A RESCHEDULING PROBLEM WITH CONTROLLABLE PROCESSING TIMES: TRADE-OFF BETWEEN NUMBER OF DISRUPTED JOBS AND RESCHEDULING COSTS

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

BY

DERYA CİNCİOĞLU

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

DECEMBER 2011

Approval of the thesis:

A RESCHEDULING PROBLEM WITH CONTROLLABLE PROCESSING TIMES: TRADE-OFF BETWEEN NUMBER OF DISRUPTED JOBS AND RESCHEDULING COSTS

submitted by **DERYA CİNCİOĞLU** in partial fulfillment of the requirements for the degree of **Master of Science in Industrial Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Sinan Kayalıgil
Head of Department, **Industrial Engineering** _____

Assist. Prof. Dr. Sinan Gürel
Supervisor, **Industrial Engineering Department, METU** _____

Examining Committee Members:

Prof. Dr. Meral Azizoğlu
Industrial Engineering Dept., METU _____

Assist. Prof. Dr. Sinan Gürel
Industrial Engineering Dept., METU _____

Prof. Dr. M. Selim Aktürk
Industrial Engineering Dept., BİLKENT UNIVERSITY _____

Assist. Prof. Dr. Pelin Bayındır
Industrial Engineering Dept., METU _____

Assist. Prof. Dr. Serhan Duran
Industrial Engineering Dept., METU _____

Date: _____

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

Name, Last Name: DERYA CİNCİOĞLU

Signature :

ABSTRACT

A RESCHEDULING PROBLEM WITH CONTROLLABLE PROCESSING TIMES: TRADE-OFF BETWEEN NUMBER OF DISRUPTED JOBS AND RESCHEDULING COSTS

Cincioğlu, Derya

M.Sc., Department of Industrial Engineering

Supervisor : Assist. Prof. Dr. Sinan Gürel

December 2011, 76 pages

In this thesis, we consider a rescheduling problem on non-identical parallel machines with controllable processing times. A period of unavailability occurs on one of the machines due to a machine failure, material shortage or broken tool. These disruptions may cause the original schedule to become inefficient and sometimes infeasible. In order to generate a new and feasible schedule, we are dealing with two conflicting measures called the efficiency and stability measures simultaneously. The efficiency measure evaluates the satisfaction of a desired objective function value and the stability measure evaluates the amount of change between the schedule before and after the disruption. In this study, we measure stability by the number of disrupted jobs. In this thesis, the job is referred as a disrupted job if it completes processing after its planned completion time in the original schedule. The efficiency is measured by the additional manufacturing cost of jobs. Decreasing number of disrupted jobs requires compressing the processing time of a job which cause an increase in its additional manufacturing cost. For that reason we cannot minimize these objectives at the same time. In order to handle this, we developed a mixed integer programming model for the considered problem by applying the ϵ-constraint approach. This approach makes focusing on the single objective

possible to get efficient solutions. Therefore, we studied the problem of minimizing additional manufacturing cost subject to a limit on the number of disrupted jobs. We also considered a convex compression cost function for each job and solved a cost minimization problem by applying conic quadratic reformulation for the model. The convexity of cost functions is a major source of difficulty in finding optimal integer solutions in this problem, but applying strengthened conic reformulation has eliminated this difficulty. In addition, we prepare an improvement search algorithm in order to find good solution in reasonable CPU times. We use our heuristic procedure on optimality properties we showed for a single machine subproblem. We made computational experiments on small and medium scale test problems. Afterwards, we compare the performance of the improvement search algorithm and mathematical model for their solution quality and durations.

Keywords: Rescheduling, Controllable Processing Time, Heuristics, Non-identical Parallel Machines, Manufacturing Cost

ÖZ

KONTROL EDİLEBİLİR İŞLEM SÜRELERİYLE YENİDEN ÇİZELGELEME: ARIZADAN ETKİLENEN İŞ SAYISI VE YENİDEN ÇİZELGELEME MALİYETİ ARASINDAKİ İLİŞKİLER

Cincioğlu, Derya

Yüksek Lisans, Endüstri Mühendisliği Bölümü

Tez Yöneticisi : Yrd. Doç. Dr. Sinan Gürel

Aralık 2011, 76 sayfa

Bu çalışma, kontrol edilebilir işlem sürelerinin söz konusu olduğu özdeş olmayan paralel makinalarda işlerin yeniden çizelgelenmesi problemini dikkate almaktadır. Makine arızası, hammadde eksikliği ya da arızalanan bir parça sebebiyle makinelerden birinin belli bir dönem kullanım dışı olması söz konusu olabilmektedir. Bu aksaklıklar mevcut çizelgenin verimsiz ilerlemesine, hatta bazen de tamamlanmasının mümkün olmamasına sebebiyet verebilmektedir. Yeni ve uygulanabilir bir çizelge oluşturabilmek içinse, kararlılık ve verimlilik gibi çatışan iki ölçüm değerini aynı anda hesaba katmak gerekmektedir. Verimlilik, istenen hedef fonksiyon değerini ölçerken, kararlılıksa arıza öncesi ve sonrası çizelgede meydana gelen sapmanın büyüklüğünü ölçmektedir. Bu çalışmada biz kararlılığı geç tamamlanan işlerin sayısı cinsinden ölçümlendirmekteyiz. Bu tezde; yeniden çizelgelenen bir iş eğer arıza oluşmadan önceki çizelgedeki planlanan tamamlanma zamanından daha geç tamamlanıyorsa, geç tamamlanan iş olarak adlandırılmaktadır. Verimlilik, işleri ek üretim maliyetleri cinsinden ölçümlendirmektedir. Geç tamamlanan işlerin azaltılması işlerin işlem sürelerinin sıkıştırılmasını gerektirmektedir, bu da işlerin ek üretim maliyetinlerinde artışa sebep olmaktadır. Bu sebeple her iki ölçüm değerini aynı anda azaltmak mümkün olmamaktadır. Bu soruna çözüm bulabilmek

amacıyla ϵ-kısıt yaklaşımı uygulanan karışık tamsayılı doğrusal programlama modeli geliştirilmiştir. Bu yaklaşım tek bir ölçüm hedefine odaklanarak verimli sonuçlar elde etmeye olanak sağlamaktadır. Bu yüzden biz de ek üretim maliyetlerinin geç tamamlanan iş sayısı kısıtlanacak şekilde en aza indirilmesi problemi üzerine çalışmalar yaptık. Ayrıca her iş için konveks sıkıştırma maliyeti fonksiyonunu hesaba katıp, maliyeti en aza indirme problemini ikinci derece konik programlama kullanarak modelimiz için çözümledik. Maliyet fonksiyonlarının konvekslik özelliği optimum çözümlerin bulunmasının önündeki en büyük sorun kaynağını oluşturmaktaydı. Fakat güçlendirilmiş konik programlama metotlarını kullanarak bu sorunun da üstesinden gelinmiştir. Ayrıca, makul CPU sürelerinde uygun çözümler bulabilmek amacıyla bir geliştirici tarama algoritması da geliştirilmiştir. Önerilen algoritmanın uygulama aşamasında kullanmak üzere, en iyileme özellikleri tek makina çizelgeleme problemi için araştırılmıştır. Küçük ve orta ölçekli test problemleri için sayısal deneyler yapılarak, matematiksel model ve geliştirici tarama algoritmasının performansları çözüm kalitesi ve süreler cinsinden karşılaştırılmıştır.

Anahtar Kelimeler: Yeniden Çizelgeleme, Değişken İşlem Zamanı, Sezgisel Algoritmalar, Özdeş Olmayan Paralel Makinalar, Üretim Maliyeti

Dedicated to
Süleyman Cincioğlu...

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my supervisor Assist. Prof. Dr. Sinan Gürel for his guidance, advice, criticism and insight throughout this study. I am very grateful to him for introducing me the subject and for his positive attitude all the time.

I want to thank Prof. Dr. Meral Azizoğlu, Prof. Dr. M. Selim Aktürk, Assist. Prof. Dr. Pelin Bayındır and Assist. Prof. Dr. Serhan Duran for their comments and suggestions that have provided valuable enhancements for my thesis.

Finally, and most importantly, I wholeheartedly thank to my husband, Süleyman Cincioğlu for his love, encouragement and sacrifice. He is my constant source of support and strength throughout this endeavor. I am indebted to him more than he knows and this dissertation is impossible without him.

I would like to thank to TÜBİTAK for providing the financial support for my MSc study in METU.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

Reactive scheduling is an important approach in coping with unexpected disruptions that may occur during the execution of a schedule. Besides resource allocation and job sequencing decisions, reactive scheduling often involves changing processing times of tasks. Compressing processing times expedites jobs and helps to recover a schedule from a disruption. On the other hand, compressing processing times requires using additional resources which result higher processing costs. Taking the advantage of processing time controllability, against a disruption one can generate alternative schedules with different additional resource costs and different levels of deviations from initial schedules. In this study, we show how rescheduling decisions can be made with controllable processing times.

A breakdown on a machine may cause the machine to become unavailable for a certain period of time. This would result a number of jobs to be disrupted with respect to their planned completion times in the original schedule. If a job completes processing after its planned completion time, it is referred as a disrupted job in this study. Therefore, such a disruption may cause delayed deliveries to the customers. Thus, a natural objective to consider in a rescheduling problem is the number of disrupted jobs. This objective is an estimate of the number of unsatisfied customers due to delayed deliveries.

Controllable processing times appear in a variety of industrial applications. CNC turning operation is a well known example for processing time controllability. In CNC turning machine example, processing time of a turning operation can be controlled via setting the cutting speed and/or the feed rate. Increasing the speed and the feed rate of a machine results compressed processing times but higher depreciation on cutting tools and hence increased tooling costs. Industrial applications that involve controllable processing times generally reflect the law of

diminishing marginal returns. Thus, marginal cost of compressing processing time of a job gets more expensive as we compress it further. This implies a nonlinear relationship between the amount of compression on the processing time of a job and the additional processing cost incurred to compress the job. In this study, we consider convex compression cost functions and we handle the nonlinear terms using second-order conic programming inequalities.

Recent works in the literature have shown the efficacy of using processing time controllability in reactive scheduling. A common approach in rescheduling literature is to insert idle times in a schedule so that effects of a possible disruption can be absorbed. However, this approach may cause lower utilization levels on a machine. Furthermore, despite careful planning, idle times may not coincide with the time of breakdown and hence may be useless. In the literature, there are few papers that make use of controllable processing times in reactive scheduling. Hence, this work contributes to fill this gap in the literature.

In this thesis, we consider a parallel machine scheduling environment where there exists an original schedule. On given schedule, a random breakdown occurs on one of the machines. At the time of breakdown, there may be jobs which are already completed and other jobs which are not processed yet. Our task is to reschedule those remaining jobs on the machines where each machine has a fixed end-time specified in the original schedule. We have two objectives the first one of which is the additional processing or manufacturing cost due to compressions on the processing times and job reallocations among machines. The second objective to minimize is the number of disrupted jobs.

We consider the ϵ-constraint approach and solve a rescheduling problem that minimizes additional manufacturing cost for a given upper bound on number of disrupted jobs. This problem gives efficient solutions for given number of disrupted jobs.

CHAPTER 2

LITERATURE REVIEW

In this Chapter, we will first give a literature review on rescheduling and then we will discuss on the rescheduling literature with controllable processing times. We will then mention the related work on predictive scheduling. We will next give a review of scheduling with controllable processing times. Finally, we will end this chapter with a review of works related to the number of tardy jobs objective which is similar to our number of disrupted jobs objective. Furthermore, we emphasize the similarities and differences of our study with the existing studies in the literature.

2.1 Rescheduling

The literature on scheduling theory is rich with numerous different problems and solution methods. Rescheduling problems have received a significant attention in the recent literature. Many different problems and solution approaches have been presented. An extensive discussion on rescheduling theory is given by Vieira et al. [1]. Types of uncertainties faced in the execution of schedules and possible reactive/predictive solution approaches were discussed by Aytug et al. [2].

The rescheduling literature covers different rescheduling environments like different machine settings and different job characteristics. There are majorly two approaches: the one which assumes a deterministic environment with all information given and the other which assumes that information is uncertain. Different approaches were considered to solve rescheduling problems such as dynamic rescheduling in which on-line methods that are invoked by events like new job arrivals or disruptions. In contrast there are papers which propose predictive

scheduling approaches which aim to reduce negative effects of possible disruptions by focusing on generating "good" initial schedules.

Another aspect of rescheduling is its frequency or timing of rescheduling activities. Alternatives can be rescheduling after each disruption or rescheduling periodically at a given frequency. It is also critical to decide between making a partial rescheduling or complete rescheduling. The first approach revises only a selected part of the disrupted schedule to avoid nervousness in the system while the latter one generates a new schedule from scratch. A comparison of partial and complete rescheduling approaches is given by Leon et al. [3]. Right-shifting is the simplest approach in rescheduling. This method is a simple shifting of the schedule by the amount of the disruption. Thus, the longer the disruption, the larger the expected shift, and the greater the increase in makespan. Right-shifting is compared to different rescheduling approaches by Leon et al. [3] and Nof and Grant [4]. A set of dispatching rules were compared in a flexible manufacturing environment by Kutanoğlu and Sabuncuoğlu [5]. Stochastic processing times were considered by Wu et al. [6] and Sabuncuoğlu and Karabük [7]. Rescheduling in response to the arrival of a new set of jobs were studied by Hall and Potts [8]. In their work, scheduling objectives (such as lateness and total completion time) and stability objectives (such as change in job's positions and completion times) were considered. They give algorithms and complexity analysis for different models. In this study, we consider rescheduling on parallel machines after a machine breakdown on one of the machines.

Alagöz and Azizoğlu [9] studied a problem with the objectives of minimizing total completion time and minimizing number of disrupted jobs in a rescheduling environment. They consider the total flow time objective as an efficiency measure and a stability measure of number of jobs reassigned to another machine than its original machine in the preschedule. They provide an algorithm for minimizing the stability measure subject to the constraint that the efficiency measure is at its minimum level. Furthermore, they propose heuristic procedures to generate a set of approximate efficient schedules relative to efficiency and stability measures.

Furthermore, Azizoğlu and Alagöz [10] studied a rescheduling problem on parallel machines where an unavailability period occurs on one of the machines. They consider the total flow time objective as an efficiency measure and a stability measure of number of jobs reassigned to another machine than its original machine in the preschedule. They show that efficient solution set for the considered objectives can be generated in polynomial time.

2.2 Rescheduling with Controllable Processing Times

Despite its flexibility and efficiency, the use of controllable processing times in rescheduling literature is limited. Turkcan et al. [11] consider reactive scheduling against machine breakdowns on non-identical parallel CNC machines. They consider minimizing the sum of earliness/tardiness of jobs and manufacturing costs while keeping the absolute difference between completion times of jobs in the new and the initial schedule at minimum. They provide a heuristic approach in order to generate a new schedule after a disruption occurs. Yang [12] considers new job arrivals to a given schedule on a single machine. The objective is to minimize the total cost after rescheduling. The objective includes three different cost terms. The first one is the sum of deviations on the start times of existing jobs, the second term is the total compression cost incurred via processing time control and the third term is one of the two following scheduling related objectives: total completion time and weighted tardiness. The author proposes a heuristic algorithm to solve those problems.

Match-up scheduling is a rescheduling approach which aims to catch up the preschedule within a certain time after disruption occurs. Match up scheduling examples are given by Bean et al. [13] and Aktürk and Görgülü [14]. These two studies propose heuristic approaches to find match-up times. Aktürk et al. [15] provide a rescheduling problem where one of the objectives is to catch up the initial schedule in the shortest time possible which is called the match-up time. The second objective they consider is the additional manufacturing cost of changing the processing times and machine assignments of jobs. They give mathematical formulations to get exact solutions and provide heuristic approaches. In this thesis, we consider a rescheduling environment that provides us controllability in processing times of jobs.

2.3 Predictive Scheduling - Idle Time Insertion

In the literature, another way of coping with schedule disruptions is preparing robust initial schedules. Most studies propose ways to decide when to insert idle times in a schedule. Yang and Geunes [16] consider idle time insertion problem in the context of project management for a firm where there are uncertain jobs in the future. They consider minimizing the sum of expected tardiness cost, schedule disruption cost, and wasted idle time cost. In order to

reduce the effects of disruption Mehta and Uzsoy [17] propose an idle time insertion approach in a job shop schedule. O'Donovan et al. [18] consider possible machine breakdowns on a single machine and provide idle time insertion method to minimize the expected deviation in completion times which may result due to a breakdown. They, observe via experimental study that idle time insertion can help to improve stability measures with little effect on other performance measures. Leus and Herroelen [19] consider minimizing expected deviation between planned and actual start time of the jobs for a schedule on a machine. They find the optimal job sequence and the amount of idle time to be inserted after each job in the schedule.

Taking the idle time insertion approach, a decision maker sacrifices utilization of machines for robustness of schedules. If no disruption occurs or if an inserted idle time is not used in absorbing a disruption then there occurs unused capacity on the machine. In industrial applications where processing time controllability exists leaving intended idle times in a schedule implies more compression on the processing times of the jobs which imply higher manufacturing costs. In this study, we use processing time controllability in a reactive scheduling problem which involves minimizing number of disrupted jobs with respect to initial completion times.

2.4 Scheduling with Controllable Processing Times

Specifically, machine scheduling is making the decision of which resources will be assigned to which jobs in which sequence and at what time in order to optimize some performance measures. Processing time controllability is an important dimension of research in scheduling, because in a real manufacturing environment, generally, we can control the processing times of the jobs within some technical constraints.

