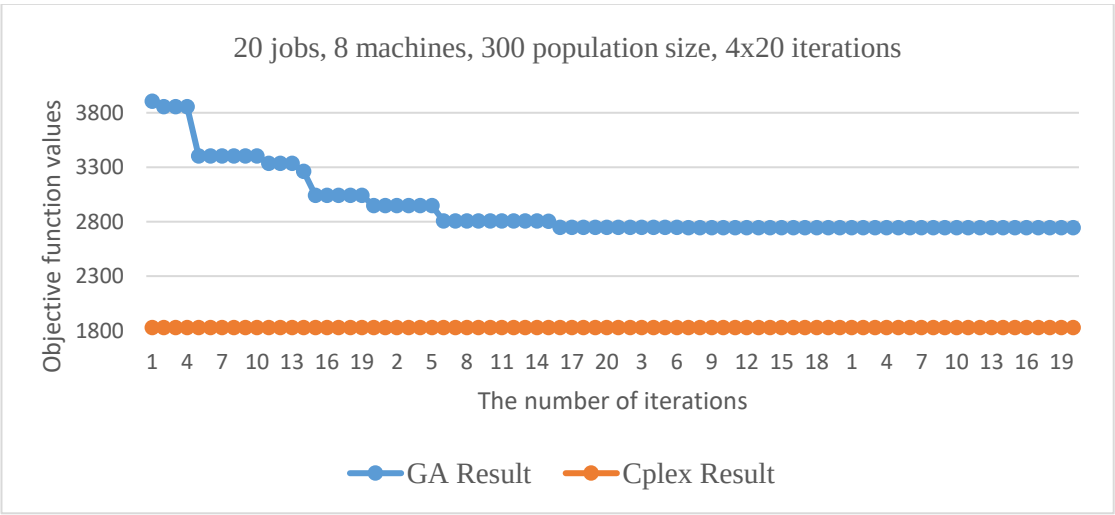

Figure 5.4. The Plot of Comparison of GA and Optimal Solution for 20$\boldsymbol{j}$ and 8$\boldsymbol{m}$

4. In the fourth parameter set, the population size was 300 and the iteration number was given as 50. In this case, the solutions obtained using the third parameter sets were included in the initial population. In addition, a rule-based algorithm (RBA) based on due dates and completion times was developed. The solution of RBA was also included in the initial population.

The rule-based algorithm (RBA) first considers the due dates of the jobs. Then it checks the completion times of the jobs. First, the due dates of the jobs are sorted from

earliest to latest. Starting from the first job in the list, a job is assigned to the machine for which the completion time of that job is the shortest. This process is repeated until all jobs are assigned to the machines. The result of this algorithm is given in Table 5.3. Since this approach is calculated manually, the solution time is entered in the table as N/A.

5. In the fifth parameter set, giving the iteration number "1" was obtained 50 different solutions by keeping the population size as 300 in all combinations. Then, the best solutions were selected from these 50 different solutions.

In the below table, "n" is the number of iterations. "ps" is the population size. All solution approaches and Cplex solutions were compared in Table 5.1. Optimal results were found by Cplex in combinations of 10j x 3m and 10j x 8m. However, in combinations of 20j x 3m and 20j x 8m, a time limit of 3600 minutes was given, and the solutions were obtained by Cplex. Following Table 5.1., the GA solution could not be obtained in the combination of a population number of 200 and 20 jobs and 8 machines. Additionally, the repeat number column means the number of times the best fitness function repeats within the given population size. The solution times were given in Table 5.2.

