A HEURISTIC TO MINIMIZE TOTAL TARDINESS ON PARALLEL MACHINES: AN AGGREGATE PLANNING APPROACH

by

Engin Sansarcı

B.S., Management Engineering, İstanbul Technical University, 2004

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Industrial Engineering

Boğaziçi University

2007

A HEURISTIC TO MINIMIZE TOTAL TARDINESS ON PARALLEL MACHINES: AN AGGREGATE PLANNING APPROACH

APPROVED BY:

Assoc. Prof. Ali Tamer Ünal (Thesis Supervisor)	.....
Prof. Tülin Aktin	.....
Assoc. Prof. Necati Aras	.....

DATE OF APPROVAL: 10.08.2007

ACKNOWLEDGEMENTS

I am grateful to my thesis supervisor, Assoc. Prof. Ali Tamer Ünal for his guidance, patience and trust.

I would like to thank to Prof. Tülin Aktin for her support in my studies and participation in the thesis defense.

I would also like to thank to Assoc. Prof. Necati Aras for his interest and participation in the thesis defense.

I would like to thank to Güzay Paşaoğlu for her encouragement and help during my study.

Finally I want to thank to my family for their encouragement and support which was my source of energy.

ABSTRACT

A HEURISTIC TO MINIMIZE TOTAL TARDINESS ON PARALLEL MACHINES: AN AGGREGATE PLANNING APPROACH

Unrelated parallel machine total tardiness problem is investigated in this study. Job-splitting property, eligibility constraints and family setup structure are assumed. A previous study focused on the exact problem does not exist. We proposed a four phased heuristic in order to cope with the problem. Phase - I aggregates the jobs given two control parameters and generates aggregate jobs. Phase - II generates a time structure based on the due dates of the aggregate jobs fed from Phase - I. After that, Phase - II constructs an aggregate planning model and solves it iteratively. Phase - III generates a feasible schedule given the output of the aggregate planning model. At the end, Phase - IV, the tuning phase, tunes the control parameters and runs previous phases for a number of times. 480 problem instances are generated in order to test the proposed heuristic. On the other hand, some of the problem instances are also solved with CPLEX. The results show that the proposed heuristic gives better result than what CPLEX found within the 24 hour CPU time limit. We conclude that the proposed heuristic performs well in a reasonable amount of CPU time.

ÖZET

PARALEL MAKİNALARDA TOPLAM GECİKMEYİ EN AZA İNDİRMEK İÇİN SEZGİSEL BİR YAKLAŞIM

Bu çalışmada ilişkisiz paralel makinalarda toplam gecikmelerin enazlanması problemine odaklanılmıştır. İşlerin bölünebilme özelliği, makina-ürün ailesi uygunluk kısıtları ve ürün ailesine göre hazırlık süreleri olduğu varsayılmaktadır. Literatürde aynı problemden bahseden bir çalışmaya rastlanılmamıştır. Çalışmada dört aşamadan oluşan sezgisel bir yaklaşım önerilmektedir. Aşama - I, iki kontrol parametresine bağlı olarak işleri birleştirip grup işleri oluşturmaktadır. Aşama - II ise aşama - I'den gelen grup işlerin terminlerine göre bir zaman yapısı oluşturmaktadır. Ardından aşama - II, bütüncül bir planlama modeli oluşturup bu modeli ardarda çözmektedir. Aşama - III, bütüncül planlama modelinin çıktılarını kullanarak uygun bir çizelge oluşturmaktadır. Son olarak, aşama - IV, yani ayarlama aşaması, kontrol parametrelerini ayarlayıp önceki aşamaları birkaç defa çözmektedir. Önerilen sezgisel yaklaşımı test etmek için 480 problem örneği yaratılmıştır. Elde edilen sonuçlar önerilen sezgisel yaklaşımın, CPLEX'in 24 saat içerisinde bulduğu en iyi sonuçtan daha iyi sonuç bulduğunu göstermektedir. Sonuç olarak, önerilen sezgisel yaklaşım daha kısa bir zamanda daha iyi sonuç vermektedir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AP	Aperture
D	Demand
d	Due Date
DF	Due Date Factor
EDD	Earliest Due Date
EF	Eligibility Factor
FF	Family Factor
LP	Linear Programming
MF	Machine Factor
MIP	Mixed Integer Programming
NOB	Number of Batches
PPS	Production Per Setup
SF	Setup Factor
SPT	Shortest Processing Time
T	Tardiness
T_{max}	Maximum Tardiness
TET	Total Earliness and Tardiness
TSP	Traveling Salesman Problem
TWCT	Total Weighted Completion Time
TWT	Total Weighted Tardiness
WMCT	Weighted Mean Completion Time

1. INTRODUCTION

In today's competitive environment, manufacturing systems compete with each other in terms of delivery times and product variety. These two objectives often conflict with each other. Strict due dates put pressure on the managers of such manufacturing systems, while product variety requires additional setup times in production.

Depending on the characteristics of the product and production systems, a setup time is needed for preparing the production line for the next product. Setup times may be negligible between similar products while significant setup may be required between certain other products. We call a set of products with similar setup characteristics a *"family of products"*, and this type of setup structure is referred to as *"family setup"*. Products belonging to the same family also have the same machine requirements and production speed.

Customer requirement is referred to as a *"job"*. Information about a job includes the product, the due date, and the quantity ordered. It is assumed that there is no setup time between two products belonging to the same family. Thus, product information in jobs are transformed into family information. Producing several jobs (or parts of jobs) belonging to the same family consecutively is referred to as *"batching"* and that production as a whole is referred to as a *"batch"*.

When the family setup times are higher, producing all jobs belonging to the same family consecutively seems to be ideal. However, if due dates are strict enough, setup times would be unavoidable. Thus, the problem turns out to be considering the effects of due dates and setup times simultaneously.

In this study, we study the unrelated parallel machine scheduling problem with setup times in order to minimize total tardiness. Since parallel machines are unrelated, processing times of different families on machines may vary. Moreover, there exists eligibility constraints, which means that not every product can be processed on every

machine. There also exists setup times between the production of different families, which are also machine dependent. On the other hand, jobs can be split to be produced on different machines and/or at different times. However, splitting jobs in the same machine does not improve the solution.

The problem we focus on is very common in the industry. Especially in the textile industry, setup times are usually family dependent. Parallel machines, either identical or not, are also very common in manufacturing systems. In some cases, machines differ in terms of technology, and have different processing times and setup structures. Those parallel machines are then said to be unrelated.

Aksa Acrylic, a company established in Yalova in 1968, is an example. Since 1971, Aksa Acrylic produces acrylic fiber. First of all, they get Acronitrile and Vanylacetate as inputs and these inputs go through polymerization step. After the output is mixed with a specific solvent, they go through a process in one of the 26 parallel machines and we get fibers as output. The product families are defined in terms of some specifications like color, desitex (density) and apre (softness). Each one of the 26 parallel machines has the ability to produce a set of families defined in terms of the specifications above.

The problem can be expressed as a mixed integer programming (MIP) formulation. A formulation example is given in Chapter 3. Omar and Teo (2006) have also proposed a MIP formulation for a similar problem and suggested some improvements in the constraints. However, it will have a large amount of binary variables and constraints, which makes it hard to find the optimal solution, especially for larger problem instances. Thus, heuristic approaches would be more beneficial to solve larger problem instances.

Parallel machine scheduling problems with setup times have been widely analyzed in the literature with several objectives. There are also studies focused on the parallel machine total tardiness problems without setup. Likewise, single machine total tardiness problems also exist in the literature. Suggested solutions include mixed integer programming, branch and bound techniques, constructive and evolutionary or hybrid

heuristic approaches.

The problem can be reduced to three basic questions as follows:

- On which machine to produce
- How much to produce
- When to produce

We may refer to these questions as *sub-problems*. It is easy to see that; one sub-problem is very much dependent on any other. For that reason, all sub-problems should be considered simultaneously.

The problem is challenging because of three factors. An effective solution method has to deal with those factors simultaneously. These factors can be listed as follows:

- Due dates
- Setup times
- Different production speeds
- Eligibility constraints

If setup times are negligible in a problem instance, only due dates and production times would be important and batching activities would not be necessary. However, if there are setup times that cannot be ignored, then batching needs to be the key point in the solution procedure.

After a comprehensive thinking, it is easy to see that batching problem is actually about the trade-off between setup times and due dates. Since a good solution needs to avoid higher setup times, jobs belonging to the same family should be produced one after another. However, larger batch sizes can possibly make other jobs tardy. Thus, especially in a constructive approach, batching means a trade-off between the tardiness of earlier jobs and latter jobs on the time line. In order to manage that trade-off in an intelligent way, a decomposition method using aggregate planning approach will be

introduced in the following chapters.

In this study, a heuristic algorithm using an aggregate planning approach is proposed. The solution procedure is mainly about forming an aggregate planning model using two control parameters, in order to determine the batch sizes, batch sequencing, and alternative machine selection. A linear programming formulation is applied to solve the aggregate planning problem. With a heuristic algorithm, the main problem is reduced to several single machine total tardiness problems, using the results of the aggregate planning problem. Finally, a search algorithm is applied to tune the control parameters.

In Chapter 2, a literature review about parallel machine scheduling will be given with offered solution methods to similar problems. In Chapter 3, a formal definition to the scheduling problem at hand is going to be made. Similarity of the problems and the suggested solution procedures in the literature with the ones focused in this study is also going to be discussed. After that, the solution procedure we proposed will be explained in detail and the logic behind the approach will be introduced in Chapter 4. In Chapter 5, implementation of the algorithm for 480 problem instances will be shown. C++ programming language is used in the implementation of the algorithm. Finally, in Chapter 6, results and suggestions for further studies will be discussed.

2. LITERATURE REVIEW

Recalling the problem, we try to solve a parallel machine problem with setup times in order to minimize total tardiness. There are lots of studies in the literature which focus on at least one of these topics. However, since each factor makes the problem harder, studies involving all of them are very rare.

Parallel machine scheduling problems with setup times vary depending on the structure of the parallel machine environment and the setup time. Allahverdi *et al.* (1999) prepared a review of the scheduling research involving setup considerations but this review cover the studies before 1997. Researches involving setup considerations include both sequence-dependent and sequence-independent setup types.

Chen and Powell (2003) studied scheduling a set of jobs on identical parallel machines with sequence dependent setup times. In this study, no setup is assumed between the jobs belonging to the same family. However, the objective function in this study was not about tardiness. Instead, they tried to minimize total weighted completion time and number of tardy jobs separately. He used column generation based branch and bound technique in order to find exact solutions. They concluded that, medium sized problems can be solved optimally with their method.

Another study for parallel machine scheduling with setup consideration is done by Rabadi and Moraga (2006). They studied unrelated parallel machine scheduling with sequence dependent setup times. Their objective was minimizing the makespan. They stated that since the problem is NP-hard, optimal solutions can be found for small problem instances only. For larger problem instances, they offered a new meta-heuristic called Meta-RaPS. In their words, Meta-RaPS (Meta-heuristic for Randomized Priority Search) uses both construction and improvement heuristics to generate high quality solutions. The heuristic constructs a schedule by inserting the job with the shortest adjusted processing time to the machine with the smallest load. Following that, the heuristic uses this schedule in an improvement phase, if necessary. Finally, this sched-

ule is passed into an evolutionary phase. Newly constructed schedules added to the evolutionary phase from time to time, each of which is different because ties are broken arbitrarily. Computational results show fairly good performance.

A similar problem of makespan minimization is investigated by Tahar *et. al.* (2006). In their study, they considered the problem of scheduling a set of independent jobs with sequence dependent setup times with job splitting property. A heuristic based on linear programming formulation is suggested to cope with this problem. In the first phase, they solved the problem as if it is a single machine problem, by converting it into a TSP. After that, the solution is divided into equal parts for each machine. At that point, linear programming formulation is used in order to improve the sub-solutions for each machine. At each iteration, sub-TSP problems are solved with an appropriate method following a linear programming implementation. TSP problems are solved using genetic algorithms and local search. This algorithm performs well on the instances up to 70 jobs and 5 machines. However, it is interesting to note that, if the setup times were fixed rather than sequence dependent, the algorithm freezes.

Another problem of parallel machine, which are unrelated machines this time, is studied by Weng *et. al.* (2001) in order to minimize the weighted mean completion time. Again, sequence dependent setup times are assumed. They outlined seven constructive heuristics and compared them using simulation. All of these heuristics are about assigning jobs on machines one at a time, with a different priority rule for each. Moreover, all of these heuristics are reduced to WSPT (weighted shortest processing time first) rule if there is only one machine and no setups. The heuristic which outperforms others is different in a way that job selection and machine selection is determined simultaneously.

Simultaneous lotsizing and scheduling problem on parallel machines is studied by Meyr (2002). Again he assumed sequence dependent setup times and parallel machines are non-identical. He aimed cost minimization including setup costs, inventory holding costs and production costs. This problem is also similar to ours in a way that jobs are grouped into job families and setup time between jobs of the same family are either

zero or negligible. He used a heuristic method of combining threshold acceptance and simulated annealing methods with dual re-optimization. He concluded that, even the solution quality of the heuristic suggested is fairly good, reducing the computation time is still a major problem.

The papers reviewed above are about parallel machine scheduling problems considering setup times. The characteristic of the setup varies. Also, parallel machines are either identical or unrelated. However, the problems concerned in the papers are different from ours in terms of the objective. Used methods to solve the problems are also various, some of them used constructive heuristics and some of them used local search methods or hybrid approaches. Integer programming and branch-and-bound strategies are also used in some studies in order to seek for optimal solutions. In general, we can conclude that, the results are either not satisfying enough, or spending lots of time for computation.

Problem of minimizing total tardiness is one of the most challenging problem types in the scheduling literature. It is NP-hard even for single machine. Before exploring the literature about parallel machine total tardiness problems, we better summarize the literature about single machine total tardiness studies.

An earlier study analyzed single machine problem is prepared by Chen (1997). In this problem, he considered several batches of jobs processed by a single machine. Accordingly, setup time is referred as 'batch setup'. He tried to minimize total earliness and tardiness penalties. He also extended his study such that due date is a decision variable and has a penalty to minimize. He offered a dynamic programming approach which runs in polynomial time in the number of jobs, but exponential in the number of batches. He concluded that this algorithm is successful in small sized problems.

Lee *et al.* (1997) analyzed a similar problem with minimizing total weighted tardiness objective. Unlike the study of Chen (1997), batching is not the subject of the discussion here. He proposed a three-phase heuristic. First phase is about analyzing the problem instance, and determining the parameters to feed the second phase. After

that, sequencing is done with a priority rule whose parameters are determined in the previous phase. He added an additional improvement phase in order to improve the quality of the solution in a relatively small time. He used some insertion and swap techniques in the improvement phase. The paper does not compare the performance of the heuristic suggested with other methods, however, it concludes that the proposed method is implemented in a number of factories. Finally, the author states that, the approach can be modified for use in parallel machines and in environments with release dates.

A recently published study about single machine total tardiness problem is proposed by Luo and Chu (2006). He assumed sequence dependent setup times and a production environment without allowing preemption. An algorithm based on branch-and-bound permutation schemes is implemented compared with a similar approach proposed by Ragatz (1993). He concluded that, his algorithm performs better than the branch-and-bound algorithm in the study of Ragatz (1993).

Other then single machine total tardiness problems, there are also parallel machine total tardiness minimization problems in the literature. One of them is studied by Liaw *et al.* (2003). They considered unrelated parallel machine and picked the objective of minimizing the total weighted tardiness. They used branch and bound method in order to find the optimal solution. They found the lower bound based on the solution of an assignment problem and obtained the upper bound by a two-phase heuristic. Dominance rules are developed based on the study of Azizoglu and Kirca (1999). They concluded that the suggested algorithm performs fairly good on problem instances with up to 18 jobs and 4 machines.

A more recent study is about parallel machine scheduling is prepared by Shim and Kim (2006). The objective was minimizing total tardiness but without the setup time factor. However it is similar to the problem we focus in terms of the job splitting property. They seek optimal solutions using a branch and bound algorithm. It is concluded that, moderate sized problems are solved perfectly in a reasonable amount of computation time.

