NONIDENTICAL PARALLEL MACHINE SCHEDULING WITH TIME-DEPENDENT DETERIORATION OF JOBS

A Thesis

by

Mücahit Kaan Kaleli

Submitted to the
Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in the
Department of Industrial Engineering

Özyeğin University
June 2021

NONIDENTICAL PARALLEL MACHINE SCHEDULING WITH TIME-DEPENDENT DETERIORATION OF JOBS

Approved by:

Assistant Professor Z. Melis Teksan,
Advisor
Department of Industrial Engineering
Özyeğin University

Assistant Professor İhsan Yanıkoğlu
Department of Industrial Engineering
Özyeğin University

Assistant Professor Erinç Albey,
Co-Advisor
Department of Industrial Engineering
Özyeğin University

Assistant Professor Mehmet Önal
Department of Industrial Engineering
Özyeğin University

Date Approved: 14 June 2021

Professor Z. Caner Taşkın
Department of Industrial Engineering
Boğaziçi University

To all who always support and encourage dreamy people

ABSTRACT

We consider scheduling of deteriorating jobs on nonidentical parallel machines where the objective is to minimize mean flow time while ensuring the average value (i.e., quality) of processed jobs on machines exceeds a threshold. Deterioration is considered to be time-dependent and has a two-fold effect on jobs. The processing time of jobs as well as their values are deteriorating with time. We analyze the scheduling problem where the processing time and the value functions take linear (piece-wise linear) forms. We first formulate the problem as a mixed integer linear program and obtain optimal solutions for limited problem sizes. We then develop a heuristic algorithm to solve the problem and several ideas to sort jobs in the system. Linear and poisson regression models are trained to predict position of jobs on machines and also used in rank aggregation with simple sorting lists to provide better sorting approaches.

ÖZETÇE

Bu tez çalışmasında, özdeş olmayan paralel makinelere atanmış zamana duyarlı işlerin ortalama kalitesinin bir eşiği aşması sağlanırken, tüm işlerin ortalama akış süresini en aza indirmeyi amaçlayan bir çizelgeleme sorunu üzerine odaklanılmıştır. İşlerin sistemde geçirdikleri bekleme süresinin artması işlem sürelerini uzatırken kalitelerini ise zamanla bozmaktadır. İşlem süresi ve kalite fonksiyonlarının doğrusal formlar aldığı çizelgeleme problemini analiz ediyoruz ve çözmeye çalışıyoruz. Problem önce karışık tamsayılı bir lineer program olarak formüle ediliyor ve sınırlı problem boyutları için en iyi çözümler elde ediliyor. Daha sonra sorunu çözmek için sezgisel bir algoritma ve bekleyen işleri sıralamak için yeni fikirler geliştiriyoruz. Doğrusal ve poisson regresyon modelleri, işlerin makinelerdeki öncelik sırasını tahmin etmek için geliştirildi ve ayrıca daha iyi sıralama yaklaşımları sağlamak için sıra kümelemesi yöntemi kullanıldı.

ACKNOWLEDGEMENTS

First of all, I would like to thank Dr. Z. Melis Teksan for accepting me as a master student under her kind supervision. I have learned innumerable things from her for years. I am also beholden to Dr. Erinç Albey for his time allocated to me and precious advices that were always ready for me.

I feel very lucky to have studied in the Department of Industrial Engineering. Especially, classes of Dr. O. Örsan Özener and Dr. Enis Kayış have widened my horizon. I am grateful to have taken my both B.S. and M.S. degrees from Özyeğin University.

If I list everyone that has been helpful and supportive for me during time I spent in Özyeğin University, it would be a long list. Therefore, I would like to thank all my friends and personnel of the university one by one. I always will remember everyone with good memories.

Finally, I would like to express my deepest appreciation to my family. They have always been there for me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Classical deterministic scheduling theory assumes predetermined, known, and constant job processing times for the entire scheduling period. However, in some practical settings the constant processing time assumption leads to oversimplified representation of the scheduling problem under consideration. It is not uncommon that the processing times of the jobs change (increase or decrease) during the scheduling period as their start time on the machines are delayed. This can be observed in manufacturing settings, maintenance operations, agricultural settings, and in any scheduling environment where the properties of the jobs change with time. The time-dependent properties of jobs include, but are not limited to, processing times and job values that are particularly considered in this study that focuses on nonidentical parallel machine scheduling problems with time-dependent job deterioration (deterioration of both job processing times and values) where the objective is to minimize the mean flow time and to ensure that the total value generated by all processed jobs exceeds some threshold.

Deterioration appears in various forms in the scheduling problems where the causes for deterioration differ by the problem setting. For instance, machine conditions may deteriorate, i.e., they may lose efficiency as jobs are processed. Therefore, later jobs may have longer processing times compared to earlier ones (see, for example, Cheng et al. [1]). In such settings, the change in the processing time depends on the decay of the machine performance. In some settings, the performance of the machine can be restored back to its original level, such that the processing times of the jobs are reverted to their initial values (see, for example, Rustogi and Strusevich [2], [3], and

[4]). Conversely, the machines (e.g., operators) may gain experience as they process more jobs. As a result, they may process later jobs faster. This phenomenon referred as learning effect in scheduling (see, for example, Yin and Xu [5]). In all these aforementioned settings, the change in the processing times of jobs depends on the machine conditions, which can be either restored if degrading the performance of the machine, or used as an advantage in scheduling the jobs if improving the performance. In the problem setting considered in this study, the machine performance is assumed to be stationary, however, the jobs deteriorate as their processing time increases. Thus, the time-dependent deterioration is only considered to be a consequence of the job characteristics.

Time-dependent deteriorating jobs appear in various settings where delaying the processing of the job results in extra time to perform it. Examples are maintenance or cleaning operations, steel rolling operations, or fire fighting work (see, for example, [6] and [7]). Any delay in processing the job is penalized by additional time to complete the job, because the job has some time-dependent properties which affect its processing time. Deterioration of jobs are considered to be time-dependent (see, for example, Wu, Shiau, and Lee [8], Cheng et al. [9], and Kuo and Yang [10]), or position-dependent (see, for example, Rustogi and Strusevich [2]), or dependent on both time and position (see, for example, Yin and Xu [5] and Yin et al. [11]). In our study, we consider that the processing time of a job is a non-decreasing function of its start time.

The time-dependent deterioration is a particularly relevant phenomenon in agricultural settings such as crops and plants harvesting and processing, dairy products processing. A particular application of the scheduling problem under investigation appears in smart farms where milking process of free-range dairy cattle takes place. In these type of settings, as the harvest time or the time of milking is delayed, the amount to be processed increases or takes more time. At the same time, the yield of

the product changes as time of the process is delayed. For instance, the protein content of the milk changes as the cattle waits longer to be processed and if it waits too long the protein content of the milk may be far away from the desired concentration. This observation motivates the study on scheduling problems with time-deteriorating jobs where their processing times and their value contribution depend on their start time.

In this study, we consider time-dependent deterioration of jobs where the deterioration depends on the nature of the job. Deterioration of a job has a two-fold effect on the job. As the start of a job is delayed, the processing time of the job increases and the value generated by the job decreases, named as quality of a job. We assume that average quality of jobs on a machine should satisfy a predefined quality lower bound. Jobs that cannot be placed on any machine due to low quality should be assigned to a dummy machine where there is not a quality threshold value. Thus, poor quality jobs can be saved from being spoiled and processed as a low quality batch. However, these rejected jobs that assigned to the dummy machine cause some kind of a penalty cost to the objective function. We first try to schedule jobs on regular machines and then to the dummy machine if it is necessary.

Scheduling environment consists of nonidentical parallel machines and, in the beginning of the scheduling period, the system is not empty, i.e., there may be jobs being processed on the machines as well as jobs waiting to be processed in the queues of the machines. We formulate the problem as a mixed integer programming model where we assume linear deterioration for processing times and piece-wise linear deterioration for job values and solve the model for various problem instances. Due to nature of the problem, heuristic solutions approaches are considered. Time dependency of jobs leads us focusing on the alternatives that respond quickly. We use simple sorting concepts that are popular in scheduling problems and try to develop these ideas with help of regression models and a famous rank aggregation method (Borda's Count).

Linear and poisson regression models are considered to predict (see, for example, [12], [13] and [14]) position of jobs that need to be scheduled on machines. We investigate that rank aggregation concept, used with various forms in mostly biological and computer science related cases (see, for example, [15] and [16]), can be beneficial in the studied scheduling problem.

The rest of the study is organized as follows. Chapter 2 reviews the related literature. Chapter 3 first provides the problem definition, then presents the notation and the mathematical model. Chapter 4 introduces the heuristic algorithm and sorting approaches. We discuss computational results in Chapter 5. Chapter 6 presents concluding remarks and possible future works.

CHAPTER II

PREVIOUS WORK

Scheduling of deteriorating jobs received attention in recent years. Cheng, Ding, and Lin [17] review scheduling problems with deteriorating jobs where they show the generalizations of problems in classical scheduling theory into the scheduling problems with time-deteriorating jobs. Gawiejnowicz [18] provides an extensive classification of time-dependent scheduling problems. The problem considered in this study falls into the class of parallel machine scheduling problems with time-deteriorating jobs. Thus, we review the parallel machine scheduling problems with deteriorating jobs and scheduling problems with job-value deterioration as most related past work. We also review single machine scheduling problems where they provide valuable insights for more general scheduling problems. Regression analysis over different data types in the literature are reviewed to help us construct regression models. Li, Wang and Xiao [15] categorize rank aggregation methods in detail. Various forms of rank aggregation methods and area of usage are also examined in the last section of this chapter.

2.1 Single Machine Scheduling Problems With Deteriorating Jobs

Single machine problems are widely studied in scheduling literature for various reasons. They provide easy-to-comprehend and investigate special cases for all other machine environments. The results provide insights and foundations for solution approaches for more general problems. This motivates researchers to study single machine environments also for settings with deteriorating jobs.

Wu, Shiau, and Lee [8] study makespan and total completion time minimization problems on a single machine with simple linear deterioration and group setups where

setup times also increase with time. Authors show that the problem can be solved in polynomial time. Cheng et al. [9] consider same problem setting with minimizing maximum tardiness objective where they provide a branch-and-bound algorithm to solve the problem. Wang and Wang [19] consider an extension to this study where they incorporate nonzero ready times and somewhat more general linear deterioration functions for processing and setup times. Here, it is assumed that some setup is needed when two jobs from different job groups are processed successively on the machine. Authors show that the case with no setups, i.e. where linear time-dependent deterioration and nonzero ready times are present, ordering the jobs in non-decreasing order of their ready dates (FIFO rule) yields an optimal sequence. They also provide the optimal sequencing rule for the case with group setups.

Lai, Lee, and Chen [6] consider time-dependent deterioration with setup times where the deterioration depends on the sum of processing times of the earlier jobs and the position of the job in the sequence. The deterioration function takes a more general form and it is assumed to be non-decreasing in all variables. Authors show that the SPT rule minimizes the makespan and the sum of completion times for a single machine when the non-decreasing deterioration function has some certain characteristics. Lee [20] extends these results where general deterioration and learning effects are present.

Wang and Wang [7] consider single machine schedule problem with resource allocation dependent processing times where jobs deteriorate with time. Here processing time of a deteriorating job can be controlled by the number of resources allocated to execute the job. Time-dependent deterioration of jobs are assumed to be linear following the form $p_j = a_j + bt$ where deterioration rate, b ($b > 0$) is the same among all jobs. The objective is to find the optimal resource allocation and processing sequence on the machine such that the total cost of resource usages and sum of completion times are minimized. Authors show that the problem is polynomially solvable and

propose an algorithm that solves the problem in $\mathcal{O}(n \log n)$ time.

Kuo and Yang [10] study the single machine case with deteriorating jobs where the processing time of a job j is a nonlinear function of a basic processing time and its start time taking the form $p_j = a_j(1 + t_j)^{\alpha}$. They show that the makespan minimization problem can be solved in polynomial time whereas only some special cases can be solved polynomially when the problem aims minimizing total completion time. Authors show that, for $\alpha \geq 1$, there exists an optimal schedule where the jobs are sequenced in non-decreasing order of their basic processing times when minimizing the sum of completion times. (For $0 < \alpha < 1$, they show a result for weighted sum of completion times where the weights and basic processing times satisfy some certain properties. There is no result for the case with no weights.)

