TWO-MACHINE FLOWSHOP SCHEDULING WITH FLEXIBLE OPERATIONS AND 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

Zeynep Uruk

August, 2011

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. Dr. 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. Dr. Hakan Gültekin (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. Dr. 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. Dr. Sinan Gürel

Approved for the Graduate School of Engineering and Science:

Prof. Dr. Levent Onural
Director of the Graduate School of Engineering and Science

ABSTRACT

TWO-MACHINE FLOWSHOP SCHEDULING WITH FLEXIBLE OPERATIONS AND CONTROLLABLE PROCESSING TIMES

Zeynep Uruk
M.S. in Industrial Engineering
Advisor: Prof. Dr. Selim Aktürk
Co-Advisor: Asst. Prof. Dr. Hakan Gültekin
August, 2011

In this study, we consider a two-machine flowshop scheduling problem with identical jobs. Each of these jobs has three operations, where the first operation must be performed on the first machine, the second operation must be performed on the second machine, and the third operation (named as flexible operation) can be performed on either machine but cannot be preempted. Highly flexible CNC machines are capable of performing different operations as long as the required cutting tools are loaded on these machines. The processing times on these machines can be changed easily in albeit of higher manufacturing cost by adjusting the machining parameters like the speed of the machine, feed rate, and/or the depth of cut. The overall problem is to determine the assignment of the flexible operations to the machines and processing times for each job simultaneously, with the bicriteria objective of minimizing the manufacturing cost and minimizing makespan. For such a bicriteria problem, there is no unique optimum but a set of nondominated solutions. Using ϵ constraint approach, the problem could be transformed to be minimizing total manufacturing cost objective for a given upper limit on the makespan objective. The resulting single criteria problem is a nonlinear mixed integer formulation. For the cases where the exact algorithm may not be efficient in terms of computation time, we propose an efficient approximation algorithm.

Keywords: Flexible manufacturing system, controllable processing times, manufacturing cost, makespan, flowshop, scheduling.

ÖZET

ESNEK OPERASYONLAR VE KONTROL EDİLEBİLİR İŞLEM ZAMANLARI İLE İKİ-MAKİNALI AKIŞ TİPİ ÇİZELGELEME

Zeynep Uruk
Endüstri Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Prof. Dr. Selim Aktürk
Yardımcı Tez Yöneticisi: Yrd. Doç. Dr. Hakan Gültekin
Ağustos, 2011

Bu çalışmada iki makinalı akış tipi çizelgeleme problemi ele alınmıştır. Bu işlerin her biri için üç operasyon vardır, ve ilk operasyon sadece birinci makinada işlenebilir, ikinci operasyon sadece ikinci makinada işlenebilir, üçüncü operasyon (esnek operasyon olarak adlandırılır) her iki makinada da işlenebilir fakat işler bölünemez. Büyük ölçüde esnek olan CNC makinaları gerekli kesici uçlar yüklendiği sürece farklı operasyonları işleme kapasitesine sahiptir. Bu makinalardaki işlem zamanları yüksek maliyete rağmen, makina hızı, besleme oranı, ve kesme derinliği gibi makina parametreleri ayarlanarak, kolayca değiştirilebilir. Problemimiz imalat maliyetini ve tamamlanma süresini en aza indiren çift kriterli amaç fonksiyonu ile her bir iş için esnek işlemin makinalara atanmasını ve işlem zamanlarını belirlemektir. Bu şekilde çift kriterli bir problem için, tek bir optimal çözüm yoktur, fakat etkin bir çözüm kümesi vardır. ϵ kısıtı yaklaşımı kullanılarak, problem tamamlanma süresi amaç fonksiyonu üzerinde bir üst limit için imalat maliyetini en aza indiren bir probleme dönüştürülebilir. Ortaya çıkan tek kriterli problem doğrusal olmayan karışık tamsayılı matematiksel bir modeldir. Kesin sonuç veren algoritmanın hesaplama zamanı açısından verimli olmadığı durumlar için, verimli bir yaklaşık algoritma öneriyoruz.

Anahtar sözcükler: Esnek imalat sistemi, kontrol edilebilir işlem zamanları, üretim maliyeti, tamamlanma zamanı, akış tipi, çizelgeleme.

Acknowledgement

I would like to express my gratitude to Prof. Dr. Selim Aktürk and Asst. Prof. Dr. Hakan Gültekin, from whom I have learned a lot, due to their supervision and support during this research. I am grateful for their numerous ideas, suggestions and for all their encouraging words in the moments of difficulty.

I am also indebted to Asst. Prof. Dr. Alper Şen and Asst. Prof. Dr. Sinan Gürel for showing keen interest to the subject matter and accepting to read and review this thesis.

I thank my family for their constant encouragement and motivation in my pursuit for higher studies and for supporting me at every step in life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In many manufacturing systems each job undergoes a series of operations in a given processing order. If all jobs have to follow the same route, then this environment is referred to as a flowshop since the machines are assumed to be set up in series. In this study, we consider a two-machine flowshop environment in which identical jobs are processed. The machines can process at most one job at a time and a job can be processed by at most one machine at a time. For each job, three operations must be performed by the machines. The first operation can only be performed by the first machine, the second operation can only be performed by the second machine. The third operation is called the flexible operation which can be performed by either one of the machines. Different assignments of these operations yield different scheduling performances. Therefore, the problem is to decide on the assignment of these flexible operations to the machines for each job to be processed along with the processing time of each job.

The scheduling environment in this study is nonpreemptive meaning that after the processing of a job is started, it must stay on the machine until it is completed.

When the jobs to be precessed are physically large (e.g., television sets, copiers), storing large quantities in between the machines may not be possible. For such production systems the buffer space in between the two machines assumed to have a limited capacity. This limited storage capacity may cause

blocking of the former machine since it cannot release a job into the buffer after completing its processing when the buffer becomes full. However, when the jobs are physically small (e.g., printed circuit boards, integrated circuits), it is relatively easy to store large quantities between machines. In such cases the buffer capacities in between machines may be assumed unlimited. In this study, we will assume unlimited buffer capacity between machines.

In most of the deterministic scheduling problems in the literature, job processing times are considered as constant parameters. However, various real-life systems allow us to control the processing times by allocating extra resources, such as energy, money, or additional manpower. When job processing times are a function of the amount of resource allocated to an operation, the amount of each resource dedicated to each machine in the system becomes a decision variable that varies over time. Under controllable processing times setting, the processing times of the jobs are not fixed in advance but chosen from a given interval. There exists upper and lower bounds for the processing time of each job. The processes on the CNC machines are well known examples of how the processing times can be controlled. By adjusting the speed, feed rate, and/or the depth of cut, the processing times on these machines can be controlled easily. Although reducing the processing times may lead to an increase in throughput rate, it incurs extra costs. Controllable processing times may also provide additional flexibility in finding solutions to the scheduling problem, which in turn can improve the overall performance of the production system. Therefore, in such systems we need to consider the trade-off between job scheduling and resource allocation decisions carefully to achieve the best scheduling performance. In this study, we assume processing times to be controllable and we will show the effectiveness of using controllable processing times.

Most of the studies with controllable processing times in scheduling literature assume that the processing time is a bounded linear function of the amount of resource allocated to the processing of the job. The manufacturing cost is also assumed to increase linearly with decreasing processing time. However, a linear resource consumption function fails to reflect the law that productivity increases at a decreasing rate with the amount of resource allocated. A better approach

to this problem is to assume that the job processing time is a convex decreasing function of the amount of resource allocated to the processing of the job. We will use a convex cost function to reflect the cost of resource consumption.

In some scheduling environments, resources are fundamentally flexible, or they can be made flexible, which means the resources can be dynamically reallocated in a production process. When job processing times are a function of the amount of resource dedicated to an operation, resource flexibility can enhance system effectiveness and efficiency. Labor is a common example to flexibility. Labor flexibility can be achieved by cross-training workers. Cross-trained workers develop the skills required to perform different tasks associated with multiple processing centers. A line of automated CNC machines is also highly flexible manufacturing systems. CNC machines can perform different operations as long as the cutting tools required for these operations are loaded in the tool magazine of the machine. However, because of the limited capacity of tool magazines it may not be possible to accommodate all the tools required to process one job. Additionally, because of the high costs of the tools, having multiple copies of tools may not be economically justifiable. Therefore, some of the tools are loaded on a single machine and the corresponding operations can only be performed by that machine, or some other tools existing in multiple copies are loaded on several machines and the corresponding operations can be performed by any of these machines.

In real life applications, scheduling problems usually involve optimization of more than one criterion. An optimal solution with respect to a given criterion might be a poor solution for some other criterion. Thus considering the trade-offs involved in several different criteria problems may be necessary in real life scheduling problems. In this study, we consider a bicriteria objective which consists of minimizing the manufacturing cost (comprised of machining and tooling costs) and the makespan with the problem of assigning the flexible operations to one of the machines for each job and the processing time values for each operation. Note that, in order to minimize the makespan, the processing times must be set to their lower bounds. However, such a choice maximizes the manufacturing cost. Hence, the makespan and the manufacturing cost are two challenging objectives and they cannot be minimized at the same time. Therefore, the optimal solution

will not be unique. We will determine a set of nondominated (efficient) discrete points of makespan and manufacturing cost objectives. A solution is said to be a nondominated one if there exists no other solution with a smaller cost and a smaller makespan value. There are some approaches to solve bicriteria problems in literature. One method is to optimize both criteria simultaneously by using suitable weights for each criterion. Another method, called the ϵ-constraint approach, optimizes one criterion subject to the other criterion represented as a constraint and upper bounded by ϵ. By using different ϵ values, different points on the efficient frontier can be generated. We will use this approach to solve our bicriteria problem where the single criterion will be minimizing the total manufacturing cost subject to an upper limit on the makespan value.

The rest of the thesis is organized as follows. In the next chapter, we present an overview of the current literature. It covers a review of studies on flowshop scheduling, controllability of processing times, selection of machining parameters and flexibility in manufacturing systems. In Chapter 3, we state the problem definition and formulate the problem as a nonlinear mixed integer problem to determine a set of efficient discrete points of makespan and manufacturing cost objectives. In Chapter 4, we demonstrate some basic properties for the problem which will be used in the development of the approximation algorithm and in Chapter 5 the approximation algorithm will be presented. We perform a computational study in Chapter 6 to test the performance of our proposed approximation algorithm by comparing it with the mathematical formulation. Chapter 7 is devoted to concluding remarks and possible future research directions.

Chapter 2

Literature Review

As introduced in the previous chapter, we consider a two-machine flowshop environment where the machines are flexible. They have the capability of performing different operations as long as the required cutting tools are loaded on these machines. Additionally, the machining parameters can be adjusted to alter the processing times. Hence, we assume the processing times to be controllable. Then, the problem is to determine the assignment of flexible operations to the machines as well as the processing time values of each operation for each job in order to both minimize the makespan and the total manufacturing cost.

In scheduling literature, flowshop scheduling problems are one of the most well known problems and studied extensively since 1950s. After the use of flexible manufacturing systems is emerged and their use in different industries became widespread, number of studies considering the problems arising in such systems increased. The main directions of research in this area include quantifying the benefits of flexibility in manufacturing and the controllability of processing times. Additionally, the dynamic assignment of operations or the workforce from one station to other is another research direction in this area.

In the following sections, we review the relevant literature to identify the position of the current study with respect to the related ones. We place the studies into one of the following subcategories: Classical flowshop scheduling,

controllable processing times, flexibility in manufacturing and scheduling with flexible operations.

2.1 Flow-Shop Scheduling

Scheduling of jobs on a flowshop to optimize the measures such as the total completion time or the completion time of the last job on the last machine (i.e., makespan) have received considerable attention in the literature. This extensive literature can be reviewed from the recent surveys by Hejazi and Saghafianz [33] and Gupta and Stafford [29]. On the other hand, this study considers a flowshop with two machines and unlimited buffer capacity in between the machines. There are a great number of studies considering the limited buffer capacity case and the no-wait case. These studies can be reviewed from the papers by Hall and Sriskandarajah [32], Smutnicki [62], Brucker et al. [7], Wang et al. [70], and Liu et al. [46].

This area of research is started in year 1954 after the seminal paper by Johnson [37] who studied two- and three-machine flowshops. The processing time and setup time for each item for each machine was known in advance. He derived an exact polynomial algorithm to minimize the makespan in two-machine flowshop environment, which is known as Johnson's rule. The three-machine version of the problem was also discussed in the paper and an optimizing technique was offered for a restricted case.

There are many studies on two-machine n-job flowshop scheduling problems with unlimited storage. Gupta and Darrow [27] considered the two-machine flowshop scheduling problem with makespan criterion where the setup times of the jobs depended on immediately preceding jobs. They showed that the problem was NP-complete. They specified conditions for the optimality of a permutation schedule and proposed four efficient approximate algorithms to find approximate schedules for the problem. Glass et al. [20] discussed the problem of scheduling

and batching in two-machine flowshop and open-shop environment. They considered minimizing makespan by making batching and sequencing decisions and a processing order of the batches on each machine. A heuristic approach was developed to solve the problem. Allaoui et al. [3] studied the problem of jointly scheduling n available jobs and the preventive maintenance in a two-machine flowshop with the objective of minimizing the makespan. They focused on the particularity of this problem and presented some properties of the optimal solution. The problem was shown to be NP-hard and the optimal solutions under some conditions were presented. T'kindt et al. [66] proposed an ant colony optimization algorithm to solve a two-machine bicriteria flowshop scheduling problem with the objective of minimizing both the makespan and the total completion time. They compared the proposed algorithm with the existing heuristics to show its effectiveness.

Kohler and Steiglitz [41] considered exact and approximate algorithms for the n-job two-machine mean completion time flowshop problem. They demonstrated the computational effectiveness of coupling approximate methods for suboptimal solutions and branch-and-bound algorithms to generate solutions with a guaranteed accuracy. Croce et al. [13] studied the scheduling problem of minimizing total completion time in a two-machine flowshop. They discussed five known lower bounds and presented two new ones. A new dominance criterion was also proposed. They derived several versions of a branch and bound method by applying these lower bounds and also presented a heuristic procedure. Ng et al. [51] discussed a two-machine flowshop scheduling problem with deteriorating jobs (e.g. job's processing time is an increasing function of its starting time) to minimize the total completion time of the jobs. Several dominance properties, some lower bounds, and an initial upper bound by using a heuristic algorithm were derived. They were applied to a branch-and-bound algorithm developed to speed up the elimination process.

Lee [44] explained that the common assumption that machines are available all the time may not be true in real industry settings and considered the two-machine flowshop problem with availability constraints under the assumption that the unavailable time is known in advance. Cheng and Wang [10] extended the study

of Lee by proposing a heuristic for the two-machine flowshop scheduling problem with an availability constraint on the first machine. They gave worst-case error bounds of the heuristic proposed by Lee and their heuristic. Blazewicz et al. [5] studied the two-machine flowshop scheduling problem where machines were not available in given time intervals with an objective of minimizing the makespan. They analyzed constructive and local search based heuristic algorithms.

Sen et al. [57] studied the problem of minimizing total job tardiness in the two-machine flowshop environment. They developed a branch-and-bound solution procedure and a computationally efficient heuristic. Pan et al. [53] also considered a two-machine flowshop scheduling problem with the objective of minimizing total tardiness. They showed that previously developed dominance conditions and a lower bound for this problem could be improved and proposed a new dominance condition. A branch-and-bound algorithm was also developed that used the improvements and new dominance condition.

The general m-machine n-job problem with unlimited buffer case is studied extensively. This problem is known to be NP-Complete. Different mathematical programming formulations and heuristics are developed by Rajendran [55], Widmer and Hertz [74], Nawaz et al. [49], and Sarin and Lefoka [56]. Garey et al. [38] proved results about the computational complexity of flowshop and job-shop scheduling problems.

2.2 Controllable Processing Times

Study of the controllable processing times in scheduling was initialized by Vickson [77] in 1980. He pointed out that controllable processing times has been studied in the area of project management where project cost/duration curves have been widely used in critical path planning. He drew attention to the problems of least cost scheduling on a single machine in which processing times of jobs were controllable. Vickson considered total weighted completion time and the total processing cost. He assumed that the cost of performing each job is a linear

function of its processing time and the cost of the schedule is the sum of the total processing cost and the cost associated with the job completion times. Therefore, the problem was reduced to find the schedule of the jobs and their processing times while minimizing cost.

Shabtay and Steiner [59] explained in the survey of scheduling with controllable processing times that processing times may be controllable by allocating resources, such as additional money, overtime, energy, fuel, catalysts, subcontracting, or additional manpower, to the job operations. In a production facility with CNC machines, processing times may be controlled by adjusting machine parameters, such as cutting speed and/or feed rate (Akturk and Ilhan [1], Gultekin et al. [24], Kayan and Akturk [40]). In labor-intensive systems, number and training of the workers are effective parameters changing the processing times (Daniels and Mazzola [16] and Daniels et al. [17]).

Nowicki and Zdrzalka [52] extended Vickson's initial research in the area of two-machine flowshop scheduling problem. The problem was to find a job sequence and processing times on each machine minimizing the total processing cost plus the maximum completion time cost. They showed that the problem is NP-complete. Shabtay et. al. [58] studied two-machine flowshop scheduling problem with controllable job-processing times under the objective of determining the sequence of the jobs and the resource allocation for each job on both machines in order to minimize the makespan. They used equivalent load method to obtain the optimal resource allocation on a series-parallel graph and reduced the problem to a sequencing one. They also showed that this problem is equivalent to a new special case of the Traveling Salesman Problem.

Karabati and Kouvelis [39] discussed simultaneous scheduling and optimal processing time decision problem for a multi-product, deterministic flow line operated under a cyclic scheduling approach. Gultekin et al. [24] studied a two- and three-machine manufacturing cells with a material handling robot configured in a flowshop producing identical parts. They determined the robot move sequence as well as the processing times of the parts on each machine to minimize the cycle time and the manufacturing cost. Gultekin et al. [23] considered a cyclic

scheduling environment through a flowshop type setting in which identical parts were processed. The parts were processed with two identical CNC machines and transportation of the parts between the machines was performed by a robot. Both the allocations of the operations to the two machines and the processing time of an operation on a machine were assumed to be controllable. Hence the problem was to determine the allocation of the operations to the machines, the processing times of the operations on the machines, and the robot move sequence with a bicriteria objective of minimizing the cycle time and the total manufacturing cost.

Van Wassenhove and Baker [76] studied a single machine scheduling problem under the objective of minimizing the maximum completion penalty. Their problem was a bicriteria problem with time/cost trade-offs which produces an efficient frontier of the possible schedules. Daniels and Sarin [14] considered single machine scheduling problem and provided a tradeoff curve between the number of tardy jobs and the total amount of allocated resource. Zdrzalka [78] considered single machine scheduling problem in which each job has a release date, a delivery time and a controllable processing time, having its own associated linearly varying cost. He gave an approximation algorithm for minimizing the overall schedule cost and associated worst-case performance analysis. Panwalkar and Rajagopalan [54] studied the common due date assignment and single machine scheduling problem with an objective of minimizing a cost function containing earliness cost, tardiness cost, and total processing cost. Cheng et al. [9] studied a due date assignment and single machine scheduling in which the jobs have compressible processing times and the objective function includes the penalties for earliness, tardiness, processing time compressions. Hoogeveen and Woeginger [35] discussed scheduling on a single machine with controllable job processing times and presented several polynomial time results for the maximum job cost criterion. They also proved NP-hardness for the total weighted job completion time criterion. Ng et al. [50] considered the single machine problem with a variable common due date under the objective of minimizing a linear combination of scheduling, due date assignment and resource consumption costs. Job processing times are assumed non-increasing linear functions of an equal amount of a resource

allocated to the jobs. Wang and Xia [71] modeled a single machine scheduling problem in which job processing times are controllable variables with linear costs as an assignment problem and concentrated on two goals, namely, 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. Akturk and Ilhan [1] considered a single CNC machine scheduling to minimize the sum of total weighted tardiness, tooling and machining costs. They formulated a nonlinear mixed integer program and proposed an heuristic to solve the problem for a given sequence and designed a local search algorithm that uses it as a base heuristic.

Wan et al. [69] considered two-agent scheduling problems where agents share either a single machine or two identical machines in parallel. The processing times of the jobs of one agent are compressible; and total completion time plus compression cost, the maximum tardiness plus compression cost, the maximum lateness plus compression cost and the total compression cost are considered as objective functions for this agent. Mastrolilli [47] studied the problem of minimizing maximum flow time subject to processing cost constraint on identical parallel machines with release dates given for each job. Jansen and Mastrolilli [36] discussed controllable processing times and gave polynomial time approximation schemes for the problems of minimizing sum of processing cost and makespan, minimizing processing cost subject to makespan constraint and minimizing makespan subject to processing cost constraint for the identical parallel machines case.

Alidaee and Ahmadian [2] studied the problem of scheduling n single-operation jobs on m non-identical machines. Processing times were assumed to be controllable and the cost of performing a job was a linear function of its processing time. They considered two different scheduling cost objectives to be minimized, which were the total processing cost plus total flow time and the total processing cost plus total weighted earliness and weighted tardiness. They proposed a polynomial time algorithm to solve the problem. Another paper that deals with the tradeoff between operating costs and scheduling objectives on non-identical parallel machines is by Trick [68]. He considered controllable processing

times where each job has a linear processing cost function. Gurel and Akturk [30] dealt with making optimal machine-job assignments and processing time decisions on non-identical parallel machines to minimize total manufacturing cost, which is a nonlinear cost function, while the makespan being upper bounded by a known value. Optimality properties and methods to compute cost lower bounds for partial schedules, which are used in developing an exact branch and bound algorithm, were given. Furthermore, a recovering beam search algorithm equipped with an improvement search procedure was presented for the cases where the exact algorithm is not efficient in terms of computation time. Gurel et al. [31] showed that an anticipative approach can be used to form an initial schedule to utilize resources more effectively in a non-identical parallel machining environment if the processing times are controllable. Leyvand et al. [45] studied scheduling problems with controllable processing times on parallel machines. The objective function is bicreterion which maximizes the weighted number of jobs that are completed exactly at their due date and minimizes the total resource allocation cost. Four different models are presented for treating the bicriteria problem.