Shabtay and Kaspi [20] studied a nonlinear relationship between processing times and resource consumption which is the problem of scheduling jobs on a single machine to minimize total weighted flow time subject to limited resource. They provided optimality properties for the problem and also a dynamic programming algorithm. Shabtay and Kaspi [21] also considered minimizing the total completion time subject to a maximal resource constraint. Controllable processing times have been considered in the scheduling literature for the last 30 years. However, most of the research has been conducted in the last 10 years. Steiner [22] pro-

vide an extensive survey on the area of scheduling with controllable processing times. With processing time controllability, one has to consider compression (or process) costs as well as time based scheduling performance measures. Therefore, usually we need to solve multi-objective scheduling problems. Hoogeven [23] gives a survey on multi-objective scheduling literature which also includes multi-objective scheduling with controllable processing times.

Controllable processing times and hence the time-cost trade-offs receive the attention of researchers for different industrial or project management applications. In a recent one, Megow et al. [24] consider turnaround scheduling problem which involves scheduling large-scale maintenance activities in a production plant which is shutdown for entire inspection, maintenance or renewal. They consider that maintenance activities can be expedited via assigning more resource such as more workers to maintenance tasks. The problem is to trade-off resource costs for the cost of production loss. Gürel and Aktürk [25] studied making processing time and preventive maintenance planning decisions simultaneously for a CNC turning machine. Furthermore, Gürel and Aktürk [26] studied making optimal machine-job assignments and processing time decisions in order to minimize total manufacturing cost. They applied epsilon-constraint approach for the considered bicriteria problem. They give optimality properties for the resulting single criterion problem and provide alternative methods to compute cost lower bounds for partial schedules, which are used in developing an exact (branch and bound) algorithm. In addition to this work, they also propose a recovering beam search algorithm equipped with an improvement search procedure in order to find efficient solutions in reasonable durations.

Gürel and Aktürk [27] consider minimizing total manufacturing cost and total completion time objectives at the same time on identical parallel CNC turning machines and they prove some optimality properties for the considered problem which assisted during the design of an efficient heuristic algorithm to generate approximate non-dominated solutions.

Aktürk et al. [28] describe a polynomial-size conic quadratic reformulation for a machine-job assignment problem with separable convex cost. We apply their conic quadratic reformulation in our study, since we have the objective function having similar properties with the considered objective function.

In this study, we consider a rescheduling problem defined on a given initial schedule on parallel machines. A given disruption on one of the machines causes some jobs to become

disrupted with respect to their planned completion times in the original schedule. We deal with the trade-off between the number of disrupted jobs and the additional manufacturing cost resulting from compression of jobs and machine-job reallocations.

2.5 The Number of Tardy Jobs Objective in Scheduling Theory

The number of tardy jobs is a due-date related objective function that gives the number of jobs that are completed later than their due dates. This objective function is useful in assessing the on-time shipment performance of a production system. In the rescheduling problem that we are concerned with a job is tardy in the new schedule if it cannot be completed before its planned completion time in the initial schedule.

For the single machine scheduling problem with the objective of minimizing the number of tardy jobs, Moore [29] developed an $O(n \log n)$ algorithm, hence the problem can be solved in polynomial time. This algorithm is also known as Hodgson's algorithm in the literature. A detailed discussion of the algorithm with an application on a numerical example is given in Pinedo [30]. If the jobs are weighted than the problem becomes $\mathcal{NP}$-hard. Also, for the parallel machine environment the problem becomes $\mathcal{NP}$-hard.

The number of tardy jobs objective has been considered in the controllable processing time scheduling literature as well. But, to the best of our knowledge the work done is limited to the single machine case. Daniels and Sarin [31] studied the problem of joint sequencing and resource allocation by considering the number of tardy jobs as a scheduling criterion. They propose theoretical that are useful in developing a procedure for constructing the tradeoff curve between the number of tardy jobs and the total amount of allocated resource.

Cheng et al. [32] considered constructing the trade-off curve between the total amount of resource consumed and the number of tardy jobs on a single machine. They assumed a linear relationship between the resource usage and the processing time. They showed that for single machine case the problem of minimizing the total amount of allocated resource subject to a limited number of tardy jobs is $\mathcal{NP}$-hard. They proposed a pseudo-polynomial-time dynamic programming algorithm for constructing the trade-off curve. He et al. [33] extended this problem and different than Cheng et al. [32] they considered the objective of maximum compression cost and discretely controllable processing times. In a more recent work Yedidsion

et al. [34] considered convex resource consumption function and dealt with the problem of constructing trade-off curve for the number of tardy jobs and resource consumption objectives. They prove that the problem of minimizing total resource consumption subject to the number of tardy jobs is $\mathcal{NP}$-hard. Different than these studies, we work on a parallel machine environment.

We consider the problem of minimizing additional manufacturing cost subject to the number of disrupted jobs. For the considered problem, we give a mathematical model which reformulates convex compression costs via conic quadratic inequalities. Moreover, we give some optimality properties and based on those properties we develop an improvement search heuristic. In the next section, we give the problem definition and mathematical model.

CHAPTER 3

PROBLEM DEFINITION AND MODELING

In this Chapter, we first define our parallel machine rescheduling problem. Next, we present the assumptions for the problem. Finally, we provide a mathematical model for the considered problem.

3.1 Problem Definition

In this study, we consider a non-identical parallel machine environment. The term non-identical expresses that each job may have a different processing time, upper bound value on compression and compression cost function on different machines. We assume that the processing times of jobs can be controlled.

Initially, we assume that a set of original jobs has been scheduled optimally in the original schedule where each job has a planned start and a completion time. After the occurrence of a disruption on one of the machines, the original schedule needs to be regenerated. When rescheduling we consider only the jobs which are not started yet and the job which is interrupted during processing at the time of breakdown. In the original schedule each job has a planned completion time. One of the objectives of our problem is to reschedule jobs so that they will be completed before their original completion time if possible. If a job finishes after its original completion time then it is referred as a disrupted job in this thesis as we mentioned before.

We also consider the limitation that the last job on each machine will finish no later than the specified makespan in the original schedule which is the time to complete all jobs. Therefore, available machining time capacity for a machine at the time of breakdown, is determined

by calculating the difference between the makespan of the original schedule and the total processing time of processed jobs on that machine until the disruption occurs. Thus, jobs that will be assigned to that machine in the new schedule has to complete their processing within that time.

The scheduling environment consists of a set of n jobs to be processed on m parallel non-identical machines. Each job requires one operation at a time and each operation can be processed by either one of the available m machines. Since each job has different manufacturing properties and each machine has a different compression cost function, manufacturing cost function differs for each job on different machines.

In this study, we consider the problem of minimizing additional manufacturing cost. The additional manufacturing cost is the difference between costs of rescheduled jobs and their precalculated costs before the disruption occurs. As we mentioned before, our task is to reschedule remaining jobs without exceeding the available machining time. In order to satisfy this constraint, we need to compress processing times of jobs. Thus we need some additional resources like increasing the cutting speed and/or feeding rate of the machines to accomplish this requirement which lead to reduced tool life and hence increases manufacturing cost.

The cost of a change in the processing time of a job can be expressed as a function of compression amount y as follows:

$$f(y) = ky^{a/b}, \tag{3.1}$$

We will be using this form of the compression cost function as a part of our manufacturing cost function in our mathematical model. As we mentioned before, each job may have different compression cost function on different machines. The difference will be appear in the values of a, b and k. Where a and b are integers satisfying $a \geq b > 0$ and $k > 0$. Therefore *f(y)* is an increasing and convex function of the compression amount.

The function *f(y)* shows us the relationship between the compression amount and related compression cost. As one decreases the processing time of a job, it becomes more expensive to compress it further. Convexity of *f(y)* is modeling the increasing marginal cost of compression. Since compression cost function differs for each job on each machine, it is critical to

make both appropriate machine-job assignments and compression amount decisions.

3.2 Assumptions

The major assumptions about the problem environment and operations are presented below:

- There are non-identical parallel machines which are available unless a disruption occurs.
- There are n independent jobs and they have the same ready time at zero.
- Each job's processing time and upper bound value on compression are known, deterministic and they may have different values on each machine.
- When a disruption occurs in the middle of processing a job then this affected job has to be reprocessed in its entirety.
- Job loading and unloading times are negligible.
- There is no precedence relationships among the jobs.
- Each machine can process only one job at a time.

According to these assumptions we will try to make optimum processing time and job-machine assignment decisions simultaneously.

3.3 Modelling the Problem

In Section 3.1, we introduced our rescheduling problem. As we mentioned before, we are dealing with two conflicting objectives. The first one is to minimize the additional processing or manufacturing cost due to compressions on the processing times and job reallocations among machines. The second objective is to minimize the number of disrupted jobs. We want to minimize those two objectives but minimizing the number of disrupted jobs requires compressing the processing times of the jobs and this increases additional manufacturing cost due to the compression cost. Therefore we can not minimize both objectives at the same time.

A mathematical formulation for this bicriteria problem with controllable processing times is as follows, where;

Decision Variables:

Z_{ij}= 1, if job j is assigned as a disrupted job on machine i.
0, otherwise.

X_{ij}= 1, if job j is assigned on machine i.
0, otherwise.

x_{ijk}= 1, if start time of job j is scheduled before job k on machine i.
0, otherwise.

Y_{ij}= Compression on the processing time of job j on machine i.

y_{ijk}= Compression on the processing time of job j
if scheduled before job k on machine i.

Parameters:

p_{ij}= Processing time of job j on machine i.

c_{ij}= Cost of job j on machine i.

u_{ij}= Maximum possible compression for job j on machine i.

$f_{ij}(Y_{ij})$= Compression cost function for job j on machine i.

D_i= Available machining time capacity on machine i after disruption.

$\mathcal{S}$= Initial schedule on which a breakdown occurs.

J= Set of jobs not yet started processing at time of disruption.

I= Set of machines.

$mc\_start\_time_i$= Start time of machine i after disruption.

s_j= Start time of job j.

e_j= End time of job j.

K= Maximum number of disrupted jobs allowable.

F_J^S= Total manufacturing cost for the jobs in J in the original schedule S.

$$\min F1 : \sum_{i \in I} \sum_{j \in J} \Big(c_{ij} X_{ij} + f_{ij}(Y_{ij}) \Big) - F_J^{\mathcal{S}}$$

$$\min F2 : \sum_{i \in I} \sum_{j \in J} \big(Z_{ij} \big)$$

$$\text{(MCL)} \quad \text{s.t.} \sum_{i=1}^{m} X_{ij} = 1 \quad \forall j \in J \tag{3.2}$$

$$x_{ijk} \leq X_{ij} \quad \forall i, j, k \text{ and } j \neq k \tag{3.3}$$

$$x_{ijk} \leq X_{ik} \quad \forall i, j, k \text{ and } j \neq k \tag{3.4}$$

$$x_{ijk} + x_{ikj} \geq X_{ij} + X_{ik} - 1 \quad \forall i, j, k \text{ and } j \neq k \tag{3.5}$$

$$x_{ijk} + x_{ikj} \leq 1 \quad \forall i, j, k \text{ and } j \neq k \tag{3.6}$$

$$x_{ijk} + x_{ik\ell} + x_{i\ell j} \leq 2 \quad \forall i, j, k \text{ and } j \neq k \neq \ell \tag{3.7}$$

$$Z_{ij} \leq X_{ij} \quad \forall i, j \tag{3.8}$$

$$mc\_start\_time[i] + \sum_{j \in J \setminus s} (p_{ij} x_{ijs} - y_{ijs}) + (p_{is} X_{is} - Y_{is})$$

$$\leq e_s (1 - Z_{is}) + D_i Z_{is} \quad \forall i, s \tag{3.9}$$

$$Y_{ij} \leq u_{ij} X_{ij} \quad \forall i, j \tag{3.10}$$

$$y_{ijk} \leq Y_{ij} \quad \forall i, j, k \text{ and } j \neq k \tag{3.11}$$

$$y_{ijk} \leq u_{ij} x_{ijk} \quad \forall i, j, k \text{ and } j \neq k \tag{3.12}$$

$$X_{ij}, x_{ijk}, Z_{ij} \in \{0, 1\} \quad i \in I, j \in J, j \neq k \tag{3.13}$$

$$Y_{ij}, y_{ijk} \in \mathbb{R}^{+} \quad i \in I, j \in J, j \neq k \tag{3.14}$$

$$\tag{3.15}$$

The objective F1 is to minimize the additional manufacturing cost due to compressions on the processing times and job reallocations among machines. The objective F2 is to minimize the number of disrupted jobs. The constraint set (3.2) assigns each job to a machine. Constraint set (3.3) and (3.4) guarantee that a job can be the predecessor of another job if both are assigned to the same machine. Constraint set (3.5) forces a job to be the predecessor of another one if they both are assigned to the same machine and also prevents a job to be the predecessor of another if they are not assigned to the same machine. Constraint set (3.6) and (3.7) prevent the schedule from a cycle of the jobs. Constraint set (3.8) prevents assigning a job as a disrupted job on a machine without assigning that job to that machine. Constraint set (3.9)

guarantees that jobs which are not assigned as disrupted jobs will be completed processing without violating their planned completion times and disrupted jobs will be completed without violating the makespan. Constraint set (3.10) provides upper bounding on the amount of compression, guaranteeing that processing time of a job on a machine can be compressed only if the job is assigned on that machine and also the compression cannot be greater than the upper bound u_{ij}. Constraint set (3.11) guarantees that if a job is assigned to a machine and compressed then as a predecessor of any other jobs, it can be able to be compressed at the same amount of its predecided compression amount. Constraint set (3.12) provides upper bounding on the amount of compression of a predecessor, guaranteeing that processing time of a predecessor on a machine can be compressed only if the job is assigned on that machine as a predecessor of another job and also the compression of the predecessor cannot be greater than the upper bound u_{ij}.

For this bicriterion problem, we use the ϵ-constraint approach to find efficient solutions. In ϵ-constraint approach, one of the objectives is sent to the constraint set with a desired upper bound on it. According to this approach, we sent the second objective, which is minimizing the number of disrupted jobs, to the constraint set. As a result, we consider the problem of minimizing additional manufacturing cost subject to an upper bound on the number of disrupted jobs and give an effective formulation for the problem. Therefore we formulate a single objective problem as follows:

$$\min F1 : \sum_{i\in I}\sum_{j\in J}\Big(c_{ij}X_{ij} + f_{ij}(Y_{ij})\Big) - F_J^S$$

$$\text{(MCL)} \quad \text{s.t.} \sum_{i\in I}\sum_{j\in J}\Big(Z_{ij}\Big) \leq K \quad \forall j \in J \tag{3.16}$$

$$\text{Constraint Set} \qquad (3.2) - (3.12)$$

$$\tag{3.17}$$

Constraint set (3.16) guarantees that the number of disrupted jobs of the schedule is less than or equal to a predefined value K. As we mentioned before, our objective function includes f(y), which is the expression of a compression cost function. Due to the fact that compression cost function includes convex and nonlinear terms, our rescheduling problem model can be considered as a mixed integer nonlinear programming (MINLP) model which requires ex-

cessive branching to find integer feasible solutions. However in reactive scheduling, solution durations for the problems are quite critical. For that reason, we applied conic quadratic reformulation to our model which is the recent work by Aktürk et al. (2009). According to this study, conic quadratic inequalities can be applied for strengthening the formulations of problems having a separable convex objective and variable upper bounding constraints. Their approach is based on second-order cone programming (SOCP), which is also called conic quadratic programming. In this study, we implement this approach to minimize additional manufacturing cost objective and proposed an improvement search heuristic in order to create best or close to best schedules in reasonable durations.

CHAPTER 4

SINGLE MACHINE SUBPROBLEM AND OPTIMALITY PROPERTIES

In this Chapter, we present some optimality properties for the considered problem and we also aim to explain how they assist us during the rescheduling process.

4.1 The Method of Sequencing Jobs on a Single Machine with The Objective of Minimizing Number of Disrupted Jobs

As we mentioned in Section 3.1, there is a given schedule consisting of a set of n jobs to be processed by m parallel non-identical machines. Later a random breakdown occurs on one of the machines. Therefore, the given schedule is no longer executable. At the time of breakdown, there may be jobs which are already completed and other jobs which are not processed yet. Our task is to reschedule those remaining jobs on the machines.

In our study, we propose an improvement search heuristic to form a new schedule for the remaining jobs. In order to apply our proposed heuristic, an initial schedule needs to be generated as a first step. For that reason, an algorithm for minimizing the number of disrupted jobs in a single machine sequencing problem, which is a study of Moore [29], is considered. The algorithm repeatedly adds jobs in the Earliest Due Date (EDD) order to the end of a partial schedule of the jobs which complete processing on its due date at the very latest. If the addition of job j results in this job being completed after its due date d_j then a job in the partial schedule with the largest processing time is removed and declared as a disrupted job. All disrupted jobs are scheduled in the arbitrary order after the completion of undisrupted jobs. Therefore, an optimum schedule in which undisrupted jobs are sequenced in EDD

order is formed, ensuring to have the minimum number of disrupted jobs for the considered schedule. Therefore, we used the EDD order while forming the initial schedule in order to apply our proposed improvement search heuristic.

4.2 Single Machine Subproblem Definition and Modelling

Suppose that we have a set of jobs (J_m) assigned to a single machine (m). Assume that some of the jobs are assigned as disrupted jobs. Each undisrupted job j has a planned completion time d_j according to the original schedule. These undisrupted jobs are sequenced by the Earliest Due Date (EDD) by taking the Moore's Algorithm into account. Each disrupted job i has a planned completion time d_i which is updated as the makespan value of the schedule and they are sequenced in the arbitrary order after undisrupted jobs.