We summarized the studies which focus on the problems similar to ours. To remind, we focus on the problem of unrelated parallel machine scheduling with setup times in order to minimize total tardiness. Studies of much more similar problems are reviewed below.

A very early study including both setup time and parallel machine factors is done by Lee and Pinedo (1997). Their objective was minimizing the sum of the weighted tardiness. They offered a three phase heuristic. In the first phase, a pre-processing procedure is used, in order to analyze the problem instance and to determine the best parameters which are going to be used later. In the second phase, with the help of the parameters fed by the first phase, a priority rule is executed, in order to dispatch the jobs to the machines and generate first sequence for each. After all, an improvement algorithm is executed in the third phase using simulated annealing method. It is concluded that, the algorithm is implemented in real time production environments, and the results were satisfactory.

Balakrishnan *et al.* (1999) went a step further and included non-zero ready times to the problem. Again they assumed sequence dependent setup times and parallel machine environment. Other features which make the problem harder are that parallel machines are non-identical and the objective is the minimization of the sum of earliness and tardiness cost. Unlike Lee and Pinedo (1997), they offered a mixed programming mathematical formulation in order to search for the optimal solution. Two binary decision variables is used in the model, one for job-machine assignments, and one for representing the sequence. They concluded that, 10-job, 2-machine problems can be solved optimally in a reasonable amount of time. For future research for larger problem instances, they offered a decomposition technique named Bender's decomposition which is explained in detail by Zionts (1974).

Bilge *et al.* (2004) considered the problem of scheduling a set of independent jobs with sequence dependent setup times on a set of uniform parallel machines in order to minimize total tardiness. This problem is very similar to ours in terms of these aspects. However, they added the non-identical ready times of the jobs to the problem, which

makes it much harder. A tabu search algorithm is implemented to cope with this problem. Initial solutions for tabu search is constructed with an EDD approach, that is, jobs sequenced with EDD are assigned to the machines one at a time, where it would finish the earliest. The proposed heuristic is tested on the problem set given in Şerifoğlu and Ulusoy (1999). It is concluded that, implemented tabu search algorithm found the best results for the benchmark problem as to that date.

Anghinolfi and Paolucci (2007) also considered the same problem type with Bilge *et al.* (2004). They used a hybrid metaheuristic approach which is a combination of tabu search, simulated annealing and variable neighborhood search. Several alternative configurations of the hybrid metaheuristic are implemented to compare the results. They used the same benchmark problems with Bilge *et al.* (2004). One of the advantages of their approach is said to be the flexibility it has, with configuring the behavior. It is concluded that some of the configurations outperformed the algorithm of Bilge *et al.* (2004).

Another recent study about the problem of minimizing total tardiness on parallel machines is prepared by Armentano and Filho (2006). Similarly, they assumed sequence dependent setup times and non-zero ready times as Anghinolfi and Paolucci (2007) and Bilge *et al.* (2004). They proposed a greedy random search adaptive procedure (GRASP) in order to deal with the problem at hand. This technique needs a greedy constructive algorithm for the first phase and a local search algorithm for the second phase. They explained why total tardiness objective is not suitable for a greedy function, and suggested a randomized construction algorithm for the first phase. After that, they examined the use of long term memory composed of an elite set of high quality and sufficiently distant solutions. They stated that, as an important advantage of the proposed algorithm, the balance between diversification and intensification can be controlled. It is concluded that the average running times to find robust results are smaller than the other algorithms compared with.

On the other hand, Logendran *et al.* (2007) assumed unrelated parallel machine environment in the same problem above. What they tried to minimize was

total weighted tardiness, instead total tardiness. Dynamic availability of the machines are also assumed as well as dynamic job release dates. They proposed six different search algorithms based on tabu search in order to cope with this complex problem efficiently. Moreover, they offered four different initial solution finding mechanisms, based on dispatching rules. After deep analysis of the results gathered, it is concluded that the search algorithm with short term memory and fixed tabu list size performs well for small problem instances. On the other hand, long term memory and minimum frequency is claimed to perform better for medium and large problem instances.

Above we summarized some studies about parallel machine total tardiness minimization problem which considers setup times. However, the setup times assumed are usually sequence dependent type in the literature and they occur between any two jobs. To remind, our scheduling problem is a family scheduling problem. Family scheduling problems have a setup time structure where setups occur only between jobs from different families. A study considering family setups belongs to Chen and Powell (2003) which is summarized above. As mentioned before, family setup scheduling problems are rare compared to others.

An earlier study by Webster (1997) is about showing the complexity of scheduling job families about a common due date. They considered parallel machine production environment and analyzed the objective of minimizing weighted deviation about a common due date. They showed that the total earliness/tardiness problem is NP-hard when the number of machines and families are arbitrary.

A very recent study from Omar and Teo (2006) looks like very similar to our problem. They aimed to minimize total earliness and tardiness on identical parallel machines with sequence dependent family setup times. They offered a mixed integer programming formulation model to deal with this problem.

The problem we concern is minimizing total tardiness on unrelated parallel machines with family setup time, job-splitting property and eligibility constraints. In the literature of parallel machine scheduling, machines are usually assumed to be identi-

cal or non-identical. Only in the studies of Rabadi and Moraga (2006), Weng *et al.* (2001), Liaw *et al.* (2003) and Logendran *et al.* (2007) unrelated parallel machines are assumed. On the other hand, job-splitting property is assumed only in the studies of Tahar *et al.* (2006) and Shim and Kim (2006). Moreover, there is not any eligibility constraint in the reviewed literature.

Table 2.1 summarizes some papers in terms of the machine environment, objective function and setup structure. Sequence dependent setup is written as "Seq. Dep.", machine environment is written as "Machine Env." and cost minimization is written as "Cost Min." shortly.

Table 2.1. Summary of the literature

Paper	**Machine Env.**	**Objective**	**Setup**
Anghinolfi and Paolucci (2007)	Identical	Total Tardiness	Seq. Dep.
Armentano and Filho (2006)	Identical	Total Tardiness	Seq. Dep.
Balakrishnan *et al.* (1999)	Non-identical	TET	Seq. Dep.
Bilge *et al.* (2004)	Identical	Total Tardiness	Seq. Dep.
Chen and Powell (2003)	Identical	TWCT	Family Setup
Chen (1997)	Single	TET	Seq. Dep.
Lee *et al.* (1997)	Single	TWT	Seq. Dep.
Lee and Pinedo (1997)	Identical	TWT	Seq. Dep.
Liaw *et al.* (2003)	Unrelated	TWT	No Setup
Logendran *et al.* (2007)	Unrelated	TWT	Seq. Dep.
Luo and Chu (2006)	Single	Total Tardiness	Seq. Dep.
Meyr (2002)	Non-identical	Cost Min.	Seq. Dep.
Omar and Teo (2006)	Identical	TET	Seq. Dep.
Rabadi and Moraga (2006)	Unrelated	Makespan	Seq. Dep.
Shim and Kim (2006)	Identical	Total Tardiness	No Setup
Tahar *et al.* (2006)	Identical	Makespan	Seq. Dep.
Weng *et al.* (2001)	Unrelated	WMCT	Seq. Dep.

Our study differs from the ones in the literature in terms of the problem type and proposed solution. There is no study in the literature which focused on the same problem. The difference is mainly about the setup structure, eligibility constraints, and job-splitting property. Family dependent setup structure is assumed in this study, thus, job sequencing does not have any effect on the setup time. However, batching becomes more important here, in order to reduce the number of setups.

The solution method proposed in this study is also different from the others in the literature. In the literature, different branch and bound techniques, constructive heuristics and search methods are proposed for similar problem types. However, the solution we proposed is different in terms of the decomposition technique, mathematical model and search algorithm. Unlike the others, the proposed mathematical model does not include binary variables. It also deals with batch sizing and alternative machine selection problems with the help of an LP model. Search is done with two parameters related to aggregation, which is proposed for the first time.

3. PROBLEM DEFINITION

In this study, we have focused on a scheduling problem which is common in many manufacturing systems. Considering a certain process in a manufacturing system, there are n jobs, m machines and F product families. Product families are shortly referred to as *"families"*. Each specific customer order is referred to as a *"job"*. All jobs are well defined in terms of family, due date and quantity and they are ready to be processed at the beginning of the scheduling. The problem is to schedule them on m parallel machines. These machines have different technological properties. These technological properties allow the machines to process certain families.

Starting to produce a certain family on a machine requires a setup. This setup is about the configuration changes and the preparation of the machine for the family. In general, a setup may need some time and cost. However, in this study, we only care about the time spent for setup and ignore the cost. Setup time may be either constant or dependent. Dependent setup times may be either sequence dependent or family dependent. In this case, family dependent setup time is assumed and it is referred as *"family setup"*. Setup time depends also on the machine which the family is produced. In another words, setup time depends on both the machine and the family.

Setup is not needed between productions of two jobs belonging to the same family. In other words, jobs belonging to the same family can be produced consecutively with only one setup. These consecutive jobs form a *"batch"*. Forming a batch is illustrated in the Figure 3.1.

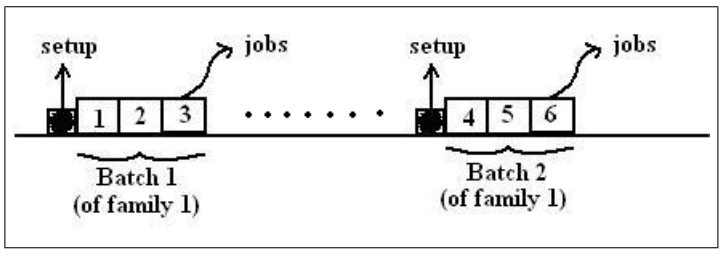

Figure 3.1. Forming a batch

In our problem we assume that the production system consists of m parallel machines. The term *"parallel"* implies that the machines are for the same purpose, in another words, there is only one process type which the parts need to pass through. In general, parallel machines may be either identical or non-identical. Identical parallel machines are those totally equivalent to each other. Non-identical machines, on the other hand, have different technological properties which lead to different production speeds. A special form of the non-identical machines is called *"unrelated machines"*. Unrelated machines have different production speeds for each product family. In another words, different families are produced on non-identical machines with proportional speeds while the production speed of them on unrelated machines are completely unrelated. In the production system focused in this study, m parallel machines are assumed to be unrelated.

While different machines have different production speeds for different families, because of the technological properties they have, some machines may not be eligible to produce some families. This adds additional constraints to the scheduling process, and they are referred to as *"eligibility constraints"*. These constraints are well-defined at the beginning of the scheduling process.

Jobs are going to be scheduled on the unrelated parallel machines considering the constraints above. However, jobs don't have to be produced at one time. A job may be produced partially on one machine and partially on another. For this reason, a job can be split into several parts to be produced on different machines and/or at different times. Figure 3.2 illustrates how a job is split into two parts to be produced on two different machines.

In such a production system as described above, one may consider several objectives. These objectives may be about cost, time, or customer satisfaction. In this study, we are going to try to produce goods on time in order to satisfy the customers. For this reason, we chose minimization of the total tardiness as the performance measure.

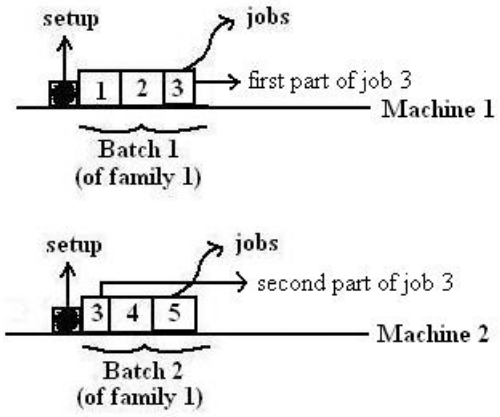

Figure 3.2. Splitting jobs

Tardiness of a job is defined as the difference between the delivery time and the due date of that job, without having a negative value. That means, if you deliver the job before its due date, then there is no penalty. In that case, it is not important that how early you delivered the job. However, if you deliver the job later than its due date, the difference between the delivery time and the due date will be the system's penalty cost.

Jobs have different quantities. In that sense, tardiness of a job with a higher quantity should have a higher penalty. For this reason, we are going to minimize total weighted tardiness where weights are the quantities of jobs. For simplicity, the term *"tardiness"* will be used in the meaning of *"weighted tardiness"* from now on.

3.1. Formal Definition

The problem at hand is the scheduling of n jobs on m unrelated parallel machines in order to minimize total tardiness. There are F product families and each job i belongs to a family f and has a due date d_i, where i is the job index and f is the family index. D_i shows the quantity ordered in job i. x_{if}^F is a binary parameter which shows if the job i belongs to family f. Q_f is the set of jobs i where $x_{if}^F = 1$. c_{fp} is the amount of time needed to produce one unit of a job of family f on machine p. A setup time S_{fp} is required whenever a family f is produced after any other family or at the first place on the machine p . There is also an eligibility constraint that means not

each family f can be produced on each machine p. x^E_{fp} is a binary parameter which equals 1 if family f can be produced on machine p and zero otherwise.

By definition, one machine can process one job at a time. The machines which are eligible to produce jobs belonging to a certain family are well defined. Machine dependent processing times and family and machine dependent setup times are also constant and known. Inventory capacity is assumed to be infinitive and there is no other constraint about production. Production costs and setup costs are ignored.

Jobs can be split. For this reason, scheduling should be done in terms of production quantities instead of jobs. Production of a family without interruption on a machine will be referred to as a *"batch"*. The solution can be represented by X^B_{pb} and x^{BF}_{pbf}, where X^B_{pb} represents the quantity of batch b on machine p and x^{BF}_{pbf} is a binary variable showing if the batch b on machine p is of family f.

A job which has a completion time later than its due date is referred to be *"tardy"*. *"Tardiness"* of a tardy job is the difference between its completion time and due date. Tardiness of a non-tardy job is zero. Total tardiness of a problem instance is the sum of tardiness for all jobs. Weighted tardiness of a job is the product of its tardiness and its quantity. Total weighted tardiness of a problem instance is the sum of weighted tardiness for all jobs. However, total tardiness will be used in the meaning of total weighted tardiness from now on.