Table 5.1. The Comparison of GA and Optimal Solutions Results

10*j* 3*m*					**20*j* 3*m***				
GA parameters	**Repeat Number**	**GA Result**	**OPL Result**	**GAP**	**GA parameters**	**Repeat Number**	**GA Result**	**OPL Result**	**GAP**
1	185	2058	2035	1.13%	1	84	7516	7365	2.05%
2	182	2062	2035	1.33%	2	156	7451	7365	1.17%
3	50	2058	2035	1.13%	3	26	7776	7365	5.58%
4	50	2058	2035	1.13%	4	50	7776	7365	5.58%
5	1	2157	2035	6.00%	5	1	8817	7365	19.71%
10*j* 8*m*					**20*j* 8*m***				
GA parameters	**Repeat Number**	**GA Result**	**OPL Result**	**GAP**	**GA parameters**	**Repeat Number**	**GA Result**	**OPL Result**	**GAP**
1	182	524	446	17.49%	1	159	2547	1829	39.26%
2	175	562	446	26.01%	2	-	-	1829	-
3	25	487	446	9.19%	3	34	2743	1829	49.97%
4	50	487	446	9.19%	4	28	2028	1829	10.88%
5	1	566	446	26.91%	5	1	2845	1829	55.55%

Table 5.2. The Solution Times of the GA

#	GA Parameters	10*j* x 3*m*	10*j* x 8*m*	20*j* x 3*m*	20*j* x 8*m*
0	**GA Time**	00:11:24:20	00:00:09:34	00:30:00:00	00:30:00:00
1	***n***=200; ***ps***=100	04:38:28:27	16:57:27:65	24:37:29:70	26:14:02:00
2	***n***=200; ***ps***=200	4 days	5 days	7 days	6 days
3	***n***=20; ***ps***=300 repeating until the same result in all population	08:40:36:00	07:03:23:00	14:51:56:00	14:22:09:34
4	***n***=50; ***ps***=300 chromosome sequences in the 3rd approach and the RBA result are given as input	07:41:03:00	06:25:24:17	07:44:58:20	09:03:43:12
5	***n***=1; ***ps***=300 50 solutions are taken and the best solution among them is kept	07:41:03:00	06:25:24:17	07:44:58:02	09:03:43:12

Table 5.3. The Rule Based Algorithm and Optimal Solutions Results

10*j* 3*m*				**20*j* 3*m***			
Repeat Number	**The RBA Result**	**OPL Result**	**GAP**	**Repeat Number**	**The RBA Result**	**OPL Result**	**GAP**
N/A	2284	2035	12.24%	N/A	9130	7365	23.96%
10*j* 8*m*				**20*j* 8*m***			
Repeat Number	**The RBA Result**	**OPL Result**	**GAP**	**Repeat Number**	**The RBA Result**	**OPL Result**	**GAP**
N/A	558	446	25.11%	N/A	2058	1829	12.52%

When we compare the solutions within themselves, the best solution for 10j x 3m is in the 1st, 3rd, and 4th parameter sets and the GAP is 1.13%. For 20j x 3m the best solution is in 2nd parameter and the GAP is 1.17%. For 10j x 8m the best solution is in parameter set 3 and 4 and the GAP is 9.19%. The best solution for 20j x 8m is in parameter set 4 and the GAP is 10.88%. When we compare the results with Cplex solutions, the biggest gap is in the combination of 20j x 8m, and the best solution is in the combination of 10j x 3m. If we compare the solutions, a maximum deviation of 10.88% is accepted and the genetic algorithm is good. At the same time, the gap increases as the

complexity of the problem increases. However, if the solution time of the genetic algorithm is interpreted, it is seen that the times are long. This shows that genetic algorithm coding can be coded in programs that can be more efficient in future studies.