Rustogi and Strusevich [2] consider general positional deterioration for job processing times where they assume that the deterioration is rooted from the efficiency loss of the machine with the number of jobs processed. They consider scheduling of maintenance activities together with sequencing the jobs with the objective of minimizing the makespan. They also consider the case where a maintenance activity does not necessarily restores the machine fully. Same authors also incorporate time-dependent deterioration in [3] for the same problem, and in [4], they only consider linear time-deterioration. In all of these studies, Rustogi and Strusevich assume that the deterioration is job-independent. In [21], same authors consider makespan minimization on a single machine with generalized job-dependent cumulative effect on job processing times. This type of deterioration assumes that the processing time of a job depends on the processing times of the jobs that are processed on the same machine prior to it.

Wang and Wang [22] consider start time dependent nonlinear deterioration of jobs. They show that single machine makespan minimization problem remains polynomially solvable even when the deterioration function takes the following nonlinear form: the

actual processing time of job j is equal to $p_j^A(t) = p_j(a + bt)^c$ $(a, b, c \geq 0)$ when the processing starts at time t. The SPT rule on the basic processing times (p_j) minimizes the makespan when $c \geq 1$. When $0 \leq c \leq 1$, scheduling the jobs in non-increasing order of their basic processing times, a.k.a LPT rule, yields the optimal solution for C_{max} problem. When $c \geq 1$, the problem with minimizing the sum of completion times can be solved in polynomial time, and SPT rule on the basic processing times minimizes the sum of completion times. When $0 < c < 1$, the problem with minimizing the sum of completion times is NP-hard. The authors show that a V-shaped optimal schedule exists. Using this property they propose a heuristic to obtain near optimal solutions.

Oron [23] considers a non-regular measure, minimizing total absolute deviation of completion times, where the processing times of jobs follow simple linear deterioration. He provides two heuristic algorithms where he utilizes some characteristics of optimal schedules.

Yin and Xu [5] generalize some results where they consider time-dependent deterioration and position-dependent learning simultaneously. According to their findings the SPT rule minimizes both makespan and the sum of completion times. Yin et al. [11] consider the problem where each job's deterioration depends on both the start time and the position of the job in the schedule. Authors show that makespan and total completion time minimization problems can be solved in $O(n \log n)$ time.

Our study is different than the ones mentioned above where we consider parallel machine case with the objective of minimization of average flow time. We also consider the job-value deterioration with the goal of satisfying a threshold total value with all processed jobs.

2.2 Parallel Machine Scheduling Problems With Deteriorating Jobs

Our work is mostly related with parallel machine scheduling literature where jobs are considered to be deteriorating. Cheng [24] shows that the parallel machine scheduling problem with simple linear deterioration where the objective is to minimize total completion times is NP-hard. Ji and Cheng [25] consider the same problem and they provide a fully polynomial approximation scheme for identical parallel machines. Processing time p_j of a job j is assumed to depend on its start time s_j; $p_j = \alpha_j s_j$ where α_j denotes the deterioration rate. Following the result of Mosheiov [26] for single machine scheduling problems with simple linear deterioration, Ji and Cheng [25] show that there exists an optimal solution where on each machine the jobs are sequenced in the nondecreasing order of their processing time growth rate. Kang and Ng [27] study the problem considering minimizing makespan criterion and where the time-dependent processing time of job j is given by $p_j = a_j + b_j t$ where $a_j, b_j > 0$. They show that an optimal solution exists where the jobs on a machine are sequenced in a nondecreasing order of a_j/b_j values. In our study, we consider cases where the processing time of a job can take different forms and the parallel machines are not necessarily identical. Therefore, processing time of the job does not only depend on the start time of the job, but also on the machine that the job is assigned for processing.

Ruiz-Torres, Paletta, and Perez [28] consider a setting where the processing times of jobs increase due to the deterioration of the machine. The deterioration of the machine depends on the sequence of jobs assigned to the machine for processing. The objective is to minimize the makespan. Authors show that the single machine version of the problem can be solved in polynomial time whereas the parallel machine case is NP-hard. To solve the latter case, they develop list scheduling algorithms and meta-heuristics. In our study, we consider deterioration of a job that depends on the

properties of the job itself, not on the machine. Therefore, each job is considered to have its own deterioration characteristics.

In our study, we consider a setting where the system is not empty at any time and the decisions are need to be made for the new jobs that come in to the system. We also incorporate the time-dependent job value deterioration.

2.3 Scheduling With Job Value Deterioration

Fisher and Krieger [29] study a single machine scheduling problem where the goal is to maximize total profit gained by processing the jobs where the profit of each job decreases with its start time. They provide a heuristic approach to solve the problem. Voutsinas and Pappis [30] work on the single machine problem with job value deterioration with time. Similar to the study of Fisher and Krieger [29], their objective is to maximize total job values and this study can be applied in remanufacturing settings. Raut, Swami, and Gupta [31] also work on single machine scheduling problem with time-deteriorating job values. It is shown that when the time dependent value function is non-increasing and concave the problem is NP-hard and when the value function is non-increasing and convex it is strongly NP-hard. The authors develop some heuristic approaches to solve the problem.

Janiak et al. [32] study a identical parallel machine scheduling problem with a job-value loss function that depends on the completion time of a job. They provide a complexity analysis of the problem where they show the NP-hardness of the general problem, and they work on some polynomially solvable special cases.

Unlike the studies mentioned above, our study considers time-dependent processing time deterioration of jobs together with time-dependent deterioration of job values. We also consider nonidentical parallel machines in our problem setting.

2.4 Regression Models for Various Data Types

Regression analysis for prediction is widely studied concept for years in the literature. Different regression-based models and machine learning methods are taken into consideration in regard to type of data. We focus on regression analysis since we try to develop models that predict order of jobs in a scheduling list with the aid of job properties. We consider start times of jobs as a continuous variable and order of jobs on machines as a count variable in two different models: Multiple linear regression and poisson regression. This case is also a main concern in [14] although it is not a scheduling problem.

Lichman and Smyth [33] focus on predicting user consumption rates in several areas such as post rates of users on a social media platform, music listening rates on a streaming service and user review rates on a reviewing platform. A poisson regression model is trained using user-item past data to predict user item consumption rates which is the response variable with nonnegative integer numbers. One benefit of this study is to sort consumers as predicted rates to see which customer is more valuable, so firms can focus them. Moskowitz and Chun [34] try to determine warranty prices in automobile industry using poisson regression model. The model responses are investigated based on producer risk behaviour. There are only two types of independent variable with different number of combinations to offer less costly warrant policies to customers. Zacharias and Pinedo [35] schedule patients' arrival to available medical facility slots under no-show of a patient at scheduled time and overbooking cases. Objective is to minimize the weighted sum of patients' waiting time, doctor's idle time and overtime. Poisson regression model is developed to predict overbooking level by the data which is generated by optimal solutions to use in a heuristic approach.

Hayat and Higgins [12] investigate which regression model is more accurate in nursing-health research. Since information is collected as count data, poisson regression outperforms linear regression in the field. Estimation method for poisson

regression which is maximum likelihood estimation works better than the method for linear regression which is ordinary least squares. They explain how the model is linearized with using logarithmic link function. ShahabiKargar et al. [13] study three machine learning methods such as linear regression, MARS and random forest in the prediction of surgery duration to schedule next surgeries. Target value is continuous variable and they show that random forest method outperforms other methods while improving current hospital method. Subramanian et al. [36] study effect of new scheduling method in medical facilities compared to traditional method. Statistics of both methods are demonstrated performing Student's t-test or Wilcoxon rank-sum test to show whether new method is significantly better than the old one.

Schlagkamp et al. [37] try to formulate parallel job scheduling problem as a mixed-integer linear program to outperform simple scheduling sorting idea while maximizing user satisfaction during job processes. Linear regression analysis is used to see relationship of user acceptable response time and jobs' runtime length. Asnaashari, McBean, Shahrour and Gharabaghi [14] perform linear and poisson regression to model watermain failure frequencies. Poisson regression generates more successful model with offering higher metric scores. Nasir and Dang [38] propose a MILP model to solve the home health care scheduling and routing problem. CPLEX is used to solve the model exactly and a variable neighbourhood search algorithm is performed as a heuristic approach. They develop two approaches to identify thresholds for parameters in the model: Bender's method and ROC curves. Bender's method uses logistic regression to interpret the threshold value of a parameter as probabilistic value. Klicos et al. [39] focus on improving performance of a scheduling algorithm that assign classes in a school. In order to predict how many times classes will be run over specific time periods, different regression models are performed. They finally choose linear regression model for the prediction and results show that performance of the current scheduling algorithm of the school is significantly improved.

Unlike the cases above, we aim to sort jobs that need to be scheduled on machines based on response variables of regression models.

2.5 Rank Aggregation Methods

Rank aggregation (RA) idea gained importance in the last decades. It is mostly evaluated in biological cases and computer-based algorithms. RA basically comes from the idea of combining input lists, including items, genes, etc., to provide a single better list which targets to increase performance of algorithms. Those lists are called as base ranker generally and several methods are already studied in the literature. They mostly work on items' positions or distributions in base ranker lists under well known concepts in statistics and operation research such as Markov chains or Monte Carlo simulations. However, several meta-search RA algorithms are existing as constructed differently from those.

Li, Wang and Xiao [15] categorize and test RA methods under simulated genomic data. All methods are divided into two groups: supervised and unsupervised RA. Training data can be constructed with some true ranks in supervised learning. However, they focus on other part of RA which includes Markov chain-based, distribution-based, optimization-based and non-optimization-based methods. Borda's Count is the well-known RA method using summary statistics of base ranker as mean, median, geometric mean and L2-norm. Markov chain-based RA tries to rank items according to stationary probabilities. Markov chains can be constructed in regard to position of them to each other. Optimization-based ones aim to minimize objective function which is a distance metric such as Kendall's or Spearman measures. Lin [40][41] focuses on heuristic RA methods while comparing between deterministic and stochastic algorithms. Borda's method and Markov chain-based RA are considered as deterministic algorithms. Cross entropy Monte Carlo (CEMC) simulation is an

iterative stochastic method that aims to satisfy distance measures. It is also investigated that how the methods work when aggregating lists' shape differs. For example, the aggregation can be done on both full lists or partial lists. Those lists also can be coming from different spaces. Aledo, Gamez and Rosete [42] try to speed up the aggregation process by partial evaluation. It uses well-known neighbourhood moves in meta-heuristics such as insertion, interchange and inversion. Partially evaluated lists are not significantly different from initially permutated lists taking Kendall's distance as a performance measure. Besides, these moves reduce time consumption considerably.

Domshlak, Gal and Roitman [16] propose several RA algorithms to provide better schema matching. Database schemata is used as base ranker such as XML, DTDs and HTML. Similarity matrices are constructed between the ranker to make an aggregation evaluation. García-Nové et al. [43] focus on the rank aggregation cyclic sequence problem which takes Kendall's tau metric as minimization criterion by evaluating dissimilarity between cyclic sequences. They also show that the problem also can be converted to Traveling Salesman Problem (TSP), so that Permutated Asymmetric TSP problem is defined with some updates to be a variant of the main problem. It is tried to prove that any optimal solution for both problems is same. Kolde, Laur, Adler and Vilo [44] consider robust rank aggregation according to scores of ranking in each gene list. Distribution of lists are important to score position of genes in lists. In such a way, significance of gene positions is tested to order them to get the aggregated final list. The experiment results this method outperforms RA method by mean scores. Pihur, Datta and Datta [45] mention that clustering is a widely used concept in machine learning and several clustering methods are available in the literature. Cross-entropy Monte Carlo based RA is taken into consideration to provide better clustering mechanism.

Klementiev, Roth and Small [46] develop an unsupervised learning algorithm for

RA problem and learning is achieved by iterative gradient search. The aggregation is divided into two parts: Training and evaluation algorithms. Base ranker is tried to combine linearly under both additive and exponential experiments. Choi et al. [47] show two classical RA methods such as Borda's and Markov chain based methods can be used in a problem named replacement of water mains. Benefit of rank aggregation in this case is to prioritize damaged pipes to replace to give more reasonable decisions. Aggregation is built under ordered sets generated according to some attributes of pipes. Yu, Li, Tang and Wang [48] adapt particle swarm optimization method into RA problem. It is updated with user preference and multi objectivity to linearize time complexity of the problem because one of main concerns in RA is exponentially increasing time complexity when number of ranking items is high. Deng, Han, Li and Liu [49] estimate quality of rankers which are divided into two parts as relevant and background entities in base lists under Bayesian concept. Quality-based sorted final aggregation lists perform better than common RA methods in the literature.

Our case dissociates from the problems above because we try to aggregate potential scheduling lists. Thus, generated final list can reduce average flow time of jobs need to be scheduled in the system.