3.2. Mathematical Model

The problem at hand can be formulated as a MIP. This model represents the formal definition of the problem. On the other hand, it is used as a base to evaluate the performance of the proposed heuristic. Variables of the model are as follows:

y_{ip} : Proportion of job i which is produced on machine p;
x_{ip} : 1 if job i is processed on machine p and 0 otherwise;
x_{0ip} : 1 if job i placed at the first place on machine p and 0 otherwise;
x_{jip} : 1 if job i comes just right after job j on machine p and 0 otherwise;
T_i : Tardiness of job i;
C_i : Completion time of job i;
C_{ip} : Completion time of job i on machine p;

Parameters of the model are as follows:

D_i : A constant which represents the quantity of job i;
d_i : A constant which represents the due date of job i;
p_{ip} : A constant which represents the processing time of job i on machine p;
s_{ip} : A constant which represents the setup time for job i on machine p;
s_{jip} : A constant which represents the setup time for job i on machine p if job j and job i belong to different families and is equal to 0 otherwise;

The mathematical model is as follows:

$$min \sum_i T_i D_i$$

subject to:

$$\sum_p y_{ip} = 1 \quad \forall_i \,, \tag{1}$$

$$x_{ip} - y_{ip} \geq 0 \quad \forall_{i,p} \,, \tag{2}$$

$$x_{ip}^{E} - x_{ip} \geq 0 \quad \forall_{i,p} \,, \tag{3}$$

$$\sum_{i} x_{0ip} = 1 \quad \forall_{p} \,, \tag{4}$$

$$\sum_{i} x_{jip} \leq 1 \quad \forall_{j,p} \,,$$

$$\sum_{j \neq i} x_{jip} + x_{0ip} - x_{ip} = 0 \quad \forall_{i,p} \,, \tag{6}$$

$$T_i - C_i + d_i \geq 0 \quad \forall_{i} \,, \tag{7}$$

$$C_{ip} - p_{ip} y_{ip} - \sum_{j \neq i} s_{jip} x_{jip} - s_{ip} x_{0ip} - \sum_{j \neq i} C_{jp} x_{jip} = 0 \quad \forall_{i,p} \,, \tag{8}$$

$$C_i - C_{ip} \geq 0 \quad \forall_{i,p} \,, \tag{9}$$

$$T_i \geq 0, y_{ip} \geq 0, C_i \geq 0, C_{ip} \geq 0 \quad \forall_{i,j,p} \,, \tag{10}$$

$$x_{ip} = \{0,1\}, x_{0ip} = \{0,1\}, x_{jip} = \{0,1\} \quad \forall_{i,j,p} \,, \tag{11}$$

Constraint 1 states that, for each job, proportions of that job which are produced on the machines sums up to 1. Constraint 2 states that, if a job is not produced on a machine, than the corresponding proportion for that job is 0. Constraint 3 states that, a job can be produced on a machine only if the machine is eligible for that job. Constraint 4 states that, there can be only one job to be placed first for each machine. Constraint 5 states that, each job can follow only one job on each machine. Constraint 6 states that, if a job is produced on a machine, it must either be the first job, or follow any other job on that machine. Constraint 7 states that, tardiness of a job is greater than or equal to completion time of that job minus its due date. They are not necessarily equal since tardiness cannot be negative. Constraint 8 states that, completion time of a partial production of a job on a machine is the sum of its processing time, setup time, and the completion time of the previous partial production of a job on the same machine. Constraint 9 states that, completion time of a job is greater than all of the completion times of the partial production of that job on the machines. Constraint 10

states that, T_i, y_{ip}, c_i and c_{ip} are greater than or equal to zero. Constraint 11 states that, x_{ip} , x_{0ip} and x_{jip} are binary integer variables.

4. THE PROPOSED METHODOLOGY

The focused problem is challenging because of the unrelated parallel machine environment, eligibility constraints, job-splitting property and family setup structure. After a number of trials, we decided that it is not possible to handle these issues simultaneously with a constructive heuristic. For this reason, we decided to apply an LP-based heuristic approach to cope with the problem.

The proposed heuristic consists of four phases. Phase - I is the aggregation phase where jobs with due dates close enough are combined in order to provide enough quantity. Two control parameters, production-per-setup and aperture, are used as inputs in this aggregation phase. Production-per-setup is a measure to see how efficient the capacity is used. Higher values of this parameters mean that a higher ratio of the capacity is used for production and higher setup times are avoided. Aperture, on the other hand, is a measure to see how far jobs are aggregated. Lower values of this parameter means that only jobs with closer due dates are aggregated, thus, no job is produced too much earlier than its due date. Figure 4.1 illustrates the meanings of the control parameters.

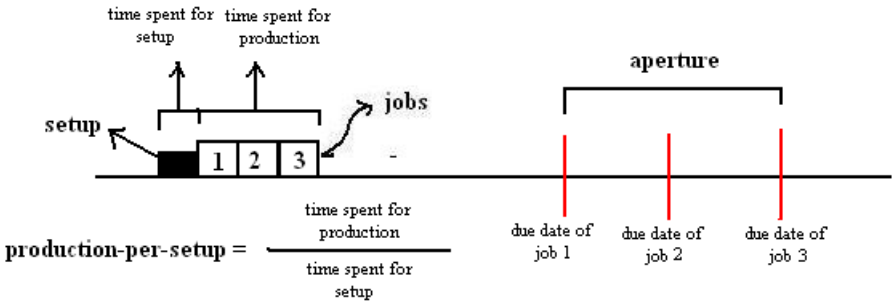

Figure 4.1. Explanation of the control parameters

Phase - II is the aggregate planning phase. In this phase, an LP model is solved iteratively. The output of the LP model does not represent a feasible schedule. Phase - III generates a feasible schedule given the output of the LP model. Phase - IV, on the other hand, is the tuning phase, which tunes two control parameters in order to

find better schedules. Figure 4.2 represents a flowchart of the heuristic.

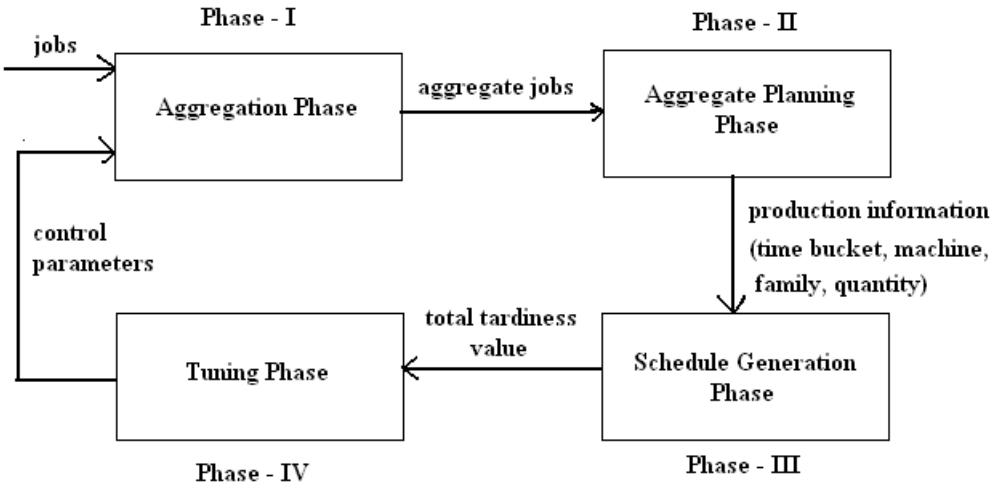

Figure 4.2. Flowchart of the heuristic

Using aggregate planning model for the problem at hand allowed us to decrease the number of decision variables and to get rid of the binary decision variables about the job sequence. The binary decision variables about setup still remain. However, they can also be removed with some assumptions.

4.1. Phase - I

In this phase, we group the jobs belonging to the same family. We call this *aggregation*. The reason we are aggregating the jobs is to form batches for avoiding higher setup times. Since we have a limited capacity, it is beneficial to use this capacity for production instead of setup. After aggregation, there will be a unique quantity and a due date for the aggregated job.

The ratio of the capacity spent for production and capacity spent for setup is an important measure. The other important measure is the range of the aggregation. With the term range, we mean the highest difference between the due dates of the aggregated jobs. It is desired to have a higher production-per-setup and a lower difference between the due dates of the aggregated jobs. The reason will be clear when the due date calculation for aggregate jobs is made.

4.1.1. Notation

Inputs of the algorithm are as follows:

PPS : Production-per-setup

AP : Aperture

The notation to be used in the algorithm is as follows:

D_i : Quantity of job i

d_i : Due date of job i

x_{if}^F : Binary parameter which is equal to 1 (respectively, 0) if job i belongs to (respectively, does not belong to) family f

S_{fp} : Setup time of family f on machine p

c_{fp} : Time needed to produce a unit of family f on machine p

d_{fk} : Due date of the aggregate job k belonging to family f

d_{1fk} : Due date of the first job added to the aggregate job k belonging to family f

D_{fk} : Total quantity of jobs aggregated to form the aggregate job k belonging to family f

4.1.2. Algorithm

The steps of the algorithm are as follows:

STEP - 0: SET $f = 1$.

STEP - 1: IF f is greater than the number of families, STOP. ELSE:

- Sequence the jobs in Q_f with non-decreasing order of d_i.
- Let $d_{(i)}$ show the due date and $D_{(i)}$ show the quantity of the ith job in the sequence.
- Set $k = 1$, $i = 1$, $d_{1fk} = 0$ and $D_{fk} = 0$.

STEP - 2: SET $D_{fk} = D_{fk} + D_{(i)}$. IF $d_{1fk} = 0$ SET $d_{1fk} = d_{(i)}$, $d_{fk} = d_{(i)}$, ELSE SET $d_{fk} = \min(d_{(i)}, d_{fk} + D_{(i)}.\sum_p c_{fp}x^E_{fp}/\sum_p x^E_{fp})$

STEP - 3: IF $d_{(i)} - d_{1fk} \geq AP$ OR $D_{fk}.\sum_p c_{fp}x^E_{fp}/(\sum_p S_{fp}x^E_{fp}) \geq PPS$ SET $k = k + 1$, $D_{fk} = 0$, $i = i + 1$, $d_{1fk} = 0$ ELSE SET $i = i + 1$

STEP - 4: IF $i > \sum_i x^F_{if}$ SET $f = f + 1$ and RETURN STEP - 1. ELSE RETURN STEP - 2.

4.1.3. Illustration of the Algorithm

Assume we have 5 jobs belonging to family f with quantities and due dates as follows:

$$d_1 = 1 \ , \ d_2 = 2 \ , \ d_3 = 3 \ , \ d_4 = 6 \ , \ d_5 = 10;$$

$$D_1 = 1 \ , \ D_2 = 2 \ , \ D_3 = 1 \ , \ D_4 = 0.5 \ , \ D_5 = 1;$$

Moreover, assume there are two machines with setup times of 1 and 3 respectively. The average setup time for this family will be 2. Assume $PPS = 4$, $AP = 2$, $c_{f1} = 2$ and $c_{f2} = 6$ for family f. With these numbers, the average time to produce a unit of family f will be 4.

The algorithm starts by adding job 1 to the aggregate job 1. After that, job 2 is added to the aggregate job 1. At that point, D_{f1} becomes 3. The maximum difference between the due dates of the jobs added is 1. Since this value is not greater than or equal to the aperture, we shall continue with the aggregation. However, we shall also look for the production-per-setup value. Since the quantity of the aggregate job is 3 and average time to produce a unit of this family is 4, total time to produce 3 units will is $3 * 4 = 12$. Since the average setup time is 2 and PPS value is 4, $2 * 4 = 8$ is the total time spend for production which is enough to stop the aggregation. Because $12 > 8$ we stop the aggregation and restart the process in order to generate aggregate job 2. To sum up, job 1 and job 2 are aggregated into one aggregate job, and job 3 will be added to the next aggregate job.

Since we stopped the aggregation for aggregate job 1, we shall continue for aggregate job 2. Job 3 will be the first job added to aggregate job 2. At this point, aperture is 0 and the ratio of production time and setup is $1/2 = 0.5$. Since these values are not enough to stop the aggregation, we shall continue with aggregating job 4. After job 4 is added to aggregate job 2, total quantity becomes 1.5 and the aperture becomes 3. Since $1.5 * 4 = 6$ and $6 < 8$, total quantity is not enough to stop the aggregation. However, we shall also look for aperture. Because $3 > 2$ we stop the aggregation for aggregate job 2 and restart the process in order to generate aggregate job 3. Only job 5 is left behind, thus, only job 5 will be added to aggregate job 3. To summarize, job 1 and job 2 form aggregate job 1, job 3 and job 4 form aggregate job 2 and job 5 forms aggregate job 3.

4.1.4. Explanation of the Algorithm

In Phase - I, the aim is to group the jobs belonging to the same family which have closer due dates. Enough number of jobs should be aggregated so that the total quantity of the aggregate job should worth the setup time. The algorithm basically aggregates jobs until either the aperture value or the required production-per-setup value is reached.

After this phase, the aggregate jobs will be treated as if they are single jobs. Number of jobs will be reduced dramatically. These aggregate jobs will be used in the LP problem in Phase - II.

4.2. Phase - II

In this phase we are going to solve an LP problem using the aggregate jobs which are generated in the previous phase. Since we do not have a feasible schedule yet, the required setup time is unknown. For this reason, setup times are set initially for all aggregate job and machine. By solving the LP iteratively, over-estimated setups are modified step-by-step.

The approach used in this phase is similar to the aggregate planning approach. In aggregate planning problems in practice, time horizon is usually divided into equal buckets. However, in our approach, we construct the time bucket structure considering the due dates of the aggregate jobs.

There will be two types of time buckets: (i) family time bucket and (ii) resource time bucket. Family time buckets are constructed for each family f and a family time bucket is basically the time between the respective d_{fk} values.

For instance, imagine there are three aggregate jobs for a family f. Let the d_{fk} values be 3, 12 and 18 respectively. Then the first family time bucket for family f would be the time between 0 and 3, the second would be the time between 3 and 12

and the third would be the time between 12 and 18.

After family time buckets are constructed, the aggregate jobs of all families are combined in order to construct resource time buckets. Suppose there are F families and K_f aggregate jobs for each family f. Let J_{fk} be the kth aggregate job of family f and let Q_f be the set of aggregate jobs of family f such that $Q_f = \{J_{f1}, ..., J_{fK_f}\}$. Let Q be the set of all aggregate jobs such that $Q = \{Q_1, ..., Q_F\}$. After that, the jobs in the set Q are sequenced with increasing order of d_{fk}. Let $d_{[i]}$ represent the due date of the ith aggregate job in the sequence in the set Q and let i be the new due date index.

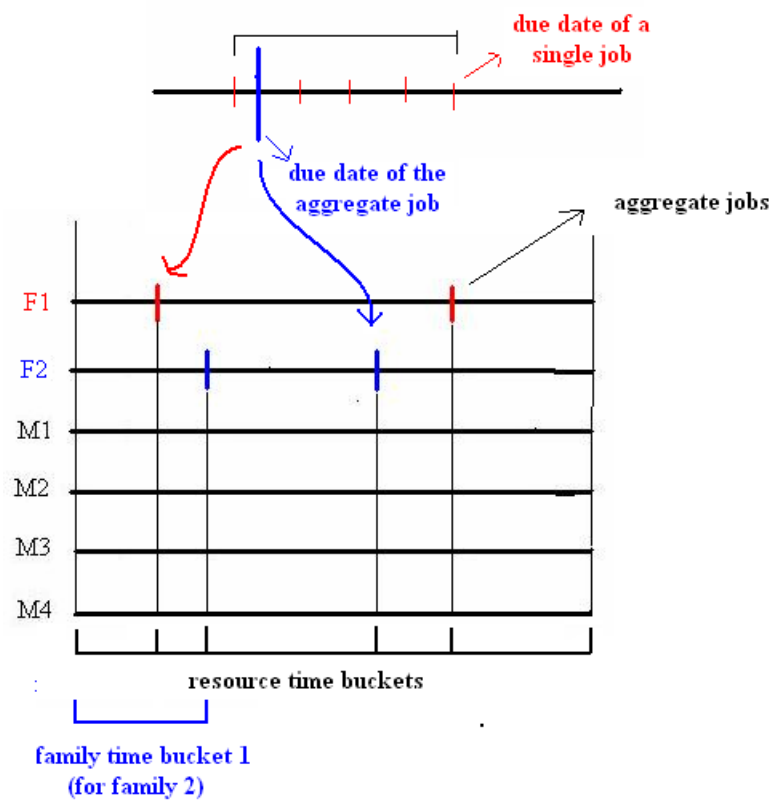

Figure 4.3. Creating time buckets

After these definitions, the resource time bucket is simply the time between the respective $d_{[i]}$ values. For instance, suppose there are two families and two aggregate jobs for each family. Let $d_{11} = 3$, $d_{12} = 18$, $d_{21} = 12$ and $d_{22} = 21$. After combining the aggregate jobs for these two families and sequence the jobs with increasing order of due dates, we will have $d_{[1]} = 3$, $d_{[2]} = 12$, $d_{[3]} = 18$ and $d_{[4]} = 21$. Then the first resource time bucket would be the time between 0 and 3, the second would be the time

between 3 and 12, the third would be the time between 12 and 18 and the forth would be the time between 18 and 21. In the LP model below, resource time buckets are represented by t and family time buckets are represented by t^f. It is important to note that, family time buckets are defined for each family while resource time buckets are defined for all machines. Figure 4.3 shows how to construct family and resource time buckets.

4.2.1. Notation

Definitions of variables and parameters for the LP model are below.

Variables:

X_{ft^f} : Total quantity of production of family f, at machine p, in family time bucket t^f.