5.2 The Randomized Search Algorithm

The randomized search algorithm developed for the fuzzy MILP model considered $P/M_i/C_{max}$ scheduling problem with identical parallel machines, using machine eligibility, fuzzy sequence-dependent set-up times, and fuzzy processing times, to find the makespan. We conduct a numerical study to present the solution quality of the proposed algorithm. We generate 200 problem instances with different combinations of number of jobs, and number of machines: number of jobs can be 20, 30, 40, 50, and 60; number of machines can be 2, 4, 6, and 8; for each pair of number of jobs and number of machines 10 problems are randomly generated with different randomly generated fuzzy processing and setup times so that defuzzified representations are from Uniform(10, 80) and Uniform(20, 40), respectively. For the numerical study it is assumed that all machines are eligible to process all the jobs. For each problem instance, we run the randomized search algorithm with three different total number of replications: $10^3, 10^4$ and 10^5. We also obtain the OPL CPLEX 20.1.0 solutions under 3600 minutes time limitation (OPL Cplex code is given in Appendix 1).

Table 5.4. The Percentage Gap Between the Randomized Algorithm and Mathematical Model Solutions

	10^5 replications					**10^4 replications**					**10^3 replications**				
m/i	**20**	**30**	**40**	**50**	**60**	**20**	**30**	**40**	**50**	**60**	**20**	**30**	**40**	**50**	**60**
2	3.51	6.32	7.09	8.49	9.04	4.18	6.99	7.62	9.27	9.65	5.22	7.79	8.23	10.1	10.04
4	3.43	4.73	5.96	6.37	6.91	4.33	5.38	6.62	6.94	7.34	5.58	6.47	7.4	8.07	8.19
6	2.31	3.03	3.82	4.86	5.49	3.47	3.98	4.36	5.62	5.89	3.96	4.7	5.45	6.38	6.96
8	2.00	2.39	2.17	3.63	3.44	2.76	3.32	2.68	4.26	4.06	4.1	4.97	3.7	4.88	5.02
Avg.	**2.81**	**4.12**	**4.76**	**5.84**	**6.22**	**3.69**	**4.92**	**5.32**	**6.52**	**6.74**	**4.72**	**5.98**	**6.19**	**7.36**	**7.55**
G. Avg.	**4.75**					**5.44**					**6.36**				

An exhaustive search for the optimal solution requires a prohibitive long time as the number of machines and jobs increase. This confirms that the optimal solution is in the NP-hard class, and we set a time limit of 3600 seconds for OPL Cplex. Alternatively, we proposed the randomized search algorithm and implemented it in

C++. We compare the solutions obtained by the algorithm and the solution of the mathematical model and report the % gap between them in Table 5.1. Intuitively, as the number of replications increase the randomized algorithm starts performing better and in return the % gap decreases in all test instances. In all test instances we observe that randomized algorithm never beats the mathematical model solution with the time limit, although the gap between is around 10% in the worst case. The performance of the heuristic aggravates as the number of jobs increases and the number of machines decreases. This is commensurate with the complexity of the problem and indicates that as the number of feasible solutions increases the randomized heuristics can only search a smaller fraction of those solutions, yielding a higher deviation from the best solution by the math model. This suggests employment of smarter search algorithms instead of purely randomized search algorithms.

As there are more jobs and machines, we observe that the algorithm's performance time grows. The increase is convex in terms of the number of machines but linear in terms of jobs. As an illustration, consider 40 jobs case when the replication count is 10^5, the run time of the algorithm are 29.47, 109.35, 141.95, and 165.23 seconds for $m = 2, 4, 6,$ and 8, respectively. Consider next, the 8 machines case when the replication count is 10^5, the run time of the algorithm are 49.64, 90.82, 165.23, and 253.25 seconds for $n = 20, 30, 40,$ and 60, respectively. The run times are well below the 3,600 seconds limit of the OPL solution.

CHAPTER 6
CONCLUSION

This thesis considered a real-life parallel machine scheduling problem with sequence dependent setup times, machine eligibility, and job splitting features. We first focus on the deterministic settings and developed six different MILP optimization models starting from the simplest one to the one covering all the problem features. In order to cover different real-life applications, we considered three different objective functions: makespan, total completion times of the jobs, and total earliness and tardiness of the jobs. We designed a Genetic Algorithm to solve the MILP models that are known to be NP-hard. The performance of the proposed algorithm was evaluated by comparing GA solutions with the OPL Cplex solutions. As a result of the comparisons, it can be said that the optimality gap of GA increases with the problem complexity. The best GA solution was for 3 machines-10 jobs case with an optimality gap of 1.13%. On the other hand, the largest gap was 10.88% that is for 20 jobs-8 machines case.

