RESCHEDULING PARALLEL MACHINES WITH CONTROLLABLE PROCESSING TIMES

A THESIS

SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL ENGINEERING

AND THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Müge Muhafız

May, 2012

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

Prof. M. Selim Aktürk (Advisor)

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

Asst. Prof. Sinan Gürel (Co-Advisor)

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

Asst. Prof. Alper Şen

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

Asst. Prof. Nagihan Çömez

Approved for the Graduate School of Engineering and Science:

Prof. Dr. Levent Onural
Director of the Graduate School

ABSTRACT

RESCHEDULING PARALLEL MACHINES WITH CONTROLLABLE PROCESSING TIMES

Müge Muhafız
M.S. in Industrial Engineering
Supervisor: Prof. M. Selim Aktürk
Co-Supervisor: Asst. Prof. Sinan Gürel
May, 2012

In many manufacturing environments, the production does not always endure as it is planned. Many times, it is interrupted by a disruption such as machine breakdown, power loss, etc. In our problem, we are given an original production schedule in a non-identical parallel machine environment and we assume that one of the machines is disrupted at time t.

Our aim is to revise the schedule, although there are some restrictions that should be considered while creating the revised schedule. Disrupted machine is unavailable for a certain time. New schedule has to satisfy the maximum completion time constraint of each machine. Furthermore, when we revise the schedule we have to satisfy the constraint that the revised start time of a job cannot be earlier than its original start time. Because, we assume that jobs are not ready before their original start times in the revised schedule.

Therefore, we have to find an alternative solution to decrease the negative impacts of this disruption as much as possible. One way to process a disrupted job in the revised schedule is to reallocate the job to another machine. The other way is to keep the disrupted job at its original machine, but to delay its start time after the end time of the disruption. Since the machines might be fully utilized originally, we may have to compress some of the processing times in order to add a new job to a machine or to reallocate the jobs after the disruption ends. Consequently, we assume that the processing times are controllable within the given lower and upper bounds.

Our first objective is to minimize the sum of reallocation and nonlinear compression costs. Besides, it is important to deliver the orders on time, not earlier

or later than they are promised. Therefore, we try to maintain the original completion times as much as possible. So, the second objective is to minimize the total absolute deviations of the completion times in the revised schedule from the original completion times.

We developed a bi-criteria non-linear mathematical model to solve this non-identical parallel machine rescheduling problem. Since we have two objectives, we handled the second objective by giving it an upper bound and adding this bound as a constraint to the problem. By utilizing the second order cone programming, we solved this mixed-integer nonlinear mathematical model using a commercial MIP solver such as CPLEX. We also propose a decision tree based heuristic algorithm. Our algorithm generates a set of solutions for a problem instance and we test the solution quality of the algorithm solving same problem instances by the mathematical model. According to our computational experiments, the proposed heuristic approach could obtain close solutions for the first objective for a given upper bound on the second objective.

Keywords: Rescheduling, Parallel machines, Controllable processing times, Bi-criteria, Total absolute deviations of completion times, Convex cost function, Reallocation.

ÖZET

KONTROL EDİLEBİLİR İŞLEM SÜRELERİYLE PARALEL MAKİNALARDA YENİDEN ÇİZELGELEME

Müge Muhafız
Endüstri Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Prof. Dr. M. Selim Aktürk
Eş-Tez Yöneticisi: Yrd. Doç. Dr. Sinan Gürel
Mayıs, 2012

İmalat sistemlerinde, üretim her zaman planlandığı gibi uygulanamaz. Çoğu zaman, makine bozulması, elektrik kesintisi gibi nedenlerden dolayı üretim aksamak zorunda kalır. Bu çalışmada, özdeş olmayan paralel makinelerin bulunduğu bir imalat ortamında, önceden planlanmış bir üretim çizelgesinde, makinelerden birinde herhangi bir t anında aksama meydana geldiği varsayımında bulunduk ve makinelerde yeniden çizelgeleme üzerine çalıştık.

Bu çalışmada önceden planlanmış çizelgeyi aksaklık sonrasında mümkün olduğunca çabuk yakalamayı ve aksaklıktan dolayı meydana gelen zaman kaybını telafi etmeyi amaçladık. Ancak yeniden çizelge oluştururken dikkat etmemiz gereken bazı kısıtlamalar bulunmaktadır. Aksaklık sona erene kadar, aksama meydana gelen makine durur ve hiçbir iş işleyemez. Diğer bir yandan, yeni çizelgede makinelerin kapasite kısıtına dikkat edilmelidir. Bununla birlikte, yeni çizelgede işlerin başlangıç zamanları önceden planlanan çizelgeki başlangıç zamanlarından daha erken olmamalıdır.

Makinede meydana gelen aksaklığın negatif etkilerini yumuşatmak için alternatif bir çözüm bulmamız gerekmektedir. Bunun bir yolu aksayan bir işi başka bir makineye taşımak ya da aksayan işi başlangıçtaki makinesinde bırakmak ancak aksamanın bitişinden sonra işlemektir. Ancak makinelerin kapasitesi tamamen dolu olabileceği göz önünde bulundurulursa, bir makineye yeni bir iş taşıyabilmek ya da aksayan makinede işleri aksama bittikten sonra işleyebilmek için işlerin işlem sürelerini sıkıştırmak zorunda kalınabilir. Sonuç olarak, bu çalışmada işlerin işlem sürelerinin en az ve en yüksek sınırları dahilinde kontrol edilebilir olduğu varsayımını kullandık.

Öncelikli amacımız, işlem sürelerinin sıkıştırılma miktarının doğrusal olmayan bir fonksiyonu olan sıkıştırma maliyeti ile taşıma maliyetini enazlamaktır. Bunun yanı sıra, çizelgede aksaklık olsa bile işlerin mümkün olan en kısa zamanda tamamlanması çok önemlidir. Bu yüzden işlerin ilk çizelgedeki bitiş sürelerini mümkün olduğunca yakalamaya çalışmayı amaçlamaktayız. Dolayısıyla, ikinci amacımız, yeniden oluşturulan çizelge ile ilk çizelgedeki işlerin bitiş zamanları arasındaki mutlak farkların toplamını enazlamaktır.

Bu problemi çözebilmek için çift hedefli doğrusal olmayan bir matematiksel model geliştirdik. Çift hedefimiz olduğu için, ikinci hedefimiz olan bitiş zamanları mutlak farklarının toplamına bir üst sınır vererek bu sınırı matematiksel modele kısıt olarak ekledik. İkinci derece konik programlama tekniğinden faydalanarak, bu modeli CPLEX ile karmaşık tam sayılı matematiksel modele çevirerek çözdük. Problemin zorluğundan dolayı, çok uzun hesaplama sürelerinde mutlak çözüm bulunamadığı durumlar için etkin çözümler üreten hızlı sezgisel tarama algoritmaları geliştirdik. Sayısal deneylerimize göre, önerdiğimiz sezgisel yöntemler ikinci amaç fonksiyonu için verilen üst sınır kısıtı altında, birinci amaç fonksiyonu açısından matematiksel modelle yakın sonuçlara ulaşmaktadır.

Anahtar sözcükler: Yeniden çizelgeleme, Paralel makineler, Kontrol edilebilir işlem süreleri, Çift hedefli optimizasyon, İş bitiş süreleri farkları mutlak değerleri toplamı, Dış bükey maliyet fonksiyonu, Yeniden dağıtma, Yeniden atama.

Acknowledgement

I would like to express my sincere gratitude to Prof. Selim Aktürk and Asst. Prof. Sinan Gürel for their invaluable guidance and support during my graduate study. They have supervised me with everlasting patience and encouragement throughout this thesis. I consider myself lucky to have a chance to work with them.

I am also grateful to Asst. Prof. Alper Şen and Asst. Prof. Nagihan Çömez for accepting to read and review this thesis. Their comments and suggestions have been invaluable.

I am indeed grateful to my fiancée İnanç Yıldız for his morale support, patience, encouragement and endless love since I met him.

I would like to thank to my precious friends Pelin Elaldı, Fevzi Yılmaz, Nurcan Bozkaya and Onur Uzunlar for their endless support, motivation and wholehearted love. Life and the graduate study would not have been bearable without them. They have always made the life easier for me.

I also would like to thank Erdem Özdemir and Onur Uzunlar for their useful technical discussions and contributions to my work and thank my sister Begüm Muhafız and my friend Pelin Elaldı for their collaboration in the design of this study.

Last but not the least; I also would like to express my deepest gratitude to my family for their eternal love, support and trust at all stages of my life and especially during my graduate study. I can't express my gratitude for my mom Suzan Muhafız in words, she gave her unconditional love and support especially for the last 2 months of this study by staying with me. Having her endless support with me has been my greatest strength. I am grateful to my dad Hüseyin Muhafız for his constant inspiration and guidance. He has done the best he can, to provide me the endless support both in my life and in this study. The constant love and support of my sister Begüm Muhafız is sincerely acknowledged. I feel very lucky

that I belong to this family.

Finally, I would like to acknowledge financial support of The Scientific and Technological Research Council of Turkey (TUBITAK) for the Graduate Study Scholarship Program.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the scheduling literature, stable environments are mainly considered. However, in many manufacturing environments, the production does not always endure as it is planned. Many times, it is interrupted by a disruption such as machine breakdown, power loss, etc. During the disruption, machine becomes unavailable and cannot process any job. When the machine becomes unavailable, the existing schedule is no longer applicable, so it is needed to reschedule the jobs on the machines.

The important issue in the rescheduling is to compensate the effects of the disruption on the original schedule while keeping the solution quality of the revised schedule as high as possible. The effect of the disruption on the original schedule is measured in terms of the stability which is the deviation between the original and the revised schedule. On the other hand, the solution quality of the schedule is measured in terms of the performance criterion that is considered.

Stability is an important measure in the rescheduling because it shows how much the revised schedule deviates from the original schedule. Minimizing the effects of the disruption in the original schedule is possible by having higher stability. There are several stability measures in the rescheduling literature. Deviation of job starting times between the original and the revised schedule, difference of job sequences between original and revised schedule, number of disrupted jobs

which are reallocated to different machines are some stability measures that are used in the rescheduling literature. Total absolute deviation of job completion times (TADC) is also a stability measure that is commonly used to measure deviation between the original and revised schedule. In manufacturing systems, it is important to deliver the jobs on time that they are promised to be. Production plans are made according to the delivery times and jobs are scheduled according to the production plan. If the completion time of a job in the revised schedule exceeds the original completion time of the job due to the disruption, the job has to be delivered late. If a job is completed in the revised schedule earlier than the completion time in the original schedule, then the job has to be hold in the inventory until the promised delivery time. So, in order to provide a high quality of service, the jobs must be delivered on time as much as possible. To do so, the completion time of the jobs in the revised schedule should be as close as possible to the ones in the original schedule. Therefore, the deviation of the job completion times between the original and revised schedule should be kept at minimum.

In rescheduling, to achieve high stability and high solution quality at the same time, the processing time decision plays an important role. In rescheduling literature, processing times are mostly assumed fixed and idle times are reserved in the original schedules to be able to absorb a disruption. However, in many industrial applications, such as in CNC metal cutting, processing times can be controlled by setting the parameters of the machine. By setting the processing speed or feed rate, processing time can be increased or decreased. Besides, setting the processing speed to control the processing time is directly related with the manufacturing cost. So, in order to keep the stability in rescheduling, processing time controllability is an important tool that we have, but it brings the compression cost consideration with it. Therefore, increasing the speed of feed rate of a machine via setting the parameters to control the processing times, results in higher manufacturing cost which is a measure of the schedule performance.

By compressing the processing times in the revised schedule, the completion times of the jobs could be made closer to their original completion times. If processing times would not be controlled, right shift scheduling would have to

be applied and the deviation of completion times would be increased. On the other hand, by being reallocated to a different machine, a disrupted job could be rescheduled so that its completion time in the revised schedule can be made closer to its original completion time. Although both methods can help to provide a stable revised schedule, they both incur additional cost which makes the schedule cost performance worse. Processing time controllability is utilized by compressing the processing times of the jobs which incurs compression cost. Reallocation of the jobs to different machines results in reallocation cost. Hence, while we get benefit of these methods, we have to consider the additional costs which incur as a result of these methods.

In rescheduling, another useful tool to keep high stability is reallocation of jobs which are being processed in the broken down machine in the original schedule, to another machine in the revised schedule. Reallocating the disrupted jobs to different machines other than the broken down machine brings us the flexibility of rescheduling the jobs so that their start and completion times deviate less from the original schedule. We get the chance of setting the start and completion times of the disrupted jobs on a different machine closer to their original start and completion times by reallocating them. But in rescheduling literature, reallocation cost is mainly neglected. However, in many manufacturing systems, tooling of the machines is done at the beginning according to the original schedule to utilize the machines efficiently. If a job has to be reallocated to a different machine due to a disruption, the machine that the job is reallocated has to be retooled. Additionally, transporting a job between the machines requires additional manpower or material handling. Therefore, retooling and transportation operations which are required to reallocate a job between machines bring with them the reallocation cost. Although reallocation is an important action in case of a disruption to have higher stability, it results in lower schedule performance which is measured in terms of additional cost of reallocation.

There is a trade-off between the stability measure which is the total absolute deviation of the completion times between the original and revised schedule and the schedule performance which is the cost of rescheduling. In the rescheduling literature, mostly, the problems with single objective are studied. Some of the

studies aim to minimize the cost while making processing time decisions and some of the studies are focused on the stability of the schedule while making scheduling decisions. Although, in many cases, the process planning and scheduling decisions are considered independently, in our study, cost performance and stability of the schedule are inversely correlated. While the stability is being kept higher by the compression of processing times and the reallocation, schedule cost performance decreases. On the other hand, while the compression and the reallocation are kept at minimum to provide higher cost, this results in the fail of the stability measure.

When the stability measure and the schedule performance are conflicted, it becomes more critical to make processing time, reallocation and sequencing decisions simultaneously. Although the problem becomes harder, since both of the stability and cost performance have to be considered and these criteria are in conflict, this gives us the flexibility of finding various alternative schedules with different cost performance and stability levels.

In this study, we present how processing time, reallocation and sequencing decisions can be made simultaneously to minimize the effects of this disruption on the original schedule.

We have to find an alternative solution to smooth the negative impacts of this disruption. Since the machines might be fully utilized initially, we may have to compress some of the processing times in order to add a new job to a machine or to reschedule the jobs on the disrupted machine after disruption. Consequently, we assume that the processing times are controllable within the given lower and upper bounds.

Our first objective is to minimize the sum of reallocation and compression costs. In our study, the compression cost is a convex function of the compression amount. Since the cost function is non-linear, it is hard to solve it by commercial solvers. We utilized the conic quadratic programming to solve the rescheduling problem with non-linear cost function. Since each machine is non-identical and jobs might have different operational requirements, compression cost is different for each job on each machine.

Moreover, as it is stated earlier, it is important to deliver the orders as close to original schedule as possible. Therefore, we try to maintain the original completion times as much as possible. So, the second objective is to minimize the total absolute deviations of the completion times in the revised schedule from the original completion times. Some of the completion times in the revised schedule might exceed the completion times of the original schedule and some of the jobs might be completed in the revised schedule earlier than they are completed in the original schedule. Therefore, our objective is to minimize the total absolute deviations of the completion times between original and revised schedule.

Therefore, in this study we aim to minimize both of the objective functions which are the total cost of reallocation and compression and the total absolute deviation of job completion times at the same time. We propose a mathematical model and a heuristic algorithm to solve this non-linear bi-criteria problem under different manufacturing environments.

Since we have two objectives, we handled the second objective by setting an upper bound and adding this bound as a constraint to the problem. By utilizing the second order cone programming, we can solve this mixed-integer non-linear mathematical model and obtain the optimal solution in terms of the first objective function for the given upper bound to the second objective function. For the cases where the exact approach requires excessive computation time, we also propose a local search based heuristic algorithm. By utilizing the heuristic algorithm, we can generate a set of solutions with varying total cost and TADC (total absolute deviation of completion times) values. For the non-dominated solutions among this set of solutions that we obtain by running the algorithm, we give the TADC values of these solutions to the mathematical model as an upper bound for the second objective function, and find optimal total cost values for these given upper bounds. According to our computational experiments, the proposed heuristic approach could obtain close solutions for the first objective for a given upper bound on the second objective in substantially decreased computation time.

In Chapter 2, we present a review of studies in the current literature. It covers the studies related to the rescheduling, controllable processing times, bi-criteria

problems on rescheduling and total absolute deviation of job completion times subjects. In Chapter 3, we will introduce the problem environment and problem definition and then give the mathematical model that we formulated to solve our problem. In Chapter 4, we first give theoretical properties which are extracted from the problem content and then we propose heuristics using these properties. We provide a numerical example using the algorithms proposed in this chapter and the mathematical model given in Chapter 3. In Chapter 5, we present the experimental factors and the results we obtained by solving the problems with the data generated by the combinations of these factors using the algoritms and the mathematical model we proposed. Finally in Chapter 6, we conclude with final remarks and the future search directions.

Chapter 2

Background

In the current rescheduling literature, the processing times are usually assumed fixed, although they can be controlled in many industrial applications. Before reviewing the studies on rescheduling parallel machines with controllable processing times, we will give a detailed literature review on the sub problems of our problem which are rescheduling, controllable processing times, stability measures and multi-objective rescheduling problems, separately on parallel machine environment.

2.1 Controllable processing times

Trick [29] considers the processing cost and makespan objectives with controllable processing times in a non-identical parallel machine environment with linear processing cost function. He shows the NP-hardness of the problem with a linear cost function.

Kayan and Akturk [17] determine the upper and lower bounds for the processing time of each job under controllable machining conditions. A set of discrete efficient points on the efficient frontier for a bi-criteria scheduling problem on a single CNC machine are found by using the proposed bounding scheme. There are two objectives to be considered; minimizing the manufacturing cost (comprised of machining and tooling costs) and minimizing makespan. They also develop an efficient frontier to establish a time/cost tradeoff for each manufacturing operation to link process planning and scheduling problems. By utilizing the proposed bounding mechanism, an exact algorithm and four heuristic approaches are developed to determine a set of discrete efficient points to approximate the continuous trade-off curve in a reasonable computation time.

Akturk and Ilhan [2] consider the scheduling of a set of jobs on a single CNC machine to minimize the sum of total weighted tardiness, tooling and machining costs by utilizing the controllable processing times. They develop an efficient dynamic programming (DP) based algorithm and indicate that there is a significant interaction between machining conditions and weighted tardiness problems and solving these two problems together gives very effective results in terms of cost of the system.

Mokhtari et al. [19] study on scheduling on a no wait job shop environment assuming the processing times are controllable. Their objective is to make optimal decisions on both the operation times and makespan. They divide the problem into three sub problems which are processing time decisions, sequencing and timetabling. For timetabling problem, they use a hybrid scheduling approach and they integrate this approach with the two-phase genetic algorithm that they propose to solve the processing time decisions and sequencing problems.

Daniels and Sarin [10] also treat the processing times as decision variables and study on the control of the processing times by additional resource allocation. They consider a joint sequencing and resource allocation problem and regarded the number of tardy jobs as the scheduling criterion. They present theoretical results for constructing the tradeoff curve between the number of tardy jobs and the total amount of allocated resource.

Shabtay and Steiner [28] give a unified framework for scheduling problems with controllable processing times by providing an up-to-date survey of the results. The quality of a solution for a scheduling problem with controllable processing times is measured by two criteria: The first one, F1, is a scheduling criterion dependent on the job completion times, and the second one, F2, is the resource consumption cost. They aim to minimize both criteria.

Leyvand et al. [18] also study scheduling problems on flexible environment and they assume both the job processing times and the delivery dates are controllable. They study a model of minimizing of scheduling costs which include the costs of due date assignment and tardiness, and the costs of controlling the job processing times as in Shabtay and Steiner [28]. But in this study, they consider the situations where these two costs are not comparable or additive. So, they consider these cost criteria as seperate and study problems of minimizing the weighted number of tardy jobs plus due date assignment cost and minimizing the total weighted resource consumption in scheduling a single machine.

Nearchou [21] studies the single machine scheduling problem of jobs with controllable processing times and compression costs. The aim of this study to minimise the total weighted job completion time and the cost of compression. Four population-based heuristics are developed to apply a multi-objective procedure to quantify the trade-off between the total weighted job completion times and the cost of compression.

Xu et al. [35] consider the problem of scheduling jobs with arbitrary release dates and due dates on a single machine. They assume that job-processing times are controllable and are function of nonlinear convex resource consumption. They present a branch and bound algorithm to determine simultaneously an optimal

processing permutation as well as an optimal resource allocation, such that no job is completed later than its due date, and the total resource consumption is minimized.

Later, Xu et al. [36] study on single machine scheduling considering controllable processing times and total tardiness. Processing times are function of nonlinear convex resource consumption. They present a polynomial time algorithm for the cases that the jobs have a common due date to obtain minimum total resource consumption with an optimal sequence as well as the optimal resource allocation, such that the total tardiness will not exceed a given limitation.

Gurel and Akturk [11] focus on CNC machines which is a well known industry application that allows controllable processing times. It is noted that there is a nonlinear relationship between the manufacturing cost and its required processing time on a CNC turning machine. This study considers the situation where both total weighted completion time and cost performance are under consideration for a CNC turning machine. In order to find a set of efficient solutions for this bi-criteria problem, a mathematical model for the total completion time case is presented first and optimality properties are derived. Then, by utilizing these properties, a new heuristic method to generate a set of approximate efficient solutions for the bi-criteria problem with the objectives of minimizing the manufacturing cost and the total weighted completion time is generated. This study integrates the process planning and scheduling decisions by considering job sequencing and processing time decisions simultaneously.

Gurel and Akturk [13] study on scheduling parallel machines with controllable processing times where the manufacturing cost of a turning operation is a non-linear convex function of its processing time. They aim to minimize manufacturing cost subject to a given total completion time level and give an effective formulation for this problem. Additionally, they present some optimality properties and propose an efficient heuristic algorithm to generate approximate non-dominated solutions.

Gurel and Akturk [12] deal with the optimal machine-job assignments and processing time decisions so as to minimize total manufacturing cost while the

makespan being upper bounded by a known value, they denote this as ϵ-constraint approach for a bi-criteria problem. They assume manufacturing cost of a turning operation is a non-linear convex function of its processing time and aim to minimize the total manufacturing cost objective for a given upper limit on the makespan objective. They consider both the makespan and total manufacturing cost objectives at the same time for a flexible machining environment and give several methods like a branch and bound algorithm, a beam search algorithm and an improvement search algorithm to find efficient solutions.

2.2 Rescheduling

Vieira et al. [32] present three primary types of studies from the rescheduling literature; methods for repairing a schedule that has been disrupted, methods for creating a schedule which is robust with respect to disruptions and studies of how rescheduling policies affect the performance of the dynamic manufacturing systems. They briefly discuss about these studies under the framework of rescheduling environment, rescheduling strategies, rescheduling policies and rescheduling methods. Then, they mention about the unexpected events which can change the system status and affect performance, they identify these events as rescheduling factors which are: machine failure, urgent job arrival, job cancellation, due date change, etc.

Sabuncuoglu and Goren [26] identify some types of response to an unexpected event in the system. One of them is rescheduling the operations of all the remaining jobs from scratch and the other one can be taking no corrective action and letting the system recover itself from the negative effects of disruptions. Between these two extremes, they identify another type of response which is repairing the schedules. They state that generating a matchup schedule can be a repair method and at some point in the future, the new schedule and the original one become the same or converge to each other by this method.

Bean et al. [8] consider the rescheduling with multiple resources when an

unexpected event prevents the use of a preplanned schedule. Whenever a machine breakdown occurs, reschedule is done to match-up with the preschedule at some point in the future. The match-up approach is compared with the no response policy and several dispatching rules. The results are obtained significantly better than results from pure static and dynamic strategies which are often used in practice. The problem is formulated as a dynamic program, and the state reached by the revised schedule is the same as that reached by the original schedule. They present heuristic procedures for solving the matchup problem which is called the Matchup Scheduling Algorithm (MUSA). The objective is to minimize weighted tardiness, summed over all jobs.

Akturk and Gorgulu [1] propose a new rescheduling strategy for a modified flow shop environment (MFS) and a match-up point determination procedure to increase both the schedule quality and stability. The proposed approach is compared with alternative reactive scheduling methods under different experimental settings. They assume that a production schedule is produced off-line and this preschedule then serves as the basis for the production planning decisions of other shop floor activities. The proposed new rescheduling approach is based on the idea of match-up scheduling which revises the reschedule after a machine breakdown. The objectives of the proposed heuristic are minimization of total tardiness of all jobs for a given match-up point on each machine under the assumption that one machine is not available for a certain period of time due to a machine breakdown. The study shows that the initial schedule has an important effect on the rescheduling problem, so, it should not be evaluated only by regular performance measures, but also by its inherent flexibility and robustness.