X_{fpt} : Total quantity of production of family f, at machine p, in resource time bucket t.

S_{fpt} : Time spent for setup for producing family f at machine p in resource time bucket t.

Parameters:

x_{fpt^f} : Equals to 1 if setup for family f is needed for machine p in family time bucket t^f.

x_{fpt} : Equals to 1 if family f can be produced on machine p in resource time bucket t.

D_{ft} : Total demand of family f to be satisfied at the end of resource time bucket t. This value is equal to the quantity of the aggregate job of which due date is the ending point of resource time bucket t. If there is not such an aggregate job belonging to family f, this value is zero.

T_{ft^f} : Unsatisfied demand of family f at the end of family time bucket t^f.

$C^*_{t^f}$: Length of the resource time bucket which comes right after family time bucket t^f.

C_t : Length of resource time bucket t.

c_{fp} : Amount of time needed to produce one unit of family f at machine p.

S_{fp} : Setup time needed for family f on machine p.

S_{ft^f} : Setup time needed for producing family f in family time bucket t^f.

4.2.2. Aggregate Planning Problem

The LP model to be used in the algorithm is as follows:

$$min \sum_f \sum_{t^f} (T_{ft^f} C^*_{t^f})$$

subject to:

$$\sum_f X_{fpt} c_{fp} + \sum_f S_{fpt} - C_t \leq 0, \forall_{p,t} \quad (1)$$

$$S_{fp} x_{fpt^f} - \sum_{t \in t^f} S_{fpt} = 0, \forall_{p,f} \quad (2)$$

$$T_{ft^f} - \sum_{t \leq t^f} (D_{ft} - \sum_p (X_{fpt} x_{fpt}) \geq 0, \forall_{f,t^f} \quad (3)$$

$$T_{ft^f} \geq 0, S_{fpt} \geq 0, X_{ft^f} \geq 0, X_{fpt} \geq 0, \forall_{f,t^f,p,t} \quad (4)$$

4.2.3. Explanation of the Model

The objective function of the LP above penalizes the total tardiness for each family and for each family time bucket of that family. Constraint 1 states that, the

sum of time spent for production and time spent for setup in a resource time bucket is limited by the capacity (length) of that resource time bucket. Constraint 2 states that, total setup time in the resource time buckets inside a family time bucket on a machine is equal to required setup for the corresponding family and machine multiplied by x_{fpt^f}. Constraint 3 states that, tardiness for a family and a family time bucket of that family is greater than or equal to the difference between cumulative demand and cumulative production of that family at that family time bucket. Constraint 4 states that, all of the decision variables are greater than or equal to zero.

4.2.4. Algorithm

In Phase - II, X_{fpt} values are calculated by the algorithm below.

STEP - 0: Let $l = 0$, $x_{fpt^f} = x_{fpt} = x^E_{fp}$ for all family f and machine p. IF $\sum_{t \in t^f} 1 = 1$ AND $t^f \neq 1$ for any t^f and f, SET $x_{fpt^f} = 0, \forall_p$.

STEP - 1: Run LP. Set $l = l + 1$.

STEP - 2: IF $X_{fpt} = 0$ and $x_{fpt} = 1$ for any resource time bucket t, family f and machine p, SET $x_{fpt} = 0$. IF $X_{fpt} = 0 \quad \forall_{t \in t^f}$ and $x_{fpt^f} = 1$ for any family time bucket t_f, famly f and machine p, SET $x_{fpt^f} = 0$, ELSE STOP.

STEP - 3: IF $l = max - iteration$ STOP. ELSE RETURN STEP - 1.

4.2.5. Explanation of the Algorithm

The LP model above determines how much, when and on which machine to produce each family in order to minimize total tardiness. However, the resulting objective value and optimal values of the decision variables will not be accurate but just an estimate since the optimization is made at an aggregate level. Another handicap of the model is that it assumes a setup time for each family on each machine and on each family time bucket. This is because which machine is used to produce a certain family

in a certain resource time bucket is unknown.

On the other hand, the algorithm which runs the LP several times is used to get rid of this handicap. After the first run of the model, setup times are modified for the second run, considering the solutions just gathered. For instance, if there is not any production of family 2 on machine 3 in the first family time bucket of that family, x_{2312} value will be set to zero, thus, there will be no setup for the concerned family, bucket and machine. Consequently, that resource capacity will be allocated for production of another family in the second run of the LP. This process continues until the maximum number of iterations is reached (which is assumed to be 5 in this study) or after two iterations giving the same results.

4.3. Phase - III

Heuristic starts with the information about jobs and the production environment, and needs to give a complete schedule at the end. In this phase, we are going to generate the final schedule. Phase - II gives the production information for resource time buckets. As mentioned before, a production of a family without interruption on a machine is referred as a batch. That's why we need to transform the production information into batch information. A complete schedule is going to be represented as a set of batches, including the information of start time, finish time, family and machine.

The problem at hand has the job-splitting property. That means, a job can be split into partial jobs to be produced on different machines and/or at different times. However, splitting jobs in the same machine does not improve the solution. The reason for this proposition is as follows.

Assume job 1 is scheduled on machine 1 in two parts, first being between time 5 and 10, second being between time 15 and 20. Between time 10 and 15, there is a setup for job 1, and there is a production of another job belonging to another family. Regardless of being tardy or not, the tardiness of job 1 would not be affected if we

move the first production, which is scheduled between time 5 and 10, to right before the production of the second part of job 1. Since the finishing time of job 1 is still 20, tardiness for this job is unaffected. However, because we moved the production between time 5 and 10 to a later time, the production of the other job between time 10 and 15 can now be scheduled earlier and perhaps the tardiness for that job would be lowered. For this reason, splitting jobs in the same machine does not improve the solution.

Following the proposition above, we come to the conclusion that production quantities in different resource time buckets belonging to the same family time bucket should be combined. The reason is clear since they are the partial productions of the same aggregate job and an aggregate job is treated as if it is a single job and has a common due date. This idea is the underlying logic of this phase.

4.3.1. Algorithm

The steps of the algorithm are as follows:

STEP - 0: For each machine p, let Q_p be the set of X_{fpt} values calculated in Phase - II. Sequence all aggregate jobs with increasing due dates d_{fk}. Let f_i show the family and $d_{[i]}$ show the due date of the ith aggregate job. Let $i = 1$.

STEP - 1: For each machine p and for each resource time bucket t, schedule X_{fpt} where the ending time of resource time bucket t is less than or equal to $d_{[i]}$ and $f = f_i$. Schedule the setup time if required. Remove scheduled X_{fpt} values from Q_p.

STEP - 2: IF i is equal to the number of aggregate jobs, GO TO STEP - 3. ELSE SET $i = i + 1$, RETURN STEP - 1.

STEP - 3: IF $Q_p \neq \emptyset$, schedule all X_{fpt} for families as one batch for each family, starting from the family f with the ealiest due date of the last aggregate job belonging to family f.

STEP - 4: IF $\sum_k D_{fk} > \sum_p \sum_t X_{fpt}$ for any family f, divide the difference to number of machines and schedule each to the machine where it can finish earliest. Start from the family with the smallest average processing time on eligible machines.

4.3.2. Explanation of the Algorithm

The algorithm above aggregates the production quantities for each resource time bucket to be produced at once in any time in the family time bucket. Since the family time buckets are constructed considering the due dates of the aggregate jobs, it does not affect the tardiness whether it is produced at the beginning or at the end of the family time bucket. As explained in Phase - II, production of an aggregate job is divided into several resource time buckets only for modeling purposes. That's why; the partial productions need to be aggregated into one batch.

After the aggregation of productions in different resource time buckets, they are scheduled on machines with the rule of EDD. However, for problem instances with strict due dates, it may not be available to produce the necessary quantity in resource time buckets. In that situation, LP may put the production in later resource time buckets. Those are also need to be scheduled in order to satisfy the demand. Third step in the algorithm handles that problem and schedules the tardy productions.

On the other hand, total capacity of the defined resource time buckets may not be enough for producing all jobs ordered. The LP model in Phase - II tries to use all available capacity for maximum production. However, there may be some problem instances where the capacity is much lower than needed. Forth step of the algorithm above schedules those productions which are referred as $unsatisfied\ demand$ at the end of the other batches. In this step, SPT rule is applied in order to choose which family to schedule first. Which machine to schedule is decided like it is done in the third step. This rule in machine selection basically says to choose the machine so that the batch going to be scheduled would be finish earliest.

4.3.3. Calculating Real Tardiness

To review the scheduling algorithm from the beginning, there are n jobs with due dates, quantities and belonging families to be scheduled on m machines with different production speeds, setup times and eligibility constraints. In the first phase, jobs belonging to the same family with closer due dates are aggregated so that the number of jobs is reduced. In the second phase, an aggregate planning problem is solved in order to determine which machine and which resource time bucket should be allocated for the production of those aggregate jobs. In the third phase, this information is used to create batches and schedule them to machines.

The aggregate planning model in Phase - II tries to minimize the total tardiness. However, since the solution is not represented by the sequence of jobs on machines, the actual starting and finish times of jobs are unknown at that point. For this reason, the actual total tardiness value is unknown. In another words, the minimized problem is not the real scheduling problem we focused in this study, instead a similar manipulated problem. Thus, the tardiness definition in the aggregate planning problem and the actual scheduling problem is different.

At the end of the third phase, a complete and feasible schedule is generated. This schedule can be represented by batches with start time, finish time, belonging family, and the quantity produced. Tardiness value for a job can be calculated considering the due date of that job, quantity of that job, and the batches in the schedule. Below is an illustration of a tardiness calculation for a job.

4.3.3.1. Illustration of Tardiness Calculation. Assume we are going to calculate the tardiness of job 1, where $d_1 = 20$ and $D_1 = 10$. Again assume job 1 belongs to family 1 and there are two batches in the final schedule for family 1. Batch 1 starts at 5, finishes at 20 and has a quantity of 5 units. Batch 2, on the other hand, starts at 15, finishes at 35 and has a quantity of 10 units. At time 5, there will not be any unit of family 1 at hand. At 20, there will be 7.5 units. Since 2.5 units more is needed to

satisfy D_1 value and the machine which batch 2 is scheduled on produces 0.5 units per unit time, job 1 would be produced at time 25. Tardiness of job 1 will be calculated as $(25 - d_1).D_1 = 50$.

Assume job 2 also belongs to family 1, where $d_2 = 25$ and $D_2 = 5$. As mentioned above, the production of batch 1 and batch 2 until time 25 is reserved for job 1. Thus, job 2 would be produced at time 35. Then, tardiness of job 2 will be calculated as $(35 - d_2).D_2 = 50$. Likewise, total tardiness for job 1 and job 2 will be calculated as 100.

4.4. Phase - IV

Previous phases give a complete feasible schedule for any two values of the parameters *(i)* production-per-setup and *(ii)* aperture. It is expected to get different solutions for different values of these two parameters. Phase - IV runs the previous phases several times for different values of the mentioned parameters in order to find a better schedule in terms of the objective.

Both of the tuning parameters can have continuous values. Thus, it is impossible to search for the optimal values of these parameters. Only a small number of values of these parameters can be searched in a reasonable amount of time. The steps of the algorithm are as follows:

4.4.1. Algorithm

STEP - 0: SET PPS and AP to their initial values. Let $f(x, y)$ be a function which returns the total tardiness value of the final schedule generated by Phase - III where $PPS = x$ and $AP = y$. Let T^c, T^n, T^s, T^e and T^w represent different tardiness values. Let Δ_{AP} and Δ_{PPS} represent the steps for the parameters.

STEP - 1: SET $T^c = f(PPS, AP)$, $T^n = f(PPS + \Delta_{PPS}, AP)$, $T^s = f(PPS - \Delta_{PPS}, AP)$, $T^e = f(PPS, AP + \Delta_{AP})$ and $T^w = f(PPS, AP - \Delta_{AP})$.

STEP - 2: IF $max\{T^c, T^n, T^s, T^w, T^e\} = T^c$ STOP. ELSE IF $max\{T^c, T^n, T^s, T^w, T^e\} = T^n$ SET $PPS = PPS + \Delta_{PPS}$, RETURN STEP - 1. ELSE IF $max\{T^c, T^n, T^s, T^w, T^e\} = T^s$ SET $PPS = PPS - \Delta_{PPS}$, RETURN STEP - 1. ELSE IF $max\{T^c, T^n, T^s, T^w, T^e\} = T^e$ SET $AP = AP + \Delta_{AP}$, RETURN STEP - 1. ELSE IF $max\{T^c, T^n, T^s, T^w, T^e\} = T^e$ SET $AP = AP - \Delta_{AP}$, RETURN STEP - 1. If there are more than one equal maximum values, choose one of them arbitrarily.

4.4.2. Explanation of the Algorithm

The algorithm basically looks north, south, east and west directions in the $PPS - AP$ plane, and moves to the best point of them. If those directions don't give better results than the center, the algorithm stops. On the other hand, the steps of Δ_{AP} and Δ_{PPS} affect the performance of the search dramatically. Too small values of these steps probably would not affect the solution because very closer values of PPS and AP would give the same aggregation.

In this study, it is suggested to run this algorithm twice, first with greater values for the steps, and secondly with smaller values for the steps. First run is referred as *long distance search* and the second run is referred as *short distance search*. The reason for two runs is that the surface of the $PPS - AP$ plane is thought to be very fluctuating. A long distance search is expected to move the solution to an area with better results and short distance search is expected to make additional adjustments.

4.5. Illustration of the Proposed Heuristic

In this section we are going to apply the proposed heuristic to a test problem. Assume we have 3 machines, 3 families and 15 jobs. Quantities, due dates and families of the jobs are represented in Table 4.1. Unit production times and setup times for all families on eligible machines are given in Table 4.2. As seen in the tables, family 0 can be produced only on machine 0, family 1 can be produced on machine 0 and machine 1 and family 2 can be produced on any machine.

Table 4.1. Job information of the test problem

Job	Quantity	Due date	Family
0	0.988495	79.9249	2
1	0.568194	38.7829	1
2	1.7882	49.0219	0
3	0.635029	88.6776	2
4	1.88855	83.8527	2
5	1.04489	53.0107	1
6	1.85766	27.1126	1
7	1.15348	31.4066	2
8	0.823572	10.5777	2
9	1.66301	97.9339	1
10	1.82183	66.7043	2
11	1.63408	80.1355	1
12	1.13627	94.2228	1
13	0.34199	77.1538	0
14	0.288583	54.6861	0

Table 4.2. Production times and setup times

Family	Machine	Production time	Setup time
0	0	16.428	2.93417
1	1	17.7154	4.84487
1	2	18.288	14.3559
2	0	18.4189	11.0121
2	1	18.2139	14.5376
2	2	18.3833	15.3872

In Phase - I, we are going to aggregate jobs for given values of the control parameters. Assume production-per-setup is chosen to be 4 and aperture is chosen to be 20. We begin with aggregating jobs of family 0. Average unit production time and average setup for family 0 is calculated to be 16.428 and 2.934 respectively. Since production-per-setup is 4, required quantity to stop aggregation becomes 0.714. Jobs of family 0 is given in Table 4.3 in non-decreasing order of due dates.

Table 4.3. Sequenced jobs of family 0

Job	Quantity	Due date
2	1.7882	49.0219
14	0.288583	54.6861
13	0.34199	77.1538

Quantity of the first job in the sequence is enough to form an aggregate job itself. Quantity of job 14 is less than 0.714, thus, job 14 and job 13 are combined in order to form the second aggregate job. Due date of the first aggregate job is equal to d_2 and due date of the second aggregate job is equal to $min(d_{13}, d_{14} + 16.428 * D_{13}) = 60.304$. Aggregate jobs of family 0 are given in Table 4.4.

Table 4.4. Aggregate jobs of family 0

Aggregate job	Quantity	Due Date
0	1.788	49.022
1	0.631	60.304

Aggregate jobs of family 1 are formed similarly. Average unit production time and average setup for family 1 is calculated to be 18.002 and 9.6 respectively. Since production-per-setup is 4, required quantity to stop aggregation becomes 2.133. Jobs of family 1 is given in Table 4.5 in non-decreasing order of due dates.