CHAPTER III

PROBLEM DEFINITION, NOTATION, AND MATHEMATICAL MODEL

We consider a parallel machine scheduling problem where each job has a number of predefined alternative machines to be processed. In the problem setting under consideration, the jobs deteriorate as they spend time in the system without being processed on a machine. When a job deteriorates with time, its processing time increases and the yield that is gained from the job upon completion of its process decreases.

We consider scheduling in a system that is not necessarily empty, that is, there might be some jobs being processed on some machines, and the in-queues of the machines might be non-empty. The set of jobs that are present initially in the system is denoted by J_{old}. The scheduling process is triggered by a set of jobs, J_{new}, which enter to the system. In the scheduling process, we need to assign the new jobs on machines, and determine their processing sequence so that the average flow time ($\bar{F}$) of all jobs in the system is minimized and the overall yield obtained by processing of all jobs in the system exceeds a predetermined threshold value. Hence, there are several decisions to be taken in this process:

- The machine assignment of each job $j \in J_{new}$.
- The processing sequence of each job $j \in J_{new}$ on its assigned machine.
- The processing sequence of each job $j \in J_{old}$ that are currently at an in-queue of its assigned machine.

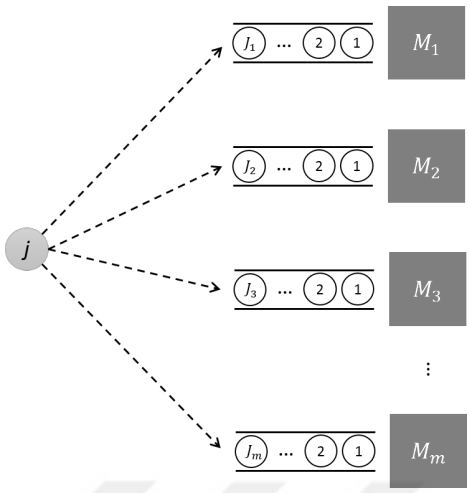


Figure 1: Job j Needs to Be Dispatched to One of Its Alternative Machines to Be Processed.

We denote the set of machines in the system with M. At any particular time, there can be jobs present in the system either being processed on a machine or waiting in the queue of its assigned machine. For the scheduling problem under consideration, we can easily show that a machine can not be left idle if there are some jobs waiting in the in-queue of that machine. That is, any schedule with inserted idle times can not be optimal for this case. Hence, at time of new job arrivals, there can be a job that is being processed on machine m. Let rp_m denote the remaining processing time of the job that is in-progress on machine m. Preemption is not allowed in this system, therefore any job that is being processed on a machine may not leave the machine (and the system) until the completion of its process.

There can be some jobs that are already assigned to some machines and waiting in the in-queues of those machines. The binary parameter a_{im} denotes whether job $i \in J_{old}$ is assigned to be processed on machine m. A job $i \in J_{old}$ has already spent some time in the system while waiting in the queue. We let W_i' denote the waiting time of job $i \in J_{old}$ up until the time of scheduling process starts.

We consider scheduling of jobs with time deterioration, and we define time-deterioration of jobs as follows: As the job waits for its processing on a machine,

its processing time on that machine increases and the value to be generated by its processing decreases. Hence, we define the processing time and value functions for each job. The processing time of job i on machine m is a nonnegative and nondecreasing function of its waiting time before the operation on machine m. We let $f_{im}(\tau)$ be the processing time function of job i on machine m, where τ is the time that elapses between the entrance of job i to the system and the start of the process of job i on machine m. In this study, we define $f_{im}(\tau)$ as a nondecreasing linear function:

$$f_{im}(\tau) = \alpha_{im} + \beta_{im}\tau, \quad \forall i, m,$$

where $\alpha_{im}, \beta_{im} \geq 0$ for all $i \in J$ and for all $m \in M$. Here, α_{im} and β_{im} are called the base processing time and the processing time deterioration rate of job i on machine m, respectively. We assume that job value (i.e., quality) degrades as the job waits to be processed. To represent this idea, we define the quality of job i on machine m as a non-negative and non-increasing function of its waiting time before being processed on machine m. We define $q_{im}(\tau)$ be the final achieved quality of job i on machine m, where τ is the time waiting time job i. In this study, we assume that the value function takes the following piece-wise linear form:

$$q_{im}(\tau) = \max\{\gamma_{im} - \theta_{im}\tau, \epsilon_{im}\}, \quad \forall i, m,$$

where $\gamma_{im}, \theta_{im}, \epsilon_{im} \geq 0$ for all $i \in J$ and for all $m \in M$. Here, γ_{im} and ϵ_{im} are, respectively, the maximum and the minimum value that can be generated by job i when processed on machine m. Here, θ_{im} is the value deterioration rate of job i when processed on machine m. The idea is that quality of a job stops to decrease when minimum value, ϵ_{im}, is reached even if it continues to wait in the queue and it is the lowest value that can be provided by this job. Table 1 provides a summary of all relevant sets and parameters.

Table 1: Summary of Sets and Parameters.

Notation	Description
J_{new}	Set of new jobs that needs to be dispatched.
J_{old}	Set of jobs that is already in the system.
J	Set of all jobs, where $J = J_{new} \cup J_{old}$.
J'	Set of all jobs for single machine scheduling problem.
i, j	Indices for jobs, where $i, j \in J$.
M	Set of all machines.
m	Index for machines, where $m \in M$.
d	Index for a dummy machine, where $d \in M$.
a_{im}	1 if job $i \in J_{old}$ is assigned to machine m, and 0 otherwise.
W'_i	The waiting time of job i in the shop floor so far where $i \in J_{old}$.
rp_m	The remaining processing time of in-process jobs on machine m.
$f_{im}(\cdot)$	The processing time function of job i on machine m.
α_{im}	The base processing time of job i on machine m.
β_{im}	The processing time deterioration rate of job i on machine m.
$q_{im}(\cdot)$	The value function of job i when assigned to machine m.
γ_{im}	The maximum value that can be generated by job i on machine m.
θ_{im}	The value deterioration rate of job i on machine m.
ϵ_{im}	The minimum value that can be generated by job i on machine m.
V	A big number.
QLB_m	Targeted minimum average quality on machine m.

The goal of this scheduling problem is to assign each of the new jobs to a machine as well as a processing sequence such that the average flow time of all jobs in the system is minimized and average value generated by all jobs in a machine exceeds a given threshold value. Therefore, QLB_m is defined in table 1 which is quality lower bound (QLB) for machine m. Average quality of jobs on same machine must satisfy QLB. To achieve this goal, the processing sequence of the jobs that are already in the system and waiting in the in-queue of their assigned machine can change as well. We denote the machine assignment decision of job i by X_{im} where $m \in M$; X_{im} equals 1 if job i is assigned to machine m and 0, otherwise. The processing sequence of jobs assigned to a machine is captured by the decision variable Y_{ijm} which takes value 1 if job i precedes job j on machine m and 0, otherwise. We also consider the case of rejection of a job from all machines. If a job cannot be put in-queue of any machine

m due to the quality restriction, it must be assigned to dummy machine d. Thus, T_i is a binary variable that takes value 1 if job i is assigned to one of the regular machines and 0 if it is assigned to dummy machine d. We divide machines as regular machines and the single dummy machine in the next chapters. The dummy machine is for saving very low quality jobs for being unassigned because there is not any quality restriction on it. However, remaining processing time for dummy machine, rp_d, is significantly higher than the time for regular machines rp_m. Assigning job i to the dummy machine causes a sharp increase in the objective function since the remaining processing times affect flow time of jobs substantially. In such a way, the model is preserved to give unreasonable decisions for jobs and rp_d can be considered as a kind of penalty cost to the objective function.

We denote the flow time of a job i as F_{im} when processed on machine m. The start time, and the waiting time of a job i when processed on machine m are denoted by S_{im} and W_{im}, respectively. The processing time of job i is not known in advance and depends both on the waiting time in the shop floor and the machine selection. Hence, the processing time of job i on machine m, P_{im}, is also a decision variable of the problem which depends on the waiting time, W_{im}, of job i on machine m. Similarly, the value of job i when processed on machine m, Q_{im}, is also a decision variable which depends on the waiting time of job i and the machine selection.

Table 2: Summary of Decision Variables.

Notation	Description
X_{im}	Binary variable indicating machine assignment of job i on machine m.
Y_{ijm}	Binary variable indicating processing sequences of jobs i and j. It takes value 1 if job i precedes job j on machine m, 0 otherwise.
Z_{im}	Binary variable indicating quality level of job i on machine m.
T_i	Binary variable indicating the regular machine assignment of job i.
P_{im}	Processing time of job i on machine m.
F_{im}	Flow time of job i on machine m.
S_{im}	Start time of job i on machine m.
W_{im}	Waiting time of job i on machine m.
Q_{im}	Quality level of job i on machine m.

The goal is to dispatch new jobs and to resequence old jobs such that the average flow time $\bar{F}$ is minimized, where

$$\bar{F} = \frac{1}{|J|} \sum_{i \in J} \sum_{m \in M} F_{im}.$$

This dispatching problem can be formulated as **(P)**. Constraints (3) and (4) guarantee that once jobs i and j are assigned to same machine m, then one of the jobs should precede the other. Given the precedence information, constraint (5) dictates the precedence relation through completion and start times. In other words, a successor job can be started on the machine only if the predecessor is completed. It means that predecessor jobs' completion time is actually start time of successor jobs because we do not assume there is a setup time in this study. Constraints (6) - (12) construct the relation between start times, processing times and waiting times. Constraints (6), (8) and (10) ensure that these time variables of job i can be positive only on its assigned machine m. Constraint (7) models the deterioration of processing time through job waiting time in the assigned machine. It cannot be less than the value of function $f_{im}(\tau)$ and τ is replaced by W_{im}. Constraint (9) indicates start time of uppermost jobs on machine m must be started after completing remaining process of that machine. Constraint (11) is about waiting time of jobs in the system. A new job waits in the queue until its processing starts but an old job waiting time includes the time that it spends before new job arrivals. This concern is represented by parameter W_i' and it does not take a positive value for job $i \in J_{new}$. Besides, Constraint (12) simply calculates flow time for job i on machine m. It is equal to completion time for new jobs but W_i' is added to flow time for old jobs.

Next constraints distinguish between jobs on regular machines and jobs on the dummy machine according to the quality level. Constraints (13) make sure that job-machine assignments for the existing jobs are preserved, but assigned jobs' sequence

within the machine could be changed. Constraints (14) and (15) guarantee that the new jobs are assigned to only one machine and jobs which cannot be assigned to regular machines must be scheduled on the dummy machine. Constraint (16) ensures that quality level of a job is determined by the only assigned machine. Constraints (17) - (18) represent the quality degradation as waiting time of job increases. These constraints are adaption of function $q_{im}(\tau)$ to the model and τ is again replaced by W_{im}. Constraint (19) makes sure that the average quality on a machine obtained from all jobs in that machine exceeds the threshold quality level denoted by QLB_m. Finally, constraints (20) - (22) indicate, respectively, the integrality and non-negativity restrictions of decision variables.

$$\textbf{(P)} \quad \text{Minimize } \bar{F} \tag{1}$$

$$\text{Subject to} \tag{2}$$

$$Y_{ijm} + Y_{jim} \geq X_{im} + X_{jm} - 1, \quad \forall i, j > i \in J, \forall m \in M \tag{3}$$

$$X_{im} + X_{jm} \geq 2(Y_{ijm} + Y_{jim}), \quad \forall i, j > i \in J, \forall m \in M \tag{4}$$

$$S_{im} + P_{im} \leq S_{jm} + V(1 - Y_{ijm}), \quad \forall i, j \neq i \in J, \forall m \in M \tag{5}$$

$$P_{im} \leq VX_{im}, \quad \forall i \in J, \forall m \in M \tag{6}$$

$$P_{im} \geq \alpha_{im} + \beta_{im}W_{im} - V(1 - X_{im}), \quad \forall i \in J, \forall m \in M \tag{7}$$

$$S_{im} \leq VX_{im}, \quad \forall i \in J, \forall m \in M \tag{8}$$

$$S_{im} \geq rp_m - V(1 - X_{im}), \quad \forall i \in J, \forall m \in M \tag{9}$$

$$W_{im} \leq VX_{im}, \quad \forall i \in J, \forall m \in M \tag{10}$$

$$W_{im} \geq W'_i + S_{im} - V(1 - X_{im}), \quad \forall i \in J, \forall m \in M \tag{11}$$

$$F_{im} = P_{im} + W_{im}, \quad \forall i \in J, \forall m \in M \tag{12}$$

$$X_{im} = a_{im}, \quad \forall i \in J_{old}, \forall m \in M \tag{13}$$

$$\sum_{m \in M \backslash \{d\}} X_{im} = T_i, \quad \forall i \in J_{new} \tag{14}$$

$$X_{id} = 1 - T_i, \quad \forall i \in J_{new} \tag{15}$$

$$Q_{im} \leq VX_{im}, \quad \forall i \in J, \forall m \in M \tag{16}$$

$$Q_{im} \leq \gamma_{im} - \theta_{im}W_{im} + V(1 - Z_{im}), \quad \forall i \in J, \forall m \in M \tag{17}$$

$$Q_{im} \leq \epsilon_{im} + VZ_{im}, \quad \forall i \in J, \forall m \in M \tag{18}$$

$$\sum_i Q_{im} \geq QLB_m \sum_i X_{im}, \quad \forall m \in M \backslash \{d\} \tag{19}$$

$$X_{im}, Z_{im}, Y_{ijm} \in \{0, 1\}, \quad \forall i, j \neq i \in J, \forall m \in M \tag{20}$$

$$T_i \in \{0, 1\}, \quad \forall i \in J_{new} \tag{21}$$

$$S_{im}, P_{im}, W_{im}, F_{im}, Q_{im} \geq 0, \quad \forall i \in J, \forall m \in M \tag{22}$$