As an extension, we also considered fuzzy sequence-dependent setup times and fuzzy processing times. For the fuzzy model, we kept machine eligibility constraints and sequence dependent setup times. As a heuristic approach we developed a randomized search algorithm, which was coded in C++. Two-hundred problem instances were generated using various numbers of machines and jobs. Number of jobs can be 20, 30, 40, 50, and 60; number of machines can be 2, 4, 6, and 8; and for each pair of number of jobs and number of machines, 10 problems are randomly generated with different randomly generated fuzzy processing and setup times so that defuzzified representations are from Uniform(10, 80) and Uniform(20, 40), respectively. In all test instances we observe that randomized algorithm never beats the mathematical model solution with the time limit, although the gap between is around 10% in the worst case.

As a future work, GA algorithm can be enriched to handle the cases with job splitting and facility eligibility. Since the genetic algorithm was coded and run in VBA Macro, the solution times took long times. As another future work all the coding practices can be handled via MATLAB or Python and thus quicker solutions can be reached even

with new set of constraints . Consequently, the developed models and solutions algorithms can be better applied to real life problems.

REFERENCES

Ahmadizar, Fardin, and Leila Hosseini. (2011). Single-machine scheduling with a position-based learning effect and fuzzy processing times. *The International Journal of Advanced Manufacturing Technology*, 56.5, 693-698.

Balin, S. (2011). Non-identical parallel machine scheduling using genetic algorithm. *Expert Systems with Applications*, 38(6), 6814-6821.

Bastos, C. E. N., & Resendo, L. C. (2020). Two-step approach for scheduling jobs to non-related parallel machines with sequence dependent setup times applying job splitting. *Computers & Industrial Engineering*, 145, 106500.

Berthier, A., Yalaoui, A., Chehade, H., Yalaoui, F., Amodeo, L., & Bouillot, C. (2022). Unrelated parallel machines scheduling with dependent setup times in textile industry. *Computers & Industrial Engineering*, 108736.

Biskup, D. (1999). Single-machine scheduling with learning considerations. *European Journal of Operational Research*, 115, 173–178.

Biskup, D., Herrmann, J., & Gupta, J. N. (2008). Scheduling identical parallel machines to minimize total tardiness. *International journal of production economics*, 115(1), 134-142.

Edis, E. B., & Ozkarahan, I. (2011). A combined integer/constraint programming approach to a resource-constrained parallel machine scheduling problem with machine eligibility restrictions. *Engineering Optimization*, 43(2), 135-157.

Ekici, A., Elyasi, M., Özener, O. Ö., & Sarıkaya, M. B. (2019). An application of unrelated parallel machine scheduling with sequence-dependent setups at Vestel Electronics. *Computers & Operations Research*, 111, 130-140.

Fanjul-Peyro, L., Ruiz, R., & Perea, F. (2019) Reformulations and an exact algorithm for unrelated parallel machine scheduling problems with setup times. *Computers & Operations Research*, 101, 173-182.

Figlali, A., & Engin, O. (2011). Akış tipi çizelgeleme problemlerinin genetik algoritma yardımı ile çözümünde uygun çaprazlama operatörünün belirlenmesi. *Doğuş Üniversitesi Dergisi*, *3*(2), 27-35.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. In *Annals of discrete mathematics* (Vol. 5, pp. 287-326). Elsevier.

Haghnevis, M., Khodadadeghan, Y., Jolai, F., & Tavakkoli-Moghaddam, R. (2006). A Mathematical Model of a Multi-Criteria Parallel Machine Scheduling Problem: A Genetic Algorithm (RESEARCH NOTE). *International Journal of Engineering*, *19*(1), 79-86.