Aggregation shall stop whenever required amount is reached or aperture between jobs exceeds 20. With this rule, job 6 and job 1, job 5 and job 11, and job 12 and job 9 are combined and three aggregate jobs are formed. Aggregate jobs of family 1 are shown in Table 4.6

Table 4.5. Sequenced jobs of family 1

Job	Quantity	Due date
6	1.85766	27.1126
1	0.568194	38.7829
5	1.04489	53.0107
11	1.63408	80.1355
12	1.13627	94.2228
9	1.66301	97.9339

Table 4.6. Aggregate jobs of family 1

Aggregate Job	Jobs	Quantity	Due Date
0	6, 1	2.426	37.341
1	5, 11	2.679	80.136
2	12, 9	2.700	97.934

Finally we shall aggregate the jobs of family 2. Jobs of family 2 are given in Table 4.7 in non-decreasing order of due dates. Average unit production time and average setup time are calculated to be 18.339 and 13.646 respectively. Required quantity to form an aggregate job is 2.976. We start the aggregation with job 8. Since quantity of job 8 is not enough, we add job 7. Total quantity is still not enough so we shall continue aggregation. However, the difference between the due dates of the jobs is $20.829 > 20$. For this reason, we stop the aggregation here, and go on for the second aggregate job. Finally, all aggregate jobs are given in Table 4.8.

Table 4.7. Sequenced jobs of family 2

Job	Quantity	Due date
8	0.823572	10.5777
7	1.15348	31.4066
10	1.82183	66.7043
0	0.988495	79.9249
4	1.88855	83.8527
3	0.635029	88.6776

Table 4.8. Aggregate jobs of family 2

Aggregate job	Quantity	Due Date
0	1.977	31.407
1	4.699	83.853
2	0.635	88.678

Aggregate jobs for families are formed. We shall construct the time structure given the due dates of the aggregate jobs. List of aggregate jobs for all families is given in non-decreasing order of due dates in Table 4.9. Using these due dates, family time buckets are formed as shown in Table 4.10. Combining these family time buckets, resource time buckets are created as shown in Table 4.11.A picture representing family time buckets and resource time buckets are given in Figure 4.4.

Table 4.9. Aggregate jobs of all families

Quantity	Due Date	Family
1.977	31.407	2
2.426	37.341	1
1.788	49.022	0
0.631	60.304	0
2.679	80.1356	1
4.699	83.853	2
0.635	88.678	2
2.700	97.934	1

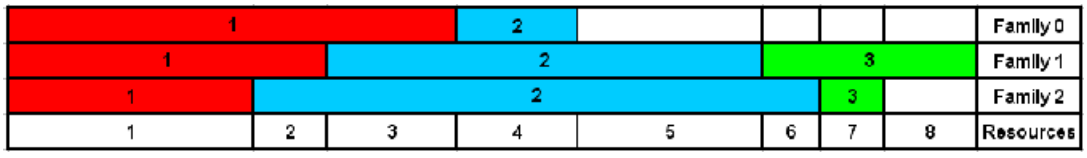

Figure 4.4. Family and resource time buckets

In Phase - II, an aggregate planning problem is modeled based on the time structure constructed in Phase - I. Since there are 8 resource buckets, there would be $8 * 3 * 3 = 72$ production variables. Number of setup variables would also be 72. There are four types of constraints as described in Section 4.2.3. Values of the pro-

Table 4.10. Family time buckets

Family	Start	Finish
0	0	49.022
0	49.022	60.304
1	0	37.341
1	37.341	80.136
1	80.136	97.934
2	0	31.407
2	31.407	83.853
2	83.853	88.678

Table 4.11. Resource time buckets

Bucket	Start	Finish
1	0	31.407
2	31.407	37.341
3	37.341	49.022
4	49.022	60.304
5	60.304	80.136
6	80.136	83.858
7	83.853	88.678
8	88.678	97.934

duction variables after the first run of the LP are given in Table 4.12. Resource time buckets are written as "R. B." shortly.

Table 4.12. Values of the production variables after the first run of the LP

Family	**Machine**	**R. B. 1**	**R. B. 2**	**R. B. 3**	**R. B. 4**
0	0	0.167	0.361	0.711	0.687
1	0	0.000	0.000	0.000	0.000
2	0	0.799	0.000	0.000	0.000
0	1	0.000	0.000	0.000	0.000
1	1	0.166	0.062	0.659	0.637
2	1	0.764	0.000	0.000	0.000
0	2	0.000	0.000	0.000	0.000
1	2	0.000	0.000	0.639	0.263
2	2	0.413	0.000	0.000	0.000
Family	**Machine**	**R. B. 5**	**R. B. 6**	**R. B. 7**	**R. B. 8**
0	0	0.493	0.000	0.000	0.000
1	0	0.000	0.000	0.000	0.000
2	0	0.241	0.000	0.262	0.503
0	1	0.000	0.000	0.000	0.000
1	1	0.235	0.000	0.272	0.249
2	1	0.000	0.000	0.000	0.000
0	2	0.000	0.000	0.000	0.000
1	2	0.000	0.000	0.000	0.000
2	2	0.000	0.000	0.000	0.000

Production quantities in Table 4.12 can be transformed into time spent for production. For this transformation, we shall multiply the production quantities with c_{fp} values given in Table 4.2. Production times are given in Table 4.13.

Capacities of resource time buckets are either spent for production or for setup if not idle. Setup times after the first run of the LP are given in Table 4.14.

Table 4.13. Production times after the first run of the LP

Family	**Machine**	**R. B. 1**	**R. B. 2**	**R. B. 3**	**R. B. 4**
0	0	2.739	5.935	11.681	11.282
1	0	0.000	0.000	0.000	0.000
2	0	14.721	0.000	0.000	0.000
0	1	0.000	0.000	0.000	0.000
1	1	2.945	1.090	11.681	11.282
2	1	13.924	0.000	0.000	0.000
0	2	0.000	0.000	0.000	0.000
1	2	0.000	0.000	11.681	4.813
2	2	7.598	0.000	0.000	0.000
Family	**Machine**	**R. B. 5**	**R. B. 6**	**R. B. 7**	**R. B. 8**
0	0	8.099	0.000	0.000	0.000
1	0	0.000	0.000	0.000	0.000
2	0	4.437	0.000	4.825	9.256
0	1	0.000	0.000	0.000	0.000
1	1	4.166	0.000	4.825	4.411
2	1	0.000	0.000	0.000	0.000
0	2	0.000	0.000	0.000	0.000
1	2	0.000	0.000	0.000	0.000
2	2	0.000	0.000	0.000	0.000

Table 4.14. Setup times after the first run of the LP

Family	**Machine**	**R. B. 1**	**R. B. 2**	**R. B. 3**	**R. B. 4**
0	0	2.934	0.000	0.000	0.000
1	0	0.000	0.000	0.000	0.000
2	0	11.012	0.000	0.000	0.000
0	1	0.000	0.000	0.000	0.000
1	1	0.000	4.845	0.000	0.000
2	1	14.538	0.000	0.000	0.000
0	2	0.000	0.000	0.000	0.000
1	2	8.421	5.935	0.000	6.469
2	2	15.387	0.000	0.000	0.000
Family	**Machine**	**R. B. 5**	**R. B. 6**	**R. B. 7**	**R. B. 8**
0	0	0	0	0	0
1	0	0	0	0	0
2	0	7.295	3.717	0	0
0	1	0	0	0	0
1	1	4.845	0	0	4.845
2	1	10.820	3.717	0	0
0	2	0	0	0	0
1	2	7.886	0.275	4.825	9.256
2	2	11.945	3.442	0	0

As seen in Table 4.13 and Table 4.14, there is no idle time in the system. However, there is no production in family time bucket 1 (which includes resource time buckets 1 and 2) and family time bucket 3 (which includes resource time buckets 6, 7 and 8) of family 1 on machine 2, although, three setups are the model assumed three setups. According to the algorithm in Phase - II, we shall remove those unnecessary setups and run the LP again. In the following runs, unnecessary setups will be eliminated, thus, a higher quantity of families would be able to be produced.

Phase - II stops when two consecutive runs of the LP give the same result, or maximum number of iterations (which is assumed to be 5 in this study) is reached. Third and forth runs of the LP give the same result, thus, the algorithm stops after the forth run of the LP. Optimum objective value is 269.796 in the first run, 120.161 in the second run and 119.963 in the third and forth run. Final production quantities are given in Table 4.15, production times are given in Table 4.16 and setup times are given in Table 4.17.

In Phase - III, a feasible schedule is generated using the production quantities given in Table 4.15. Firstly, production quantities in different resource time buckets which belong to the same family time bucket are combined. These combined quantities are referred to as *"batches"*. Thirdly, EDD rule is applied to the batches on each machine.

Family time bucket 1 of family 0 includes resource time buckets 1, 2 and 3. Thus, production quantities in resource time buckets 1, 2 and 3 needs to be combined in order to form the first batch of family 0 on each machine. Other production quantities are also combined in the same way. Total production quantities for each family, machine and family time bucket is given in Table 4.18. Family time buckets are written as "F. B." shortly.

In order to apply EDD rule to the batches, we simply sequence the family time buckets with non-decreasing orders of the ending points of the family time buckets. This sequence is given in Table 4.19. Production on each machine should be done according

Table 4.15. Values of the production variables after the last run of the LP

Family	**Machine**	**R. B. 1**	**R. B. 2**	**R. B. 3**	**R. B. 4**
0	0	1.104	0.361	0.000	0.631
1	0	0.000	0.000	0.000	0.000
2	0	0.561	0.000	0.000	0.000
0	1	0.000	0.000	0.000	0.000
1	1	1.646	0.062	0.386	0.637
2	1	0.124	0.000	0.000	0.000
0	2	0.000	0.000	0.000	0.000
1	2	0.000	0.000	0.639	0.617
2	2	1.708	0.000	0.000	0.000
Family	**Machine**	**R. B. 5**	**R. B. 6**	**R. B. 7**	**R. B. 8**
0	0	0.000	0.000	0.000	0.000
1	0	0.000	0.000	0.000	0.000
2	0	1.077	0.000	0.262	0.503
0	1	0.000	0.000	0.000	0.000
1	1	1.119	0.000	0.209	0.522
2	1	0.000	0.000	0.000	0.000
0	2	0.000	0.000	0.000	0.000
1	2	0.000	0.000	0.000	0.000
2	2	0.000	0.000	0.000	0.000

Table 4.16. Production times after the last run of the LP

Family	**Machine**	**R. B. 1**	**R. B. 2**	**R. B. 3**	**R. B. 4**
0	0	18.133	5.935	0.000	10.359
1	0	0.000	0.000	0.000	0.000
2	0	10.340	0.000	0.000	0.000
0	1	0.000	0.000	0.000	0.000
1	1	29.151	1.090	6.836	11.282
2	1	2.256	0.000	0.000	0.000
0	2	0.000	0.000	0.000	0.000
1	2	0.000	0.000	11.681	11.282
2	2	31.407	0.000	0.000	0.000
Family	**Machine**	**R. B. 5**	**R. B. 6**	**R. B. 7**	**R. B. 8**
0	0	0.000	0.000	0.000	0.000
1	0	0.000	0.000	0.000	0.000
2	0	19.831	0.000	4.825	9.256
0	1	0.000	0.000	0.000	0.000
1	1	19.831	0.000	3.697	9.256
2	1	0.000	0.000	0.000	0.000
0	2	0.000	0.000	0.000	0.000
1	2	0.000	0.000	0.000	0.000
2	2	0.000	0.000	0.000	0.000

Table 4.17. Setup times after the last run of the LP

Family	**Machine**	**R. B. 1**	**R. B. 2**	**R. B. 3**	**R. B. 4**
0	0	2.934	0.000	0.000	0.000
1	0	0.000	0.000	0.000	0.000
2	0	0.000	0.000	6.372	0.923
0	1	0.000	0.000	0.000	0.000
1	1	0.000	4.845	4.845	0.000
2	1	0.000	0.000	0.000	0.000
0	2	0.000	0.000	0.000	0.000
1	2	0.000	0.000	0.000	0.000
2	2	0.000	0.000	0.000	0.000
Family	**Machine**	**R. B. 5**	**R. B. 6**	**R. B. 7**	**R. B. 8**
0	0	0.000	0.000	0.000	0.000
1	0	0.000	0.000	0.000	0.000
2	0	0.000	3.717	0.000	0.000
0	1	0.000	0.000	0.000	0.000
1	1	0.000	3.717	1.128	0.000
2	1	0.000	0.000	0.000	0.000
0	2	0.000	0.000	0.000	0.000
1	2	14.356	0.000	0.000	0.000
2	2	0.000	0.000	0.000	0.000

Table 4.18. Total production quantities in family time buckets

Family	**Machine**	**F. B. 1**	**F. B. 2**	**F. B. 3**
0	0	1.465	0	
1	0	0	0	0
2	0	0.561	1.077	0.765
0	1	0	0	
1	1	1.707	2.142	0.731
2	1	0.124	0	0
0	2	0	0	
1	2	0	1.256	0
2	2	1.708	0	0

to the sequence in that table. For example, on machine 0, first batch of family 2 should be produced first. After that, first batch of family 1 should be produced. However, as seen in Table 4.18, quantity of batch 1 of family 1 on machine 0 is zero. For this reason, we skip that batch and continue the production on machine 0 with the first batch of family 0. Productions on other machines are calculated the same way. Table 4.20 shows the production sequences on each machine. Productions should be done in the given order.

Table 4.19. EDD order of batches

Order	**Family**	**Batch**
1	2	1
2	1	1
3	0	1
4	0	2
5	1	2
6	2	2
7	2	3
8	1	3

After productions are scheduled in the order given in Table 4.20, we shall insert

Table 4.20. Production sequences on machines

Machine	Quantity	Family
0	0.561	2
0	1.465	0
0	1.841	2
1	0.124	2
1	4.580	1
2	1.708	2
2	1.256	1

setup times between consecutive productions of different families. Then, we will have a feasible schedule. However, the schedule is still not complete. There is a production quantity for family 2 on machine 0 at resource time bucket 8. This production should be added to the end on machine 0. Moreover, total production is still lower than total quantity ordered. We shall add 1.378 units of family 1 and 3.076 units of family 2 at the end of the schedule. Generated schedule is given in Table 4.21. The schedule can be represented with Figure 4.5, Figure 4.6 and Figure 4.7.

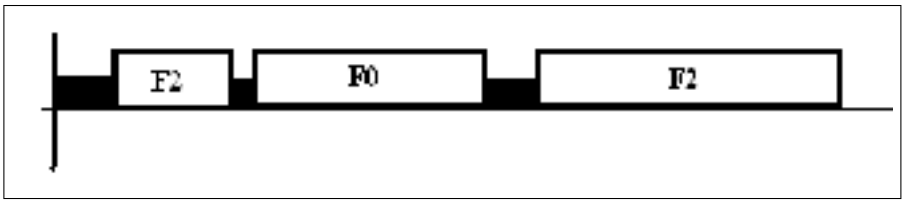

Figure 4.5. Gantt chart for machine 0

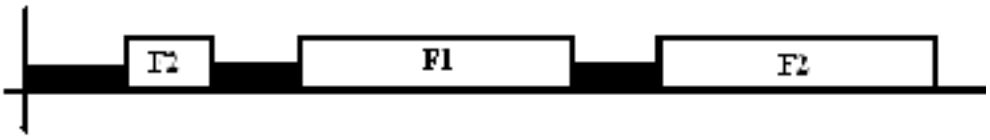

Figure 4.6. Gantt chart for machine 1

After the schedule is generated, weighted tardiness all jobs of each family in EDD order. For family 0, first job is job 2 with a due date of 49.022. Quantity of job 2 is 1.788 and this quantity is provided by machine 0 at 53.662. Thus, weighted tardiness of job

Table 4.21. Final schedule generated for PPS=4 and AP=20

Machine	**Family**	**Start**	**Finish**	**Quantity**
0	2	11.012	21.352	0.561
0	0	24.286	53.662	1.788
0	0	53.662	64.022	0.631
0	2	75.034	94.865	1.0767
0	2	94.865	99.690	0.262
0	2	99.690	108.946	0.503
0	2	108.946	127.832	1.025
0	2	127.832	146.718	1.025
1	2	14.538	16.794	0.124
1	1	21.638	51.879	1.707
1	1	51.879	89.828	2.142
1	1	89.828	102.782	0.731
1	1	102.782	110.923	0.460
1	1	110.923	119.065	0.460
1	2	133.603	152.279	1.025
2	2	15.387	46.794	1.708
2	1	61.150	84.113	1.256
2	1	98.469	111.076	0.689
2	1	111.076	119.481	0.460

Figure 4.7. Gantt chart for machine 2

2 is $1.788 * (53.662 - 49.022) = 8.298$. Weighted tardiness for other jobs are calculated similarly. Weighted tardiness of all jobs is shown in Table 4.22. Total tardiness is calculated as 412.123.

