UNRELATED PARALLEL MACHINE SCHEDULING WITH SEQUENCE DEPENDENT SETUP TIMES BY ANT COLONY OPTIMIZATION IN TEXTILE INDUSTRY

EBRU ÖNEM

SEPTEMBER 2018

UNRELATED PARALLEL MACHINE SCHEDULING WITH SEQUENCE DEPENDENT SETUP TIMES BY ANT COLONY OPTIMIZATION IN TEXTILE INDUSTRY

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

OF

IZMIR UNIVERSITY OF ECONOMICS

by

ÖNEM, EBRU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF SCIENCE

IN THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

SEPTEMBER 2018

Approval of the Graduate School of Natural and Applied Sciences

(Prof. Dr. Abbas Kenan ÇİFTÇİ)
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

(Assoc. Prof. Dr. Selin Ozpeynirci)
Head of Department

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

(Asst. Prof. Dr. Kamil Erkan KABAK)
Supervisor

Examining Committee Members

Date: 14/09/2018

Asst. Prof. Dr. Kamil Erkan KABAK
Dept. of Industrial Engineering, IUE

Asst. Prof. Dr. Hamdi Giray REŞAT
Dept. of Industrial Engineering, IUE

Asst. Prof. Dr. Mehmet Ali ILGIN
Dept. of Industrial Engineering, CBU

ABSTRACT

UNRELATED PARALLEL MACHINE SCHEDULING WITH SEQUENCE DEPENDENT SETUP TIMES BY ANT COLONY OPTIMIZATION IN TEXTILE INDUSTRY

ÖNEM, Ebru

M.Sc. in Industrial Engineering
Graduate School of Natural and Applied Sciences

Supervisor: Asst. Prof. Dr. Kamil Erkan KABAK
September 2018, 89 pages

This study involves a real production problem of minimizing total weighted tardiness in knitted fabric stage of a textile company. The knitted fabric production has a number of unrelated parallel machines. Also, setup times are sequence dependent in the knitted fabric production system. In addition, different and varied types of release dates for customer orders are defined in the system. To solve the problem, a mixed-integer mathematical model is proposed and it is justifed as NP-hard through experimental results. After, a new heuristic algorithm based on ant colony optimization (ACO) approach is generated to solve the problem with varying problem instances tested with the experimental design. The results show that ACO is an practicable application that can give sufficiently quick solutions.

Keywords: unrelated parallel machine scheduling, sequence dependent setups, weighted tardinesss, ant colony optimization, textile industry

ÖZ

TEKSTİL SEKTÖRÜNDE SIRALAMA BAĞIMLI KURULUM SÜRESİ KISITLI İLİŞKİSİZ PARALEL MAKİNE ÇİZELGELEMESİNİN KARINCA KOLONİSİ İLE OPTİMİZASYONU

Önem, Ebru

Endüstri Mühendisliği Yüksek Lisans Programı

Fen Bilimleri Enstitüsü

Tez Danışmanı: Yard. Doç. Dr. Kamil Erkan KABAK

Eylül 2018, 89 sayfa

Bu çalışma bir tekstil firmasının örgü kumaş aşamasındaki toplam ağırlıklandırılmış gecikmeyi en aza indirgeyen gerçek bir üretim problemini içermektedir. Örgü kumaş üretiminde belirli sayıda ilişkisiz paralel makine vardır. Ayrıca, örgü kumaş üretim sisteminde kurulum zamanları sıralamaya bağlıdır. Buna ek olarak, sistemde farklı ve değişen çeşitte müşteri siparişlerinin üretimine başlayabileceği zamanlar da tanımlanmıştır. Problemi çözmek için, bir karışık tamsayılı matematiksel model önerilmiştir ve problemin zor bir problem olduğu deneysel sonuçlarla gösterilmiştir. Sonra, deneysel tasarımla test edilen değişen problem durumlarıyla çözülerek test edilen karınca kolonisi eniyilemesi yaklaşımı tabanlı yeni bir sezgisel algoritma geliştirilmiştir. Sonuçlar, algoritmanın yeterince hızlı çözümler üreten pratik bir uygulama olduğunu göstermektedir.

Anahtar kelimeler: ilişkisiz parallel makine çizelgelemesi, sıralama bağımlı kurulumlar, ağırlıklandırılmış gecikme, karınca kolonisi optimizasyonu, tekstil endüstrisi

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor, Dr Kamil Erkan KABAK, who has supported me during my research. I wish to thank him for his excellent guidance, motivation, usefull comments and patience. I am grateful to Asst. Prof. Dr. Kamil Erkan Kabak for enlightening me for all the time of this study. This thesis would have never been accomplished without his invaluable guidance and persistent commitment in every step throughout the process,

I would like to express my gratitude to my parents Şükran Önem and Hüsamettin Önem, also to my brother Eyüp Mansur Önem for supporting and encouraging me with their best wishes.

Finally, I also would like to thank to Saygın Dikmen for his full support and help.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

1.1 Introduction

This chapter starts with the research motivation of this thesis in Section 1.2. Then, definition of unrelated parallel machine scheduling and knitted fabric production system scheduling are highlighted in Section 1.3. Also, main characteristics and challenges of this scheduling problem including sequence dependent setup times are explained in Section 1.4. In Sections 1.5 and 1.6, problem definition and research methodology are presented, respectively. Finally, thesis chapters are summarized in Section 1.7.

1.2 Research Motivation

According to the sectoral evaluation report of the first quarter of 2018 of the Aegeans Exporters' Associations (AEA, 2018), exportation in textile and raw materials such as cotton, yarn, knitted and woven fabric presents a continuously increasing trend. This trend is also pointed out by Ngai et al. (2014, p. 87). They mention increased number of research and applications of decision support and intelligent technologies in the textile industry.

Knitted fabric production has several complexities (see Section 1.4). The knitting machines might be unrelated and parallel. The machines that have similar properties could show differences due to their ages or brands. This results in increase the complexities of the problem (Kerkhove et al., 2014; p. 2630). Furthermore, setup times among products have high varieties, and these product varieties are significantly high in the textile industry. Anderson (1995) mentions this complexity and investigates the impact of product mix heterogeneity (PMH) on manufacturing overhead cost in three different fabric production companies. She highlights variations in sequence dependent setup times that are causes of product varieties in the textile industry.