Sabuncuoglu and Bayız [25] study the reactive scheduling problems in a job shop environment and measure the effect of system size and load allocation (uniform and bottleneck) on the performance of off-line and on-line scheduling methods. They measure the performance of the system with the mean tardiness and makespan criteria. They also study on the partial scheduling under both deterministic and stochastic environments for several system configurations.

Sabuncuoglu and Karabuk [27] study the scheduling/rescheduling problem in

a multi-resource FMS environment. The purpose of this paper is to study the frequency of rescheduling in the multi-resource environment of a flexible manufacturing system (FMS) with random machine breakdowns and processing times. The authors propose several reactive scheduling policies to address the effects of machine breakdowns and processing time variations. The performance of the system is measured based on mean tardiness and makespan criteria. The results indicate that a periodic response with an appropriate period length would be sufficient to handle with interruptions. They also observe that machine breakdowns have more significant impact on the system performance than processing time variations.

2.2.1 Rescheduling Parallel Machines

Vieira, Herrmann and Lin [31] present new analytical models to predict the performance of rescheduling strategies for the parallel machine systems. Periodic, hybrid, and event-driven size rescheduling strategies are studied. Additionally, they present analytical models which require less computational effort than simulation models, and the results show that the models estimate important performance measures like average flow time, machine utilization, and average setup frequency, average rescheduling frequency, average schedule execution time, average setup time percentage, average processing time percentage, average repair time percentage and average idle time percentage.

The experimental results also show that the analytical models can accurately predict the performance measures, especially as the rescheduling period increases. Moreover, rescheduling period affects significantly both objectives of avoiding setups and reducing flow time which are conflicting objectives. They conclude that all three rescheduling strategies yield approximately equal system performance.

Alagoz and Azizoglu [5] study rescheduling caused by the change in machine eligibility constraints in parallel machines environment with total flow time objective. They consider total flow time as efficiency measure and the number of jobs processed on different machines in the initial and revised schedules as a

stability measure. They propose an LP model for the rescheduling problem of minimizing total flow time and then they present a branch and bound algorithm for the hierarchical problem of minimizing number of disrupted jobs subject to the constraint that total flow time is kept at its minimum level which is found by the LP model. Additionally, they propose heuristic procedures to generate a set of approximate efficient schedules with respect to the total flow time and number of disrupted jobs criteria.

Curry and Peters [9] study on rescheduling which is triggered by the arrival of new jobs to the system. They define the proportion of rescheduled jobs that change machine assignment as the measure of schedule nervousness. They examine the trade-off between schedule stability and tardiness cost. They develop rescheduling mechanisms that react to the arrival of new jobs to the system, but avoid unnecessary and excessive schedule nervousness by solving a NP-hard deterministic scheduling problem within a simulation with a branch and price algorithm or with a heuristic if run time restrictions are exceeded.

Azizoglu and Alagoz [7] study on identical parallel machines rescheduling results from the unavailability of a machine. They find solution procedures for finding the set of efficient schedules for the objectives of total flow time and number of jobs processed on different parallel machines in rescheduling. They measure efficiency in terms of the total flow time and measure stability in terms of the number of jobs processed on different machines in the original and new schedules. They propose a polynomial-time algorithm that finds a set of schedules that are efficient with respect to these two criteria.

Arnaout and Rabadi [6] introduce new repair and rescheduling algorithms for the unrelated parallel machine environment. The rules developed are respectively right shift repair, fit job repair, partial rescheduling, and complete rescheduling. In this study, schedule quality is measured based on C_{max} difference. C_{max} difference refers to the difference between the realized and predictable schedules makespan. Schedule stability is evaluated with match-up time and shifted jobs. Shifted jobs refer to the number of jobs that will be shifted from one machine to another.

Ozlen and Azizoglu [23] develop a branch and bound algorithm to generate all efficient solutions with respect to the total flow time and total disruption cost criteria. This is unique rescheduling study for unrelated parallel machines, whereas Ozlen and Azizoglu [24], optimize any non-decreasing function of these two criteria. They consider the efficiency measure as the total flow time, and the schedule deviation measure as the total disruption cost caused by the differences between the initial and current schedules. The disruption cost incurs if a job is assigned to different machines in the initial and current schedules. They provide polynomial-time solution methods to the following hierarchical optimization problems: minimizing total disruption cost among the minimum total flow time schedules and minimizing total flow time among the minimum total disruption cost schedules. Then they propose exponential time algorithms to generate all efficient solutions and to minimize a specified function of the measures.

2.2.2 Rescheduling parallel machines with controllable processing times

Akturk, Atamturk and Gurel [4] work with the controllable processing times. In their study, they face with the trade-off between match-up time and manufacturing cost objectives in a non-identical parallel machines environment. They aim to minimize three objectives; total manufacturing cost for jobs not yet started before disruption, sum of match-up time on the machines, maximum match-up time for new schedule and propose exact and heuristic solution approaches to find efficient solutions for two of three objectives. They conclude that improvement in one of these objectives is not possible without degrading the other objective. Controllable processing times have not been considered in the match-up time scheduling problems before this study.

Gurel, Korpeoglu and Akturk [14] study anticipative scheduling on a non-identical parallel machining environment, where processing times are controllable with a certain compression cost. When a disruption occurs in the initial schedule,

a match-up time strategy is utilized such that a repaired schedule has to catch-up initial schedule at some point in future. This requires changing machine−job assignments and processing times for the rest of the schedule, which implies increased manufacturing costs. In reactive scheduling problem, the objective is to minimize the rescheduling cost, subject to the constraint that the repaired schedule has to matchup with the initial schedule at a given time point after disruption.

Turkcan et al. [30] study on a machine scheduling problem with controllable processing times in a parallel-machine environment. The objectives are the minimization of manufacturing cost, which is a convex function of processing time, and total weighted earliness and tardiness. They assume that there are earliness and tardiness penalties and distinct due dates of jobs, and idle time is allowed. They first propose methods to find initial schedules in predictive scheduling. Then they revise these proposed methods to incorporate a stability measure for reacting to unexpected disruptions.

2.3 Multiple objectives

In scheduling literature; more than one objective at the same time are usually considered, and generally one is minimizing cost and one is a scheduling objective. The aim of the process planning decisions is generally to minimize the manufacturing cost, on the other hand scheduling decisions focus on a scheduling criterion. When these two decisions are integrated, cost and scheduling critera should be considered simultaneously. Gurel and Akturk [12] consider minimizing total manufacturing cost subject to a bound on the makespan objective in non-identical parallel machines. Shabtay and Steiner [28] consider minimizing a scheduling criterion dependent on the job completion times, and the resource consumption cost. Kayan and Akturk [17] try to minimize the manufacturing cost and minimize makespan. Trick [29] also considers the processing cost and makespan objectives. Gurel and Akturk [11] handle with two criteria, total weighted completion time and cost performance. Yedidsion et al. [37] handle a

single machine scheduling problem to minimize maximum lateness and resource consumption simultaneously. Jansen and Mastrolilli [16] try to minimize the processing cost and makespan with controllable processing times on identical parallel machines and they contribute by presenting new polynomial time approximation algorithms. Akturk and Gorgulu [1] minimize the job tardiness and the matchup point. Curry and Peters [9] consider total disruption cost and total tardiness. Azizoglu and Alagoz [7] try to minimize total flow time and number of jobs processed on different parallel machines in rescheduling results from the unavailability of a machine. Ozlen and Azizoglu [23] handle with the total flow time and total disruption cost criteria. Akturk, Atamturk and Gurel [4] consider match- up time and manufacturing cost objectives.

2.4 Total Absolute Deviation of Job Completion Times

Total absolute deviation of job completion times (TADC) is a commonly used stability measure. It is the measure of deviation between the original and revised schedule.

Huang and Wang [15] consider parallel identical machines scheduling problems with deteriorating jobs whose processing times are the function of their start times. They focus on minimizing total absolute differences in completion times (TADC) and total absolute differences in waiting times (TADW).

Wang and Wei [33] consider identical parallel machines scheduling problems with a deteriorating maintenance activity. In this model, each machine has a deteriorating maintenance activity which means that delaying the maintenance increases the time required to perform it. They also focus on minimizing total absolute differences in completion times (TADC) and total absolute differences in waiting times (TADW).

Oron [22] studies on a single machine scheduling problem with simple linear deterioration. Job processing times are the simple linear function of a job-dependent growth rate and the job starting times. He aims to minimize the total absolute deviation of completion times (TADC) to find the optimal schedule.

Mor and Mosheiov [20] show that minimizing TADC is polynomially solvable under the position-dependent processing times assumption on uniform and unrelated machines and for a bi-criteria objective consisting of a linear combination of total job completion times and TADC.

Wang and Xia [34] study a single machine scheduling problem in which job processing times are controllable variables with linear costs. They focus on two goals separately, minimizing a cost function containing total completion time, total absolute differences in completion times and total compression cost; minimizing a cost function containing total waiting time, total absolute differences in waiting times and total compression cost.

2.5 Summary

In this chapter, we presented a review of studies related to the rescheduling, controllable processing times, bi-criteria problems on rescheduling and total absolute deviation of job completion times subjects. We observed that these subjects are studied in the literature before with different perspectives. In the next chapter, we will give the problem definition and the mathematical model.

Chapter 3

Problem Environment and Modeling

In our problem environment, there are non-identical parallel machines at which the jobs will be processed. These machines can process the jobs concurrently.

3.1 Problem Definition

Although most scheduling studies assume that the processing times of the jobs are known and fixed, in many manufacturing applications the processing times can be controlled and changed. The processing time of a job can be changed within a lower and upper bound limits by compressing or decompressing the processing time.

On CNC machines, compression and decompression of processing times are applied to the jobs via setting machining parameters such as machining speed and feed rate. These actions cause changes in the tooling cost. Decreasing the processing time of a job exerts more force on the machine and it results with an additional tooling cost. Machine spends more effort to process the job within a

smaller time interval than to process the job within the time interval of upper bound of processing time. Additional tooling cost, which is caused by the compression of the processing time of a job, is due to increased cutting speed and feed rate. On the other hand, if the machine processes the job within a time interval longer than its original processing time, it spends less effort and additional tooling cost, which is caused by the decompression of the processing time of a job. Additional tooling cost becomes negative, since decreased cutting speed and feed rate requires less tooling cost.

We calculate the impact of the compression and decompression of the processing times in terms of additional cost. Each job has a different cost function and different processing time upper and lower bounds.

In our problem, we are given an original schedule. We assume that this schedule is formed by manufaturing cost minimization. In this schedule we have machine-job assignments and job sequence for each machine. Each job is processed at its processing time upper bound. We assume that one of the machines is disrupted at time t. This disruption can be caused by a machine breakdown, delay in the arrival of resources or power loss, etc.

We have to take an action so that this disruption can be compensated as much as possible. The action should be revising the schedule. The schedule until the disruption should stay the same. After the disruption time, the remaining jobs will be considered to be rescheduled. The job which was being processed at the broken down machine at time t will be started processing again after the disruption. That is, preempt-repeat strategy will be applied at the disrupted machine. All other jobs which have already started are considered to be completed at their original machines. We consider the rescheduling of remaining jobs and disrupted job in order to catch the original schedule as soon as possible by assigning the jobs to different machines or compressing their processing times.

Each machine has an earliest start time for the rescheduling problem. For the disrupted machine, since the first disrupted job is going to be processed from scratch, the earliest start time is the end time of the disruption. For other machines, since the jobs which are started to be processed before the disruption

time considered to be completed, the completion of processing these jobs will be waited and the earliest start time is going to be the completion time of the first job completed after the disruption start time. That is, it is the minimum of all the start times at this machine after the disruption.

There are some restrictions that should be considered while creating the revised schedule. Until the disruption ends, disrupted machine is unavailable and cannot process any job. So, the jobs, which are originally processed at the disrupted machine during the disruption length, cannot be processed by that machine during the disruption and has to wait until the disruption ends. It causes delays in the completion times of the orders.

Another restriction is the maximum completion times of the machines. Even the jobs would be shifted right in the schedule without considering the delay in the delivers, the machine has an available machining time which is until the maximum completion time of that machine and we cannot go beyond this time. The maximum completion time of each machine is the original completion time of the last job in the original schedule on that machine. Revised schedule has to match the maximum completion time of each machine.

Furthermore, we have to consider that when we revise the schedule we have to comply with the constraint that in the revised schedule, the start time of a job should be greater than or equal to the original start time of that job. Because we assume that a job cannot be available earlier than its original start time, in the revised schedule, start time of a job cannot be smaller than the original start time of that job. Because, we assume that the job cannot be available earlier than the original start time of itself.

Therefore, we have to find an alternative solution. We can reallocate a disrupted job to another machine or we can keep the disrupted job at its original machine, but to process it after the disruption. Since we should matchup the maximum completion time of a machine in the revised schedule, it may not be possible to add a new job to that machine or to shift the jobs after disruption without changing the processing times of the jobs. Therefore, even if we either reallocate a job to a different machine or shift the job after the disruption at its

original machine, we may have to change the processing times of the jobs on the affected machines.

If we reallocate a job to a different machine, this new machine is an affected machine as well as the disrupted machine. The processing times can be changed by being compressed or being decompressed as mentioned earlier. Originally, all the jobs are being processed at their processing time upper bounds. Thus, when we add a new job to a machine or shift the jobs after disruption, we have to adjust the processing times so that the sum of the processing times in the revised schedule is equal to the difference between earliest start time of the machine and the maximum completion time of the machine. The alternatives to revise the schedule are to reallocate the disrupted jobs to different machines and to change their processing times or to shift the disrupted jobs to the right after disruption at the disrupted machine and to change their processing times. But both alternatives result with an additional cost. Reallocating the jobs brings with itself a reallocation cost. Because reallocating a job means that transferring the job from a machine to a different machine. It requires material handling or additional machine work, hence an additional handling cost. In some cases, we assume that, the machines are parallel and this reallocation cost increases linearly with the increase in the distance between the machines. That is, if there are three machines, while the distance between the first and second machine and the distance between the second and third machine is one unit, the distance between the first and third machine is 2 units. As the distance increases, reallocation cost increases. On the other hand, this reallocation cost can be considered fixed between the machines. In this case, the reallocation cost between the 1^{st} machine and the 2^{nd} machine is considered to be the same with the reallocation cost between the 1st machine and the 3^{rd} machine.

Changing the processing times brings some additional cost. Because the jobs are originally at their processing time upper bounds, to be able to fit the sum of the processing times of the jobs when we add a new job to a machine or we shift the job to the right after the disruption, we have to decrease the processing time of some of the jobs. To do so, we have to compress some of the processing time of the jobs. As it is mentioned earlier, it requires an additional tooling cost.

When we find the alternative solution to matchup the maximum completion times of the machines, we aim to minimize the cost which results from reallocation and compression. Therefore, our first objective is to minimize the total of these costs.

Moreover, as it is stated earlier, it is important to deliver the orders as soon as possible. We have to decrease the impacts of the disruption on the order delays. The other important thing is that we also have to avoid the early deliveries. Although the job start times in the revised schedule are restricted to be greater than or equal to the job start times in the original schedule, revised completion times of some of the jobs may be smaller than the original completion times because of either compression or change in the sequence. This causes carrying the inventory and this also results in additional cost.

Therefore, we try to maintain the original completion times as much as possible. Thus, the second objective is to minimize the total absolute deviations of job completion times in the revised schedule from the original completion times.

3.2 Mathematical Modeling

In order to solve this problem, we propose a bi-criteria non-linear mathematical model. The notation that is used in the mathematical model and the formulation of the model are as follows:

Decision Variables

x_{ij} : 1 if job i is assigned to machine j in the revised schedule, else 0

z_{ikj} : 1 if job i precedes job k on machine j in the revised schedule, else 0

$\tilde{S}_{ij}$: start time of job i on machine j in the revised schedule

$\tilde{p}_{ij}$: processing time of job i on machine j in the revised schedule

$\tilde{C}_i$: completion time of job i in the revised schedule

w_{ij} : compression amount of processing time of job i at machine j

$f_{ij}(w_{ij})$: cost function of compression amount of processing time of job i on machine j

Parameters

y_{ij} : 1 if job i is originally assigned to machine j, else 0

c_{ij} : manufacturing cost of job i on machine j

S_i : original start time of job i

p_{ij} : original processing time of job i at machine j

pu_i : processing time upper bound of job i

pl_i : processing time lower bound of job i

b_{ij}, k_{ij} : compression cost coefficients for job i on machine j, $b \geq 0$, $k \geq 0$

D_j : maximum completion time of machine j

γ_j : capacity of machine j

rt_j : ready time of machine j in the revised schedule

d_{jk} : reallocation cost of reallocating a job between machines j and k

C_i : original completion time of job i

B : bound for the second objective function

m : number of machines

Definition

$[n]$: n^{th} positioned job on a machine j where $j = 1 \ldots m$

Sets

N : set of all jobs

$\mathcal{N}$: set of jobs to be rescheduled

$\mathcal{J}$: set of machines

$\mathcal{P}$ = $\{(i,j) : y_{ij} = 1\}$

$F1 : \min \ \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{J}} f_{ij}(w_{ij}) + \sum_{(i,j)\in\mathcal{P}} \sum_{k\in\mathcal{J}} (d_{jk}\ x_{ik})$

$F2 : \min \ \sum_{i\in\mathcal{N}} \left|\tilde{C}_i - C_i\right|$

subject to

$$\sum_{j\in\mathcal{J}} x_{ij} = 1, \qquad \forall\ i \in \mathcal{N}, \quad (3.1)$$

$$\tilde{S}_{ij} \geq rt_j - (rt_j + 1)(1 - x_{ij}), \qquad \forall\ i \in \mathcal{N}, \forall\ j \in \mathcal{J}, \quad (3.2)$$

$$\tilde{S}_{ij} \geq \tilde{S}_{kj} + \tilde{p}_{kj} - (D_j + 1)(1 - z_{kij}), \forall\ i \in \mathcal{N}, \forall\ k \in \mathcal{N},\ k \neq i, \forall\ j \in \mathcal{J}, \quad (3.3)$$

$$\tilde{S}_{kj} \geq \tilde{S}_{ij} + \tilde{p}_{ij} - (D_j + 1)(1 - z_{ikj}), \forall\ i \in \mathcal{N}, \forall\ k \in \mathcal{N},\ k \neq i, \forall\ j \in \mathcal{J}, \quad (3.4)$$

$$x_{ij} + x_{kj} \geq 2(z_{ikj} + z_{kij}), \qquad \forall\ i \in \mathcal{N}, \forall\ k \in \mathcal{N},\ k \neq i, \forall\ j \in \mathcal{J}, \quad (3.5)$$

$$x_{ij} + x_{kj} \leq z_{ikj} + z_{kij} + 1, \qquad \forall\ i \in \mathcal{N}, \forall\ k \in \mathcal{N},\ k \neq i, \forall\ j \in \mathcal{J}, \quad (3.6)$$

$$z_{ikj} + z_{kij} \leq 1, \qquad \forall\ i \in \mathcal{N}, \forall\ k \in \mathcal{N},\ k \neq i, \forall\ j \in \mathcal{J}, \quad (3.7)$$

$$\tilde{S}_{ij} \leq (D_j + 1)(x_{ij}), \qquad \forall\ i \in \mathcal{N}, \forall\ j \in \mathcal{J}, \quad (3.8)$$

$$\tilde{p}_{ij} \leq (D_j + 1)(x_{ij}), \qquad \forall\ i \in \mathcal{N}, \forall\ j \in \mathcal{J}, \quad (3.9)$$

$$\sum_{j\in\mathcal{J}} (\tilde{S}_{ij} + \tilde{p}_{ij}) = \tilde{C}_i, \qquad \forall\ i \in \mathcal{N}, \quad (3.10)$$

$$\tilde{p}_{ij} = pu_i\ x_{ij} - w_{ij}, \qquad \forall\ i \in \mathcal{N}, \forall\ j \in \mathcal{J}, \quad (3.11)$$

$$\sum_{j\in\mathcal{J}} \tilde{S}_{ij} \geq S_i, \qquad \forall\ i \in \mathcal{N}, \quad (3.12)$$

$$\sum_{j\in\mathcal{J}} \tilde{p}_{ij} \geq pl_i, \qquad \forall\ i \in \mathcal{N}, \quad (3.13)$$

$$\sum_{j\in\mathcal{J}} \tilde{p}_{ij} \leq pu_i, \qquad \forall\ i \in \mathcal{N}, \quad (3.14)$$

$$\tilde{S}_{ij} + \tilde{p}_{ij} \leq D_j, \qquad \forall\ i \in \mathcal{N}, \forall\ j \in \mathcal{J}, \quad (3.15)$$

$$\tilde{S}_{ij}, \tilde{p}_{ij}, w_{ij}, \tilde{C}_i, \tilde{C}_{ij} \geq 0, \qquad \forall\ i \in \mathcal{N}, \forall\ j \in \mathcal{J}, \quad (3.16)$$

$F1$ is the first objective function which aims to minimize the sum of total reallocation cost and compression cost. In order to calculate the total reallocation cost, the term $d_{jk}\ x_{ik}$ is summed over the (i,j) machine-job pair set which is formed by the machine-job pairs of the original schedule. So if (i,j) is a pair in the original schedule, that is job i is assigned to job j in the original schedule and if in the revised schedule, job i is assigned to machine k which is different from machine j, reallocation cost between machine j and k is added to the total reallocation cost. If the machines j and k are same, then the reallocation cost is 0 between these machines. Total compression cost is calculated by summing the cost function $f_{ij}(w_{ij})$ over all jobs and machines. The cost function $f_{ij}(w_{ij})$ is equal to $b_{ij}\ (w_{ij}^{k_{ij}})$ where b_{ij} and k_{ij} are compression cost coefficients.

$F2$ is the second objective function which aims to minimize the total absolute values of the completion time differences. If we try to keep the processing times closer to their upper bounds to minimize the compression cost in the revised schedule, since all the jobs will move towards to the right of the timeline, the second objective will get worse. Therefore, we cannot minimize both objectives $F1$ and $F2$ at the same time. So, the problem is to generate an efficient solution set. A solution $(F1(x), F2(x))$ is efficient if there does not exist another solution $(F1(y), F2(y))$ such that $F1(y) \leq F1(x)$ and $F2(y) \leq F2(x)$ and one inequaliy is strict. Therefore we try to keep $F2$ at a given maximum level and find efficient solutions for this level of $F2$. In order to do this we give a bound B to the $F2$, remove $F2$ from the objectives and add the constraint $\sum_{i \in \mathcal{N}} \left| \tilde{C}_i - C_i \right| \leq B$ to the constraint set.

Constraint (3.1) ensures that each job should be assigned to a machine. Constraint (3.2) requires that if the original start time of a job is greater than or equal to disruption time, start time of that job should be greater than or equal to the ready time of the machine which the job is assigned to.

Constraints (3.3)-(3.7) are disjunctive constraints, which provide that no two jobs can be operated on the same machine simultaneously. Constraints (3.8) and (3.9) force the start time and the processing time of a job to be equal to zero for the machines which that job is not assigned to, respectively. Constraint

(3.10) calculates the completion time of a job by summing the start time of that job and the processing time of that job. Constraint (3.11) implies that the compression amount of a job is equal to the difference between the upper bound of the processing time of that job and the processing time of that job. Constraint (3.12) guarantees that the start time of a job on the machine that the job is assigned to is greater than or equal to the original start time of that job.

Constraint (3.13) and (3.14) ensure that the processing time of a job should be between the lower bound and the upper bound of the processing time of that job. Constraint (3.18) provides that completion time of any job cannot be greater than the maximum completion time of the machine which the job is assigned to. Constraints (3.16) are the nonnegativity constraints.

Since the cost function in the first objective function $F1$ is non-linear, we reformulate the mathematical model in order to handle this non-linearity. Model is put into conic optimization problem with linear objective and conic constraints. In order to do this, we replace each term $b_{ij}\ (w_{ij}^{k_{ij}})$ in the objective $F1$ with an auxiliary variable $t_{ij} \geq 0$ and add $b_{ij}\ (w_{ij}^{k_{ij}}) \leq t_{ij}$ into the constraints. After the reformulation, the constraints are strengthened and can be represented as conic quadratic constraints as in Akturk, Atamturk and Gurel [3]. The reformulated model is as follows:

$$\begin{aligned}
&\min \ \textstyle\sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{J}} t_{ij} \\
&\text{subject to} \\
&\quad \text{(3.1)-(3.16)} && (3.17)\\
&\quad b_{ij}\ (w_{ij}^{k_{ij}}) \leq t_{ij}, \ \forall\ i \in \mathcal{N}, \forall\ j \in \mathcal{J}, && (3.18)\\
&\quad t_{ij} \geq 0, \qquad\qquad \forall\ i \in \mathcal{N}, \forall\ j \in \mathcal{J}, && (3.19)
\end{aligned}$$

Trick [29] assumed that processing times are controllable and focused on the processing cost and makespan objectives. He showed the NP-hardness of the problem in a non-identical parallel machine environment with linear processing cost function. Therefore, our problem with nonlinear cost function of processing time compression is also NP-hard and we propose heuristics to solve this problem in the next chapter.