A well known example of controlling processing times is the turning operation on CNC turning machines. Processing time of an operation can be controlled on a CNC turning machine by setting the machining parameters. Trade-offs between cutting parameters and manufacturing cost for the turning operation problem has been widely studied in the literature. Hitomi [34] studied mathematical models and solution methods for different objectives of machine parameter selection problem. Lamond and Sodhi [42] considered the cutting speed selection and tool loading decisions on a single cutting machine so as to minimize total processing time. Sodhi et al. [64] studied determining the optimal processing speeds, tool loading and part allocations on several flexible machines with finite capacity tool magazines under the objective of minimizing the makespan. Kayan and Akturk [40] provided a mechanism to determine upper and lower bounds for the processing time of a turning operation. They also showed that manufacturing cost of a turning operation can be expressed as a nonlinear function of its processing time.

2.3 Flexibility in Manufacturing

When job processing times are assumed to be controllable and to be a function of the amount of resources allocated to an operation, scheduling performance may be affected by the resource flexibility. Project management is one of the subjects that controllability of processing times through resource allocation has appeared in the literature ([48], [75], [72], [60], [73], [61], and [43]).

Daniels [15] presented two extensions to the joint sequencing/resource allocation scheduling model for single-stage production initially proposed by Van Wassenhove and Baker [76]. He disscussed impact of specified limits on individual job tardiness on optimal sequencing and single resource allocation. Daniels also considered the existence of multiple resources available for processing time control.

Dobson and Karmarkar [18] studied a simultaneous sequencing and resource allocation problem that processing times and multiple resource requirements were specified for each job. They gave two formulations for the problem. They developed a Lagrangian relaxation and a surrogate relaxation which provide lower bounds on the optimal solution. An enumeration procedure to determine the optimal solution was also described. They concluded that the Lagrangian relaxation performed better on problems with a low degree of simultaneity and the surrogate relaxation did better with a high degree of simultaneity.

In 1994, Daniels and Mazzola [16] considered a flowshop setting when resources have complete flexibility, which means the resources can be allocated to any stage of the production process. They considered labor flexibility and cross-trained workers to perform all of the required operations. They formulated the flexible-resource scheduling problem with the objective of determining the permutation job sequence, resource allocation policy, and operation start times to optimize system performance. They discussed problem complexity, established lower bounds for optimal schedules, developed optimal and heuristic solution approaches. They concluded that the performance improvements associated with flexible-resource scheduling are significant according to computational results. In

2004, Daniels et al. [17] discussed partial resource (labor) flexibility, where each worker can only perform a subset of the required operations. The workers were cross-trained to perform a subset of the tasks occurring on the assembly line. They provided a mathematical formulation for flowshop scheduling with partial resource flexibility. They suggested that a relatively small investment in cross-training provided a large portion of the available benefit associated with labor flexibility.

Assembly line balancing problems were surveyed by Becker and Scholl [4] and Boysen et al. [6]. Line balancing models assign the tasks to workstations with an objective of minimizing the cycle time or minimizing the number of stations needed to achieve a predetermined cycle time.

Another example to flexibility in manufacturing is the automated CNC machines which can perform different operations. Crama et al. [11] surveyed cyclic scheduling problems in robotic flowshops, models and complexity of these problems. They claimed that the most basic and important problems are well understood, at least from a complexity viewpoint.

Sodhi et al. [63] showed that the decision of tool loading to maximize routing flexibility is shown to be a minimum cost network flow problem when routing flexibility is a function of the average workload per tool aggregated over tool types, or of the number of possible routes through the system. They developed a linear programming model to plan a set of routes for each part type with an objective of either minimizing either the material handling requirement or the maximum workload on any machine. They investigated the impact of these tool addition strategies on the material handling and workload equalization and presented computational results. They concluded that the overall approach was favorable in terms of computational simplicity at each step and the ability to react to dynamic changes.

Stecke [65] defined a set of five production planning problems that must be solved for efficient use of a flexible manufacturing system, and addressed the machine grouping and tool loading problems. They formulated nonlinear 0-1 mixed integer programs for the problems. They examined several linearization methods

and reduced the constraint size of the linearized integer problems according to various methods to decrease computational time.

2.4 Scheduling with Flexible Operations

There are several researches on decision rules for the assignment of the flexible operations. Gupta et al. [28] studied two-machine flowshop processing nonidentical jobs that the buffer has infinite capacity. Each job had three operations, one of which was a flexible operation. They analyzed two algorithms. One algorithm used arbitrary assignment of the flexible operations and an arbitrary processing order, and the other, improved algorithm, constructed four schedules and selected the best which were polynomial time approximation schemes.

Burdett and Kozan [8] considered a scheduling environment that station tasks or operations could be shifted or redistributed to adjacent stations. In this case, the stations should have appropriate equipment and the workers should be cross-trained. Mathematical models, recurrence equations and solution techniques were provided for the intermediate storage, no-intermediate storage and no-wait flowshop problem cases.

Gultekin et al. [21] studied the problem of two-machine robotic cell scheduling with operational flexibility. They assumed that the parts to be processed were identical and had a number of operations to be completed. The machines were also assumed to be capable of performing all of the operations. They determined the optimal robot move cycle and the corresponding optimal allocation of operations with an objective of minimizing the cycle time. Gultekin et al. [22] dealt with the robotic cell scheduling problem with two machines and identical parts. They discussed that the assumption of CNC machines being capable of performing all the required operations as long as the required tools are stored in their tool magazines may be unrealistic because of the limited tool magazines capacity. Hence, they assumed that some operations could only be processed on the first

machine while some others on the second machine due to tooling constraints. Allocation of the remaining operations (which can be processed on either machine) to the machines and the optimal robot move cycle that minimized the cycle time were determined. A sensitivity analysis on the results was also conducted.

Guo et al. [25] considered a scheduling problem in the flexible assembly line with the objectives of balancing the production flow, which considers assignment of flexible operations, and minimizing the weighted sum of tardiness and earliness penalties. They presented a mathematical model for the problem. A bi-level genetic algorithm, a heuristic initialization process and modified genetic operators were proposed. Guo et al. investigated a production control problem on a flexible assembly line with flexible operation assignment and variable operative efficiencies. They formulated a mathematical model. They also developed an intelligent production control decision support system, a production control decision support model comprising a bi-level genetic optimization process and a heuristic operation routing rule.

Crama and Gultekin [12] considered a production line consisting of two machines operating as a flowshop and a buffer located between the machines. The jobs processed were identical and each job had three operations, one of which was a flexible operation. The assignment of the flexible operations to the machines for each job was determined under the objective of maximizing the throughput rate. Several cases were considered regarding the number of parts to be produced and the capacity of the buffer.

2.5 Summary

In this chapter, classical flowshop scheduling, controllable processing times, flexibility in manufacturing and scheduling with flexible operations literatures were reviewed. The study by Crama and Gultekin [12] considering processing of identical jobs and assignment of flexible operations is the most relevant one to our study. Gupta et al. [28] also study two-machine flowshop with assignment of

flexible operations but the jobs are assumed to be nonidentical. However, both studies consider fixed processing times and a single objective function criterion.

Current literature ignores a bicriteria problem that considers the allocation of flexible operations and controllability of processing times at the same time. A bicriteria model provides useful insights to the decision maker. For example, a solution which minimizes the makespan can be a worse solution in terms of the manufacturing cost. This study is much more realistic than the existing ones and can present better results, so this thesis considers the deficiencies of the current literature. We study a bicriteria problem in which the processing times of each operation and the assignments of flexible operations to one of the machines for each job should be determined optimally. The objective is to minimize the manufacturing cost and the makespan value.

In the next chapter, we will present the definition and modeling of the problem.

Chapter 3

Problem Definition and Modeling

In this chapter, we present the definition of the problem and two mathematical formulations. We discuss some characteristics of the problem and present numerical examples.

3.1 Problem Definition

In this study, we have n identical jobs to be processed on two machines in a flowshop environment. For each job, three operations must be performed by the machines. The first (second) operation can only be performed by the first (second) machine and its processing time is equal to f_j^1 (f_j^2) for job j. The third operation is flexible so can be performed by either one of the machines and the processing time of this operation is s_j for job j. The assignment of flexible operations to the machines for each job is a decision that should be made. All jobs are first processed by the first machine and then by the second machine. Preemption is not allowed. Different than most of the studies in the machine scheduling literature, we assume the processing times to be controllable within an upper and a lower bound. As a consequence of the controllability of the processing times and the dynamic assignment of the flexible operations from one part to the other, although the jobs are assumed to be identical they may have

different processing times on the machines. Hence, they are identical in the sense that, they all require the same set of operations. Because of this reasoning, the job index, j, denotes the job in the j^{th} position. Our objective is to determine the processing times of operations for each job and the assignment of flexible operations of each job to one of the machines under the bicriteria objective of minimizing the manufacturing cost and the makespan.

3.2 Mathematical Model Formulation

In this study, we assume the machines to be CNC machines. For these machines, the manufacturing cost of an operation can be expressed as the sum of the operating and the tooling costs. Operating cost of a machine is the cost of running this machine. Tooling cost for CNC operation can be calculated by the cost of the tools used times tool usage rate of the operation. Kayan and Akturk [40] showed that manufacturing cost of a CNC operation can be expressed as a function of processing time. Although we consider the manufacturing cost incurred for a CNC machine, our analysis is valid for any convex cost function.

The notation used throughout the thesis is as follows:

Decision variables

f_j^1	processing time for the 1^{st} operation of job j on machine 1
f_j^2	processing time for the 2^{nd} operation of job j on machine 2
s_j	processing time for flexible operation of job j on the assigned machine
x_j	decision variable, that controls if flexible operation of job j is assigned to machine 1
$T_{j,m}$	starting time of the j^{th} job on machine m

Parameters

n	number of jobs to be processed
C	operating cost of machines
$F_j^1(f_j^1)$	manufacturing cost function of processing time for the 1^{st} operation of job j on machine 1
$F_j^2(f_j^2)$	manufacturing cost function of processing time for the 2^{nd} operation of job j on machine 2
$S_j(s_j)$	manufacturing cost function of processing time for flexible operation of job j on the assigned machine
f_l^1, f_u^1	processing time lower and upper bounds for the 1^{st} operation on machine 1, respectively
f_l^2, f_u^2	processing time lower and upper bounds for the 2^{nd} operation on machine 2, respectively
s_l, s_u	processing time lower and upper bounds for flexible operation, respectively
b	tooling cost exponent, (note that, $b < 0$)
A^1, A^2, A^s	tooling cost multipliers for the 1^{st}, 2^{nd}, and flexible operations, respectively.

In this study, the following nonlinear form of the manufacturing cost function will be used.

For the first and second operations:

$$F_j^i(f_j^i) = C \cdot f_j^i + A^i \cdot (f_j^i)^b \quad \text{for} \quad i = 1, 2 \quad \text{and} \quad j = 1, ..., n.$$

For the flexible operation:

$$S_j(s_j) = C \cdot s_j + A^s \cdot (s_j)^b \quad \text{for} \quad j = 1, ..., n.$$

Note that these cost functions are strictly convex and have unique minimizers. Kayan and Akturk [40] stated that a value of processing time greater than the minimizer of the cost function is inferior both in terms of the manufacturing cost and a regular scheduling performance measure. Therefore, the optimal processing time value will never exceed the minimizers of these functions. As a consequence, these values can be named as upper bounds of processing times and denoted by f_u^i and s_u. They showed that $F_j^i(f_j^i)$ and $S_j(s_j)$ are minimized at processing time levels f_u^i and s_u, respectively. They proposed that since the cost functions are convex, the values of f_u^i and s_u can be determined by taking derivatives of the objective function with respect to f_j^i and s_j and solving them by equating to zero. Then, we have $f_u^1 = \sqrt[b-1]{\frac{-C}{A^1 \cdot b}}$, $f_u^2 = \sqrt[b-1]{\frac{-C}{A^2 \cdot b}}$ and $s_u = \sqrt[b-1]{\frac{-C}{A^s \cdot b}}$ as upper bounds for f_j^1, f_j^2, and s_j. Moreover, there exists a processing time lower bound f_l^i and s_l which is determined by the manufacturing properties of job j and the maximum applicable power of CNC machine.

We can formulate the bicriteria problem as a nonlinear mixed integer program as follows:

$$\text{Min.} \quad Z_1 = (\sum_{j=1}^{n}\sum_{i=1}^{2} f_j^i + \sum_{j=1}^{n} s_j) \cdot C + \sum_{j=1}^{n}\sum_{i=1}^{2}(f_j^i)^b \cdot A^i + \sum_{j=1}^{n}(s_j)^b \cdot A^s \qquad (3.1)$$

$$\text{Min.} \quad Z_2 = T_{n,2} + f_n^2 + s_n \cdot (1 - x_n) \qquad (3.2)$$

$$\text{s.t.} \quad T_{j,1} \geq T_{j-1,1} + f_{j-1}^1 + s_{j-1} \cdot x_{j-1} \qquad j \geq 2 \qquad (3.3)$$

$$T_{j,2} \geq T_{j-1,2} + f_{j-1}^2 + s_{j-1} \cdot (1 - x_{j-1}) \qquad j \geq 2 \qquad (3.4)$$

$$T_{j,2} \geq T_{j,1} + f_j^1 + s_j \cdot x_j \qquad \forall j \qquad (3.5)$$

$$T_{1,1} \geq 0 \qquad (3.6)$$

$$f_l^i \leq f_j^i \leq f_u^i \qquad \forall i \text{ and } \forall j \qquad (3.7)$$

$$s_l \leq s_j \leq s_u \qquad \forall j \qquad (3.8)$$

$$x_j \in \{0, 1\} \qquad \forall j \qquad (3.9)$$

Equation (3.1) represents one of the objective functions which minimizes total manufacturing cost. Equation (3.2) represents the second objective function which minimizes makespan. Constraints (3.3) and (3.4) express the condition that the j^{th} job can start on the first (resp., second) machine only after the previous job is completed on this machine. Constraint (3.5) also explains the condition that the processing of a job on the second machine can be started only after the processing of this job is completed on the first machine. Constraint (3.6) is the nonnegativity constraint of the variable $T_{1,1}$. Upper and lower bounds of the processing time of the first operation, second operation, and flexible operation are represented by the constraints (3.7) and (3.8), respectively. Finally, constraint (3.9) expresses the assignment of flexible operation to only one of the two machines for each job.

One of the methods used for bicriteria problems in the literature is the so called ϵ-constraint approach discussed by T′kindt and Billaut [67]. This method represents one of the objectives as a constraint with an auxilary upper bound and optimizes over the second objective. By searching over different values for the upper bound one can generate a set of discrete nondominated points; points on the efficient frontier. We will use ϵ-constraint approach to solve our bicriteria problem. For this purpose, we will represent makespan objective as a constraint and optimize over the manufacturing cost objective. Therefore, our problem turns out to be minimizing total manufacturing cost objective for a given upper limit ϵ on the makespan objective and let us name this mathematical programming formulation as Model 1.

$$\begin{aligned} \textbf{Model 1:}\ \text{Min.}\quad & Z_1 \\ \text{s.t.}\quad & Z_2 \leq \epsilon \\ & \text{Constraints } (3.3) - (3.9) \end{aligned} \tag{3.10}$$

The constraints (3.3), (3.4), (3.5), and (3.10) of Model 1 are nonlinear because of multiplication of two variables. Using auxiliary variables, these can be linearized. As we mentioned before, manufacturing cost function consists of operating cost and tooling cost functions. Since tooling cost function is nonlinear

because of the exponent term, it cannot be linearized. Hence, the objective function remains nonlinear. After replacing $s_j \cdot x_j$ with q_j^1 and $s_j \cdot (1 - x_j)$ with q_j^2, constraints (3.3), (3.4), (3.5), and (3.10) are linearized and we have the following model with nonlinear objective function but linear constraints. The formulation with linear constraints named as Model 2 is as follows:

Model 2: Min. Z_1

$$\text{s.t.} \quad T_{n,2} + f_n^2 + q_n^2 \leq \epsilon \tag{3.11}$$

$$T_{j,1} \geq T_{j-1,1} + f_{j-1}^1 + q_{j-1}^1 \qquad j \geq 2 \tag{3.12}$$

$$T_{j,2} \geq T_{j-1,2} + f_{j-1}^2 + q_{j-1}^2 \qquad j \geq 2 \tag{3.13}$$

$$T_{j,2} \geq T_{j,1} + f_j^1 + q_j^1 \qquad \forall j \tag{3.14}$$

$$T_{1,1} \geq 0 \tag{3.15}$$

$$s_j - M \cdot (1 - x_j) \leq q_j^1 \leq s_j + M \cdot (1 - x_j) \qquad \forall j \tag{3.16}$$

$$-M \cdot x_j \leq q_j^1 \leq M \cdot x_j \qquad \forall j \tag{3.17}$$

$$q_j^2 = s_j - q_j^1 \qquad \forall j \tag{3.18}$$

$$f_l^i \leq f_j^i \leq f_u^i \qquad \forall i \text{ and } \forall j \tag{3.19}$$

$$s_l \leq s_j \leq s_u \qquad \forall j \tag{3.20}$$

$$x_j \in \{0, 1\} \qquad \forall j \tag{3.21}$$

In the above formulation, q_j^m is an artificial variable, where m and j are the machine index and job index, respectively. This variable represents the processing time for flexible operation of job j on machine m. Moreover, M represents a large number associated with this artificial variable.

As mentioned in Section 2.4, Gupta et al. [28] studied the problem of minimizing makespan in a two-machine flowshop environment with flexible operations. They claimed that when all operations, except the flexible ones, had zero processing times, then each job had just one operation that should be processed on either one of the machines. Thus, the problem turned out to be scheduling jobs on two identical parallel machines to minimize the makespan. This problem was known to be NP-hard in the ordinary sense, so they claimed that their problem was

also NP-hard at least in the ordinary sense. Although we have identical parts requiring the same set of operations, similar to Gupta et al. [28], these parts may have different processing time values. Additionally, the objective function is a nonlinear one. As a consequence, we conjecture that our problem could be NP-Hard as well.

3.3 Characteristics of the Problem

Crama and Gultekin [12] showed that the optimal schedule in fixed processing times case can actually be found more efficiently than a mathematical programming formulation. They proved that the optimal assignment of the flexible operation for each job can be found by the following formula:

$$r = \frac{(n-1)\cdot(f^2+s-f^1)+s}{2\cdot s} \tag{3.22}$$

Here, r represents the total number of jobs for which the flexible operation is assigned to the first machine. f^1, f^2, and s are the processing times of first operation, second operation, and flexible operation, respectively. Then using the same idea as with the Johnson's algorithm, the first $n-r$ flexible operations should be assigned to the second machine and the remaining ones to the first machine.

By definition, r must be an integer but this formula may yield noninteger values. If the resulting value is non-integer, then the optimal makespan can be found using either the largest integer smaller than r, $\lfloor r \rfloor$, or the smallest integer larger than r, $\lceil r \rceil$, and the following formula.

$$C_{max} = min\{(f^1 + n\cdot f^2 + (n-\lfloor r \rfloor)\cdot s), (n\cdot f^1 + f^2 + \lceil r \rceil \cdot s)\} \tag{3.23}$$

Figure 3.1 represents an efficient frontier of makespan and total manufacturing cost objectives. In this figure, Z_1 denotes the manufacturing cost and Z_2 denotes

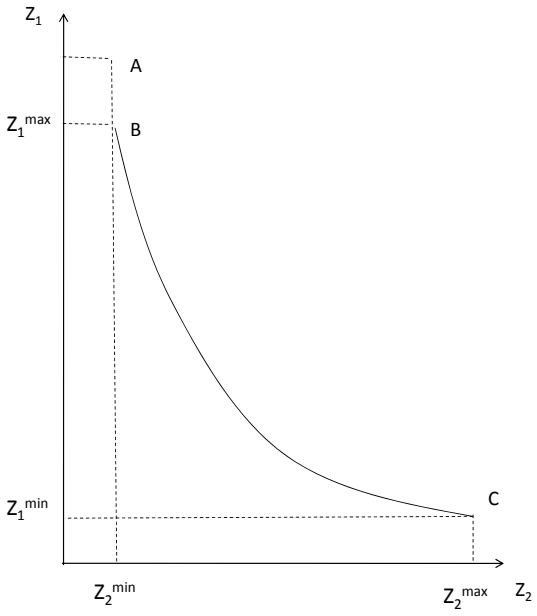


Figure 3.1: Efficient frontier of makespan and total manufacturing cost objectives

the makespan value. [67] discussed that a point (Z_1^b,Z_2^b) is said to be efficient with respect to cost and makespan criteria if there does not exist another point (Z_1^d,Z_2^d) such that $Z_1^d \leq Z_1^b$ and $Z_2^d \leq Z_2^b$ with at least one holding as a strict inequality. In literature, the processing time upper bounds are mostly preferred in terms of manufacturing cost objective because processing the jobs at their upper bounds allows to achieve minimum manufacturing cost. At point C on Figure 3.1, Z_1 is at the minimum value (Z_1^{min}) while Z_2 is at the maximum value (Z_2^{max}). This point is reached when the processing times are set to their upper bounds.