Purpose of constructing the mathematical model **(P)** is to find optimal solutions for the discussed scheduling problem. Because of binary decisions, the model is formulated as a mixed-integer linear programming (MILP). It is coded in Python and we make an experiment to solve the problem using Gurobi which is a highly preferred solver in linearly modeled problems. It first tries to find a solution to LP relaxation of the proposed MILP model. Idea of LP relaxation is evaluating binary variables as non-negative continuous variables to initiate a searching solution mechanism. This solution is kept as a best bound and then, the solver starts to discover nodes of potential real solutions as a best objective. Searching in the feasible region is stopped when gap between the best bound and the best objective is closed. Moreover, the search can continue until specified number of nodes are discovered or time limit is reached. Acceptable time consumption for solving an instance is determined as two hours in our experiment that includes several types of instance sets with different number of jobs and machines. It shows that finding an optimal solution is not possible for even small sized problems. Thus, we develop a heuristic algorithm and sorting criteria to sort the jobs in Chapter 4. Detailed computational experiment is also discussed in Chapter 5

CHAPTER IV

HEURISTIC SOLUTION APPROACHES

The simplest heuristic idea in scheduling problems is to sort items that need to be scheduled according to a criterion and process them. There is not detailed search in the feasible region in such heuristic approaches. Performance of them depends on quality of the sorting criteria and several simple criteria have already discussed for the last decades in the literature such as Shortest Processing Time (SPT), Earliest Due Date (EDD), First Come First Served (FCFS), etc. We develop two simple sorting criteria inspired by SPT and EDD which are first compared to each other and then tried to be enhanced with new sorting ideas. Two different regression models are trained to predict sequence of jobs need to be scheduled. Simple sorting approaches use few attributes of jobs (α_{im}, β_{im}, etc.) but we aim to see if more attributes can be included in a sorting criterion. Therefore, those attributes are evaluated as independent variables in the regression models. Another idea to generate a better sorting list is to combine information of existing lists. Rank aggregation method is applied finally in aggregation of lists by criteria above.

The algorithm 1 uses criteria that predefined above to schedule jobs into machines. We design it to respond quickly since it is the scheduling problem of time-dependent deterioration of jobs. Therefore, it can be solved in just seconds even in the most crowded experimental set. There are some important points considered in the algorithm which are nonidentical parallel machine scheduling and quality threshold value of each machine. Working principle of the heuristic as follows. New jobs are sorted firstly based on one of the sorting ideas and assign them one by one to machines. Sorted list of jobs are represented as $\mathbb{S}$. We decide job assignment to machines as

a contribution to the objective function. In other words, a job will be assigned to one of the machines that provide minimum average flow time while ensuring average quality of that machine. When the average quality does not equal or greater than the quality lower bound (QLB) which is represented in the constraint (19) of the mathematical model, jobs cannot be assigned that machine even if it provides lower average flow time. Jobs that cannot be assigned any machine because of QLB restriction are gathered in a list named rejected list and represented as $\mathbb{S}_r$ in the algorithm. We try to re-schedule those rejected jobs after assigning all jobs in $\mathbb{S}$. It means, average quality value in machines could be increased after placement of new jobs and jobs in the rejected set can find a place anymore. This re-scheduling process is ensured by I parameter. If there are still unassigned jobs left in $\mathbb{S}_r$ after the process, they will be assigned to the dummy machine with the order of main sorting criterion. Moreover, decision of job sequence in each machine is a single machine scheduling problem (SMSP). It might be an idea to use the mathematical model that finds optimal order of jobs. However, it is not useful in our case because trying to find an optimal solution in each SMSP increases time consumption sharply and there is no solution in reasonable time when average job per machine is more than 10. Optimal SMSP solutions also do not provide better global solutions always. In conclusion, sorting criterion for SMSP is same as initial sorting criterion.

Algorithm 1 Sorting Criterion Based Scheduling Algorithm

Input: J_{new}, J_{old}, M.

Output: Final schedule consisting of all jobs J and ordered set of rejected jobs.

Step 1. Sort the jobs in J_{new} according to a criterion. Let $\mathbb{S}$ be that sequence of new jobs. Let I be 1 if re-scheduling process is done, 0 otherwise. Set $I = 0$

Step 2. Let $\mathbb{S}_r$ be the ordered set of rejected jobs and set $\mathbb{S}_r = \emptyset$. Go to Step 5.

Step 3. If $I = 1$, assign $\mathbb{S}_r$ to d and STOP.

Step 4. If $\mathbb{S}_r \neq \emptyset$, set $\mathbb{S} = \mathbb{S}_r$, $I = 1$ and $\mathbb{S}_r = \emptyset$. Otherwise, STOP.

Step 5. Let job i be the first job in $\mathbb{S}$.

Step 6. Sort the jobs in each $m \in M \backslash \{d\}$ with job i according to a criterion, and let $\zeta_{i,m}$ be its solution.

Step 7. Sort the solutions $\zeta_{i,m}$ for all $m \in M \backslash \{d\}$ in non-decreasing order of average flow time.

Step 8. Let $\zeta_{i,m}[l]$ be the l^{th} best solution and set $l = 1$.

Step 9. If $\zeta_{i,m}[l]$ is infeasible according to the constraint 19.

Step 9.1 If $l = |M - 1|$, remove the job $\mathbb{S}$ and add it to the end of $\mathbb{S}_r$. Go to Step 11.

Step 9.2 Otherwise set $l = l + 1$ and go to Step 9.

Step 10. Apply the solution $\zeta_{i,m}[l]$ and remove job i from $\mathbb{S}$.

Step 11. If $\mathbb{S} = \emptyset$, go to Step 3. Otherwise, go to Step 5.

The solution quality of the heuristic depends on the initial sequencing of new jobs that need to be scheduled. We propose several sorting approaches for new jobs in the next sections.

4.1 Simple Sorting Approach

First sorting criterion is motivated by the well-known SPT, Shortest Processing Time first, rule which minimizes average flow time for a single machine problem. The idea is to schedule the jobs with lower processing times earlier than others. In the case with deteriorating jobs, the job's processing time as well as its deterioration rate determines its sequence. It makes sense to schedule the jobs with lower processing times and higher deterioration rates earlier, so that the overall average flow time can be decreased. Therefore, we propose to sort the new jobs in non-decreasing order of the ratio of mean basic processing time. Hence, we define the processing priority ratio of job i as R_i^p, where

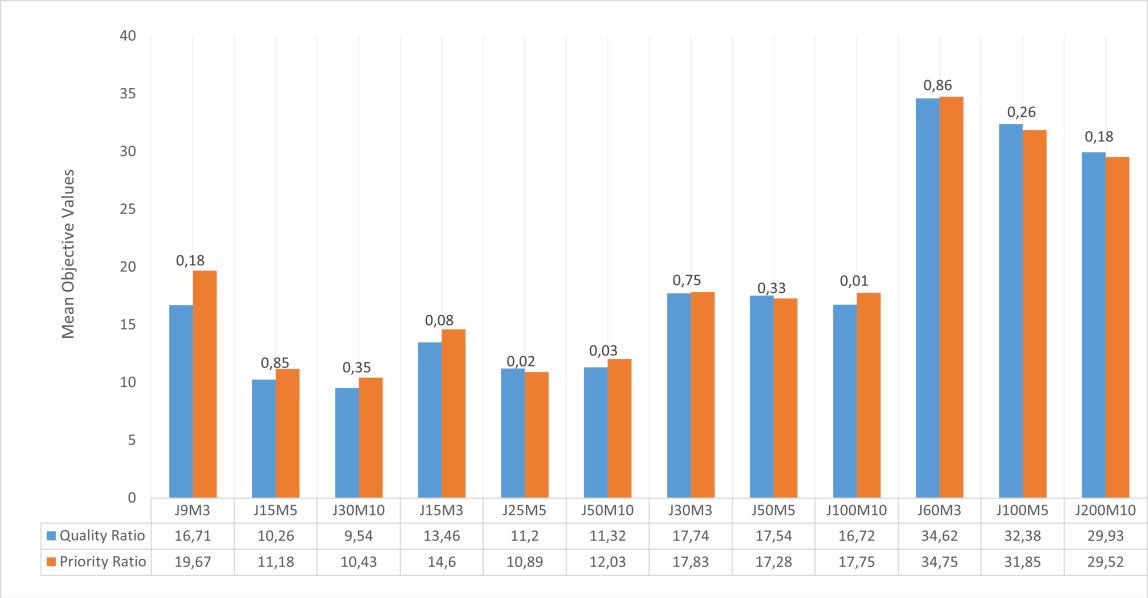
$$R_i^p = \frac{\sum_{m \in M} \alpha_{im}}{\sum_{m \in M} \beta_{im}}.$$

In the second place, the goal of the scheduling problem under investigation is to minimize average flow time while ensuring the overall quality level exceeds a predefined threshold. This implies that time sensitive jobs should be scheduled earlier than the jobs with lower quality degradation rates. EDD rule inspires this sorting ratio because if jobs cannot be processed in an acceptable time limit, they will be rejected. Therefore, we propose to sort the jobs in non-decreasing order of base quality level to quality degredation ratios, R_i^q, where

$$R_i^q = \frac{\sum_{m \in M} \gamma_{im}}{\sum_{m \in M} \theta_{im}}.$$

There are now two simple sorting ideas which are shortly called as priority ratio and quality ratio to be a criterion in the heuristic algorithm. Design of experimental sets and results are shown in Chapter 5 in detail but we partly discuss results here. Figure 2 shows how simple sorting-based criteria works over 12 sets which consists of different number of jobs and machines. For example, first set is named as J9M3 because there are 9 jobs need to be scheduled on 3 machines. Y-axis indicates average objective values of 30 instances in each set. Probability values on the graph are pairwise test scores (Wilcoxon and t-test) to see if mean objectives are significantly different from each other. It is hard to say that one of them completely outperforms the other one. Simple criteria' performance compared to new approaches are discussed later in the computational results.