Table 4.22. Weighted tardiness of jobs

Job	**Family**	**Quantity**	**Due date**	**Finish**	**Tardiness**
2	0	1.788	49.022	53.662	8.298
14	0	0.289	54.686	58.403	1.073
13	0	0.342	77.154	64.021	0
6	1	1.858	27.113	54.548	50.965
1	1	0.568	38.783	62.909	13.708
5	1	1.045	53.011	72.312	20.168
11	1	1.634	80.136	89.829	15.840
12	1	1.136	94.223	105.876	13.241
9	1	1.663	97.934	119.481	35.832
8	2	0.824	10.578	19.644	7.467
7	2	1.153	31.407	36.635	6.031
10	2	1.822	66.704	97.537	56.172
0	2	0.988	79.925	115.049	34.720
4	2	1.889	83.853	141.156	108.220
3	2	0.635	88.678	152.279	40.388
Total					412.123

5. EXPERIMENTATION

In order to test the performance of the proposed solution, we need a set of problem instances. In this chapter, we give five factors to consider while generating a problem instance. After defining the levels of the factors, we explain how we generate the problem instances. At the end, numerical results are given and comments are made based on the numerical results.

5.1. Problem Generation

There are several factors to consider while generating a problem instance. These are:

- Number of machines
- Number of families
- Due date structure
- Setup time
- Eligibility constraints

In order to see the performance of the proposed heuristic under different levels of the factors, an experimental design below is prepared. Factors and levels are as follows:

- Number of machines : 5 or 10
- Number of families : 10 or 20
- Setup time per batch / Processing time per order : U(0, 1) or U(0, 2)
- Number of eligible machines for each family : U(0.2, 0.8)*(number of machines) or U(0.4, 0.6)*(number of machines)
- Due date: Triangular(0, 25, 100) or U(0, 100) or Triangular(0, 75, 100)

Other parameters are generated commonly as follows:

- Number of jobs : 100
- Quantity of each job: U(0, 2*number of machines/5)
- Time needed to produce a unit each family on each machine: U(4, 5)

The parameters are chosen in order to make the problem instances meaningful. Assuming there are 5 machines, thus quantity of a job will be expected to be 1, and total time needed to produce all the jobs will be at least 400, at most 500. Assuming the production is divided to machines equally, total time needed to produce all jobs will be between 80 and 100 on each machine. Since the time horizon, which the due dates of jobs are spread over, is between 0 and 100, it will be enough for the time needed for production, ignoring the setup time.

Ten problem instances are randomly generated for each level combination of factors described above. That means there are 48 level combinations of factors and 480 problem instances in total. With the solutions of those 480 instances, the effect of the factors pointed above on the performance of the proposed solution will be analyzed.

5.2. Implementation of the Algorithm

As mentioned before, the proposed solution is composed of four phases: *(i)* aggregation phase, *(ii)* aggregate planning phase, *(iii)* schedule generation phase and *(iv)* tuning phase. Total tardiness of the final schedule mainly depends on two factors: *(i)* starting point of the search and *(ii)* steps to be used in the search. A constant starting point can be used in the search algorithm. However, it can also be selected randomly. On the other hand, steps to be used in the search can be either constant or changing.

Proposed solution explained in the previous chapters is implemented as four heuristic methods HM-1, HM-2, HM-3 and HM-4. These heuristic methods are as follows:

- HM-1: One-phase search with a constant starting point
- HM-2: Two-phase search with a constant starting point

- HM-3: One-phase search with the best of four random points being the starting point
- HM-4: Two-phase search with the best of four random points being the starting point

Let m be the number of machines, F be the number of families and S be the average setup time. Constant starting points are decided to be $PPS = 4$ and $AP = 5SF/m$. PPS value is decided to be 4 because the problem instances are generated such that production of all the orders takes at least 80% of the time horizon. AP value is chosen to be consistent with PPS value. Four random (PPS, AP) points are chosen from the plane of $(0, 0) - (8, 40)$ for HM-3 and HM-4.

One-phase search is done with the parameters Δ_{PPS} and Δ_{AP} being 0.5 and $5SF/8m$ respectively. One-phase search is used in HM-1 and HM-3. For HM-2 and HM-4, two-phase search is used. In two-phase search, the parameters Δ_{PPS} and Δ_{AP} are chosen as 1 and $5SF/4m$ respectively for the long-distance search. In the short-distance search, those parameters are chosen as 0.25 and $5SF/16m$.

5.3. Numerical Results

As mentioned before, 480 different problem instances are generated, and labeled between 0 and 479. In order to compare the performance of HM-1, HM-2, HM-3 and HM-4, one for each of ten problems is used (Problem 0, Problem 10, ... , Problem 470). The results are given in the Table A.1. Considering the average values found for 48 problems, HM-2 gives %10, HM-3 gives %22 and HM-4 gives %24 lesser result compared to HM-1. The minimum result is found by HM-1 for 8% of the problems while by HM-2 for 19%, by HM-3 for 31% and by HM-4 for 54% of the problems. With these data, HM-4 seems to give the best results.

Average computation times for different heuristic methods are also different. HM-1 takes 3.07 minute, HM-2 takes 9.87 minutes, HM-3 takes 14.22 minutes and HM-4 takes 23.18 minutes in average. The decision about which method to use for all of the

480 instances shall be made considering both the performance and the computation time of the heuristic method. HM-1 takes the least time compared to others while HM-4 gives the best result. We decided to use HM-4 because 23.18 minute is still a fairly small amount of time for a problem instance to solve. Results of all 480 problems are given in Table A.2.

5.3.1. Comparison of HM-4 and CPLEX Solutions

After getting the results for HM-4, we decided to compare these results with the solutions of mathematical model given in Section 3.2. That mathematical model is coded in CPLEX and ran for 12 problem instances (Problems labeld as 0, 10, ... , 100, 110), each for 24 hours. The comparison of the results shows that HM-4 gives better results than the mathematical model which seeks for the optimum result. Comparison of the results are given in Table 5.1.

Table 5.1. Comparison of the proposed heuristic and mathematical model solutions

Problem No	HM-4	Mathematical Model
0	1002.790	3188.132
10	526.785	2462.108
20	150.877	1913.590
30	907.443	2087.226
40	976.933	3607.862
50	205.780	1748.765
60	247.730	4820.400
70	854.701	2516.123
80	145.508	3321.661
90	820.870	5257.623
100	536.310	2373.129
110	650.263	4264.230

Mathematical model which is seeking optimal solution could not be ran for all problem instances because of lack of time. However, the model is applied to one

problem instance for all level combinations of the factors of due date, setup time and eligibility constraints. Table 5.2 shows the average ratios of the results of HM-4 to mathematical model for different levels of factors.

Table 5.2. Average ratios of HM-4 to mathematical model

Levels	Setup	Eligibility	Due Date
1	0.238	0.174	0.239
2	0.162	0.226	0.263
3			0.098

According to the values in table 5.2, we can say that HM-4 outperforms mathematical model when due date level is 3, which means, distribution of the due dates of jobs is triangular distribution with parameters (0, 75, 100). In other words, when due dates are right-skewed, the proposed heuristic performs better.

5.3.2. Relationship Between the Number of Batches and the Objective Values

Problems including five machines (problems labeled from 0 to 239) are used in order to see if there is any relationship between the number of batches in the solution and the objective value found. Table A.3 shows the number of batches for the first 240 problem instances. Figure 5.1 is a scatter graph for these values. The graph do not gives a clear idea about the relationship between the number of batches and the objective value. Slope is calculated to be 4.35 and correlation coefficient is calculated to be 0.05. Either of these numbers are not enough to make a judgment about the relationship.

Figure 5.2 and 5.3 represents the changes in the number of batches during the search procedure for problems 468 and 469 respectively. Considering these graphs, a small amount of decrease can be claimed during the search procedure.

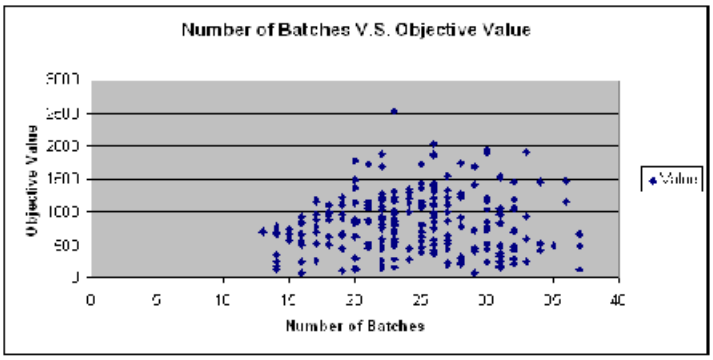

Figure 5.1. Relationship between the number of batches and the objective values of the solutions

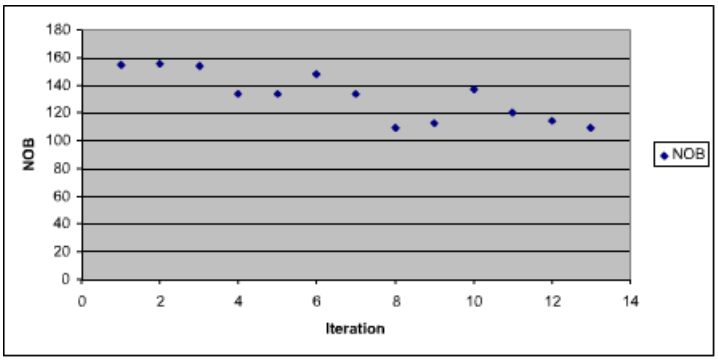

Figure 5.2. Number of batches for problem 468

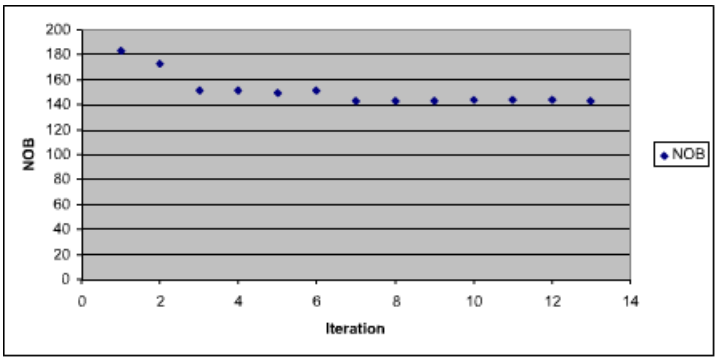

Figure 5.3. Number of batches for problem 469

5.3.3. Effects of Factors

Problem set is generated considering different levels of five such as machine factor, family factor, setup factor, eligibility factor and due date factor. Table 5.3 summarizes the levels of the factors.

Table 5.3. Factors and levels

Levels	MF	FF	SF	EF	DF
1	5	10	U(0, 1)	U(0.2, 0.8)*(# of machines)	Triangular(0, 25, 100)
2	10	20	U(0, 2)	U(0.4, 0.6)*(# of machines)	U(0, 100)
3					Triangular(0, 75, 100)

Average results for different levels of these factors may give some insight about the performance of the proposed solution. Table 5.4 shows the average results of HM-4 for different levels of the factors. According to these average results, proposed solution performs better when number of machines is smaller, number of families is smaller and due dates are right-skewed. There is not a significant difference between the levels of setup factor and eligibility factor.

Table 5.4. Average results of HM-4 for different levels of factors

Levels	MF	FF	SF	EF	DF
1	934.389	1353.615	1437.968	1478.790	1907.862
2	1990.729	1571.503	1487.150	1446.328	1769.411
3					710.404

The fact that proposed solution performs better when due dates are right-skewed was expected. The reason is that earlier due dates require a large number of batches at the beginning of the scheduling period. Expectedly, this adds additional setup to the system and possibly makes the successive jobs tardy.

The average result for level 2 of machine factor is more than twice of the average result for level 1. A ratio of 2 was expected between the levels of machine factor. The reason is that quantities of jobs are directly related with the number of machines.

However, we can claim that proposed solution perform worse when the number of machine increases, regardless of the increase in job quantities, since this ratio is more than 2.

On the other hand, number of families has a slight effect on the performance of the proposed solution. The result is better when the number of families is lesser. Again this result was expected, because, small values of the number of families means small amount of setup time.

Eligibility factor does not affect the performance of the proposed solution. However, it is interesting that; setup factor does not seem to affect the performance either. There is a very slight difference between levels. The reason may be the fact that numbers of batches are very small. As a result, only the due date factor seems to have a significant effect on the performance of the proposed heuristic.

5.3.4. Search Space

Phase - IV tunes for better values of two control parameters. In order to see the performance of the search algorithm, three problem instances is solved for 400 different values of the control parameters. Problems 0, 10 and 80 are used. Solutions on the search space for these instances are presented in Figure 5.4, Figure 5.5 and Figure 5.6.

As seen in the figures, search space is very floating. Better values can be found either in the middle or near the borders. The minimum and the average values are summarized in Table 5.5.