Suppose that the jobs are indexed according to their planned completion times such that $d_1 \leq d_2 \leq d_3 \ldots \leq d_k$. The problem is to minimize the total manufacturing costs of jobs subject to the constraint that the latest time to complete a job is its planned completion time. We again have the compression cost for each job and the processing time of each job can be compressed by u_j at most.

Then, the problem can be formulated as below:

$$\min : \sum_{j \in J_m} f_j(y_j)$$

$$\text{(SPm)} \quad \text{s.t.} \sum_{\ell=1}^{j} p_\ell - y_\ell \leq d_j \qquad j \in J_m \tag{4.1}$$

$$0 \leq y_{ij} \leq u_{ij} \qquad j \in J_m \tag{4.2}$$

According to the model given above, our single machine subproblem includes nonlinear terms in its objective function. For that reason, we use conic quadratic reformulation to solve this nonlinear programming model. If solution of the given set of jobs on a machine turns out to be infeasible, this means that no complete schedule can be achieved from this given set of jobs. Therefore, in order to form a feasible schedule alternative job assignments or allowable number of disrupted jobs need to be reconsidered.

In the next section, we study structural properties of an optimal solution to the single machine subproblem.

4.3 Optimality Properties

As we mentioned before, nonlinear and convex terms are included in our objective function. Therefore, the nonlinear convex cost function is tried to be minimized. For that reason optimality properties are studied to be used in our proposed algorithms to solve the problem. These properties are very important because they state the relationship of jobs' compression amounts and their related costs on the same machine in an optimal solution. Therefore by using these optimality properties, we can calculate the estimated cost lower bounds. So, optimality properties are studied for the single machine subproblem and they are used for calculations during our improvement search heuristic.

Since compression amounts vector y^* is a regular point, such a point must satisfy the Karush-Kuhn Tucker optimality conditions. Using these optimality conditions, we give some optimality properties for the single machine subproblem which is referred as SPm in this Chapter.

In Proposition 1, we studied the relationship between the marginal costs of compressed jobs having planned completion times respectively.

Proposition 1 *Let y_j^* be the optimal compression for job j in an optimal solution to SPm and $\lambda_j = \frac{\partial f_j}{\partial y_j}(y_j^*)$ be the derivative of $f_j(y_j)$ at y_j^*. Then, if $d_j \leq d_k$ and $y_j^* < u_j$ the following inequality always holds*

$$\lambda_j \geq \lambda_k$$

Proof. First we write the Lagrangian function for SPm.

$$L(y, \mu, \rho, \eta) = \sum_{j \in J_m} f_j(y_j) + \sum_{j \in J_m} \mu_j \left(\sum_{\ell=1}^{j} (p_\ell - y_\ell) - d_j \right) + \sum_{j \in J_m} \rho_j (y_j - u_j) - \sum_{j \in J_m} \eta_j y_j$$

where μ, ρ and η are the nonnegative Lagrangian dual variables for constraints (4.1)-(4.2), respectively. At the optimal solution

$$\frac{\partial L}{\partial y_j}(y_j^*) = 0$$

must hold. Then, for job each job j:

$$\frac{\partial f_j}{\partial y_j}(y_j^*) - \sum_{\ell=j}^{\ell=|J_m|} \mu_\ell + \rho_j - \eta_j = 0$$

holds. For two consecutive jobs j and $j+1$, it is easy to see that

$$\frac{\partial f_j}{\partial y_j}(y_j^*) - \frac{\partial f_{j+1}}{\partial y_{j+1}}(y_{j+1}^*) = \mu_j - \rho_j + \eta_j + \rho_{j+1} - \eta_{j+1}$$

Due to feasibility, μ_j, η_j and ρ_{j+1} must be nonnegative. Condition $y_j^* < u_j$ implies $\rho_j = 0$. $\eta_{j+1} > 0$ is only possible if $y_{j+1}^* = 0$ in which case $\frac{\partial f_{j+1}}{\partial y_{j+1}}(y_{j+1}^*) = 0 \leq \frac{\partial f_j}{\partial y_j}(y_j^*)$. Hence, $\lambda_j \geq \lambda_{j+1}$ always holds unless $y_j^* = u_j$. The proof for two consequent jobs is obviously sufficient for proving the general case. ■

As we mentioned before, the term λ is the derivative of $f_j(y_j)$ at y_j^* which expresses the marginal cost of compression for the considered job and machine. According to Proposition 1, if planned completion time of one job is earlier planned completion time of another job, earlier job's marginal cost of compression is greater than the later job's. Proposition 2 studies on a group of jobs, some of which complete their processing strictly on their planned completion times whereas others complete processing earlier than their planned completion times.

Proposition 2 *In an optimal schedule to problem SPm, consider a subsequence of jobs*

$$j, j+1, j+2, \ldots, j+\ell$$

such that job $j-1$ and $j+\ell$ complete at d_{j-1} and $d_{j+\ell}$, respectively, and jobs $j, j+1, \ldots, j+\ell-1$ finish strictly before their planned completion times. Then, for $k \in \{j, j+1, \ldots, j+\ell\}$ one of the following holds:

i. if $y_k^* < u_k$, then $\frac{\partial f_k}{y_k}(y_k^*) = \Lambda$.

ii. if $y_k^* = u_k$, then $\frac{\partial f_k}{y_k}(y_k^*) \leq \Lambda$.

where $\Lambda \geq 0$ is a constant.

Proof. First consider two jobs $k, k+l \in \{j, j+1, \ldots, j+\ell\}$. As shown in the proof of Proposition 1, we can write

$$\lambda_k - \lambda_{k+l} = \sum_{n=k}^{k+l-1} \mu_n - \rho_k + \eta_k + \rho_{k+l} - \eta_{k+l}$$

Consider four cases below:

Case I. If $0 < y_k^* < u_k$ and $0 < y_{k+l}^* < u_{k+l}$, then since $\mu_k = \mu_{k+1} = \ldots = \mu_{k+l}$ and $\rho_k = \eta_k = \rho_{k+l} = \eta_{k+l} = 0$, we have $\lambda_k = \lambda_{k+l} = \Lambda$.

Case II. If $y_k^* = 0$, then due to Proposition 1, $\lambda_k = \lambda_{k+l} = 0$.

Case III. If $y_{k+l}^* = 0$, then $\lambda_{k+l} = 0$.

i) If $0 < y_k^* < u_k$, then $\lambda_k = \lambda_{k+1} - \eta_{k+l}$ which implies $\lambda_k \leq 0$, so $\lambda_k = 0$.

ii) If $y_k^* = u_k$, then $\lambda_k = -\rho_k - \eta_{k+l}$ which is only possible when $\lambda_k = \rho_k = \eta_{k+l} = 0$.

Hence, $\lambda_k = \lambda_{k+l} = 0$.

Case IV. If $y_k^* = u_k$ and $0 < y_{k+l}^* < u_{k+l}$, then $\lambda_k = \lambda_{k+l} - \rho_k$, so, $\lambda_k \leq \Lambda$.

Case V. If $y_{k+l}^* = u_{k+l}$ and $0 < y_k^* < u_k$, then $\lambda_{k+1} = \lambda_k - \rho_{k+l}$, so $\lambda_{k+l} \leq \Lambda$.

■

Proposition 2 states that if we consider a group of jobs starting from one job which completes before its planned completion time and add other jobs to this group respectively until finding a job which completes strictly at its planned completion time. The job which completes strictly at its planned completion time will also be in that group. In conclusion, the jobs which are compressed less than their maximum possible compression amount has equal marginal cost of compression and the jobs having a compression amount equal to their maximum possible compression amount has less marginal cost of compression than the jobs which are compressed less than their maximum possible compression amount. By taking Proposition 2 into account Proposition 3 is generated as presented below.

Proposition 3 *An optimal schedule for SPm problem can be partitioned into subsequences*

$$\underbrace{1, \ldots, j_1}_{S_1}, \underbrace{j_1 + 1, \ldots, j_2}_{S_2}, \ldots, \underbrace{j_{k-1} + 1, \ldots, |J|}_{S_k}$$

such that jobs $j_1, j_2, \ldots, j_{k-1}, |J|$ *finish at their planned completion times and all other jobs finish strictly before their planned completion times. For each subsequence* S_i *there exists a common* Λ_i *as defined in Proposition 2. Furthermore, for the subsequences the following relation holds:*

$$\Lambda_1 \geq \Lambda_2 \geq \ldots \geq \Lambda_k$$

Proof. Obviously, any schedule can be partitioned into subsequences of the kind described in Proposition 2. Proposition 2 states that, with some exceptions, all jobs in such a subsequence S share a common level of λ_S which equals to $\frac{\partial f_j}{\partial y_j}$ for job $j \in S$. This, along with Proposition 1, which states that $\lambda_j \geq \lambda_k$ if $k < j$, implies the proposition. ■

As a result, Proposition 3 states that a given schedule can be partitioned into subsequent group of jobs by considering jobs' planned completion times. These subsequences includes jobs which complete before their planned completion times and a job which completes strictly at its planned completion time.

CHAPTER 5

IMPROVEMENT SEARCH HEURISTIC

In Chapter 3, we defined our problem and introduced the characteristics of it. We presented the assumptions for the problem. Afterwards, we provided a mathematical model for our problem. In Chapter 4, a set of optimality properties are presented. As we mentioned before, we are given a schedule consisting a set of n jobs to be processed by m parallel non-identical machines. After a random breakdown occurs, the original schedule becomes infeasible. There are both processed and unprocessed jobs until the disruption occurs and the need for rescheduling arises for the unprocessed jobs.

In this thesis we provide two alternatives in order to generate a new and feasible schedule, one of which is the rescheduling model and the other is the improvement search heuristic which will be presented in this Chapter. The optimality properties discussed in Chapter 4 will be important to understand our proposed improvement search heuristic.

5.1 Initial Schedule Algorithm

The approach of improvement search heuristic is fundamentally about applying two different moves to a schedule by shifting a job from its current interval on its current machine to another interval on another machine or exchange two jobs from their current location to each other's location. In order to apply such moves, intervals need to be determined which requires an initial schedule to be a basis for applying these moves. In this section, we propose our initial schedule algoritm in detail.

Initial schedule algorithm starts with a list of jobs, affected from the disruption, which are listed in ascending order according to their planned completion times in the original schedule.

At the beginning, we assume that all machines are empty and algorithm starts with the first job in the list. This scheduling process consists of two alternative assignment conditions for each job on each machine. The first condition is that the considered job will complete processing at its planned completion time at the very latest. The second condition is that the considered job will be assigned as a disrupted job which means that it will complete processing after than its planned completion time.

Initially, the first condition is applied to the considered job on each candidate machine for the assignment. For each assignment, single machine subproblem is solved and the objective function value is kept. If there exists no feasible solution by applying first condition and the allowable number of disrupted jobs is bigger than zero, the second condition is applied to the same considered job. For each assignment, single machine subproblem is solved again and the objective function value is kept as well. For the next step, objective function values are listed in ascending order and the machine having the minimum objective function value is selected and the job is assigned to that machine. We repeat this process until we assign all jobs to the machines. Initial solution algorithm either ends with a feasible schedule for the problem or fails to find a feasible schedule and stops. We repeat this process until we assign all jobs to the machines. Initial solution algorithm either ends with a feasible schedule for the problem or fails to find a feasible schedule and stops. If initial schedule algorithm finds a feasible schedule, we consider this initial schedule as a basis to produce our improvement search heuristic.

5.2 1-move Algorithm

The first move which is applied by the proposed improvement search heuristic is called 1-move. 1-move method aims cost improving move of a job from its current interval on its current machine to another interval on another machine by considering the assignment decision of the considered job.

If the job is assigned as a disrupted job while it is not a disrupted job on its current machine, its planned completion time will be updated with the makespan. Then, by considering each interval's start and end times on the candidate machine the considered job is placed on the interval in which the job's planned completion time lies. Due to this kind of assignment, the

number of disrupted jobs will be increased. For that reason for every disrupted job assignment, the number of disrupted jobs is updated. When it equals to the allowable number of disrupted jobs, disrupted job assignment is not allowed.

If the job is assigned as an undisrupted job while it is also an undisrupted job on its current machine or assigned as a disrupted job while it is also a disrupted job on its current machine, its planned completion time will not need any updates. Then again, by considering each interval's start and end times on the candidate machine the considered job is placed on the interval in which the job's planned completion time lies.

A 1-move yields compression cost improvement in its original machine since the compression for the remaining jobs can be decreased due to the additional machining time capacity that becomes available when the job leaves. On the other hand it increases the compression cost on the new machine as the jobs on that machine need to be compressed further to make up space for the new job to get a feasible schedule.

Different 1-move alternatives by considering both on-time and late assignments are listed below:

- If a job is a disrupted job on its current machine:
  - It can be assigned as a disrupted job to another machine.
  - It can be assigned as an undisrupted job to another machine.
- If a job is an undisrupted job on its current machine:
  - If the allowable number of disrupted job is greater than zero, it can be assigned as a disrupted job to another machine.
  - It can be assigned as an undisrupted job to another machine.

As a summary; 1-move algorithm is presented below:

5.2.1 Improvements by `1-move` Steps

In this subsection, we calculate a cost lower bound for the applied one move in a given schedule.

Algorithm 1 1-move Search Algorithm

Require: An initial schedule I and its cost F(I).

Initialize: improved $\leftarrow TRUE$;

while *improved* **do**

Generate all feasible 1-moves for each j in S;

Calculate LB for all moves;

if $LB \geq 0$ for all feasible moves **then**

BREAK;

else

Make a list of moves with $LB < 0$ in ascending order of LB's and select the first five of them.

Initialize: $found\_improving\_move \leftarrow FALSE$, $end\_of\_list \leftarrow FALSE$;

while NOT $found\_improving\_move$ and NOT end_of_list **do**

Do the next move in the list;

Solve single machine problem for affected machines;

New schedule is I';

if $COST(I') < COST(I)$ **then**

$I \leftarrow I'$;

$found\_improving\_move \leftarrow TRUE$, $improved \leftarrow TRUE$;

end if

end while

end if

end while

Proposition 4 *(Lower Bound for a 1-move) For a given schedule let Λ_{S_1} and Λ_{S_2} be optimal dual prices for subsequence S_1 and subsequence S_2, respectively. Suppose that S_1 is a subsequence on machine i_1 and S_2 is a subsequence on machine i_2. Let $y_{i_1 j}$ be the compression of job j in subsequence S_1. Then, a lower bound for the cost change that will result by moving job j from S_1 to S_2 is as stated below:*

$$LB(j : (S_1 \rightarrow S_2)) = -\Lambda_{S_1}(p_{i_1 j} - y_{i_1 j}) - c_{i_1 j} - f_{i_1 j}(y_{i_1 j}) + c_{i_2 j} + f_{i_2 j}(\hat{y}_{i_2 j}) + \Lambda_{S_2}(p_{i_2 j} - \hat{y}_{i_2 j}),$$

where $\hat{y}_{i_2 j} = \min((\partial f_{i_2 j}/\partial y_{i_2 j})^{-1}(\Lambda_{S_2}), u_{i_2 j})$.

Proof. The first three terms in $LB(j : (i_1 \rightarrow i_2))$ give a lower bound on the cost reduction by removing job j from subsequence S_1 on machine i_1; whereas the last three terms give a lower bound on the cost increase by inserting job j into subsequence S_2 on machine i_2. ■

5.3 2-swap Improvement Search

The second move which is applied by the proposed improvement search heuristic is called 2-swap, which is to exchange two jobs on each other's machine. 2-swap move can be considered as a combination of two 1-move's. As we mentioned in 1-move improvement search section; Every disrupted job assignment for undisrupted jobs, the number of disrupted jobs will be increased. For that reason, the number of disrupted jobs is updated. When it equals to the allowable number of disrupted jobs, disrupted job assignment is not allowed.

Different 2-swap alternatives of these jobs by considering both undisrupted and disrupted job assignments are listed below:

Without regarding if j_1 is an disrupted and undisrupted job on mc_1:

- j_1 can be assigned as an undisrupted job to mc_2.
- If the allowable number of disrupted job is greater than zero, j_1 can be assigned as a disrupted job to mc_2.

Without regarding if j_2 is an disrupted and undisrupted job on mc_2:

- j_2 can be assigned as an undisrupted job to mc_1.

- If the allowable number of disrupted job is greater than zero, j_2 can be assigned as a disrupted job to mc_1.

As a summary; 2-swap algorithm is presented below:

Algorithm 2 `2-swap` search algorithm

Require: A given schedule I and its cost F(I).

Initialize: $improved \leftarrow TRUE$;
while *improved* **do**
  Generate all feasible 2-swaps for each j in S;
  Calculate LB for all moves;
  if $LB \geq 0$ for all feasible moves **then**
    BREAK;
  else
    Make a list of moves with $LB < 0$ in ascending order of LB's and select the first five of them.
    Initialize: $found\_improving\_move \leftarrow FALSE$, $end\_of\_list \leftarrow FALSE$;
    while NOT $found\_improving\_move$ and NOT end_of_list **do**
      Do the next move in the list;
      Solve single machine problem for affected machines;
      New schedule is I';
      if $COST(I') < COST(I)$ **then**
        $I \leftarrow I'$;
        $found\_improving\_move \leftarrow TRUE$, $improved \leftarrow TRUE$;
      end if
    end while
  end if
end while

5.3.1 Improvements by `2-swap` Steps

In this subsection, we calculate a cost lower bound for the applied two swap in a given schedule.