Figure 2: Simple Criteria Results for Various Number of Jobs and Machines. P-values on Bars Show Pairwise Test Significance Scores



	J9M3	J15M5	J30M10	J15M3	J25M5	J50M10	J30M3	J50M5	J100M10	J60M3	J100M5	J200M10
Quality Ratio	16,71	10,26	9,54	13,46	11,2	11,32	17,74	17,54	16,72	34,62	32,38	29,93
Priority Ratio	19,67	11,18	10,43	14,6	10,89	12,03	17,83	17,28	17,75	34,75	31,85	29,52

4.2 Regression Based Sorting Approach

It is known that the scheduling problems are generally NP-hard including the one under investigation. We cannot solve real size problem instances using the model **(P)** as shown in Chapter 5. However, we can utilize the solutions of the small sized problems to gain insights about the effect of job properties, i.e., α_{im}, β_{im}, γ_{im}, θ_{im}, and ϵ_{im}, on the optimal solutions. Therefore, we generate random instances that can be solved optimally using the mathematical model of SMSP. We define **(P1**$[J']$**)** for single machine scheduling problem with time-dependent deterioration of jobs. It is the single machine version of **(P)** where the quality threshold constraint is modified. The algorithm 1 requires a criterion to initiate the process and it can be generated by predicting sequence of jobs. We train regression models to use them as a criterion. These models are trained using optimal solutions of **(P1**$[J']$**)**. Its working principle is totally same as the main model but only difference is that there is not a dummy machine. Rejected jobs will be scheduled after regular jobs due to constraints (27)

& (28). Thus, we define a non-negative continuous variable, C_{max}, as a threshold value that distinguishes between accepted and rejected jobs. Other constraints are basically single machine version of the main model. Finally, SMSP is solved with job numbers 7,8,9 and 10. Instances with each job number are randomly generated 100 times.

$$\textbf{(P1}[J']\textbf{)} \quad \text{Minimize } \bar{F} \tag{23}$$

$$\text{Subject to} \tag{24}$$

$$Y_{ij} + Y_{ji} = 1, \quad \forall i, j > i \in J' \tag{25}$$

$$S_i + P_i \leq S_j + V(1 - Y_{ij}), \quad \forall i, j \neq i \in J' \tag{26}$$

$$S_i + P_i - V(1 - T_i) \leq C_{max}, \quad \forall i \in J' \tag{27}$$

$$C_{max} \leq S_i + VT_i, \quad \forall i \in J' \tag{28}$$

$$P_i \geq V(1 - T_i), \quad \forall i \in J' \tag{29}$$

$$P_i \geq \alpha_i + \beta_i W_i, \quad \forall i \in J' \tag{30}$$

$$S_i \geq rp, \quad \forall i \in J' \tag{31}$$

$$W_i \geq W_i' + S_i, \quad \forall i \in J' \tag{32}$$

$$F_i = P_i + W_i, \quad \forall i \in J' \tag{33}$$

$$Q_i \leq \gamma_i - \theta_i W_i + V(1 - Z_i), \quad \forall i \in J' \tag{34}$$

$$Q_i \leq \epsilon_i + VZ_i, \quad \forall i \in J' \tag{35}$$

$$Q_i \leq VT_i, \quad \forall i \in J' \tag{36}$$

$$\sum_{i \in J'} Q_i \geq QLB \sum_{i \in J'} T_i \tag{37}$$

$$Z_i, Y_{ij}, T_i \in \{0, 1\}, \quad \forall i, j \neq i \in J' \tag{38}$$

$$S_i, P_i, W_i, F_i, Q_i, C_{max} \geq 0, \quad \forall i \in J' \tag{39}$$

We construct two different regression models: *Multiple Linear Regression* and *Poisson Regression* that take α_i, β_i, γ_i, θ_i, and ϵ_i values as independent variables. We also generate meta columns of the job properties as independent variables that include other jobs' mean (μ) and standard deviation (σ) in a sample excluding the one which is taken into consideration. Number of jobs in a list is considered as another independent variable. So, all 16 independent variables in the training model are as follow: α_i, β_i, γ_i, θ_i, ϵ_i, $\mu(\alpha_i)$, $\mu(\beta_i)$, $\mu(\gamma_i)$, $\mu(\theta_i)$, $\mu(\epsilon_i)$, $\sigma(\alpha_i)$, $\sigma(\beta_i)$, $\sigma(\gamma_i)$, $\sigma(\theta_i)$, $\sigma(\epsilon_i)$ and number of jobs. Variables are continuous entries except number of jobs column which consists of positive integer numbers. The regression models yield the *potential* start time for linear regression and *rank* of jobs for poisson regression as response variables. We calculate average value of parameters as to machines in prediction and plug them into the models. For example, α_i value of a job in prediction will be $(\sum_{m \in M} \alpha_{im})/M$. Final sequence of jobs is determined by sorting them according to non-decreasing order of the response values.

Regression models are trained in programming language R. Table 3 shows the final linear regression model which tries to predict start time of jobs. All variables are initially put into the model and they are removed according to backward elimination which is a highly used method in machine learning to have simpler and better models. Insignificant parameters which have the highest p-values are deducted from the model step by step until only significant variables left. F-test score indicates resulting model is significant as well. When we look at the table below, it is seen that the model considers intercept values of linear functions $f_{im}(\tau)$ and $q_{im}(\tau)$, i.e., α_i and γ_i, as significant values. Increase in α_i value of jobs causes being in the back row but increase in γ_i value of jobs causes being in the front row. Besides, higher quality degradation rate, θ_i, affects start times downward. The model indicates that meta columns of γ_i and θ_i affect the response variable in the same way as main properties affect. Prediction in the linear regression analysis is mostly dependent on quality

parameters of jobs.

Table 3: Linear Regression Model

Coefficients	Intercept	α_i	γ_i	θ_i	$\mu(\gamma_i)$	$\mu(\theta_i)$	$\sigma(\gamma_i)$	# of Jobs
Estimate	194,38	6,73	-28,38	104,51	-22,24	119,61	-36,35	3,22
Std. Error	9,89	0,72	0,72	15,86	1,96	42,49	3,81	0,38

Poisson regression is preferable in count data sets since it uses maximum likelihood estimation method fundamentally. Predicted value of this model is assumed to be position of jobs in scheduling sequence. We try to predict potential start times of jobs in linear regression but there is nothing to do with predicted times directly. The purpose is to sort them in non-decreasing order to decide rank of jobs. So that performing poisson regression and linear regression do not have difference in use of purpose. Table 4 shows final poisson regression model. Output of the model is logarithmic value of job ranks because of the link function of poisson regression. Count data models called as generalized linear models and it is provided by log function in this case. Significant variables are determined by backward elimination method as in the linear regression model. We see again that α_i and γ_i values are significant and quality threshold value of the jobs has influence on assigning jobs to first places in a scheduling list. There is a symmetry in the model compared to table 3. It means that meta columns affect a job position in contrast with α_i and γ_i. Moreover, model in the poisson regression analysis uses information of processing and quality values in half shares and degradation rates (β_i and θ_i) are not considered as significant variables as well as ϵ_i is not significant in both models.

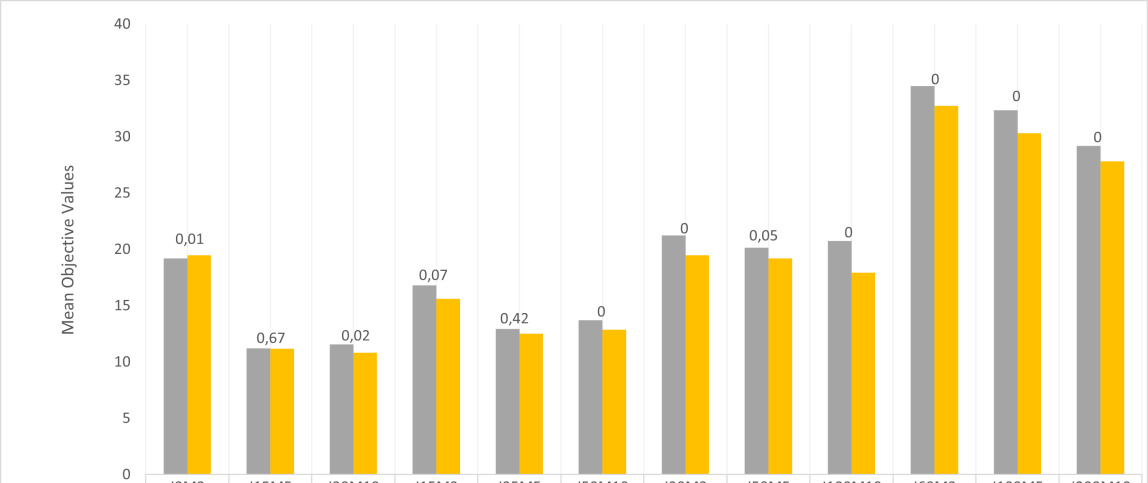
Table 4: Poisson Regression Model

Coefficients	Intercept	α_i	γ_i	$\mu(\alpha_i)$	$\mu(\gamma_i)$	# of Jobs
Estimate	0,62	0,32	-0,47	-0,32	0,46	0,11
Std. Error	0,19	0,01	0,01	0,04	0,04	0,008

As a next step, we examine and compare performance of both models. Common idea is looking at some metrics to evaluate performance of predictions such as

R-squared, mean of squared errors (MSE) or absolute deviations (MAD). We see R-squared value of the linear model as 0.36. It means the model is successful to explain 36% of the data. On the side of poisson regression, 44% of the data can be explained by the model. This percentage is calculated by 1 - (residual deviance / null deviance). Both models' performance does not seem satisfactory enough based on these values. However, we do not aim to use these response values directly into the heuristic algorithm. Since jobs are sorted according to regressed variables, success in order of them is the main concern in this study. If the models are succeeded to predict a job position in scheduling lists, then we can make a performance measurement. Therefore, the heuristic algorithm is run with criteria provided by regression models. Figure 3 shows results with the same setting in comparison of simple sorting criteria. Mean objective values of 30 instances in each set is considered as a metric to make comparison. Paired test scores are demonstrated on the graph to see if the mean differences are significant. Linear model provides a significantly better mean objective in the first set. Poisson model outperforms the other model in the rest of sets but there are some insignificant scores when average job number per machine is less than or equal to 5. We can conclude that the poisson model tends to provide significantly better results in increased average number of jobs per machine.

Figure 3: Regression-Based Criteria Results for Various Number of Jobs and Machines. P-values on Bars Show Pairwise Test Significance Scores



4.3 Rank Aggregation Based Sorting Approach

Rank aggregation (RA) is a method that looks for efficient ways of combining input lists. There are many different approaches for the rank aggregation problem and they dissociates according to statistical differences of aggregating lists. We investigate and try to implement RA methods that should respond quickly due to nature of the problem in this study. Optimization-based methods are first focused on. However, these methods are not in accordance with objective of our problem because they try to minimize a distance metric such as Kendall's tau or Spearman distances while deciding rank of items. Minimized distance metrics do not necessarily provide improvements in the objective function. Another concern in solving RA problem optimally is the time consumption whether all permutations of ranks is obtained or heuristic approaches as CEMC method.

Secondly, non-optimization based methods are considered such as Borda's Count and Markov Chain (MC). These methods' effectiveness does not rely on any distance

metric. List of items, i.e., jobs, that need to be aggregated are evaluated as scores provided by the methods and success of them can be measured according to output of the algorithm 1. We calculate these scores as stationary probabilities of MC based methods. There are several ways of establishing Markov's chains in the literature which are mostly preferable when number of base ranker lists is high. However, we aggregate only 4 lists, simple sorting and regression-based lists, in this study. Borda's count is performed based on arithmetic average, median, geometric average and L2 norm values of full ranked lists. Borda's method is also simpler and faster to implement.

In this study, we use full Borda aggregation that includes all 4 basic metrics. We simply solve an instance using Borda metrics and take the best one as a solution. Since this operation does not require a complex search and it is a very basic heuristic solution, there is no time complexity. We can get the best solution of four in few seconds for the most crowded set. Table 5 is a used example of Borda aggregation in our experiments. It is an instance for the first set which includes 9 jobs and 3 machines. Job 1 is already assigned to one of the machines as a old job and the remaining are evaluated as new jobs. Firstly, we have four different sequences of new jobs to be processed in the heuristic algorithm. They are base ranker generated by quality ratio (Q), priority ratio (P), linear regression (R1) and poisson regression (R2). *Base Rank* column shows each job position in four main lists and next columns indicate how they can be aggregated based on Borda's metrics. For example, job 8's position is third in Q, second in P, third in R1 and third in R2. Therefore, mean score of these positions is 2.75, median score is 3, geometric average score is 2.71 and l2 score is 2.78. Job 8's aggregated rank is determined by non-decreasing sorted scores in different lists and it is mostly put in top of the scheduling lists. There is also an attentive point in the aggregation method that is when scores of two or more jobs are equal to each other as jobs 7 and 9 in *Mean* column, their order is identified

according to first base ranker. Quality ratio (Q) places job 9 before job 7, then job 9 will be fourth and job 7 will be fifth in the mean aggregation because Q is the first base ranker in this example.

Table 5: Borda's Rank Aggregation Example. Position of Jobs in Main Sorting Lists. Scores of Job Positions Using Borda's Metrics and Relevant Ranks.