The main motivation of this study is based on low scheduling performance of knitted fabric production of the selected company for this thesis study. This low scheduling performance is explained by total penalty costs attributed for delayed customer orders. The penalty cost of the company consists of the cost of quality, cost of tardiness and the other conflicts that are faced with the customers. Forty percentage of the penalty costs of the interested company is arised from high tardiness. This tardiness depends on the weights of customers that are defined according to priorities of customers. For this reason, minimizing total weighted tardiness in the knitted fabric production is significant for this company. Another motivation of this study is that only few studies exist on scheduling of knitted fabric stage in the textile production literature, and systematic scheduling policies are not widely applied in this area (Koulamas et al., 1996, Pimentel et al., 2006 and Kerkhove et al., 2014) (see Chapter 4).

Next section introduces unrelated machine scheduling and defines scheduling in knitted fabric production systems briefly.

1.3 Scheduling in Knitted Fabric Production Systems

Unrelated parallel machine scheduling (Rm) is the generalization of the identical parallel machine scheduling (Pm) and uniform parallel machines scheduling (Qm) (see Pinedo, 1995; p. 14). Arnout et al. (2009) mention that Rm is generalization of Pm. In the identical parallel machines (Pm) environment, there are m identical parallel machines. A job j needs only one production operation and this operation can be performed by any of these identical parallel machines (Pm) with the same speed v_k where k is in machine and $v_k = 1$, so the processing time of job j on machine k is $p_j = p_{jk} = p_j/v_k$ (Pinedo, 1995; p. 14). When speeds of these machines are different and they do not depend on the job types, then the problem is identified as parallel machines with different speeds (Qm), and the notation of speeds of the machines is shown as v_k. However, v_k is not equal for the machines (Pinedo, 1995; p. 14). If machines are unrelated and parallel, then the speeds of the machines may be different for the different jobs (Pinedo, 1995; p. 14). While the speed of a job is shown by the notation, v_k in the identical and uniform parallel machine scheduling, it is shown in the unrelated parallel machine scheduling by the notation v_{jk}, which is identified by the speed of job j on machine k (Pinedo, 1995; p. 14). The processing time of each machine is calculated by the formula of $p_j = p_{jk} = p_j / v_{jk}$ (Pinedo, 1995; p. 14). For both identical, uniform and unrelated parallel machine scheduling, if there is a machine eligibility (M_j) constraint, then the jobs can be assigned only to machines in the eligibile machine set (Pinedo, 1995; p. 17).

In the production systems, sequence dependent setup times (S_{ij}) are incurred when a time is needed for preparing the machine if the job i precedes the job j for the next job that is processed on that station or machine (Pinedo, 1995; p. 70). Sequence-dependent setup times affect performance of the schedule (Pinedo, 1995; p. 84). Hamzadayi et al. (2017) and Kerkhove et al. (2014) are the studies that use the sqeuence-dependent setup time constraints.

In the knitted fabric production facilities, machines are unrelated and parallel (Rm). In other words, processing times are different for different products for the same machine, and sequence dependent setup times (S_{ij}) affect the performance of the production schedule. Furthermore, products cannot be assigned to all machines available in the machine set. They can be assigned to the machine that is in the eligibile machine set (M_j) of this product.

Next section presents the challenges in knitted fabric production systems briefly.

1.4 Challanges in Knitted Fabric Production

The main challenges of knitted fabric production are listed in the following. Each challenge is then discussed shortly.

1. High product variety (Sen, 2008 and Ngai, 2014)
2. Long sequence-dependent setup times (Kerkhove et al., 2014)
3. Unrelated parallel machines (Kerkhove et al., 2014)
4. Short lead times and product life cycles (Sen, 2008 and Ngai, 2014)
5. Unpredictable global fashion market demand (Sen, 2008 and Ngai, 2014).

The textile industry has high product variety, short life cycles, unbalanced and unpredictable global market demand that makes all processes from production to delivery hard to manage (Sen, 2008 and Ngai, 2014). In the production system of interest, the product variety varies from 1 to 380 (see Chapter 5). The market demand for the products is collected from a real system and they are analysed in Chapter 5. Accordingly, it does not represent a unique pattern (see Chapter 5). Besides all of the complexities of market characteristics, long sqeuence-dependent setup times and different machine types with different processing times also cause the complexities in the production part of the knitted fabric (Kerkhove et al., 2014; p. 2630). In this study, the data for the sequence dependent setup times varies from 1 hours to 10 hours (see Chapter 5).

1.5 Problem Definition

In this thesis, a scheduling problem in a knitted fabric production facility is studied. The sequence dependent setup times, machine eligibility and order release dates are defined as processes restrictions and constraints for this problem. Setup times are sequence dependent since knitted machines require additional gauges and preparations for each product. Each product may need different needle permutation to give the requested effect to the fabric produced. Furthermore, additional time is also required to clean the machine for the next product. Therefore, durations of the setups may change from product to product in general in knitted fabric production. Furthermore, the processing time of each product is different according to the machine that the product is produced, and products cannot be assigned to any machine in the production facility. Each product has an eligible machine set for production.

Tardiness is defined as the number of tardy days of an order, and tardiness function is one of due date related penalty cost function (Pinedo, 1995; p. 18). The late delivery causes a penalty cost that is the combination of customer dissatisfaction and financial cost of tardiness (Pinedo, 1995; p. 2). Total weighted tardiness, which is the generalization of total tardiness function (Pinedo, 1995; p. 57), is computed as the performance measure of this problem since each order has a weight that could be more critical for some clients. For this reason, due date performance becomes an important measure under some challenges like unpredictable fashion market demand, short lead times and short product life cycles (see Section 1.2).

In the interested company, the weighted tardiness cost is one of the major penalty cost that is targeted to minimize (see Section 1.2). Therefore, studies that minimize penalty costs carry a vital importance for the company of interest. By increasing the scheduling performance, rapid changes in the demand and the other challenges in the processes restrictions and constraints can be managed more efficiently. That is, decreasing delays and increasing customer satifaction can be reduced implicitly.