Proposition 5 *(Lower Bound for a* `2-swap`*) For a given schedule let* Λ_{S_1} *and* Λ_{S_2} *be optimal dual prices for subsequence* S_1 *and subsequence* S_2*, respectively. Suppose that* S_1 *is a subsequence on machine* i_1 *and* S_2 *is a subsequence on machine* i_2*. Let* $y_{i_1 j_1}$ *and* $y_{i_2 j_2}$ *be the compression of the jobs* j_1 *and* j_2 *in subsequences* S_1 *and* S_2 *on machines* i_1 *and* i_2*, respectively. Then, a lower bound for the cost change that will result by swapping jobs* j_1 *and* j_2 *between subsequences* S_1 *and* S_2 *is calculated as below:*

$$\begin{aligned} LB(j_1 \leftrightarrow j_2) = \Lambda_{S_1}(p_{i_1 j_1} - y_{i_1 j_1} - p_{i_1 j_2} + \hat{y}_{i_1 j_2}) - c_{i_1 j_1} - f_{i_1 j_1}(y_{i_1 j_1}) + c_{i_1 j_2} + f_{i_1 j_2}(\hat{y}_{i_1 j_2}) \\ + \lambda_{S_2}(p_{i_2 j_2} - y_{i_2 j_2} - p_{i_2 j_1} + \hat{y}_{i_2 j_1}) - c_{i_2 j_2} - f_{i_2 j_2}(y_{i_2 j_2}) + c_{i_2 j_1} + f_{i_2 j_1}(\hat{y}_{i_2 j_1}), \end{aligned}$$

where $\hat{y}_{i_2 j_1} = \min((\frac{\partial f_{i_2 j_1}}{\partial p_{i_2 j_1}})^{-1}(\Lambda_{S_2}), u_{i_2 j_1})$ *and* $\hat{y}_{i_1 j_2} = \min((\frac{\partial f_{i_1 j_2}}{\partial y_{i_1 j_2}})^{-1}(\Lambda_{S_1}), u_{i_1 j_2})$.

Proof. Similar to the proof of Lemma 4. ■

5.4 Improvement Search Algorithm

As we mentioned before, our improvement search algorithm starts with a schedule obtained from the initial schedule algorithm. Therefore, we have an initial schedule in hand to generate new and improved schedules. In order to improve the schedules, improvement search algorithm uses 1-move and 2-swap moves as we clearly presented before. The possibility of the improvement is estimated by calculating cost lower bounds of these moves. Cost lower bound calculations were explained and proved in this Chapter. By comparing jobs' completion times and planned completion times (according to the original schedule), we generate subsequent of jobs and build intervals for each group on each machine. Therefore, each interval includes at least one job and it has a start time which is the start time of the first job in the interval and has an end time which is the completion time of the last job in the interval. Each interval will have its own λ value which helps to calculate cost lower bounds.

The considered job which will be assigned to a new machine by one move or two swap, is tried to be placed on its new machine by considering intervals' start and end times. After placing the considered job to its candidate interval on the new machine, cost lower bound calculations are made. If the cost change lower bound for a move is non-negative, then it is sure that the

move cannot improve the cost. If it is negative, we call the move as a promising move. A promising move may improve the cost, but since we just have a negative lower bound for the cost change, the real cost change after implementing the move may still be positive.

The heuristic uses promising one moves to improve the initial schedule. To do this, it generates all possible 1-moves for this schedule and calculates the cost change lower bound for each of them. Afterwards, cost lower bounds of all promising moves are listed in ascending order and the first twenty possible moves are selected. The heuristic then applies the most promising move, which is the first one, and solves the single machine problem for the affected machines.

If an improvement is achieved, the schedule needs to be updated. Next, new moves are generated for the new schedule. If no improvement is achieved by this move, the heuristic tries the next most promising move, until an improvement is achieved or no promising move is left. When no improvement is possible for the current schedule by using one moves, the heuristic considers two swap moves. It tries to improve the solution by two swap moves in the same way as we did by one moves and stops when no improvement is possible.

As a summary; Improvement Search Algorithm is presented below:

Algorithm 3 Improvement Search Algorithm

Require: A given schedule O and a disruption on one of the machines.

Apply Initial Schedule Algorithm to find initial schedule I.

repeat

Apply 1-move Algorithm.

Apply 2-swap Algorithm.

Report solution achieved.

until $no\_possible\_update = TRUE$. //$T_i = E_i$ for all i.

CHAPTER 6

NUMERICAL EXAMPLE

In this Chapter, we present a numerical example for our rescheduling problem. This is a parallel machine rescheduling problem arising due to a breakdown on one of the machines. As we mentioned before, the problem is to reschedule a set of jobs on non-identical parallel machines where each job has different upper bound value on compression and processing time data on different machines. When there is a machine breakdown on one of the machines, we illustrate how our procedure can be used to remedy this unavailability period using a Gantt Chart representation for each progress during the rescheduling process throughout this Chapter.

We present our approach through the following 2 machine and 15 job problem instance. The design attributes for the problem are the allowable number of disrupted jobs K and the disruption length DL which are specified as 3 and 2 respectively. Besides; the processing times, compression upper bound values and compression cost functions of each job on each machine are presented in Table 6.1 and Table 6.2 respectively.

Table 6.1: Data for the example problem on machine 1.

Machine 1											
Job j	p_{1j}	y_{1j}	$f(y_{1j})$	Job j	p_{1j}	y_{1j}	$f(y_{1j})$	Job j	p_{1j}	y_{1j}	$f(y_{1j})$
1	1.80	1.10	$1.9y^{1.5}$	6	1.50	0.90	$3.1y^{2.5}$	11	1.80	1.20	$2.4y^{2.5}$
2	2.40	1.70	$1.2y^{2.5}$	7	2.30	1.90	$1.9y^{2}$	12	1.50	1.30	$2.5y^{1}$
3	2.60	1.40	$2.5y^{2}$	8	1.20	0.90	$2.8y^{1.5}$	13	1.50	0.70	$1.5y^{4}$
4	1.70	1.20	$1.9y^{1.5}$	9	1.40	1.00	$2y^{3}$	14	1.60	0.90	$1.1y^{4}$
5	2.30	1.90	$2.8y^{1}$	10	1.70	1.50	$1.9y^{2.5}$	15	1.20	0.90	$2.3y^{2.5}$

Table 6.2: Data for the example problem on machine 2.

Machine 2											
Job j	p_{2j}	y_{2j}	$f(y_{2j})$	Job j	p_{2j}	y_{2j}	$f(y_{2j})$	Job j	p_{2j}	y_{2j}	$f(y_{2j})$
1	1.90	1.10	$3y^{1}$	6	1.00	0.70	$1.4y^{2}$	11	1.20	0.60	$1.7y^{4}$
2	2.90	2.50	$1.4y^{2}$	7	1.20	0.80	$1.2y^{2.5}$	12	2.00	1.70	$1.9y^{2}$
3	2.00	1.70	$2y^{1.5}$	8	1.50	0.70	$2y^{4}$	13	1.10	0.70	$1.3y^{2}$
4	2.80	1.40	$2y^{4}$	9	2.10	1.70	$2.1y^{2.5}$	14	2.00	1.70	$2y^{3}$
5	2.90	1.90	$2.3y^{2.5}$	10	2.70	2.20	$2y^{3}$	15	2.30	1.60	$2.2y^{4}$

The example starts with an original schedule which is the schedule used before the disruption occurred and it is presented in Fig. 6.1. In order to generate this schedule, a machine-job assignment problem with cost minimization objective is solved in the beginning. Thus, optimal machine-job assignments are calculated with optimum compression value. Afterwards, Shortest Processing Time (SPT) rule is applied for sorting the jobs by providing minimum total completion time. So that the job with minimum processing time will be chosen first for the processing.

Thus, an original schedule is generated. The start time, completion time and compression amount of each job on each machine according to the original schedule are shown in Table 6.3 and Table 6.4 respectively. The makespan of the schedule is calculated as 9 for this

numerical example.

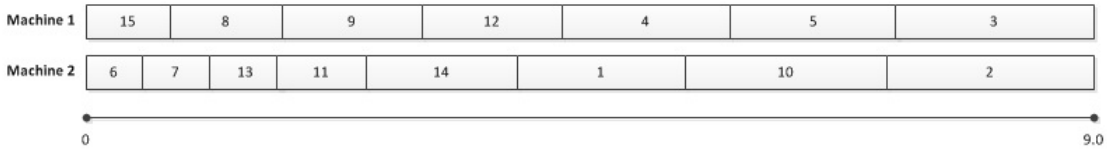

Figure 6.1: The Original Schedule

Table 6.3: The original schedule information for machine 1.

Machine 1				
Job j	Start Time	End Time	Processing Time	Compression Amount
15	0.00	0.67	1.20	0.53
8	0.67	1.49	1.20	0.38
9	1.49	2.39	1.40	0.50
12	2.39	3.55	1.50	0.34
4	3.55	5.03	1.70	0.23
5	5.03	6.94	2.30	0.38
3	6.94	9.00	2.60	0.54

Table 6.4: The original schedule information for machine 2.

Machine 2				
Job j	Start Time	End Time	Processing Time	Compression Amount
6	0.00	0.35	1.00	0.65
7	0.35	0.75	1.20	0.80
13	0.75	1.15	1.10	0.70
11	1.15	1.76	1.20	0.59
14	1.76	3.18	2.00	0.59
1	3.18	4.63	1.90	0.44
10	4.63	6.75	2.70	0.59
2	6.75	9.00	2.90	0.65

In this numerical example, a breakdown occurs during the processing of job 9 at time 1.5 on machine 1 as shown in Fig. 6.2. In practice, the exact time of a machine breakdown is not known, but end time can be determined right after its occurrence. Therefore, we assume that the breakdown time is not known a priori, but immediately after the event occurs the down duration can be determined. The disruption length is calculated as 2 for this numerical example. At this point, the jobs 15 and 8 have been completed on machine 1 and the jobs 6, 7 and 13 have been completed on machine 2. The job 11 is in process on machine 2 at the time of the breakdown. After the processing of job 11, machine 2 will be available for rescheduling.

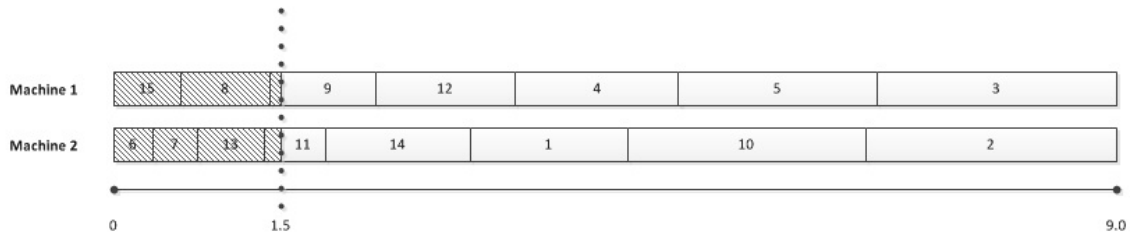

Figure 6.2: Schedule after disruption and prior to the application of rescheduling.

After the disruption, the original schedule is no longer executable. At the time of breakdown, there are jobs which are not processed yet and other jobs which are under process. Since

machine 1 is the disrupted machine, job 9 which is interrupted at the time of breakdown and it needs to be reprocessed. On the other hand, job 11 is under process on machine 2 at the time of breakdown. Since machine 2 is not disrupted, job 11 can complete processing without any interruption. Therefore, affected jobs from disruption are the ones which are not processed yet and the one which is interrupted during its processing at the time of breakdown. Our task is to reschedule those affected jobs by taking into account their planned completion times according to the original schedule and available machining time data which are presented in Table 6.5 and Table 6.6 respectively.

Table 6.5: Planned completion times of affected jobs at the time of the breakdown.

Job j	Planned completion time
9	2.39
12	3.55
4	5.03
5	6.94
3	9.00
14	3.18
1	4.63
10	6.75
2	9.00

Table 6.6: Available machining time capacity of each machine.

Machine i	Available Machining Time
1	5.50
2	7.24

As seen in Fig. 6.2 above, there are 9 jobs to be rescheduled. We know that the number of allowable disrupted jobs is limited and equals to 3 for this numerical example. According to the given information about the rescheduling environment, both processing time and machine-job reallocation decisions will be made.

As mentioned before, a mathematical model is provided at first for the problem and an improvement search algorithm is presented afterwards to generate a feasible schedule as soon as possible. Design parameters and affected jobs' information are provided for the mathematical model and the results obtained by solving recheduling mathematical model are presented in Table 6.7 and Table 6.8 respectively and illustrated in Fig. 6.3 as well. Jobs which are assigned as disrupted jobs are striped in Fig. 6.3. The Z values of these jobs are equal to 1 in Table 6.7 and Table 6.8.

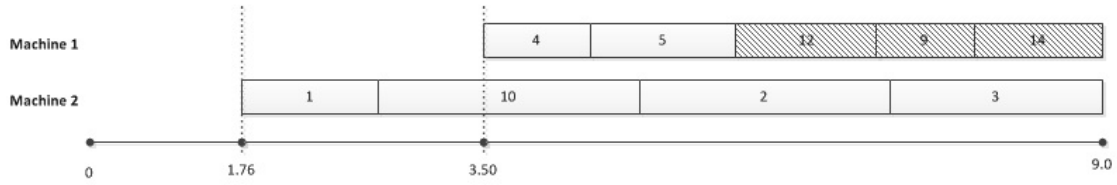

Figure 6.3: Schedule after disruption found by solving Rescheduling Mathematical Model.

Table 6.7: Rescheduling Mathematical Model Results

Machine 1					
Job j	Start Time	End Time	Processing Time	Compression Value	Z
4	3.50	4.83	1.70	0.37	0
5	4.83	6.64	2.30	0.49	0
12	6.64	7.68	1.50	0.46	1
9	7.68	8.30	1.40	0.78	1
14	8.30	9.00	1.60	0.90	1

Table 6.8: Rescheduling Mathematical Model Results

Machine 2					
Job j	Start Time	End Time	Processing Time	Compression Value	Z
1	1.76	3.20	1.90	0.46	0
10	3.20	5.25	2.70	0.65	0
2	5.25	7.32	2.90	0.83	0
3	7.32	9.00	2.00	0.32	0

According to the solution of the rescheduling mathematical model; jobs 9, 12 and 14 are selected to be the disrupted jobs. Jobs 4, 5, 1, 10, 2 complete processing before their planned completion times and job 3 complete processing strictly at its planned completion time. The manufacturing cost of rescheduled jobs in the original schedule is calculated as 24.26 and the rescheduling cost of those jobs is calculated as 27.96. Thus, the additional manufacturing cost is 3.70, which is obtained as the optimal objective function value for this problem instance.

As it has been mentioned before, solving problems rapidly is critical in rescheduling. In order to create best or close to best schedules in reasonable durations, an improvement search heuristic is provided. In the execution stage of the improvement search algorithm, the proposed optimality properties assist in making recheduling decisions.

At the begining, an initial schedule is needed to apply the improvement search heuristic. In order to generate an inital schedule, an initial schedule algorithm is provided. At first, a list of affected jobs, which are sorted in ascending order according to their planned completion times, is required. The related list is shown in Table 6.9 as follows:

Table 6.9: The list of affected jobs according to their planned completion times.

Job j	Planned Completion Time
9	2.39
14	3.18
12	3.55
1	4.63
4	5.03
10	6.75
5	6.94
3	9.00
2	9.00

According to the initial schedule algorithm, it is assumed that all machines are empty at the beginning and the algorithm starts scheduling with the first job in the list and then continues scheduling with the other jobs respectively.

This scheduling process consists of two alternative assignment conditions for each job on each machine. The first condition is that the considered job will complete processing at its planned completion time at the very latest. The second condition is that the considered job will be assigned as a disrupted job which means that it will complete processing after its planned completion time.

Initially, the first condition is applied to the considered job on each candidate machine for the assignment. For each assignment, single machine subproblem is solved and the objective function value is kept. If there exists no feasible solution by applying the first condition and the allowable number of disrupted jobs is greater than zero, the second condition is applied to the same job. For each assignment, single machine subproblem is solved again and the objective function value is kept as well. Afterwards, objective function values are listed in ascending order and the machine having the minimum objective function value is selected and the job is assigned to that machine. We repeat this process until all jobs are assigned to the machines. Initial solution algorithm either ends with a feasible schedule for the problem or fails to find a feasible schedule and stops. If initial schedule algorithm finds a feasible

schedule, this initial schedule will be considered as a basis for applying the improvement search heuristic.

The initial schedule generated for this numerical example is presented in Fig. 6.4 and the results are presented in Table 6.10 and Table 6.11, respectively.

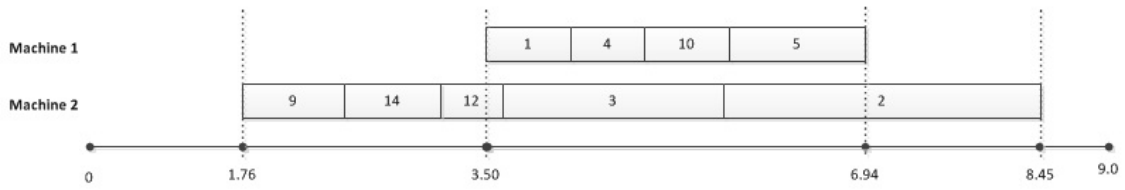

Figure 6.4: The Initial Schedule

Table 6.10: Initial Schedule Information

Machine 1					
Job j	Start Time	End Time	Processing Time	Compression Valuet	Z
1	3.50	4.23	1.80	1.07	0
4	4.23	4.85	1.70	1.07	0
10	4.85	5.48	1.70	1.07	0
5	5.48	6.94	2.30	0.84	0

Table 6.11: Initial Schedule Information

Machine 2					
Job j	Start Time	End Time	Processing Time	Compression Value	Z
9	1.76	2.39	2.10	1.47	0
14	2.39	3.04	2.00	1.36	0
12	3.04	3.55	2.00	1.49	0
3	3.55	5.55	2.00	0.00	0
2	5.55	8.45	2.90	0.00	0

The additional manufacturing cost of initial schedule is calculated as 20.28. As mentioned

in Chapter 4, intervals are determined for the considered jobs on each machine. Afterwards, the intervals' λ values, which are the marginal cost of compression of jobs lying in the same interval, are calculated. The proposed heuristic applies two different moves to improve the initial schedule by taking machine intervals into account.