	Base Rank				**Mean**		**Median**		**Geo.Mean**		**L2**	
Job	Q	P	R1	R2	Score	Rank	Score	Rank	Score	Rank	Score	Rank
2	2	8	4	6	5	6	5	5	4,43	6	5,48	6
3	7	7	1	1	4	3	4	3	2,65	1	5,00	5
4	6	4	2	2	3,5	2	3	1	3,13	3	3,87	2
5	5	5	8	8	6,5	8	6,5	7	6,32	8	6,67	8
6	8	1	7	7	5,75	7	7	8	4,45	7	6,38	7
7	4	3	5	5	4,25	4	4,5	4	4,16	5	4,33	3
8	3	2	3	3	2,75	1	3	1	2,71	2	2,78	1
9	1	6	6	4	4,25	4	5	5	3,46	4	4,72	4

This aggregation example is the combination of four main lists represented as QPR1R2 in the next chapter. Other combinations can be made as subsets of a set with four elements excluding the empty set and one element sets. Hence, we qualify eleven combinations of the base ranker lists.

CHAPTER V

COMPUTATIONAL RESULTS

Firstly, proposed mathematical model **(P)** is tried to solve at optimality under sets of instances consisting of various number of jobs and machines. Each set includes 30 different instances for same number of jobs and machines with uniformly regenerated random parameters. Quality lower bounds (QLB) of each set are arranged to schedule on average 10% of total jobs to the dummy machine. Lower and upper bounds of parameters and QLB values are showed in Appendix A. In addition, 10% of jobs are assumed to be old jobs that are already assigned on a regular machine and M size in the tables below shows number of regular machines without adding the dummy machine.

Hence, 12 sets of 30 instances make 360 solution trials totally. Distribution of jobs and machines over sets can be seen in the table 6. In this design, we aim to examine tendency of results when J size increase for fixed M size and demonstrate how solutions are responding on J/M rate that enhances over sets as well. The experiment proves that complexity of the problem increases through main diagonal of JxM matrix in the table. It means optimal solutions in the first set can be found easily but it is not valid for other sets and performance of the solver gets worse from first set to the last.

Table 6: Experimental Sets with Various Number of Jobs J and Machines M.

Set	**M**	3	5	10	**J/M**
1-3	**J**	9	15	30	3
4-6		15	25	50	5
7-9		30	50	100	10
10-12		60	100	200	20

The experiment includes different job sizes which vary from 9 to 200 and their corresponding machine sizes vary from 3 to 10. We can find optimal solutions for the set with $J = 9$, $M = 3$ and $QLB = 4$ in seconds but it is not possible for bigger sizes. Therefore, 3 instances from the remaining sets are randomly chosen and tried to solve in GUROBI solver. Running time for each instance is 2 hours because time dependent deteriorating jobs are considered in this study. Table 7 indicates GUROBI performance. Gap column shows percentage difference between the best bound and the best objective values when time limit is reached. The gap is doubled when number of job is also doubled for constant number of machines. The set with $J = 15$ and $M = 5$ seems less complex than others since we see an optimal solution with tolerance 0.01 in run 10. When the system becomes more crowded, we cannot find any objective as in runs 19 and 20. Accordingly, the sets with $J = 50$,100,200 and $M = 10$ do not include a gap in the table.

Table 7: Gurobi Results. Proportional Gap Between the Best Bound and the Best Objective Over Experimental Sets.

Run	J	M	Gap %	Run	J	M	Gap %	Run	J	M	Gap %
-	9	3	Optimally Solved	10	15	5	0.01	22	30	10	28
				11	15	5	7	23	30	10	23
				12	15	5	12	24	30	10	24
1	15	3	25	13	25	5	42	25	50	10	-
2	15	3	21	14	25	5	37	26	50	10	-
3	15	3	22	15	25	5	38	27	50	10	-
4	30	3	50	16	50	5	74	28	100	10	-
5	30	3	62	17	50	5	76	29	100	10	-
6	30	3	52	18	50	5	77	30	100	10	-
7	60	3	94	19	100	5	-	31	200	10	-
8	60	3	90	20	100	5	-	32	200	10	-
9	60	3	83	21	100	5	96	33	200	10	-

As a result, the problem cannot be solved at optimality for bigger sizes. We next perform the heuristic algorithm for each instance under sorting criteria we discussed in the Chapter 4. We first examine results of simple sorting criteria versus results of regression-based sorting criteria and then rank aggregation-based criteria is involved

to the comparison.

5.1 Comparison Between Simple Lists & Regression Based Lists

Table 8 shows performance of the best simple list and the best regression model. If average objective values in each set is better than the other average, it is shown as bold number. P-values which are under the worse objective value are pairwise test scores to see if there is a statistical difference between mean objective values. The table is divided into 4 main lines and each of the lines includes 3 sets. Common point of each line is that they have same average number of jobs per machine. We see that the regression model which is poisson regression gives better results only in fourth line which includes the sets where average number of jobs per machine is 20. In conclusion, poisson regression-based sorting list outperforms other three main sorting lists in the most crowded instances and it is useful when job number per machine is high.

Table 8: Simple Lists vs Regression Based Lists. Bold Numbers for Minimum Mean Objective in a Set. Pairwise Significance Test Results Shown Under Worse Mean Objectives.

3 Machines			5 Machines			10 Machines		
Jobs	Simple	Regression	Jobs	Simple	Regression	Jobs	Simple	Regression
9	**16,71** -	19,19 0,15	15	**10,26** -	11,17 0,24	30	**9,54** -	10,83 0,08
15	**13,46** -	15,6 0,05	25	**10,89** -	12,51 0,00	50	**11,32** -	12,85 0,00
30	**17,74** -	19,49 0,01	50	**17,28** -	19,21 0,00	100	**16,72** -	17,92 0,00
60	34,62 0,00	**32,73** -	100	31,85 0,00	**30,31** -	200	29,52 0,00	**27,83** -

5.2 Statistical Analysis and Comparison For Aggregated Lists

Table 9 shows how Borda aggregation improves the mean objective values in percentage compared to both simple and regression-based sorting lists. In the table, we

show only four different combinations of main lists which are the most significant aggregations compared to other possible combinations. Q is for the quality-based list, P is for the priority-based list, R1 is for the linear regression and R2 is for the poisson regression models. So, QPR1R2 is the aggregation of all 4 main lists in the table. We already know that simple lists significantly provide lower mean objective values than regression lists in the first 9 sets only. QPR1 and QR2 improve the mean objective value more than 10% for the first set but QPR2 outperforms other aggregation-based sorting lists in the remaining sets compared to simple lists. When comparison is for the regression lists, there are slight improvements in the last set. It is moreover seen that QPR1 makes the mean objective value worse in the last set. Hence, idea of aggregating all lists together does not necessarily provide better results since performance of the aggregation does not only depend on individual scores of the base lists but also depends on how they are coordinated to each other. Aggregating only simple sorting lists is not efficient enough to improve objective functions over all sets. Including regression-based sorting lists into the aggregation is essential in our case. The most successful aggregation with two sorting lists is combination of quality ratio, Q, and poisson regression-based list, R2, named as QR2. Quality ratio list is included other combinations as well in the table 9. It indicates that rank aggregation without knowledge of quality properties of the jobs is not effective.

Table 9: Percentage Improvements in Mean Objectives by Aggregation Lists Compared to Simple and Regression Lists Over Sets.

Compared to Simple Lists

Lists	J9M3	J15M5	J30M10	J15M3	J25M5	J50M10	J30M3	J50M5	J100M10	J60M3	J100M5	J200M10
QPR1R2	2,69	5,46	3,77	2,97	3,03	3,89	3,89	2,95	2,45	9,21	7,63	6,2
QPR1	11,19	6,43	6,81	6,17	3,03	4,77	3,95	3,82	3,11	7,83	6,09	4,74
QPR2	5,86	8,38	7,97	7,43	4,96	5,92	6,43	4,69	4,25	9,99	7,69	6,06
QR2	13,23	4,87	5,97	8,1	3,12	3,89	4,57	2,78	2,87	8,23	6,12	5,93

Compared to Regression Lists

Lists	J9M3	J15M5	J30M10	J15M3	J25M5	J50M10	J30M3	J50M5	J100M10	J60M3	J100M5	J200M10
QPR1R2	15,27	13,16	15,24	16,28	15,59	15,33	12,52	12,7	8,98	3,97	2,94	0,5
QPR1	22,67	14,06	17,91	19,04	15,59	16,11	12,57	13,48	9,6	2,51	1,32	-1,04
QPR2	18,03	15,85	18,93	20,13	17,27	17,12	14,83	14,26	10,66	4,8	3	0,36
QR2	24,44	12,62	17,17	20,71	15,67	15,33	13,13	12,55	9,38	2,93	1,35	0,22

Shapiro test scores for lists of objective values are shown in Appendix A. We can say that the lists are distributed normally with significance score of 0,05 when average number of jobs per machine starts to increase in general. There are some lists distributed completely normal and some distributed non-normal in the last sets. So, it is hard to mention a pure pattern about distribution of objective values in the sets. Table 10 shows the pairwise statistical difference test scores between aggregation based and main sorting lists. We have two particular tests which are wilcoxon test and ttest. Wilcoxon is more accurate when two lists are distributed non-normally and ttest is for normal distributed lists. Comparing only mean objective values may not be informative enough as we do in the table above since we cannot recognize if the gap between two sorting lists is because of few differences. In order to see if there is valuable complete difference, we can use these tests which basically evaluate objective values in pairwise comparison with significance score of 0,05. Shapiro test scores show that the lists start to distribute mostly normal after the 7th set which is J30M3. So, it is better to follow the ttest scores after the 7th set to compare the lists in the table 10. Noted rank aggregation-based sorting lists make important improvements except in the last set compared to the poisson regression based list. Table 8 shows that

poisson regression performs approximately 6% better than quality ratio-based list in the set J200M10.

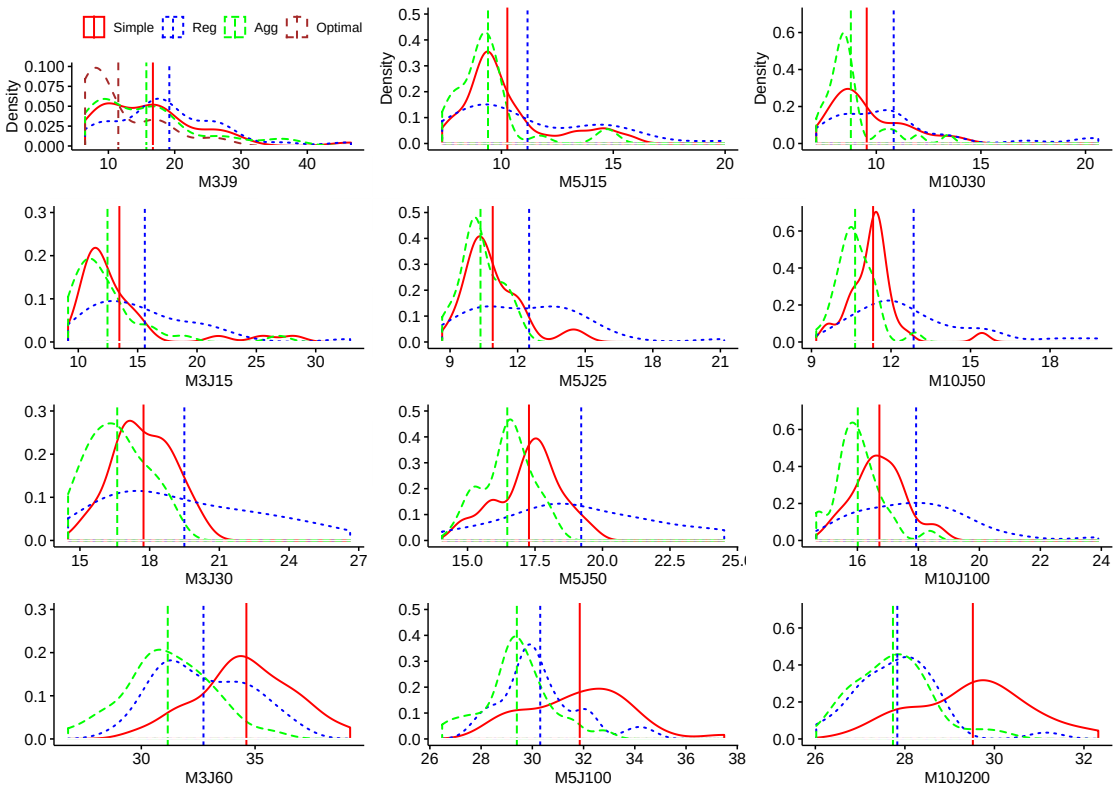
Table 10: Wilcoxon Test and T-Test Scores Indicating Statistical Difference of Aggregation Lists Compared to Simple and Regression Lists.