On the other hand, we would prefer using processing time lower bounds for any regular scheduling measure. When the processing times are set to their lower bounds, we get the point, A, on the Figure 3.1. However, for this particular case r value may not be an integer, so the machines may sometimes become idle which is unavoidable in fixed processing time. This may incur extra cost. However controllability allows us to prevent idleness by increasing some of the the processing times of jobs and changing the assignment of the flexible operations, which yields a reduction in manufacturing cost without increasing the makespan. Hence, a schedule can be obtained with the same makespan value but at a lower cost. This idea will be highlighted via examples in the next section. The old point

A becomes a dominated point in this case. The new point is a nondominated point, which is represented by B in Figure 3.1. This means the optimal solution with fixed processing times is dominated by a nondominated optimal solution with controllable processing times. At point B, Z_1 has its maximum value (Z_1^{max}) and Z_2 has its minimum (Z_2^{min}).

3.4 Numerical Examples

In this section, we present two examples to demonstrate the benefits of controllable processing times over fixed processing times case.

Example 1 Let us consider two cases, one of which assumes fixed processing times and the other has controllable processing times.

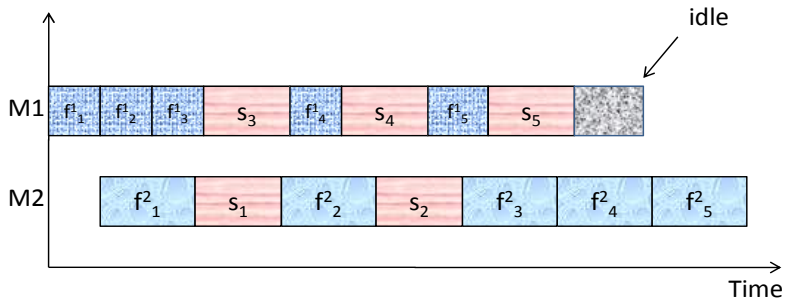
Case 1: Fixed processing time case

In this problem, number of jobs is selected to be 5. Let the processing times be $f_j^1 = 1.2$, $f_j^2 = 2$, and $s_j = 1.8\ \forall j$. Let the operation cost of machines, tooling cost multiplier, and tooling cost exponent be $C = 4$, $A = 8$, and $b = -2$, respectively. Using the solution procedure developed by [12], the optimal makespan value is found to be 14.8 time units. The Gantt chart of the optimal solution is depicted in Figure 3.2a, in which the flexible operations of jobs 3, 4, and 5 are assigned to the first and the remaining ones are assigned to the second machine. With given parameters, the associated cost of this solution is 62.62.

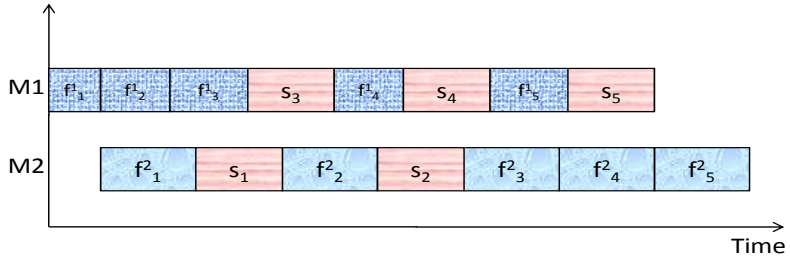
Case 2: Controllable processing time case

Let the number of jobs to be the same as in Case 1 and the processing time ranges be $1.2 \leq f_j^1 \leq 4.7$, $2 \leq f_j^2 \leq 2.8$, and $1.8 \leq s_j \leq 5.6\ \forall j$. Lower bound processing time values, tooling cost multiplier and tooling cost exponent are selected to be the same as in Case 1. Let the upper limit of makespan in Constraint 3.10 be set to same as Case 1 (14.8).

The problem is solved using BARON MINLP solver of GAMS and because



(a) Case 1



(b) Case 2

Figure 3.2: Optimal schedule of Example 1

of the convex nature of the problem the solver guarantees an optimal solution. The solution is found to be $f_1^1 = 1.2$, $f_j^1 = 1.55$ for j=2 to 5, $f_j^2 = 2\ \forall j$, and $s_j = 1.8\ \forall j$. In Figure 3.2b, the optimal solution of the problem with cost 54.42 and makespan 14.8 time units is depicted.

As can be seen from Figures 3.2a and 3.2b, while there is an idle time in the schedule of Case 1, there is no idle time on the machines just after they start processing jobs until they complete the processing of the last job in the schedule of Case 2. By means of controllability of processing times, the durations of f_j^1 for jobs j=2 to 5 are increased, which in this case decreased the manufacturing cost by 15.1%. Therefore, a new schedule is obtained with the same makespan at a lower cost and the optimal solution with fixed processing times is dominated by a nondominated optimal solution with controllable processing times.

Controllability of the processing times is not the only factor affecting the idle times (manufacturing cost) on the machines. The assignment of flexible operations also affect the manufacturing cost. The following example is given to highlight this idea.

Example 2 Let us consider another two cases, one of which again assumes fixed processing times and the other has controllable processing times.

Case 1: Fixed processing time case

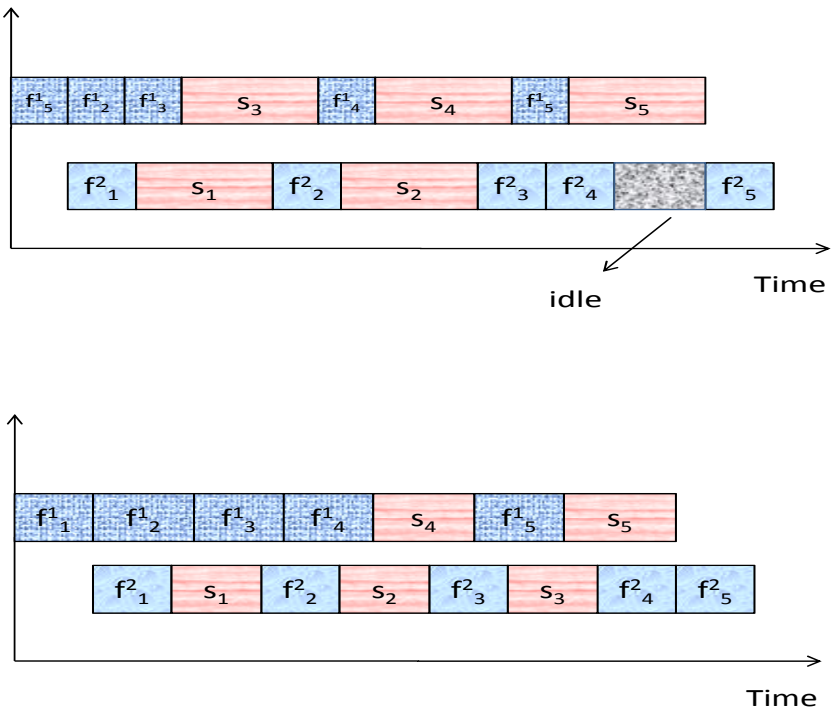
Let the number of jobs be the same as in Example 1 and the processing times be $f_j^1 = 1.8$, $f_j^2 = 2.4$, and $s_j = 4.5\ \forall j$. Let tooling cost multiplier and tooling cost exponent also be the same as in Example 1. In Figure 3.3a, the optimal solution of the problem with makespan 24.9 time units is depicted. With given parameters, the corresponding cost of this solution is 43.01.

Case 2: Controllable processing time case

Let the processing time values, tooling cost multiplier, and tooling cost exponent be the same as in Case 2 of Example 1. Let the upper limit of makespan in Constraint 3.10 be set to same as Case 1 (24.9).

The solution is found to be $f_1^1 = 2.77$, $f_j^1 = 3.17$ for j=2 to 5, $f_j^2 = 2.77$ for jobs j=1 to 5, $s_j = 2.77$ for j=1,2,3, and $s_j = 3.17$ for jobs j=4,5. In Figure 3.3b, the optimal solution of the problem with cost 36.14 and makespan 24.9 time units is depicted.

As can be seen from the Figures 3.3a and 3.3b, while there is an idle time in the schedule of Case 1 there is no idle time in the schedule of Case 2. In the optimal schedule of Case 1, the flexible operations of jobs 3, 4, and 5 are assigned to the first machine and the other two to the second machine. However, in the optimal schedule of Case 2, the flexible operations of jobs 4 and 5 are processed on the first machine and the other three on the second machine. The processing times of operations are also different than the fixed processing time case. While f_j^i for i=1,2 and j=1 to 5 are increased, s_j for j= 1 to 5, are decreased. This is



(b) Case 2

Figure 3.3: Optimal schedule of Example 2

by means of controllability of processing times which in this case decreased the manufacturing cost by 19.0%. Therefore, a new schedule is obtained with the same makespan at a lower cost.

As highlighted in the examples, the solution procedures proposed by [12] and [28], which present optimal solutions for fixed processing time, may not be optimal for controllable processing time. Also, although the jobs are assumed to be identical in this study, the flexible operations, s_j, can have different values at different machines. Therefore, we need to develop a new algorithm for the problem.

3.5 Summary

In this chapter, we presented the definition of the problem. We built two mathematical formulations to determine the optimal processing time values and the assignment of flexible operations giving the minimum manufacturing cost. We discussed some properties of the problem and presented two numerical examples.

In the next chapter, we will discuss some basic properties of the problem and characteristics of the approximation algorithm that will be presented in Chapter 5.

Chapter 4

Theoretical Results

In this chapter, we demonstrate some optimally properties for the problem which will be used in the development of the approximation algorithm and characteristics of the approximation algorithm that will be presented in the next chapter.

4.1 Optimality Properties of the Problem

The first lemma considers the property of the second objective function represented as a constraint in Model 1 and Model 2.

Lemma 1 *In an optimal solution to the problem, either constraint (3.10) is satisfied as equality or the processing times for all parts are set to their upper bounds.*

Proof. Let $Z_1^\star = \sum_{j=1}^{n} \sum_{i=1}^{2} (F_j^i(f_j^{i\star}) + S_j(s_j^\star))$ be the optimal objective function value with optimal processing time vectors $f^\star$ and $s^\star$. Assume to the contrary that $C_{max} = T_{n,2} + f_n^2 + s_n \cdot (1 - x_n) < \epsilon$ and there exists $\hat{j}$ such that $f_{\hat{j}}^i < f_u^i$. Consider another solution with $\hat{f}_j^{i\star} = f_j^{i\star}$, $\forall j \neq \hat{j}$. Let $\hat{f}_{\hat{j}}^{i\star} = f_{\hat{j}}^{i\star} + \beta$ for some $\beta,\ 0 < \beta \leq min\{f_u^i - f_{\hat{j}}^{i\star}, \epsilon - (T_{n,2} + f_n^2 + s_n \cdot (1 - x_n))\}$. Note that, if all processing times are on their upper bounds, there is no such β. Processing times for all operations except $\hat{j}$ is identical with the previous solution and note that

$\hat{f}^{i^\star}_{\hat{j}} > f^{i^\star}_{\hat{j}}$. The objective function value of the new solution, $\hat{Z}^\star_1$, satisfies $\hat{Z}^\star_1 < Z^\star_1$, because the cost function is decreasing with respect to processing times. However, this contradicts with $f^\star$ being the optimal solution. □

As a consequence of the above lemma, we know that makespan value (C_{max}) is equal to the upper bound ϵ in an optimal solution which will be used in the proof of the Lemma 3.

Lemma 2 is about machine idle times and helpful in characterizing the optimal solution. In any schedule, both of the machines are initially idle. Then, while the first job is being processed on the first machine, the second machine is still idle waiting for the job during the interval $[0, T_{1,2}]$. This idle time on the second machine cannot be avoided and its length is at least equal to f^1_1. Similarly, some idle time of length at least f^2_n cannot be avoided on the first machine while the second machine processes the last job. The idle times except these are denoted as unforced idle times. If there is an unforced idle time on machine 1, it could only happen after the last job on the first machine has been completed. Unlike machine 1, there may be unforced idle times at machine 2 during the time interval $[T_{1,2}, T_{n,2}]$ because of precedence constraint (3.5). As we mentioned in the numerical examples, these unforced idle times are not desirable and elimination of them decreases manufacturing cost. In the following lemma, $C_{j,m}$ represents the completion time of the j^{th} job on machine m.

Lemma 2 *In an optimal solution to the problem, the following conditions hold.*

1. *Either* $C_{n-1,2} \leq C_{n,1}$ *or* $f^1_j = f^1_u \quad \forall j = 1, 2, ..., n$
2. *Either* $C_{k,1} \leq C_{k-1,2}$ *or* $f^2_j = f^2_u \quad \forall k = 1, 2, ..., n$ *and* $\forall j = 1, 2, ..., k-1$

Proof. Let $Z^\star_1 = \sum_{j=1}^{n} \sum_{i=1}^{2} (F^i_j(f^{i^\star}_j) + S_j(s^\star_j))$ be the optimal objective function value with optimal processing time vectors $f^\star$ and $s^\star$.

Assume to the contrary $C_{n-1,2} > C_{n,1}$ and there exists $\hat{j}$ such that $f^1_{\hat{j}} < f^1_u$. Hence, consider another solution with $\hat{f}^{1^\star}_j = f^{1^\star}_j \ \forall j \neq \hat{j}$ and $\hat{f}^{1^\star}_{\hat{j}} = f^{1^\star}_{\hat{j}} +$

$min\{f_u^1 - f_{\hat{j}}^{1^\star}, C_{n-1,2} - C_{n,1}\}$. This new solution has identical processing times for all operations except $\hat{j}$ and $\hat{f}_{\hat{j}}^{1^\star} > f_{\hat{j}}^{1^\star}$. Since the cost function is decreasing with respect to processing times, the objective function of the new solution, $\hat{Z}_1^\star$, satisfies $\hat{Z}_1^\star < Z_1^\star$. However, this contradicts with $f^\star$ being the optimal solution.

Similarly, assume to the contrary that there exists k such that $C_{k,1} > C_{k-1,2}$ and $\hat{j} = 1, .., k-1$ such that $f_{\hat{j}}^2 < f_u^2$. Hence, consider another solution with $\hat{f}_j^{2^\star} = f_j^{2^\star}\ \forall j \neq \hat{j}$ and $\hat{f}_{\hat{j}}^{2^\star} = f_{\hat{j}}^{2^\star} + min\{f_u^2 - f_{\hat{j}}^{2^\star}, C_{k,1} - C_{k-1,2}\}$. This new solution has identical processing times for all operations except $\hat{j}$ and $\hat{f}_{\hat{j}}^{2^\star} > f_{\hat{j}}^{2^\star}$. Since the cost function is decreasing with respect to processing times, the objective function of the new solution, $\hat{Z}_1^\star$, satisfies $\hat{Z}_1^\star < Z_1^\star$. However, this contradicts with $f^\star$ being the optimal solution. □

Lemma 2 indicates that in an optimal schedule we need to prevent the unforced idleness of machines till the processing time variables reach to their upper bounds.

We present the following lemma which will play an essential role in the development of the solution procedure. It states the relationships between processing times of jobs on the same machine in an optimal solution with respect to the derivatives of the cost functions of the jobs. Every operation of jobs are partitioned into two sets with respect to their processing time values in the optimal solution. For the 1^{st} operation, the jobs which have processing time values greater than their lower bounds appear in set J_1^1 while the jobs which have processing time values equal to their lower bounds appear in set J_2^1. The assignment of the jobs to the sets are done similarly for the 2^{nd} operation and the flexible operation. Note that the first job and the last job are excluded from the sets J_2^1 and J_2^2, respectively. As mentioned in Lemma 2, there exists no idle times on both machines unless the processing time variables reach to their upper bounds. Hence, in some cases the processing time values, except the first operation of the first job and the second operation of the last job, are increased to eliminate idle times. The first operation of the first job and the second operation of the last job cannot be increased to eliminate idle times because they directly affect the makespan value. Moreover, the jobs are partitioned into two subsets namely M^1 and M^2 depending on the assignment of their flexible operations. We represented

$\partial F_j^1(f_j^1)/\partial f_j^1$, $\partial F_j^2(f_j^2)/\partial f_j^2$, and $\partial S_j(s_j)/\partial s_j$ as $\partial F_j^1(f_j^1)$, $\partial F_j^2(f_j^2)$, and $\partial S_j(s_j)$ for simplicity.

Lemma 3 *In an optimal solution to the problem, let* $f^{1^\star}$, $f^{2^\star}$, *and* $s^\star$ *be the optimal processing times vectors of the variables* $f_j^{1^\star}$, $f_j^{2^\star}$, *and* $s_j^\star$, *respectively. Let* j *be the job index and*

$J_1^1 = \{h : f_u^1 \geq f_h^{1^\star} > f_l^1\}$ *and* $J_2^1 = \{h : h \neq 1, f_h^{1^\star} = f_l^1\}$

$J_1^2 = \{j : f_u^2 \geq f_j^{2^\star} > f_l^2\}$ *and* $J_2^2 = \{j : j \neq n, f_j^{2^\star} = f_l^2\}$

$J_1^s = \{k : s_u \geq s_k^\star > s_l\}$ *and* $J_2^s = \{k : s_k^\star = s_l\}$

$M^1 = \{$*set of jobs whose flexible operations are processed on machine* 1 $\}$

$M^2 = \{$*set of jobs whose flexible operations are processed on machine* 2 $\}$.

Then the following conditions hold:

1. $\partial F_h^1(f_h^1) \mid_{f_h^1 = f_h^{1^\star}} \leq \partial S_k(s_k) \mid_{s_k = s_k^\star}$ $\quad \forall h \in J_1^1$ *and* $\forall k \in J_2^s \cap M^1$
2. $\partial F_j^2(f_j^2) \mid_{f_j^2 = f_j^{2^\star}} \leq \partial S_k(s_k) \mid_{s_k = s_k^\star}$ $\quad \forall j \in J_1^2$ *and* $\forall k \in J_2^s \cap M^2$
3. $\partial S_k(s_k) \mid_{s_k = s_k^\star} \leq \partial F_h^1(f_h^1) \mid_{f_h^1 = f_h^{1^\star}}$ $\quad \forall k \in J_1^s \cap M^1$ *and* $\forall h \in J_2^1$
4. $\partial S_k(s_k) \mid_{s_k = s_k^\star} \leq \partial F_j^2(f_j^2) \mid_{f_j^2 = f_j^{2^\star}}$ $\quad \forall k \in J_1^s \cap M^2$ *and* $\forall j \in J_2^2$
5. $\partial F_h^1(f_h^1) \mid_{f_h^1 = f_h^{1^\star}} = \partial S_k(s_k) \mid_{s_k = s_k^\star}$ $\quad \forall h \in J_1^1$ *and* $\forall k \in J_1^s \cap M^1$
6. $\partial F_j^2(f_j^2) \mid_{f_j^2 = f_j^{2^\star}} = \partial S_k(s_k) \mid_{s_k = s_k^\star}$ $\quad \forall j \in J_1^2$ *and* $\forall k \in J_1^s \cap M^2$.

Proof. We assume that there exists at least one job such that $f_h^{1^\star} > f_l^1$, $f_j^{2^\star} > f_l^2$, and $s_k^\star > s_l$, then the optimal processing times vectors $f^{1^\star}$, $f^{2^\star}$, and $s^\star$ are regular points. In other words, the gradients of the active inequality constraints and the gradients of the equality constraints are linearly independent at that point. Such a point must satisfy the Karush Kuhn Tucker (KKT) conditions. From Lemma 1, constraint (3.10) is satisfied as equality. Moreover, since

makespan is a regular scheduling measure, increasing the completion time of any job will not improve the makespan value. Consequently, any processing time value greater than f_u^1, f_u^2 and s_u will lead to an inferior solution because the value of objective function will get worse. Therefore, we can replace the constraints $f_l^1 \leq f_h^1 \leq f_u^1$, $f_l^2 \leq f_j^2 \leq f_u^2$, and $s_l \leq s_k \leq s_u$ with $f_l^1 \leq f_h^1$, $f_l^2 \leq f_j^2$, and $s_l \leq s_k$.

1. The Lagrangian function for point $f^{1\star}$ can be written as follows:

$$L(f^{1\star}, \lambda^\star, \mu_1^\star) = \sum_{h=1}^{n} F_h^1(f_h^1) + \lambda^\star \cdot (C_{max} - \varepsilon) + \sum_{h=1}^{n} \mu_{1_h}^\star \cdot (f_l^1 - f_h^{1\star}).$$

The term $\lambda^\star$ is unrestricted in sign and $\mu_{1_h}^\star \geq 0$. If we set $\nabla(L(f^{1\star}, \lambda^\star, \mu_1^\star)) = 0$, we get $\partial F_h^1(f_h^1) \mid_{f_h^1 = f_h^{1\star}} + \lambda^\star - \mu_{1_h}^\star = 0 \; \forall h$. Since $h \in J_1^1$, then $\mu_{1_h}^\star = 0$. Thus, $\partial F_h^1(f_h^1) \mid_{f_h^1 = f_h^{1\star}} = -\lambda^\star$.

The Lagrangian function for point $s^\star$ can be written as follows:

$$L(s^\star, \lambda^\star, \mu_s^\star) = \sum_{k=1}^{n} S_k(s_k) + \lambda^\star \cdot (C_{max} - \varepsilon) + \sum_{k=1}^{n} \mu_{s_k}^\star \cdot (s_l - s_k^\star).$$

Similarly, the term $\lambda^\star$ is unrestricted in sign and $\mu_{s_k}^\star \geq 0$. If we set $\nabla(L(s^\star, \lambda^\star, \mu_s^\star)) = 0$, we get $\partial S_k(s_k) \mid_{s_k = s^{k\star}} + \lambda^\star - \mu_{s_k}^\star = 0 \; \forall k$. Since $k \in (J_2^s \cap M^1)$, then $\mu_{s_k}^\star \geq 0$. Thus, $\partial S_k(s_k) \mid_{s_k = s^{k\star}} = -\lambda^\star + \mu_{s_k}^\star$.