Hamzadayi, A., & Yildiz, G. (2017). Modeling and solving static m identical parallel machines scheduling problem with a common server and sequence dependent setup times. *Computers & Industrial Engineering*, 106, 287-298.

Joo, C. M., & Kim, B. S. (2012). Non-identical parallel machine scheduling with sequence and machine-dependent setup times using meta-heuristic algorithms. *Industrial Engineering and Management Systems*, 11(1), 114-122.

Kim, H. J., & Lee, J. H. (2021). Scheduling uniform parallel dedicated machines with job splitting, sequence-dependent setup times, and multiple servers. *Computers & Operations Research*, 126, 105115.

Körez, M. T. (2005). Sıralı akış tipi çizelgeleme problemlerinde genetik algoritma uygulaması.

Li, Kai, et al. (2019). Uniform parallel machine scheduling with fuzzy processing times under resource consumption constraint. *Applied Soft Computing* 82, 105585.

Mensendiek, A., Gupta, J. N., & Herrmann, J. (2015). Scheduling identical parallel machines with fixed delivery dates to minimize total tardiness. *European Journal of Operational Research*, 243(2), 514-522.

Min, L., & Cheng, W. (1999). A genetic algorithm for minimizing the makespan in the case of scheduling identical parallel machines. *Artificial Intelligence in* Engineering, 13(4), 399-403.

Mokotoff, E. (2004). An exact algorithm for the identical parallel machine scheduling problem. European Journal of Operational Research, 152(3), 758-769.

Nagar, A., Haddock, J., & Heragu, S. (1995). Multiple and bicriteria scheduling: A literature survey. *European journal of operational research*, 81(1), 88-104.

Omar, M. K., & Teo, S. C. (2006). Minimizing the sum of earliness/tardiness in identical parallel machines schedule with incompatible job families: An improved MIP approach. *Applied Mathematics and Computation*, 181(2), 1008-1017.

Peng, Jin, and Baoding Liu. (2004). Parallel machine scheduling models with fuzzy processing times. *Information Sciences* 166.1-4, 49-66.

Pinedo, M. L. (2008). *Scheduling: Theory, Algorithms, and Systems*. 3rd edn. New York, NY: Prentice Hall

Rocha, P. L., Ravetti, M. G., Mateus, G. R., & Pardalos, P. M. (2008). Exact algorithms for a scheduling problem with unrelated parallel machines and sequence and machine-dependent setup times. *Computers & Operations Research*, 35(4), 1250-1264.

Su, L. H. (2009). Minimizing earliness and tardiness subject to total completion time in an identical parallel machine system. *Computers & operations research*, 36(2), 461-471.

Tahar, D. N., Yalaoui, F., Chu, C., & Amodeo, L. (2006). A linear programming approach for identical parallel machine scheduling with job splitting and sequence-dependent setup times. *International journal of production economics*, 99(1-2), 63-73.

Unlu, Y., & Mason, S. J. (2010). Evaluation of mixed integer programming formulations for non-preemptive parallel machine scheduling problems. *Computers & Industrial Engineering*, 58(4), 785-800.

Vallada, E., & Ruiz, R. (2011). A genetic algorithm for the unrelated parallel machine scheduling problem with sequence-dependent setup times. *European Journal of Operational Research*, 211(3), 612-622.

Wang, Chengyao, et al. (2002). The single machine ready time scheduling problem with fuzzy processing times. *Fuzzy sets and systems* 127.2, 117-129.

Wojakowski, P., & Warżołek, D. (2014). The classification of scheduling problems under production uncertainty. *Research in Logistics & Production,* 4(3), 245-256.

Xue, Feng, Wansheng Tang, and Ruiqing Zhao. (2008). The expected value of a function of a fuzzy variable with a continuous membership function. *Computers & Mathematics with Applications* 55.6, 1215-1224.