3.3 Summary

In this chapter, we introduce the problem environment and then give the mathematical model. We formulate a nonlinear bi-criteria mathematical model to solve our problem and then reformulated it into conic quadratic programming. We conclude that our problem is NP-hard and we will propose a local search heuristic in the next chapter.

Chapter 4

Proposed Heuristic Algorithm

It is hard to solve the mathematical model given in Chapter 3, as it involves discrete decision variables with nonlinear objective function. So, we developed a heuristic algorithm to solve the problem in a reasonable computation time.

4.1 Theoretical Properties

We extracted some properties from the problem considerations and utilized them while developing the algorithm. These properties are given below:

Rule 1. *The processing times and the sequence of the jobs on a machine do not change in the revised schedule, if it is not a disrupted machine and if no jobs are removed from or added to this machine.*

Justification: The jobs are scheduled to be processed at their upper bounds in the original schedule, and the processing time of a job cannot exceed its processing time upper bound. Any change in the processing time of a job can only be achieved by compression and even a small amount of compression incurs compression cost and deviation of completion times and any change in the sequence also incurs deviation of completion times.

Rule 2. *If a disruption occurs on a machine and all the jobs after the disruption become late since the schedule is right shifted, compressing the processing time of the jobs which are sequenced earlier gives better results in terms of TADC.*

Justification: Let
$n_{[r]}$*: number of jobs that succeed the* r^{th} *positioned job on the machine, that the job is assigned to, including the job itself*
$F2$*: TADC objective function value*
$w_{[r]}$*: compression amount on the processing time of the* r^{th} *positioned job*

One unit of compression on processing time of r^{th} positioned job on a machine results in the gain of $n_{[r]}$ units in TADC value.

If $w_{[r]} = \delta$, then $\Delta F1 = -\delta \times n_{[r]}$. Let us assume that there are two jobs whose positions are v and y such that $v \leq y$.

If $v \leq y$, then $n_{[v]} \geq n_{[y]}$, $\Delta F1^v \geq \Delta F1^y$. As it can be seen in Figure 4.1,

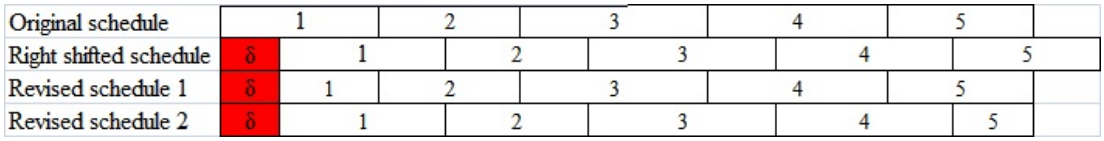

Figure 4.1: Right shift scheduling and rescheduling by compressing the processing times

when there is a disruption for δ unit of time, if we do not compress the processing times of the jobs and right shift the jobs due to the disruption, we have 5δ units of TADC and we violate the maximum completion time constraint. In order to match-up the maximum completion time of the machine, we have to compress the processing times of the jobs on this machine for δ unit.

If we apply total of the δ unit of compression on the 1^{st} positioned job as it can be seen in the Revised Schedule 1 in Figure 1, TADC value becomes 0, that is we gain 5δ units of TADC value compared to the right shifted schedule since $n_{[1]} = 5$.

On the other hand, if we apply total of the δ unit of compression on the last

job (5^{th} positioned job) as it can be seen in the Revised Schedule 2 in Figure 1, TADC value becomes 4δ, that is we gain only δ units of TADC value compared to the right shifted schedule since $n_{[5]} = 1$.

Rule 3. *In order to decide the sequence of two jobs on the same machine, we propose a slope criterion, which is the marginal change in the cost when we compress the processing time of a job by one more unit. We calculate this slope criterion for each job, separately. We sequence the jobs in the order of their slopes starting from the smallest one.*

Justification: Let us assume that the compression on processing time of job i on machine j is w_{ij}. Then, marginal change in the compression cost is;

$$\partial f_{ij}/\partial w$$

If we compress the processing time of a job by Δ units, a lower bound on the compression cost is

$$\Delta \times \partial f_{ij}/\partial w$$

Then we compare the slopes of the jobs. It means that if we compress the processing times of these jobs by same amount of additional units, compressing the processing time of the job with larger slope costs more than the cost of compressing the processing time of the job with smaller slope. Since compressing the processing time of the jobs which are sequenced earlier gives better results in terms of TADC (see Rule 2), also in order to get better results in terms of the compression cost we sequence the job with the smaller slope early on the machine.

Rule 4. *For each job, we find the number of succeeding jobs on the same machine in the revised sequence. This number is called as $afterJobs$ value of a job. Then, we sum $afterJobs$ values over each job and find the proportion of compression amount for each job by dividing their $afterJobs$ value to this sum. Find the required compression amounts for each job by multiplying the total required compression amount with the proportion of $afterJobs$ value of each job over this sum.*

4.2 Decision Tree Algorithm

In our heuristic algorithm, we apply a search algorithm on a decision tree structure which will be denoted as Decision Tree Algorithm (DTA) during the study. This method helps us to find an efficient frontier. In the decision tree, we start with a solution at the root node and we generate a number of child nodes from the root node. Each node represents a different solution. We call the total number of child nodes of one node in the tree as *branchSize*. We evaluate the child nodes of the root node and eliminate some of these nodes. Then, the selected nodes are used to generate new child nodes in the next level of the tree. We call the number of nodes that we select to generate the next level of the tree with their child nodes as *beamSize*. In each level of the tree, we generate new child nodes for each selected node of the previous level in *branchSize*. Then, we evaluate all the child nodes of that level and select the best nodes in *beamSize* to generate new child nodes by filtering the remaining nodes. We generate the next level of the tree with the child nodes of these selected nodes. In Figure 4.2, the white nodes represent the nodes selected and striped nodes represent the nodes fathomed. In the last level, the red nodes represent the non-dominated solutions.

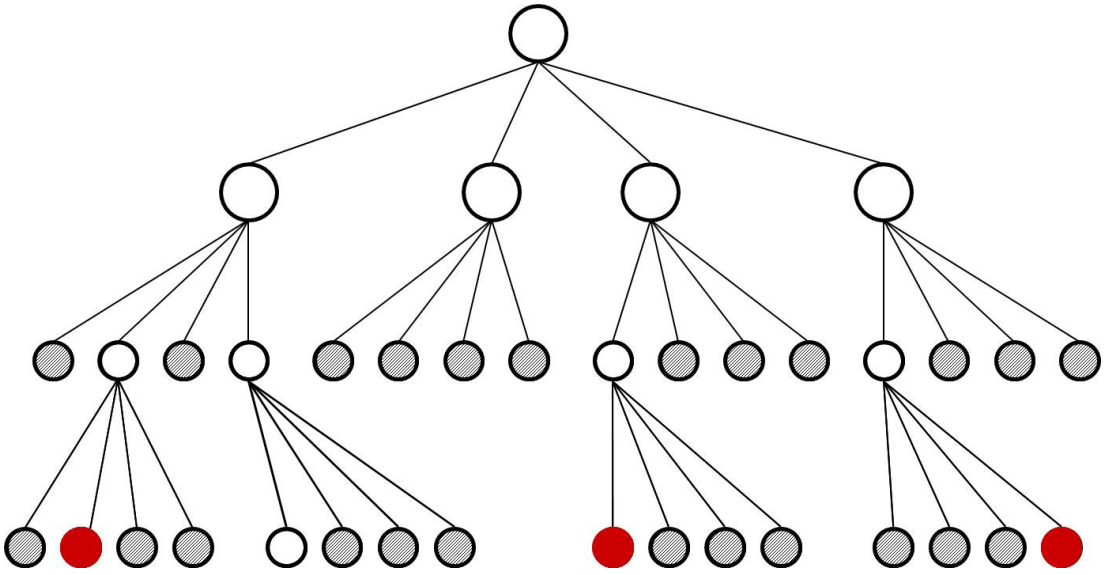

Figure 4.2: Tree structure in the algorithm

In our algorithm, we start with the original schedule in the root node by

considering the disruption duration as a blocked time. At each level of the tree, we insert a disrupted job from the disrupted job list DJ to one of the machines. The number of levels in the tree is equal to the number of disrupted jobs. In each child node, we fix the schedule of its parent node and try to insert a disrupted job, which is considered at that level of the tree, to the schedule by doing "move" and "swap" operations. In $Move(k, l, n)$ operation, job k is inserted to the position n of machine l. In $Swap(k, l, s)$ operation, job k is swapped with the job s which is at the first position of machine l. The details of these operations will be explained in the steps of the DTA. By utilizing Rule 2, we try to move a disrupted job to the early positions in the sequence. So, we try just two earliest positions of each machine to move a disrupted job to and try to swap a disrupted job with the first positioned job of each machine. We try just one position which is the earliest position of the disrupted machine to move a disrupted job to. Because, since the earliest position of the disrupted machine is after the end time of the disruption, even this position is late of the disrupted job in terms of TADC. Therefore, we branch three child nodes for each undisrupted machine (two for move operations and one for swap operation) and one child node for the disrupted machine (for move operation) from one parent node. The $branchSize$ of the DTA is $(3 \times (m-1)+1)$. Then we select the nodes generated within a level in $beamSize$ b and go to the next level with the number of nodes b.

We first give the additional notation that will be used throughout the algorithm and then start with the steps of the algorithm.

Notation

$$
\begin{aligned}
EST_j &: \text{earliest start time of machine } j \\
\tilde{j} &: \text{disrupted machine} \\
DJ &= \{i : t \leq S_i \leq t + disr\} \text{ and } \left\{i : p_{i\tilde{j}} > 0\right\} \\
L_j &= \{i : S_{ij} \geq EST_j\} \bigcap \{i : p_{ij} > 0\} \text{ for } j = 1 \ldots m \\
|n| &: n^{th} \text{ positioned job in } L_j \text{ where } j = 1 \ldots m
\end{aligned}
$$

STEP 1: Find EST_j for $j \in J$ (For disrupted machine, end time of the disruption, for other machines, smallest start time which is greater than or equal to the disruption start time.)
STEP 2: Start with the job $k = argmin\{S_i\}$ where $i \in DJ$.
STEP 3: $Move(k, l, |n|)$ for $k \in DJ$, $l \in J$ and $n = 1, 2$.

There are four alternatives to insert the disrupted job k on machine l; if $n = 1$, job k is inserted just before or just after the first job in L_l, if $n = 2$, job k is inserted just before or just after the second job in L_l. First job in L_l is the job which has the smallest start time among the jobs in L_l, second job in L_l is the job which has the second smallest start time among the jobs in L_l.

STEP 3.1: Check if $Move(k, l, |n|)$ is feasible, if not, then fathom this node.
(Set $\tilde{p}_{il} = pl_i$ where $i \in L_l$ and $i = k$
If($\sum_i \tilde{p}_{il} \leq D_l - EST_l$) where $i \in L_l$ and $i = k$, then it is feasible
Else moving the disrupted job to this machine is infeasible.)

STEP 3.2: Set the start time of the disrupted job k at machine l ($\tilde{S}_{kl}$).

In order to compress job k more than other jobs on machine l and utilize Rule 2, we try to insert job k to the earliest positions in the revised schedule.

In order to decide where to insert job k, we calculate the slope of job k and the n^{th} job in L_l according to Rule 3 and we sequence the job with the smaller slope just before the other one.

If we insert job k just before the n^{th} job in L_l, we set the start time of job k in the revised schedule to the original start time of the n^{th} job of machine l and update L_l such that $L_l = L_l \cup \{k\}$, otherwise we leave the start time of the n^{th} job of machine l same in the revised schedule and set the start time of job k in the revised schedule to the completion time of the n^{th} job of machine l in the revised schedule and update $L_l = L_l \setminus \{|n|\} \cup \{k\}$.

In a move operation, we sequence the disrupted job k and the n^{th} job in L_l

according to Rule 3, and we do not change the sequence of other jobs on machine l.

For the machines other than machine l, we keep the schedule same with the schedule of the parent node as we explained in Rule 1.

For machine l, we set the processing times of all jobs in L_l and job k to their upper bounds and sum these upper bounds.

STEP 3.3: Sum the processing time upper bounds of jobs in (L_l)

$$SUM_l = \sum_i pu_i \text{ where } i \in L_l$$

If $SUM_l \leq$available time on machine l, fix the processing times to their upper bounds and go to STEP 3.7.
Otherwise, go to STEP 3.4 to find the required compression amount.
STEP 3.4: Find the required compression amount which is the difference between this sum and the time between maximum completion time and start time of job k on machine l

$$EXCESS_l = D_l - \acute{S}_{kl} - SUM_l$$

STEP 3.5: Distribute the required compression amount to the jobs in L_l according to the Rule 4.
STEP 3.6: Subtract the required compression amount distributed to each job (w_{il} where $i \in L_l$) from their processing times set at the previous level and calculate new processing times of the jobs

$$\tilde{p}_{il} = p_i - w_{il} \text{ where } i \in L_l$$
If $\tilde{p}_{il} < pl_i$ then $\tilde{p}_{il} = pl_i$ where $i \in L_l$

STEP 3.7: Calculate new start times and completion times of the jobs in L_l.
STEP 3.8: If the completion time of last job at the machine l exceeds the maximum completion time of this machine, apply maximum completion time feasibility rule.

Maximum completion time feasibility rule: Although we calculate the required

compression amount of each job such that the sum of the processing times of jobs should be equal to $D_l - \acute{S}_{kl}$, after we subtract the required compression amount of each job from their processing times, completion time of the last job may be greater than or equal to the maximum completion time of the machine. The reason is that if the processing time of a job after compressing it goes below its lower bound, we set it at its lower bound. Since the compression amount of the jobs early in the sequence is more than the compression of succeeding jobs, the processing times of these jobs are closer to their lower bounds than the succeeding jobs are. So, we start from the last job in the sequence of machine l and subtract the over time amount, which is the difference between the completion time of last job at a machine and the maximum completion time of that machine, from the processing time of last job. If it goes below the lower bound of the last job, fix the processing time of last job at its lower bound and go to the preceding job of last job, and subtract the remaining over time amount from the processing time of this job and check the lower bound, if it goes below the lower bound, go to the preceding job of this job and go on until the over time amount becomes 0.

The earliest start time of the machine becomes the completion time of the disrupted job.

> **STEP 3.9:** $L_l = L_l \setminus \{k\}$ and update the number of jobs and earliest start times of all machines and update the positions of all jobs at all machines

Updating the earliest start times:

$EST_j = EST_j \;\; \forall j = 1 \ldots m$ and $j \neq l$

$EST_l = \tilde{C}_{kl}$

Updating the position of the jobs: The positions of the jobs at the machines other than machine l remain the same.

At machine l;

If $\tilde{C}_{kl} \leq \tilde{S}_{|1|l}$ then $|v| = |v|$

If $\tilde{C}_{|1|l} \leq \tilde{S}_{kl}$ or $\tilde{C}_{kl} \leq \tilde{S}_{|2|l}$ then $|v-1| = |v|$

If $\tilde{C}_{|2|l} \leq \tilde{S}_{kl}$ then $|v-2| = |v|$

We do not change the schedules of the other machines (Rule 1). At each level, we calculate the additional compression cost and reallocation cost for each job. That is, we subtract the required compression amount from the processing time upper bound of each job and calculate its compression cost. Afterwards, we find the cost of compressing the processing time of the job from its upper bound to its processing time value at the beginning of that level and find the difference between these two costs as the additional compression cost of that job at that level. Reallocation cost of the disrupted job incurred by this move operation is added to the additional compression cost and this is the additional cost of the node in which this move operation is done.

STEP 4: $Swap(k, l, s)$ for $k \in DJ$, $l \in J$, $s \in L_l$ and $s = |1|$

Another operation used in the DTA is swap operation. In swap operation, we exchange two jobs between two machines. We try to insert the disrupted job k to the position of job s on machine l in the original schedule and insert the job s before the first job in $L_{\tilde{j}}$ on machine $\tilde{j}$.

STEP 4.1: Check the feasibility of moving job k to the machine l and job s to the disrupted machine $\tilde{j}$, if it is infeasible, then fathom this node

Checking the feasibility:

Set $\tilde{p}_{il} = pl_i$ where $i \in L_l$ and $i = k$ and $\tilde{p}_{i\tilde{j}} = pl_i$ where $i \in L_{\tilde{j}}$ and $i = s$

If $(\sum_i \tilde{p}_{il} \leq D_l - EST_l)$ where $i \in L_l$ and $i = k$ and $(\sum_i \tilde{p}_{i\tilde{j}} \leq D_{\tilde{j}} - EST_{\tilde{j}})$ where $i \in L_{\tilde{j}}$ and $i = s$, then it is feasible.

Else swapping the disrupted job k and job s is infeasible, fathom this node.

STEP 4.2: $L_l = L_l \backslash \{s\} \cup \{k\}$ and $L_{\tilde{j}} = L_{\tilde{j}} \backslash \{k\} \cup \{s\}$

STEP 4.3: Set the start time of job s and job k ($\tilde{S}_{kl}$, $\tilde{S}_{s\tilde{j}}$)

Setting the start time:

$\tilde{S}_{s\tilde{j}} = EST_{\tilde{j}}$

If $S_{sl} \geq S_{k\tilde{j}}$ then $\tilde{S}_{kl} = S_{sl}$

Else

If $\sum_i pl_i \leq D_l - S_{k\tilde{j}}$ where $i \in L_l$ then $\tilde{S}_{kl} = S_{k\tilde{j}}$

Else this swap is infeasible, filter this node

For all other jobs in L_l

$\tilde{S}_{|v|l} = \tilde{S}_{|v-1|l} + \tilde{p}_{|v-1|l}$

For all other jobs at the disrupted machine $\tilde{j}$

$\tilde{S}_{|v|\tilde{j}} = \tilde{S}_{|v-1|\tilde{j}} + \tilde{p}_{|v-1|\tilde{j}}$

We do not change the sequence of other jobs on both machines.

STEP 4.4: Sum the processing time upper bounds of jobs in $L_{\tilde{j}}$

$SUM_{\tilde{j}} = \sum_i pu_i$ where $i \in L_{\tilde{j}}$

STEP 4.5: Find the difference between this sum and the time between maximum completion time of machine $\tilde{j}$ and start time of job s

$EXCESS_{\tilde{j}} = D_{\tilde{j}} - \tilde{S}_{s\tilde{j}} - SUM_{\tilde{j}}$

STEP 4.6: Distribute this excess amount to the jobs in $L_{\tilde{j}}$ according to the Rule 4 $\left(w_{i\tilde{j}}\right)$

STEP 4.7: Subtract the amount $w_{i\tilde{j}}$ distributed to each job i where $i \in L_{\tilde{j}}$ from their processing time upper bounds and calculate new processing times of the jobs

$\tilde{p}_{i\tilde{j}} = pu_i - w_{i\tilde{j}}$ where $i \in L_{\tilde{j}}$

If $\tilde{p}_{i\tilde{j}} < pl_i$ then $\tilde{p}_{i\tilde{j}} = pl_i$ where $i \in L_{\tilde{j}}$

STEP 4.8: Calculate the start time and completion time of the jobs in $L_{\tilde{j}}$

The start time of a job is the completion time of the previous job in the sequence. Then we calculate the start and completion time of the jobs by adding the processing times to their start times.

STEP 4.9: If the completion time of last job at the disrupted machine $\tilde{j}$ exceeds the maximum completion time of this machine, apply maximum completion time feasibility rule

We do the **STEPS 4.4-4.9** for machine l and job k.

STEP 4.10: $L_l = L_l \setminus \{k\}$, $L_{\tilde{j}} = L_{\tilde{j}} \setminus \{s\}$ and update the number of

jobs and earliest start times of all machines and update the positions of all jobs at all machines

Updating the earliest start times:

$EST_j = EST_j \ \forall j = 1 \dots m$ and $j \neq l, \tilde{j}$

$EST_l = \tilde{C}_{kl}$

$EST_{\tilde{j}} = \tilde{C}_{s\tilde{j}}$

Updating the positions of the jobs: The positions of the jobs at the machines other than machine l remain same.

At machine l;

$|v - 1| = |v|$

Lastly, the total cost of this swap operation is calculated by summing the additional compression cost and the reallocation cost occurred by inserting the disrupted job k to machine l and inserting the job s to the disrupted machine $\tilde{j}$. This is the additional cost of the node in which this swap operation is done.

As it is stated before, only the jobs which are started to be processed after the disruption time, are considered to be rescheduled. In the first level of tree, we branch nodes in number of *branchSize* for the first disrupted job. In each node, we try to place the first disrupted job in a different position. In the first node, we move the first disrupted job to the first position of the first machine. In the second node, we move the first disrupted job to the second position of the first machine. In the third node, we swap the disrupted job with the first job of the first machine. We finish the operations with the first machine and we pass to the second machine. In the fourth and fifth nodes, we move the first disrupted job to the first and second positions of the second machine, respectively. In sixth node, we swap the first disrupted job with the first job of the second machine. Thus, we finish the operations of the second machine and we pass to the next machine. To sum up, we apply move operations at two nodes and one swap operation at one node for each machine for a disrupted job. When we finish all operations for each machine for the first level, we calculate the cost and the sum of absolute value of completion time differences of each node and we sort the costs of all nodes at that level.

STEP 5: Calculate the additional cost of each move and swap operations
STEP 6: Keep the best b nodes in this level and fathom the remaining nodes

In the previous steps, for each node, we determine the processing times of the jobs and the job sequence of the machines. We could improve the solution quality of a node by solving a single machine sub-problem. For each node kept in this level, for each machine on which the "move" or "swap" operations are applied on this node, the sequence and TADC value which are obtained in the previous steps, are given as input to the NLP model which solves a single machine sub-problem. Then the NLP model is solved with these inputs to determine the optimal processing times to minimize the total compression cost.

STEP 7: For each node among the best b nodes kept in this level, solve a single machine sub-problem using an NLP model to determine the optimal processing times (For a given sequence on each machine, determine the optimal processing times for each job for a given TADC value and the maximum completion time constraints.)

This model is bi-criteri single machine model. The objective functions are the compression costs of the jobs on the machine on which the model is solved and TADC value of all jobs at all machines.

In a node, if "move" operation is applied, this model is solved for only machine l. If this is a "swap" operation, this NLP model is first solved for the disrupted machine $\tilde{j}$ and then it is solved for the machine l. In this model, we know the sequence of the machine on which the model is solved and overall TADC upper bound value B is given. Set N_j denotes the set of jobs included in the sequence of machine j. If this is a "move" operation, we include disrupted job k in L_l and if this is a "swap" operation, we include disrupted job k in L_l and swapped job s in $L_{\tilde{j}}$. We also use the rt_j, D_j, pu_i, pl_i, $disr$, b_{ij} and k_{ij} values of the original schedule as the parameters.

$$\min \sum_{i \in N_j} b_{ij} \ (w_{ij}^{k_{ij}})$$

s.t.

$$\sum_{i \in N_j} \left| \tilde{C}_i - C_i \right| \leq B - \sum_{i \in \mathcal{N}, i \notin N_j} \left| \tilde{C}_i - C_i \right|, \tag{4.1}$$

$$\tilde{S}_{ij} \geq rt_j, \qquad \forall \ i \in L_j, \tag{4.2}$$

$$\tilde{S}_{[k+1]j} \geq \tilde{S}_{[k]j} + \tilde{p}_{[k]j}, \qquad \forall \ [k], [k+1] \in N_j, \tag{4.3}$$

$$\tilde{S}_{ij} + \tilde{p}_{ij} = \tilde{C}_i, \qquad \forall \ i \in N_j, \tag{4.4}$$

$$\tilde{p}_{ij} = pu_i - w_{ij}, \qquad \forall \ i \in N_j, \tag{4.5}$$

$$\tilde{S}_{ij} \geq S_i, \qquad \forall \ i \in N_j, \tag{4.6}$$

$$\tilde{p}_{ij} \geq pl_i, \qquad \forall \ i \in N_j, \tag{4.7}$$

$$\tilde{p}_{ij} \leq pu_i, \qquad \forall \ i \in N_j, \tag{4.8}$$

$$\tilde{S}_{ij} + \tilde{p}_{ij} \leq D_j, \qquad \forall \ i \in N_j, \tag{4.9}$$

$$\tilde{S}_{ij}, \tilde{p}_{ij}, w_{ij}, \tilde{C}_i \geq 0, \qquad \forall \ i \in N_j, \tag{4.10}$$

This model aims to minimize the compression cost on machine j. Compression cost is calculated by summing the cost function $f_{ij}(w_{ij})$ over all jobs at the machine j. The cost function $f_{ij}(w_{ij})$ is equal to b_{ij} $(w_{ij}^{k_{ij}})$ where b_{ij} and k_{ij} are compression cost coefficients.