Compared to Simple Lists												
Lists	J9M3	J15M5	J30M10	J15M3	J25M5	J50M10	J30M3	J50M5	J100M10	J60M3	J100M5	J200M10
QPR1R2_wilcoxon	0,1	0	0,05	0,06	0	0	0	0	0	0	0	0
QPR1R2_ttest	0,83	0,3	0,4	0,7	0,27	0,11	0,06	0,08	0,09	0	0	0
QPR1_wilcoxon	0,01	0	0	0,02	0	0	0	0	0	0	0	0
QPR1_ttest	0,35	0,21	0,11	0,4	0,28	0,03	0,08	0,02	0,01	0	0	0
QPR2_wilcoxon	0,05	0	0	0	0	0	0	0	0	0	0	0
QPR2_ttest	0,64	0,08	0,06	0,33	0,06	0	0	0,01	0	0	0	0
QR2_wilcoxon	0	0	0,01	0,01	0	0	0	0,03	0	0	0	0
QR2_ttest	0,27	0,36	0,21	0,24	0,24	0,08	0,02	0,1	0,03	0	0	0
Compared to Regression Lists												
Lists	J9M3	J15M5	J30M10	J15M3	J25M5	J50M10	J30M3	J50M5	J100M10	J60M3	J100M5	J200M10
QPR1R2_wilcoxon	0	0	0	0	0	0	0	0	0	0	0	0,29
QPR1R2_ttest	0,17	0,04	0,02	0,03	0	0	0	0	0	0,01	0,02	0,57
QPR1_wilcoxon	0	0	0	0	0	0	0	0	0	0	0,06	0,03
QPR1_ttest	0,04	0,02	0	0,01	0	0	0	0	0	0,1	0,29	0,23
QPR2_wilcoxon	0	0	0	0	0	0	0	0	0	0	0	0,53
QPR2_ttest	0,11	0,01	0	0,01	0	0	0	0	0	0	0,02	0,68
QR2_wilcoxon	0	0	0	0	0	0	0	0	0	0	0,09	0,51
QR2_ttest	0,02	0,05	0,01	0	0	0	0	0	0	0,05	0,26	0,79

There are 3 major sorting categories considered thus far: simple sorting-based, regression-based and rank aggregation-based. Figure 4 summarizes the results in this study. Simple category reduced to one criterion, quality ratio or priority ratio, according to the best in the related set. Regression (Reg) category is established likewise although linear regression outperforms poisson regression in the first set only. Finally, it is discussed which aggregation list is more satisfactory in the paragraph above and QPR2 is more consistent than others, so that it is chosen as representative of the aggregation (Agg) category. The figure shows density of objective values in each set and vertical lines shows mean objective values. Density and mean of optimal solutions is also shown in the first set. Regression-based approach is not effective in the

instances where average job per machine, J/M, is lower than 20. When it reaches 20, poisson regression-based sorting approach becomes significantly successful. However, we achieve improving solution quality of simple sorting ideas by aggregating ranks in each set.

Figure 4: Density of Objective Values and Means as Vertical Lines for the Best Criterion of Simple,Regression and Aggregation Based Ideas in the Sets.



Combination of quality ratio, priority ratio and poisson regression, QPR2, makes significant gaps between simple-based and regression-based sorting lists but it is hardly significant for the first set although 10% improvement is possible in the other aggregation lists. Besides, difference between mean objectives by aggregation and regression lists for the last set is very close to each other. Until now, we focus on single aggregation manner that is need to be necessarily valid for the all sets. We see that improving performance of the aggregation for particular sets is also possible in the next section.

In the beginning of this chapter, solving the problem at optimality is discussed and gap between the best objective and the best bound by the solver is shown. Now, gap between the best objective values by Gurobi until time limit is reached and the objective value by the heuristic algorithm with criterion QPR2 for each run is seen in the table 11. The *Gap %* column shows percentage improvement in the objective function obtained by the algorithm 1 which starts to outperform exact solutions by the solver when number of machines in the system increases as well as J/M rate increases. Even finding a solution after run 25 is not possible for Gurobi. Although the solver generates better solutions in the uncrowded sets, these are output of 2 hours long performance. However, the heuristic solutions are instantaneously generated.

Table 11: Proportional Gap Between Objective Values by 2-Hours Running Gurobi and the Heuristic Algorithm with Criterion QPR2. Minus Gap When Gurobi Outperforms. No Gap When No Solution By Gurobi.

Run	J	M	Gap %	Run	J	M	Gap %	Run	J	M	Gap %
-	9	3	Optimally Solved	10	15	5	-76	22	30	10	-16
				11	15	5	-18	23	30	10	-14
				12	15	5	-10	24	30	10	-10
1	15	3	-10	13	25	5	-22	25	50	10	-
2	15	3	-26	14	25	5	-32	26	50	10	-
3	15	3	-19	15	25	5	-30	27	50	10	-
4	30	3	-14	16	50	5	32	28	100	10	-
5	30	3	-17	17	50	5	43	29	100	10	-
6	30	3	-21	18	50	5	32	30	100	10	-
7	60	3	181	19	100	5	-	31	200	10	-
8	60	3	87	20	100	5	-	32	200	10	-
9	60	3	27	21	100	5	401	33	200	10	-

5.3 Advancing Aggregation Lists

The computational experiment indicates that solutions by the sorting lists tend to behave independently in each set and they are sensitive to number of jobs and machines. For example, there are only 9 jobs to assign to 3 machines in the first set. We also assume that range of rp_{dummy} parameter is stable in each set as seen in the appendix A. It causes that when a job must be assigned to dummy machine, it costs

the objective function more in the first set because of less number of jobs. Although it does not ensure providing better objective function, sorting jobs based on quality ratio decreases the possibility of assigning jobs to the dummy machine. We see this situation in the table 8 which state that quality lists performs better 10% averagely than other main lists in the first 4 sets. Besides, the poisson regression model is trained by the data consists of job sizes 7,8,9 and 10 for SMSP outperforms main lists 5% averagely in the last three sets. We conclude that when average number of jobs per machine is low, quality ratio outperforms but high average number of jobs per machine causes poisson regression performs better. The heuristic algorithm sorts jobs using same sorting criterion in steps 1 and 6. Therefore, idea of advancing performance of heuristic solutions is to use quality ratio and poisson regression-based sorting criteria in step 6 which is also a SMSP. We continue to sort jobs based on the aggregation in step 1. Table 12 shows how QPR2 is advanced when quality ratio and poisson regression used as SMSP criterion compared to SMSP with rank aggregation in the table 9. In the first set, percentage improvement increases from 5,86% to 13,64% in comparison to simple lists. Scores get worse in the remaining sets except 4th set but they are already significantly well. Wilcoxon test score decreases from 0,05 to 0,01, so it gets significance when we use quality ratio in SMSP for the first set. When comparison is for the regression lists percentage improvement increases for last three sets and we now gain significance with p-value of 0 in the last set. As a result, modifying the heuristic algorithm according to individual success of main sorting lists in the sets makes sense to enhance the performance.

Table 12: Advancing QPR2 Performance by Changing SMSP Sorting Criterion from Aggregation to Quality Ratio and Poisson Regression.

Compared to Simple Lists												
SMSP Criterion	J9M3	J15M5	J30M10	J15M3	J25M5	J50M10	J30M3	J50M5	J100M10	J60M3	J100M5	J200M10
Aggregation	5,86	8,38	7,97	7,43	4,96	5,92	6,43	4,69	4,25	9,99	7,69	6,06
Quality	13,64	8,28	3,77	7,5	2,94	2,39	2,48	2,08	1,44	3,52	2,39	1,42
Compared to Regression Lists												
SMSP Criterion	J9M3	J15M5	J30M10	J15M3	J25M5	J50M10	J30M3	J50M5	J100M10	J60M3	J100M5	J200M10
Aggregation	18,03	15,85	18,93	20,13	17,27	17,12	14,83	14,26	10,66	4,8	3	0,36
Regression	12,87	11,82	15,14	16,09	17,43	15,72	14,37	14,73	10,32	5,07	4,32	2,12

As a result, we aim to outperform well-known basic sorting ideas for the studied scheduling problem by training regression-based models and applying rank aggregation concept that inspired from Borda's count. Linear and poisson regression-based sorting approaches are limited to achieve the purpose in the whole experiment. However, rank aggregation is completely efficient in outperforming basic sorting ideas, called as priority and quality ratios. It is essential to add at least one of regression models into the aggregation and the poisson model has more influence than the linear model. It is besides shown that aggregation performance can be enhanced in the particular sets by using SMSP sorting criterion as quality ratio and poisson regression.

CHAPTER VI

CONCLUSION

In this thesis study, we propose novel sorting approaches for a nonidentical parallel machines scheduling problem which has some difficult points that make the problem cannot be solved at a reasonable time for crowded scheduling environments. For example, processing times and quality value of jobs are time-dependent decision variables. The system can include old jobs that are already in-process as well as they can be waiting jobs in queues of machines before starting dispatching new jobs.

The problem is formulated as a mixed integer linear programming and tried to solve at optimality under various experimental sets. We see that it is not possible to solve the problem exactly when size of the instances, number of jobs and number of machines in the system, starts to grow. Instances with 9 jobs and 3 machines which are the first set in the experiment are solved optimally in just seconds but it is not valid for other bigger sizes even if they are tried to solve until time limit, 2 hours, is reached. Moreover, we cannot find any potential objective value after number of jobs is higher than 100 or number of machines is higher than 10.

We then develop a heuristic algorithm which is fast and dependent on a sorting criterion. It tries to schedule jobs from the sorted list one by one on machines in a way to lower the objective function that is average flow time of all jobs in the system. Hence, solution quality of the algorithm is directly based on initial sorting criterion. We implement two basic sorting criteria by properties of the jobs named as priority and quality ratios. These criteria come from nature of the problem setting and there is not extra work here. We aim to develop new criteria as fast as basic sorting ideas while outperforming them. First is training regression models: multiple

linear regression and poisson regression to make prediction of average start times of jobs on machines and order of jobs to process on machines respectively. Rank of jobs is considered as count data to perform poisson regression analysis and start times of jobs are already continuous variables that are suitable for linear regression analysis. Results of simple sorting lists and regression-based sorting lists show that poisson regression generates better objective values than quality and priority ratios in the experimental sets where average number of jobs per machine is 20. It means that the poisson model can be preferable as a criterion in the crowded sets. However, these new sorting ideas are not fully efficient to outperform simple sorting criteria when number of jobs and machines varies.

Secondly, we consider another idea to generate better sorting lists that is rank aggregation. Borda's count metrics are used to calculate aggregation scores to decide rank of jobs. Four sorting lists above can be aggregated as base ranker. Hence, different combinations of base ranker lists are performed in the experiment. We see that the quality ratio is necessarily added in all important aggregated sorting lists. Adding at least one of the regression models in aggregation definitely improves the performance and poisson regression has more effects than the linear model in aggregation. Individual performance of base ranker lists has parallel influence on aggregated sorting list. Besides, aggregating all potential lists together does not assure generating better final list because QPR2 list is more successful than QPR1R2. As a result, we achieve to outperform simple sorting ideas fully by the rank aggregation in all sets. Moreover, it is possible to advance aggregation performance in some sets by changing SMSP sorting criterion from aggregated lists to the best base ranker while keeping QPR2 as main sorting criterion.

In this study, we test to use rank aggregation concept in scheduling problems. Linear and poisson regression analysis is also added to RA. Although we have good results, there are still many improvements can be done in both rank aggregation

concept and regression models. They are simply implemented to see if usage of them is effective in the considered scheduling problem. However, these promising results can cause performing of them in detail. As a future work, developing a meta-heuristic rank aggregation algorithm which obtained by natural properties of scheduling jobs makes sense. Also, ways of promoting rank aggregation can be investigated by several machine learning methods instead of the regression models in this study. Meta-heuristic algorithms in operations research typically needs for various of high quality initial solutions to start searching in feasible regions. This can be done in an effective way by RA methods.

APPENDIX A

ANCILLARY TABLES

Table 13: Lower and Upper Bounds For Randomly Generated Instance Parameter Values.

Parameter	Lower Bound	Upper Bound
α_{im}	1	3
β_{im}	0.01	0.1
γ_{im}	3	5
θ_{im}	0.01	0.1
ϵ_{im}	0.001	0.01
rp_m	1	5
rp_d	75	80
W_i'	5	15

Table 14: Quality Lower Bounds (QLB) Over Sets.

Set	J9M3	J15M5	J30M10	J15M3	J25M5	J50M10
QLB	4	4	4,3	3,75	3,75	4,15
Set	J30M3	J50M5	J100M10	J60M3	J100M5	J200M10
QLB	3,25	3,4	3,75	2,5	2,65	2,75

Table 15: Comparison of Priority and Quality Ratios Based Sorting Lists. Bold Numbers for Minimum Mean Objective in a Set. Pairwise Significance Test Results Shown Under Worse Mean Objectives.