The aim of this study is to minimize the total weighted tardiness while scheduling the given orders on the predefined number of unrelated parallel machines (Rm) under complexities like sequence dependent setup times (S_{ij}), release date of orders (r_j) and machine eligibility (M_j). In the scheduling environment, these types of problems are denoted by the following notation, $Rm \mid r_j, S_{ij}, M_j \mid \sum w_i T_i$ (Pinedo, 1995) (see Chapter 3).

1.6 Research Methodology

In this section, the methodology approach followed in this thesis is explained briefly.

The research methodology of this thesis can be defined under the category of quantitative-model based Operations Management (OM) research. Bertrand and Fransoo (2002) distinguished the quantitative-model based OM research by two subcategories. These subcategories are axiomatic and empirical research. In particular, axiomatic quantitative research is applied as the research methodology of this thesis. Furthermore, this thesis consists of four operational research phases, which are conceptualization, modelling, model solving and implementation (Bertrand and Fransoo, 2002; p .253). The Chapter 2 and Chapter 3 cover the conceptualization parts of this thesis. Additionaly, data collection, data analyses and data structure of the study are given in Sections 5.2 and 5.3 of Chapter 5 as the quantitative analyses of the system. Modelling and model solving phases are covered in Chapter 6 and 7 which include model development and heuristic application, respectively.

Final phase, implementation phase, experimental design and analysis of variance (ANOVA) are given, respectively in Chapter 8.

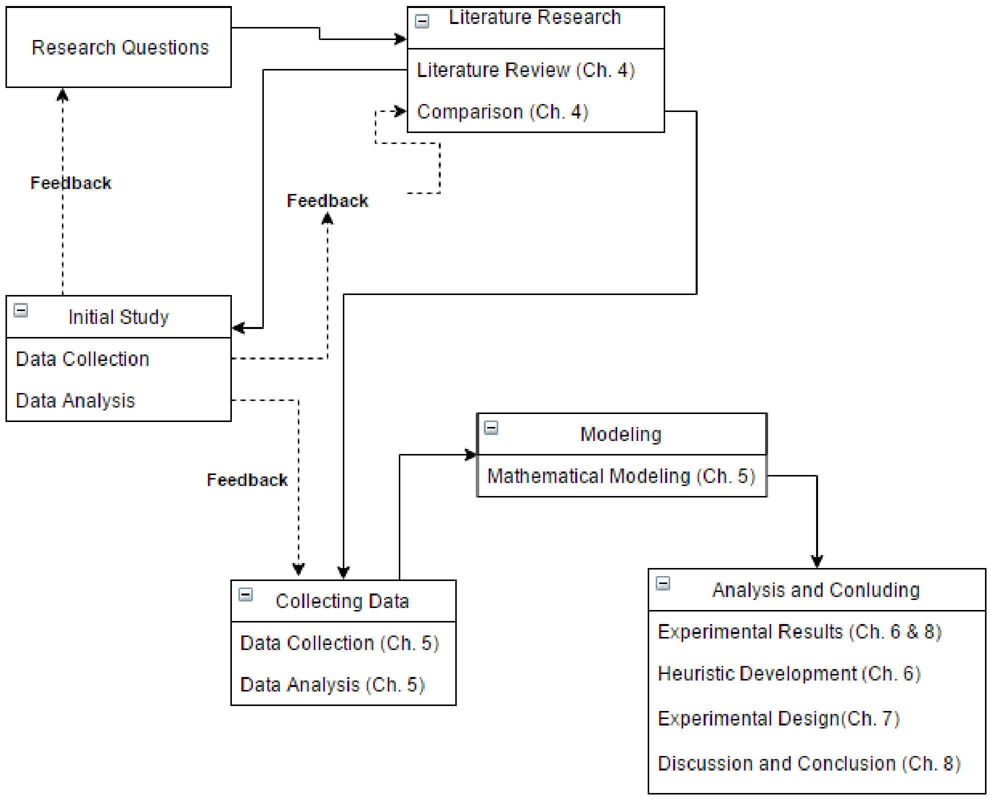


Figure 1 Research Procedure

According to Figure 1, data collection and analysis are the starting points of this study. The relevant literature is discussed and it provides feedbacks on research questions and problem statement of this study. Before developing the mathematical model of the problem, conceptual model of the problem is given. Then, the mathematical model is developed with the collected data. With regard to the experimental results of the mathematical model, it is proven that a heuristic algorithm is necessary to solve the real problem. For this reason, a heuristic algorithm based on ant colony approach is developed and applied using real dataset. An application on total weighted tardiness minimization is created to be used in textile industry. The study is ended with concluding remarks and discussions.

1.7 Summary

This section summarizes the steps of this study briefly. In Chapter 2, main production stages of the textile industry are briefly explained and the knitted fabric production steps are clarified. In Chapter 3, problem statement is presented. The literature research on scheduling and textile industry is covered in Chapter 4. Literature survey and discussion on the relevent papers are presented in this chapter. Chapter 5 describes the proposed methodology for this thesis. Also, the steps of methodology, data collection and data structure in the system are presented in this chapter. Furthermore, the mathematical model of the problem and the experimental results are defined in Chapter 5. After justification of the heuristic algorithm, the proposed ACO heuristic algortihm is presented in Chaper 6. Chapter 7 shows computational results of the proposed heuristic based on the experimental design. Finally, Chapter 8 covers concluding remarks and presents future research.

CHAPTER 2: PRODUCTION SYSTEM OF KNITTED FABRIC

2.1 Introduction

In this chapter, main production stages of the textile industry and relations of among these stages are introduced in Section 2.2. Then, knitted fabric production processes that are observed from a real production facility are explained in Section 2.2. In Section 2.3, the production planning in the knitted fabric facility is explained briefly.

2.2 Main Production Stages of the Textile Industry

Three major processes exist in the textile industry. First major process is yarn production. Second one is knitted fabric production, and the third one is garment production or in the other words the apparel industry. These classifications of the processes are shown in Figure 2. These major processes and their subprocesses are explained briefly in the rest of this section.