In Constraint (4.1), TADC value which is found in the previous steps is given to this model as an upper bound for TADC value.

Constraint (4.2) requires that start times of the job in L_j is greater than or equal to disruption time, new start times of these jobs should be greater than or equal to the ready time of the machine j on which the model is solved.

Constraints (4.3) assures that start time of the job at position $[k+1]$ of machine j is greater than or equal to the sum of the start time and processing time of the job at position $[k]$. Constraint (4.4) calculates the completion time of a job at the machine j by summing the start time and the processing time of that job. Constraint (4.5) implies that the compression amount of a job at the

machine j is equal to the difference between the upper bound of the processing time of that job and the processing time of that job at the machine j. Constraint (4.6) guarantees that new start time of a job at the machine j is greater than or equal to the start time of that job in the original schedule.

Constraint (4.7) and (4.8) ensure that the processing time of a job on machine j should be between the lower bound and the upper bound of the processing time of that job. Constraint (4.9) provides that completion time of any job at the machine j cannot be greater than the maximum completion time of the machine. Constraints (4.10) are the nonnegativity constraints.

Since the objective function in this model is non-linear, we reformulate this model putting into conic optimization problem with linear objective and conic constraints as we did to handle the non-linearity in the original mathematical model given in Chapter 3. Each term b_{ij} $(w_{ij}^{k_{ij}})$ in the objective function is replaced with an auxiliary variable $t_{ij} \geq 0$ and the constraint b_{ij} $(w_{ij}^{k_{ij}}) \leq t_{ij}$ is added into the constraints as in Akturk, Atamturk and Gurel [3].

This NLP model differs from the original mathematical model given in Chapter 3 with the exclusion of the sequence constraints (3.3)-(3.7). One of the aspects of the hardness of original mathematical model is these constraints. Since we give the sequence of the machines as an input to the model, NLP model in this step of the algorithm is easily solved relative to the original mathematical model.

Each node corresponds to a solution and as we keep a node, we keep a solution and we proceed with that solution through the next levels.

STEP 8: $DJ = DJ \setminus \{k\}$
STEP 9: Go to next level by selecting the next job $k = argmin\{S_i\}$ where $i \in DJ$, if $DJ = \emptyset$ then go to STEP 11
STEP 10: For each solutions kept in the previous level do the STEPS 3-7, go to STEP 8.

When we pass to the second level, we branch nodes in number of $branchSize\ b$ for each node that we kept in the previous level. In this level, in each node, we

try to place the second disrupted job in a different position by keeping the start, completion and the processing time of the first disrupted job at the parent node. For the first node that we kept in the first level, we branch nodes in number of branchSize at the second level. In the first child node, we move the second disrupted job to the first position of the first machine. In the second child node, we move the second disrupted job to the second position of the first machine. In the third child node, we swap the second disrupted job with the first job of the first machine. We apply move operations at two child nodes and swap operation at one child node for each machine for this parent node. When we branch the child nodes of the second parent node that we kept in the previous level, we again apply move operations at two child nodes and swap operation at one child node for each machine and each operation corresponds to one child node. We branch nodes in number of branchSize for each parent node that we kept in the previous level. Then we calculate the costs and the sum of the absolute value of the completion time differences for each child node at the second level and we again eliminate the nodes correspond to the solutions with the cost more than the b^{th} smallest cost among all solutions at this level and we keep the nodes correspond to the solutions with the cost less than or equal to the b^{th} smallest cost. We proceed with these nodes, which we kept in this level, through the next levels.

Then we pass to the next level to place the next disrupted job. In each level, we keep the position and the processing time of the disrupted jobs that we placed in the previous levels.

We branch child nodes for the parent nodes until we place all of the disrupted jobs. When there are no remaining disrupted jobs to be placed, we stop and examine the nodes that we kept at the last level. In the last level, we place the last disrupted job by keeping the places and the processing times of the previous disrupted jobs. So, in the last level, as we place the last disrupted job, in each node we kept, we have a solution of a complete schedule. After we calculate the costs and the sum of absolute value of the completion time differences of the nodes we kept in the last level, we look for the non-dominated solutions.

STEP 11: Calculate the sum of absolute value of completion time differ-

ences and total cost of the solutions kept at the last level

Sum of absolute value of completion time differences: Throughout the DTA, we regard to place the jobs to keep the difference between the new completion time of a job and the original completion time of that job minimum as much as possible. This is the second objective of our problem, which is calculated by summing the absolute value of the completion time differences over all jobs.

We calculate the total cost by summing the compression cost of each job resulted from subtracting the slack from the processing times and the total reallocation cost.

STEP 12: If there are any feasible solutions at the last level, find the non-dominated solutions among solutions kept the at the last level and STOP, else go to STEP 13
STEP 13: Increase the maximum completion time of each machine by Δ units and go to STEP 1

If there is a solution that there is not any solution with lower total cost and lower TADC at the same time comparing to this solution, this is a non-dominated solution. Each node with a solution which is among the non-dominated solutions corresponds to a final solution of our problem.

4.3 Summary

In this chapter, we first gave theoretical properties and then we proposed a search algorithm using these properties. First we determined the sequence and the processing times in the early steps of the algorithm and in STEP 7, for a given sequence on a machine, we determined the optimal processing times for a given TADC upper bound. In Chapter 5, we will give numerical examples to clarify the proposed DTA heuristic. First we will give an example without including the STEP 7, and then we will give an example in which the STEP 7 is included in the algorithm.

Chapter 5

Numerical Example

In Figure 5.1, we give an original schedule example. There are 2 machines and disruption occurs on machine 1. Disruption starts at time 3.58 and ends at time 8.68. So, the disrupted jobs include the jobs 2, 7 and 10, i.e., $DJ = \{2, 7, 10\}$. Original start times, processing times, completion times, processing time upper and lower bounds and the machine coefficients for each job can be seen in Tables 5.1 and 5.2.

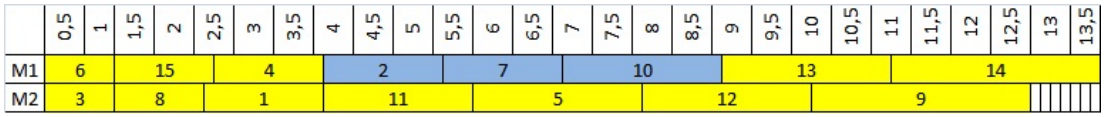

Figure 5.1: Original schedule

In the search tree, each node has 4 $((3 \times (2-1)) + 1)$ immediate child nodes. For this example, these immediate child nodes of a parent node corresponds to the solutions that;

- 1^{st} **child node**; moving the disrupted job of that level to the first position of machine 1
- 2^{nd} **child node**; moving the disrupted job of that level to the first position of machine 2

- 3^{rd} **child node**; moving the disrupted job of that level to the second position of machine 2
- 4^{th} **child node**; swapping the disrupted job of that level with the swapped job of that level on machine 2.

M/C	Job	Original Start Time	Original Comp Time	Original Process Time
1	6	0	1	1
1	15	1	2.28	1.28
1	4	2.28	3.58	1.3
1	2	3.58	5.1	1.52
1	7	5.1	6.65	1.55
1	10	6.65	8.68	2.03
1	13	8.68	10.74	2.06
1	14	10.74	13.44	2.7
2	3	0	1	1
2	8	1	2.09	1.09
2	1	2.09	3.58	1.49
2	11	3.58	5.54	1.96
2	5	5.54	7.62	2.08
2	12	7.62	9.79	2.17
2	9	9.79	12.45	2.66

Table 5.1: Original Start, Processing and Completion Times

There will be 3 levels that the algorithm will be executed. At each level, we will try to move the disrupted job to two positions on the second machine and be swapped with the first job of L_2. Additionally, the disrupted job will also be moved to the first position on the disrupted machine. We will try only one position on the disrupted machine, because it will be the first position after the end time of the disruption and even this position is late for the disrupted job in terms of TADC. In this example, we assume that the reallocation cost of moving a job between the machines is 4. Note that, first we will solve the problem without including STEP 7 and then solve it adding the STEP 7.

The disrupted job will be job 2 at the first level, job 7 at the second level and job 10 at the third level. And since there will be no jobs after job 10 in the disrupted job list, the algorithm will terminate.

Job i	Coeff b_{i1}	Coeff b_{i2}	Coeff k_{i1}	Coeff k_{i2}	Upper Bound	Lower Bound
1	14.95	13.10	1.6	3	1	0,44
2	14.75	12.86	1.5	1.3	1,28	0,57
3	12.45	19.60	2.2	3	1,3	0,43
4	18.20	15.89	2.6	1.5	1,52	0,23
5	13.27	16.14	1.6	1.2	1,55	0,17
6	19.05	18.57	1.4	1.7	2,03	0,72
7	18.36	14.48	1.6	2.1	2,06	0,73
8	12.96	13.03	2.1	1.3	2,7	0,46
9	15.72	18.29	1.7	1.6	1	0,45
10	11.37	17.92	1.8	2.1	1,09	0,28
11	16	10.01	2.2	3	1,49	0,4
12	12.29	19.63	2.9	1.9	1,96	0,93
13	18.94	19.69	1.3	2.7	2,08	1,02
14	19	12.26	2.2	1.9	2,17	0,39
15	16.37	11.85	2.5	2.1	2,66	1,29

Table 5.2: Cost Coefficients and Processing Time Upper and Lower Bounds

For the first level, the disrupted job $k = 2$.. The swapped job will be job 11 at machine 2, because of Rule 2, we select the first job in the sequence of machine 2 as it can be seen in Figure 5.2. Earliest start time is time 3.58 for machine 2 and 8.68 for machine 1, since it is the end time of the disruption (See Figure 5.2).

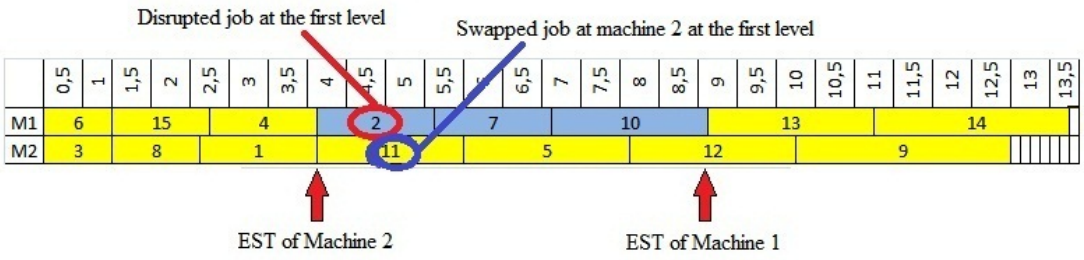

Figure 5.2: Disrupted and swapped jobs and EST of machines at the first level

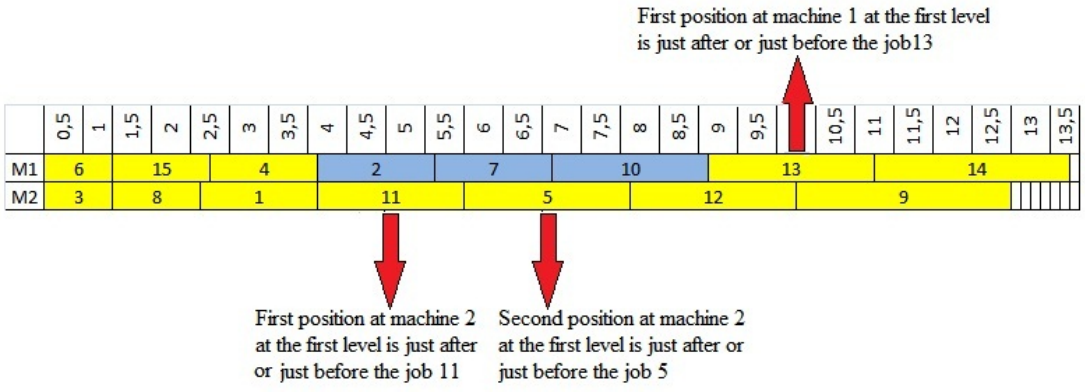

Figure 5.3: Positions at the first level

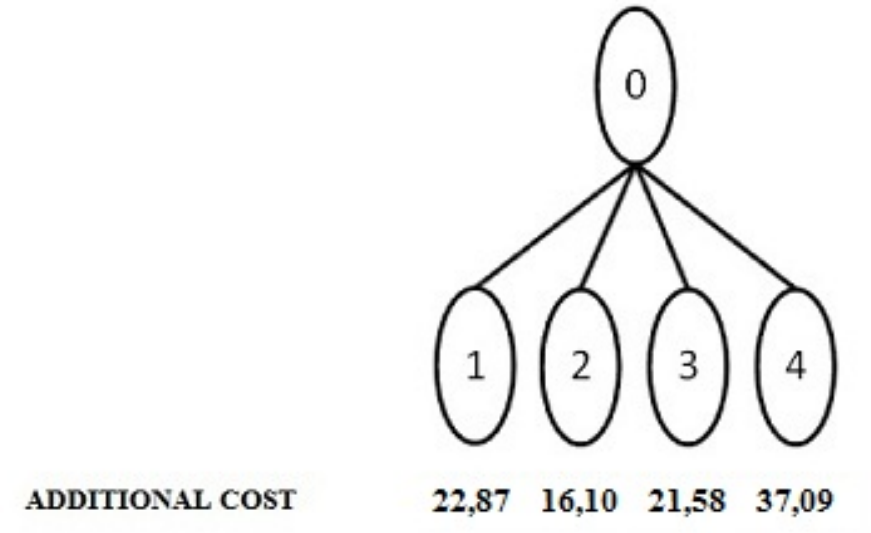

Figure 5.4: Additional costs of all solutions at the first level

5.1 Solution without STEP 7

In Figure 5.4, at node 1, we try to move job 2 to the first position at disrupted machine. For machine 1, first position is just after or just before the job 13 for the first level.

The job list of machine 1 includes $L_1 = \{13, 14\}$, since the start times of these jobs at the disrupted machine is greater than or equal to the disruption end time. In order to check the feasibility of this move operation, we set the processing times of the jobs in the job list and the disrupted job to their lower bounds and sum these processing times. This sum is equal to 1.42 and this can fit the time interval between earliest start time and maximum completion time of the disrupted machine which is equal to $13.44 - 8.68 = 4.76$. Therefore, we check the feasibility and conclude that this move operation is feasible.

Since we try to place the job 2 into the first position of the disrupted machine and the start time of job 13 is 8.68 which is greater than the original start time of job 2 which is 3.58, we compare the slopes of job 2 and job 13. Slope of job 2 is $k_{2,1} \times b_{2,1} \times (pu_2 - (p_{2,1} - 1))^{(k_{2,1}-1)} = 16.71$ and slope of job 13 is $k_{13,1} \times b_{13,1} \times (pu_{13} - (p_{13,1} - 1))^{(k_{13,1}-1)}) = 24.62$. Since the change in the compression cost will be less when we compress job 2 additional one more unit comparing to the compressing job 13 additional one more unit, we decide to place the job 2 just before the job 13. Start time of job 2 becomes 8.68 and job 13 stays in the L_1 and it becomes $L_1 = L_1 \cup \{2\}$.

Then we apply the compression rule. There are 2 jobs after job 2, one job after job 13, there is no job after job 14. Sum of these numbers is $2+1+0 = 3$ and compression proportion of job 2 is 2/3, compression proportion of job 13 is 1/3 and compression proportion of job 14 is 0. When we set the processing times of the jobs in $L_1 = \{2, 13, 14\}$ to their upper bounds and sum these upper bounds we obtain 5.65. Difference between the time between maximum completion time and the start time of job 2 and this sum is equal to $5.65 - (13.44 - 8.68) = 1.52$. So, we try to distribute 1.52 to the jobs 2, 13 and 14 according to their compression proportion. We compress $(1.52 \times 2/3) = 1.01$ units from job 2, $(1.52 \times 1/3) = 0.51$

units from job 13 and 0 unit from job 14. So, their new processing times becomes $(1.52-1.01) = 0.51$, $(2.06-0.51) = 1.55$ and 2.7, respectively. All new processing times are greater than or equal to the lower bounds.

New schedule at the end of the first level is shown in Figure 5.5. Number of jobs in the disrupted machine and the positions stays same and the earliest start time becomes the completion time of job 2 which is 9.19. L_1 is also updated as $L_1 = L_1 \setminus \{2\}$.

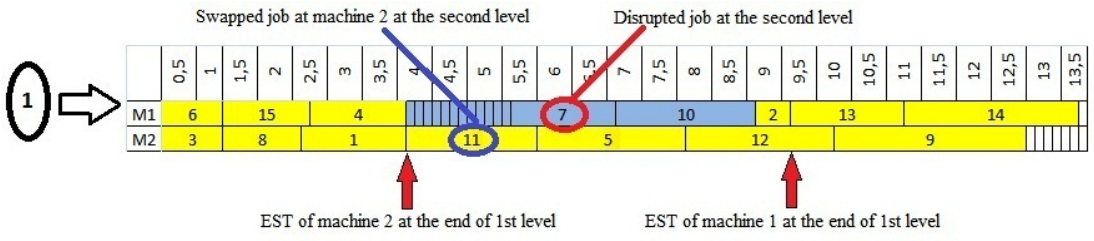

Figure 5.5: Schedule at the end of the first level at node 1

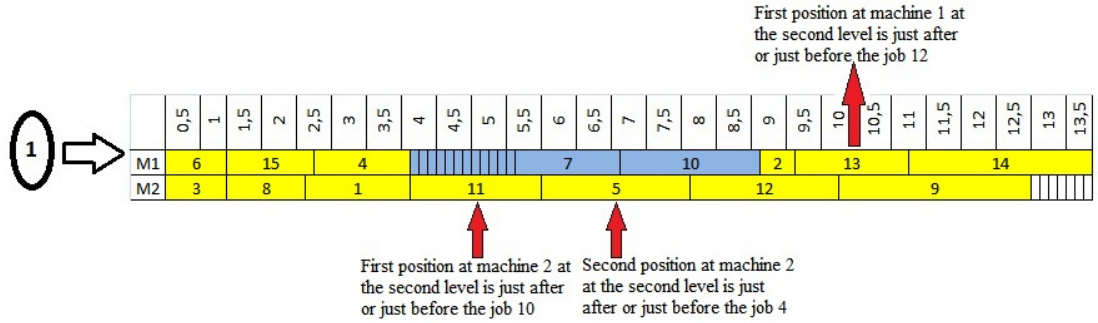

Figure 5.6: Positions at the end of the first level at node 1

At node 2, we try to move job 2 to the first position at machine 2. For machine 2, first position is just after or just before the job 11 for the first level (Figure 5.3). New schedule at the end of the first level is shown in Figure 5.7.

At node 3, we try to move job 2 to the second position at machine 2. For machine 2, second position is just after or just before the job 5 for the first level. At node 4, we try the job to swap with the first job after EST of machine 2, the swapped job will be job 11 at machine 2 at the first level (Figure 5.3). Then we calculate the additional cost of each move and swap operations as it can be seen in Figure 5.4 and update the disrupted job list ($DJ = \{7, 10\}$).

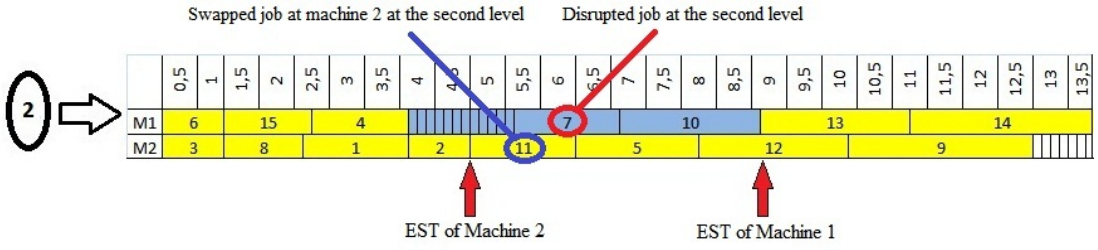

Figure 5.7: Schedule at the end of the first level-at node 2

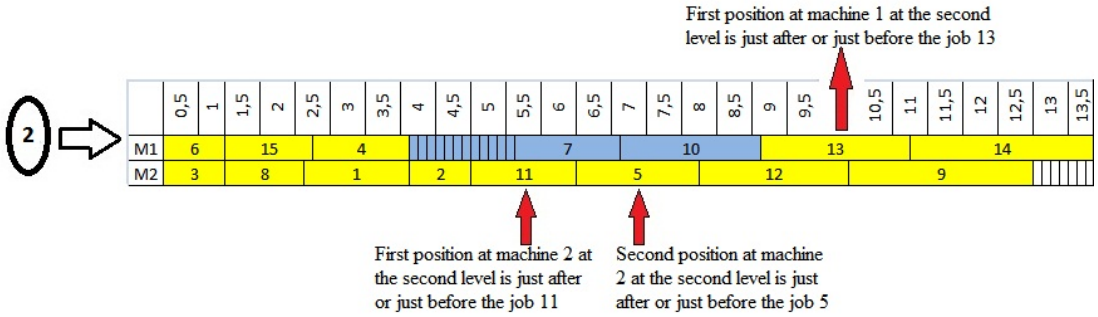

Figure 5.8: Positions at the end of the first level at node 2

In this problem, we choose the $beamSizeb = 7$. Therefore, we keep the best 7 nodes in this level and fathom all other nodes and go to next level by selecting the next disrupted job from the disrupted job list. Since there are 4 nodes at the first level, we keep all nodes in this level to go to the next level. The disrupted job is $k = 7$ at the second level.

For each nodes kept in the previous level, we apply the same operations as we do in the previous level (Figure 5.9).

In the second level, at child node 6 of parent node 1, job 7 is moved to the first position of machine 2 (See Figure 5.6).

The job list of machine 2 includes $L_2 = \{11, 5, 12, 9\}$. Sum of the processing time lower bounds of the jobs in L_2 and the disrupted job 7 is equal to 3.8 and it is less than time interval between earliest start time and maximum completion time of the disrupted machine which is equal to $12.45 - 3.58 = 8.$. This move operation is also feasible.

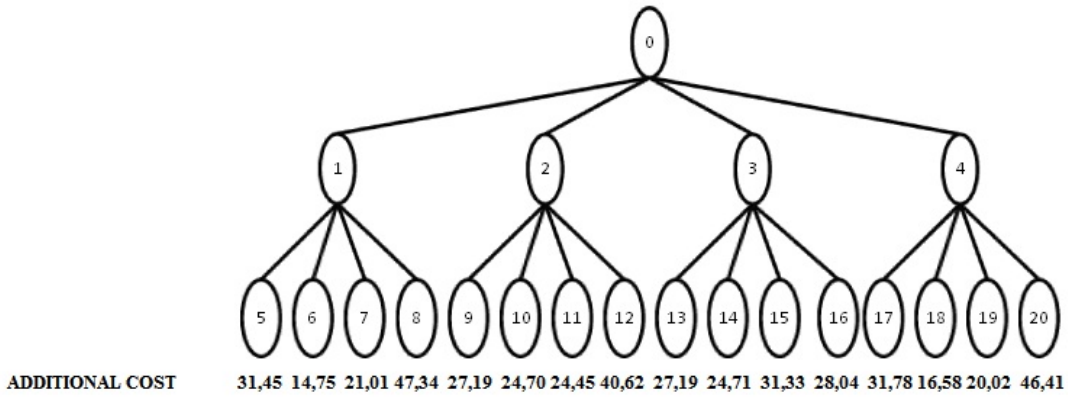

Figure 5.9: Additional costs of all solutions at the second level

The first job in L_2 is job 11 and its start time is 3.58. This is less than 5.1 which is the original start time of job 7. The difference ($5.1 - 3.58 = 1.52$), 1.52 is within the upper and lower bounds of processing time of job 11. So, we set the start time of job 11 to 3.58 and completion time of job 11 to 5.1 and insert job 7 just after the job 11. Start time of job 7 becomes the completion time of job 11 and $L_2 = L_2 \setminus \{11\} \cup \{7\}$.

There are 3 jobs succeeding job 7, 2 jobs succeeding job 5, one job succeeding job 12 and there is no job succeeding job 9. Sum of these numbers is 3+2+1+0 = 6 and compression proportion of job 7 is 3/6 and compression proportion of job 5 is 2/6, compression proportion of job 12 is 1/6 and compression proportion of job 9 is 0. Sum of the processing times of jobs in L_2 is equal to 8.9. Difference between the time between maximum completion time and the start time of job 2 and this sum is $8.46 - (12.45 - 5.1) = 1.11$. So, we compress $(1.11 \times 3/6) = 0.56$ units of job 7, $(1.11 \times 2/6) = 0.37$ units of job 5, $(1.11 \times 1/6) = 0.19$ units of job 12 and 0 units of job 9. So, new processing times become $(1.55 - 0.47) = 1.08$, $(2.08-0.37) = 1.71$, $(2.17-0.19) = 1.98$ and 2.66, respectively. All new processing times are greater than or equal to the lower bounds.