3 Machines			5 Machines			10 Machines		
Jobs	Priority	Quality	Jobs	Priority	Quality	Jobs	Priority	Quality
9	19,67 0,18	**16,71** -	15	11,18 0,85	**10,26** -	30	10,43 0,35	**9,54** -
15	14,60 0,08	**13,46** -	25	**10,89** -	11,20 0,02	50	12,03 0,03	**11,32** -
30	17,83 0,75	**17,74** -	50	**17,28** -	17,54 0,33	100	17,75 0,01	**16,72** -
60	34,75 0,86	**34,62** -	100	**31,85** -	32,38 0,26	200	**29,52** -	29,93 0,18

Table 16: Shapiro Test P-values for Noted Aggregations and the Best of Simple and Regression-Based Lists Over Sets

Lists	J9M3	J15M5	J30M10	J15M3	J25M5	J50M10	J30M3	J50M5	J100M10	J60M3	J100M5	J200M10
QPR1R2	0	0	0	0	0,04	0	0,43	0,19	0,32	0,93	0,5	0
QPR1	0	0	0	0	0,03	0,01	0,03	0,22	0,45	0,9	0,21	0,58
QPR2	0	0	0	0	0,53	0,22	0,47	0,59	0,12	1	0,65	0,35
QR2	0	0	0	0	0,37	0,07	0,86	0,94	0,53	0,13	0,97	0,54
Simple	0	0	0	0	0	0	0,87	0,46	0,9	1	0,29	0,81
Reg	0,01	0	0	0	0,02	0,01	0,12	0,64	0,07	0,23	0,06	0,03

Bibliography

[1] M. Cheng, P. R. Tadikamalla, J. Shang, and S. Zhang, "Bicriteria hierarchical optimization of two-machine flow shop scheduling problem with time-dependent deteriorating jobs," *European Journal of Operational Research*, vol. 234, no. 3, pp. 650 – 657, 2014.

[2] K. Rustogi and V. A. Strusevich, "Single machine scheduling with general positional deterioration and rate-modifying maintenance," *Omega*, vol. 40, no. 6, pp. 791–804, 2012.

[3] K. Rustogi and V. A. Strusevich, "Combining time and position dependent effects on a single machine subject to rate-modifying activities," *Omega*, vol. 42, no. 1, pp. 166–178, 2014.

[4] K. Rustogi and V. A. Strusevich, "Single machine scheduling with time-dependent linear deterioration and rate-modifying maintenance," *Journal of the Operational Research Society*, vol. 66, pp. 500–515, Mar 2015.

[5] Y. Yin and D. Xu, "Some single-machine scheduling problems with general effects of learning and deterioration," *Computers & Mathematics with Applications*, vol. 61, no. 1, pp. 100–108, 2011.

[6] P.-J. Lai, W.-C. Lee, and H.-H. Chen, "Scheduling with deteriorating jobs and past-sequence-dependent setup times," *The International Journal of Advanced Manufacturing Technology*, vol. 54, pp. 737–741, May 2011.

[7] X.-R. Wang and J.-J. Wang, "Single-machine scheduling with convex resource dependent processing times and deteriorating jobs," *Applied Mathematical Modelling*, vol. 37, no. 4, pp. 2388–2393, 2013.

[8] C.-C. Wu, Y.-R. Shiau, and W.-C. Lee, "Single-machine group scheduling problems with deterioration consideration," *Computers & Operations Research*, vol. 35, no. 5, pp. 1652–1659, 2008.

[9] T. E. Cheng, C.-J. Hsu, Y.-C. Huang, and W.-C. Lee, "Single-machine scheduling with deteriorating jobs and setup times to minimize the maximum tardiness," *Computers & Operations Research*, vol. 38, no. 12, pp. 1760–1765, 2011.

[10] W.-H. Kuo and D.-L. Yang, "Single-machine scheduling with deteriorating jobs," *International Journal of Systems Science*, vol. 43, no. 1, pp. 132–139, 2012.

[11] Y. Yin, W.-H. Wu, T. Cheng, and C.-C. Wu, "Single-machine scheduling with time-dependent and position-dependent deteriorating jobs," *International Journal of Computer Integrated Manufacturing*, vol. 28, no. 7, pp. 781–790, 2015.

[12] M. J. Hayat and M. Higgins, "Understanding poisson regression," *Journal of Nursing Education*, vol. 53, no. 4, pp. 207–215, 2014.

[13] Z. ShahabiKargar, S. Khanna, N. Good, A. Sattar, J. Lind, and J. O'Dwyer, "Predicting procedure duration to improve scheduling of elective surgery," *PRICAI 2014: Trends in Artificial Intelligence*, pp. 998–1009, 2014.

[14] A. Asnaashari, E. A. McBean, I. Shahrour, and B. Gharabaghi, "Prediction of watermain failure frequencies using multiple and poisson regression," *Water Supply*, vol. 9, no. 1, pp. 9–19, 2009.

[15] X. Li, X. Wang, and G. Xiao, "A comparative study of rank aggregation methods for partial and top ranked lists in genomic applications," *Briefings in Bioinformatics*, vol. 20, no. 1, pp. 178–189, 2019.

[16] C. Domshlak, A. Gal, and H. Roitman, "Rank aggregation for automatic schema matching," *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, vol. 19, no. 4, 2007.

[17] T. Cheng, Q. Ding, and B. Lin, "A concise survey of scheduling with time-dependent processing times," *European Journal of Operational Research*, vol. 152, no. 1, pp. 1–13, 2004.

[18] S. Gawiejnowicz, *Time-dependent scheduling*. Springer Science & Business Media, 2008.

[19] J.-B. Wang and J.-J. Wang, "Single machine group scheduling with time dependent processing times and ready times," *Information Sciences*, vol. 275, pp. 226–231, 2014.

[20] W.-C. Lee, "Single-machine scheduling with past-sequence-dependent setup times and general effects of deterioration and learning," *Optimization Letters*, pp. 1–10, 2014.

[21] K. Rustogi and V. A. Strusevich, "Single machine scheduling with a generalized job-dependent cumulative effect," *Journal of Scheduling*, Sep 2016.

[22] J.-B. Wang and M.-Z. Wang, "Single-machine scheduling with nonlinear deterioration," *Optimization Letters*, vol. 6, no. 1, pp. 87–98, 2012.

[23] D. Oron, "Single machine scheduling with simple linear deterioration to minimize total absolute deviation of completion times," *Computers & Operations Research*, vol. 35, no. 6, pp. 2071–2078, 2008.

[24] Z.-L. Chen, "Parallel machine scheduling with time dependent processing times," *Discrete Applied Mathematics*, vol. 70, no. 1, pp. 81–93, 1996.

[25] M. Ji and T. E. Cheng, "Parallel-machine scheduling with simple linear deterioration to minimize total completion time," *European Journal of Operational Research*, vol. 188, no. 2, pp. 342–347, 2008.

[26] G. Mosheiov, "Scheduling jobs under simple linear deterioration," *Computers & Operations Research*, vol. 21, no. 6, pp. 653–659, 1994.

[27] L. Kang and C. Ng, "A note on a fully polynomial-time approximation scheme for parallel-machine scheduling with deteriorating jobs," *International Journal of Production Economics*, vol. 109, no. 1, pp. 180–184, 2007.

[28] A. J. Ruiz-Torres, G. Paletta, and E. PéRez, "Parallel machine scheduling to minimize the makespan with sequence dependent deteriorating effects," *Computers & Operations Research*, vol. 40, no. 8, pp. 2051–2061, 2013.

[29] M. L. Fisher and A. M. Krieger, "Analysis of a linearization heuristic for single-machine scheduling to maximize profit," *Mathematical Programming*, vol. 28, no. 2, pp. 218–225, 1984.

[30] T. G. Voutsinas and C. P. Pappis, "Scheduling jobs with values exponentially deteriorating over time," *International Journal of Production Economics*, vol. 79, no. 3, pp. 163–169, 2002.

[31] S. Raut, S. Swami, and J. N. Gupta, "Scheduling a capacitated single machine with time deteriorating job values," *International Journal of Production Economics*, vol. 114, no. 2, pp. 769–780, 2008.

[32] A. Janiak, T. Krysiak, C. P. Pappis, and T. G. Voutsinas, "A scheduling problem with job values given as a power function of their completion times," *European Journal of Operational Research*, vol. 193, no. 3, pp. 836–848, 2009.

[33] M. Lichman and P. Smyth, "Prediction of sparse user-item consumption rates with zero-inflated poisson regression," *WWW 2018: The 2018 Web Conference*, 2018.

[34] H. Moskowitz and Y. H. Chun, "A poisson regression model for two-attribute warranty policies," *Naval Research Logistics*, vol. 41, pp. 355–376, 1994.

[35] C. Zacharias and M. Pinedo, "Appointment scheduling with no-shows and overbooking," *Production and Operations Management*, vol. 23, no. 5, pp. 788–801, 2014.

[36] U. Subramanian, R. T. Ackermann, E. J. Brizendine, C. Saha, M. B. Rosenman, D. R. Willis, and D. G. Marrero, "Effect of advanced access scheduling on processes and intermediate outcomes of diabetes care and utilization," *Journal of General Internal Medicine*, vol. 24, no. 3, pp. 327–333, 2009.

[37] S. Schlagkamp, M. Hofmann, L. Eufinger, and R. F. da Silva, "Increasing waiting time satisfaction in parallel job scheduling via a flexible milp approach," *International Conference on High Performance Computing & Simulation*, 2016.

[38] J. A. Nasir and C. Dang, "Quantitative thresholds based decision support approach for the home health care scheduling and routing problem," *Health Care Management Science*, vol. 23, pp. 215–238, 2019.

[39] N. Klicos, J. Winzeler, D. York, B. Tai, and R. Bailey, "Developing an algorithm-based course scheduling tool," *IEEE Systems and Information Engineering Design Symposium*, 2008.

[40] S. Lin, "Rank aggregation methods," *Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 5, pp. 555–570, 2010.

[41] S. Lin, "Space oriented rank-based data integration," *Statistical Applications in Genetics and Molecular Biology*, vol. 9, no. 1, 2010.

[42] J. A. Aledo, J. A. Gamez, and A. Rosete, "Partial evaluation in rank aggregation problems," *Computers and Operation Research*, vol. 78, pp. 299–304, 2016.

[43] E. M. García-Nové, J. Alcaraz, M. Landete, J. F. Monge, and J. Puerto, "Rank aggregation in cyclic sequences," *Optimization Letters*, vol. 11, pp. 667–678, 2017.

[44] R. Kolde, S. Laur, P. Adler, and J. Vilo, "Robust rank aggregation for gene list integration and meta-analysis," *Bioinformatics*, vol. 28, no. 4, pp. 573–580, 2012.

[45] V. Pihur, S. Datta, and S. Datta, "Weighted rank aggregation of cluster validation measures: a monte carlo cross-entropy approach," *Bioinformatics*, vol. 23, no. 13, pp. 1607–1615, 2007.

[46] A. Klementiev, D. Roth, and K. Small, "An unsupervised learning algorithm for rank aggregation," *European Conference on Machine Learning*, vol. 4701, pp. 616–623, 2007.

[47] G. B. Choi, J. W. Kim, J. C. Suh, K. H. Jang, and J. M. Lee, "A prioritization method for replacement of water mains using rank aggregation," *Korean Journal of Chemical Engineering*, vol. 34, pp. 2584–2590, 2017.

[48] W. Yu, S. Li, X. Tang, and K. Wang, "An efficient top-k ranking method for service selection based on -admopso algorithm," *Neural Computing and Applications*, vol. 31, pp. 77–92, 2019.

[49] K. Deng, S. Han, K. J. Li, and J. S. Liu, "Bayesian aggregation of order-based rank data," *Journal of the American Statistical Association*, pp. 1023–1039, 2014.

VITA

Mücahit Kaan Kaleli graduated from Izmir Nevvar Salih İşgören Anatolian High School in 2013. He received his B.S degree in Industrial Engineering from Özyeğin University in February 2019. After graduation, he joined Master of Science program in Industrial Engineering at Özyeğin University and has worked under supervision of Dr. Z. Melis Teksan and Dr. Erinç Albey. He has worked as a teaching assistant and assisted Dr. Teksan in course of Productions Systems Analysis. His research focuses on optimization in scheduling problems, machine learning and rank aggregation methods.