Yeh, Wei-Chang, et al. (2014). Parallel machine scheduling to minimize makespan with fuzzy processing times and learning effects. *Information Sciences,* 269, 142-158.

Yu, C., Semeraro, Q., & Matta, A. (2018). A genetic algorithm for the hybrid flow shop scheduling with unrelated machines and machine eligibility. *Computers & Operations Research*, 100, 211-229.

APPENDIX 1 – OPL Cplex Code of Model-6 and Fuzzy Model

Model 6:

```
execute PARAMS {

  cplex.tilim=3600; //Time limit
}

int jobs=...;
int machine=...;
range i =0..jobs;
range j =1..jobs;
range m=1..machine;

//set and indices

int P[i]=...;
int S[i][j]=...;
int Q[i]=...;
int d[i]=...;
int r[i]=...;
int M[i][m]=...; //machine eligibility

//decision variables

dvar boolean X[i][j][m];
dvar int+  C[i][m];
dvar int+  T[i];
dvar int+  E[i];


minimize sum(i1 in j)(T[i1]+E[i1]); //objective function

subject to{

forall(j1 in j)
 sum(m1 in m, i1 in i:i1!=j1)X[i1][j1][m1]*M[i1][m1]*M[j1][m1] ==
1;

forall(m1 in m, j1 in j, k1 in j:k1!=j1)
  X[j1][k1][m1]*M[j1][m1]*M[k1][m1]<= sum(i1 in i:i1!=j1)
X[i1][j1][m1]*M[i1][m1]*M[j1][m1];

forall(m1 in m, i1 in i, j1 in j:j1!=i1)
  C[j1][m1]-C[i1][m1]>= S[i1][j1]+(Q[j1]*P[j1])-100000*(1-
X[i1][j1][m1]);

forall(m1 in m, i1 in i, j1 in j:j1!=i1)
  C[j1][m1]-S[i1][j1]-Q[j1]*P[j1]>=r[j1]-100000*(1-X[i1][j1][m1]);

forall(j1 in j,m1 in m)
  C[j1][m1]<= 100000*Q[j1];
```

```
forall(m1 in m, i1 in i)
  sum(j1 in j:j1!=i1)X[i1][j1][m1]*M[i1][m1]*M[j1][m1]<= 1;

forall(j1 in j, m1 in m)
T[j1]>=C[j1][m1]-d[j1];

forall(j1 in j,m1 in m)
E[j1]>=d[j1]-C[j1][m1];

forall(j1 in j)
  T[j1] >= 0;

forall(j1 in j)
  E[j1] >= 0;
}
```

Fuzzy Model:

```
execute PARAMS{
   cplex.tilim=3600; //Time limit
}

}
//set and indices
int jobs=...;
int machine=...;
range j =0..jobs;
range q =1..jobs;
range m=1..machine;

float P[j]=...;
float S[j][q]=...;

//decision variables
dvar boolean x[j][q][m];
dvar float+  Cmax;
dvar float+  C[j];

minimize Cmax; //objective function

subject to{

forall(q1 in q)
  sum(m1 in m, j1 in j:j1!=q1) x[j1][q1][m1] == 1;

forall(m1 in m)
  sum(q1 in q) x[0][q1][m1]== 1;

forall(m1 in m, j1 in j:j1!=0)
  sum(q1 in q) x[j1][q1][m1]<=sum(p1 in j) x[p1][j1][m1];
```

```
forall(m1 in m)
Cmax>= sum(q1 in q) sum (j1 in
j)((S[j1][q1]+P[q1])*x[j1][q1][m1]);

forall(m1 in m, j1 in j, q1 in q:q1!=j1)
 c[q1]-P[q1]>= C[j1]+ S[j1][q1]-10000*(1-x[j1][q1][m1]);

 forall(j1 in j)
  Cmax>= C[j1];
}
```