The first applied move is called one move. We apply one move by shifting a job from its current interval on its current machine to another interval on another machine by considering the assignment condition of the considered job. If the job is assigned as a disrupted job (the first condition) on its candidate machine, its planned completion time will be updated with the makespan. If the job is not assigned as a disrupted job (the second condition) on its candidate machine, there is no need to update its planned completion time. Afterwards, by considering each interval's start and end times on the candidate machine, the considered job is placed on the interval in which the job's planned completion time lies. After placing the considered job on its candidate interval, related cost lower bound calculations are made with the help of intervals' lambda values.

If the cost change lower bound for a move is non-negative, then it is certain that the move cannot improve the cost. If it is negative, the move is called as a promising move. A promising move may improve the cost, but since we just have a negative lower bound for the cost change, the real cost change after implementing the move may still be positive.

The second move is called two swap, which is to exchange two jobs on each other's machine. It tries to improve the schedule in the same way as one move steps do. If no improvement is possible, it stops and the improvement search heuristic is completed. A presentation of machines' intervals with the information of related jobs and their calculated lambda values are shown in Table 6.12 below.

Table 6.12: Information about the intervals on machine 1.

Machine 1				
Interval i	The Jobs in Interval i	Interval λ Value	Interval Start Time	Interval End Time
1	1, 4, 10, 5	4.05	3.50	6.94

Table 6.13: Information about the intervals on machine 2.

Machine 2				
Interval i	The Jobs in Interval i	Interval λ Value	Interval Start Time	Interval End Time
1	9	6.42	1.76	2.39
2	14, 12	2.71	2.39	3.55
3	3, 2	0.02	3.55	8.45

The heuristic uses promising one moves to improve the initial schedule. To do this, it generates all possible one moves for the schedule and calculates the cost change lower bound for each of them. Afterwards, cost lower bounds of all promising moves are listed in ascending order and the first five calculations are selected. The list including the first five promising one moves and their calculated cost change lower bound values and assignment conditions for the considered job are presented in Table 6.14 as follows:

Table 6.14: List of promising one moves and their calculated cost change lower bound values.

Job i	Current Machine	Candidate Machine	Calculated Cost Lower Bounds	Assignment Condition
9	0	1	-4.47	First
9	0	1	-4.47	Second
8	1	0	-4.42	First
8	1	0	-4.42	Second
3	0	1	-1.47	First

The heuristic then applies the most promising move, which is the first one, and solves the single machine problem for the affected machines. If an improvement is achieved, the schedule needs to be updated and new one moves are generated for the new schedule. If no improvement is achieved by one move, the heuristic tries the next most promising move until an improvement is achieved or no promising move is left. When no improvement is possible for the existing schedule by using one moves, the heuristic considers two swap moves the same

way as one moves. For this numerical example, none of the possible one moves improve the schedule. For this reason, the heuristic tries to improve the solution by two swap moves and stops when no improvement is possible.

The list including the first five promising two swap moves and their calculated cost change lower bound values and assignment conditions for each job which is referred as *AC* are presented in Table 6.15 as follows:

Table 6.15: List of promising two swap moves and their calculated cost change lower bound values.

Job 1	Job 2	Machine 1	Machine 2	Calculated Cost Lower Bounds	AC for Job 1	AC for Job 2
10	9	1	2	-12.78	Second	Second
10	9	1	2	-12.78	First	Second
1	9	1	2	-12.20	First	First
1	9	1	2	-12.20	Second	First
1	3	1	2	-0.04	First	Second

The heuristic improved the initial schedule by applying the first possible two swap move. Figure 6.5 illustrates the considered two swap move to clarify the procedure. The dotted jobs 9 and 10 in the Figure 6.5 are the jobs which will be swapped in the schedule. Both of these jobs will be assigned as disrupted jobs. Therefore, the planned completion time of these jobs will be updated as 9.00 which is the makespan value of the schedule. The completion time of job 10 is greater than the start time of the second interval on machine 2 and since this interval is the last interval on machine 2, job 10 will be assigned to that interval. The completion time of job 9 is greater than the start time of the first interval on machine 1 and since this interval is the last interval on machine 1, job 9 will be assigned to that interval.

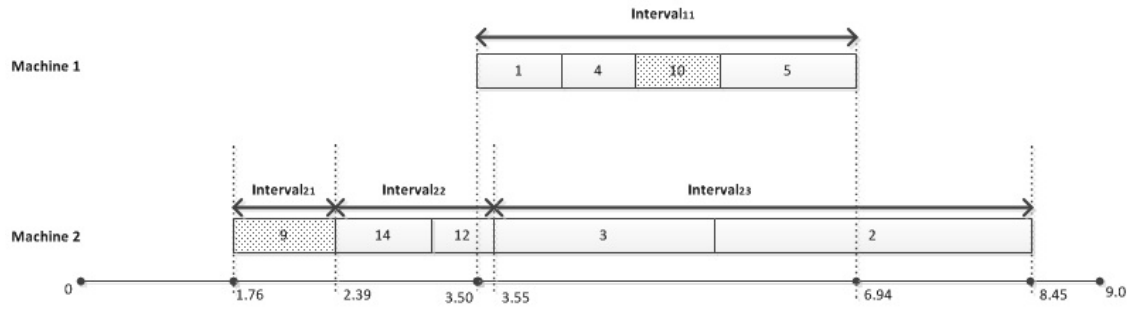

Figure 6.5: The presentation of the considered two swap move.

After assigning each job, the single machine subproblem for the affected machines is solved. According to the solution of the single machine subproblem, the additional manufacturing cost is calculated as 8.09 which is 12.19 less than the initial schedule's additional manufacturing cost value. An improvement is achieved and the schedule needs to be updated. The results obtained by solving improvement search heuristic are presented in Table 6.16 and Table 6.17 respectively and illustrated in Fig. 6.6 as well. The jobs which are assigned as disrupted are job 9 and 10. They take 1 for the Z values in Table 6.16 and Table 6.17.

Table 6.16: Improvement search heuristic results.

Machine 1					
Job j	Start Time	End Time	Processing Time	Compression Value	Z
1	3.50	4.31	1.90	0.99	0
4	4.31	5.03	2.70	0.99	0
5	5.03	6.94	2.90	0.38	0
9	6.94	8.34	2.00	0.00	1

Table 6.17: Improvement search heuristic results.

Machine 2					
Job j	Start Time	End Time	Processing Time	Compression Value	Z
14	1.76	2.69	2.00	1.07	0
12	2.69	3.55	2.00	1.14	0
3	3.55	5.19	2.00	0.36	0
2	5.19	7.02	2.90	1.07	0
10	7.02	9.00	2.70	0.72	1

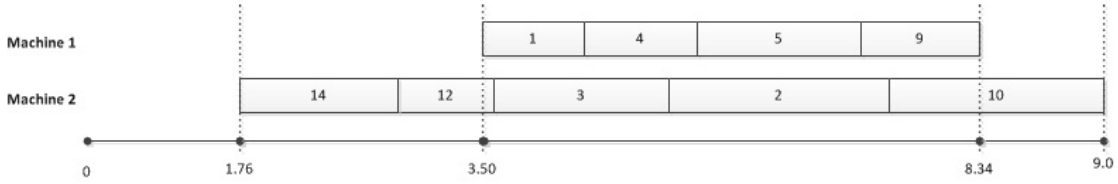

Figure 6.6: Schedule found by applying two swap move.

For the next step; new intervals, interval lambdas and two swap moves are generated for the improved schedule. New machine intervals with the information of related jobs and their calculated lambda values are shown in Table 6.18 and Table 6.19 below.

Table 6.18: Information about the intervals on machine 1.

Machine 1				
Interval i	The Jobs in Interval i	Interval Lambda Value	Interval Start Time	Interval End Time
1	1, 4	3.76	3.50	5.03
2	5	1.00	5.03	6.94
3	9	0.00	6.94	8.34

Table 6.19: Information about the intervals on machine 2.

Machine 2				
Interval *i*	The Jobs in Interval *i*	Interval Lambda Value	Interval Start Time	Interval End Time
1	14, 12	2.14	1.76	3.55
2	3, 2, 10	1.44	3.55	9.00

The list including the first five promising two swap moves and their calculated cost change lower bound values and assignment conditions for each job, which is referred as AC, are presented in Table 6.20 as follows:

Table 6.20: List of promising two swap moves and their calculated cost change lower bound values.

Job 1	Job 2	Mc 1	Mc 2	Calculated Cost Lower Bounds	AC for Job 1	AC for Job 2
1	3	1	2	-5.38	First	Second
1	2	1	2	-4.65	Second	First
1	2	1	2	-4.65	First	First
1	3	1	2	-4.38	Second	First
4	12	1	2	-0.52	First	Second

The heuristic improved the schedule further by the means of minimizing the additional manufacturing cost by applying the first possible two swap move. Figure 6.7 illustrates the considered two swap move which swaps job 1 and 3 on each other's machine. Job 3 will be assigned as a disrupted job. Then, updated planned completion time of job 3 is greater than the start time of the third interval on machine 1 and since this interval is the last interval on machine 1, job 3 will be assigned to that interval. Job 1 will be assigned according to the first condition. The planned completion time of job 1 is 4.63 and it is between the start and completion time of the second interval on machine 2. In addition, the planned completion time of job 1 is less than the planned completion time of job 2. Therefore, job 1 will be assigned to that interval

before job 2.

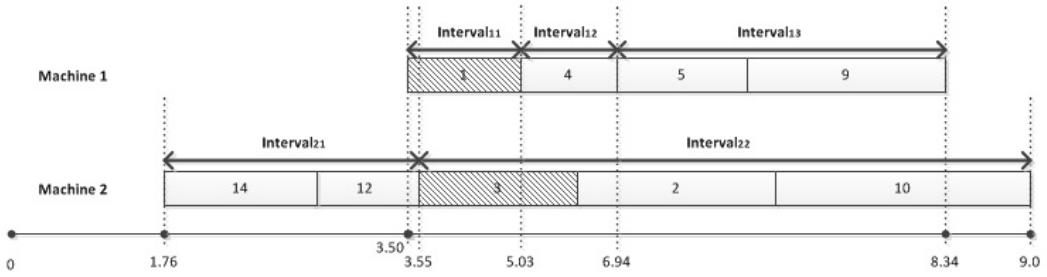

Figure 6.7: The presentation of the schedule after applying two swap move.

After solving the single machine problem for the affected machines, the additional manufacturing cost is calculated as 3.83 which is 4,26 less than previous additional manufacturing cost value. Improvement is achieved by assigning job 3 as a disrupted job on machine 1. Therefore, the schedule needs to be updated. The results obtained by solving improvement search heuristic are presented in Table 6.21 and Table 6.22 respectively and also illustrated in Fig. 6.9. According to the schedule, the disrupted jobs are the job 3, 9 and 10. They take 1 for the Z values in Table 6.21 and Table 6.22.

Table 6.21: Improvement Search Heuristic Results

Machine 1					
Job j	Start Time	End Time	Processing Time	Compression Value	Z
4	3.50	4.80	1.70	0.40	0
5	4.80	6.59	2.30	0.51	0
9	6.59	7.16	1.40	0.83	1
3	7.16	9.00	2.60	0.76	1

Table 6.22: Improvement Search Heuristic Results

Machine 2					
Job j	Start Time	End Time	Processing Time	Compression Value	Z
14	1.76	2.60	2.00	1.16	0
12	2.60	3.36	2.00	1.25	0
1	3.36	4.63	1.90	0.62	0
2	4.63	6.89	2.90	0.65	0
10	6.89	9.00	2.70	0.59	1

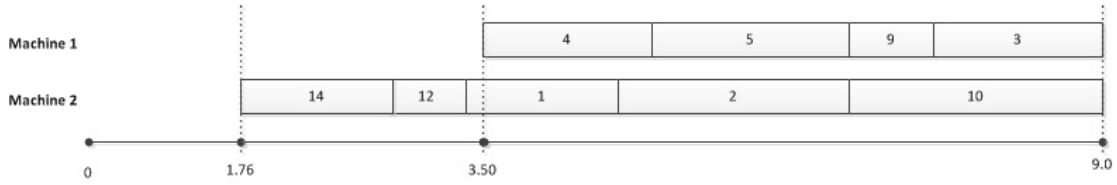

Figure 6.8: The presentation of the schedule after applying two swap move.

For the next step, new two swap moves are generated for the improved schedule. The list including the first five promising two swap moves and calculated cost change lower bound values and assignment conditions for each job which is referred as AC are presented in Table 6.23 as follows:

Table 6.23: Possible Two Swap Moves

Job 1	Job 2	Mc 1	Mc 2	Calculated Cost Lower Bounds	AC for Job 1	AC for Job 2
2	1	1	2	-1.54	First	First
2	13	1	2	-0.83	First	First
2	0	1	2	-0.69	First	First
2	9	1	2	-0.20	First	Second
3	1	1	2	-0.05	First	First

None of the possible two swap moves listed above improve the schedule. For this reason, the heuristic stops.

As a conclusion, the final schedule generated from the initial schedule by applying improvement search heuristic is presented in Fig. 6.9. The calculated additional manufacturing cost by the rescheduling mathematical model and the improvement search heuristic is 3.69 and 3.83 respectively. According to the calculations, rescheduling mathematical model solution has the minimum additional manufacturing cost for the problem. Improvement search heuristic solution is close to the optimal solution. Therefore, applying improvement search heuristic is a good approach in order to find a feasible schedule in reasonable durations.

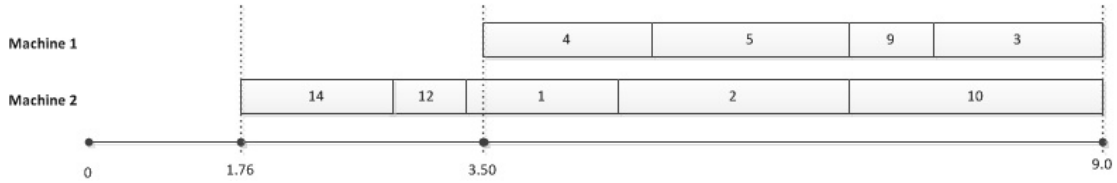

Figure 6.9: The presentation of the schedule after applying improvement search heuristic.

CHAPTER 7

COMPUTATIONAL STUDY

In this Chapter, we present the results of our computational study for testing the efficiency of proposed rescheduling mathematical model and improvement search heuristic. We compared the computation time and solution quality of the proposed model and heuristic on a set of randomly generated test problems. Design parameters, their explanations and values are presented in Table 7.1.

Table 7.1: Design Parameters

Design Parameters		
N	Number of jobs.	30,50,100
M	Number of machines.	2
ρ	Capacity factor used to generate the makespan of the schedule.	0.40,0.45
ld	The length of disruption.	1.5,2.0
c_{ij}	Cost of job j on machine i.	Uniform[2.0, 4.0]
k_{ij}	The coefficient of the compression cost function.	Uniform[1.1, 3.1]
a_{ij}/b_{ij}	The coefficient of the compression cost function.	Uniform[1.0, 4.1]
p_{ij}	Processing time of job j on machine i.	Uniform[1.0, 3.0]
u_{ij}	The compression upper bound.	p_{ij}*Uniform[0.5, 0.9]

We set the makespan of the schedule equal to:

$$C_{max} = \rho * \sum_{i \in I} \sum_{j \in J} (\mathrm{p}_{ij})/\mathrm{m}$$

For each problem instance, resheduling mathematical model is run initially. Afterwards, improvement search heuristic algorithm is applied. Improvement search heuristic was coded in C++ language. All codes were run on the operating system Windows 7 on a PC HP ProBook 6550b with Intel(R) Core(TM) i7 CPU and 4GB memory. All experiments were perfomed using ILOG CPLEX Version 12.1 with a 600 CPU seconds time limit. For each experimental setting, 5 replications were taken.

For rescheduling to be required, there has to occur a disruption on a machine which makes the original schedule nonexecutable. However, there does not exist any original schedule at the beginning. So an original schedule has to be generated initially. In order to construct an original schedule, the machine-job assignment problem is solved in the beginning. It aims to minimize total manufacturing cost subject to given capacity for the makespan of the schedule. Afterwards, the allocated jobs are sequenced on each machine by using the shortest processing time (SPT) rule, which gives the minimum total completion time for a given set of jobs on a machine. After constructing an original schedule, a breakdown on the schedule is generated by randomly selecting a machine and a job. So that a breakdown occurs during the execution of the selected job and lasts for a specified duration. Rescheduling problem includes the interrupted job on the breakdown machine and the jobs planned to be started for processing at the time of breakdown or later in the original schedule.

In the following paragraphs, the results of rescheduling mathematical model and improvement search heuristic are presented. These results consist of average additional manufacturing cost values, average CPU times and average number of disrupted jobs of five replications for each problem setting. The total number of optimal, feasible and infeasible solution for five replications is also presented. In addition to these information, the percentage cost gap between the results of rescheduling mathematical model and improvement search heuristic are shown as well.