We have $\partial F_h^1(f_h^1) \mid_{f_h^1 = f_h^{1\star}} = -\lambda^\star \leq \partial S_k(s_k) \mid_{s_k = s_k^\star} = -\lambda^\star + \mu_{s_k}^\star \; \forall h \in J_1^1$ and $\forall k \in (M_1 \cap J_2^s)$, since $\mu_{s_k}^\star \geq 0$. This proves Case 1 of the lemma.

2. The Lagrangian function for point $f^{2\star}$ can be written as follows:

$$L(f^{2^\star}, \lambda^\star, \mu_2^\star) = \sum_{j=1}^{n} F_j^2(f_j^2) + \lambda^\star \cdot (C_{max} - \varepsilon) + \sum_{j=1}^{n} \mu_{2_j}^\star \cdot (f_l^2 - f_j^{2^\star}).$$

The term $\lambda^\star$ is unrestricted in sign and $\mu_{2_j}^\star \geq 0$. If we set $\nabla(L(f^{2^\star}, \lambda^\star, \mu_2^\star)) = 0$, we get $\partial F_j^2(f_j^2) \mid_{f_j^2 = f_j^{2^\star}} + \lambda^\star - \mu_{2_j}^\star = 0 \ \forall j$. Since $j \in J_1^2$, then $\mu_{2_j}^\star = 0$. Thus, $\partial F_j^2(f_j^2) \mid_{f_j^2 = f_j^{2^\star}} = -\lambda^\star$.

The Lagrangian function for point $s^\star$ can be written as follows:

$$L(s^\star, \lambda^\star, \mu_s^\star) = \sum_{k=1}^{n} S_k(s_k) + \lambda^\star \cdot (C_{max} - \varepsilon) + \sum_{k=1}^{n} \mu_{s_k}^\star \cdot (s_l - s_k^\star).$$

Similarly, the term $\lambda^\star$ is unrestricted in sign and $\mu_{s_k}^\star \geq 0$. If we set $\nabla(L(s^\star, \lambda^\star, \mu_s^\star)) = 0$, we get $\partial S_k(s_k) \mid_{s_k = s^{k^\star}} + \lambda^\star - \mu_{s_k}^\star = 0 \, \forall k$. Since $k \in (J_2^s \cap M^2)$, then $\mu_{s_k}^\star \geq 0$. Thus, $\partial S_k(s_k) \mid_{s_k = s^{k^\star}} = -\lambda^\star + \mu_{s_k}^\star$.

We have $\partial F_j^2(f_j^2) \mid_{f_j^2 = f_j^{2^\star}} = -\lambda^\star \leq \partial S_k(s_k) \mid_{s_k = s_k^\star} = -\lambda^\star + \mu_{s_k}^\star \ \forall j \in J_1^2$ and $\forall k \in (M_2 \cap J_2^s)$, since $\mu_{s_k}^\star \geq 0$. This proves Case 2 of the lemma.

3. The Lagrangian function for point $f^{1^\star}$ can be written as follows:

$$L(f^{1^\star}, \lambda^\star, \mu_1^\star) = \sum_{h=1}^{n} F_h^1(f_h^1) + \lambda^\star \cdot (C_{max} - \varepsilon) + \sum_{h=1}^{n} \mu_{1_h}^\star \cdot (f_l^1 - f_h^{1^\star}).$$

The term $\lambda^\star$ is unrestricted in sign and $\mu_{1_h}^\star \geq 0$. If we set $\nabla(L(f^{1^\star}, \lambda^\star, \mu_1^\star)) = 0$, we get $\partial F_h^1(f_h^1) \mid_{f_h^1 = f_h^{1^\star}} + \lambda^\star - \mu_{1_h}^\star = 0 \ \forall h$. Since $h \in J_2^1$, then $\mu_{1_h}^\star \geq 0$. Thus, $\partial F_h^1(f_h^1) \mid_{f_h^1 = f_h^{1^\star}} = -\lambda^\star + \mu_{1_h}^\star$.

The Lagrangian function for point $s^\star$ can be written as follows:

$$L(s^\star, \lambda^\star, \mu_s^\star) = \sum_{k=1}^{n} S_k(s_k) + \lambda^\star \cdot (C_{max} - \varepsilon) + \sum_{k=1}^{n} \mu_{s_k}^\star \cdot (s_l - s_k^\star).$$

Similarly, the term $\lambda^\star$ is unrestricted in sign and $\mu_{s_k}^\star \geq 0$. If we set $\nabla(L(s^\star, \lambda^\star, \mu_s^\star)) = 0$, we get $\partial S_k(s_k) \mid_{s_k = s^{k\star}} + \lambda^\star - \mu_{s_k}^\star = 0 \, \forall k$. Since $k \in (J_1^s \cap M^1)$, then $\mu_{s_k}^\star = 0$. Thus, $\partial S_k(s_k) \mid_{s_k = s^{k\star}} = -\lambda^\star$.

We have $\partial F_h^1(f_h^1) \mid_{f_h^1 = f_h^{1\star}} = -\lambda^\star + \mu_{1_h}^\star \geq \partial S_k(s_k) \mid_{s_k = s_k^\star} = -\lambda^\star \; \forall h \in J_2^1$ and $\forall k \in (M_1 \cap J_1^s)$, since $\mu_{1_h}^\star \geq 0$. This proves Case 3 of the lemma.

4. The Lagrangian function for point $f^{2^\star}$ can be written as follows:

$$L(f^{2^\star}, \lambda^\star, \mu_2^\star) = \sum_{j=1}^{n} F_j^2(f_j^2) + \lambda^\star \cdot (C_{max} - \varepsilon) + \sum_{j=1}^{n} \mu_{2_j}^\star \cdot (f_l^2 - f_j^{2^\star}).$$

The term $\lambda^\star$ is unrestricted in sign and $\mu_{2_j}^\star \geq 0$. If we set $\nabla(L(f^{2^\star}, \lambda^\star, \mu_2^\star)) = 0$, we get $\partial F_j^2(f_j^2) \mid_{f_j^2 = f_j^{2^\star}} + \lambda^\star - \mu_{2_j}^\star = 0 \; \forall j$. Since $j \in J_2^2$, then $\mu_{2_j}^\star \geq 0$. Thus, $\partial F_j^2(f_j^2) \mid_{f_j^2 = f_j^{2^\star}} = -\lambda^\star + \mu_{2_j}^\star$.

The Lagrangian function for point $s^\star$ can be written as follows:

$$L(s^\star, \lambda^\star, \mu_s^\star) = \sum_{k=1}^{n} S_k(s_k) + \lambda^\star \cdot (C_{max} - \varepsilon) + \sum_{k=1}^{n} \mu_{s_k}^\star \cdot (s_l - s_k^\star).$$

Similarly, the term $\lambda^\star$ is unrestricted in sign and $\mu_{s_k}^\star \geq 0$. If we set $\nabla(L(s^\star, \lambda^\star, \mu_s^\star)) = 0$, we get $\partial S_k(s_k) \mid_{s_k = s^{k\star}} + \lambda^\star - \mu_{s_k}^\star = 0 \, \forall k$. Since $k \in (J_1^s \cap M^2)$, then $\mu_{s_k}^\star = 0$. Thus, $\partial S_k(s_k) \mid_{s_k = s^{k\star}} = -\lambda^\star$.

We have $\partial F_j^2(f_j^2) \mid_{f_j^2 = f_j^{2\star}} = -\lambda^\star + \mu_{2_j}^\star \geq \partial S_k(s_k) \mid_{s_k = s_k^\star} = -\lambda^\star \; \forall j \in J_2^2$ and $\forall k \in (M_2 \cap J_1^s)$, since $\mu_{2_j}^\star \geq 0$. This proves Case 4 of the lemma.

5. The Lagrangian function for point $f^{1^\star}$ can be written as follows:

$$L(f^{1^\star}, \lambda^\star, \mu_1^\star) = \sum_{h=1}^{n} F_h^1(f_h^1) + \lambda^\star \cdot (C_{max} - \varepsilon) + \sum_{h=1}^{n} \mu_{1_h}^\star \cdot (f_l^1 - f_h^{1^\star}).$$

The term $\lambda^\star$ is unrestricted in sign and $\mu_{1_h}^\star \geq 0$. If we set $\nabla(L(f^{1^\star}, \lambda^\star, \mu_1^\star)) = 0$, we get $\partial F_h^1(f_h^1) \mid_{f_h^1 = f_h^{1^\star}} + \lambda^\star - \mu_{1_h}^\star = 0 \ \forall h$. Since $h \in J_1^1$, then $\mu_{1_h}^\star = 0$. Thus, $\partial F_h^1(f_h^1) \mid_{f_h^1 = f_h^{1^\star}} = -\lambda^\star$.

The Lagrangian function for point $s^\star$ can be written as follows:

$$L(s^\star, \lambda^\star, \mu_s^\star) = \sum_{k=1}^{n} S_k(s_k) + \lambda^\star \cdot (C_{max} - \varepsilon) + \sum_{k=1}^{n} \mu_{s_k}^\star \cdot (s_l - s_k^\star).$$

Similarly, the term $\lambda^\star$ is unrestricted in sign and $\mu_{s_k}^\star \geq 0$. If we set $\nabla(L(s^\star, \lambda^\star, \mu_s^\star)) = 0$, we get $\partial S_k(s_k) \mid_{s_k = s^{k^\star}} + \lambda^\star - \mu_{s_k}^\star = 0 \ \forall k$. Since $k \in (J_1^s \cap M^1)$, then $\mu_{s_k}^\star = 0$. Thus, $\partial S_k(s_k) \mid_{s_k = s^{k^\star}} = -\lambda^\star$.

We have $\partial F_h^1(f_h^1) \mid_{f_h^1 = f_h^{1^\star}} = -\lambda^\star = \partial S_k(s_k) \mid_{s_k = s_k^\star} = -\lambda^\star \ \forall h \in J_1^1$ and $\forall k \in (M_1 \cap J_1^s)$. This proves Case 5 of the lemma.

6. The Lagrangian function for point $f^{2^\star}$ can be written as follows:

$$L(f^{2^\star}, \lambda^\star, \mu_2^\star) = \sum_{j=1}^{n} F_j^2(f_j^2) + \lambda^\star \cdot (C_{max} - \varepsilon) + \sum_{j=1}^{n} \mu_{2_j}^\star \cdot (f_l^2 - f_j^{2^\star}).$$

The term $\lambda^\star$ is unrestricted in sign and $\mu_{2_j}^\star \geq 0$. If we set $\nabla(L(f^{2^\star}, \lambda^\star, \mu_2^\star)) = 0$, we get $\partial F_j^2(f_j^2) \mid_{f_j^2 = f_j^{2^\star}} + \lambda^\star - \mu_{2_j}^\star = 0 \ \forall j$. Since $j \in J_1^2$, then $\mu_{2_j}^\star = 0$. Thus, $\partial F_j^2(f_j^2) \mid_{f_j^2 = f_j^{2^\star}} = -\lambda^\star$.

The Lagrangian function for point $s^\star$ can be written as follows:

$$L(s^\star, \lambda^\star, \mu_s^\star) = \sum_{k=1}^{n} S_k(s_k) + \lambda^\star \cdot (C_{max} - \varepsilon) + \sum_{k=1}^{n} \mu_{s_k}^\star \cdot (s_l - s_k^\star).$$

Similarly, the term $\lambda^\star$ is unrestricted in sign and $\mu_{s_k}^\star \geq 0$. If we set $\nabla(L(s^\star, \lambda^\star, \mu_s^\star)) = 0$, we get $\partial S_k(s_k) \mid_{s_k=s^{k\star}} + \lambda^\star - \mu_{s_k}^\star = 0 \, \forall k$. Since $k \in (J_1^s \cap M^2)$, then $\mu_{s_k}^\star = 0$. Thus, $\partial S_k(s_k) \mid_{s_k=s^{k\star}} = -\lambda^\star$.

We have $\partial F_j^2(f_j^2) \mid_{f_j^2=f_j^{2\star}} = -\lambda^\star = \partial S_k(s_k) \mid_{s_k=s_k^\star} = -\lambda^\star \; \forall j \in J_1^2$ and $\forall k \in (M_2 \cap J_1^s)$. This proves Case 6 of the lemma. □

Lemma 3 is very important for the development of our proposed approximation algorithm. As a consequence of this lemma, in order to generate a new nondominated solution from an existing one by increasing the makespan value with a predetermined amount, we know which processing times has the greatest contribution to the cost. As a result of this lemma, an algorithm to generate a number of approximate nondominated solutions starts with an initial solution where all processing times are set to their lower bounds. By this way, the lower bound on the makespan can be attained using the procedure of Crama and Gultekin [12]. Then, by setting epsilon to this lower bound, the cost can be minimized by eliminating the idle times in the schedule (if any), by reassigning the flexible operations and increasing the processing times of some operations. This solution becomes the first nondominated solution. Then, ϵ is increased by a small amount to generate the next nondominated solution which will have a greater makespan than the initial one and smaller cost. The new processing times corresponding to the new value of makespan are calculated using Lemma 3 so that they approximate the optimal cost. We know by Lemma 1, that the constraint (3.10), which represents objective function upper bounded by ϵ, is satisfied as equality in an optimal solution to the problem. The determination of new processing time values of variables is performed by comparison of derivatives of the cost functions with respect to the processing times. Since derivatives show the contribution of a change in the processing time to the manufacturing cost, increasing the variable which has the smallest derivative of the cost function is beneficial in terms of cost.

Therefore, we increase the processing time of an operation of job j that has the smallest derivative value, and compare the new values of the derivatives. If the derivative of the cost function of increased variable is still smaller than the others on the same machine, the processing time value of this variable becomes valid. Otherwise, if the derivative of the cost function of increased variable becomes larger than any other on the same machine, we rearrange the values of processing times so that their derivatives become equal.

The following corollary determines the relation among the processing time values of operations of jobs processed on the same machine.

Corollary 1 *In an optimal solution to the problem, the following conditions hold.*

1. $f_1^{1\star} \leq f_h^{1\star} = f_j^{1\star} \quad \forall h, j \in \{2, 3, ..., n\}$
2. $f_n^{2\star} \leq f_h^{2\star} = f_j^{2\star} \quad \forall h, j \in \{1, 2, ..., n-1\}$
3. $s_h{}^\star = s_j{}^\star \quad \forall h, j \in M_1$
4. $s_h{}^\star = s_j{}^\star \quad \forall h, j \in M_2$

Corollary 1 is a direct consequence of Lemma 3. It tells us that the processing time values of operation 1 of jobs 2 to n should be equal. Likewise we know that the processing time values of operation 2 of jobs 1 to $n-1$ should be equal. For the flexible operation, the processing time values of jobs on the same machine should have the same values. Taking advantage of Corollary 1, we can represent the processing times of flexible operation for all jobs by two variables, which are s^1 and s^2. s^1 and s^2 stand for the processing time of flexible operations on machine 1 and machine 2, respectively.

4.2 Characteristics of the Approximation Algorithm

The following algorithmic rule points out the application of presented lemmas in the proposed approximation algorithm for elimination of dominated solutions to get nondominated ones. As mentioned before, a schedule with idle times may lead to a dominated solution, because a new schedule can be obtained with the same makespan value at a lower cost via elimination of idle times. This idea was highlighted in Example 1 in Section 3.4. Rule 1 determines the values of the new processing times in this nondominated solution found after elimination of idle times. In the following rule, r represents the number of flexible operations on machine 1, which is mentioned in Section 3.3. In this case, r can only have integer values.

Rule 1 Let $IdleM1$ represents the idle time amount on machine 1 that should be eliminated. Let, $f_j^{1^{new}}$ and $s^{1^{new}}$ be the values of the 1^{st} operation of job j and flexible operations of jobs processed on machine 1 after elimination of idle time, respectively. Then one of the following conditions hold:

1. $s^{1^{new}} = s^1$ and $f_j^{1^{new}} = \dfrac{IdleM1 + (n-1)\cdot f_n^1}{n-1}$ for j=2 to n.

2. $s^{1^{new}} = \dfrac{IdleM1 + r\cdot s^1}{r}$ and $f_j^{1^{new}} = f_j^1$ for j=2 to n.

3. $s^{1^{new}} = \dfrac{IdleM1 + (n-1)\cdot f_n^1 + r\cdot s^1}{r + (n-1)\cdot \sqrt[b-1]{\frac{A_s}{A_1}}}$ and
$f_j^{1^{new}} = \dfrac{IdleM1 + (n-1)\cdot f_n^1 + r\cdot s^1}{n-1+r\cdot \sqrt[b-1]{\frac{A_1}{A_s}}}$ for j=2 to n.

This algorithmic rule can be justified in following way. We know from Lemma 3 that the processing times values are determined making use of derivatives of cost functions. Let ∂F_j^1 and ∂S_j be the derivatives of cost functions $F_j^1(f_j^1)$ and $S_j(s_j)$ of job j, respectively. By means of Corollary 1, ∂F_j^1 and ∂S_j have the same values

for the jobs j=2 to n and the jobs processed on machine 1, respectively. Hence, let ∂S^1 represents the derivatives of cost functions $S_j(s_j)$ processed on machine 1. Since our aim is to eliminate idle time without increasing the makespan value, we can not make any change in the value of variable f_1^1. Such an increase in f_1^1 directly increases the makespan value.

If min $\{\partial F_j^1\ , \partial S^1\}$ is ∂F_j^1, the following procedure is applied. At first, the new values of f_j^1 for j=2 to n are computed assuming s^1 remains the same. The idle time is distributed to these $n-1$ variables by the formula $f_j^{1^{new}} = (IdleM1 + (n-1) \cdot f_j^1)/(n-1)$ for j=2 to n. The idle time distributed to the variables is equal as a consequence of Corollary 1. Then, the new values of ∂F_j^1, $\partial F_j^{1^{new}}$, are computed and compared with ∂S^1. If $\partial F_j^{1^{new}} \leq \partial S^1$, the new values of f_j^1 for j=2 to n is decided to be valid. This means Case 1 holds. If $\partial F_j^{1^{new}} \geq \partial S^1$, the assignment becomes invalid. In this case, the idle time should be distributed to f_j^1 for j=2 to n and s^1 by considering the equality of derivatives by the 5^{th} Item of Lemma 3. Then, we know:

$$IdleM1 = (n-1) \cdot (f_j^{1^{new}} - f_j^1) + r \cdot (s^{1^{new}} - s^1) \quad \text{for} \quad j = 2 \quad \text{to} \quad n \tag{4.1}$$

$$A^1 \cdot b \cdot (f_j^{1^{new}})^{b-1} = A^s \cdot b \cdot (s^{1^{new}})^{b-1} \quad \text{for} \quad j = 2 \quad \text{to} \quad n \tag{4.2}$$

Using above relations we get $f_j^{1^{new}}$ and $s^{1^{new}}$ as presented below which means Case 3 holds.

$$f_j^{1^{new}} = \frac{IdleM1 + (n-1) \cdot f_j^1 + r \cdot s^1}{n - 1 + r \cdot \sqrt[b-1]{\frac{A^1}{A^s}}} \quad \text{for} \quad j = 2 \quad \text{to} \quad n \tag{4.3}$$

$$s^{1^{new}} = \frac{IdleM1 + (n-1) \cdot f_j^1 + r \cdot s^1}{r + (n-1) \cdot \sqrt[b-1]{\frac{A^s}{A^1}}} \quad \text{for} \quad j = 2 \quad \text{to} \quad n \tag{4.4}$$

If min $\{\partial F_j^1\ , \partial S^1\}$ is ∂S^1, the following procedure is applied. At first, the new value of s^1 is computed assuming f_j^1 remains the same. Since, s^1 represents the processing time value of jobs whose flexible operations are processed on machine 1 and we have r flexible operations on machine 1, the idle time is distributed equally as a consequence of Corollary 1 to these r variables by the formula $s^{1^{new}} =$

$(IdleM1 + r \cdot s^1)/r$. Then, the new derivative of cost function of variable $s^{1^{new}}$, $\partial S^{1^{new}}$, is computed and compared with ∂F_j^1. If $\partial S^{1^{new}} \leq \partial F_j^1$, the new value of s^1 is decided to be valid. Then Case 2 holds. If $\partial S^{1^{new}} \geq \partial F_j^1$, the assignment becomes invalid. Proceeding similarly as before we again conclude that Case 3 holds. The analysis for elimination of the idle time on machine 2 is similar.

Given a nondominated solution, to determine the next efficient point on the frontier, the upper limit ϵ of the constraint representing makespan objective function is increased by "E" and the new processing time values are determined accordingly. The following rule determines the processing time values for a given makespan increment "E" to minimize manufacturing cost while satisfying the makespan constraint, (3.10), as equality.

Rule 2 Let $f_j^{1^{new}}$ and $s^{1^{new}}$ be the values of the 1^{st} operation of job j and flexible operations of jobs processed on machine 1 at the next point on the efficient frontier, respectively. Similarly, let $f_j^{2^{new}}$ and $s^{2^{new}}$ be the values of the 2^{nd} operation of job j and flexible operations of jobs processed on machine 2 at the next point on the efficient frontier, respectively. Then,

1. If min $\{\partial F_j^1$, ∂F_j^2, ∂S^1, $\partial S^2\}$ is ∂F_j^1 for j=1 to n, then one of the conditions, a and b, hold for the processing time variables on machine 1 and one of the conditions, c, d and e, hold for the processing time variables on machine 2 as long as the assignment of flexible operations remains the same.

2. If min $\{\partial F_j^1$, ∂F_j^2, ∂S^1, $\partial S^2\}$ is ∂S^1, then one of the conditions, b and f, hold for the processing time variables on machine 1 and one of the conditions, c, d and e, hold for the processing time variables on machine 2 as long as the assignment of flexible operations remains the same.