APPENDIX 2 – PROBLEM DATA OF MODEL-6

Seq No	Unit Process Time	Quantity	Ready Date	Due Date
1	7	49	23	300
2	4	49	22	250
3	9	27	30	180
4	1	50	35	50
5	9	27	22	250
6	3	37	41	150
7	2	48	21	100
8	7	14	25	100
9	10	42	31	430
10	6	38	40	250
11	8	30	25	250
12	2	44	19	100
13	3	27	48	85
14	9	15	31	200
15	2	33	20	500
16	5	37	37	250
17	1	30	18	100
18	8	33	24	300
19	8	41	32	400
20	1	41	17	50

Machine Eligibility Matrix (20 jobs x 8 machines)

M_i	1	2	3	4	5	6	7	8
1	0	1	1	1	1	0	1	0
2	1	1	0	1	1	0	0	1
3	1	1	0	0	1	1	0	0
4	1	0	0	0	1	1	1	1
5	1	1	1	1	1	1	1	1
6	0	1	0	1	1	0	0	0
7	1	0	1	0	1	1	1	1
8	1	0	1	0	1	1	0	0
9	0	1	1	1	1	1	0	1
10	1	0	1	1	1	0	1	1
11	1	1	0	1	1	0	1	1
12	0	1	0	0	1	0	0	1
13	1	0	1	1	1	1	1	1
14	1	1	0	0	1	0	0	0
15	1	0	1	1	1	1	1	1

16	1	0	1	1	1	1	1	1
17	0	1	0	0	0	0	0	1
18	1	0	1	1	0	1	0	1
19	1	0	0	1	1	0	1	0
20	0	0	1	0	0	0	1	1

Setup Time Matrix (20 jobs x 20 jobs)

S_{ij}	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
1	40	36	41	22	21	17	32	26	42	35	39	41	22	22	32	29	41	43	28	24
2	21	40	31	24	23	49	39	39	20	27	19	38	21	18	23	39	42	36	18	27
3	27	43	22	29	47	50	33	38	34	26	38	41	32	17	41	49	40	26	38	36
4	40	24	48	22	47	34	42	20	47	30	16	36	25	15	44	23	45	20	31	35
5	49	49	24	19	41	27	27	45	32	39	34	27	25	25	21	44	18	39	43	19
6	18	22	35	49	34	23	41	44	26	46	35	38	40	41	35	25	27	29	40	26
7	28	45	20	43	35	49	25	45	15	41	30	36	47	22	28	27	41	42	20	28
8	26	18	41	35	38	46	36	32	22	47	49	43	43	22	33	27	41	23	44	33
9	19	31	37	34	41	35	23	21	49	40	46	15	49	49	21	30	37	35	21	25
10	32	47	41	30	25	35	25	36	20	40	16	28	50	18	34	47	16	39	44	26
11	35	19	33	17	25	43	34	29	16	30	17	41	44	25	20	35	21	49	49	39
12	31	37	16	25	43	40	18	30	32	30	26	22	38	17	42	19	39	37	34	33
13	36	50	43	18	30	34	48	31	49	19	41	40	23	16	25	27	49	39	27	40
14	29	50	19	44	30	33	41	30	40	45	47	37	27	41	23	29	43	19	45	15
15	29	31	26	25	17	40	15	45	39	26	41	48	39	16	43	23	35	49	44	30
16	44	36	18	30	29	47	33	32	35	43	38	27	19	36	29	33	24	23	42	41
17	17	16	32	36	19	22	46	20	44	43	38	18	18	25	38	39	24	19	31	37
18	28	46	33	22	39	42	39	25	32	37	41	27	15	49	15	46	28	21	48	45
19	23	28	15	15	37	22	20	45	41	20	20	17	35	50	20	33	15	31	17	33
20	20	46	21	20	18	17	39	21	36	46	47	26	21	44	24	44	25	26	26	40