New schedule at the end of the second level is shown in Figure 5.10. Number of jobs in the disrupted machine is decreased by 1 and the positions of each job on machine 2 are decreased by 1 and the earliest start time becomes the completion time of job 7 which is 6.1. L_2 becomes $L_1 = L_1 \setminus \{7\}$.

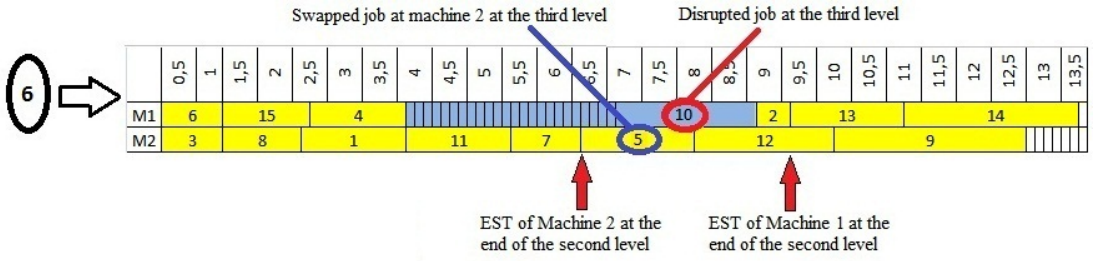

Figure 5.10: Schedule at the end of the second level at node 6

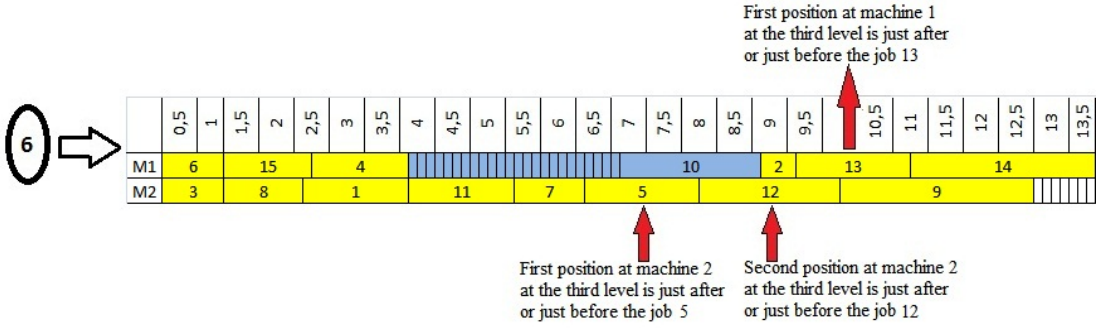

Figure 5.11: Positions at the end of the second level at node 6

In the second level, at child node 10 of parent node 2, we move job 2 to the first position at machine 2. For machine 2, first position is just after or just before the job 11 for the first level.(Figure 5.8) New schedule at the end of the second level is shown in Figure 5.12.

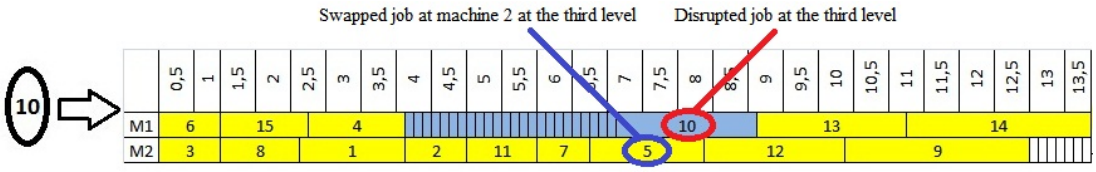

Figure 5.12: Schedule at the end of the first level-at node 10

After we complete all the required operations in this level and calculate the additional costs correspond to each operation as it can be seen in Figure 5.9, we keep the best 7 nodes in this level and fathom all other nodes. The nodes we selected in this level are the nodes 6, 7, 10, 11, 14, 18 and 19. We update the disrupted job list ($DJ = \{10\}$). We go to next level by selecting the next disrupted job. The disrupted job is $k = 10$ at the third level.

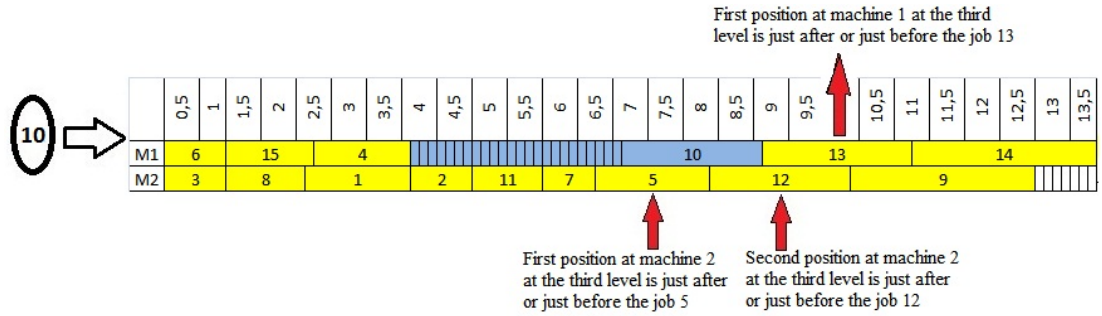

Figure 5.13: Positions at the end of the first level at node 10

For each nodes kept in the previous level, we apply the same operations as we do in the previous level (Figure 5.14).

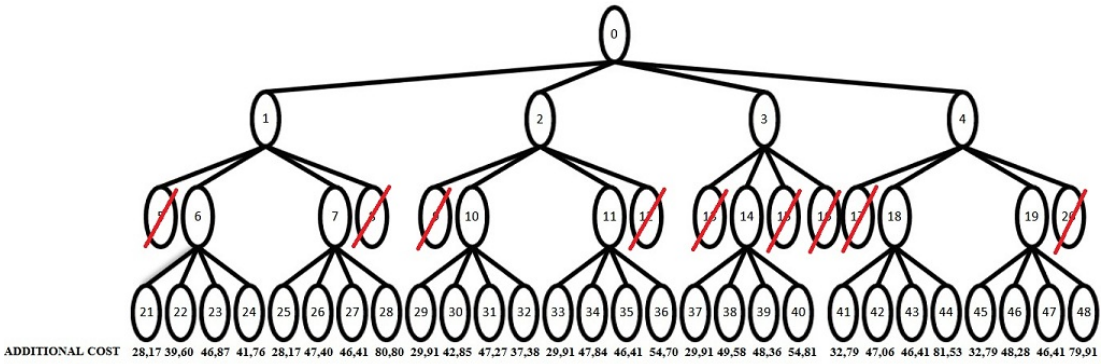

Figure 5.14: Additional costs of all solutions at the third level

In the third level, at child node 21 of parent node 6, job 10 is moved to the first position of the disrupted machine (See Figure 5.11).

The processing time lower bounds of jobs in $L_1 = \{13, 14\}$ and job 10 is equal to 1.91 and this is smaller than the time interval between earliest start time and maximum completion time of the disrupted machine which is equal to $13.44 - 9.19 = 4.25$. Hence, the feasibility of this move operation is checked.

First job in L_2 is job 13 and the start time of job 13 is 9.19. It is greater than 6.65 which is the original start time of job 10. Slope of job 10 is $k_{10,1} \times b_{10,1} \times (pu_{10} - (p_{10,1} - 1))^{(k_{10,1}-1)} = 20.47$ and slope of job 13 is $k_{13,1} \times b_{13,1} \times (pu_{13} - (p_{13,1} - 1))^{(k_{13,1}-1)} = 27.86$. Hence, we decide to insert job 10 just before the job 13. Start time of job 10 becomes 9.19 and L_1 becomes $L_1 = L_1 \cup \{10\}$.

There are 2 jobs succeeding job 10, one job succeeding job 13; there is no job succeeding job 14. Sum of these numbers is $2 + 1 + 0 = 3$ and sum of the processing time upper bounds in L_1 is 6.79. Difference between the time between maximum completion time and the start time of job 10 and this sum is equal to $6.79 - (13.44 - 9.19) = 2.54$. So, the required compression amounts are $(2.54 \times 2/3) = 1.70$, $(2.54 \times 1/3) = 0.85$ and 0, for jobs 10, 13 and 14 respectively. So, new processing times are $(2.03 - 1.70) = 0.33$, $(2.06 - 0.85) = 1.21$ and 2.7, respectively. But new processing time of job 10 is less than its lower bound. So we set the processing time of job 10 to its lower bound 0.72. In this case, the completion time of job 14 becomes 13.83. This violates the maximum completion time constraint, the completion time of the last job at a machine should be less than or equal to the maximum completion time and the maximum completion time of the machine 1 is 13.44. So, we try to decrease the amount exceeds the maximum completion time from the processing times of the jobs starting from the last job of the machine. We decrease the processing time of job 14 by $13.83 - 13.44 = 0.39$ and it becomes $2.7 - 0.39 = 2.31$. This is within the upper and lower bounds of job 14. Now, all new processing times are within their upper and lower bounds.

New schedule at the end of the third level is shown in Figure 5.15. Number of jobs in the disrupted machine and the positions stays same and the earliest start time becomes the completion time of job 10 which is 9.91. L_1 becomes $L_1 = L1 \setminus \{10\}$.

Figure 5.15: Schedule at the end of the second level at node 6

In the third level, at child node 29 of parent node 10, job 10 is moved to the first position of machine 1. For machine 1, first position is just after or just before the job 13 for the first level (Figure 5.13). New schedule at the end of the third level at node 29 is shown in Figure 5.16.

After we complete the required operations at nodes from 21 to 48, we calculate

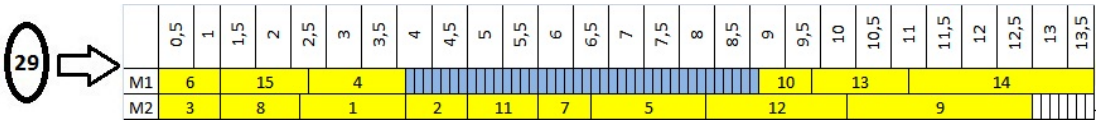

Figure 5.16: Schedule at the end of the third level at node 29

the additional costs correspond to each operation as it can be seen in Figure 5.14, we keep the best 7 nodes in this level and fathom all other nodes. At the end of the third level, since disrupted job list becomes empty ($DJ = \emptyset$), the algorithm terminates.

Nodes selected in the third level are nodes 21, 22, 29, 32, 37, 41 and 45 (See Figure 5.14). We calculate the sum of absolute value of completion time differences of the solutions correspond to the nodes kept at the last level as it can be seen in Figure 5.17.

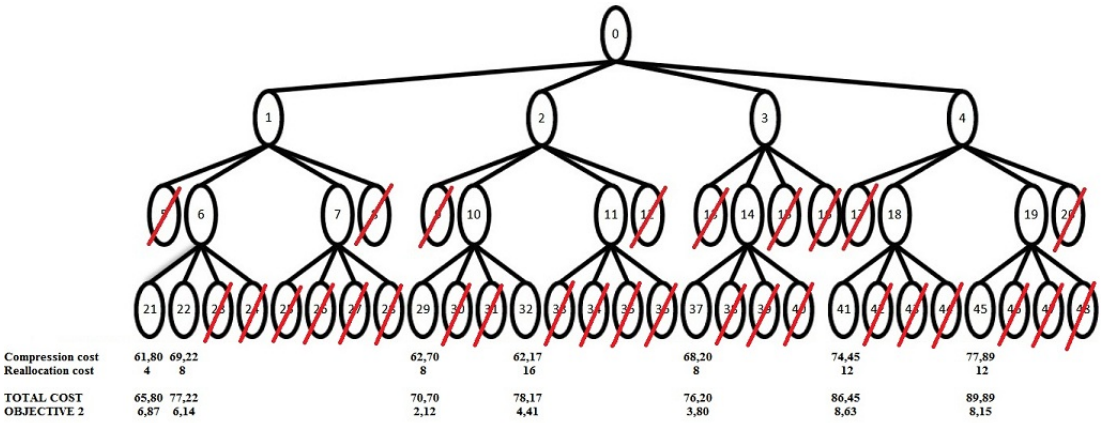

Figure 5.17: Total cost and sum of completion time differences of all solutions kept in the third level

Finally, we find the non-dominated solutions among the solutions kept at the last level. As we see in the Figure 5.17, non-dominated solutions of this problem corresponds to the nodes 21 and 29 in the tree. First solution corresponds to node 21 and it means that, at the first level, at node 1, job 2 is moved to the first position of the disrupted machine, at the second level, at node 6, job 7 is moved to the first position of machine 2, at the third level, at node 21, job 10 is moved to the first position of the disrupted machine. Second solution corresponds to

node 29 and it means that, at the first level, at node 2, job 2 is moved to the first position of machine 2, at the second level, at node 10, job 7 is moved to the first position of machine 2, at the third level, at node 29, job 10 is moved to the first position of machine 2.

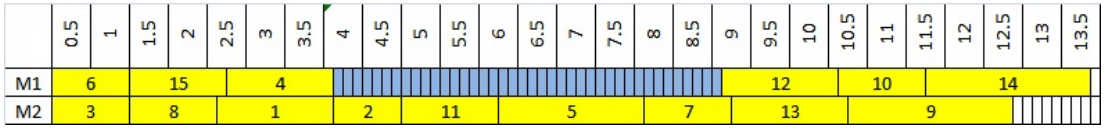

Figure 5.18: Optimal schedule corresponding to solution 1 of the algorithm

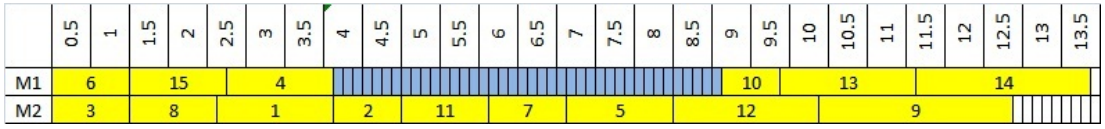

Figure 5.19: Optimal schedule corresponding to solution 2 of the algorithm

The objective 1 (total cost) of the solution 1 which corresponds to node 21 is 65.79 and the objective 2 (TADC) of the solution 1 is 6.87 (See Figure 5.17). When we give the objective 2 of this solution to the MIP model as an upper bound, we obtain the optimal schedule which is shown in Figure 5.18. The objective 1 of this optimal schedule of solution 1 is 58.67. So, the deviation between the objective 1 values of the algorithm and the MIP model for given same upper bound on TADC is 12.14% which is calculated as $(65.79 - 58.67)/58.67$.

The objective 1 of the solution 2 (node 29) is 70.70 and the objective 2 of the solution 2 is 2.12 (See Figure 5.17). When we give the objective 2 of this solution to the MIP model as a bound, we obtain the optimal schedule which is shown in Figure 5.19 and the objective 1 of this optimal schedule of solution 1 as 63.74. Hence, we obtain the deviation between the objective 1 values of the algorithm and model as 10.91% calculating $(70.70 - 63.74)/63.74$.

For these two non-dominated solutions, average deviation is found as 11.53.

5.2 Solution with STEP 7

Before we apply STEP 7, we follow the previous steps to find the additional cost of the nodes at each level. After we calculate the additional costs of each node generated at a level and we filter some of the nodes, we solve the single machine sub-problem which is used in STEP 7 to calculate the total cost of corresponding selected nodes.

We start with the same original schedule which is shown in Figure 5.1. Then, we applied the previous steps to find the additional costs and so we obtain the additional costs as in Figure 5.4 for the first level. We select all the nodes to generate child nodes, since there are less than 7 nodes in this level.

Before we go to the next level, we solve the NLP model given in STEP 7 for the machines on which processing times of the jobs are changed or there is a job arrival or removal.

For example, in node 1, we try to move the first disrupted job 2 at the disrupted machine after the end time of the disruption as we stated earlier. We obtained the schedule in Figure 5.5 in node 1 by following the steps in DTA. Then we give the sequence of this schedule as an input and TADC value of this schedule as an upper bound to the NLP model in STEP 7 and we solve this NLP model to find the optimal processing times that minimize the total compression cost with these given parameters. The schedule which is found by the NLP model in STEP 7 is shown in Figure 5.20.

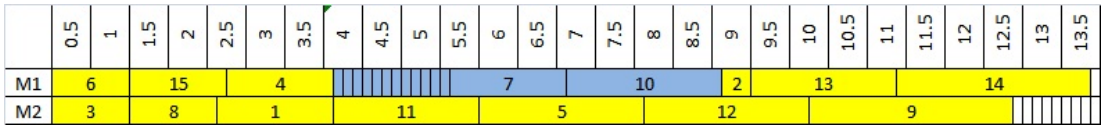

Figure 5.20: Schedule in node 1 found by STEP 7

Besides, in node 2, we move the first disrupted job 2 to the first position of machine 2 and at the end of the STEP 6, the schedule in Figure 5.7 is obtained. Then we solve the NLP model for the 2^{nd} machine and the optimal schedule is

attained as in Figure 5.21.

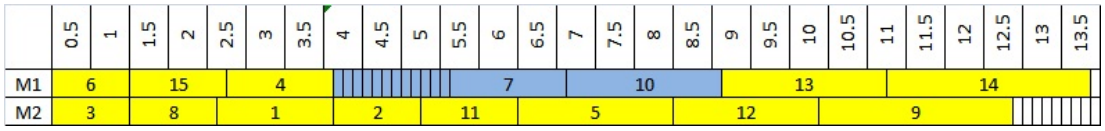

Figure 5.21: Schedule in node 2 found by STEP 7

At the end of the first level, cost values, obtained by solving NLP model in STEP 7 for the selected nodes, is shown in Figure 5.22.

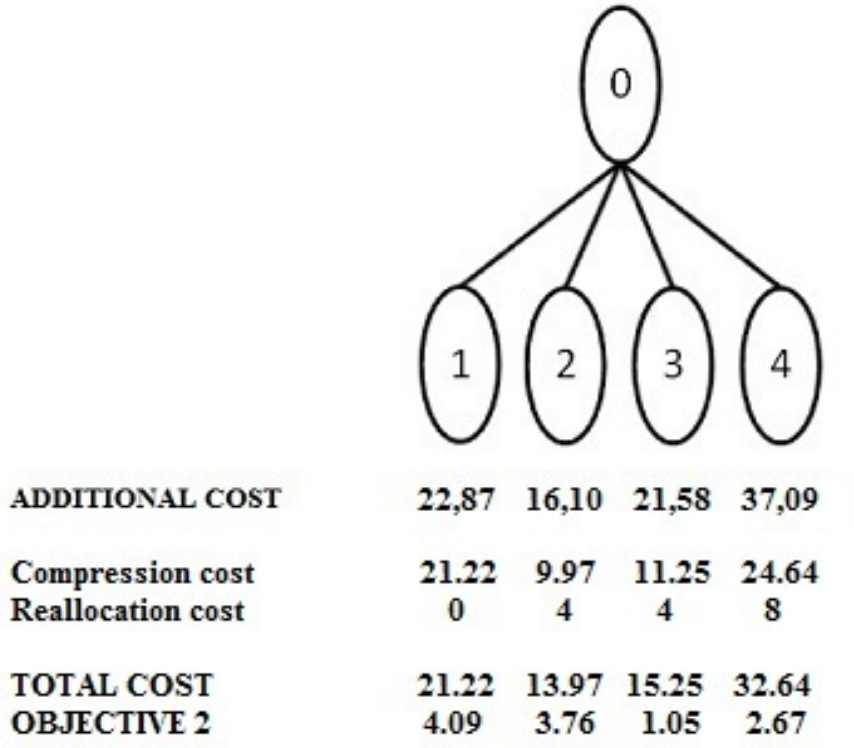

Figure 5.22: Objective values at the end of the first level

Then we go to the next level and generate the child nodes of the nodes selected in the previous level. We calculate the additional costs following the previous steps of DTA; the additional costs of the second level can be seen in Figure 5.23.

We select the best 7 nodes among these solutions and solve NLP model in

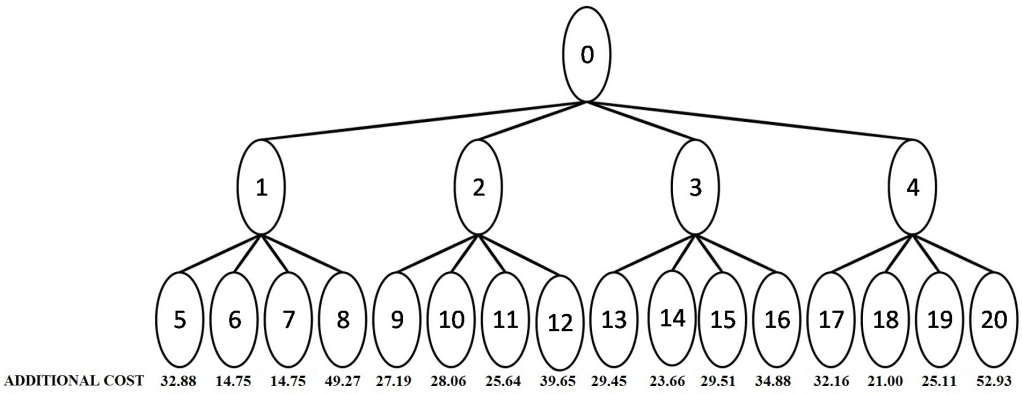

Figure 5.23: Additional costs of all solutions at the second level

STEP 7 for these nodes. The selected nodes of these level are nodes 6, 9, 10, 11, 14, 18 and 19. In child node 6 of parent node 1, we try to move the second disrupted job 7 to the first position of machine 2 and first we get the schedule in Figure 5.24 by applying the previous steps of DTA.

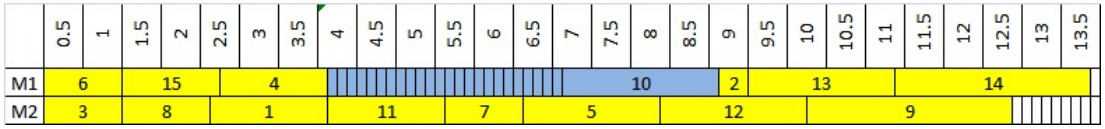

Figure 5.24: Schedule in node 6 at the end of the STEP 6

Then we solve the NLP model in STEP 7 for machine 2 giving the sequence and TADC value of the schedule in Figure 5.24 as inputs to the model and we obtain the optimal schedule for given parameters as in Figure 5.25.

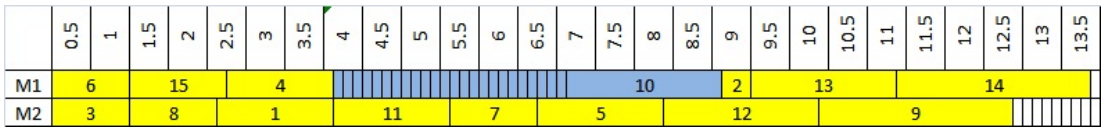

Figure 5.25: Schedule in node 6 found by STEP 7

As well, in child node 10 of parent node 2, we try to move the second disrupted job 7 to the first position of machine 2. The schedules obtained at the end of the STEP 6 and by the STEP 7 are shown in Figure 5.26 and Figure 5.27, respectively.

Figure 5.26: Schedule in node 10 at the end of the STEP 6

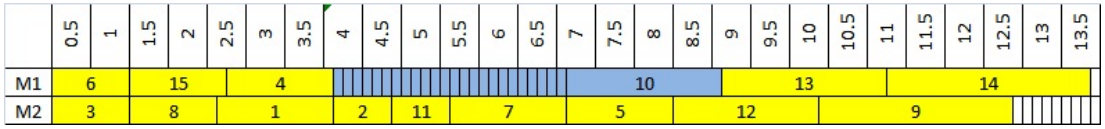

Figure 5.27: Schedule in node 10 found by STEP 7

After we solve NLP model in STEP 7 for the selected nodes of the second level, we calculate the total costs of these nodes as it can be seen in Figure 5.28.