3. If min $\{\partial F_j^1$, ∂F_j^2, ∂S^1, $\partial S^2\}$ is ∂F_j^2 for j=1 to n, then one of the conditions, g and h, hold for the processing time variables on machine 2 and one of conditions, i, j and k, hold for the processing time variables on machine

1 as long as the assignment of flexible operations remains the same.

4. If min $\{\partial F_j^1$, ∂F_j^2, ∂S^1, $\partial S^2\}$ is ∂S^2, then one of the conditions, h and l, hold for the processing time variables on machine 2 and one of the conditions, i, j and k, hold for the processing time variables on machine 1 as long as the assignment of flexible operations remains the same.

a. $f_j^{1new} = \dfrac{E + \sum_{j=1}^{n} f_j^1}{n}$ for j=1 to n and $s^{1new} = s^1$

b. $f_j^{1new} = \dfrac{E + \sum_{j=1}^{n} f_j^1 + r \cdot s_1}{n + r \cdot \sqrt[b-1]{\frac{A^1}{A^s}}}$ for j=1 to n and

$s^{1new} = \dfrac{E + \sum_{j=1}^{n} f_j^1 + r \cdot s_1}{r + n \cdot \sqrt[b-1]{\frac{A^s}{A^1}}}$

c. $f_j^{2new} = \dfrac{E - (f_1^{1new} - f_1^1) + \sum_{j=1}^{n-1} f_j^2}{n-1}$ for j=1 to $n-1$,

$f_n^{2new} = f_n^2$ and $s^{2new} = s^2$

d. $f_j^{2new} = f_j^2$ for j=1 to n and $s^{2new} = \dfrac{E - (f_1^{1new} - f_1^1) + (n-r) \cdot s^2}{n-r}$

e. $f_j^{2new} = \dfrac{E - (f_1^{1new} - f_1^1) + \sum_{j=1}^{n-1} f_j^2 + (n-r) \cdot s^2}{n - 1 + (n-r) \cdot \sqrt[b-1]{\frac{A_1}{A_s}}}$ for j=1 to $n-1$,

$f_n^{2new} = f_n^2$ and $s^{2new} = \dfrac{E - (f_1^{1new} - f_1^1) + \sum_{j=1}^{n-1} f_j^2 + (n-r) \cdot s^2}{n - r + (n-1) \cdot \sqrt[b-1]{\frac{A_s}{A_1}}}$

f. $f_j^{1new} = f_j^1$ for j=1 to n and $s^{1new} = \dfrac{E + r \cdot s^1}{r}$

g. $f_j^{2new} = \dfrac{E + \sum_{j=1}^{n} f_j^2}{n}$ for j=1 to n and $s^{2new} = s^2$

h. $f_j^{2new} = \dfrac{E + \sum_{j=1}^{n} f_j^2 + (n-r) \cdot s_2}{n + (n-r) \cdot \sqrt[b-1]{\frac{A^2}{A^s}}}$ for j=1 to n and

$$s^{2new} = \frac{E + \sum_{j=1}^{n} f_j^2 + (n-r) \cdot s_2}{n - r + n \cdot \sqrt[b-1]{\frac{A^s}{A^2}}}$$

i. $f_j^{1new} = \dfrac{E - (f_n^{2new} - f_n^2) + \sum_{j=2}^{n} f_j^1}{n-1}$ for j=2 to n,

$$f_1^{1new} = f_1^1 \quad \text{and} \quad s^{1new} = s^1$$

j. $f_j^{1new} = f_j^1$ for j=1 to n and $s^{1new} = \dfrac{E - (f_n^{2new} - f_n^2) + r \cdot s^1}{r}$

k. $f_j^{1new} = \dfrac{E - (f_n^{2new} - f_n^2) + \sum_{j=2}^{n} f_j^1 + r \cdot s^1}{n - 1 + r \cdot \sqrt[b-1]{\frac{A_2}{A_s}}}$ for j=2 to n,

$$f_1^{1new} = f_1^1 \quad \text{and} \quad s^{1new} = \frac{E - (f_n^{2new} - f_n^2) + \sum_{j=2}^{n} f_j^1 + r \cdot s^1}{r + (n-1) \cdot \sqrt[b-1]{\frac{A_s}{A_2}}}$$

l. $f_j^{2new} = f_j^2$ for j=1 to n and $s^{2new} = \dfrac{E + (n-r) \cdot s^2}{n-r}$.

This algorithmic rule can be justified in the following way. We know by Lemma 1 that the makespan constraint should be satisfied as equality. Since we

increase the makespan value by "E" to determine the next efficient point on the frontier, we need to increase the processing time variables accordingly.

First, the minimum of ∂F_j^1, ∂F_j^2, ∂S^1, and ∂S^2 is selected. If the minimum is ∂F_j^1, assuming that s^1 remains the same, "E" is distributed equally among variables f_j^1 for j=1 to n as a consequence of Corollary 1 by the following formula:

$$f_j^{1^{new}} = \frac{E + \sum_{j=1}^{n} f_j^1}{n} \quad \text{for} \quad j = 1 \quad \text{to} \quad n \tag{4.5}$$

Then, the new derivative of cost function of variable $f_j^{1^{new}}$, $\partial F_j^{1^{new}}$, is computed and compared with ∂S^1. If $\partial F_j^{1^{new}} \leq \partial S^1$, $f_j^{1^{new}}$ for j=1 to n are decided to be valid. Then, Case a holds. If this is not the case, that is $\partial F_j^{1^{new}} > \partial S^1$, the assignment becomes invalid. In this case, "E" should be distributed to f_j^1 for j=1 to n and s^1 by considering the equality of derivatives by the 5^{th} item of Lemma 3. Then, we know:

$$E = \sum_{j=1}^{n} (f_j^{1^{new}} - f_j^1) + r \cdot (s^{1^{new}} - s^1) \tag{4.6}$$

$$A^1 \cdot b \cdot (f_j^{1^{new}})^{b-1} = A^s \cdot b \cdot (s^{1^{new}})^{b-1} \quad \text{for} \quad j = 1 \quad \text{to} \quad n \tag{4.7}$$

Using above relations we get $f_j^{1^{new}}$ and $s^{1^{new}}$ as presented below:

$$f_j^{1^{new}} = \frac{E + \sum_{j=1}^{n} f_j^1 + r \cdot s_1}{n + r \cdot \sqrt[b-1]{\frac{A^1}{A^s}}} \quad \text{for} \quad j = 1 \quad \text{to} \quad n \tag{4.8}$$

$$s^{1^{new}} = \frac{E + \sum_{j=1}^{n} f_j^1 + r \cdot s_1}{r + n \cdot \sqrt[b-1]{\frac{A^s}{A^1}}} \quad \text{for} \quad j = 1 \quad \text{to} \quad n \tag{4.9}$$

Then, Case b holds. In this rule up to now, we selected the minimum of cost function derivatives of processing time variables and according to the machine on which the selected processing time variable is processed, we made necessary

adjustments on the variables processed on that machine. In this case, we selected ∂F_j^1 as minimum and assigned the new processing time values to the operations on machine 1. According to the Lemma 2, the idle time appeared on machine 2 should be covered. After the elimination of idle time on machine 2, the Cases c, d, e holds for $f_j^{2^{new}}$ and $s^{2^{new}}$.

The assignment of the new processing time values can be done similarly according to the selection of min $\{\partial F_j^1$, ∂F_j^2, ∂S^1, $\partial S^2\}$.

The proposed approximation algorithm generates a set of nondominated solutions. The following lemma presents this important property of the algorithm.

Lemma 4 *Any two solutions generated by the approximation algorithm can not dominate each other.*

Proof. Let $Z_1^\star = \sum_{j=1}^{n} \sum_{i=1}^{2} (F_j^i(f_j^{\star i}) + S_j(s_j^\star))$ be the objective function value with processing time vectors $f^\star$ and $s^\star$ and let ${C_{max}}^\star = T_{n,2} + f_n^{\star 2} + s_n^\star \cdot (1 - x_n)$ be makespan value generated by the algorithm. A new solution is generated by incrementing the makespan value. Let $\hat{Z}_1 = \sum_{j=1}^{n} \sum_{i=1}^{2} (F_j^i(\hat{f}_j^i) + F_j(\hat{s_j}))$ be the objective function value with processing time vectors $\hat{f}$ and $\hat{s}$ and let $\hat{C_{max}} = T_{n,2} + \hat{f_n^2} + \hat{s_n} \cdot (1 - x_n)$ be makespan value of this new solution. Hence, $\hat{C_{max}} > {C_{max}}^\star$, which means at least one of the processing time values has increased in the new solution. Since the cost function is decreasing with respect to processing times, the objective function of the new solution, $\hat{Z}_1$, satisfies $\hat{Z}_1 < Z_1^\star$. This proves that the solutions generated by the approximation algorithm can not dominate each other and a new nondominated solution is generated at each iteration of the algorithm. □

4.3 Summary

In this chapter, we considered some theoretical properties of the problem that will be used in the development of approximation algorithm. We also presented

some rules to demonstrate the application of the presented lemmas. Next chapter presents the approximation algorithm.

Chapter 5

Approximation Algorithm

In this chapter, we propose a heuristic search algorithm, that will be named as Main Algorithm throughout the text. Main Algorithm generates a set of approximately efficient solutions which are equally spaced on the efficient frontier that can be presented to the decision maker quickly. Since we are using a heuristic approach to find the minimum manufacturing cost value, for a given makespan value, these solutions do not dominate each other but there can be another solution generated by an exact algorithm that could dominate any one of them. The algorithm determines the processing time values of the operations and allocates the flexible operations to the machines.

Main Algorithm starts with an initial schedule at which the processing times are at their lower bounds and the optimal assignment of flexible operations to machines are determined according to the procedure developed by Crama and Gultekin [12]. As we mentioned before, if there exists idle times in the solution obtained via the procedure Crama and Gultekin [12] proposed, this solution may be a dominated point. We extend this proposition by eliminating idle times making use of controllability as in Example 1. We use Rule 1 to determine the new processing time values and get a nondominated initial solution.

Starting from a nondominated initial solution, an increment value "E" is selected and then the corresponding allocation of the flexible operations and the

processing time values are decided according to the new makespan value to determine a set of efficient discrete points of makespan and manufacturing cost objectives. The pseudo-code is displayed as Main Algorithm. Let r_l and C_{maxl} represent the number of flexible operations assigned to machine 1 and the value of makespan, respectively, when the processing time variables are equal to their lower bounds. Also let r_u and C_{maxu} represent the number of flexible operations assigned to machine 1 and the value of makespan, respectively, when the processing time variables are equal to their upper bounds.

Main algorithm gets the lower bounds of operation processing times as an input and initially assigns lower bounds to the processing time variables. Then the algorithm computes r_l and C_{maxl}. The algorithm assigns the allocation of flexible operations to the machines to binary variable x_j according to r_l. x_j is assigned "1" if the flexible operation is assigned to the first machine, and "0" otherwise. According to the flexible operation allocations, starting times of jobs on each machine, $T_{j,m}$, are computed. At this instant, we get a schedule in which the processing times of operations of all jobs are at their lower bounds. However, if there exists an idle time in the schedule, we get a dominated solution because increasing the processing times of operations of jobs till there exists no idle time decreases the cost without changing the makespan value as shown in Example 1. Therefore, algorithm checks out for idle times on machines. If there exists any, it covers idle times as discussed in Rule 1. After that, the algorithm computes r_u and C_{maxu}. In the for loop of approximation algorithm, makespan value is increased by "E" to determine efficient points on the efficient frontier as discussed in Rule 2. Allocation of flexible operations and processing time values are determined using Subroutines M1 or M2 till we reach to C_{maxu}.

The Subroutines M1 and M2 start with increasing makespan by "E" on machines 1 and 2, respectively. Increasing makespan on the selected machine incurs an idle time on the other machine. $\Delta(f_1^1)$ and $\Delta(f_n^2)$ represents ($f_1^{1^{new}}$ - f_1^1) and ($f_n^{2^{new}}$ - f_n^2), respectively, where $f_1^{1^{new}}$ and $f_n^{2^{new}}$ are the processing time values of f_1^1 and f_n^2 after increasing the total processing time on machines 1 and 2 by E. Hence, the subroutines cover the idle time using the idea presented in Rule

Main Algorithm Approximation Algorithm

Input: f_l^1, f_l^2, s_l, f_u^1, f_u^2, s_u, F_j^1 and F_j^2, S^1, S^2, E

Output: A number of nondominated (C_{max}, Total Cost) pairs with corresponding, r, f_j^1, f_j^2, s^1, s^2, $T_{j,m}$ values for each point

Set processing times to their lower bounds
Compute r_l and C_{maxl}
Assign the flexible operations to the machines according to r_l
Compute starting times of jobs on each machine, $T_{j,m}$
if $T_{n,1} + f_n^1 + x_n \cdot s_1 \leq T_{n,2}$ **then**
    $IdleM1 \leftarrow T_{n,2} - (T_{n,1} + f_n^1 + x_n \cdot s^1)$
    Cover idle time on machine 1 using Rule 1
end if
if $T_{n-1,2} + f_1^2 + (1 - x_{n-1}) \cdot s^2 \leq T_{n,1} + f_n^1 + x_n \cdot s^1$ **then**
    $IdleM2 \leftarrow T_{n,1} + f_n^1 + x_n \cdot s^1 - (T_{n-1,2} + f_1^2 + (1 - x_{n-1}) \cdot s^2)$
    Cover idle time on machine 2 using Rule 1
end if
Compute r, r_u and C_{maxu}
for $j = 1$ to $(C_{maxu} - C_{maxl})/E$ **do**
    if $min\{\partial F_j^1, \partial F_j^2, \partial S^1, \partial S^2\} = \partial F_j^1$ or ∂S^1 **then**
        Use Subroutine M1 to determine allocation of flexible operations and processing time values
    else if $min\{\partial F_j^1, \partial F_j^2, \partial S^1, \partial S^2\} = \partial F_j^2$, or ∂S^2 **then**
        Use Subroutine M2 to determine allocation of flexible operations and processing time values
    end if
end for

Subroutine M1 Increasing the total processing time on machine 1 by E

Increase the total processing time on machine 1 by E using Rule 2
$IdleM2 \leftarrow E - \Delta(f_1^1)$
Cover idle time on machine 2 using Rule 1
Check upper bounds of variables for any violation and make necessary corrections
Compute manufacturing cost
if $r > r_u$ **then**
  $IdleM2 \leftarrow E - \Delta(f_1^1)$
  if $s_l - IdleM2 \leq (n-r) \cdot (s^2 - s_l) + (n-1) \cdot (f_1^2 - f_l^2)$ **then**
    $r \leftarrow r - 1$
    $IdleM2 \leftarrow IdleM2 - s^2$
    Cover idle time on machine 2 using Rule 1
    $IdleM1 \leftarrow s^1$
    Cover idle time on machine 1 using Rule 1
    Check upper and lower bounds of variables for any violation and make necessary corrections
    Compute manufacturing cost
  end if
end if
if $r < r_u$ **then**
  if $s_l - E \leq r \cdot (s^1 - s_l) + (n-1) \cdot (f_n^1 - f_l^1)$ **then**
    $r \leftarrow r + 1$
    $IdleM1 \leftarrow E - s^1$
    Cover idle time on machine 1 using Rule 1
    $IdleM2 \leftarrow E + s^2$
    Cover idle time on machine 2 using Rule 1
    Check upper and lower bounds of variables for any violation and make necessary corrections
    Compute manufacturing cost
  end if
end if
Output: the minimum manufacturing cost and associated processing time variables

1. Afterwards, any upper bound violations of processing time variables are investigated. If there exists an operation processing time which is greater than its upper bound, it is assigned to its upper bound, and the total reduction in the value of this variable is distributed to the other operations on the same machine as an increase. If this operation also becomes greater than its upper bound after increasing its processing time, it is also assigned to its upper bound. In such a case all operations on this machine hit their upper bounds, so we cannot make any increase of processing times on this machine any more and some idle time remain uncovered. After this correction, manufacturing cost is computed with the processing time variables using the following formula:

$$\text{Manufacturing Cost} = (f_1^1+(n-1)\cdot f_n^1+(n-1)\cdot f_1^2+f_n^2+r\cdot s^1+(n-r)\cdot s^2)\cdot C+ ((f_1^1)^b+(n-1)\cdot(f_1^1)^b)\cdot A^1+((n-1)\cdot(f_1^2)^b+(f_n^2)^b)\cdot A^2+(r\cdot(s^1)^b+(n-r)\cdot(s^2)^b)\cdot A^s.$$

Then Subroutines M1 and M2 compares r value with r_u to check whether changing the assignment of one of the flexible operations reduces the cost or not. If the number of flexible operations on machine 1, represented as r, is equal to the number of flexible operations on machine 1 when all processing times are at their upper bounds, represented as r_u, the assignment of flexible operations should not change and remain the same till makespan reaches to C_{maxu}. In this case, the subroutines terminate by returning the value of manufacturing cost and associated processing time values of operations of jobs. If r is greater than r_u, the last flexible operation on machine 1 is assigned to the machine 2 if possible. This assignment may not always be possible because of makespan limitation constraint, so the subroutines investigate whether a flexible operation can fit on the machine 2 while all processing time variables are at their lower bounds by the following if statements:

IF $s_l - IdleM2 \leq (n-r)\cdot(s^2-s_l)+(n-1)\cdot(f_1^2-f_l^2)$.

IF $s_l - E \leq (n-r)\cdot(s^2-s_l)+(n-1)\cdot(f_1^2-f_l^2)$.

If it is proved to be feasible, the assignment of flexible operation and the values of processing times change accordingly. This may cause some of the processing time variables to decrease or increase. If any processing time value exceeds its

Subroutine M2 Increasing the total processing time on machine 2 by E

Increase the total processing time on machine 2 by E using Rule 2
$IdleM1 \leftarrow E - \Delta(f_n^2)$
Cover idle time on machine 1 using Rule 1
Check upper bounds bounds of variables for any violation and make necessary corrections
Compute manufacturing cost
if $r > r_u$ **then**
  if $s_l - E \leq (n-r) \cdot (s^2 - s_l) + (n-1) \cdot (f_1^2 - f_l^2)$ **then**
    $r \leftarrow r - 1$
    $IdleM2 \leftarrow E - s^2$
    Cover idle time on machine 2 using Rule 1
    $IdleM1 \leftarrow E + s^1$
    Cover idle time on machine 1 using Rule 1
    Check upper and lower bounds bounds of variables for any violation and make necessary corrections
    Compute manufacturing cost
  end if
end if
if $r < r_u$ **then**
  $IdleM1 \leftarrow E - \Delta(f_n^2)$
  if $s_l - IdleM1 \leq r \cdot (s^1 - s_l) + (n-1) \cdot (f_n^1 - f_l^1)$ **then**
    $r \leftarrow r + 1$
    $IdleM1 \leftarrow IdleM1 - s^1$
    Cover idle time on machine 1 using Rule 1
    $IdleM2 \leftarrow s^2$
    Cover idle time on machine 2 using Rule 1
    Check upper and lower bounds bounds of variables for any violation and make necessary corrections
    Compute manufacturing cost
  end if
end if
Output: the minimum manufacturing cost and associated processing time variables

upper bound, the necessary modification is made as already mentioned. On the other hand, in case of a lower bound violation of an operation processing time, the variable is assigned to its lower bound, and the total increase in the value of this variable is distributed to the other operation on the same machine as a reduction. If this operation also becomes smaller than its lower bound after reducing its processing time, it is assigned to its lower bound. At the worst case, all processing times hit their lower bounds. Afterwards, manufacturing cost is computed. For the case of r being smaller than r_u, the first flexible operation on machine 2 is assigned to the machine 1 if possible. The same procedure is applied with the former case. Finally the subroutines output the minimum manufacturing cost and associated processing time variables.

In the next chapter, an experimental design and analysis of the results for the approximation algorithm will be provided.

Chapter 6

Computational Results

In this chapter, we perform a computational study to test the performance of our proposed approximation algorithm by comparing it with the mathematical formulations, Model 1 and 2. Mixed integer nonlinear programs are formulated in GAMS 22.0. One of the alternatives to solve MINLP formulations is to call BARON solver. This solver guarantees to provide global optima under fairly general assumptions using deterministic global optimization algorithms of the branch-and-bound type. However the CPU time requirement of BARON may be prohibitive and it may not be possible to solve problems in a reasonable time, so we run BARON with a time limit of 1000 seconds. Another alternative for generating good quality solutions in smaller CPU times is to use DICOPT solver. DICOPT guarantees to converge only under certain convexity assumptions. Although the algorithm has provisions to handle non-convexities, it does not necessarily obtain the global optimum. Due to CPU restrictions, it is not possible to run the mathematical model till the end by DICOPT. Therefore, we also run DICOPT with the same time limit as BARON. We observed that at some replications DICOPT cannot generate integer solutions but presents relaxed solution with this time limit. On the contrary, no such points exist for BARON.

Approximation algorithm is coded in the C++ language and compiled with Gnu compiler. The DICOPT is ran on a computer with 3GB memory and dual

core Intel Pentium processor with 2.1 GHz CPU. However, due to licensing limitations, the BARON software is ran on a computer with 1294 MB memory and Pentium III 1133 MHz CPU with a time limit of 1000 seconds.

6.1 Experimental Settings

There are four experimental factors that can affect the efficiency of the algorithm as listed in Table 6.1. The experimental design is a 2^4 full factorial design. The factors A^1, A^2 and A^s are effective on the upper bounds of the processing times and on the manufacturing cost as presented before. We choose the factors from two different levels, which allows us to have different upper bounds for the processing times of operations and different cost functions for the operations. The Levels 1 and 2 will be represented as L and H, meaning low and high, in the next tables, respectively. The diversity of upper bounds changes the assignment of flexible operations which is expected to affect the efficiency of the proposed approximation algorithm. The reason of A^1, A^2, and A^s being selected from the same intervals at the same level randomly is to enable the processing time variable upper bounds to get the same values in some cases which allows us to observe the efficiency of the algorithm in case of equal upper bounds of processing time variables. The factor n determines the size of the problem. When the problem size is large, it takes more CPU time to solve the problem, and the resulting cost and makespan objective values are expected to be high.