First, cotton is collected and then it goes through some production processes to obtain fibers. Then, fibers are used to produce yarns, and it is ended by the yarn production.

In the second major process, the yarn is processed on the knitted machine (see Figure 3), and knitted fabric is produced. After this process, knitted fabrics are received from knitted machine transfered to the dyeing or printing facility. In dyeing process, the knitted fabrics are encolored with chemicals and dyes according to the recipes that are predefined by the laboratorians. In the shop floor, recipes are used to define the proportions of the dyes and the chemicals used in dyeing machines.

The knitted fabrics enter into the printing process after dying if the order request is printed fabric. After the production of dyed fabric, fixing and quality control processes are performed.

In the third major process, the fabrics, outputs of the second major process, are used for garment production. In the garment production, first, fabrics are prepared for the cutting process. The fabrics are cut up in order to obtain the minimum waste. After that, fabric pieces which are cuts of body sizes of garments enter into the printing, emroidery or embelishment processes if any of them is neccessary. Then, sewing process is started. Finally, quality control and ironing process is completed.

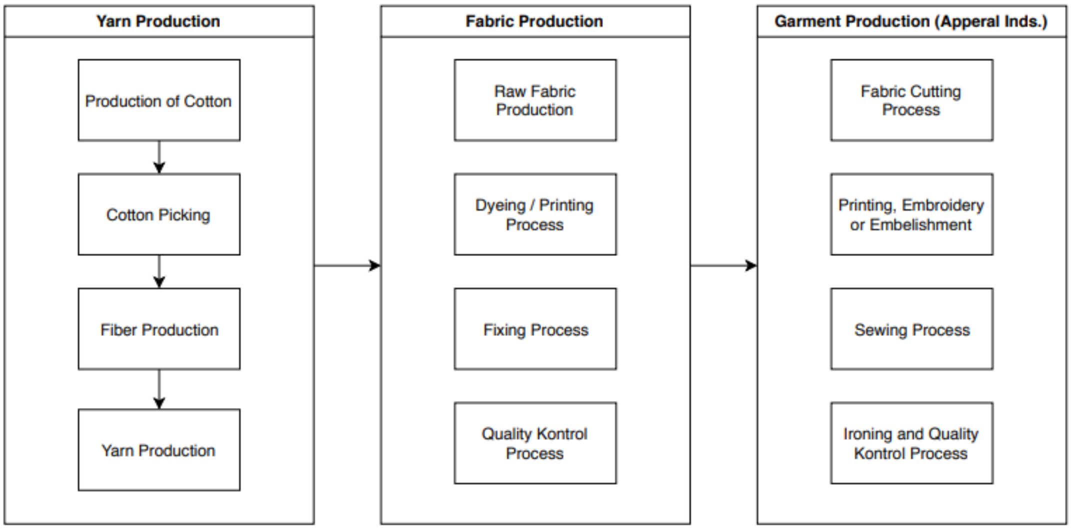


Figure 2 General production flow chart of textile industry.

The next section describes the fabric production in the textile production of interest.

2.3 Description of Knitted Fabric Production System

Two different shapes for knitted fabric machines exist in the facility, *open end* and *tube*. Open-end fabric machines are used to produce single-layer fabric. The knitted machines have round shapes as shown in Figure 3. If the fabric is not cut before wraping to the cyclinder during the process in the machine, it is called tube fabric. Otherwise, it is cut with an additional equipment in the machine and named as an open-end fabric.

Figure 3 An example of an open end knitting machine view (TTM Machine Company, 2018).

The production process of knitted fabric has just one production stage as shown in the Figure 4. The yarns are aligned on the knitting machine and production starts. Almost 20-22 kg knitted fabric is received in the roll-shape.

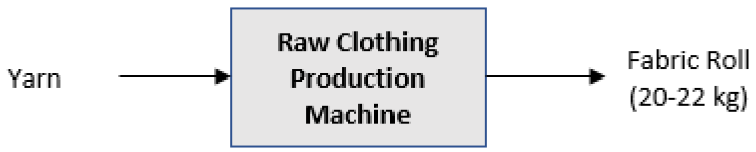


Figure 4 Production stage of knitted fabric.

New yarns are aligned on the machine again before finishing yarns on the machine, and by this way, the machine is not interrupted and does not wait for the yarns until the quantity of the order on that machine finishes. The sequence dependent set-up times exist between the orders and it is differentiated according to the product types of orders.

The interested company focuses on the second major process of the textile industry. That is, it procures the yarns from the suppliers and produces the knitted fabric. Dyeing and printing processes are performed after knitted fabric production. Then, fixing and quality control processes are also performed in this facility. However, this thesis only covers the knitted fabric production step of second major process (See Figure 2).

2.4 Production Planning in Knitted Fabric System

Production planning in the knitted fabric system first starts with entering the the customer orders into the ERP system by the customer representatives (see Figure 5). If there is sufficient knitted fabric in the inventory that is left over from previous orders or canceled orders, the required knitted fabric quantity is supplied from the stock. Otherwise, the dyeing house planning gives an order to the knitted fabric plant. After that, the procedure begins with checking the inventory level of the yarn. If there is not enough yarn in the stock, then the production planners receive the due date information from the procurement department, then they build the production plan according to critical production constraints. General work flow of the planning process until knitted fabric production planning is shown in the Figure 5.