According to the data above, tuning phase of the proposed heuristic found a better solution than the minimum of 400 points on the search space for two of three instances. For the other instance, tuning phase found a very close value to the minimum of 400 points. During the tests, HM-4 searched 16 points in average for all problem

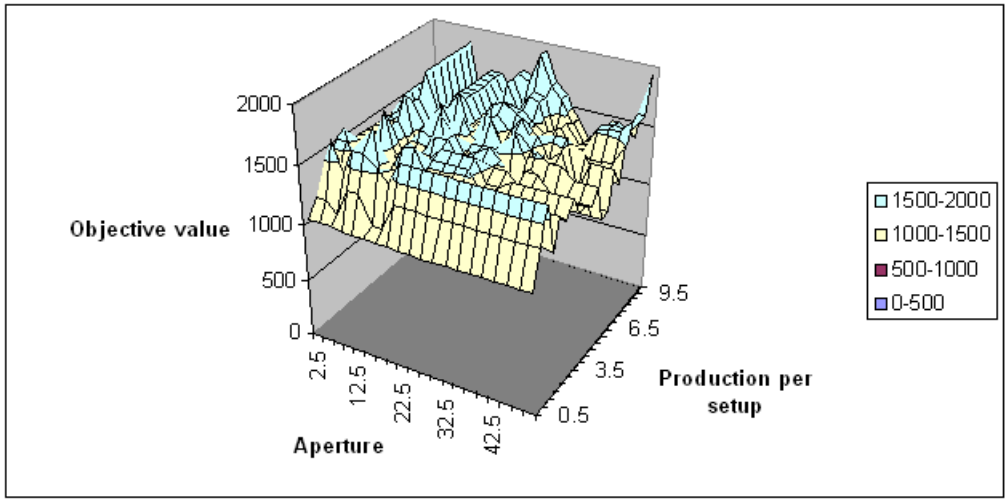

Figure 5.4. Search space of problem 0

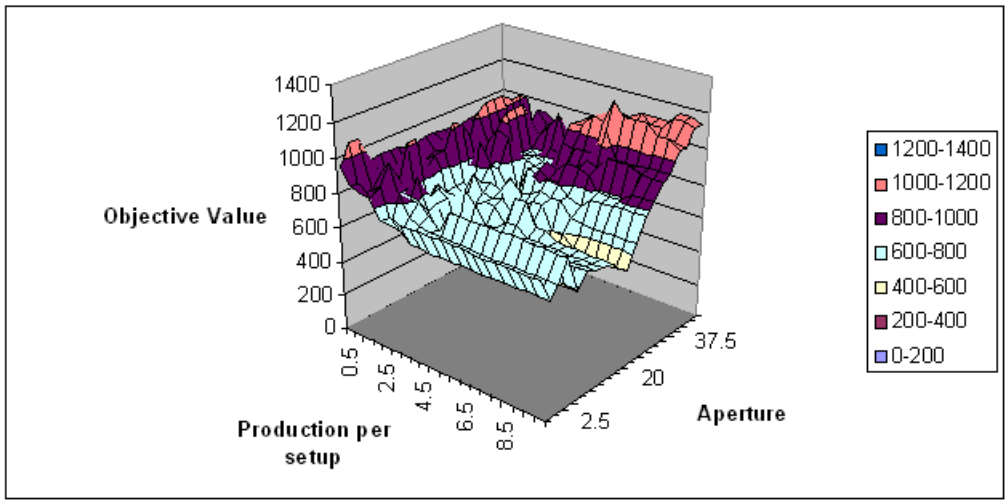

Figure 5.5. Search space of problem 10

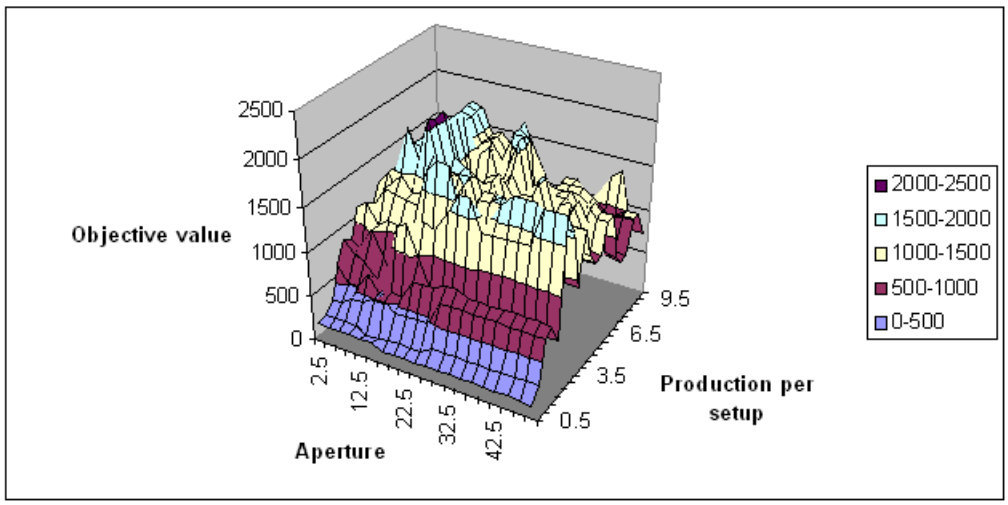

Figure 5.6. Search space of problem 80

Table 5.5. Search space statistics

Problem No	Minimum	Average	HM-4
0	1032.94	1447.073	1002.79
10	523.806	796.383	526.785
80	145.513	1067.576	145.508

instances. As a result, we can claim that tuning phase, in HM-4, performs well in a reasonable amount of time.

6. CONCLUSIONS

Increasing the performance of manufacturing systems is one of the major application areas of operational research. Performance of a manufacturing system is usually represented by utilization and efficiency. An important factor which makes it harder to achieve higher efficiency is product variety. Those systems which deliver different products to different customers at different times and with different quantities need to use their limited capacity in an efficient way in order to meet the due dates of the orders. This is the point where scheduling begins.

Manufacturing systems which produce different products may have shared non-consumable resources for those products. In that situation, resource allocation plays an important role for the efficiency of the system. Moreover, if there are multiple orders for each product, sequencing and resource allocation should be done simultaneously. This case is a very common case in many industries.

In this study, we analyzed a special case for the problem above. In the focused problem, there are several parallel machines with different technological properties in order to produce different customer orders for different product families. Machines can produce two products of the same family without any setup in between. There is a setup time between productions of two families on the same machine consecutively. This setup is referred to as family setup and is dependent to the latter family and the machine. Machine properties define the families which can be produced on each machine. On the other hand, every family is produced with different speeds on each eligible machine.

Scheduling literature includes numerous studies which focus on problems similar to the problem at hand. However, there is no study which focuses on the exact problem. The problem at hand is challenging because of eligibility constraints, job-splitting property and family setup structure. Most of the solution approaches in literature cannot cope with these challenging issues.

In scheduling problems with tardiness related objectives, it is hard to seek optimal solutions while generating the schedule step-by-step. This is because scheduling a job may possibly make other jobs tardy. In an unrelated parallel machine environment, it is even harder because of different production speeds of jobs on machines. Eligibility constraints, job-splitting property and family setup structure make the problem even more complex. Evolutionary heuristics could be applied but would be challenging because, representing a feasible solution and defining a neighborhood structure are difficult.

The problem can be represented by a MIP formulation. This formulation is given in Chapter 3. A large number of decision variables are necessary in order to model the problem. Moreover, some of these decision variables would be binary variables. For a problem instance with 3 machines and 10 jobs, there would be 440 decision variables and 90 of them would be binary variables.

We proposed a four phased heuristic with four different application methods for the problem. The first three phases generate a schedule given two control parameters, and Phase - IV tunes the control parameters. Four application methods are different from each other in terms of starting point and steps.

The proposed heuristic starts with aggregating jobs with closer due dates in Phase I. Jobs are aggregated in order to provide enough quantity considering setup. Another benefit of the aggregation is that it reduces the number of jobs as well as the number of decision variables. Only consecutive jobs are aggregated and two control parameters determine how many jobs will be aggregated. These control parameters are production-per-setup (PPS) and aperture (AP). Production-per-setup determines the quantity which is enough for aggregation. On the other hand, aperture determines if the differences of the due dates of jobs are acceptable for aggregation.

Tuning phase, Phase - IV, tries to find a good combination of the control parameters which will lead to better schedules. In other words, tuning phase runs the first three phases with different values of the control parameters. Heuristic starts with the

initial values of the control parameters at the beginning. After that, tuning phase looks if an increase or a decrease in the parameters by a constant step leads to a better result. The search stops if either an increase or a decrease in the parameters does not lead to a better schedule. Four different methods (HM-1, HM-2, HM3 and HM-4), are proposed for Phase - IV. All of the methods are the same except the starting points and steps. HM-1 starts with a constant starting point and uses a one phase search. HM-2, on the other hand, chooses 4 random points and selects the best one as the starting point of a one phase search. HM-3 and HM-4 are the same as HM-1 and HM-2 respectively, except they use a two phase search. Two phase search is basically searching twice with higher steps at first and with lower steps after. Search with lower steps starts from the ending point of the previous phase.

In order to implement the proposed heuristic and investigate its performance, a problem set is generated. According to the test results, HM-4 seems to perform best but takes more computation time. After that, the results of HM-4 for 12 instances are compared with the best result found in 24 hours by the run of mathematical model in CPLEX.

Average computation time for HM-4 is 23 minutes. Moreover, the HM-4 over performed in all of the 12 problem instances which used in the comparison. The ratio of the objective values of HM-4 and CPLEX is nearly 20%. That means proposed heuristic gives fairly good results in a small amount of computation time.

Finally, all of the 480 problem instances in the problem set is solved with HM-4 and the results are presented in Table A.2. Average of these results for different levels of factors are also summarized in Table 5.4. According to these average values, only due date factor has an important effect on the performance of the proposed heuristic. Moreover, the performance gets slightly worse with more machines, families and setup time.

The proposed methodology can be modified in the further studies. These modifications may be about the first three phases or the tuning phase. In Phase - I, different

ways for aggregation can be tested. In this study, aggregation is done straightforward. Backward aggregation or other possible methods can be tested and provide an improvement in the performance.

Furthermore, stopping criteria of the aggregation can be modified. Instead of stopping whenever either of the parameters are exceeded, a different criterion could possibly give better results. However, a large amount of modifications can be suggested and each modification would need additional tests.

Tuning phase tries to find the better values of the control parameters. However, looking a large amount of possible values can be very time consuming. On the other hand, the surface of the search space is highly non-linear, as seen in Figure 5.4, Figure 5.5 and Figure 5.6. This makes it hard to find good values in the valleys. Threshold acceptance or simulated annealing methods can be applied in this tuning phase. Setting the search steps dynamically may also be a good idea and can improve the performance.

In this study, the control parameters used in the aggregation are common for all families. Using different control parameters for each family, or developing a meta-heuristic may be considered in future researches. Moreover, the aggregate planning approach used in this study can be applied to other scheduling problems with different objectives like T_{max}.

We do not have the optimum results of the instances in the problem set we generated. Finding the optimum values is very time consuming because of the huge amount of binary variables. Efficient branch and bound techniques or cut mechanisms can possibly decrease the computation time. Also, lower bound calculations would be very beneficial in order to evaluate the proposed solution in this study and other solutions to be proposed in the future.

To sum up, we focused on a challenging scheduling problem in this study. The problem is challenging because of the unrelated parallel machine environment, job-splitting property, eligibility constraints and family setup structure. We proposed a

heuristic based on the aggregate planning approach. 480 test problems are generated in order to see the performance of the proposed solution. Mathematical model of the problem is also solved for several problems in the problem set using CPLEX. However, this process is very time consuming because of a huge amount of decision variables, especially binary variables. For this reason, we put a limit of 24 hours on the CPU runtime and get the best result found so far.

The results of CPLEX and the proposed solution are compared. According to the results, the proposed solution performs well in a fairly small amount of time. Optimum values of the problem instances and possible modifications of some aspects of the proposed solution is left open for further studies.

APPENDIX A: RESULTS OF THE PROPOSED HEURISTIC

Table A.1: Comparison of the heuristic methods

Problem No	**HM-1**	**HM-2**	**HM-3**	**HM-4**
0	1274.83	1185.59	1242.01	**1002.79**
10	562.733	563.731	625.344	**526.785**
20	163.28	**68.2344**	174.248	150.877
30	921.127	921.127	**876.054**	907.443
40	924.619	940.977	**914.812**	976.933
50	329.816	248.334	250.732	**205.78**
60	1686.49	1686.49	**247.73**	**247.73**
70	**709.693**	**709.693**	770.494	854.701
80	894.161	913.721	165.325	**145.508**
90	1219.58	825.166	823.484	**820.87**
100	929.273	985.634	950.84	**536.31**
110	**392.358**	**392.358**	512.325	650.263
120	2129.92	2113.22	1591.88	**1151.48**
130	**1319.04**	**1319.04**	1399.44	1373.58
140	501.912	564.524	**479.464**	500.937
150	1324.86	**1107.59**	1170.88	1143.62
160	1327.56	1316.33	**1115.13**	1433.78
170	526.233	422.735	195.318	**119.575**
180	1494.08	1056.67	1271.13	**938.42**
190	1493.68	**1352.65**	1947.53	1955.51
200	784.475	784.475	179.088	**177.421**
210	1256.86	722.451	**347.055**	1429.59
220	**1147.09**	1209.91	1257.21	1729.71
230	1395.4	495.953	**342.781**	467.882

Continued on next page

Table A.1 – continued from previous page

Problem No	HM-1	HM-2	HM-3	HM-4
240	3000.78	2995.51	**2714.13**	**2714.13**
250	1926.97	1926.97	**1669.05**	1848.17
260	1104.84	1104.84	1104.84	**1026.65**
270	2492.31	2492.31	**2065.03**	2147.95
280	2150.9	1938.13	**1633.63**	1893.29
290	1277.94	721.945	534.37	**337.237**
300	3020.77	3020.77	**2133.63**	2298.11
310	1959.18	2104.02	1413.13	**973.884**
320	1438.81	1326.97	962.812	**757.838**
330	2363.99	2982.77	**1264.39**	2778.61
340	2706.39	2773.11	2236.47	**2116.17**
350	1995.28	2021.69	1411.75	**1000.88**
360	3848.07	**3588.64**	3743.92	3723.32
370	2511.08	**2042.97**	2179.66	2140.13
380	1238.09	1238.09	1386.09	**456.543**
390	3257.17	3257.17	3249.55	**3233.3**
400	1228.74	927.184	**875.024**	**875.024**
410	940.23	797.529	801.569	**557.982**
420	3054.22	2361.42	2079.94	**1730.37**
430	1849.5	1843.13	**1343.96**	1864.48
440	1685.37	2750.56	2097.1	**1221.27**
450	1770.41	798.295	767.893	**591.64**
460	3351.34	2328.09	2249.83	**2207.05**
470	2435.07	**441.975**	1687.45	721.337
Average	1610.760833	1451.889404	1259.490042	**1222.142917**

Table A.2: All results for HM-4

Problem No	**Result**	**Problem No**	**Result**	**Problem No**	**Result**
0	1002.79	1	1218.27	2	1355.91
3	1038.74	4	1411.6	5	1362.03
6	1022.54	7	1585.49	8	1292.27
9	1443.98	10	526.785	11	576.291
12	1185.02	13	588.056	14	1071.88
15	713.524	16	1909.21	17	1012
18	1222.8	19	781.134	20	150.877
21	290.119	22	294.195	23	771.764
24	45.1641	25	775.826	26	218.267
27	315.851	28	79.2786	29	351.298
30	907.443	31	858.559	32	848.85
33	2136.3	34	1412.05	35	1437.32
36	1674.09	37	844.627	38	1204.32
39	1270.8	40	976.933	41	967.781
42	701.682	43	874.827	44	969.984
45	485.344	46	894.933	47	1255.05
48	1220.68	49	890.703	50	205.78
51	263.764	52	384.63	53	429.705
54	671.783	55	422.929	56	487.718
57	363.691	58	204.029	59	640.007
60	247.73	61	1113.77	62	1692.23
63	267.691	64	555.68	65	953.751
66	1139.61	67	1054.57	68	253.554
69	995.834	70	854.701	71	1377.72
72	946.809	73	1208.74	74	1224.56
75	516.852	76	1634.09	77	827.297
78	1357.7	79	617.214	80	145.508

Continued on next page

Table A.2 – continued from previous page

Problem No	Result	Problem No	Result	Problem No	Result
81	93.4391	82	146.301	83	223.352
84	1318.24	85	986.981	86	1014.42
87	761.432	88	526.851	89	303.087
90	820.87	91	180.264	92	212.504
93	1479.77	94	799.223	95	250.346
96	730.401	97	1278.41	98	300.752
99	1872.21	100	536.31	101	217.946
102	1416.74	103	1565.23	104	1817.68
105	635.374	106	1412.61	107	1102.23
108	1009.14	109	723.749	110	650.263
111	523.642	112	445.8	113	622.442
114	366.553	115	124.582	116	1278.93
117	190.886	118	376.979	119	622.459
120	1151.48	121	1561.23	122	2611.99
123	420.337	124	492.744	125	2525.81
126	1630.64	127	1502.95	128	640.72
129	1948.78	130	1373.58	131	1679.48
132	1355.56	133	1450.27	134	1860.84
135	1231.24	136	1777.27	137	2464.76
138	909.206	139	1893.43	140	500.937
141	860.708	142	95.1226	143	285.288
144	1311.59	145	211.628	146	692.625
147	95.0029	148	555.265	149	657.256
150	1143.62	151	2067.64	152	2001.99
153	1730.4	154	1399.92	155	721.637
156	1487.63	157	1306.84	158	1604.33
159	1652.23	160	1433.78	161	1561.24
162	1094.05	163	1523.27	164	1752.71