Table 6.1: Experimental Factors

Factors	Definition	Level 1	Level 2
A^1	Tool cost multiplier of first operation	U[5,8]	U[12,15]
A^2	Tool cost multiplier of second operation	U[5,8]	U[12,15]
A^s	Tool cost multiplier of flexible operation	U[5,8]	U[12,15]
n	Number of jobs	20	30

Most of the exact algorithms need huge CPU times in case of large problem sizes, so our aim is to provide an approximation algorithm that presents qualified solutions in small CPU times. The other parameters are selected randomly from

the intervals of C = U[0.4 , 0.8], f_l^1 = U[1.2 , 1.7], f_l^2 =U[1.4 , 1.9], s_l =U[1.6 , 2.1], b =U[-1.7 , -1.3], where U[a,b] is uniform distribution in interval [a , b].

We took five replications for each factor combination and 25 different efficient point generation iterations for each replication resulting in $2^4 \cdot 5 \cdot 25 = 2000$ individual runs for proposed approximation algorithm. The performance measures used in evaluating the experimental results are the percentage deviations of the approximation algorithm from the mathematical formulations, Model 1 and 2, and the run times in CPU seconds.

The percent deviation of each run is calculated as $100 \cdot (A - B)/B$, where A represents the manufacturing cost value delivered by the approximation algorithm and B represents the solution of mathematical formulation obtained by DICOPT or BARON solvers for a given makespan value.

The algorithm underlying DICOPT starts by solving the nonlinear program (NLP) in which the 0-1 conditions on the binary variables are relaxed. If the solution to this problem yields an integer solution the search stops. Otherwise, it continues with an alternating sequence of NLP's called subproblems and mixed-integer linear programs (MIP) called master problems. The NLP subproblems are solved for fixed 0-1 variables that are predicted by the MIP master problem at each (major) iteration. In this algorithm, linearization of the constraints is thought to be helpful, so we also take runs of Model 2. Since Models 1 and 2 can not be decided to be better than the other and at each iteration there exists efficient points at which the models improve each other, we present runs of both models with DICOPT. BARON runs are only taken for Model 1, because overall percent deviations of approximation algorithm from Model 1 and 2, solved by DICOPT, show that Model 1 performs better than Model 2.

6.2 Sample Replication Analysis

The points generated by the algorithm and mathematical programming formulations using DICOPT, Model 1 and 2, for one sample replication with a factor

combination A^1= U[12,15] , A^2= U[5,8], A^s= U[5,8], n=20, and the corresponding % deviations are presented in Table 6.2. The best (smallest) manufacturing cost is written in bold for each makespan value. As can be seen in the table, while some of the deviations are positive, the others are negative. A positive deviation means that DICOPT finds a better solution than our proposed algorithm. On the other hand, a negative deviation means our approximation algorithm can find a better solution than DICOPT at that makespan value. This is possible since DICOPT does not necessarily obtain the global optimum. The minimum deviation, which is -0.0008, from Model 1 appears at a makespan value 85.2. The maximum deviation, which is 0.0018, from Model 1 appears at three different makespan values 60.6, 62.2, and 63.8. The minimum deviation, which is -0.0080, and the maximum deviation, which is 0.0018, from Model 2 appear at a makespan value 81.9 and 60.6, respectively. As can be seen in Table 6.2, the deviations are small for smaller makespan values, then slightly increase through the mid-points of the efficient frontier. Deviations get smaller through the upper bound of the makespan.

For the same factor combination given above, % deviations of the approximation algorithm from Model 1 solved by BARON are presented in Table 6.3. The minimum deviation, which is 0, appears at the first point and last three points generated. The maximum deviation of the approximation algorithm from Model 1 solved by BARON is the same as DICOPT and appears at the same points. As can be seen in the table, all deviations are nonnegative. This means solver BARON performs better than both DICOPT and the proposed approximation algorithm. However, even the maximum deviation is very small which indicates that proposed approximation algorithm generates high quality solutions. Moreover, the deviations get smaller through the lower and upper bounds of the makespan.

The minimums of manufacturing costs for Model 1 and 2 generated by DICOPT for the sample replication are presented in Table 6.4 to compare them with the manufacturing costs for Model 1 for the same replication generated by BARON. The deviations are given as percent deviations and calculated by $100 \cdot (DICOPT - BARON)/BARON$. As can be seen in the table, BARON

Table 6.2: Manufacturing Costs and % Deviations (DICOPT) for One Replication

	Manufacturing Cost			% Deviation	
Makespan	Proposed Algorithm	Model 1	Model 2	Model 1	Model 2
52.4	**312.90**	**312.90**	**312.90**	0.00	0.00
54.0	296.37	**296.21**	**296.21**	0.05	0.05
55.6	284.24	**284.02**	**284.02**	0.08	0.08
57.3	274.85	**274.55**	**274.55**	0.11	0.11
58.9	268.31	**267.87**	**267.87**	0.16	0.16
60.6	262.58	**262.11**	**262.11**	0.18	0.18
62.2	257.73	**257.27**	257.68	0.18	0.02
63.8	253.38	**252.93**	254.36	0.18	-0.39
65.5	249.24	**248.82**	250.55	0.17	-0.52
67.1	245.76	**245.39**	246.71	0.15	-0.38
68.8	242.48	**242.23**	242.78	0.10	-0.13
70.4	239.73	**239.51**	240.62	0.09	-0.37
72.0	237.30	**237.06**	237.48	0.10	-0.08
73.7	235.02	**234.82**	234.90	0.09	0.05
75.3	233.16	**232.99**	233.32	0.07	-0.07
77.0	**231.43**	231.53	231.54	-0.04	-0.05
78.6	230.04	**229.94**	230.22	0.05	-0.07
80.2	228.86	**228.78**	**228.78**	0.04	0.04
81.9	227.82	**227.75**	229.65	0.03	-0.80
83.5	227.01	**226.97**	227.79	0.02	-0.34
85.2	226.34	226.51	**226.31**	-0.08	0.01
86.8	**225.87**	225.90	226.05	-0.01	-0.08
88.5	225.52	**225.51**	225.60	0.00	-0.03
90.1	**225.33**	**225.33**	**225.33**	0.00	0.00
91.7	**225.27**	**225.27**	**225.27**	0.00	0.00

Table 6.3: Manufacturing Costs and % Deviations (BARON) for One Replication

	Manufacturing Cost		
Makespan	Proposed Algorithm	Model 1	% Deviation
52.4	312.90	312.90	0.00
54.0	296.37	296.21	0.05
55.6	284.24	284.02	0.08
57.3	274.85	274.55	0.11
58.9	268.31	267.87	0.16
60.6	262.58	262.11	0.18
62.2	257.73	257.27	0.18
63.8	253.38	252.93	0.18
65.5	249.24	248.83	0.17
67.1	245.76	245.39	0.15
68.8	242.48	242.15	0.13
70.4	239.73	239.45	0.12
72.0	237.30	237.06	0.10
73.7	235.02	234.82	0.09
75.3	233.16	232.99	0.07
77.0	231.43	231.30	0.06
78.6	230.04	229.94	0.05
80.2	228.86	228.78	0.04
81.9	227.82	227.75	0.03
83.5	227.01	226.97	0.02
85.2	226.34	226.31	0.01
86.8	225.87	225.85	0.01
88.5	225.52	225.51	0.00
90.1	225.33	225.33	0.00
91.7	225.27	225.27	0.00

Table 6.4: DICOPT vs BARON for One Replication

	Manufacturing Cost		
Makespan	DICOPT	BARON	% Deviation
52.4	312.9	312.9	0.00
54	296.21	296.21	0.00
55.6	284.02	284.02	0.00
57.3	274.55	274.55	0.00
58.9	267.87	267.87	0.00
60.6	262.11	262.11	0.00
62.2	257.27	257.27	0.00
63.8	252.93	252.93	0.00
65.5	248.82	248.83	0.00
67.1	245.39	245.39	0.00
68.8	242.23	242.15	**0.03**
70.4	239.51	239.45	**0.03**
72	237.06	237.06	0.00
73.7	234.82	234.82	0.00
75.3	232.99	232.99	0.00
77	231.53	231.3	**0.10**
78.6	229.94	229.94	0.00
80.2	228.78	228.78	0.00
81.9	227.75	227.75	0.00
83.5	227.79	226.97	**0.36**
85.2	226.31	226.31	0.00
86.8	225.9	225.85	**0.02**
88.5	225.51	225.51	0.00
90.1	225.33	225.33	0.00
91.7	225.27	225.27	0.00

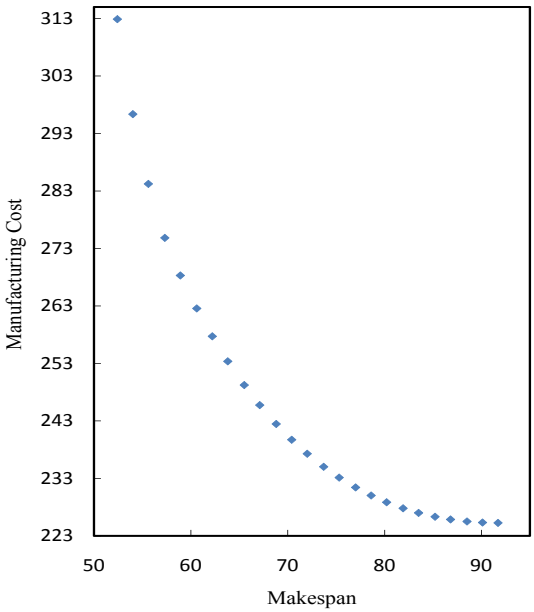
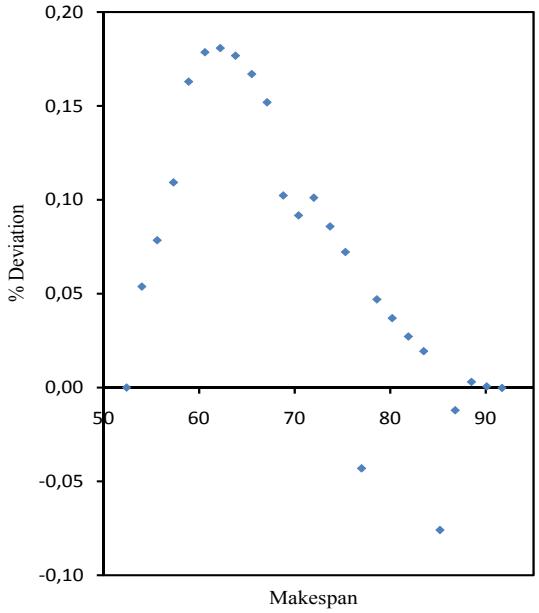
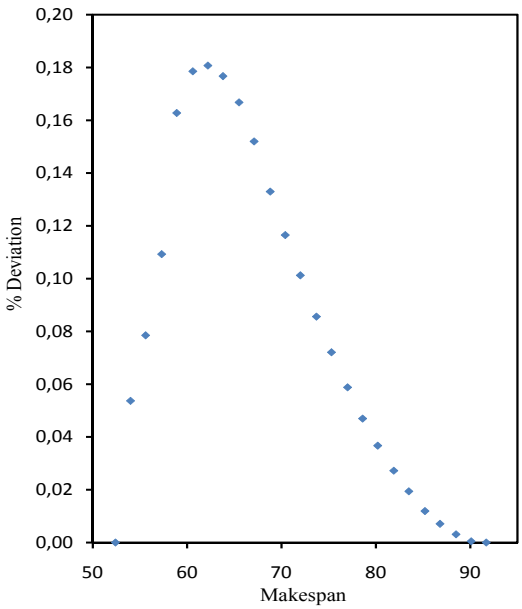


Figure 6.1: Efficient Frontier Generated by the Proposed Algorithm for One Replication



(a) Percent Deviations (DICOPT)



(b) Percent Deviations (BARON)

Figure 6.2: Deviations on Different Regions of the Efficient Frontier

either improves DICOPT or presents the same solution with DICOPT. The positive deviations are written in bold to highlight the cases which BARON improves DICOPT.

The points generated for this replication by the proposed approximation algorithm are plotted in Figure 6.1 to indicate the shape of the efficient frontier of our bicriteria problem. Percent deviations of the proposed algorithm from Model 1 solved by DICOPT and BARON are plotted in Figures 6.2a and 6.2b, respectively, to visualize the deviations. The first points (maximum makespan, minimum cost) and the last points (minimum makespan, maximum cost) on Figures 6.2a and 6.2b correspond to the points B and C on Figure 3.1, respectively.

6.3 Percent Deviations of Each Factor Combination

For the approximation algorithm, the minimum, average, and maximum values of the percent deviations from DICOPT are given for all factor combinations in Table 6.5. Since we took 5 replications for each factor combination and each replication has 25 efficient points, average deviation considers these $25 \cdot 5 = 125$ points. The worst performance of the approximation algorithm is at a factor combination n=20, A^1=H, A^2=H, and A^s=L with respect to mean deviation from both Model 1 and 2. On the other hand, the best performance is at a factor combination n=30, A^1=H, A^2=L, and A^s=L and n=30, A^1=L, A^2=L, and A^s=H again with respect to mean deviation from both Model 1 and 2, respectively. The maximum deviation of the algorithm from Model 2 appears at factor combination n=20, A^1=H, A^2=H, and A^s=L. Approximation algorithm improves the solution of Model 1 by 0.0595 at factor combination n=30, A^1=L, A^2=L, and A^s=H and the solution of Model 2 by 0.0438 at factor combination n=30, A^1=H, A^2=L, and A^s=L.

The minimum, average, and maximum values of the percent deviations of the approximation algorithm from Model 1 solved by BARON are also given for

Table 6.5: Percent Deviations (DICOPT) for Each Factor Combination

				Percent Deviations (DICOPT)					
				Model 1			Model 2		
n	A^1	A^2	A^s	Min	Mean	Max	Min	Mean	Max
20	L	L	L	-2.72	0.05	0.30	-0.65	0.01	0.33
20	H	L	L	-1.95	0.05	0.40	-0.70	-0.01	0.39
20	L	H	L	-0.94	0.07	0.37	-1.11	0.03	0.41
20	H	H	L	-1.62	0.11	0.42	-0.42	0.09	0.61
20	L	L	H	-1.09	-0.03	0.33	-1.24	-0.02	0.33
20	H	L	H	-2.12	0.05	0.55	-0.86	-0.03	0.55
20	L	H	H	-3.44	-0.06	0.35	-0.92	-0.03	0.34
20	H	H	H	-3.75	0.06	0.41	-0.83	0.06	0.42
30	L	L	L	-5.22	-0.17	0.13	-0.84	-0.03	0.14
30	H	L	L	-2.07	-0.03	0.38	-4.38	-0.20	0.25
30	L	H	L	-1.40	0.04	0.22	-2.55	-0.06	0.26
30	H	H	L	-1.71	-0.04	0.26	-0.57	0.02	0.32
30	L	L	H	-5.95	-0.19	0.13	-0.71	-0.05	0.13
30	H	L	H	-0.47	0.06	0.28	-1.14	-0.14	0.29
30	L	H	H	-3.54	-0.06	0.29	-1.42	-0.08	0.22
30	H	H	H	-4.27	-0.04	0.30	-0.55	0.04	0.3

all factor combinations in Table 6.6. Mean deviations indicate that approximation algorithm performs worst with a factor combination n=30, A^1=H, A^2=L, and A^s=L and best with a factor combination n=30, A^1=H, A^2=H, and A^s=L. Maximum and minimum deviations among all replications are 0.017 at replication with a factor combination n=30, A^1=H, A^2=L, and A^s=H and -0.0019 at replication with a factor combination n=30, A^1=H, A^2=L, and A^s=L, respectively. Negative deviations mean that our algorithm finds a better solution than BARON.

As mentioned before, the upper bounds of the processing times are affected by the factors A^1, A^2 and A^s. The assignment of flexible operations may also change with respect to the processing time values selected from different intervals which is expected to affect the efficiency of the proposed approximation algorithm. According to the results of percent deviations for each factor combination presented, we may claim that the performance of the proposed approximation algorithm is independent of the level of the factors. The algorithm generates good quality

Table 6.6: Percent Deviations (BARON) for Each Factor Combination

				Percent Deviations (BARON) Model 1		
n	A^1	A^2	A^s	Min	Mean	Max
20	L	L	L	0.00	0.10	0.33
20	H	L	L	-0.06	0.10	0.39
20	L	H	L	0.00	0.12	0.52
20	H	H	L	-0.05	0.09	0.33
20	L	L	H	-0.01	0.18	0.61
20	H	L	H	-0.02	0.10	0.35
20	L	H	H	0.00	0.10	0.55
20	H	H	H	0.00	0.14	0.42
30	L	L	L	-0.03	0.04	0.17
30	H	L	L	-0.19	0.32	1.00
30	L	H	L	-0.04	0.07	0.24
30	H	H	L	-0.08	0.04	0.13
30	L	L	H	0.00	0.10	0.32
30	H	L	H	-0.16	0.08	1.70
30	L	H	H	-0.08	0.07	0.29
30	H	H	H	0.00	0.09	0.30

solutions at both levels. Although the factor n mainly determines the size of the problem and CPU time to solve the problem, presented results indicates that the approximation algorithm performs better for high level of number of jobs.

The percent deviations of approximation algorithm from Model 1 and 2 solved by DICOPT and from Model 1 solved by BARON are arranged according to each factor being at levels L and H, which are presented in Tables 6.7 and 6.8. When mean deviations are considered in Table 6.7, approximation approximation algorithm performs better when A^1 and A^2 are at low levels and A^s is at high level with respect to both Model 1 and 2. The algorithm has the best performance when n=30. The algorithm outputs minimums of min, mean and max deviations at this level. According to the deviations presented in Table 6.8, the approximation algorithm generates better solutions at low levels of A^1 as in Table 6.7, but worse solutions at low levels of A^2 which is different than Table 6.7. It also shows the same performance for both levels of A^s. As in Table 6.7, the approximation algorithm seems to perform better for n=30 in Table 6.8.

Table 6.7: Percent Deviations (DICOPT) for Each Factor Level

		Percent Deviations (DICOPT)					
		Model 1			Model 2		
Factor	Level	Min	Mean	Max	Min	Mean	Max
A^1	L	-5.95	-0.04	0.37	-2.55	-0.03	0.41
	H	-4.27	0.03	0.55	-4.38	-0.02	0.61
A^2	L	-5.95	-0.026	0.55	-4.38	-0.06	0.55
	H	-4.27	0.01	0.42	-2.55	0.01	0.61
A^s	L	-5.22	0.01	0.42	-4.38	-0.02	0.61
	H	-5.95	-0.026	0.55	-1.42	-0.03	0.55
n	L	-3.75	0.038	0.55	-1.24	0.01	0.61
	H	-5.95	-0.054	0.38	-4.38	-0.06	0.32

Table 6.8: Percent Deviations (BARON) for Each Factor Level

		Percent Deviations (BARON) Model 1		
Factor	Level	Min	Mean	Max
A^1	L	-0.08	0.10	0.61
	H	-0.19	0.12	1.70
A^2	L	-0.19	0.13	1.70
	H	-0.08	0.09	0.55
A^s	L	-0.19	0.11	1.00
	H	-0.16	0.11	1.70
n	L	-0.06	0.12	0.61
	H	-0.19	0.10	1.70

Table 6.9: Algorithm vs. DICOPT Solutions for Each Factor Combination

					Model 1				Model 2			
n	A^1	A^2	A^s	Total	$= 0$	> 0	< 0	X	$= 0$	> 0	< 0	X
20	L	L	L	125	19	92	14	-	19	63	39	4
20	H	L	L	125	16	96	13	-	16	59	50	-
20	L	H	L	125	14	98	13	-	16	73	36	-
20	L	L	H	125	22	84	18	1	22	64	39	-
20	H	H	L	125	17	101	7	-	16	85	24	-
20	L	H	H	125	16	90	19	-	18	58	49	-
20	H	L	H	125	14	98	13	-	11	58	56	-
20	H	H	H	125	18	98	9	-	18	78	29	-
30	L	L	L	125	22	85	16	2	19	44	57	5
30	H	L	L	125	18	77	21	9	10	31	72	12
30	L	H	L	125	20	93	12	-	20	58	46	1
30	L	L	H	125	26	84	15	-	22	45	57	1
30	H	H	L	125	18	80	25	2	14	68	38	5
30	L	H	H	125	13	88	19	5	18	39	64	4
30	H	L	H	125	18	91	14	2	12	43	66	4
30	H	H	H	125	18	84	23	-	14	69	41	1

Table 6.10: Algorithm vs. BARON Solutions for Each Factor Combination

					Model 1			
n	A^1	A^2	A^s	Total	$= 0$	> 0	< 0	X
20	L	L	L	125	22	103	-	-
20	H	L	L	125	18	97	10	-
20	L	H	L	125	18	107	-	-
20	L	L	H	125	23	101	1	-
20	H	H	L	125	17	107	1	-
20	L	H	H	125	21	103	1	-
20	H	L	H	125	19	106	-	-
20	H	H	H	125	19	106	-	-
30	L	L	L	125	29	93	3	-
30	H	L	L	125	24	96	5	-
30	L	H	L	125	23	97	5	-
30	L	L	H	125	32	88	5	-
30	H	H	L	125	20	105	-	-
30	L	H	H	125	29	84	12	-
30	H	L	H	125	21	98	6	-
30	H	H	H	125	20	105	-	-

Tables 6.9 and 6.10 compare the solutions generated by the approximation algorithm with the solutions of mathematical Models 1 and 2 solved by DICOPT and mathematical Model 1 solved by BARON, respectively. "*Total*" represents the number of efficient points considered. Since we take 5 replications for each factor combination and we generated 25 efficient points for a replication, 125 efficient points exist for each factor combination. " $= 0$", " < 0", and " > 0" represent the number of efficient points generated by the mathematical model which presents the same manufacturing cost as the approximation algorithm, bigger, and smaller manufacturing costs than the approximation algorithm, respectively. "X" is the number of efficient points that the mathematical model cannot generate integer solutions but presents relaxed solution. As can be seen in the tables, although DICOPT can not generate integer solutions at some replications, no such points exist for BARON. According to Table 6.9, the number of these points increase for $n = 30$. The efficient points that belong to the case " < 0" is smaller for Model 1 than Model 2 solved by DICOPT. The number of efficient points of this case for Model 1 solved by BARON is also smaller than DICOPT.