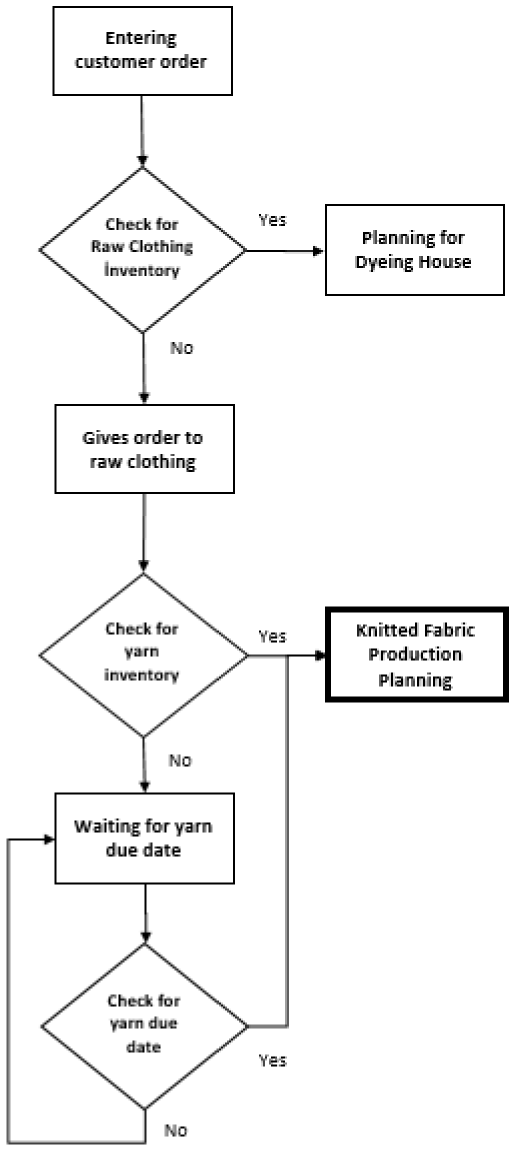


Figure 5 General work flow of the process until planning.

2.5 Concluding Remarks

In this chapter, main production stages of the textile industry and knitted fabric production system in the interested company are described briefly. Furthermore, production-planning procedure in the knitted fabric production system is explained shortly. Next chapter describes main issues in the fabric production and introduces the problem statement of this thesis.

CHAPTER 3: PROBLEM STATEMENT

3.1 Introduction

In this chapter, main issues in the textile industry are briefly descrined in Section 3.2. Then, weighted tardiness minimization is explained by means of references from the scheduling literature in Section 3.3. Finally, the problem statement is given and discussed in Section 3.4.

3.2 Issues in the Knitted Fabric Production

In the knitted fabric production area, production planners assign an production order to the machines according to the machine types, machine availability, order release date, due dates and sequence dependent setup times. As mentioned previously in Section 1.3, one of the sectoral challenges is high product variety (Sen, 2008 and Ngai, 2014). High product variety affects the scheduling performance directly in the case of sequence dependent setup times (Anderson, 1995; p. 366).

The other challenge is existence of variation in the quantities of demands (Chen et al., 2007; p.182). High production flexibility to be able supply wider ranges of quantities is a competitive advantage for a company in stiff competing environments (Chen et al., 2007; p. 182). Furthermore, in the knitted fabric production systems, the machines are unrelated and parallel (Kerkhove et al., 2014; p. 2630). This type of machine environment is defined as NP-hard class in the complexities hierarchy of deterministic scheduling problems (Pinedo, 1995; p. 51). Under these complexities, the companies in this sector should be able to produce efficient production schedule and maximize the customer satisfaction to be able to stay ahead of competitors (Chen et al., 2007; p. 182). For this purpose, scheduling the machines is studied to decrease the total weighted tardiness according to the customer orders in this thesis.

3.3 Weighted Tardiness Minimization in Knitted Fabric Production

Total weighted tardiness (TWT) ($\sum w_i T_i$) can be defined as the sum of the weighted tardiness of each job in the problem and it is defined as a due date related objective function (Pinedo, 1995; p. 57). Furthermore, total weighted tardiness funtion is the generalization of the total tardiness (Pinedo, 1995; p. 57). Weigths (w_i) are used as an importance factor like holding cost. In this study, it represents the priority of the customers.

Minimizing total weighted tardness is one of the ways to minimize the penalty costs of the company of interest (see Section 1.2), and a due date related objective function is necessary to decrease this penalty cost of scheduling of the company. The total tardiness fuction is also a due date related objective function, however, the weigths of customers are significant parameters for the company of interest. This is the reason that total weighted tardiness function is used as the objective function in this study. Karp et al. (1972) and Ho and Chang et al. (1995) showed that parallel machine scheduling are NP-Hard problems. Nevertheless, they have just two parallel machine in their problem. It is also an another reason to consider the problem as NP-hard in this thesis.

3.4 Problem Statement

This section introduces the problem statement of this thesis. The problem statement is defined according to the planning performance. That is, it is the tardiness minimization for all jobs for the knitted fabric production. After introducing the production challenges in previous section, the problem statement is given in the following.

Problem Statement: the aim of this study is to minimize the total weighted tardiness under sequence dependent setup times with unrelated parallel machines in the knitted production of textile production environment.

In the company of interest, the machines are unrelated and parallel (see Section 1.4). Total weighted tardiness, which is a due date related function, tends to be hard even for single machine scheduling ($1||\sum w_i T_i$) (Pinedo, 1995; p. 137). Therefore, parallel machine scheduling with the objective of total weigted tardiness is not as easy as the single machine scheduling with the objective of total weigted tardiness (Pinedo, 1995; p. 137).

Yarns are the primary raw material for the production. Procurement due date of yarns is defined as the release date of an order. Release date means the earliest time that is the starting time of a job on a machine or a station to its processing (Pinedo, 1995; p. 14). Therefore, it is concluded that procurement due date of yarns forces the starting time of an order on a machine that is more than or equal to due date of yarns in this problem.

The other significant parameter that is considered in knitted fabric production and scheduling is sequence dependent setup times. In manufacturing environment, setup times means the required time to prepare a machine or station for a new item that is processed on the same machine or a station (Pinedo, 1995; pp. 70). It is possible to classify setup times as sequence-dependent and sequence-independent (Pinedo, 1995, p. 16; Allahverdi, Aldowaisan, & Gupta, 1999). In the sequence independent types, the setup times are added to processing times of the jobs (Pinedo, 1995; p. 16; Hamzadayi et al., 2017; p. 287). However, in the sequence dependent setup, setup time does not only depend on the job that is currently scheduled, but also it depends on the previous scheduled job (Hamzadayi et al., 2017; p. 287). In the knitted fabric production, setup times can have a range approximately from 15 minutes to more than 480 minutes (Kerkhove et al., 2014; p. 2630). In the interested company, this range is between approximately 60 minutes and 24 hours for the complex setups. Therefore, it is concluded that setup times have significant effect on finish times of the jobs (Kerkhove et al., 2014; p. 2630). In the company of interest, there are almost 380 different types of products. These products are grouped according to their technical specifications like "*knitting types*", "*pus*" and "*fein*". Therefore, eligibile machine set of products directly depends on the groups of products in the company of interest.