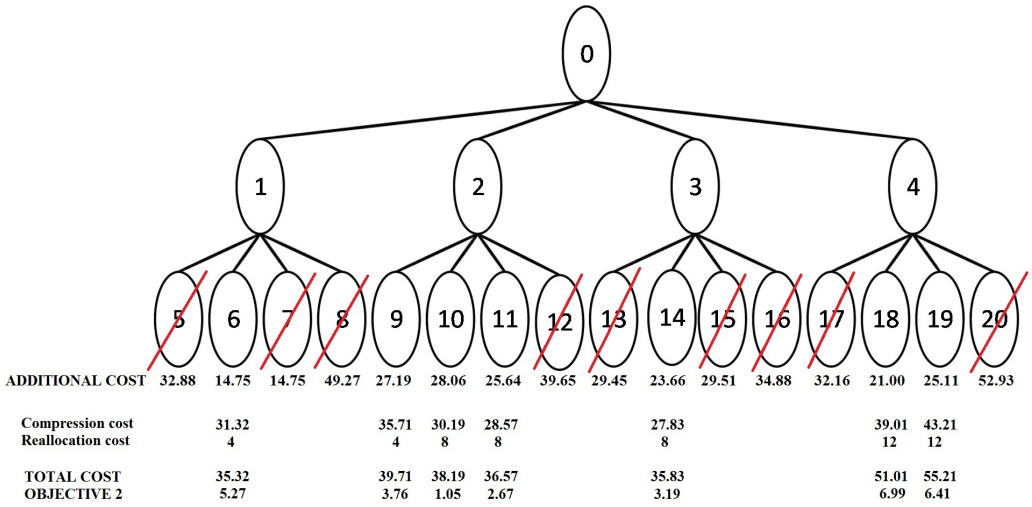

Figure 5.28: Objective values at the end of the second level

We go to the next level by generating the child nodes of the selected parent nodes of the second level and calculate the additional costs of these child nodes applying the procedures until STEP 7 as it is shown in Figure 5.29. It is the last level in the decision tree, since job 10 is the last disrupted job. We select the best 7 nodes among these child nodes in terms of the additional cost and filter the remaining nodes. In this level, the selected nodes are nodes 21, 22, 24, 25, 27, 29 and 41. We solve the NLP model in STEP 7 for these selected nodes.

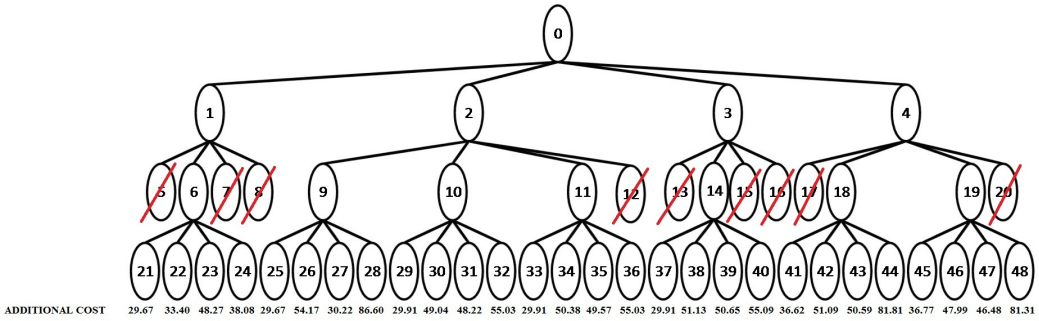

Figure 5.29: Additional costs of all solutions at the third level

In child node 21 of parent node 6, we move the last disrupted job 10 to the first position of the disrupted machine, that is, we move this job after the end time of the disruption. We first get the schedule in Figure 5.30 at the end of the STEP 6.

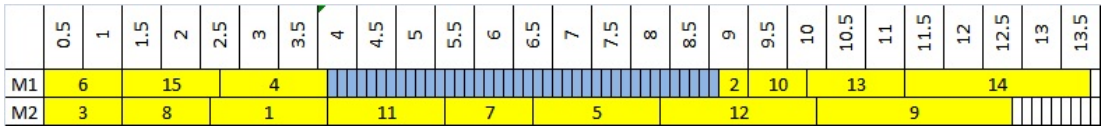

Figure 5.30: Schedule in node 21 at the end of the STEP 6

After we apply STEP 7 giving this schedule as an input to the NLP model in STEP 7 and solve this sub-problem for the disrupted machine, we get the optimal processing times for the disrupted machine and new schedule becomes as in Figure 5.31.

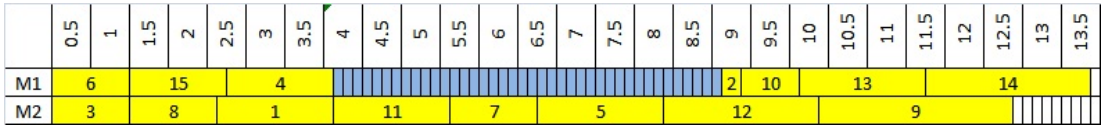

Figure 5.31: Schedule in node 21 found by STEP 7

Besides, in child node 29 of parent node 10, we also move the disrupted job 10 after the disruption end time on the disrupted machine and the schedule obtained at the end of the STEP 6 is shown in Figure 5.32.

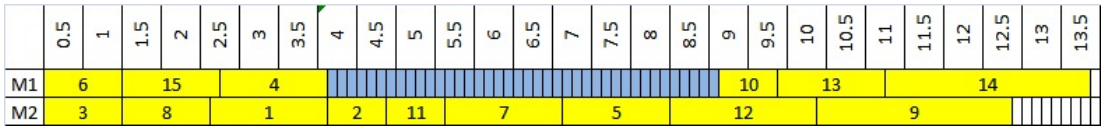

Figure 5.32: Schedule in node 29 at the end of the STEP 6

Then, we solve the NLP model in STEP 7 for the disrupted machine and we get the schedule with optimal processing times on the disrupted machine for given sequence and TADC upper bound as in Figure 5.33.

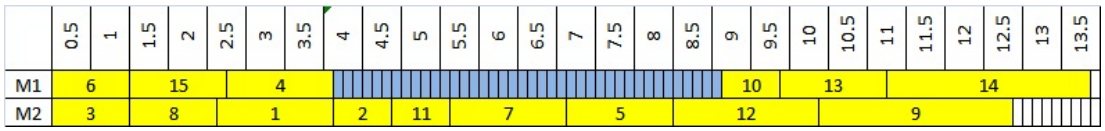

Figure 5.33: Schedule in node 29 found by STEP 7

After we solve the NLP model in STEP 7 for all the selected nodes at the last level, we obtain the cost values as it is given in Figure 5.34.

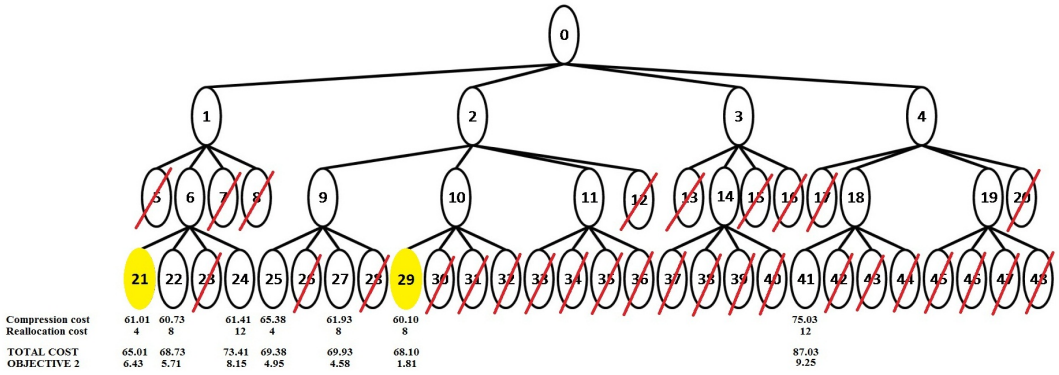

Figure 5.34: Objective values at the end of the third level

As it can be seen in Figure 5.34 that, when we apply STEP 7 for the selected nodes of the last level, non-dominated solutions correspond to the nodes 21 and 29. Then, we give the TADC values of these nodes as an upper bound to the mathematical model given in Chapter 3 and find the optimal schedules for these TADC values.

For solution 1 which corresponds to node 21, we first moved the disrupted job 2

to the first position on the disrupted machine, then we moved the second disrupted job 7 to the first position of the 2^{nd} machine and finally we moved the disrupted job 10 to the first position on the disrupted machine, that is, we reallocated only 1 job to different machine and so the reallocation cost is found 4. After these operations, we get the total cost which is the objective 1 as 65.01 and second objective which is TADC value as 6.43 with the schedule given in Figure 5.31. If we give 6.43 as an upper bound for TADC value to the mathematical model, we get the optimal total cost value 58.67 with this given TADC upper bound. So, the deviation between the objective 1 values of the model and algorithm is $(65.01-58.67)/58.67 = 9.75\%$. Optimal schedule for solution 1 is shown in Figure 5.35.

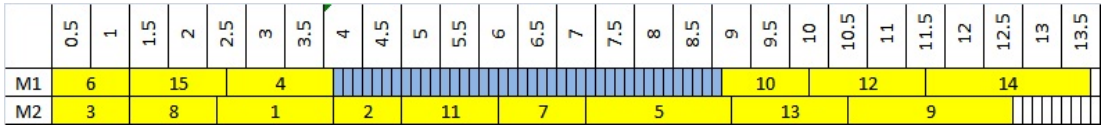

Figure 5.35: Optimal schedule for given TADC upper bound of solution 1

In solution 2 which is the node 29, first disrupted job 2 is moved to the first position of the 2^{nd} machine and then the second disrupted job 7 is also moved to the first position of the 2^{nd} machine and last disrupted job 10 is moved to the first position of the disrupted machine. So, we reallocated two jobs to different machines and we get the reallocation cost 8 and total cost as 68.10 with the second objective TADC value of 1.81. We give this TADC value as an upper bounf to the mathematical model, we get the optimal schedule which is shown in Figure 5.36 with the optimal total cost value of 66.33. The deviation for this non-dominated solution is $(68.10 - 66.33)/66.33 = 2.66\%$.

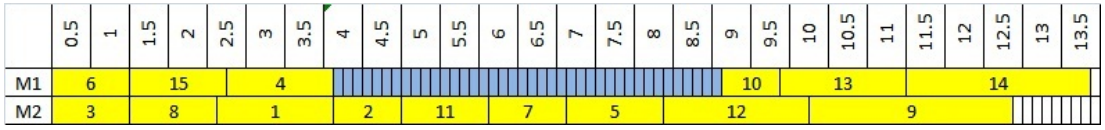

Figure 5.36: Optimal schedule for given TADC upper bound of solution 2

Hence, for these two non-dominated solutions, the average deviation between the objective 1 values of the MIP model and the algorithm with STEP 7 is 6.21 for given upper bound on TADC.

5.3 Summary

In this chapter, we solved numerical examples using the proposed DTA algorithm excluding and including the STEP 7. For each non-dominated solution, we gave the TADC value of the solutions obtained by the algorithm to the mathematical model as an upper bound for TADC value. We compared the objective functions 1 of the solutions obtained by heuristics and the model for given same objective function 2. We observed that the average objective 1 deviation of the non-dominated solutions is decreased from 11.53% to 6.21% with inclusion of the STEP 7 to the algorithm. In the next chapter, we will first present the experimental factors and then analyze the solutions obtained by using the proposed heuristics and the mathematical model.

Chapter 6

Computational Study

In the computational study, we tested the performance of heuristic algorithm for generating approximate efficient solutions. We compared the computation time and solution quality of the exact solution approach and the heuristic algorithm on a set of randomly generated test problems.

6.1 Experimental Factors

We used some experimental factors to generate test problems randomly. These factors and the values that these factors can take are listed in Table 6.1.

Factors	Level 1	Level 2	Level 3
(Job,Machine) Pairs	(40,3)	(50,4)	(60,4)
# of Disrupted Jobs	5	6	
Distance Between Parallel Machines	4	8	16
Reallocation Cost Between Machines	Constant	Linear	

Table 6.1: Experimental design factors

We choosed the (Job,Machine) pairs with the values in Table 6.1, because we solved the model for smaller size problems in terms of number of jobs and number of machines and we observed that when the number of jobs increases to 40 and number of machines is 3, it becomes harder for mathematical model to solve the

problem. Then, we increased the number of jobs to 50 and 60 and number of machines to 4 and solved the problem for these pairs.

Number of disrupted jobs affect the disruption duration. For a problem instance with 50 jobs and 4 machines, there are 12 jobs on one machine on the average. So, 5 job disruption duration is a long time for a machine and it is hard to compensate this duration. So, we increased the number of disrupted jobs by 1 more job and solved the problem for these values (5 and 6).

Distance between machines is very important in our problem since the reallocation cost is a function of the distance which is a part of the objective 1, e.g., total cost. We observed that when the distance between machines is very small such as 4, the mathematical model tries to solve the problem by reallocating more jobs to different machines, since this small reallocation cost can be compensated with the decrease in TADC and compression cost. But as the distance increases to 16, number of reallocated jobs decreased, because reallocation of one job caused the 75% increase in the reallocation cost. So, we solved the problem for three levels of distance between machines.

We used the reallocation cost as another factor in the experimental design. For level 1, we assumed the reallocation cost to be a constant function of the distance between machines. So, reallocation cost of reallocating a job from machine 1 to machine 3 and the reallocation cost of reallcating a job from machine 2 to machine 3 are equal. For level 2, we assume the reallocation cost to be a linear function of the distance between machines. So, when the distance between machines is equal to 8, reallocating a job from machine 1 to machine 3 costs 16, but reallocating a job from machine 1 to machine 2 costs 8.

b_{ij}	Uniform$[10, 20]$
k_{ij}	Uniform$[1.1, 3.1]$
pu_i	Uniform$[1.0, 3.0]$
pl_i	Uniform$[0.1, 2.5]$
γ_j	$1.5 \times (\sum_{i \in N} pu_i/m)$
beamSize b	7

Table 6.2: Parameters

The remaining parameters are selected from a given distribution randomly for each replication. These parameters are given in Table 6.2.

We first generated an original schedule by solving the following mathematical model;

$$\begin{aligned}
&\min \ \textstyle\sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{J}} c_{ij}\ y_{ij} \\
&\text{subject to} \\
&\sum_{j\in\mathcal{J}} y_{ij} = 1, \qquad \forall\ i \in N \qquad (6.1) \\
&\sum_{j\in\mathcal{J}} (pu_i \times y_{ij}) \leq \gamma_j, \forall\ j \in \mathcal{J} \qquad (6.2)
\end{aligned}$$

In this model y_{ij} are decision variables which will be used as parameters in the revised schedule. We try to minimize the manufacturing cost of all jobs at all machines. In Constraint 6.1, we ensure that each job should be assigned to only one machine. Constraint 6.2 implies that, sum of the processing times of the jobs which are assigned to a machine should be less than or equal to the capacity of that machine. So, originally the jobs are processed at their processing time upper bounds.

By this mathematical model, we obtained the original machine–job assignments. Then we sequenced the jobs on a machine according to the SPT (shortest processing time) rule.

After we formed the original schedule, we generated a breakdown by selecting a disrupted machine and disruption start time randomly such that the disruption will start at the first 20% of the job load of the disrupted machine. We set the duration of the disruption as the sum of the original processing times of the disrupted jobs at the disrupted machine. We set the ready times of the machines such that for the disrupted machine it is the disruption end time and for other machines it is the smallest start times of the jobs which are not started to be processed yet at that machine before the disruption start time. Then we found the maximum completion time of each machine in the original schedule.

The heuristics were coded in C++ language and compiled in the Visual Studio

2010 environment. Linear programming models are solved by using the library routines of ILOG CPLEX version 12.1 solver. All the heuristics and the mathematical models were run on a personal computer with 2.20GHz Intel Core 2 Duo CPU and 4GB of RAM.

6.2 Computational Experiments

We ran the algorithm for each instance and get approximate efficient solutions. There are $2 \times 3 \times 2 \times 3 = 36$ full combinations of the experimental factors and we run each combination for 3 replications resulting in 108 randomly generated schedules. For each non-dominated solution of an instance, we gave the TADC value of the solution as an upper bound to the mathematical program and solved the same instance by running the model. Since the problem is NP-hard, when the problem size gets larger, it becomes harder to solve the problem as reported in Table 6.4. We could not get optimal solutions in acceptable amount of CPU time for the instances that are generated by using the factors in Table 6.1. So, we gave a time limit of 10,000 CPU seconds to the mathematical program to solve each instance.

Therefore, we compared the solutions that are found by heuristic algorithm and found by CPLEX within given time limit and TADC value upper bound and we have done the analysis based on the deviation between these solutions. For each instance, we calculated the deviation as follows:

$$\tau = 100 \times (F1^A - F1^M)/F1^M$$

where $F1^A$ is the total cost value of the solution which is found by the proposed heuristic and $F1^M$ is the total cost value of the solution which is found by solving the mathematical program.

For each instance, the algorithm finds a number of solutions in short CPU times in the decision tree as reported in Table 6.4 in detail, whereas the mathematical model runs for 10,000 seconds to obtain one feasible solution.

In Table 6.3 we can see the analysis of deviations obtained by heuristic with

and without STEP 7 of the algorithm for the same replication. We ran the heuristic with STEP 7 and without STEP 7 for replication 2, separately. In Table 6.3, first column represents the inclusion of the STEP 7 in the algorithm. In this column, "With" represents the inclusion of STEP 7 in the algorithm, and "Without" represents the exclusion of STEP 7 from the algorithm. The second column in this table shows the average τ, third column shows maximum of τ, fourth column shows minimum of τ. Fifth column shows the number of efficient solutions obtained with $\tau \geq 0$ and seventh column shows the average τ of these solutions, sixth column shows the number of solutions obtained with $\tau < 0$ and eighth column shows the average τ of these solutions. Ninth column shows the average CPU time that the heuristic takes to obtain these solutions.

It can be seen in Table 6.3 that we obtain results with smaller deviations with STEP 7 in average, although the average CPU time of the heuristic with STEP 7 is greater than the CPU time of heuristic without STEP 7 observably. Since we achieve an almost 10% decrease in average deviation with STEP 7, we accept the CPU time increase of this step and solved the randomly generated problem instances for other 2 replications (1 and 3) by including the STEP 7 and compared the solutions obtained by heuristic with this step and the solutions of the mathematical model. Heuristic with STEP 7 still has the CPU time advantage on the mathematical model. The results of the efficient solutions obtained by the inclusion of STEP 7 and the results of the efficient solutions obtained by the exclusion of STEP 7 are given in Appendix A and B, respectively.

The number of efficient solutions obtained by the heuristic could vary for a given problem instance. Therefore, we obtained a total of 206 efficient solutions, since the heuristic could generate more than one efficient solution for each instance.

STEP 7	Avg (%)	Max (%)	Min (%)	# of Soln. ($\tau \geq 0$)	# of Soln. ($\tau < 0$)	Avg(%) ($\tau \geq 0$)	Avg(%) ($\tau < 0$)	Avg. CPU time(sec.)
Without	-5.60	20.99	-54.10	30	26	13.07	-27.15	7.2
With	-15.19	17.28	-50.61	15	60	6.41	-20.59	482.92

Table 6.3: Analysis of the STEP 7 of the algorithm in terms of τ

(Job,Machine) size	(40,3)		(50,4)		(60,4)	
# of disrupted jobs	5	6	5	6	5	6
# of nodes the algorithm generated	203	252	290	360	290	360
# of nodes at which model in STEP 7 is solved	35	42	35	42	35	42
Selected efficient solutions	7	7	7	7	7	7
Algorithm CPU time(sec.)	344	418	473	585	525	603

Table 6.4: CPU Time and generated solutions

As it can be seen in Table 6.4, CPU time increases as the solution size (in terms of the number of jobs and number of machines) and number of disrupted jobs increase. Since as the number of disrupted jobs increase, the level of decision tree and number of nodes explored in the tree increase. So, it is reflected in the CPU time of algorithm. As it is stated earlier, the mathematical model requires extensive computation time to obtain optimal solution. For the problem instances which are generated randomly for the experimental design, we gave a time limit to the mathematical model and the average gap between the best node and best integer that CPLEX achieved in 10,000 CPU seconds is 76.14%. Note that, the gap is calculated by (best integer − best node)/best node. As it can be seen in Table 6.5, when the problem size gets larger, gap increases, because it gets harder to solve the problem. It is also observed that number of disrupted jobs affects the gap, because it increases the complexity of the problem. These gaps are achieved by the mathematical model which is strengthened by reformulating as a conic quadratic problem. If we did not solve it by conic constraints, the nonlinear mathematical model would not even find these solutions in a given time limit.

(job,mac)	# of disrupted jobs	Min(%)	Avg(%)	Max(%)
40,3	5	34.45	72.69	98.86
	6	45.96	77.54	97.16
50,4	5	48.47	71.55	98.51
	6	56.69	76.33	97.40
60,4	5	59.14	89.13	100
	6	56.97	89.13	100

Table 6.5: Analysis of gap between best node and best integer

# of Soln.	Avg(%)	Max(%)	Min(%)	# of Soln. ($\tau \geq 0$)	# of Soln. ($\tau < 0$)	Avg(%) ($\tau \geq 0$)	Avg(%) ($\tau < 0$)
206	-6.42	22.95	-48.50	81	125	7.12	-15.19

Table 6.6: Analysis of τ

Since the total costs of the solutions which are obtained by the algorithm for different instances are not comparable because of different factors and replications, we analyzed the deviation between the total costs of the solutions which are obtained by the heuristic and the model for given TADC value for each instance of 3 replications as reported in Table 6.6.

In Table 6.6, first column represents the total number of efficient soluitons obtained by the heuristic using the experimental factors. The second column in this table shows the average τ of these solutions, third column shows maximum of τ, fourth column shows minimum of τ among these solutions. Fifth column shows the number of efficient solutions obtained with $\tau \geq 0$ and seventh column shows the average τ of these solutions, sixth column shows the number of solutions obtained with $\tau < 0$ and eighth column shows the average τ of these solutions.

Minimum deviation in Table 6.6 is obtained on the instance of $(60 \text{ jobs}, 4 \text{ machines})$, with 6 disrupted jobs and reallocation cost of 4. Since, the problem size in terms of number of jobs and number of machines is large, the gap between the best node and best integer stayed at 99.99%. Maximum deviation in Table 6.6 is obtained on the instance of $(60 \text{ jobs}, 4 \text{ machines})$ with 6 disrupted jobs and reallocation cost of 16. Since our algorithm tends to reallocate more jobs compared to the mathematical model, when the number of disrupted jobs to be reallocated and the reallocation cost between the machines get higher, this causes the difference between the total reallocation costs of the model and the algorithm to get higher. For this instance with deviation 22.95%, the deviation, between the compression costs of the solutions obtained by algorithm and the mathematical model, is -23.39% (see Table 6.6). Since the algorithm reallocated two more jobs than the model did in this instance, it brought along this deviation between the total costs of two approaches.

Rep	# of Soln.	Avg(%)	Max(%)	Min(%)	# of Soln. ($\tau < 0$)	# of Soln. ($\tau \geq 0$)	Avg(%) ($\tau < 0$)	Avg(%) ($\tau \geq 0$)
1	70	-7.52	14.05	-48.50	49	21	-12.96	5.18
2	66	-13.20	17.28	-39.50	52	14	-18.38	6.04
3	70	1.38	22.95	-35.60	24	46	-12.85	8.34

Table 6.7: Analysis of τ for each replication

Replications cause changes in the deviations as it can be seen in Table 6.7, because replication affects the original schedule. Especially, in replication 3, loads are distributed to the machines unbalanced. Some machines have heavier loads (more jobs), whereas some machines have very few. In other replications, machines have relatively more balanced loads. They have almost same maximum completion times for the revised schedule. This unbalanced job distributions cause higher deviations between heuristic and model in terms of total cost.

# of disrupted jobs	(job,mac)	# of soln.	Avg(%)	Min(%)	Max(%)	Avg(%) ($\tau < 0$)	Avg(%) ($\tau \geq 0$)
5	40,3	36	-8.43	-37.95	6.35	-16.69	3.14
	50,4	34	-3.10	-24.09	19.20	-11.12	7.06
	60,4	35	-10.21	-38.80	17.51	-21.32	9.93
6	40,3	38	-2.45	-35.60	17.28	-10.13	7.04
	50,4	36	-1.70	-44.49	18.94	-10.47	10.58
	60,4	27	-14.88	-48.50	22.95	-21.41	6.17

Table 6.8: Analysis of effects of number of disrupted jobs on τ

Number of disrupted jobs is an important factor in our problem. It affects the duration of the disruption and as it increases, it requires more compression and we need to reallocate more jobs. Another side effect of the number of disrupted jobs factor is that it increases the search tree level in the algorithm, so the computation time also increases as the number of disrupted jobs increases. It also affects the complexity of the problem, as the number of disrupted jobs increases; it becomes harder for the mathematical model to solve each problem. As it can be seen in Table 6.8, τ increases with the increase in the number of disrupted jobs for the instances with (40 jobs, 3 machines) and (50 jobs, 4 machines). This deviation increase is coming from the increase in the reallocation cost. In Table 6.8, we

Reallocation cost function	Distance between two machines	# of solutions	Avg(%)	Min(%)	Max(%)
Constant	4	29	-14.52	-48.5	18.94
	8	36	-6.8	-36.56	18.32
	16	39	-3.22	-37.95	22.95
Linear	4	36	-7.76	-45.17	11.77
	8	36	-5.95	-37.47	19.2
	16	32	-1.55	-38.8	16.61

Table 6.9: Analysis of effects of reallocation cost on τ

can see that only in the instances of (60 jobs, 4 machines), with 6 disrupted jobs, there is a decrease in the average deviation compared to the instances of (60 jobs, 4 machines), with 5 disrupted jobs. The hardest instances to solve for the mathematical model are the instances with (60 jobs, 4 machines) and 6 disrupted jobs. So, the average gap between the best node and the best integer that the model could achieve stayed at 99.90% for these instances.