Detailed tables including the results of each replication for all factor combinations are presented in Appendix A.

6.4 Overall Percent Deviations

In Table 6.11, the overall percent deviations of all factor combinations are presented. Mean deviations show that our proposed approximation algorithm improves both Model 1 and 2 solved using DICOPT. DICOPT outputs better results for Model 1 than Model 2 when mean deviations are considered. The mean deviation of the approximation algorithm from Model 1 solved by BARON is 0.0011, which indicates that the approximation algorithm generates good quality solutions.

Table 6.11: Overall Percent Deviations

Percent Deviations								
Model 1(DICOPT)			Model 2(DICOPT)			Model 1(BARON)		
Min	Mean	Max	Min	Mean	Max	Min	Mean	Max
-5.95	-0.003	0.55	-4.38	-0.025	0.61	-0.19	0.11	1.70

Table 6.12: CPU Times of Proposed Algorithm

				Proposed Algorithm		
n	A^1	A^2	A^s	Min (x10^{-2})	Mean (x10^{-2})	Max (x10^{-2})
20	L	L	L	0.44	0.72	0.91
20	H	L	L	0.29	0.72	1.07
20	L	H	L	0.80	1.12	1.88
20	L	L	H	0.44	0.59	0.85
20	H	H	L	0.62	1.00	1.43
20	L	H	H	0.50	0.72	1.25
20	H	L	H	0.50	0.80	1.00
20	H	H	H	0.62	1.23	2.93
30	L	L	L	0.44	0.79	1.58
30	H	L	L	0.68	1.15	1.87
30	L	H	L	0.62	1.05	1.87
30	L	L	H	0.62	0.76	0.87
30	H	H	L	0.81	1.32	1.87
30	L	H	H	0.87	1.57	3.00
30	H	L	H	0.62	1.15	2.31
30	H	H	H	1.25	1.81	2.56

6.5 CPU Times

CPU results of all factor combinations are presented for the approximation algorithm, Model 1 and 2 solved by DICOPT in Tables 6.12 and 6.13, respectively.

Due to CPU restrictions, even for 30 jobs processed, it is not possible to run the mathematical model till the end by DICOPT. Therefore, we run the DICOPT with a time limit of 1000 seconds, which is the default. This same limit is also used for BARON. BARON solver did not stop before the time limit exceeded for all replications, which means every replication has a CPU time of 1000 seconds. Hence, we do not present a CPU time requirement table for BARON.

Table 6.13: CPU Times of Mathematical Models (DICOPT)

				GAMS (DICOPT)					
				Model 1			Model 2		
n	A^1	A^2	A^s	Min	Mean	Max	Min	Mean	Max
20	L	L	L	0.16	10.6	1000	0.19	6.11	115
20	H	L	L	0.02	7.02	651	0.12	5.69	87
20	L	H	L	0.01	26.7	1000	0.21	4.97	86
20	L	L	H	0.16	3.56	74	0.18	2.86	59
20	H	H	L	0.02	9.94	805	0.01	5.69	113
20	L	H	H	0.08	6.96	594	0.16	6.09	235
20	H	L	H	0.06	9.18	748	0.14	2.6	33
20	H	H	H	0.16	2.85	79	0.2	6.04	150
30	L	L	L	0.14	66.4	1000*	0.16	82.4	1000*
30	H	L	L	0.14	90.14	1000*	0.2	129.71	1000*
30	L	H	L	0.14	49.41	1000	0.17	88.72	1000*
30	L	L	H	0.11	74.17	1000	0.2	24.92	1000*
30	H	H	L	0.02	65.2	1000*	0.19	81.59	1000*
30	L	H	H	0.09	57.37	1000*	0.16	40.76	1000*
30	H	L	H	0.2	81.24	1000*	0.23	72.62	1000*
30	H	H	H	0.02	25.04	1000	0.22	40.54	1000*

The results indicate that the proposed algorithm can generate a relatively good solution in a very small CPU time which is much smaller than DICOPT and BARON can generate. As the number of jobs increases the increase in the CPU time is very small for the algorithm. As can be seen in Table 6.13 DICOPT can not generate any integer solutions in 1000 seconds for some of the replications, represented as 1000*. Also for some of the replications, DICOPT stops due to 1000 seconds CPU restriction and reports the best integer solution. Solver BARON can generate integer solutions for all replications unlike DICOPT. According to the analysis in Sections 6.3 and 6.4 BARON generates better solutions than DICOPT. However, the CPU requirement of BARON is much higher as compared with DICOPT. On the contrary, presented approximation algorithm is advantageous to both of these in terms of CPU times.

Overall CPU time requirements are also presented in Tables 6.14 and 6.15. The results show that CPU requirement of approximation algorithm is the smallest and BARON solver is the largest.

Table 6.14: Overall CPU Times of Approximation Algorithm

Min	Mean (x10^{-2})	Max
0.29	1.03	2.93

Table 6.15: Overall CPU Times of Mathematical Models

Model 1(DICOPT)			Model 2(DICOPT)			Model 1(BARON)		
Min	Mean	Max	Min	Mean	Max	Min	Mean	Max
0.01	36.61	1000*	0.01	37.58	1000*	1000	1000	1000

6.6 Summary

In this chapter, we performed a computational study. Percent deviations of approximation algorithm from the mathematical models and required CPU times are presented to test the performance of our proposed approximation algorithm. The results indicate that the approximation algorithm finds high quality solutions in small CPU times.

Chapter 7

Conclusion and Future Work

This thesis considers a scheduling problem with controllable processing times and flexible operations. Controllability of processing times and flexibility in manufacturing systems subjects has been widely studied individually in scheduling literature. However, this is the first study that considers both assignment of flexible operations and controllability through a bicriteria objective at the same time.

In the following section, we mention the contributions of the current study and in section 7.2 we suggest possible extensions and future research directions.

7.1 Contributions

In this thesis, we studied a nonpreemptive two-machine flowshop environment in which identical jobs are processed. Each job has three operations, one of which is a flexible operation. Hence, the flexible operations should be assigned to one of the machines in order to minimize the makespan. In this study, the processing times of operations are assumed to be controllable, so the processing times of operations should also be determined. We considered a bicriteria objective of minimizing the manufacturing cost and the makespan. Since these two objectives

cannot be minimized at the sane time, we determined a set of efficient discrete points of makespan and manufacturing cost objectives.

We proposed a mixed integer nonlinear program (Model 1) to solve the problem. One of the alternatives to solve MINLP formulations is to call DICOPT solver of GAMS. Because of the underlying algorithm in DICOPT, linearization of the constraints is thought to be helpful. Hence, the constraints of Model 1 are linearized and presented as an alternative model as Model 2. Models 1 and 2 could not be decided to be better than the other because at each iteration there exists efficient points at which the models improve each other. Therefore, we present runs of both models for DICOPT. Another alternative for generating good quality solutions is to use BARON solver. Since overall percent deviations of approximation algorithm from Model 1 and 2, solved by DICOPT, indicates that Model 1 performs better than Model 2, BARON runs are only taken for Model 1. Because of the CPU time restriction, it is not possible to run the mathematical models by both solvers till the end. Hence, we run the models with a time limit of 1000 seconds.

Benefits of controllable processing times was presented via examples. Idle times can be eliminated by means of controllable processing times to achieve the same makespan values as fixed processing time production case at a lower cost. The first example eliminates idle times by expanding the processing time values. However, controllability of the processing times also affects the assignment of flexible operations as well as the processing time values to eliminate idle times. The second example was given to highlight this idea.

We provided some theoretical optimality properties of the problem which are used in the development of approximation algorithm. We also presented some algorithmic rules to point out the application of presented lemmas in the proposed approximation algorithm. The first rule eliminates dominated solutions to get nondominated ones, and the second rule determines the next efficient point on the frontier given a nondominated solution.

Since the mathematical programming formulation may not be efficient in terms of CPU time, we proposed an approximation algorithm. It is coded in

C++ language and compiled with a GNU compiler. The algorithm generates a set of approximately efficient solutions that can be presented to the decision maker quickly.

We presented computational results for the proposed models and approximation algorithm. We observed that mathematical model solved by BARON generates better results than DICOPT, but it needs more CPU time to generate solutions. Although DICOPT can not generate integer solutions at some replications in 1000 seconds, BARON can generate integer solutions for all replications in this time limit. Percent deviations of the approximation algorithm from the mathematical models and CPU times were given. Percent deviations indicate that approximation algorithm generates good quality solutions in a very small CPU time as compared with the mathematical models. The algorithm improves the solutions of mathematical formulation solved by DICOPT on the average. The deviations from the mathematical formulation solved by BARON are reasonable.

In this section, we presented the contributions of our study. In the next section, we will offer some future research directions about the problem.

7.2 Future Research Directions

This study can be extended by increasing the number of machines and the number of operations. As the number of machines and operations are increased, the assignment of flexible operations to the machines will be a harder decision. In this study, each job has only one flexible operation. The number of flexible operations can also be increased in parallel with the number of machines and operations.

Another extension of this thesis is to study processing nonidentical jobs in the same scheduling environment. In our case, we have three different cost functions for three operations and they are the same for all identical jobs. In processing nonidentical jobs case, the cost functions of operations of all jobs could be different. In that case, the outcomes of Corollary 1 may not hold, so the number

of processing time variables may increase. The mathematical formulation and proposed approximation algorithm should be changed accordingly.

This problem can be studied for other scheduling objectives such as total completion time. The scheduling environment may also be different. Jobshop and openshop environments may be studied instead of flowshop.

7.3 Summary

In this chapter, we reviewed our study and discussed the contributions of this study to the scheduling literature. We also suggested some future research directions as an extension to this study.

Bibliography

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

[2] Alidaee, B. and Ahmadian, A. Two parallel machine sequencing problems involving controllable job processing times. *European Journal of Operational Research* 70, 335-341, 1993.

[3] Allaoui, H., Lamouri, S., Artiba, A., and Aghezzaf, E. Simultaneously scheduling n jobs and the preventive maintenance on the two-machine flow shop to minimize the makespan. *International Journal of Production Economics* 112, 161-167, 2008.

[4] Becker, C. and Scholl, A. A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research* 168, 694-715, 2006.

[5] Blazewicz, J., Breit, J., Formanowicz, P., Kubiak, W., and Schmidt, G. Heuristic algorithms for the two-machine flowshop with limited machine availability. *Omega* 29(6), 599-608, 2001.

[6] Boysen, N., Fliedner, M., and Scholl, A. A classification of assembly line balancing problems. *European Journal of Operational Research* 183, 674-693, 2007.

[7] Brucker, P., Heitmann, S., and Hurink, J. Flow-shop problems with intermediate buffers. *OR Spectrum* 25, 549-574, 2003.

[8] Burdett, R. L. and Kozan, E. Sequencing and scheduling in flowshops with task redistribution. *Journal of the Operational Research Society* 52, 1379-1389, 2001.

[9] Cheng, T.C.E., Oguz, C. and Qi, X.D. Due-date assignment and single machine scheduling with compressible processing times. *International Journal of Production Economics* 43, 29-35, 1996.

[10] Cheng, E. T. C. and Wang, G. An improved heuristic for two-machine flowshop scheduling with an availability constraint. *Operations Research Letters* 26, 223-229, 2000.

[11] Crama, Y., Kats, V., van de Klundert, J., and Levner, E. Cyclic scheduling in robotic flowshops. *Annals of Operations Research* 96, 97-124, 2000.

[12] Crama, Y. and Gultekin, H. Throughput optimization in two-machine flowshops with flexible operations. *Journal of Scheduling* 13, 227-243, 2010.

[13] Croce, F. D., Narayan, V., and Tadei, R. The two-machine total completion time flow shop problem. *European Journal of Operational Research* 90, 227-237, 1996.

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

[15] Daniels, R. L. A multi-objective approach to resource allocation in single machine scheduling. *European Journal of Operational Research* 48, 226-241, 1990.

[16] Daniels, R. L. and Mazzola, J. B. Flow-shop scheduling with resource flexibility. *Operations Research* 42, 504-522, 1994.

[17] Daniels, R. L., Mazzola, J. B., and Shi, D. Flow shop scheduling with partial resource flexibility. *Management Science* 50(5), 658-669, 2004.

[18] Dobson, G. and Karmarkar, U. S. Simultaneous resource scheduling to minimize weighted flow times. *Operations Research* 37, 592-600, 1989.

[19] Dutta, S. K. and Cunningham, A. A. Sequencing two-machine flow-shops with finite intermediate storage. *Management Science* 21, 989-996, 1975.

[20] Glass, C.A., Potts, C.N., and Strusevich, V.A. Scheduling batches with sequential job processing for two-machine flow and open shops. *Informs Journal on Computing* 13, 120-137, 2001.

[21] Gultekin, H., Akturk, M. S., and Karasan, O. E. Robotic cell scheduling with operational flexibility. *Discrete Applied Mathematics* 145(3), 334-348, 2005.

[22] Gultekin, H., Akturk, M. S., and Karasan, O. E. Cyclic scheduling of a a 2-machine robotic cell with tooling constraints. *European Journal of Operational Research* 174, 777-796, 2006.

[23] Gultekin, H., Akturk, M. S., and Karasan, O. E. Bicriteria robotic operation allocation in a flexible manufacturing cell. *Computers and Operations Research* 37, 779-789, 2010.

[24] Gultekin, H., Akturk, M. S., and Karasan, O. E. Bicriteria robotic cell scheduling. *Journal of Scheduling* 11, 457-473, 2008.

[25] Guo, Z. X., Wong, W. K., Leung, S. Y. S., Fan, J. T., and Chan, S. F. A genetic-algorithm-based optimization model for scheduling flexible assembly lines. *Int. J. Adv. Manuf. Technol.* 36, 156-168, 2008.

[26] Guo, Z. X., Wong, W. K., Leung, S. Y. S., and Fan, J. T. Intelligent production control decision support system for flexible assembly lines. *Int. Expert Systems with Applications* 36, 4268-4277, 2009.

[27] Gupta, J. N. D. and Darrow, W. P. The two-machine sequence dependent flowshop scheduling problem. *European Journal of Operational Research* 24(3), 439-446, 1986.

[28] Gupta, J. N. D., Koulamas, C. P., Kyparisis, G. J., Potts, C. N., and Strusevich, V. A. Scheduling three-operation jobs in a two machine flow shop to minimize makespan. *Annals of Operations Research* 129, 171-185, 2004.

[29] Gupta, J. N. D. and Stafford, E. F. Jr. Flowshop scheduling research after five decades. *European Journal of Operational Research* 169, 699-711, 2006.

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

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

[32] Hall, N. G. and Sriskandarajah, C. A Survey of Machine Scheduling Problems with Blocking and No-Wait in Process. *Operations Research* 44, 510-525, 1996.

[33] Hejazi, S. R. and Saghafianz, S. Flowshop-scheduling problems with makespan criterion: a review. *International Journal of Production Research* 43, 2895-2929, 2005.

[34] Hitomi, K. Manufacturing Systems Engineering: A Unified Approach to Manufacturing Technology and Production. Management. *Taylor and Francis, London* 1979.

[35] Hoogeveen, H. and Woeginger, G.J. Some comments on sequencing with controllable processing times. *Computing* 68, 181-192, 2002.

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

[37] Johnson, S. M. Optimal two- and three-stage production schedules with setup times included. *Nav. Res. Logist. Quart.* 1, 61-68,1954.

[38] Garey, M. R., Johnson, D. S., Sethi, R. The complexity of flowshop and jobshop scheduling. *J. Mathematics of Operations Research* 1, 117-129, 1976.

[39] Karabati, S., Kouvelis, P. Flow-line scheduling problem with controllable processing times. *IIE Transactions* 29, 1-14 , 1997.

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

[41] Kohler, W. H. and Steiglitz, K. Exact, approximate and guaranteed accuracy algorithms for the flowshop problem n/2/F/F. *J. Assoc. Comput. Mach.* 22, 106-114, 1975.

[42] Lamond, B.F., Sodhi, M.S. Using tool life models to minimize processing time on a flexible machine. *IIE Transactions* 29, 611-621, 1997.

[43] Lawrence, S. R. and Morton, T. E. Constrained multi-project scheduling with tardy costs: Comparing myopic, bottleneck, and resource pricing heuristics. *European Journal of Operational Research* 64, 168-187, 1993.

[44] Lee C.-Y. Two-machine flowshop scheduling with availability constraints. *European Journal of Operational Research* 114, 420-429, 1999.

[45] Leyvand, Y., Shabtay, D., Steiner, G., and Yedidsion, L. Just-in-time scheduling with controllable processing times on parallel machines. *Journal of Combinatorial Optimization* 19, 347-368, 2010.

[46] Liu, B., Wang, L., and Jina Y.-H. An effective hybrid PSO-based algorithm for flow shop scheduling with limited buffers. *Computers and Operations Research* 35, 2791-2806, 2008.

[47] Mastrolilli, M. Notes on max flow time minimization with controllable processing times. *Computing* 71, 375-386, 2003.

[48] Moder, J. J., Phillips C. C., and Davis, E. W. Profect management with CPM and precedence diagramming. *Van Nostrand Reinhold, New York* 1970.

[49] Nawaz, M., Enscore Jr, E. E., and Ham, I. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega* 11(1), 91-95, 1983.

[50] Ng, C.T.D., Cheng, T.C.E., Kovalyov, M.Y., and Lam, S.S. Single machine scheduling with a variable common due date and resource-dependent processing times. *Computers and Operations Research* 30, 1173-1185, 2003.

[51] Ng, C.T., Wang, J.-B., Cheng, T.C.E., and Liu, L.L. A branch-and-bound algorithm for solving a two-machine flow shop problem with deteriorating jobs. *Computers and Operations Research* 37, 83-90, 2010.

[52] Nowicki, E., Zdrzalka S. A two-machine flow shop scheduling problem with controllable job processing times. *European Journal of Operational Research* 34, 208-220, 1988.

[53] Pan, J. C.-H., Chen, J.-S., and Chao, C.-M. Minimizing tardiness in a two-machine flow-shop. *Computers and Operations Research* 29, 869-885, 2002.

[54] Panwalkar, S.S. and Rajagopalan, R. Single machine sequencing with controllable processing times. *European Journal of Operational Research* 59, 298-302, 1992.

[55] Rajendran, C. A no-wait flowshop scheduling heuristic to minimize makespan. *The Journal of the Operational Research Society* 45, 472-478, 1994.

[56] Sarin, S. and Lefoka, M. Scheduling heuristic for the n-job m-machine flow shop. *Omega* 21(2), 229-234, 1993.

[57] Sen, T., Dileepan, P., and Gupia, J. N. D. The two-machine flowshop scheduling problem with total tardiness. *Computers and Operations Research* 16(4), 333-340, 1989.

[58] Shabtay, D. and Steiner, G. The no-wait two-machine flow shop scheduling problem with convex resource-dependent processing times. *IIE Transactions* 39, 539-557, 2007.

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

[60] Slowinski, R. and Weglarz, J. Solving the general project scheduling problem with multiple constrained resources by mathematical programming. *Lect. Notes Cont. and Infor. Syst.* 7, 278-289, 1978.

[61] Slowinski, R. Multiobjective network scheduling with efficient use of renewable and nonrenewable resources. *European Journal of Operational Research* 7, 265-273, 1981.

[62] Smutnicki, C. A two-machine permutation flow shop scheduling problem with buffers. *OR Spectrum* 20, 229-235, 1998.

[63] Sodhi, M. S., Agnetis, A., and Askin, R. G. Tool addition strategies for flexible manufacturing systems. *International Journal of Flexible Manufacturing Systems* 6, 287-310, 1994.

[64] Sodhi, M.S., Lamond, B.F., Gautier, A., Noel, M. Heuristics for determining economic processing rates in a flexible manufacturing system. *European Journal of Operational Research* 129, 105-115, 2001.

[65] Stecke, K. E. Formulation and solution of nonlinear integer production planning problems for flexible manufacturing systems. *Management Science* 29, 273-288, 1983.

[66] T'kindt, V., Monmarch, N., Tercinet, F., and Lagt, D. An ant colony optimization algorithm to solve a 2-machine bicriteria flowshop scheduling problem. *European Journal of Operational Research* 142, 250-257, 2002.

[67] T′kindt, V., Billaut, J. C. Multicriteria Scheduling: Theory, Models and Algorithms. *Second ed. Springer, Berlin* 2006.

[68] Trick, M.A. Scheduling multiple variable-speed machines. *Operations Research* 42, 234-248, 1994.

[69] Wan, G., Vakati, S.R., Leung, J.Y.T., and Pinedo, M. Scheduling two agents with controllable processing times. *European Journal of Operational Research* 205, 528-539, 2010.

[70] Wang, L., Zhang, L., and Zheng, D.-Z. An effective hybrid genetic algorithm for flow shop scheduling with limited buffers. *Computers and Operations Research* 33, 2960-2971, 2006.