In Tables 7.2, 7.3, 7.4 and 7.5, the computational results or rescheduling mathematical model and improvement search heuristic for the problem instance which has 30 jobs being rescheduled on 2 machines are presented. As seen in the problem instance of 30 jobs, 2 machines, 3 allowable disrupted jobs, ro having a value of 0,40; the additional manufacturing cost increases when the disruption length is increased from 1.5 to 2.0. Likewise, in the problem instances having the same values of jobs, machines, allowable number of disrupted jobs and ro coefficients; the additional manufacturing cost always rises with the increase in disruption length, since the available machining time capacity decreases and the schedule becomes tighter.

In the problem instance of 30 jobs, 2 machines, 3 allowable disrupted jobs and 1.5 disruption length; the additional manufacturing cost decreases when the capacity factor ρ is increased from 0.40 to 0.45. This situation can be observed in the problem instances having the same values of jobs, machines, allowable number of disrupted jobs and disruption length. Since the ro coefficient is directly proportional to the makespan value, when it increases, available machining time capacity increases and the schedule becomes looser.

In the problem instance of 30 jobs, 2 machines, ro having a value of 0,40 and 1.5 disruption length; the additional manufacturing cost decreases when the allowable number of disrupted jobs is increased from 3 to 4. Since with the increase in number of allowable disrupted jobs, the requirement for compressing jobs diminishes. Thus additional manufacturing cost decreases.

Table 7.2: Computational results of rescheduling mathematical model for the problem instance of 30 jobs and 2 machines.

Disruption Length = 1.5										
Design Parameters					**Rescheduling Mathematical Model**					
N	**M**	**K**	ρ	**Number of affected jobs**	**Cost**	**CPU**	**GAP**	**Optimal**	**Feasible**	**Infeasible**
30	2	3	0.40	19	2.89	600.05	10.61	-	4	1
30	2	3	0.45	20	1.73	600.04	4.84	-	5	-
30	2	4	0.40	19	3.19	600.02	12.34	-	4	1
30	2	4	0.45	20	1.75	600.06	5.19	-	5	-

Table 7.3: Computational results of improvement search heuristic for the problem instance of 30 jobs and 2 machines.

Disruption Length = 1.5							
Design Parameters					**Improvement Search Heuristic**		
N	**M**	**K**	ρ	**Number of affected jobs**	**Cost**	**CPU**	**The Percentage Cost GAP**
30	2	3	0.40	19	3.05	8.39	5.54
30	2	3	0.45	20	1.92	22.66	10.98
30	2	4	0.40	19	2.64	18.18	-17.24
30	2	4	0.45	20	1.95	20.59	11.43

Table 7.4: Computational results of rescheduling mathematical model for the problem instance of 30 jobs and 2 machines.

Disruption Length = 2.0										
Design Parameters					**Rescheduling Mathematical Model**					
N	**M**	**K**	ρ	**Number of affected jobs**	**Cost**	**CPU**	**GAP**	**Optimal**	**Feasible**	**Infeasible**
30	2	3	0.40	19	4.87	600.03	13.20	-	4	1
30	2	3	0.45	20	3.14	600.03	6.00	-	5	-
30	2	4	0.40	19	4.24	600.06	12.97	-	4	1
30	2	4	0.45	20	3.07	600.03	6.49	-	5	-

Table 7.5: Computational results of improvement search heuristic for the problem instance of 30 jobs and 2 machines.

Disruption Length = 2.0							
Design Parameters					**Improvement Search Heuristic**		
N	**M**	**K**	ρ	**Number of affected jobs**	**Cost**	**CPU**	**The Percentage Cost GAP**
30	2	3	0.40	19	5.10	19.26	4.72
30	2	3	0.45	20	3.26	20.57	3.82
30	2	4	0.40	19	4.36	17.69	2.83
30	2	4	0.45	20	3.06	16.73	-0.33

The observations given above are also valid for the results of rescheduling mathematical model and improvement search heuristic. However, there are some exceptions in the results for the improvement search algorithm part. Design parameters are playing an important role while generating an initial schedule, one moves and two swaps during the improvement search heuristic.

For some problem instances, the additional manufacturing cost of the schedule having 3 disrupted jobs has been calculated less than the additional manufacturing cost of the schedule having 4 disrupted jobs after applying improvement search heuristic. This condition has arisen since generated possible two swaps set were different for the two schedules. Considered schedules have remained same by the means of job sequences and job compressions until three jobs have been assigned as disrupted jobs on each schedule. Afterwards, new possible swaps have been generated in order to improve the existing ones. Assigning 1 more job as a disrupted job can be considered in the schedule which allows 4 disrupted jobs. However, the schedule which allows 3 disrupted jobs cannot consider assigning 1 more job as a disrupted job. Thus, instead of this alternative, the schedule considers different improving moves. Therefore, improving moves for each schedule were appeared to be different and naturally the improvements achieved by applying these moves have had different effects on each schedule. As a result; if the cost improvements achieved in the schedule which allows 4 disrupted jobs are less than the cost improvements achieved in the schedule which allows 3 disrupted jobs, the schedule which allows 3 disrupted jobs will have the minimum additional manufacturing cost with respect to the schedule which allows 4 disrupted jobs.

In addition to this observation mentioned above, the additional manufacturing cost of a schedule, which has smaller capacity factor (ρ), has been calculated less than the additional manufacturing cost of another schedule which has higher capacity factor ρ according to the other schedule's capacity factor. This condition has arisen since initial schedules have been generated differently for each problem setting. Therefore, affected jobs from the disruption, their total number and improving moves have differantiated from one case to another.

Furthermore, it is seen for some problem instances that the additional manufacturing cost of a schedule has not increased when the disruption length has been increased from 1.5 to 2.0. We have observed this situation since generated initial schedules have been different for each schedule. The initial schedule, generated for the environment of having 1.5 disruption length, included 1 disrupted job while the initial schedule, generated for the environment of having 2.0 disruption length included 2 disrupted jobs. Therefore, the initial schedules and generated improving moves to be applied have been different.

In this thesis, it is aimed to study the trade off between the additional manufacturing cost and the number of allowable disrupted jobs. This relation for the problem instance of 30 jobs and 2 machines is illustrated in Figure 7.1 and Figure 7.2 for the results of rescheduling mathematical model and improvement search algorithm, respectively.

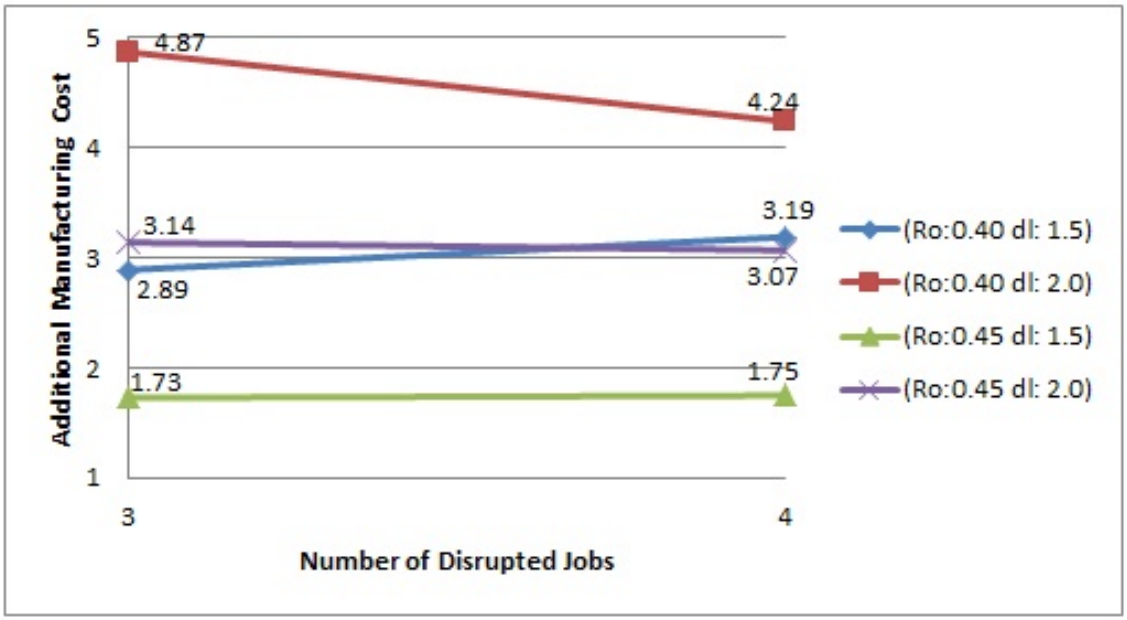

Figure 7.1: The relationship of conflicting objectives obtained by applying rescheduling mathematical model for the problem instance of 30 jobs and 2 machines.

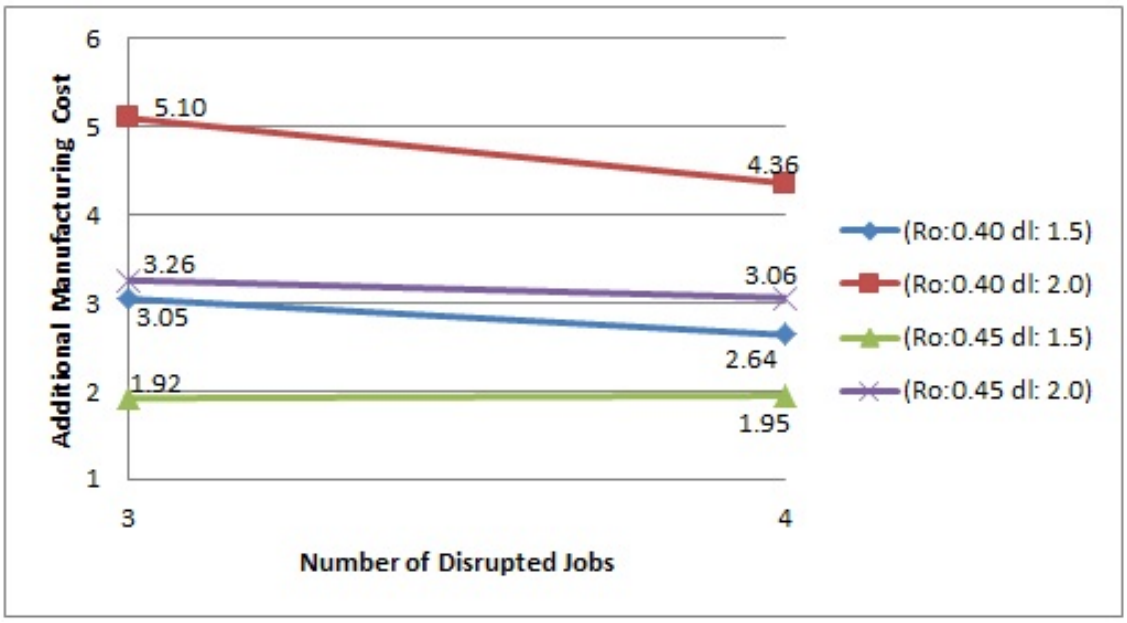

Figure 7.2: The relationship of conflicting objectives obtained by applying improvement search heuristic for the problem instance of 30 jobs and 2 machines.

Beside of this trade off relation, the solution quality of rescheduling mathematical model and improvement search heuristic is compared as well. The comparison considers both CPU times and additional manufacturing cost values. As can be seen from the Tables 7.2, 7.3, 7.4 and 7.5 above, rescheduling mathematical model has much higher CPU times than the improvement search heuristic. Additionally, improvement search heuristic is able to generate additional manufacturing cost close to and also better than the result of rescheduling mathematical model with a reasonable gap value.

In Tables 7.6, 7.7, 7.8 and 7.9 the computational results for the problem instance which has 50 jobs being rescheduled on 2 machines are shown. The observations presented for the problem instance of 30 jobs are also valid for this problem instance.

Table 7.6: Computational results of rescheduling mathematical model for the problem instance of 50 jobs and 2 machines.

Disruption Length = 1.5										
Design Parameters					**Rescheduling Mathematical Model**					
N	**M**	**K**	ρ	**Number of affected jobs**	**Cost**	**CPU**	**GAP**	**Optimal**	**Feasible**	**Infeasible**
50	2	3	0.40	38	6.42	600.56	10.54	-	5	-
50	2	3	0.45	39	4.08	600.25	5.66	-	5	-
50	2	4	0.40	38	4.31	600.15	5.60	-	5	-
50	2	4	0.45	39	3.20	600.18	4.78	-	5	-

Table 7.7: Computational results of improvement search heuristic for the problem instance of 50 jobs and 2 machines.

Disruption Length = 1.5							
Design Parameters					**Improvement Search Heuristic**		
N	**M**	**K**	ρ	**Number of affected jobs**	**Cost**	**CPU**	**The Percentage Cost GAP**
50	2	3	0.40	38	4.65	36.02	-27.57
50	2	3	0.45	39	4.29	34.69	5.15
50	2	4	0.40	38	4.49	34.47	4.18
50	2	4	0.45	39	2.93	32.59	-8.44

Table 7.8: Computational results of rescheduling mathematical model for the problem instance of 50 jobs and 2 machines.

Disruption Length = 2.0										
Design Parameters					**Rescheduling Mathematical Model**					
N	**M**	**K**	ρ	**Number of affected jobs**	**Cost**	**CPU**	**GAP**	**Optimal**	**Feasible**	**Infeasible**
50	2	3	0.40	38	6.88	600.07	7.85	-	4	1
50	2	3	0.45	39	5.18	600.20	4.91	-	5	-
50	2	4	0.40	38	7.50	600.16	10.06	-	5	-
50	2	4	0.45	39	5.65	600.11	6.33	-	5	-

Table 7.9: Computational results of improvement search heuristic for the problem instance of 50 jobs and 2 machines.

Disruption Length = 2.0							
Design Parameters					**Improvement Search Heuristic**		
N	**M**	**K**	ρ	**Number of affected jobs**	**Cost**	**CPU**	**The Percentage Cost GAP**
50	2	3	0.40	38	5.88	9.27	-14.53
50	2	3	0.45	39	5.71	9.67	10.23
50	2	4	0.40	38	6.71	9.69	-10.53
50	2	4	0.45	39	5.05	10.51	-10.62

The relation between the additional manufacturing cost and the number of allowable disrupted jobs is shown in Figure 7.3 and 7.4 for the rescheduling mathematical model and the improvement search algorithm for the problem instance of having 50 jobs, respectively. As can be seen from the Figure 3, the additional manufacturing cost increases when the allowable number of disrupted jobs is increased from 3 to 4 for two problem settings. As we mentioned before, the solutions of rescheduling mathematical model are obtained as feasible. Therefore, when we increase the time limitations our rescheduling model may find the solutions which will be consistent with our previous observations.

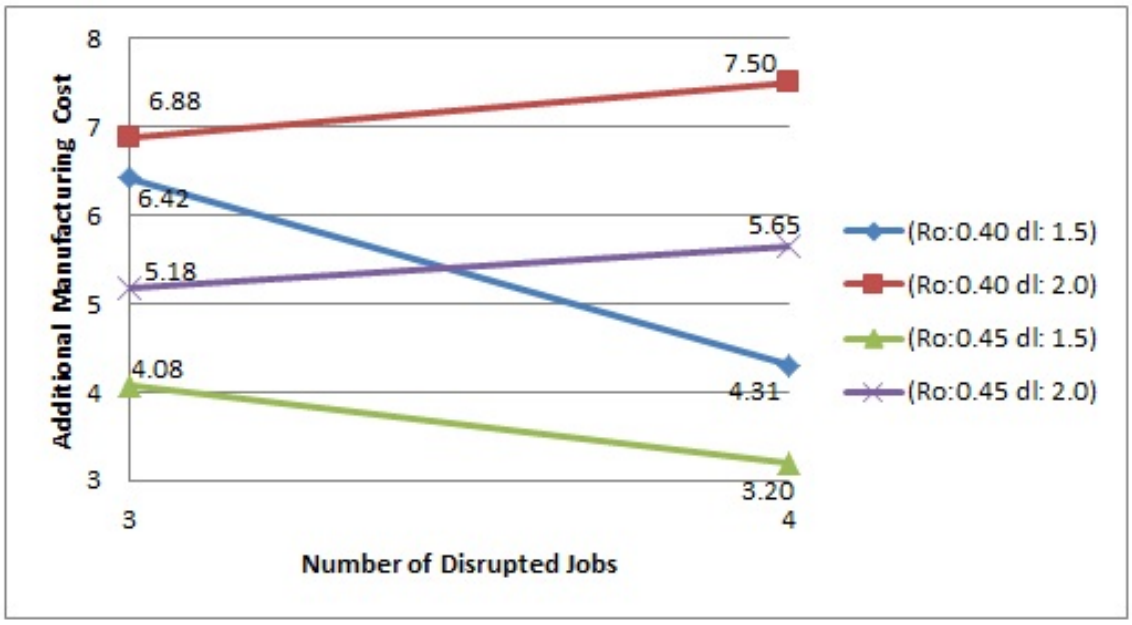

Figure 7.3: The relationship of conflicting objectives obtained by applying rescheduling mathematical model for the problem instance of 50 jobs and 2 machines.