Continued on next page

Table A.2 – continued from previous page

Problem No	Result	Problem No	Result	Problem No	Result
165	946.213	166	1071.27	167	1289.3
168	1453.5	169	1780.06	170	119.575
171	87.8306	172	911.381	173	1115.76
174	467.453	175	239.509	176	1476.85
177	381.098	178	624.805	179	468.603
180	938.42	181	667.691	182	573.028
183	515.737	184	534.453	185	704.157
186	912.713	187	228.565	188	1831.4
189	2495.03	190	1955.51	191	271.979
192	1163.5	193	1944.19	194	367.259
195	277.776	196	1714.82	197	1504.23
198	592.023	199	2962.71	200	177.421
201	122.239	202	1319.86	203	96.2539
204	822.229	205	306.463	206	152.349
207	95.2459	208	176.756	209	800.375
210	1429.59	211	1402.31	212	591.756
213	317.713	214	1792.19	215	1291.59
216	1698.29	217	234.434	218	1879.94
219	640.175	220	1729.71	221	180.942
222	356.401	223	570.009	224	333.443
225	2033.84	226	2169.97	227	2007.36
228	556.194	229	169.742	230	467.882
231	97.6863	232	127.753	233	584.666
234	1321.92	235	100.682	236	1347.71
237	1376.5	238	753.524	239	359.484
240	2714.13	241	2183.01	242	1611.01
243	2971.23	244	1765.69	245	2431.74
246	1709.68	247	1833.75	248	2614.05

Continued on next page

Table A.2 – continued from previous page

Problem No	Result	Problem No	Result	Problem No	Result
249	2458.42	250	1848.17	251	1326.22
252	846.929	253	2369.88	254	3698.35
255	1805.88	256	2908.3	257	2952.51
258	2985.46	259	2185.4	260	1026.65
261	400.757	262	602.363	263	1416.38
264	1343.07	265	332.27	266	324.291
267	1301.98	268	618.455	269	601.954
270	2147.95	271	2532.89	272	2113.85
273	2635.17	274	2311.31	275	2666.91
276	2531.78	277	2680.99	278	3180.58
279	2276.37	280	1893.29	281	1366.27
282	992.012	283	1971.78	284	1376.54
285	1778.06	286	1952.18	287	1155.16
288	1886.73	289	940.468	290	337.237
291	1447.01	292	631.738	293	1286.11
294	191.198	295	1039.51	296	729.305
297	811.061	298	595.095	299	1300.55
300	2298.11	301	1100.7	302	3419.65
303	570.983	304	2605	305	774.077
306	3227.35	307	3756.89	308	4558.2
309	3923.34	310	973.884	311	2801.95
312	1498.9	313	965.465	314	2919.46
315	3759.9	316	1521.33	317	4721.42
318	3154.59	319	2629.27	320	757.838
321	617.124	322	357.478	323	2094.23
324	1772.37	325	1226.73	326	400.129
327	1509.7	328	858.911	329	359.404
330	2778.61	331	2792.98	332	572.94

Continued on next page

Table A.2 – continued from previous page

Problem No	Result	Problem No	Result	Problem No	Result
333	1428.39	334	1305.11	335	5032.56
336	2835.36	337	2427.65	338	2807.65
339	3637.06	340	2116.17	341	2512.6
342	2818.23	343	1670.08	344	3672.09
345	2313.46	346	3496.76	347	2495.06
348	2774.21	349	1545.92	350	1000.88
351	1723.54	352	572.941	353	546.647
354	2405.4	355	669.142	356	627.269
357	1793.83	358	602.568	359	1367.63
360	3723.32	361	3096.45	362	2703.76
363	3250.31	364	3421.22	365	1216.74
366	2854.8	367	4151.46	368	2452.25
369	2897.66	370	2140.13	371	3658.53
372	1865.03	373	3117.33	374	2060.03
375	3912.17	376	2980.62	377	1538.17
378	2229.52	379	1290.16	380	456.543
381	731.047	382	324.691	383	1018.25
384	378.691	385	1304.52	386	1173.28
387	349.778	388	1548.72	389	786.942
390	3233.3	391	3117.06	392	3656.02
393	2694.74	394	2824.15	395	2462.05
396	3389.18	397	2940.66	398	2435.7
399	2711.83	400	875.024	401	2896.86
402	2212.57	403	2602.09	404	726.603
405	2396.29	406	905.793	407	1556.93
408	2428.2	409	1987.06	410	557.982
411	622.768	412	474.403	413	2313.82
414	1302.21	415	774.37	416	1121.51

Continued on next page

Table A.2 – continued from previous page

Problem No	Result	Problem No	Result	Problem No	Result
417	1628.05	418	371.116	419	1133.27
420	1730.37	421	805.035	422	3405.85
423	3165.83	424	3633.18	425	4477.38
426	975.787	427	6476.31	428	6152.81
429	849.77	430	1864.48	431	3553.78
432	2020.9	433	2304.57	434	1747.71
435	2002.9	436	2391.12	437	2600.01
438	2408.57	439	2642.03	440	1221.27
441	1393.03	442	2057.36	443	2309.07
444	676.234	445	509.744	446	240.612
447	228.046	448	226.635	449	1939.86
450	591.64	451	1011.93	452	2834.22
453	2380.05	454	3999.35	455	2650.1
456	4573.75	457	1741.88	458	2226.23
459	1311.43	460	2207.05	461	2742.47
462	2281.19	463	3280.33	464	2876.95
465	2888.2	466	4801.86	467	5063.79
468	4093.35	469	4765.04	470	721.337
471	1081.49	472	535.469	473	949.474
474	792.157	475	931.237	476	911.042
477	165.865	478	611.689	479	336.154

Table A.3: Number of batches for problems between 0 and 239

Problem No	**NOB**	**Value**	**Problem No**	**NOB**	**Value**
0	24	1002.79	1	18	1218.27
2	21	1355.91	3	20	1038.74
4	27	1411.6	5	25	1362.03
6	31	1022.54	7	26	1585.49
8	30	1292.27	9	25	1443.98
10	27	526.785	11	29	576.291
12	27	1185.02	13	28	588.056
14	25	1071.88	15	29	713.524
16	22	1909.21	17	25	1012
18	22	1222.8	19	21	781.134
20	23	150.877	21	27	290.119
22	30	294.195	23	25	771.764
24	29	45.1641	25	30	775.826
26	22	218.267	27	28	315.851
28	26	79.2786	29	30	351.298
30	26	907.443	31	27	858.559
32	25	848.85	33	21	2136.3
34	24	1412.05	35	25	1437.32
36	28	1674.09	37	27	844.627
38	23	1204.32	39	22	1270.8
40	18	976.933	41	30	967.781
42	32	701.682	43	22	874.827
44	26	969.984	45	37	485.344
46	37	894.933	47	23	1255.05
48	28	1220.68	49	31	890.703
50	32	205.78	51	27	263.764

Continued on next page

Table A.3 – continued from previous page

Problem No	NOB	Value	Problem No	NOB	Value
52	25	384.63	53	30	429.705
54	22	671.783	55	26	422.929
56	31	487.718	57	24	363.691
58	28	204.029	59	27	640.007
60	14	247.73	61	17	1113.77
62	20	1692.23	63	15	267.691
64	16	555.68	65	18	953.751
66	19	1139.61	67	21	1054.57
68	19	253.554	69	21	995.834
70	20	854.701	71	20	1377.72
72	18	946.809	73	19	1208.74
74	26	1224.56	75	16	516.852
76	19	1634.09	77	20	827.297
78	21	1357.7	79	22	617.214
80	22	145.508	81	17	93.4391
82	27	146.301	83	20	223.352
84	14	1318.24	85	26	986.981
86	19	1014.42	87	19	761.432
88	22	526.851	89	14	303.087
90	16	820.87	91	13	180.264
92	17	212.504	93	15	1479.77
94	20	799.223	95	16	250.346
96	18	730.401	97	18	1278.41
98	14	300.752	99	17	1872.21
100	23	536.31	101	19	217.946
102	19	1416.74	103	22	1565.23
104	22	1817.68	105	20	635.374
106	23	1412.61	107	22	1102.23

Continued on next page

Table A.3 – continued from previous page

Problem No	NOB	Value	Problem No	NOB	Value
108	16	1009.14	109	15	723.749
110	19	650.263	111	21	523.642
112	25	445.8	113	23	622.442
114	23	366.553	115	20	124.582
116	22	1278.93	117	22	190.886
118	19	376.979	119	26	622.459
120	23	1151.48	121	20	1561.23
122	26	2611.99	123	31	420.337
124	24	492.744	125	23	2525.81
126	24	1630.64	127	26	1502.95
128	22	640.72	129	27	1948.78
130	26	1373.58	131	26	1679.48
132	27	1355.56	133	30	1450.27
134	29	1860.84	135	28	1231.24
136	32	1777.27	137	33	2464.76
138	32	909.206	139	31	1893.43
140	26	500.937	141	32	860.708
142	32	95.1226	143	34	285.288
144	32	1311.59	145	32	211.628
146	32	692.625	147	29	95.0029
148	35	555.265	149	31	657.256
150	27	1143.62	151	23	2067.64
152	34	2001.99	153	26	1730.4
154	32	1399.92	155	23	721.637
156	23	1487.63	157	24	1306.84
158	30	1604.33	159	25	1652.23
160	26	1433.78	161	32	1561.24
162	31	1094.05	163	36	1523.27

Continued on next page

Table A.3 – continued from previous page

Problem No	NOB	Value	Problem No	NOB	Value
164	21	1752.71	165	26	946.213
166	28	1071.27	167	30	1289.3
168	32	1453.5	169	36	1780.06
170	37	119.575	171	31	87.8306
172	31	911.381	173	27	1115.76
174	31	467.453	175	33	239.509
176	33	1476.85	177	28	381.098
178	34	624.805	179	31	468.603
180	16	938.42	181	22	667.691
182	14	573.028	183	19	515.737
184	22	534.453	185	17	704.157
186	17	912.713	187	23	228.565
188	26	1831.4	189	24	2495.03
190	30	1955.51	191	21	271.979
192	16	1163.5	193	19	1944.19
194	23	367.259	195	23	277.776
196	26	1714.82	197	22	1504.23
198	23	592.023	199	29	2962.71
200	14	177.421	201	23	122.239
202	23	1319.86	203	16	96.2539
204	23	822.229	205	32	306.463
206	24	152.349	207	20	95.2459
208	14	176.756	209	22	800.375
210	26	1429.59	211	25	1402.31
212	22	591.756	213	19	317.713
214	27	1792.19	215	24	1291.59
216	26	1698.29	217	17	234.434
218	20	1879.94	219	20	640.175

Continued on next page

Table A.3 – continued from previous page

Problem No	NOB	Value	Problem No	NOB	Value
220	28	1729.71	221	21	180.942
222	30	356.401	223	25	570.009
224	17	333.443	225	26	2033.84
226	26	2169.97	227	22	2007.36
228	14	556.194	229	18	169.742
230	25	467.882	231	14	97.6863
232	24	127.753	233	30	584.666
234	33	1321.92	235	19	100.682
236	23	1347.71	237	25	1376.5
238	25	753.524	239	22	359.484

REFERENCES

Allahverdi, A., J. N. D. Gupta and T. Aldowaisan, 1999, "A Review of Scheduling Research Involving Setup Considerations", *International Journal of Management Science*, Vol. 27, pp. 219-239.

Anghinolfi, D. and M. Paolucci, 2007, "Parallel Machine Total Tardiness Scheduling with a New Hybrid Metaheuristic Approach", *Computers & Operations Research*, Vol. 34, pp. 3471-3490.

Armentano, V. A. and M. F. F. Filho, 2006, "Minimizing Total Tardiness in Parallel Machine Scheduling with Setup Times: An Adaptive Memory-based GRASP Approach", *European Journal of Operational Research*, doi:10.1016/j.ejor.2006.09.077.

Azizoglu, M. and O. Kirca, 1999, "Scheduling Jobs on Unrelated Parallel Machines to Minimize Regular Total Cost Functions", *IEEE Transactions*, Vol. 31, pp. 153-159.

Balakrishnan, N., J. J. Kanet and S. V. Sridharan, 1999, "Early/Tardy Scheduling With Sequence Dependent Setups on Uniform Parallel Machines", *Computers & Operations Research*, Vol. 26, pp. 127-141.

Bilge U., F. Kirac, M. Kurtulan and P. Pekgun, 2004, "A Tabu Search Algorithm for Parallel Machine Total Tardiness Problem", *Computers & Operations Research*, Vol. 31, pp. 397-414.

Chen, Z. L., 1997, "Scheduling With Batch Setup Times and Earliness-Tardiness Penalties", *European Journal of Operational Research*, Vol. 96, pp. 518-537.

Chen, Z. L. and W. B. Powell, 2003, "Exact Algorithms for Scheduling Multiple Families of Jobs on Parallel Machines", *Naval Research Logistics*, Vol. 50, pp. 823840.

Lee, Y. H., K. Bhaskaran and M. Pinedo, 1997, "A Heuristic to Minimize the Total Weighted Tardiness with Sequence-Dependent Setups", *IEEE Transactions*, Vol. 29, pp. 45-52.

Lee, Y. H. and M. Pinedo, 1997, "Scheduling Jobs on Parallel Machines with Sequence-Dependent Setup Times", *European Journal of Operational Research*, Vol. 100, pp. 464-474.

Liaw, C. F., Y. K. Lin, C. Y. Cheng and M. Chen, 2003, "Scheduling Unrelated Parallel Machines to Minimize Total Weighted Tardiness", *Computers & Operations Research*, Vol. 30, pp. 1777-1789.

Logendran, R., B. McDonell and B. Smucker, 2007, "Scheduling Unrelated Parallel Machines with Sequence-Dependent Setups", *Computers & Operations Research*, Vol. 34, pp. 3420-3438.

Luo, X. and F. Chu, 2006, "A Branch and Bound Algorithm of the Single Machine Schedule with Sequence Dependent Setup Times for Minimizing Total Tardiness", *Applied Mathematics and Computation*, Vol. 183, pp. 575588.

Meyr, H., 2002, "Simultaneous Lotsizing and Scheduling on Parallel Machines", *European Journal of Operational Research*, Vol. 139, pp. 277-292.

Omar, M. K. and S. C. Teo, 2006, "Minimizing the Sum of Earliness/Tardiness in Identical Parallel Machines Schedule with Incompatible Job Families: An Improved MIP Approach", *Applied Mathematics and Computation*, Vol. 181, pp. 1008-1017.

Rabadi, G. and R. J. Moraga, 2006, "Heuristics for the Unrelated Parallel Machine Scheduling Problem with Setup Times", *Journal of Intelligent Manufacturing*, Vol. 17, pp. 85-97.

Ragatz, G.L., 1993, "A Branch-and-Bound Method for Minimum Tardiness Ssequencing on a Single Processor with Sequence Dependent Setup Times", *Proceedings:*

Twenty-fourth Annual Meeting of the Decision Sciences Institute, pp. 13751377.

Shim, S. O. and Y. D. Kim, 2006, "A Branch and Bound Algorithm for an Identical Parallel Machine Scheduling Problem with a Job Splitting Property", *Computers & Operations Research*, doi:10.1016/j.cor.2006.04.006.

Şerifoğlu, F. S. and G. Ulusoy, 1999, "Parallel Machine Scheduling with Earliness and Tardiness Penalties", *Computers & Operations Research*, Vol. 26, pp. 773-787.

Tahar, D. N., F. Yalaoui, C. Chu and L. Amodeo, 2006, "A Linear Programming Approach for Identical Parallel Machine Scheduling with Job Splitting and Sequence-Dependent Setup Times", *International Journal of Production Economics*, Vol. 99, pp. 63-73.

Webster, S. T., 1997, "The Complexity of Scheduling Job Families About a Common Due Date", *Operations Research Letters*, Vol. 20, pp. 65-74.

Weng, M. X., J. Lu and H. Ren, 2001, "Unrelated Parallel Machine Scheduling with Setup Consideration and a Total Weighted Completion Time Objective", *International Journal of Production Economics*, Vol. 70, pp. 215-226.

Zionts, S., 1974, *Linear and Integer Programming*, Prentice-Hall, Englewood Hills, CA.