The demands of products are deterministic, the production is not a make to stock type production, and the range of order sizes is between 20 and 10000 kg. The annual production volume of the interested company is almost 800 tones according to the last year data, and it is assumed that 66 tones knitted fabric are produced monthly. Therefore, order volume variety is quite high.

In the production facility, in total 97 knitting machines and 46 different types of machines exist. These types are classified according to the "*knitting types"*, "*pus"* and "*fein"* like the specifications of the products. The terms "*pus*" and "*fein"* are used to indicate the diameter of the knitting machine and the number of the needle in one inch, respectively. The orders are assigned to the machines according to their knitting types, *pus* and also *fein*. The machines knitting *type Id, pus* and *fein* values are given in Appendix C, machine knitting type definitions are shown in Appendix B, and a list of *pus* and *fein* values are given in Appendix A.

In this thesis, real knitted fabric production facility is observed and the sample data is gathered from this facility. The interested company does not control and schedule the knitted fabric production system using with an optimization tool, and majority of the textile companies is still lack of an application using an optimization tool for their production facilities (The Textile Hub, 2013). To illustrate, a recent study by Kerkhove et al. (2014) develops a meta-heuristic algorithm for a knitted fabric production scheduling which is a combination of a genetic algortihm and a simulated annealing to minimize total weigted tardiness. In their paper, they show an algorithm that population-based meta-heuristics show better results for real scale problems, and ant-colony optimization is the other most commonly used population-based heuristic algorithm (Kerkhove et al. 2014).

However, there are not any existing applications for the ACO algortihm in these types of scheduling problems in the textile industry when the literature research is surveyed. For this reason, this study represents a methodology to decrease the total weigted tardiness in the textile industry with a rarely used heuristic algorithm, ant colony optimization (see Chapter 4.3). The survey of relevant literature is given in the following section.

CHAPTER 4: LITERATURE REVIEW

4.1 Introduction

This section presents the relevant literature review on knitted machine scheduling, weigted tardiness minimization and scheduling, and also meta-heuristic applications in Section 4.2. Then, a summary table of the literature is presented in Section 4.3. Finally, concluding remarks about the literature review of this study are given in Section 4.4.

4.2 Literature Review

The optimal solution for single machine scheduling problems are known as NP-hard (Pinedo, 1995; p. 50). For this reason, unrelated parallel machine scheduling problems with sequence dependent setup times can be characterized as NP-hard too. The compexity for these types of scheduling problems in most of the papers are discussed first in the literature (Behnamian et al. 2013, Joo et al. 2015, Kayvanfar et al. 2014, Hamzadayi et al. 2017). Therefore, simple heuristics, local search and population-based heuristics or meta-heuristics are used to evaluate better solutions with large intances from more realistic and real life scenarios.

Pimentel et al. (2006) solve a knitted fabric-scheduling problem with the aim of minimizing the lateness. They generate an integer programing model, however, because of the NP-hardness, the model does not give a solution in a proper time. Therefore, they develop a simple heuristic method for this problem.

However, their study excludes the setup times and the machines are identical in parallel. Kerkhove et al. (2014) also solve a knitted fabric-scheduling problem with the aim of minimizing total weighted tardiness with unrelated parallel machines. Their problem includes the sequence dependent setup times, order relase dates and due dates.

Additionally, this paper includes the changeover interference problem that occurs when required number of changeover exceeds the number of technicians. They solve this problem into two different phases. In the first phase, a mathematical model is improved and solved for small instances. Then, a hybrid meta-heuristic algorithm is generated with a combination of genetic algorithm and simulated annealing.

They construct the initial solution by simple heuristics like earliest due date, shortest and longest processing time. After that, they apply some machine selection rules such as machine load balancing and minimal production time. Local simple heuristics are also applied to improve the initial solution. Then, the initial solution is used in the hybrid metaheuristic. In the second phase, changeover interference problem is solved with simple heuristic dispatching rules. Chen et al. (2007) use genetic algorithm for knitted fabric production scheduling with wide ranges of quantities of demanded. They prove that the makespan increases by the rise in the range of the quantities. However, in this paper, the machines are identical in parallel and the setup times just depend on the machines.

Koulamas et al. (1997) generate a decomposition and hybrid simulated annealing heuristic to minimize total tardiness under identical parallel machines. Bilge et al. (2004) apply the tabu search algorithm to minimize the total tardiness. In their problem, the machines are uniform in parallel and the jobs have different due dates and release times. They conclude that their algorithm gives better solution when it is compared to the other applications in the literature. Tavakkoli et al. (2009) generate a genetic algorithm to solve bi-objective scheduling problem. In their problem, the unrelated parallel machines exist and the orders have non-identical due dates and release dates. Furthermore, the setup times are sequence dependent. Their study shows effective results for small and large size instances. Lin and Hsieh (2013) generate a mixed integer programing model to find the minimum value of total weighted tardiness. Machines are unrelated in parallel and setup times are sequence and machine dependent. Jobs have non-identical due dates and release dates in their problem. Also, they provide a heuristic and iterated hybrid meta-heuristic algorithm that can find nearly optimal solution for the same problem. Then, they compare their iterated hybrid meta-heuristic algorithm with tabu search and ant colony optimization. Their results present that their suggested algorithm shows better results than TS and ACO for all size of instances in the terms of total weighted tardiness.

Lin et al. (2013) apply the ACO to solve unrelated parallel machine scheduling to minimize total weigted tardiness. The jobs have non-identical due dates, and all of them are ready to be produced at time zero. Additionally, their problem does not include the setup times. Vallada et al. (2011) generate a genetic algorithm that includes fast local search and local search developed crossover operator. In their conclusion, they prove that their suggested algorithm performs better than the existing applications by comparing with using the benchmark sets of instances.