Reallocation cost is a an important factor in our problem. Since our algorithm tends to reallocate more jobs than the mathematical model does, as the reallocation cost between machines increases, total cost increases sharply and it causes huge deviations between the total costs of the solutions obtained by the algorithm and the mathematical model as it is analyzed in Table 6.9.

6.3 Summary

In this chapter, we presented the experimental factors and the results we obtained by solving the problems with the data generated by the combinations of these factors using the algoritms and the mathematical model we proposed. We analyzed the solutions obtained by model and heuristics under some conditions, statistically. In the next chapter, we will give the conclusion of our study with the final remarks and the future search directions.

Chapter 7

Conclusion

In this study, we focused on rescheduling non-identical parallel machines due to a disruption at one of the machines. Although in many scheduling and rescheduling studies, processing times are assumed to be fixed parameters, in practice, they can be controlled by setting machining parameters or using additional resources. We assumed that the processing times are controllable and we can control them by setting the machining parameters. Processing time controllability is a very important tool in rescheduling, especially if there is a maximum completion time constraint. If the maximum completion time constraint would not exist, right shift scheduling could be applied in the rescheduling and disrupted jobs could be processed by right shifting. Because of the maximum completion time constraint we are restricted to the time limit up to the maximum completion time of each machine. For this reason, processing time controllability is very useful to matchup the maximum completion time of the machines by compressing the processing times. We also utilized the reallocation of the jobs to different machines to be able to realize the processing of the disrupted jobs by matching up the maximum completion time constraint. In many rescheduling studies, although the reallocation of the jobs to different machines is used to compensate for the disruption, cost of this operation is generally neglected. We assumed that the reallocation operation incurs a reallocation cost which is a function of the distance between the machines. Another restriction in our problem is that the

start times of the jobs in the revised schedule cannot be less than the start times of the jobs in the original schedule.

We aimed to minimize the total absolute deviation of job completion times between the original and revised schedule, because even a disruption occurs, it is important to deliver the jobs on time that is promised to be in the original schedule, not earlier or not later. So, processing time controllability and the reallocation became very useful for us, since they gave us the chance of matching up the maximum completion time and the original completion times. But on the other hand, we needed to consider the costs incurred from both compression and reallocation to minimize. Two objectives which are cost and TADC are in conflict, when one of them decreases, the other one increases.

7.1 Contributions

In this study, we aimed to minimize the cost and schedule performance (TADC) at the same time such that both criteria are in conflict in our study. We have taken into account the reallocation cost, although the reallocation cost is generally neglected in the studies in which the reallocation is utilized for the rescheduling. We tried to solve this bi-criteria problem to minimize both of the objectives at the same time. We developed a mathematical model to solve this problem and applied an ϵ-constraint approach (Gurel and Akturk [12]) to handle these two conflicting objectives. We integrated the nonlinear compression cost and the reallocation cost in the cost objective and we regarded the total absolute deviation of the job completion times as the schedule stability objective. We gave an upper bound to the TADC objective function and added it to the constraints and tried to minimize the compression and reallocation costs in the objective function.

In many rescheduling studies in which the processing times are assumed controllable, cost of compression is taken into account as a linear function of processing times. We considered the compression cost function as a nonlinear function

of compression amount. We handled this non-linearity by utilizing the conic quadratic programming with a linear objective and conic quadratic constraints by reformulating the problem.

The problem is NP-hard, so even (40 jobs, 3 machines)-size problems require extensive computation time. So, we proposed a decision tree algorithm (DTA) to solve the problems in reasonable computation time. We could obtain solutions by this algorithm in very short CPU time. We used some experimental factors and generated problem instances for different combinations of these factors randomly. We ran the heuristic for each randomly generated problem instance and we obtained efficient solutions for each instance. Afterwards, we gave the TADC value of each solution of each instance to the mathematical model as an upper bound and solved the model for each solution of each problem instance. We obtained very close solutions with the algorithm and the model for each solution as summarized in Chapter 6.

7.2 Future Research Directions

In this study, we integrated the compression cost and the reallocation cost in the cost objective. We concluded that the main drawback of our heuristics is the reallocation. Although we obtained very close solutions with the heuristics and the mathematical model in terms of the compression cost, the maximum deviation between the solutions of the heuristics and the mathematical model is obtained at the instances with the maximum reallocation cost and maximum number of disrupted jobs. Because our proposed search algorithm tends to reallocate the jobs as much as possible in order to decrease the compression cost by trying to relax the disrupted machine. On the other hand, in STEP 7 of the algorithm, we give the schedule of all machines to the NLP model and we find the optimal processing times for this sequence. For this reason, compression cost is minimized optimally for the given sequence by the heuristics, but reallocation cost is obtained by the heuristics further from the optimal reallocation cost. We have foreseen this, so we used the reallocation cost factor in the experimental design and when

we compared the solutions obtained by the heuristics and the mathematical model for the same instance, we observed that, the heuristics generally reallocated more jobs than the mathematical model did. Hence, in the instances with the maximum reallocation cost which is a linear function of the distance between machines, we have seen that one more reallocated job caused the total cost to increase almost 25-30%. So, in the future research, the heuristics can be improved to handle with the greater reallocation costs.

As we stated earlier, there are many scheduling stability objectives. In the future research, another stability objective can be studied. For instance, we regarded the reallocated jobs in the cost objective by considering the reallocation cost. It can be also regarded as a schedule stability objective by considering the number of reallocated jobs to different machines.

As another extension to this problem, as we considered the TADC in the scheduling objective, the deviation between the original and revised job completion times can also be regarded as earliness and tardiness and stability objective can be the total cost of earliness and tardiness. This problem can also be supported by relaxing the constraint which does not allow the revised start times to be earlier than the original start times.

Bibliography

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

[2] M. S. Akturk and T. Ilhan. Single CNC machine scheduling with controllable processing times to minimize total weighted tardiness. *Computers and Operations Research*, 38:771–781, 2011.

[3] M. S. Akturk, A. Atamturk, and S. Gurel. A strong conic quadratic reformulation for machine–job assignment with controllable processing times. *Operations Research Letters*, 37:187–191, 2009.

[4] M. S. Akturk, A. Atamturk, and S. Gurel. Parallel machine match-up scheduling with manufacturing cost considerations. *Journal of Scheduling*, 13:95–110, 2010.

[5] O. Alagoz and M. Azizoglu. Rescheduling of identical parallel machines under machine eligibility constraints. *European Journal of Operational Research*, 149:523–532, 2003.

[6] J. P. Arnaout and Rabadi G. Rescheduling of unrelated parallel machines under machine breakdowns. *International Journal of Applied Management Science*, 1:75–89, 2008.

[7] M. Azizoglu and O. Alagoz. Parallel machine rescheduling with machine disruptions. *IIE Transactions*, 37:1113–1118, 2005.

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

[9] J. Curry and B. Peters. Rescheduling parallel machines with stepwise increasing tardiness and machine assignment stability objectives. *International Journal of Production Research*, 43:3231–3246, 2005.

[10] R. L. Daniels and R. K. Sarin. Single machine scheduling with controllable processing times and number of jobs tardy. *Operations Research*, 37:981–984, 1989.

[11] S. Gurel and M. S. Akturk. Considering manufacturing cost and scheduling performance on a CNC turning machine. *European Journal of Operational Research*, 177:325–343, 2007.

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

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

[14] S. Gurel, E. Korpeoglu, and M. S. Akturk. An anticipative scheduling approach with controllable processing times. *Computers and Operations Research*, 37:1002–1013, 2010.

[15] X. Huang and M. Z. Wang. Parallel identical machines scheduling with deteriorating jobs and total absolute differences penalties. *Applied Mathematical Modelling*, 35:1349–1353, 2011.

[16] K. Jansen and M. Mastrolilli. Approximation schemes for parallel machine scheduling with controllable processing times. *Computers and Operations Research*, 31:1565–1581, 2004.

[17] R. K. Kayan and M. S. Akturk. A new bounding mechanism for the CNC machine scheduling problems with controllable processing times. *European Journal of Operational Research*, 167:624–643, 2005.

[18] Y. Leyvand, D. Shabtay, and G. Steiner. Optimal delivery time quotation to minimize total tardiness penalties with controllable processing times. *IIE Transactions*, 42:221–231, 2010.

[19] H. Mokhtari, I. N. K. Abadi, and S. H. Zegordi. Production capacity planning and scheduling in a no-wait environment with controllable processing times: An integrated modeling approach. *Expert Systems with Applications*, 38: 12630–12642, 2011.

[20] B. Mor and G. Mosheiov. Total absolute deviation of job completion times on uniform and unrelated machines. *Computers and Operations Research*, 38:660–665, 2011.

[21] A. C. Nearchou. Scheduling with controllable processing times and compression costs using population-based heuristics. *International Journal of Production Research*, 48:7043–7062, 2010.

[22] D. Oron. Single machine scheduling with simple linear deterioration to minimize total absolute deviation of completion times. *Computers and Operations Research*, 35:2071–2078, 2008.

[23] M. Ozlen and M. Azizoglu. Generating all efficient solutions of a rescheduling problem on unrelated parallel machines. *International Journal of Production Research*, 47:5245–5270, 2009.

[24] M. Ozlen and M. Azizoglu. Rescheduling unrelated parallel machines with total flow time and total disruption cost criteria. *Journal of Operations Research Society*, 62:152–164, 2011.

[25] I. Sabuncuoglu and M. Bayiz. Analysis of reactive scheduling problems in a job shop environment. *European Journal of Operational Research*, 126: 567–586, 2000.

[26] I. Sabuncuoglu and S. Goren. Hedging production schedules against uncertainty in manufacturing environment with a review of robustness and stability research. *International Journal of Computer Integrated Manufacturing*, 22:138–157, 2009.

[27] I. Sabuncuoglu and S. Karabuk. Rescheduling frequency in an FMS with uncertain processing times and unreliable machines. *Journal of Manufacturing Systems*, 18:268–283, 1999.

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

[29] M. A. Trick. Scheduling multiple variable-speed machines. *Operations Research*, 42:23–248, 1994.

[30] A. Turkcan, M. S. Akturk, and R. H. Storer. Predictive/reactive scheduling with controllable processing times and earliness-tardiness penalties. *IIE Transactions*, 41:1080–1095, 2009.

[31] G. E. Vieira and E. Herrmann, J. W.and Lin. Predicting the performance of rescheduling strategies for parallel machine systems. *Journal of Manufacturing Systems*, 19:256–266, 2000.

[32] 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.

[33] J. B. Wang and C. M. Wei. Parallel machine scheduling with a deteriorating maintenance activity and total absolute differences penalties. *Applied Mathematics and Computation*, 217:8093–8099, 2011.

[34] J. B. Wang and Z. Q. Xia. Single machine scheduling problems with controllable processing times and total absolute differences penalties. *European Journal of Operational Research*, 177:638–645, 2007.

[35] K. Xu, Z. Feng, and L. Ke. A branch and bound algorithm for scheduling jobs with controllable processing times on a single machine to meet due dates. *Annals of Operations Research*, 181:303–324, 2010.

[36] K. Xu, Z. Feng, and L. Ke. Single machine scheduling with total tardiness criterion and convex controllable processing times. *Annals of Operations Research*, 186:383–391, 2011.

[37] L. Yedidsion, D. Shabtay, and M. Kaspi. A bicriteria approach to minimize maximal lateness and resource consumption for scheduling a single machine. *Journal of Scheduling*, 10:341–352, 2007.

Appendix A

Results with STEP 7

In the table below, we show the results of the computational studies with the inclusion of STEP 7 in the algorithm. "Prob. no" is the problem number and each line in the table corresponds to an efficient solution. "Rep" is the replication. "'J, M/C" defines the (job, machine) size of the instance. "Dist." is the distance between the parallel machines. "Realloc. Cost Func." shows that the reallocation cost is whether a constant function of the distance between machines or a linear function of the distance between machines. "# of Disr. Jobs" is the number of disrupted jobs for corresponding instance. "# of Non-domin. sol." shows how many non-dominated solutions obtained for given instance. "Non-domin. sol. no" is the non-dominated solution number among all non-dominated solutions of the given instance. "TADC Value" is total absolute deviation of completion times value of corresponding non-dominated solution which is obtained by the algorithm. "Algo. Obj. 1" is the objective 1 value of the algorithm for corresponding non-dominated solution. "Model Obj. 1" is the objective 1 value of the model for given TADC value of this corresponding non-dominated solution. "Best node model achieved" is the objective 1 value obtained by the model for given time limit of 10,000 CPU seconds for corresponding non-dominated solutions.

Prob. no	Rep.	J, M/C	Dist.	Realloc. Cost Func	# of Disr. Jobs	# of Non-domin. sol.	Non-domin. sol. no	TADC value	Algo. Obj. 1	Model Obj. 1	Best node model achieved	Gap between best node and best integer	Deviation between objective 1 of model and algorithm
1	1	50,4	16	linear	5	2	1	37.68	98.2	95.47	37.08	61.16%	2.86%
2	1	50,4	16	linear	5	2	2	31.15	116.56	105.13	37.72	64.12%	10.88%
3	1	50,4	16	constant	5	2	1	37.47	98.03	94.01	32	65.96%	4.28%
4	1	50,4	16	constant	5	2	2	32.21	104.1	103.77	37.47	63.89%	0.32%
5	1	50,4	8	constant	5	1	1	25.27	71.52	87.72	25.61	70.80%	-18.46%
6	1	50,4	8	linear	5	3	1	36.33	85.13	85.14	12.67	85.12%	-0.01%
7	1	50,4	8	linear	5	3	2	29.75	86.19	90.83	18.73	79.38%	-5.11%
8	1	50,4	8	linear	5	3	3	27.6	87.77	87.98	26.34	70.06%	-0.24%
9	1	40,3	8	linear	5	1	1	29.6	66.67	75.16	24.22	67.78%	-11.30%
10	1	40,3	8	constant	5	1	1	29.6	66.67	66.57	24.5	63.20%	0.15%
11	1	40,3	16	linear	5	2	1	32.26	76.3	77.76	50.17	35.48%	-1.88%
12	1	40,3	16	linear	5	2	2	29.6	82.67	79.42	52.06	34.45%	4.09%
13	1	40,3	16	constant	5	2	1	32.26	76.3	77.29	48.28	37.53%	-1.29%
14	1	40,3	16	constant	5	2	2	29.6	82.67	78.9	51.05	35.30%	4.78%
15	1	40,3	16	constant	6	3	1	44.78	93.58	92.23	32	65.30%	1.47%
16	1	40,3	16	constant	6	3	2	41.54	98.38	95.57	34.2	63.98%	2.95%
17	1	40,3	16	constant	6	3	3	41.19	101.18	96.41	25.65	73.39%	4.95%

continued from previous page

Prob. no	Rep.	J, M/C	Dist.	Realloc. Cost Func	# of Disr. Jobs	# of Non-domin. sol.	Non-domin. sol. no	TADC value	Algo. Obj. 1	Model Obj. 1	Best node model achieved	Gap between best node and best integer	Deviation between objective 1 of model and algorithm
18	1	40,3	16	linear	6	3	1	44.78	93.58	95.54	32	66.50%	-2.05%
19	1	40,3	16	linear	6	3	2	41.54	98.38	96.9	37.26	61.55%	1.53%
20	1	40,3	16	linear	6	3	3	41.19	101.18	95.16	31.91	66.47%	6.33%
21	1	40,3	8	linear	6	2	1	41.54	82.38	89.83	13.42	85.06%	-8.29%
22	1	40,3	8	linear	6	2	2	41.19	85.18	89.78	10.35	88.48%	-5.13%
23	1	40,3	8	constant	6	2	1	41.19	85.18	87.14	10.88	87.55%	-2.25%
24	1	40,3	8	constant	6	2	2	41.54	82.38	82.68	16	80.65%	-0.36%
25	1	50,4	4	constant	5	1	1	22.93	58.23	72.17	15.6	78.39%	-19.32%
26	1	50,4	4	linear	5	2	1	30.56	73.14	80.78	8.69	89.25%	-9.46%
27	1	50,4	4	linear	5	2	2	31.01	70.08	75.79	10.22	86.52%	-7.53%
28	1	50,4	4	linear	6	2	1	36.14	92.97	95.61	9.56	90.00%	-2.76%
29	1	50,4	4	linear	6	2	2	41.76	83.38	128.59	3.34	97.40%	-35.16%
30	1	50,4	4	constant	6	1	1	50.29	73.3	86.83	5.33	93.86%	-15.58%
31	1	50,4	8	constant	6	1	1	34.4	85.59	98.45	21.3	78.36%	-13.06%
32	1	50,4	8	linear	6	2	1	41.76	99.38	101.49	17.14	83.11%	-2.08%
33	1	50,4	8	linear	6	2	2	40.43	101.55	105.12	11.69	88.88%	-3.40%

continued from previous page

Prob. no	Rep.	J, M/C	Dist.	Realloc. Cost Func	# of Disr. Jobs	# of Non-domin. sol.	Non-domin. sol. no	TADC value	Algo. Obj. 1	Model Obj. 1	Best node model achieved	Gap between best node and best integer	Deviation between objective 1 of model and algorithm
34	1	50,4	16	linear	6	2	1	46.43	117.84	114.03	25.41	77.69%	3.34%
35	1	50,4	16	linear	6	2	2	31.96	147.15	130.06	45.18	65.26%	13.15%
36	1	50,4	16	constant	6	4	1	50.69	105.12	116.57	15.66	86.54%	-9.82%
37	1	50,4	16	constant	6	4	2	45.86	115.38	122.26	44.27	63.78%	-5.63%
38	1	50,4	16	constant	6	4	3	41.81	115.62	116.89	29.69	74.59%	-1.09%
39	1	50,4	16	constant	6	4	4	40.4	133.76	125.97	25.86	79.48%	6.18%
40	1	40,3	4	constant	6	1	1	37.07	85.99	79.72	11.59	85.46%	7.87%
41	1	40,3	4	linear	6	2	1	46.16	89.93	98.08	7.71	92.14%	-8.32%
42	1	40,3	4	linear	6	2	2	42.55	92.74	100.89	6.6	93.46%	-8.08%
43	1	40,3	4	linear	5	3	1	30.92	70.88	83.05	12.44	85.03%	-14.65%
44	1	40,3	4	linear	5	3	2	30.36	79.1	78.66	11.23	85.72%	0.57%
45	1	40,3	4	linear	5	3	3	29.75	82.15	80.97	11.81	85.42%	1.46%
46	1	40,3	4	constant	5	1	1	26.99	65.23	92.83	1.05	98.86%	-29.73%
47	1	60,4	4	constant	5	2	1	32.75	63.18	68.54	6.11	91.08%	-7.82%
48	1	60,4	4	constant	5	2	2	29.59	65.65	57.46	9.34	83.74%	14.25%
49	1	60,4	4	linear	5	2	1	30.74	65.52	58.65	8.1	86.19%	11.71%

continued from previous page

Prob. no	Rep.	J, M/C	Dist.	Realloc. Cost Func	# of Disr. Jobs	# of Non-domin. sol.	Non-domin. sol. no	TADC value	Algo. Obj. 1	Model Obj. 1	Best node model achieved	Gap between best node and best integer	Deviation between objective 1 of model and algorithm
50	1	60,4	4	linear	5	2	2	29.59	65.65	66.06	11.19	83.07%	-0.62%
51	1	60,4	8	linear	5	3	1	49.81	49.5	68.38	15.12	77.89%	-27.60%
52	1	60,4	8	linear	5	3	2	48.71	49.73	79.53	16.09	79.76%	-37.47%
53	1	60,4	8	linear	5	3	3	48.1	57.86	75.92	17.03	77.56%	-23.79%
54	1	60,4	8	constant	5	3	1	49.81	49.5	61.43	16.15	77.29%	-19.41%
55	1	60,4	8	constant	5	3	2	48.71	49.73	43.6	16.12	77.42%	14.05%
56	1	60,4	8	constant	5	3	3	48.1	57.86	75.42	19.41	100.00%	-23.29%
57	1	60,4	16	constant	5	2	1	57.44	74.52	86.55	22.11	74.46%	-13.90%
58	1	60,4	16	constant	5	2	2	47.72	75.2	75.23	23.05	75.07%	-0.03%
59	1	60,4	16	linear	5	3	1	47.8	72.25	108.6	32	62.90%	-33.47%
60	1	60,4	16	linear	5	3	2	47.4	73.48	75.33	29.06	66.91%	-2.45%
61	1	60,4	16	linear	5	3	3	47.31	79.64	75.21	34.96	59.14%	5.88%
62	1	60,4	16	linear	6	2	1	55.37	94.86	102.49	0	100.00%	-7.44%
63	1	60,4	16	linear	6	2	2	54.32	97.01	103.15	0	100.00%	-5.95%
64	1	60,4	16	constant	6	2	1	67.63	95.35	114.67	0	100.00%	-16.85%
65	1	60,4	16	constant	6	2	2	55.85	99.18	117.62	0	100.00%	-15.68%

continued from previous page

Prob. no	Rep.	J, M/C	Dist.	Realloc. Cost Func	# of Disr. Jobs	# of Non-domin. sol.	Non-domin. sol. no	TADC value	Algo. Obj. 1	Model Obj. 1	Best node model achieved	Gap between best node and best integer	Deviation between objective 1 of model and algorithm
66	1	60,4	8	constant	6	1	1	54.87	74.6	101.16	1.67	98.31%	-26.26%
67	1	60,4	8	constant	6	2	2	47.6	102.99	92.16	7.24	92.14%	11.75%
68	1	60,4	8	linear	6	1	1	54.87	74.6	95.33	0.08	99.91%	-21.75%
69	1	60,4	4	linear	6	1	1	56.89	49.63	90.51	8.26	89.86%	-45.17%
70	1	60,4	4	constant	6	1	1	56.89	49.63	96.37	8.63	87.95%	-48.50%
71	1	60,4	4	constant	6	1	1	31.96	84.47	77.41	11.27	85.44%	9.13%
72	2	60,4	4	linear	6	3	1	41.69	63.08	69.38	2.53	96.36%	-9.08%
73	2	60,4	4	linear	6	3	2	40.76	63.9	124.62	2.4	98.07%	-48.72%
74	2	60,4	4	linear	6	3	3	40,33	64.65	106.86	3.3	96.91%	-39.50%
75	2	60,4	4	constant	6	3	1	41.69	63.08	77.65	2.14	97.25%	-18.76%
76	2	60,4	4	constant	6	3	2	40.76	63.9	87.03	2.5	97.13%	-26.58%
77	2	60,4	4	constant	6	3	3	40,33	64.65	87.62	2.01	97.71%	-26.21%
78	2	60,4	8	constant	6	2	1	42.23	81.61	108.42	0.91	99.16%	-24.73%
79	2	60,4	8	constant	6	2	2	41.37	87.71	95.9	2.98	96.89%	-8.54%
80	2	60,4	8	linear	6	3	1	45.69	76.29	114.17	0	100.00%	-33.18%
81	2	60,4	8	linear	6	3	2	42.23	81.61	97.88	2.3	97.65%	-16.63%

continued from previous page

Prob. no	Rep.	J, M/C	Dist.	Realloc. Cost Func	# of Disr. Jobs	# of Non-domin. sol.	Non-domin. sol. no	TADC value	Algo. Obj. 1	Model Obj. 1	Best node model achieved	Gap between best node and best integer	Deviation between objective 1 of model and algorithm
82	2	60,4	8	linear	6	3	3	41.37	87.71	84.35	2.71	96.78%	3.98%
83	2	60,4	16	linear	6	1	1	36.88	99	98.84	20.48	79.28%	0.15%
84	2	60,4	16	constant	6	1	1	37.24	99.78	106.71	16.87	84.19%	-6.49%
85	2	60,4	16	constant	5	1	1	36.96	71.28	104.06	18.73	82.00%	-31.49%
86	2	60,4	16	linear	5	2	1	36.96	71.28	116.48	16.55	85.80%	-38.80%
87	2	60,4	16	linear	5	2	2	32.82	73.24	97.7	34.23	64.96%	-25.04%
88	2	60,4	8	linear	5	3	1	38.71	63.74	98.35	7.6	92.26%	-35.19%
89	2	60,4	8	linear	5	3	2	37.54	65.78	88	7.57	91.39%	-25.25%
90	2	60,4	8	linear	5	3	3	37.48	67.71	94.97	8.41	91.14%	-28.70%
91	2	60,4	8	constant	5	4	1	39.63	55.21	87.02	8	90.81%	-36.56%
92	2	60,4	8	constant	5	4	2	38.71	63.74	95.62	8.14	91.49%	-33.34%
93	2	60,4	8	constant	5	4	3	37.54	65.78	86.84	6.77	92.20%	-24.25%
94	2	60,4	8	constant	5	4	4	37.48	67.71	95.83	5.22	94.55%	-29.34%
95	2	50,4	4	linear	5	4	1	20.08	61.56	67.22	13.05	80.58%	-8.41%
96	2	50,4	4	linear	5	4	2	18.44	62.69	67.18	12.23	81.80%	-6.68%
97	2	50,4	4	linear	5	4	3	17.88	63.88	60.66	15.38	74.65%	5.32%