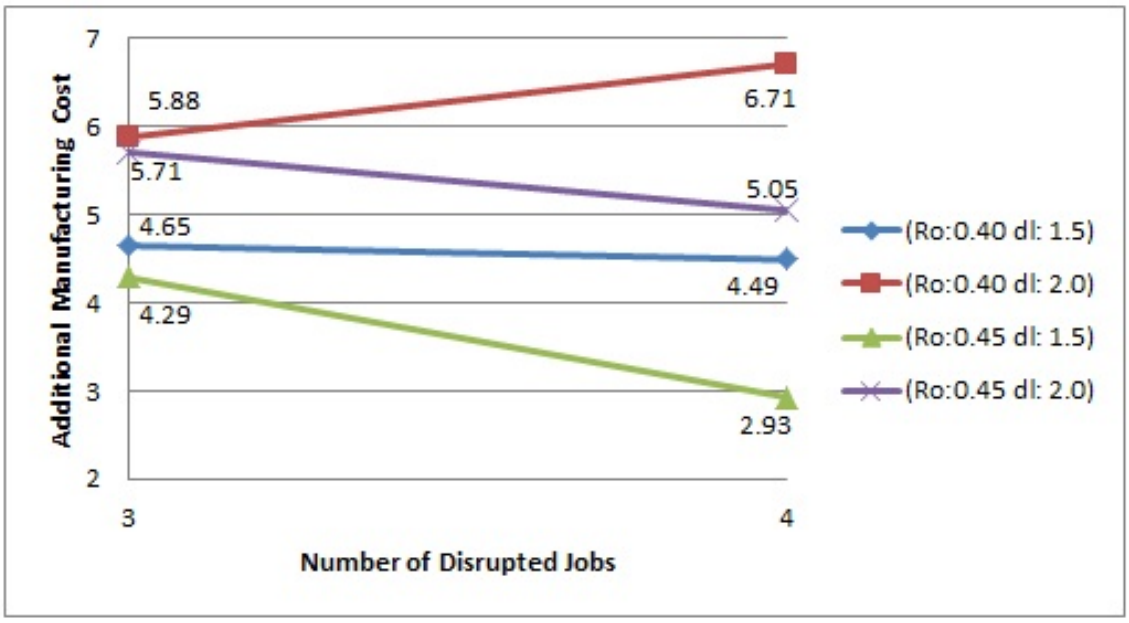

Figure 7.4: The relationship of conflicting objectives obtained by applying improvement search heuristic for the problem instance of 50 jobs and 2 machines.

Computational results for the problem instances having 100 jobs are presented in Table 7.10

and 7.11 respectively. These results are obtained by applying the improvement search heuristic, since our rescheduling mathematical model is able to solve the rescheduling problem for the environment of having less than 100 jobs. For more jobs and machines settings, rescheduling mathematical model fails to find solutions within the specified time limit. The observations presented above for the problem instances of 30 and 50 jobs are also valid for 100 problem instances.

Table 7.10: Computational results of improvement search heuristic for the problem instance of 100 jobs and 2 machines.

Disruption Length = 1.5						
Design Parameters					**Improvement Search Heuristic**	
N	**M**	**K**	ρ	**Number of affected jobs**	**Cost**	**CPU**
100	2	3	0.40	68	12.10	51.40
100	2	3	0.45	71	12.63	53.80
100	2	4	0.40	68	10.45	43.13
100	2	4	0.45	71	11.43	56.51

Table 7.11: Computational results of improvement search heuristic for the problem instance of 100 jobs and 2 machines.

Disruption Length = 2.0						
Design Parameters					**Improvement Search Heuristic**	
N	**M**	**K**	ρ	**Number of affected jobs**	**Cost**	**CPU**
100	2	3	0.40	68	14.20	29.46
100	2	3	0.45	71	16.46	42.90
100	2	4	0.40	68	14.06	28.02
100	2	4	0.45	71	13.86	47.72

The relation between the additional manufacturing cost and the number of allowable disrupted jobs is shown in Figure 7.5 for the improvement search algorithm for the problem instance of having 100 jobs.

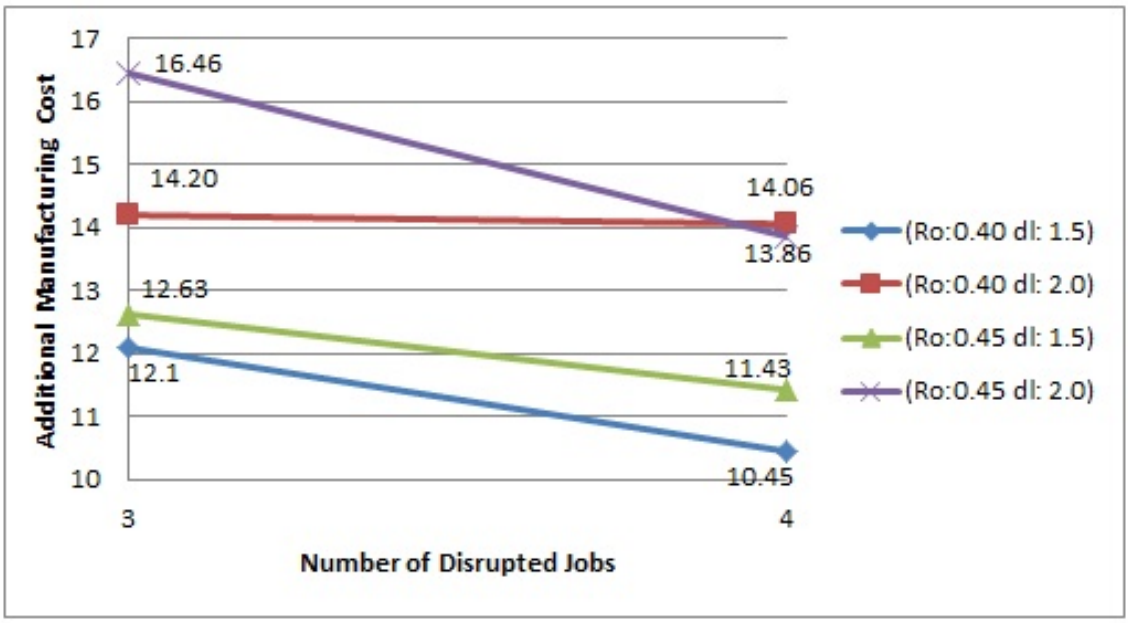

Figure 7.5: The relationship of conflicting objectives obtained by applying improvement search heuristic for the problem instance of 100 jobs and 2 machines.

According to the results, it is observed that minimizing both of the additional manufacturing cost and the allowable number of disrupted jobs is not possible. So that, there is a trade off relation between these two measures. It was seen that mathematical model can only find solutions up to 100 jobs problem instances due to the memory limitations. However, improvement search heuristic is able to generate feasible schedules having the additional manufacturing cost close to the result of rescheduling mathematical model within reasonable durations. Thus, it can find solutions for problem instances having more than 100 jobs. In addition to these observations, it is seen that improvement search heuristic can able to find better solutions than rescheduling mathematical model within the same time limit. The solutions to our rescheduling mathematical model are all appeared to be feasible and the additional manufacturing cost of rescheduling mathematical model for some problem instances were calculated higher than the additional manufacturing cost of improvement search algorithm.

As a result, we can generate feasible schedules quickly after the disruption at a reasonable manufacturing cost by applying our improvement search heuristic and/or we can distribute the effects of disruption to the entire schedule by creating alternative schedules with different levels of the number of disrupted jobs and related additional manufacturing costs.

CHAPTER 8

CONCLUSION

In this thesis, we have considered rescheduling with controllable processing times. In particular, we show that in contrast to fixed processing times, if we have the flexibility to control the processing times of the jobs, we can generate alternative reactive schedules in response to a disruption such as machine breakdown. We consider a non-identical parallel machine environment where processing times of the jobs are compressible at a certain cost which is a convex function of the compression on the processing time. We use the conic quadratic reformulation approach in solving the considered problems. We have developed a mathematical model and an improvement search heuristic in order to help making rescheduling decisions. In this Chapter, a brief summary of this thesis will be presented as well as the contributions to the literature. Future research topics related to this area will be introduced as well.

We performed studies presented in Chapter 3, 4 and 5 to explore the trade-off between the additional manufacturing cost objective and the number of disrupted jobs. In Chapter 3, we have defined and modelled our rescheduling problem. We have provided mathematical expressions for a compression cost of a job as a function of its compression amount. Afterwards, we have presented the assumptions for the problem and have provided a mathematical model. We have studied ϵ-constraint approach in order to handle the conflict of minimizing additional manufacturing cost and number of disrupted jobs simultaneously. Since the compression cost function includes convex and nonlinear terms, we have applied conic quadratic reformulation to our model in order to generate feasible schedules in reasonable durations.

In Chapter 4, we have studied single machine subproblem which is solved during our improvement search heuristic for each assignment. We have derived optimality properties which state the relationship of jobs' compression amounts and their related costs on a machine in an opti-

mal solution. Those properties led us to design our improvement search algorithm. In Chapter 5, proposed improvement search algorithm has been presented. In Chapter 6, a numerical example has been provided in order to make our solution procedure clear. In Chapter 7, we made experiments in order to compare the performance and solution quality of rescheduling mathematical model and improvement search heuristic.

As we mentioned before, the additional manufacturing cost and the allowable number of disrupted job objectives are conflicting and minimizing the additional manufacturing cost and the number of allowable disrupted jobs is not possible. So that, there is a trade off relation between these two measures which we investigated during our computational studies. As a result, it is seen that when the allowable number of disrupted jobs decreases, the additional manufacturing cost increases. In order to generate feasible and efficient solutions, we tried to minimize the additional manufacturing cost objective by putting a limit on the number of disrupted jobs and we achieved to form feasible schedules. We provided a rescheduling mathematical model to solve the considered problem.

According to our computational studies, it is seen that rescheduling mathematical model can only find solutions up to 100 jobs problem instances by taking the memory limitations into account. In addition to this, the solution time of the rescheduling mathematical model could not meet the time expectations for the problem instances which it was able to solve. To overcome this situation, we developed an improvement search algorithm which found good solutions in reasonable CPU times. Furthermore, we also showed that the improvement search heuristic is able to generate solutions close to and/or better than the rescheduling mathematical model solutions by the means of additional manufacturing cost.

As a future work, proposed improvement search algorithm can be developed further for the purpose of having more qualified solutions in even smaller CPU times. This aim can be achieved by improving our proposed initial schedule algorithm which is generated in order to apply one move and two swap algorithms. In our study, initial schedule algorithm tries to maximize the number of non-disrupted jobs while forming a new schedule after disruption. Therefore, the additional manufacturing cost of generated initial schedule occurs to be reasonably high and for that reason improving initial schedule requires higher solution times as manufacturing cost increases. An increase in quality of the initial schedule by the means of manufacturing cost may have positive effect on applying the improving moves. Beside

improving initial schedule algorithm, one move and two swap algorithms can be improved to have better solution quality and CPU times as well. In our study, in order to improve the existing schedules, we have tried 20 promising one moves and two swaps for each assignment which cause an increase in solution time of the improvement search algorithm. For that reason, the possibility of the improvement by applying promising moves can also be considered by taking additional indicators into account.

BIBLIOGRAPHY

[1] G.E. Vieira, J.W. Herrmann, and E. Lin. Rescheduling manufacturing systems: A framework of strategies, policies and methods. *Journal of Scheduling*, 6:39–62, 2003.

[2] H. Aytug, M.A. Lawley, K. McKay, S. Mohan, and R. Uzsoy. Executing production schedules in the face of uncertainties: A review and some future directions. *European Journal of Operational Research*, 161:86–110, 2005.

[3] V.J. Leon, S.D. Wu, and R.H. Storer. Robustness measures and robust scheduling for job shops. *IIE Transactions*, 26(5):32–43, 1994.

[4] S.Y. Nof and F.H. Grant. Adaptive/predictive scheduling: review and a general framework. *Production Planning and Control*, 2(4):298–312, 1991.

[5] E. Kutanoğlu and İ. Sabuncuoğlu. Routing-based reactive scheduling policies for machine failures in job shops. *International Journal of Production Research*, 39:3141–3158, 2001.

[6] S.D. Wu, E. Byeon, and R.H. Storer. A graph-theoretic decomposition of the job shop scheduling problem to achieve scheduling robustness. *Operations Research*, 47(1):113–124, 1999.

[7] İ. Sabuncuoğlu and S. Karabük. Rescheduling frequency in an FMS with uncertain processing times and unreliable machines. *Journal of Manufacturing Systems*, 18(4):268–283, 1999.

[8] N.G. Hall and C.N. Potts. Rescheduling for new orders. *Operations Research*, 52(3):440–453, May-June 2004.

[9] O. Alagöz and M. Azizoğlu. Rescheduling of identical parallel machines under machine eligibility constraint. *European Journal of Operational Research*, 149:523–532, 2003.

[10] M. Azizoğlu and O. Alagöz. Parallel-machine rescheduling with machine disruptions. *IIE Transactions*, 37:1113–1118, 2005.

[11] A. Türkcan. Predictive/reactive scheduling with controllable processing times and earliness-tardiness penalties. *IIE Transactions*, 41(12):1080–1095, 2009.

[12] B. Yang. Single machine rescheduling with new jobs arrivals and processing time compression. *International Journal of Advanced Manufacturing Technology*, 34(3-4):378–384, 2007.

[13] J.C. Bean, J.R. Birge, J. Mittenthal, and C.E. Noon. Match-up scheduling with multiple resources, release dates and disruptions. *Operations Research*, 39(3):470–483, 1991.

[14] M.S. Aktürk and E. Görgülü. Match-up scheduling under a machine breakdown. *European Journal of Operational Research*, 112:81–97, 1999.

[15] M.S. Aktürk, A. Atamtürk, and S. Gürel. Parallel machine match-up scheduling with manufacturing cost considerations. *Journal of Scheduling*, 13:95–110, 2010.

[16] B. Yang and J. Geunes. Predictive-reactive scheduling on a single resource with uncertain future jobs. *European Journal of Operational Research*, 189(3):1267 – 1283, 2008.

[17] S.V. Mehta and R.M. Uzsoy. Predictable scheduling of a job shop subject to breakdowns. *IEEE Trans. Robot. Autom.*, 14:365–378, 1998.

[18] R. O'Donovan, R.M. Uzsoy, and K.N. McKay. Predictable scheduling of a single machine with breakdowns and sensitive jobs. *International Journal of Production Research*, 37(18):4217–4233, 1999.

[19] R. Leus and W. Herroelen. Scheduling for stability in single-machine production *Journal of Scheduling*, 10:223–235, 2007.

[20] D. Shabtay and M. Kaspi. Minimizing the total weighted flow time in a single machine with controllable processing times. *Computers and Operations Research*, 31:2279–2289, 2004.

[21] D. Shabtay and M. Kaspi. Parallel machine scheduling with a convex resource consumption function. *European Journal of Operational Research*, 173(1):92–107, 2006.

[22] D. Shabtay and G. Steiner. A survey of scheduling with controllable processing times. *Discrete Applied Mathematics*, 155(13):1643–1666, 2007.

[23] H. Hoogeven. Multicriteria scheduling. *European Journal of Operational Research*, 167:592–623, 2005.

[24] N. Megow, R.H. Möhring, and J. Schulz. Decision support and optimization in shutdown and turnaround scheduling. *INFORMS Journal on Computing*, 23(2):189–204, 2011.

[25] S. Gürel and M.S. Aktürk. Scheduling preventive maintenance on a single CNC machine. *International Journal of Production Research*, 46(24):6997–6821, 2008.

[26] S. Gürel and M.S. Aktürk. Optimal allocation and processing time decisions on parallel CNC machines: ϵ-constraint approach. *European Journal of Operational Research*, 183:591–607, 2007.

[27] S. Gürel and M.S. Aktürk. Scheduling parallel CNC machines with time/cost trade-off considerations. *Computers and Operations Research*, 34:2774–2789, 2007.

[28] M.S. Aktürk, A. Atamtürk, and S. Gürel. A strong conic quadratic reformulation for machine-job assignment with controllable processing times. *Operations Research Letters*, 37(3):187 – 191, 2009.

[29] J.M. Moore. An n-job, one machine sequencing algorithm for minimizing the number of late jobs. *Management Science*, 15:102 – 109, 1968.

[30] M. Pinedo. *Scheduling: theory, algorithms and systems*. Prentice Hall, New Jersey, second edition, 2002.

[31] R.L. Daniels and R.K. Sarin. Single machine scheduling with controllable processing times and number of jobs tardy. *Operations Research*, 37(6):981–984, November-December 1989.

[32] T. C. E. Cheng, Z.-L. Chen, and C.L. Li. Single-machine scheduling with trade-off between number of tardy jobs and resource allocation. *Operations Research Letters*, 19(5):237 – 242, 1996.

[33] Y. He, Q. Wei, and T.C.E. Cheng. Single-machine scheduling with trade-off between number of tardy jobs and compression cost. *Journal of Scheduling*, 10:303–310, 2007.

[34] L. Yedidsion, D. Shabtay, E. Korach, and M. Kaspi. A bicriteria approach to minimize number of tardy jobs and resource consumption in scheduling a single machine. *International Journal of Production Economics*, 119(2):298 – 307, 2009.

APPENDIX A

COMPUTATIONAL RESULTS

Table A.1: Rescheduling Mathematical Model Results for 30 jobs.