4.3 Discussion of the Literature

This section discusses the studies that are mentioned in previous sections. Comparison of literature review section is analysed under the following specifications. These specifications are properties of problem, machine types, solution methodology and objective of the problem in addition to the specifications given by Kerkhove et al. (2014). The summary table organised under these specifications is given in Table 1.

Table 1 proves that exact optimization solution methodology is not used alone to solve parallel machine scheduling problems. Additional heuristic algorithms are necessary to solve these types of scheduling problems. Furthermore, it is also concluded that the second solution methodology, simple heuristics, are not generally used without additional local search or population based algorithm. These types of heuristics are applied to solve small problems. Furthermore, these types of heuristics are applied as a part of the algorithm of local search and population based heuristics (Kerkhove et al. 2014).

The third solution methodology is local search based meta-heuristics. Simulated annealing (Koulamas (1997), Kim et al (2002), Radhakrishnan et al. (2010), Lin et al. (2011), Lin et al. (2013)) and tabu search algorithms (Mendes et al (2002), Bilge et al. (2004) and Lee et al. (2013)) are observed as the most popular of them. Besides of these known algorithms, Lin et al. (2013) generate an iterated hybrid meta-heuristic that can find nearly optimal solutions for unrelated parallel machine scheduling problem with machine and sequence dependent setup times, different job releases dates and due dates.

Table 1 Summary Table for the Literature of PMS

	Properties of Problems				Machine Types			Solution Methodology				Objectives of Problem				
	DD	RD	ST	PC	I	UNF	UNR	EO	SH	LSM	PBM	MS	L	T	ET	S
Behnamian et al. (2009)	-	-	SD	-	x	-	-	-	-	x	x	x	-	-	-	-
Bilge et L. (2004)	x	x	-	-	-	x	-	-	-	x	-	-	-	x		-
Joo, Kim (2015)	-	x	MSD	-	-	-	x	x	-	-	x	x	-	-	-	-
Kerkhove, Vanhoucke (2014)	x	x	SD	-	-	-	x	-	x	x	x	-	-	x	-	-
Koulamas (1997)	x	-	-	-	x	-	-	-	x	x	-	-	-	x	-	-
Lin, Lee, Ying, Lu (2011)	x	x	SD	-	x	-	-	-	x	-	-	-	x	-	-	-
Mendes et al. (2002)	-	-	SD	-	x	-	-	-	-	x	x	x	-	-	-	-
Kayvanfar et al. (2014)	x	-	MSD	-	-	-	x	x	-	-	x	x	-	-	x	-
Hamzadayi, Yildiz (2017)	-	-	SD	-	x	-	-	x	-	x	x	x	-	-	-	-
Radhakrishnan, Ventura (2010)	x	-	SD	-	x	-	-	-	-	x	-	-	-	-	x	-
Tavakkoli-Moghaddam et al. (2009)	x	x	MSD	x	-	x	-	-	-	-	x	x	-	x	-	-
Arnaout, Rabadi, Musa (2009)	-	-	MSD	-	-	-	x	-	-	-	x	x	-	-	-	-
Lin, Lin, Hsieh (2013)	x	-	-	-	-	-	x	-	-	x	x	-	-	x	-	-
Vallada, Ruiz (2011)	x	-	MSD	-	-	-	x	-	-	-	x	x	x	x	x	-
Lee et al. (2013)	x	-	MSD	-	-	-	x	-	x	x	-	-	-	x	-	-
Lin, Hsieh al. (2013)	x	x	MSD	-	-	-	x	x	x	x	-	-	-	x	-	-
Pimentel et al. (2006)	x	-	-	-	x	-	-	x	x	-	-	-	x	-	-	-
Chen, Hung, Wu (2007)	x	-	MD	-	x	-	-	-	-	-	x	-	-	-	-	x

* DD: Due date of jobs, RD: Release date of jobs, ST: Setup Time (Sequence Dependent Setup(SD), Sequence Independent Setup(SI), Machine and Sequende Dependent Setup(MSD), PC: Precedence constraint of jobs, I: Identical Machines, UNF: Uniform Machines, UNR: Unrelated Machines
** EO: Optimization, SH: Simple Heuristic, LSM: Local Search Based Heuristic, PBM: Population Based Heuristic, MS: Makespan, L: Lateness, T: Tardiness, ET: Earliness and Tardiness, S: Setup Time

The population based meta-heuristic algorithms show better performance when comparing with local search based meta-heuristics because of their ability to find better solutions by combining good solutions in wider search area (Kerkhove et al. 2014). Genetic algorithm and ant-colony optimization are the best-known population based meta-heuristic algorithms used in parallel machine scheduling area. Kerkhove et al. (2014), Joo et al. (2015), Kayvanfar et al. (2014), Arnout et al. (2009) and Lin et al. (2013), Vallada et al. (2011) apply population-based meta-heuristic algorithms in their studies, and both of them solve unrelated parallel machine scheduling problems. Table 1 shows most of the studies on unraleted parallel machine scheduling that use the population-based meta-heuristic algortihms stand-alone or together with the other solution methodologies.

4.4 Concluding Remarks

The literature review in knitted fabric production scheduling and unrelated parallel machine scheduling with total weighted tardiness minimization is given in Section 4.1. Furthermore, the discussion table is given, and the methologies used generally in parallel machine scheduling are mentioned in Section 4.2. The next chapter describes the steps of the methodology applied in this thesis.

CHAPTER 5: METHODOLOGY

5.1 Introduction

In this chapter, the methodology followed in this thesis is described briefly. Accordingly, first the proposed methodology and its steps are explained in Section 5.1. Then, the data collection and data types are mentioned in Section 5.2.

5.2 Proposed Methodology

In this thesis, the system is analyzed to determine the problem statement in a knitted fabric production facility. High product variety, number of different unrelated parallel machines, high setup times between the orders and variety of order amounts complicate the assignment and scheduling the orders on the machines. First, objectives and constraints, data types, parameters and performance measures are explained in Section 5.3 briefly. They are determined according to the problem definition by observing the real system. Data assumptions are specified in order to determine the boundaries of the problem. Furthermore, an improved mathematical model is defined and applied with small intances to represent its complexity in Section 5.4.