continued from previous page

Prob. no	Rep.	J, M/C	Dist.	Realloc. Cost Func	# of Disr. Jobs	# of Non-domin. sol.	Non-domin. sol. no	TADC value	Algo. Obj. 1	Model Obj. 1	Best node model achieved	Gap between best node and best integer	Deviation between objective 1 of model and algorithm
98	2	50,4	4	linear	5	4	4	16.91	65.15	62.76	13.67	78.22%	3.81%
99	2	50,4	4	constant	5	3	1	18.79	52.66	55.19	14.75	73.27%	-4.59%
100	2	50,4	4	constant	5	3	2	17.97	54.02	56.32	15.03	73.31%	-4.09%
101	2	50,4	4	constant	5	3	3	17.2	55.71	57.25	14.05	75.46%	-2.70%
102	2	50,4	8	constant	5	4	1	18.79	68.66	74.62	29.63	60.29%	-8.00%
103	2	50,4	8	constant	5	4	2	17.97	70.02	88.98	38.73	56.48%	-21.31%
104	2	50,4	8	constant	5	4	3	17.2	71.71	76.95	32	58.41%	-6.81%
105	2	50,4	8	constant	5	4	4	15.35	82.55	78.24	32	59.10%	5.50%
106	2	50,4	8	linear	5	3	1	22.18	74.87	68.06	32.62	52.08%	10.01%
107	2	50,4	8	linear	5	3	2	20.59	78.65	74.99	24.86	66.85%	4.88%
108	2	50,4	8	linear	5	3	3	18.44	78.69	74.58	32	57.08%	5.52%
109	2	50,4	16	linear	5	1	1	23.39	93.87	95.11	48.76	48.74%	-1.31%
110	2	50,4	16	constant	5	2	1	24.12	93.84	149.73	4.79	94.28%	-37.33%
111	2	50,4	16	constant	5	2	2	20.56	110.62	98.97	51	48.47%	11.77%
112	2	50,4	16	constant	6	2	1	30.17	135.26	138.06	48.02	65.22%	-2.03%
113	2	50,4	16	constant	6	2	2	30.09	137.04	277.46	16.44	94.08%	-50.61%

continued from previous page

Prob. no	Rep.	J, M/C	Dist.	Realloc. Cost Func	# of Disr. Jobs	# of Non-domin. sol.	Non-domin. sol. no	TADC value	Algo. Obj. 1	Model Obj. 1	Best node model achieved	Gap between best node and best integer	Deviation between objective 1 of model and algorithm
114	2	50,4	16	linear	6	1	1	31.32	141.81	137.84	50.33	63.48%	2.88%
115	2	50,4	8	linear	6	1	1	23.74	112.66	107.19	30.46	71.58%	5.11%
116	2	50,4	8	constant	6	3	1	23.4	92.9	125.73	40.78	67.57%	-26.11%
117	2	50,4	8	constant	6	3	2	19.67	101.96	110.07	34.2	68.93%	-7.37%
118	2	50,4	8	constant	6	3	3	18.41	104.66	119.32	38.06	68.10%	-12.28%
119	2	50,4	4	constant	6	3	1	24.55	76.15	90.98	15.04	83.46%	-16.30%
120	2	50,4	4	constant	6	3	2	22.29	82.11	86.35	15.38	82.18%	-4.91%
121	2	50,4	4	constant	6	3	3	19.98	83.27	81.62	16.07	80.32%	2.01%
122	2	50,4	4	linear	6	1	1	19.98	87.27	95.55	16.33	82.91%	-8.67%
123	2	40,3	4	linear	6	3	1	28.89	87.15	107.24	12.47	88.37%	-18.73%
124	2	40,3	4	linear	6	3	2	27.98	92.22	100.61	11.87	88.20%	-8.34%
125	2	40,3	4	linear	6	3	3	26.26	98.15	102.26	13.62	86.68%	-4.02%
126	2	40,3	4	constant	6	2	1	25.76	73.3	96.55	5.92	93.87%	-24.08%
127	2	40,3	4	constant	6	2	2	25.33	77.91	88.57	5.93	93.31%	-12.03%
128	2	40,3	8	constant	6	2	1	31.49	92.8	97.75	21.16	78.36%	-5.07%
129	2	40,3	8	constant	6	2	2	27.97	93.15	103.09	9.3	90.98%	-9.64%

continued from previous page

Prob. no	Rep.	J, M/C	Dist.	Realloc. Cost Func	# of Disr. Jobs	# of Non-domin. sol.	Non-domin. sol. no	TADC value	Algo. Obj. 1	Model Obj. 1	Best node model achieved	Gap between best node and best integer	Deviation between objective 1 of model and algorithm
130	2	40,3	8	linear	6	2	1	34.34	120.36	104.32	18.35	82.41%	15.37%
131	2	40,3	8	linear	6	2	2	29.16	125.76	107.23	24.35	77.29%	17.28%
132	2	40,3	16	linear	6	1	1	33.42	130.47	127.18	48	62.26%	2.58%
133	2	40,3	16	constant	6	1	1	30.25	114.29	122.8	42.36	65.50%	-6.93%
134	2	40,3	16	constant	5	3	1	35.22	86.49	123.61	35.24	71.49%	-30.02%
135	2	40,3	16	constant	5	3	2	31.15	87.59	118.11	47.36	59.90%	-25.84%
136	2	40,3	16	constant	5	3	3	26.32	88.68	142.92	22.69	84.12%	-37.95%
137	2	40,3	16	linear	5	2	1	38.24	93.61	133.08	22.85	82.83%	-29.66%
138	2	40,3	16	linear	5	2	2	32.02	96.24	122.79	44.69	65.30%	-21.62%
139	2	40,3	8	linear	5	3	1	34.02	89.25	105.12	19.4	81.54%	-15.10%
140	2	40,3	8	linear	5	3	2	29.16	97.04	109.6	21.93	80.00%	-11.46%
141	2	40,3	8	linear	5	3	3	28.32	100.34	106.71	25.2	76.38%	-5.97%
142	2	40,3	8	constant	5	1	1	26.32	64.68	88.63	25.94	70.74%	-27.03%
143	2	40,3	4	constant	5	2	1	22.27	52.28	85.63	14.12	83.50%	-38.94%
144	2	40,3	4	constant	5	2	2	21.36	57.13	87.68	12.92	85.26%	-34.84%
145	2	40,3	4	linear	5	2	1	24.67	62.09	98.19	14.17	85.57%	-36.77%

continued from previous page

Prob. no	Rep.	J, M/C	Dist.	Realloc. Cost Func	# of Disr. Jobs	# of Non-domin. sol.	Non-domin. sol. no	TADC value	Algo. Obj. 1	Model Obj. 1	Best node model achieved	Gap between best node and best integer	Deviation between objective 1 of model and algorithm
146	2	40,3	4	linear	5	2	2	24.16	70.25	109.29	15.93	85.43%	-35.72%
147	3	40,3	4	constant	5	1	1	23.46	43.89	53.38	12	77.52%	-17.79%
148	3	40,3	8	constant	5	3	1	29.41	57.53	55.54	16	71.59%	3.58%
149	3	40,3	8	constant	5	3	2	27.17	59.9	61.27	17.77	70.98%	-2.23%
150	3	40,3	8	constant	5	3	3	27.09	61.61	57.94	16.95	70.74%	6.35%
151	3	40,3	16	constant	5	4	1	39.73	70.94	71.9	5.96	91.71%	-1.34%
152	3	40,3	16	constant	5	4	2	35.72	72.68	71.78	16.2	77.41%	1.26%
153	3	40,3	16	constant	5	4	3	34.81	73.92	72.55	16.67	77.03%	1.89%
154	3	40,3	16	constant	5	4	4	33.03	77.69	74.04	18.07	75.60%	4.93%
155	3	40,3	16	constant	6	2	1	45.34	103.39	96.12	13.65	85.80%	7.56%
156	3	40,3	16	constant	6	2	2	43.06	112.42	96.4	16.41	82.98%	16.61%
157	3	40,3	8	constant	6	3	1	40.62	78.54	84.08	5.82	93.07%	-6.60%
158	3	40,3	8	constant	6	3	2	39.76	78.92	77.21	9.17	88.13%	2.22%
159	3	40,3	8	constant	6	3	3	37.19	81.43	83.52	10.18	87.81%	-2.49%
160	3	40,3	4	constant	6	2	1	29.96	59.82	92.88	2.64	97.16%	-35.60%
161	3	40,3	4	constant	6	2	2	28.02	60.7	58.77	11.4	80.61%	3.28%

continued from previous page

Prob. no	Rep.	J, M/C	Dist.	Realloc. Cost Func	# of Disr. Jobs	# of Non-domin. sol.	Non-domin. sol. no	TADC value	Algo. Obj. 1	Model Obj. 1	Best node model achieved	Gap between best node and best integer	Deviation between objective 1 of model and algorithm
162	3	50,4	4	linear	6	3	1	24.12	87.97	96.97	12.41	87.20%	-9.28%
163	3	50,4	4	linear	6	3	2	23.6	91.91	82.23	13.74	83.29%	11.77%
164	3	50,4	4	linear	6	3	3	22.18	94.47	85.44	13.56	84.13%	10.58%
165	3	50,4	4	constant	6	1	1	24.63	79.81	67.1	11.73	82.52%	18.94%
166	3	50,4	8	constant	6	1	1	26.56	97.13	84.12	25.3	69.93%	15.47%
167	3	50,4	8	linear	6	3	1	23.54	117.25	104.35	26.27	74.83%	12.36%
168	3	50,4	8	linear	6	3	2	22.87	122.69	105.1	26.22	75.05%	16.74%
169	3	50,4	8	linear	6	3	3	21.37	125.23	102.98	29.59	71.27%	21.60%
170	3	50,4	16	linear	6	3	1	32.93	149.52	175.49	7.15	95.92%	-14.80%
171	3	50,4	16	linear	6	3	2	29.61	152.01	120.71	45.86	62.01%	25.93%
172	3	50,4	16	linear	6	3	3	28.11	156.18	122.26	49.27	59.70%	27.75%
173	3	50,4	16	constant	6	4	1	32.69	121.35	104.42	34.77	66.71%	16.20%
174	3	50,4	16	constant	6	4	2	26.71	127.66	116.91	45.22	61.33%	9.20%
175	3	50,4	16	constant	6	4	3	26.02	141.33	124.15	41.48	66.59%	13.84%
176	3	50,4	16	constant	6	4	4	24.42	142.46	158.81	19.81	87.43%	-10.29%
177	3	50,4	16	constant	5	1	1	27.04	82.77	74.71	32.54	56.45%	10.79%

continued from previous page

Prob. no	Rep.	J, M/C	Dist.	Realloc. Cost Func	# of Disr. Jobs	# of Non-domin. sol.	Non-domin. sol. no	TADC value	Algo. Obj. 1	Model Obj. 1	Best node model achieved	Gap between best node and best integer	Deviation between objective 1 of model and algorithm
178	3	50,4	8	constant	5	3	1	27.04	66.77	60.36	16.63	72.45%	10.62%
179	3	50,4	8	constant	5	3	2	23.87	76.16	64.37	18.04	71.98%	18.32%
180	3	50,4	8	constant	5	3	3	21.04	83.04	103.08	5.34	94.82%	-19.44%
181	3	50,4	8	linear	5	1	1	20.88	87.47	73.38	24	67.29%	19.20%
182	3	50,4	4	linear	5	2	1	20.88	63.47	63.22	12	81.02%	0.39%
183	3	50,4	4	linear	5	2	2	19.24	67.25	65.22	12.65	80.61%	3.12%
184	3	50,4	4	constant	5	1	1	19.24	55.25	65.22	13.75	78.92%	-15.28%
185	3	60,4	8	linear	5	2	1	38.47	59.25	63.14	4.45	92.95%	-6.16%
186	3	60,4	8	linear	5	2	2	29.89	66.79	71.01	12.5	82.39%	-5.95%
187	3	60,4	8	constant	5	3	1	48.61	56.69	50,48	0.06	99.89%	12.31%
188	3	60,4	8	constant	5	3	2	39.84	58.63	55.96	4.6	91.78%	4.77%
189	3	60,4	8	constant	5	3	3	31.27	66.16	61.91	12.22	80.27%	6.88%
190	3	60,4	4	constant	5	3	1	40.45	50.76	72.99	0.53	99.27%	-30.46%
191	3	60,4	4	constant	5	3	2	38.65	50.98	47.12	1.09	97.69%	8.19%
192	3	60,4	4	constant	5	3	3	32.52	56.29	47.91	4.72	90.15%	17.51%
193	3	60,4	4	linear	5	2	1	40.45	54.76	57.11	0.66	98.84%	-4.12%

continued from previous page

Prob. no	Rep.	J, M/C	Dist.	Realloc. Cost Func	# of Disr. Jobs	# of Non-domin. sol.	Non-domin. sol. no	TADC value	Algo. Obj. 1	Model Obj. 1	Best node model achieved	Gap between best node and best integer	Deviation between objective 1 of model and algorithm
194	3	60,4	4	linear	5	2	2	20.81	57.94	59.46	11.9	79.99%	-2.56%
195	3	60,4	16	linear	5	2	1	42.61	80.75	76.01	3.17	95.84%	6.24%
196	3	60,4	16	linear	5	2	2	41.43	81.1	71.44	5.64	92.11%	13.53%
197	3	60,4	16	constant	5	2	1	62.68	62.24	58.99	0	100.00%	5.52%
198	3	60,4	16	constant	5	2	2	52.17	71.55	63.58	0	100.00%	12.53%
199	3	60,4	16	constant	6	1	1	55.73	94.46	92.83	1.57	98.31%	1.75%
200	3	60,4	16	linear	6	3	1	81.3	88.14	83.98	0	100.00%	4.95%
201	3	60,4	16	linear	6	3	2	69.45	96.28	94.99	0	100.00%	1.35%
202	3	60,4	16	linear	6	3	3	53.05	109.68	99.92	4.18	95.82%	9.77%
203	3	60,4	8	linear	6	1	1	54.36	79.09	105.15	0	100.00%	-24.79%
204	3	60,4	8	constant	6	1	1	54.54	78.81	75.14	0.48	99.36%	4.88%
205	3	60,4	4	constant	6	2	1	70.68	89.66	95.2	0	100.00%	-5.82%
206	3	60,4	4	constant	6	2	2	38.11	90.94	136.61	4.39	96.79%	-33.43%
207	3	60,4	4	linear	6	1	1	47.45	89.47	85.58	2.17	97.47%	4.55%

Table A.1: Results with STEP 7

Appendix B

Results without STEP 7

In the table below, we show the results of the computational studies by excluding STEP 7 from the algorithm.

Prob. no	Rep.	J, M/C	Dist.	Realloc. Cost Func	# of Disr. Jobs	# of Non-domin. sol.	Non-domin. sol. no	TADC value	Algo. Obj. 1	Model Obj. 1	Gap between best node and best integer	Deviation between objective 1 of model and algorithm
1	2	60,4	4	constant	6	3	1	51.4278	84.5472	90.673	99.60%	-6.76%
2	2	60,4	4	constant	6	3	2	49.6004	88.8772	101.89	99.60%	-12.77%
3	2	60,4	4	constant	6	3	3	47.5693	90.5279	94.375	99.34%	-4.08%
4	2	60,4	4	linear	6	3	1	51.4278	84.5472	117.115	99.85%	-27.81%
5	2	60,4	4	linear	6	3	2	49.6004	88.8772	105.893	99.60%	-16.07%
6	2	60,4	4	linear	6	3	3	47.5693	90.5279	104.136	99.44%	-13.07%
7	2	60,4	8	linear	6	1	1	38.2389	108.706	95.283	91.86%	14.09%
8	2	60,4	8	constant	6	1	1	38.2389	108.706	96.553	91.02%	12.59%
9	2	60,4	16	constant	6	4	1	44.9433	121.687	108.644	95.81%	12.01%
10	2	60,4	16	constant	6	4	2	44.4934	127.467	110.351	94.65%	15.51%
11	2	60,4	16	constant	6	4	3	43.2396	129.096	115.147	90.71%	12.11%
12	2	60,4	16	constant	6	4	4	42.6302	129.378	117.398	86.37%	10.20%
13	2	60,4	16	linear	6	2	1	40,3599	120.489	111.011	85.30%	8.54%
14	2	60,4	16	linear	6	2	2	38.0376	124.46	135.8368	90.74%	-8.38%
15	2	60,4	16	linear	5	3	1	47.8011	72.2523	108.6034	80.39%	-33.47%
16	2	60,4	16	linear	5	3	2	47.3997	73.4789	75.3268	85.02%	-2.45%
17	2	60,4	16	linear	5	3	3	47.3107	79.6394	75.2149	79.98%	5.88%

continued from previous page

Prob. no	Rep.	J, M/C	Dist.	Realloc. Cost Func	# of Disr. Jobs	# of Non-domin. sol.	Non-domin. sol. no	TADC value	Algo. Obj. 1	Model Obj. 1	Gap between best node and best integer	Deviation between objective 1 of model and algorithm
18	2	60,4	16	constant	5	2	1	57.4426	74.5208	86.55	100.00%	-13.90%
19	2	60,4	16	constant	5	2	2	47.7177	75.204	75.3021	98.98%	-0.13%
20	2	60,4	8	constant	5	3	1	49.8121	49.5042	61.4278	100.00%	-19.41%
21	2	60,4	8	constant	5	3	2	48.7076	48.7056	87.703	98.60%	-44.47%
22	2	60,4	8	constant	5	3	3	48.1019	48.1019	75.4238	100.00%	-36.22%
23	2	60,4	8	linear	5	4	1	49.8121	49.5042	78.979	99.11%	-37.32%
24	2	60,4	8	linear	5	4	2	48.7076	49.7262	72.3987	100.00%	-31.32%
25	2	60,4	8	linear	5	4	3	48.1019	57.8561	76.9766	100.00%	-24.84%
26	2	60,4	8	linear	5	4	4	31.3134	83.9415	64.472	62.32%	30.20%
27	2	60,4	4	linear	5	3	1	49.8908	33.6684	62.0203	100.00%	-45.71%
28	2	60,4	4	linear	5	3	2	48.542	39.1147	85.216	99.91%	-54.10%
29	2	60,4	4	linear	5	3	3	47.0794	49.9278	48.795	99.54%	2.32%
30	2	60,4	4	constant	5	3	1	49.8908	33.6684	67.208	99.94%	-49.90%
31	2	60,4	4	constant	5	3	2	48.542	39.1147	72.7007	99.61%	-46.20%
32	2	60,4	4	constant	5	3	3	47.0794	49.9278	73.05	99.94%	-31.65%
33	2	50,4	4	constant	5	1	1	18.291	77.8735	66.982	77.21%	16.26%

continued from previous page

Prob. no	Rep.	J, M/C	Dist.	Realloc. Cost Func	# of Disr. Jobs	# of Non-domin. sol.	Non-domin. sol. no	TADC value	Algo. Obj. 1	Model Obj. 1	Gap between best node and best integer	Deviation between objective 1 of model and algorithm
34	2	50,4	4	linear	5	2	1	24.8282	67.1541	59.364	80.61%	13.12%
35	2	50,4	4	linear	5	2	2	22.628	83.024	151.077	98.51%	-45.05%
36	2	50,4	8	linear	5	1	1	22.5953	93.4297	158.67	96.54%	-41.12%
37	2	50,4	8	constant	5	1	1	20.5239	80.5835	76.451	65.41%	5.41%
38	2	50,4	16	constant	5	2	1	28.7181	107.049	89.489	70.43%	19.62%
39	2	50,4	16	constant	5	2	2	27.3747	116.441	99.677	72.32%	16.82%
40	2	50,4	16	linear	5	1	1	22.5543	108.829	105.308	74.32%	3.34%
41	2	50,4	16	linear	6	1	1	24.8744	164.484	143.66	58.07%	14.50%
42	2	50,4	16	constant	6	1	1	39.5109	168.552	126.119	56.21%	33.65%
43	2	50,4	8	constant	6	1	1	38.0152	118.438	100.113	54.32%	18.30%
44	2	50,4	8	linear	6	3	1	34.5541	122.807	190.907	97.10%	-35.67%
45	2	50,4	8	linear	6	3	2	30.4979	128.817	112.893	71.09%	14.11%
46	2	50,4	8	linear	6	3	3	28.8159	132.21	105.97	65.32%	24.76%
47	2	50,4	4	linear	6	2	1	24.014	95.7428	81.139	66.34%	18.00%
48	2	50,4	4	linear	6	2	2	23.8528	102.301	85.791	67%	19.24%
49	2	50,4	4	constant	6	1	1	22.2407	111.412	146.548	88.97%	-23.98%

continued from previous page

Prob. no	Rep.	J, M/C	Dist.	Realloc. Cost Func	# of Disr. Jobs	# of Non-domin. sol.	Non-domin. sol. no	TADC value	Algo. Obj. 1	Model Obj. 1	Gap between best node and best integer	Deviation between objective 1 of model and algorithm
50	2	60,4	8	linear	6	1	1	38.2389	108.706	95.283	91.86%	14.09%
51	2	60,4	8	constant	6	1	1	38.2389	108.706	96.553	91.02%	12.59%
52	2	60,4	16	constant	6	4	1	44.9433	121.687	108.644	95.81%	12.01%
53	2	60,4	16	constant	6	4	2	44.4934	127.467	110.351	94.65%	15.51%
54	2	60,4	16	constant	6	4	3	43.2396	129.096	115.147	90.71%	12.11%
55	2	60,4	16	constant	6	4	4	42.6302	129.378	117.398	86.37%	10.20%
56	2	60,4	16	linear	6	1	1	40,3599	120.489	111.011	85.30%	8.54%
57	2	60,4	16	linear	5	1	1	47.3107	79.6394	75.2149	79.98%	5.88%
58	2	60,4	4	linear	5	1	1	47.0794	49.9278	48.795	99.54%	2.32%
59	2	50,4	4	linear	5	1	1	24.8282	67.1541	59.364	80.61%	13.12%
60	2	50,4	8	constant	5	1	1	20.5239	80.5835	76.451	65.41%	5.41%
61	2	50,4	16	linear	6	1	1	24.8744	164.484	143.66	58.07%	14.50%
62	2	50,4	8	linear	6	1	1	30.4979	128.817	112.893	71.09%	14.11%
63	2	40,3	4	constant	6	2	1	20.9664	121.647	118.1596	85.11%	2.95%
64	2	40,3	4	constant	6	2	2	18.5437	123.862	119.7624	86.41%	3.42%
65	2	40,3	4	linear	6	2	1	20.9664	121.647	109.5425	84.00%	11.05%

continued from previous page

Prob. no	Rep.	J, M/C	Dist.	Realloc. Cost Func	# of Disr. Jobs	# of Non-domin. sol.	Non-domin. sol. no	TADC value	Algo. Obj. 1	Model Obj. 1	Gap between best node and best integer	Deviation between objective 1 of model and algorithm
66	2	40,3	4	linear	6	2	2	18.5437	123.862	118.3775	86.02%	4.63%
67	2	40,3	8	constant	6	2	1	26.1904	149.079	134.2058	79.30%	11.08%
68	2	40,3	8	constant	6	2	2	24.6046	155.337	132.3146	78.54%	17.40%
69	2	40,3	16	constant	6	6	1	31.321	172.895	151.7748	51.10%	13.92%
70	2	40,3	16	constant	6	6	2	31.121	175.357	152.0347	54.46%	15.34%
71	2	40,3	16	constant	6	6	3	29.1321	178.81	148.0421	45.96%	20.78%
72	2	40,3	16	constant	6	6	4	28.9321	181.272	154.059	49.98%	17.66%
73	2	40,3	16	constant	6	6	5	27.5809	186.774	155.4153	51.18%	20.18%
74	2	40,3	16	constant	6	6	6	27.3809	189.236	156.8552	50.53%	20.64%

Table B.1: Results without STEP 7