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

[72] Weglarz, J., Blazewicz, J., Cellary, J., and Slowinski, R. An automatic revised simplex method for constrained resource network scheduling. *ACM Trans. Math. Soft.* 3, 295-300, 1977.

[73] Weglarz, J. Control in resource allocation systems. *Found. Cont. Eng.* 5, 159-180, 1980.

[74] Widmer M. and Hertz A. A new heuristic method for the flow shop sequencing problem. *European Journal of Operational Research* 41, 186193, 1989.

[75] Wiest, J. and Levy, F. A management guide to PERT/CPM with GERT/PDM/DCPM and other networks. *Prentice-Hall, Englewood Cliffs, New Jersey* 1977.

[76] Van Wassenhove, L.N., and Baker, K.R. A bicriterion approach to time/cost trade-offs in sequencing. *European Journal of Operational Research* 11, 48-52, 1982.

[77] Vickson, R.G. Two single-machine sequencing problems involving controllable job processing times. *AIEE Transactions* 12, 258-262 , 1980.

[78] Zdrzalka, S. Scheduling jobs on a single machine with relese dates, delivery times and controllable processing times: worst-case analysis. *Operations Research Letters* 10, 519-524, 1991.

Appendix A

Detailed Computational Results

Table A.1: Algorithm vs. DICOPT Solutions of Model 1

Factors				Algorithm vs. Model 1 (DICOPT)							
A^1	A^2	A^s	n	Rep	$= 0$	> 0	Avg (x10^{-2})	Case Avg (x10^{-2})	< 0	Avg (x10^{-2})	Case Avg (x10^{-2})
			20	1	4	19	0.09		2	-1.48	
			20	2	5	14	0.14		6	-0.15	
0	0	0	20	3	3	19	0.10	0.11	3	-0.10	-0.35
			20	4	4	19	0.11		2	-0.34	
			20	5	3	21	0.14		1	-0.02	
			20	1	2	21	0.17		2	-0.96	
			20	2	5	19	0.12		1	-1.30	
1	0	0	20	3	3	19	0.13	0.14	3	-0.66	-0.57
			20	4	2	19	0.16		4	-0.52	
			20	5	4	18	0.10		3	-0.04	
			20	1	3	19	0.10		3	-0.51	
			20	2	3	19	0.15		3	-0.33	
0	1	0	20	3	3	19	0.16	0.13	3	-0.06	-0.23
			20	4	2	21	0.12		2	-0.08	
			20	5	3	20	0.11		2	-0.04	
			20	1	6	17	0.14		2	-0.18	
			20	2	3	19	0.09		3	-2.61	
0	0	1	20	3	5	17	0.13	0.11	2	-0.76	-0.71
			20	4	4	14	0.07		7	-0.16	
			20	5	4	17	0.09		4	-0.49	
			20	1	4	20	0.12		1	-0.02	
			20	2	3	20	0.17		2	-0.61	
1	1	0	20	3	3	22	0.17	0.16	0	-	-0.42
			20	4	4	18	0.15		3	-0.03	
			20	5	3	21	0.19		1	-1.62	
			20	1	3	17	0.20		5	-0.85	
			20	2	4	18	0.09		3	-1.14	
1	0	1	20	3	2	16	0.09	0.13	7	-0.96	-1.02
			20	4	3	19	0.15		3	-0.91	
			20	5	4	20	0.10		1	-2.27	
			20	1	3	20	0.14		2	-1.14	
			20	2	2	19	0.14		4	-0.31	
0	1	1	20	3	4	18	0.10	0.13	3	-0.15	-0.46
			20	4	1	20	0.14		4	-0.50	
			20	5	4	21	0.11		0	-	
			20	1	4	20	0.19		1	-3.75	
			20	2	2	21	0.18		2	-0.40	
1	1	1	20	3	4	18	0.18	0.17	3	-1.29	-1.10
			20	4	4	18	0.16		3	-0.23	
			20	5	4	21	0.13		0	-	

Factors				Algorithm vs. Model 1 (DICOPT)							
A^1	A^2	A^s	n	Rep	$= 0$	> 0	Avg (x10^{-2})	Case Avg (x10^{-2})	< 0	Avg (x10^{-2})	Case Avg (x10^{-2})
			30	1	4	18	0.05		3	-2.68	
			30	2	7	16	0.06		2	-0.05	
0	0	0	30	3	3	18	0.08	0.06	3	-3.14	-1.58
			30	4	4	16	0.05		5	-1.20	
			30	5	4	17	0.04		3	-0.56	
			30	1	4	19	0.09		2	-0.56	
			30	2	3	14	0.10		5	-0.45	
1	0	0	30	3	2	15	0.07	0.09	5	-0.22	-0.51
			30	4	5	13	0.08		4	-0.72	
			30	5	4	16	0.09		5	-0.66	
			30	1	5	17	0.10		3	-0.50	
			30	2	4	20	0.12		1	-0.59	
0	1	0	30	3	4	20	0.08	0.09	1	-0.28	-0.30
			30	4	3	19	0.10		3	-0.30	
			30	5	4	17	0.07		4	-0.09	
			30	1	4	18	0.06		3	-0.27	
			30	2	6	14	0.09		5	-2.45	
0	0	1	30	3	5	18	0.05	0.06	2	-3.06	-1.30
			30	4	6	17	0.04		2	-0.03	
			30	5	5	17	0.07		3	-0.05	
			30	1	3	14	0.07		8	-0.32	
			30	2	4	20	0.12		0	-	
1	1	0	30	3	4	18	0.10	0.10	2	-0.42	-0.51
			30	4	3	16	0.11		6	-0.23	
			30	5	4	12	0.05		9	-0.88	
			30	1	2	18	0.07		5	-0.87	
			30	2	2	18	0.09		3	-1.19	
1	0	1	30	3	4	19	0.09	0.09	2	-1.22	-0.84
			30	4	3	15	0.10		5	-0.63	
			30	5	2	18	0.11		4	-0.62	
			30	1	4	18	0.10		3	-0.11	
			30	2	2	20	0.11		1	-0.03	
0	1	1	30	3	5	16	0.06	0.10	4	-0.12	-0.10
			30	4	4	19	0.15		2	-0.25	
			30	5	3	18	0.08		4	-0.01	
			30	1	3	20	0.11		2	-1.51	
			30	2	3	16	0.15		6	-0.98	
1	1	1	30	3	4	17	0.13	0.12	4	-0.40	-0.66
			30	4	5	15	0.11		5	-0.79	
			30	5	3	16	0.11		6	-0.13	

Table A.2: Algorithm vs. DICOPT Solutions of Model 2

Factors				Algorithm vs. Model 2 (Dicopt)							
A^1	A^2	A^s	n	Rep	$=0$	>0	Avg (x10^{-2})	Case Avg (x10^{-2})	<0	Avg (x10^{-2})	Case Avg (x10^{-2})
			20	1	4	9	0.10		12	-0.17	
			20	2	4	13	0.13		8	-0.18	
0	0	0	20	3	4	17	0.11	0.11	4	-0.06	-0.14
			20	4	3	13	0.10		9	-0.12	
			20	5	4	11	0.14		10	-0.10	
			20	1	3	17	0.16		5	-0.11	
			20	2	5	11	0.13		9	-0.25	
1	0	0	20	3	3	8	0.16	0.14	14	-0.12	-0.19
			20	4	2	14	0.15		9	-0.16	
			20	5	3	9	0.08		13	-0.26	
			20	1	5	16	0.11		4	-0.07	
			20	2	5	12	0.20		8	-0.19	
0	1	0	20	3	1	17	0.14	0.14	7	-0.22	-0.17
			20	4	2	16	0.15		7	-0.14	
			20	5	3	12	0.08		10	-0.19	
			20	1	6	12	0.13		7	-0.40	
			20	2	4	11	0.08		10	-0.35	
0	0	1	20	3	3	17	0.18	0.11	5	-0.41	-0.25
			20	4	4	10	0.06		11	-0.10	
			20	5	5	14	0.07		6	-0.07	
			20	1	4	13	0.173		8	-0.159	
			20	2	4	19	0.163		2	-0.082	
1	1	0	20	3	2	21	0.143	0.169	2	-0.054	-0.125
			20	4	4	16	0.195		5	-0.148	
			20	5	2	16	0.182		7	-0.103	
			20	1	3	14	0.18		8	-0.07	
			20	2	4	12	0.09		9	-0.25	
1	0	1	20	3	4	4	0.09	0.12	17	-0.23	-0.22
			20	4	4	17	0.13		4	-0.09	
			20	5	3	11	0.11		11	-0.35	
			20	1	2	14	0.16		9	-0.24	
			20	2	2	11	0.17		12	-0.08	
0	1	1	20	3	3	12	0.11	0.14	10	-0.16	-0.21
			20	4	2	12	0.16		11	-0.43	
			20	5	2	9	0.11		14	-0.18	
			20	1	5	16	0.21		4	-0.17	
			20	2	3	15	0.21		7	-0.25	
1	1	1	20	3	3	16	0.17	0.18	6	-0.15	-0.21
			20	4	3	17	0.16		5	-0.04	
			20	5	4	14	0.14		7	-0.38	

Factors				Algorithm vs. Model 2 (Dicopt)							
A^1	A^2	A^s	n	Rep	$= 0$	> 0	Avg (x10^{-2})	Case Avg (x10^{-2})	< 0	Avg (x10^{-2})	Case Avg (x10^{-2})
			30	1	6	6	0.05		12	-0.14	
			30	2	5	11	0.05		9	-0.11	
0	0	0	30	3	2	11	0.06	0.05	11	-0.09	-0.11
			30	4	3	4	0.04		16	-0.07	
			30	5	3	12	0.04		9	-0.16	
			30	1	2	6	0.07		16	-0.73	
			30	2	2	8	0.11		12	-0.18	
1	0	0	30	3	2	5	0.03	0.08	15	-0.42	-0.34
			30	4	1	7	0.09		14	-0.11	
			30	5	3	5	0.10		15	-0.17	
			30	1	6	12	0.10		7	-0.49	
			30	2	3	15	0.12		6	-0.54	
0	1	0	30	3	4	6	0.09	0.11	15	-0.22	-0.29
			30	4	3	14	0.12		8	-0.22	
			30	5	4	11	0.09		10	-0.15	
			30	1	4	10	0.06		11	-0.19	
			30	2	5	12	0.08		8	-0.16	
0	0	1	30	3	4	7	0.05	0.06	14	-0.14	-0.16
			30	4	5	7	0.06		13	-0.16	
			30	5	4	9	0.05		11	-0.14	
			30	1	1	17	0.08		6	-0.12	
			30	2	3	14	0.13		7	-0.07	
1	1	0	30	3	4	14	0.10	0.10	6	-0.14	-0.12
			30	4	2	16	0.10		7	-0.07	
			30	5	4	7	0.10		12	-0.17	
			30	1	4	7	0.06		14	-0.21	
			30	2	4	10	0.06		10	-0.12	
1	0	1	30	3	3	11	0.07	0.07	11	-0.11	-0.21
			30	4	4	6	0.06		13	-0.30	
			30	5	3	5	0.14		16	-0.27	
			30	1	2	9	0.08		14	-0.30	
			30	2	1	9	0.16		13	-0.19	
0	1	1	30	3	4	4	0.07	0.12	15	-0.46	-0.33
			30	4	1	12	0.16		12	-0.26	
			30	5	4	9	0.08		12	-0.40	
			30	1	1	14	0.11		9	-0.04	
			30	2	3	15	0.13		7	-0.14	
1	1	1	30	3	2	15	0.12	0.12	8	-0.08	-0.09
			30	4	4	11	0.11		10	-0.09	
			30	5	4	14	0.10		7	-0.09	

Table A.3: Algorithm vs. BARON Solutions of Model 1

Factors				Algorithm vs. Model 1 (BARON)							
A^1	A^2	A^s	n	Rep	$= 0$	> 0	Avg (x10^{-2})	Case Avg (x10^{-2})	< 0	Avg (x10^{-2})	Case Avg (x10^{-2})
			20	1	4	21	0.10		0	-	
			20	2	5	20	0.14		0	-	
0	0	0	20	3	4	21	0.11	0.12	0	-	-
			20	4	5	20	0.12		0	-	
			20	5	4	21	0.14		0	-	
			20	1	1	14	0.18		10	-0.04	
			20	2	5	20	0.13		0	-	
1	0	0	20	3	4	21	0.12	0.13	0	-	-0.04
			20	4	4	21	0.16		0	-	
			20	5	4	21	0.10		0	-	
			20	1	4	21	0.11		0	-	
			20	2	4	21	0.19		0	-	
0	1	0	20	3	4	21	0.16	0.14	0	-	-
			20	4	2	23	0.12		0	-	
			20	5	4	21	0.11		0	-	
			20	1	5	20	0.13		0	-	
			20	2	5	20	0.09		0	-	
0	0	1	20	3	4	21	0.15	0.11	0	-	-0.05
			20	4	4	20	0.09		1	-0.05	
			20	5	5	20	0.09		0	-	
			20	1	4	21	0.19		0	-	
			20	2	4	21	0.20		0	-	
1	1	0	20	3	3	22	0.18	0.20	0	-	-0.01
			20	4	4	21	0.26		0	-	
			20	5	2	22	0.18		1	-0.01	
			20	1	3	22	0.17		0	-	
			20	2	4	20	0.09		1	-0.02	
1	0	1	20	3	5	20	0.10	0.12	0	-	-0.02
			20	4	4	21	0.14		0	-	
			20	5	5	20	0.10		0	-	
			20	1	4	21	0.14		0	-	
			20	2	3	22	0.14		0	-	
0	1	1	20	3	5	20	0.11	0.13	0	-	-
			20	4	3	22	0.14		0	-	
			20	5	4	21	0.10		0	-	
			20	1	5	20	0.20		0	-	
			20	2	4	21	0.18		0	-	
1	1	1	20	3	3	22	0.16	0.17	0	-	-
			20	4	4	21	0.18		0	-	
			20	5	3	22	0.12		0	-	

Factors				Algorithm vs. Model 1 (BARON)							
A^1	A^2	A^s	n	Rep	$= 0$	> 0	Avg (x10^{-2})	Case Avg (x10^{-2})	< 0	Avg (x10^{-2})	Case Avg (x10^{-2})
			30	1	8	17	0.05		0	-	
			30	2	6	19	0.05		0	-	
0	0	0	30	3	5	18	0.08	0.05	2	-0.02	-0.02
			30	4	5	20	0.04		0	-	
			30	5	5	19	0.04		1	-0.02	
			30	1	5	20	0.06		0	-	
			30	2	5	19	0.11		1	-0.02	
1	0	0	30	3	4	21	0.12	0.10	0	-	-0.02
			30	4	6	17	0.07		2	-0.10	
			30	5	4	19	0.12		2	-0.02	
			30	1	5	19	0.09		1	-0.04	
			30	2	5	20	0.11		0	-	
0	1	0	30	3	5	20	0.08	0.09	0	-	-0.02
			30	4	4	21	0.10		0	-	
			30	5	4	17	0.09		4	-0.01	
			30	1	5	18	0.07		2	-0.05	
			30	2	6	18	0.07		1	-0.05	
0	0	1	30	3	8	17	0.03	0.06	0	-	-0.04
			30	4	8	17	0.05		0	-	
			30	5	5	18	0.07		2	-0.02	
			30	1	4	21	0.11		0	-	
			30	2	4	21	0.12		0	-	
1	1	0	30	3	3	22	0.13	0.12	0	-	-
			30	4	4	21	0.12		0	-	
			30	5	5	20	0.10		0	-	
			30	1	8	14	0.07		3	-0.05	
			30	2	4	18	0.19		3	-0.03	
1	0	1	30	3	6	17	0.09	0.12	2	-0.07	-0.06
			30	4	6	17	0.13		2	-0.10	
			30	5	5	18	0.10		2	-0.06	
			30	1	4	19	0.09		2	-0.02	
			30	2	4	20	0.10		1	-0.08	
0	1	1	30	3	7	17	0.05	0.09	1	-0.02	-0.04
			30	4	2	21	0.13		2	-0.04	
			30	5	4	21	0.06		0	-	
			30	1	4	21	0.10		0	-	
			30	2	4	21	0.12		0	-	
1	1	1	30	3	4	21	0.12	0.12	0	-	-
			30	4	4	21	0.10		0	-	
			30	5	4	21	0.10		0	-	

Table A.4: DICOPT vs. BARON Solutions of Model 1

Factors				DICOPT vs BARON (Model 1)							
A^1	A^2	A^s	n	Rep	$= 0$	> 0	Avg (x10^{-2})	Case Avg (x10^{-2})	< 0	Avg (x10^{-2})	Case Avg (x10^{-2})
			20	1	19	5	1.11		0	-	
			20	2	18	6	0.30		1	-0.02	
0	0	0	20	3	18	6	0.12	0.40	1	-0.01	-0.01
			20	4	23	2	0.51		0	-	
			20	5	20	4	0.01		1	-0.01	
			20	1	1	3	0.84		21	-0.11	
			20	2	20	5	0.30		0	-	
1	0	0	20	3	21	4	0.54	0.39	0	-	-0.11
			20	4	19	6	0.41		0	-	
			20	5	20	5	0.05		0	-	
			20	1	21	4	0.5		0	-	
			20	2	19	6	0.38		0	-	
0	1	0	20	3	18	7	0.07	0.22	0	-	-0.09
			20	4	19	6	0.08		0	-	
			20	5	22	2	0.14		1	-0.10	
			20	1	23	2	0.23		0	-	
			20	2	22	3	2.72		0	-	
0	0	1	20	3	19	6	0.38	0.63	0	-	-0.14
			20	4	17	8	0.24		0	-	
			20	5	19	5	0.45		1	-0.14	
			20	1	19	6	0.26		0	-	
			20	2	20	5	0.43		0	-	
1	1	0	20	3	22	3	0.10	0.34	0	-	-0.02
			20	4	22	7	0.39		0	-	
			20	5	19	4	0.43		2	-0.02	
			20	1	22	2	0.30		1	-0.08	
			20	2	22	2	0.19		1	-0.02	
1	0	1	20	3	20	5	0.42	0.24	0	-	-0.05
			20	4	22	3	0.14		0	-	
			20	5	21	3	0.03		1	-0.09	
			20	1	22	2	0.11		1	-0.01	
			20	2	21	3	0.58		1	-0.05	
0	1	1	20	3	21	4	0.23	0.20	0	-	-0.04
			20	4	21	3	0.14		1	-0.05	
			20	5	17	5	0.02		3	-0.04	
			20	1	19	4	0.02		2	-0.01	
			20	2	23	2	0.07		0	-	
1	1	1	20	3	19	6	0.62	0.29	0	-	-0.09
			20	4	20	5	0.33		0	-	
			20	5	21	3	0.05		1	-0.26	

Factors				DICOPT vs BARON (Model 1)							
A^1	A^2	A^s	n	Rep	$= 0$	> 0	Avg (x10^{-2})	Case Avg (x10^{-2})	< 0	Avg (x10^{-2})	Case Avg (x10^{-2})
			30	1	15	3	2.79		7	-0.02	
			30	2	21	3	0.06		1	-0.01	
0	0	0	30	3	10	5	2.03	1.12	9	-0.03	-0.02
			30	4	10	10	0.64		5	-0.02	
			30	5	21	3	0.60		0	-	
			30	1	20	3	0.62		2	-0.18	
			30	2	11	7	0.25		4	-0.03	
1	0	0	30	3	11	9	0.13	0.38	2	-0.01	-0.07
			30	4	7	6	0.56		9	-0.08	
			30	5	14	8	0.53		3	-0.03	
			30	1	17	2	0.86		6	-0.06	
			30	2	12	4	0.20		9	-0.05	
0	1	0	30	3	18	5	0.08	0.18	2	-0.02	-0.05
			30	4	16	8	0.15		1	-0.04	
			30	5	9	9	0.11		7	-0.05	
			30	1	17	5	0.20		3	-0.05	
			30	2	9	12	1.10		4	-0.06	
0	0	1	30	3	13	5	1.28	0.61	7	-0.02	-0.03
			30	4	15	6	0.04		4	-0.01	
			30	5	14	7	0.07		4	-0.01	
			30	1	13	11	0.36		1	-0.03	
			30	2	20	1	0.04		3	-0.04	
1	1	0	30	3	19	5	0.36	0.47	0	-	-0.03
			30	4	13	10	0.23		2	-0.03	
			30	5	13	10	0.94		2	-0.03	
			30	1	9	7	0.65		9	-0.05	
			30	2	12	3	1.45		8	-0.06	
1	0	1	30	3	14	4	0.66	0.66	7	-0.06	-0.07
			30	4	9	7	0.48		7	-0.11	
			30	5	10	6	0.49		8	-0.08	
			30	1	19	3	0.17		3	-0.08	
			30	2	17	1	0.16		5	-0.08	
0	1	1	30	3	17	5	0.13	0.10	3	-0.09	-0.05
			30	4	11	6	0.10		8	-0.04	
			30	5	12	5	0.02		8	-0.03	
			30	1	17	4	0.81		4	-0.03	
			30	2	11	7	0.95		7	-0.06	
1	1	1	30	3	18	5	0.39	0.53	2	-0.02	-0.05
			30	4	14	9	0.50		2	-0.01	
			30	5	14	8	0.15		3	-0.07	

VITA

Zeynep Uruk was born on February 26, 1985 in Sakarya, Turkey. She received her high school education at Sakarya Anadolu Lisesi, Turkey. She graduated from Middle East Technical University Chemical Engineering Department with a degree of honor in 2008. She joined the department of Industrial Engineering, Bilkent University, in 2008. Since then, she has been working with Dr. M. Selim Aktürk and Dr. Hakan Gültekin on her graduate study.