Disruption Length = 1.5												
Design Parameters						Rescheduling Mathematical Model						
N	**M**	**K**	ρ	**Replication**	**Number of Affected Jobs**	**Cost**	**CPU**	**Gap**	**K***	**Optimal**	**Feasible**	**Infeasible**
30	2	3	0.40	1	18	2.36	600.09	7.06	3	-	1	-
30	2	3	0.40	2	-	-	-	-	-	-	-	1
30	2	3	0.40	3	17	3.41	600.03	13.02	3	-	1	-
30	2	3	0.40	4	23	2.33	600.01	9.18	3	-	1	-
30	2	3	0.40	5	18	3.47	600.04	13.19	3	-	1	-
30	2	3	0.45	1	18	1.36	600.03	3.72	3	-	1	-
30	2	3	0.45	2	20	1.66	600.10	4.33	3	-	1	-
30	2	3	0.45	3	17	1.63	600.03	4.76	3	-	1	-
30	2	3	0.45	4	25	2.25	600.07	5.29	3	-	1	-
30	2	3	0.45	5	19	1.77	600.01	6.11	3	-	1	-
30	2	4	0.40	1	18	2.36	600.03	8.12	4	-	1	-
30	2	4	0.40	2	-	-	-	-	-	-	-	1
30	2	4	0.40	3	17	3.72	600.03	14.40	4	-	1	-
30	2	4	0.40	4	23	2.42	600.01	9.91	4	-	1	-
30	2	4	0.40	5	18	4.27	600.01	16.92	4	-	1	-
30	2	4	0.45	1	18	1.34	600.06	3.81	4	-	1	-
30	2	4	0.45	2	20	1.66	600.01	4.89	4	-	1	-
30	2	4	0.45	3	17	1.75	600.03	5.36	4	-	1	-
30	2	4	0.45	4	25	2.25	600.14	5.54	4	-	1	-
30	2	4	0.45	5	19	1.77	600.01	6.33	4	-	1	-

Table A.2: Improvement Search Algorithm Results for 30 jobs.

Disruption Length = 1.5									
Design Parameters						**Improvement Search Algorithm**			
N	**M**	**K**	ρ	**Replication**	**Number of Affected Jobs**	**Cost**	**CPU**	**Initial Schedule Cost**	**Initial Schedule CPU**
30	2	3	0.40	1	18	1.88	12.37	13.14	3.86
30	2	3	0.40	2	-	-	-	-	-
30	2	3	0.40	3	17	5.40	5.76	13.60	3.09
30	2	3	0.40	4	23	1.74	9.20	32.19	5.46
30	2	3	0.40	5	18	3.17	6.24	20.94	3.60
30	2	3	0.45	1	18	1.34	21.25	27.10	3.76
30	2	3	0.45	2	20	2.03	16.52	6.50	4.84
30	2	3	0.45	3	17	1.98	18.74	13.60	3.20
30	2	3	0.45	4	25	2.51	38.06	32.19	5.16
30	2	3	0.45	5	19	1.72	18.72	20.94	3.51
30	2	4	0.40	1	18	1.88	12.31	13.14	4.11
30	2	4	0.40	2	-	-	-	-	-
30	2	4	0.40	3	17	3.82	24.01	24.16	3.95
30	2	4	0.40	4	23	1.66	19.02	11.32	5.56
30	2	4	0.40	5	18	3.18	17.38	8.94	5.07
30	2	4	0.45	1	18	1.34	19.87	27.1	3.78
30	2	4	0.45	2	20	2.03	16.54	5.85	4.56
30	2	4	0.45	3	17	1.60	16.07	13.60	3.55
30	2	4	0.45	4	25	3.23	34.48	32.19	5.54
30	2	4	0.45	5	19	1.54	15.97	20.94	4.12

Table A.3: Rescheduling Mathematical Model Results for 30 jobs.

Disruption Length = 2.0												
Design Parameters						**Rescheduling Mathematical Model**						
N	**M**	**K**	ρ	**Replication**	**Number of Affected Jobs**	**Cost**	**CPU**	**Gap**	**K***	**Optimal**	**Feasible**	**Infeasible**
30	2	3	0.40	1	18	2.91	600.03	7.72	3	-	1	-
30	2	3	0.40	2	-	-	-	-	-	-	-	1
30	2	3	0.40	3	17	6.12	600.01	19.34	3	-	1	-
30	2	3	0.40	4	23	5.01	600.03	10.04	3	-	1	-
30	2	3	0.40	5	18	5.42	600.06	15.69	3	-	1	-
30	2	3	0.45	1	18	1.86	600.03	4.54	3	-	1	-
30	2	3	0.45	2	20	4.62	600.03	6.35	3	-	1	-
30	2	3	0.45	3	17	3.12	600.01	7.28	3	-	1	-
30	2	3	0.45	4	25	3.79	600.03	4.47	3	-	1	-
30	2	3	0.45	5	19	2.32	600.03	7.35	3	-	1	-
30	2	4	0.40	1	18	4.13	600.06	10.69	4	-	1	-
30	2	4	0.40	2	-	-	-	-	-	-	-	1
30	2	4	0.40	3	17	4.43	600.01	15.61	4	-	1	-
30	2	4	0.40	4	23	4.13	600.6	9.94	4	-	1	-
30	2	4	0.40	5	18	4.26	600.03	15.65	4	-	1	-
30	2	4	0.45	1	18	2.03	600.03	4.97	4	-	1	-
30	2	4	0.45	2	20	4.35	600.03	6.98	4	-	1	-
30	2	4	0.45	3	17	2.77	600.01	7.40	4	-	1	-
30	2	4	0.45	4	25	3.86	600.04	5.73	4	-	1	-
30	2	4	0.45	5	19	2.32	600.07	7.37	4	-	1	-

Table A.4: Improvement Search Algorithm Results for 30 jobs.

Disruption Length = 2.0									
Design Parameters						**Improvement Search Algorithm**			
N	**M**	**K**	ρ	**Replication**	**Number of Affected Jobs**	**Cost**	**CPU**	**Initial Schedule Cost**	**Initial Schedule CPU**
30	2	3	0.40	1	18	4.62	17.07	24.00	4.16
30	2	3	0.40	2	-	-	-	-	-
30	2	3	0.40	3	17	4.58	16.38	17.82	4.89
30	2	3	0.40	4	23	6.38	29.66	38.29	3.85
30	2	3	0.40	5	18	4.83	13.93	12.95	3.28
30	2	3	0.45	1	18	1.76	16.75	14.67	4.24
30	2	3	0.45	2	20	2.94	22.62	7.65	4.59
30	2	3	0.45	3	17	4.97	20.79	18.53	4.03
30	2	3	0.45	4	25	2.85	29.89	25.34	6.23
30	2	3	0.45	5	19	3.79	12.82	18.79	3.97
30	2	4	0.40	1	18	3.08	16.01	24.00	4.02
30	2	4	0.40	2	-	-	-	-	-
30	2	4	0.40	3	17	4.58	14.79	17.82	3.32
30	2	4	0.40	4	23	5.33	25.79	38.29	4.54
30	2	4	0.40	5	18	4.45	14.18	12.95	3.73
30	2	4	0.45	1	18	1.44	17.92	14.67	3.87
30	2	4	0.45	2	20	3.33	17.74	7.65	3.52
30	2	4	0.45	3	17	3.40	17.85	18.53	2.65
30	2	4	0.45	4	25	3.33	23.12	25.34	4.24
30	2	4	0.45	5	19	3.79	7.02	18.79	2.41

Table A.5: Rescheduling Mathematical Model Results for 50 jobs.

Disruption Length = 1.5												
Design Parameters						**Rescheduling Mathematical Model**						
N	**M**	**K**	ρ	**Replication**	**Number of Affected Jobs**	**Cost**	**CPU**	**Gap**	**K***	**Optimal**	**Feasible**	**Infeasible**
50	2	3	0.40	1	45	8.07	600.03	13.62	3	-	1	-
50	2	3	0.40	2	33	5.74	600.25	7.82	3	-	1	-
50	2	3	0.40	3	39	5.36	600.18	13.02	3	-	1	-
50	2	3	0.40	4	36	3.83	600.12	6.64	3	-	1	-
50	2	3	0.40	5	38	9.08	600.56	11.58	3	-	1	-
50	2	3	0.45	1	47	4.94	600.20	7.89	3	-	1	-
50	2	3	0.45	2	33	1.07	600.23	1.82	3	-	1	-
50	2	3	0.45	3	41	4.98	600.34	3.86	3	-	1	-
50	2	3	0.45	4	36	7.95	600.20	12.23	3	-	1	-
50	2	3	0.45	5	39	1.44	600.35	2.49	3	-	1	-
50	2	4	0.40	1	45	3.10	600.18	7.54	4	-	1	-
50	2	4	0.40	2	33	2.23	600.13	1.52	4	-	1	-
50	2	4	0.40	3	39	6.96	600.23	3.85	4	-	1	-
50	2	4	0.40	4	36	3.90	600.03	6.36	4	-	1	-
50	2	4	0.40	5	38	5.38	600.10	8.71	4	-	1	-
50	2	4	0.45	1	47	5.50	600.04	8.91	4	-	1	-
50	2	4	0.45	2	33	1.02	600.25	1.88	4	-	1	-
50	2	4	0.45	3	41	4.94	600.09	3.89	4	-	1	-
50	2	4	0.45	4	36	2.85	600.11	6.12	4	-	1	-
50	2	4	0.45	5	39	1.69	600.42	3.10	4	-	1	-

Table A.6: Improvement Search Algorithm Results for 50 jobs.

Disruption Length = 1.5									
Design Parameters						**Improvement Search Algorithm**			
N	**M**	**K**	ρ	**Replication**	**Number of Affected Jobs**	**Cost**	**CPU**	**Initial Schedule Cost**	**Initial Schedule CPU**
50	2	3	0.40	1	45	6.74	35.02	31.54	10.65
50	2	3	0.40	2	33	3.83	25.23	27.71	7.97
50	2	3	0.40	3	39	5.79	37.46	29.94	10.45
50	2	3	0.40	4	36	3.92	24.06	22.68	8.71
50	2	3	0.40	5	38	2.96	58.34	58.24	10.56
50	2	3	0.45	1	47	4.13	39.64	22.80	9.58
50	2	3	0.45	2	33	0.78	27.60	5.32	11.45
50	2	3	0.45	3	41	5.46	56.19	35.33	17.82
50	2	3	0.45	4	36	5.25	21.76	11.76	7.46
50	2	3	0.45	5	39	5.84	28.24	16.58	10.23
50	2	4	0.40	1	45	6.36	29.83	31.54	9.49
50	2	4	0.40	2	33	3.41	44.29	27.71	8.60
50	2	4	0.40	3	39	5.74	27.03	29.94	8.41
50	2	4	0.40	4	36	2.17	24.26	22.68	8.25
50	2	4	0.40	5	38	4.78	46.93	58.24	9.84
50	2	4	0.45	1	47	3.14	35.12	22.80	9.61
50	2	4	0.45	2	33	0.78	24.49	5.32	10.27
50	2	4	0.45	3	41	5.31	43.10	35.33	15.50
50	2	4	0.45	4	36	3.66	24.40	11.76	6.93
50	2	4	0.45	5	39	1.75	35.83	16.58	10.02

Table A.7: Rescheduling Mathematical Model Results for 50 jobs.

Disruption Length = 2.0												
Design Parameters						**Rescheduling Mathematical Model**						
N	**M**	**K**	ρ	**Replication**	**Number of Affected Jobs**	**Cost**	**CPU**	**Gap**	**K***	**Optimal**	**Feasible**	**Infeasible**
50	2	3	0.40	1	-	-	-	-	-	-	-	1
50	2	3	0.40	2	33	5.83	600.15	7.31	3	-	1	-
50	2	3	0.40	3	39	5.45	600.10	6.62	3	-	1	-
50	2	3	0.40	4	36	7.95	600.14	9.64	3	-	1	-
50	2	3	0.40	5	38	8.30	600.07	7.83	3	-	1	-
50	2	3	0.45	1	47	6.93	600.17	9.87	3	-	1	-
50	2	3	0.45	2	33	3.58	600.21	1.99	3	-	1	-
50	2	3	0.45	3	41	5.14	600.26	3.23	3	-	1	-
50	2	3	0.45	4	36	8.23	600.09	6.42	3	-	1	-
50	2	3	0.45	5	39	2.03	600.25	3.03	3	-	1	-
50	2	4	0.40	1	45	8.75	600.17	14.01	4	-	1	-
50	2	4	0.40	2	33	5.16	600.07	7.08	4	-	1	-
50	2	4	0.40	3	39	11.43	600.09	12.63	4	-	1	-
50	2	4	0.40	4	36	5.54	600.20	7.74	4	-	1	-
50	2	4	0.40	5	38	6.62	600.26	8.85	4	-	1	-
50	2	4	0.45	1	47	5.54	600.09	8.68	4	-	1	-
50	2	4	0.45	2	33	2.40	600.09	2.00	4	-	1	-
50	2	4	0.45	3	41	7.92	600.17	5.86	4	-	1	-
50	2	4	0.45	4	36	9.14	600.11	10.59	4	-	1	-
50	2	4	0.45	5	39	3.25	600.14	4.51	4	-	1	-

Table A.8: Improvement Search Algorithm Results for 50 jobs.

Disruption Length = 2.0									
Design Parameters						**Improvement Search Algorithm**			
N	**M**	**K**	ρ	**Replication**	**Number of Affected Jobs**	**Cost**	**CPU**	**Initial Schedule Cost**	**Initial Schedule CPU**
50	2	3	0.40	1	-	-	-	-	-
50	2	3	0.40	2	33	3.93	37.02	22.22	9.66
50	2	3	0.40	3	39	4.56	33.07	13.58	9.72
50	2	3	0.40	4	36	9.61	21.93	45.74	8.59
50	2	3	0.40	5	38	5.43	30.15	33.47	9.09
50	2	3	0.45	1	47	4.02	27.60	15.50	8.54
50	2	3	0.45	2	33	3.67	24.73	11.08	10.98
50	2	3	0.45	3	41	10.69	32.82	49.90	11.81
50	2	3	0.45	4	36	7.13	24.07	23.72	8.24
50	2	3	0.45	5	39	3.06	40.70	23.56	8.80
50	2	4	0.40	1	45	7.37	24.91	47.84	6.77
50	2	4	0.40	2	33	5.24	25.94	22.22	8.66
50	2	4	0.40	3	39	8.67	43.18	13.58	13.07
50	2	4	0.40	4	36	9.36	62.09	45.74	11.18
50	2	4	0.40	5	38	2.90	48.72	33.47	8.76
50	2	4	0.45	1	47	4.33	26.33	15.50	8.18
50	2	4	0.45	2	33	1.96	24.84	11.08	10.23
50	2	4	0.45	3	41	6.91	44.93	49.90	14.48
50	2	4	0.45	4	36	8.72	34.84	23.72	9.08
50	2	4	0.45	5	39	3.33	30.23	23.56	10.56

Table A.9: Improvement Search Algorithm Results for 100 jobs.

Disruption Length = 1.5									
Design Parameters						**Improvement Search Algorithm**			
N	**M**	**K**	ρ	**Replication**	**Number of Affected Jobs**	**Cost**	**CPU**	**Initial Schedule Cost**	**Initial Schedule CPU**
100	2	3	0.40	1	74	13.42	50.12	33.89	35.94
100	2	3	0.40	2	65	11.83	55.44	37.44	24.85
100	2	3	0.40	3	62	6.63	47.19	25.48	29.64
100	2	3	0.40	4	71	16.53	52.84	32.56	32.81
100	2	3	0.40	5	-	-	-	-	-
100	2	3	0.45	1	76	16.17	70.92	30.94	41.18
100	2	3	0.45	2	66	7.17	55.40	33.86	30.12
100	2	3	0.45	3	68	9.46	53.77	16.12	44.72
100	2	3	0.45	4	74	20.39	38.22	32.40	25.13
100	2	3	0.45	5	72	9.96	50.70	16.91	36.96
100	2	4	0.40	1	74	11.92	51.54	33.89	37.87
100	2	4	0.40	2	65	10.87	44.26	37.44	23.45
100	2	4	0.40	3	62	8.66	46.52	25.48	31.94
100	2	4	0.40	4	71	14.45	37.91	32.56	27.86
100	2	4	0.40	5	68	6.35	35.44	15.95	24.26
100	2	4	0.45	1	76	12.52	87.91	30.94	43.70
100	2	4	0.45	2	66	7.66	44.60	33.86	28.80
100	2	4	0.45	3	68	7.47	58.69	16.12	41.41
100	2	4	0.45	4	74	19.73	53.23	32.40	33.61
100	2	4	0.45	5	72	9.75	38.11	16.91	27.94

Table A.10: Improvement Search Algorithm Results for 100 jobs.

Disruption Length = 2.0									
Design Parameters						**Improvement Search Algorithm**			
N	**M**	**K**	ρ	**Replication**	**Number of Affected Jobs**	**Cost**	**CPU**	**Initial Schedule Cost**	**Initial Schedule CPU**
100	2	3	0.40	1	74	32.38	45.99	51.28	36.38
100	2	3	0.40	2	65	19.19	47.36	44.11	23.96
100	2	3	0.40	3	62	36.19	36.77	50.08	30.28
100	2	3	0.40	4	71	17.02	34.90	28.13	27.25
100	2	3	0.40	5	-	-	-	-	-
100	2	3	0.45	1	76	32.98	51.37	41.78	41.7
100	2	3	0.45	2	66	8.86	94.32	34.72	74.46
100	2	3	0.45	3	68	9.59	74.18	12.05	43.63
100	2	3	0.45	4	74	16.14	49.09	22.52	30.75
100	2	3	0.45	5	72	14.75	33.04	35.14	23.98
100	2	4	0.40	1	74	15.80	68.39	51.28	35.41
100	2	4	0.40	2	65	13.69	37.64	44.11	22.15
100	2	4	0.40	3	62	15.04	42.93	50.08	30.73
100	2	4	0.40	4	71	13.87	40.92	28.13	28.1
100	2	4	0.40	5	68	11.90	46.05	32.06	23.73
100	2	4	0.45	1	76	22.12	99.79	41.78	41.56
100	2	4	0.45	2	66	7.59	112.63	34.72	95.05
100	2	4	0.45	3	68	9.35	80.09	12.05	46.17
100	2	4	0.45	4	74	16.02	69.06	22.52	32.03
100	2	4	0.45	5	72	14.22	32.82	35.14	23.82