5.3 Conceptual Model

This section includes the objective and constraints, data types and the assumptions of the thesis problem under the following subsections.

5.3.1 Objective and Constraints

The conceptual model for the problem together with its objective function, and its constraints are given in the following.

Minimize Total Weighted Tardiness

Subject to

Constraint 1: A position of a machine can process at most one order at the same time

Constraint 2: Each order can be assigned to one position of any machines at the same time

Constraint 3: An order can be assigned to the first empty position on a machine

Constraint 4: Completion time of an order must be greater than or equal to the summation of processing time of that order on that machine, sequence dependent processing time between this order, the previous order and completion time of the previous order

Constraint 5: Completion time must be greater than or equal to the summation of processing time of this order on that machine and the release date of that order

Constraint 6: Completion time of jobs which are assigned to position 1 of any machines must be greater than or equal to total processing time of that order on that machine

Constraint 7: Completion time of an order must be greater than or equal to the summation of release time of that order and processing time of that order on that machine

Constraint 8: Number of tardy days of an order must be greater than or equal to substracting of due date of that order from the completion day of that order

The identical parallel machines scheduling problem with the aim of minimizing $Cmax$ is considered as NP-hard even when number of machine is equal to 2 (Karp 1972; Garey and Johnson1979). Therefore, it is possible to say that $Rm \mid r_j, S_{ij}, M_j \mid \sum w_i T_i$ is also NP-hard. Additionally, the NP-hardness of the problem is also proven by the computation time of the mathematical model that is discussed in the Section 5.4.

5.3.2 Data Analysis and Types

The data used in this problem is deterministic and defined in Table 1 briefly in the following.

Table 2 Data Used in This Problem

Data Name	**Explanation**
Unit Processing Time (P_{im})	Time is needed to produce 1 kg of order *i* on machine *m* (in minutes)
Release Time of Order (R_i)	Ready time to produce order *i* (in hours)
Due Date of Order (D_i)	Customer due date of order *i* (in days)
Quantity of Order (Q_i)	Demand quantity of order *i* (in kg)
Sequence Dependent Setup time (S_{ij})	Time is needed to produce order *j* after order *i* (in hours)
Machine Eligibility Matrix (U_{im})	Eligible machines of order *i* can be assigned
Weight (w_i)	Customer associated weight for order *i*
Number of Machines (*M*)	Total number of machines
Number of Orders (*N*)	Total number of orders
Number of Positions (*K*)	Number of position on each machine (this value is equal to number of orders because of the possibility to assign the all orders on only one machine)

Unit processing time is the time needed to produce 1 kg of an order on a specific machine. Release time of an order is defined as ready time to produce an order on any machine. Due dates (D_j) and quantities (Q_j) of orders are gathered from the ERP system of the facility. Sequence dependent setup times are gathered by the time study in the shop floor. Machine eligibility matrix is prepared by analyzing of $fabric\ type,\ pus$ and $fein$ values of the orders and machines (see Sections 3.4).

These parameters were not available in the database of the company. Therefore, it was necessary to analyze these parameters from the given information like machines' "pus", "$fein$" and "$type$" that was avaliable in the database separately. An order has the same type of specifications with the machines. These specifications are "pus", "$fein$" and "$type$". Each triplet of "pus", "$fein$" and "$type$" corresponds to one type of machine and order. To be able to assign an order onto a machine is constrained by these triplets ($pus/fein/type$) (see Sections 3.4). Weights of orders (w_i) represent the priorities of the orders according to the customers. These priorities are defined according to groups of the customers which are A, B and C. The major customers are group A customers, and the customers that have less priorities than others are group C customers. Group B customers have intermediate priorities in the facility. In the problem, these groups are enumerated as 1, 2 and 3 for groups C, B and A, respectively.

5.3.3 Assumptions

The assumptions that are considered in this study are summarized as follows.

- No pre-emption is allowed
- The mean time between failures and mean time to failure of the machines are ignored
- Orders to be scheduled are known at the beginning of the scheduling
- Setup operators' availability constraint is ignored. That is, machines do not wait for available setup operators
- Variable tardiness cost of the orders are weighted between the values {1,2,3} according to the priorities of customers
- There is no lot sizing

5.4 Model Development

In Section 5.4.1, the generated mathematical model and its formulation is given. Then, a simple application of the mathematical and experimental results are shown in Section 5.4.2.

5.4.1 Mathematical Model

There are m different parallel machines in the system and setup times are dependent to the sequence of orders. Furthermore, M_j stands for the set of machines that can process order j. The objective function minimizes total weighted tardiness cost. This mathematical model is the modified version of the model that is used by Kerkhove et al. (2014). According to the characteristics of the problem, some addditional variables are determined in this version. One of the additional parameters is $U[j,m]$ which is used to determine the eligible machine set of each order. The other parameters are $Cday$ and Q. $Cday$ is necessary to convert the completion time to completion day. The processing time unit is in minutes. Then, it is needed to convert the completion times to completion days to calculate tardy days of orders. The facility has three shifts, therefore the production works 24 hours. Therefore, the $Cday$ of an order is calculated by the division of the completion time of that order into 24 hours. For instance, if completion time of an order I (C_i) is 60 hours, the $Cday$ is calculated by 60 hours divided by 24 hour/day. Then, $Cday_i$ is found as 3 days, when C_i is rounded up. It is necessary to calculate the number of tardy days ($Tday$) in a day since the due date of each order is given in days in the problem set. Finally, Q is added to define the order amounts, and P is redefined as the unit processing time of an order on a machine. The different location part property of the previous version of the model, that is generated by Kerhove et al. (2009), was not applied in this model though it is not required for this problem.

<u>Indices</u>

i, j	$Order: i, j \in N = \{1, \dots, N\}$
k	$Position: k \in K = \{1, \dots, K\}$
m	$Machine: m \in M = \{1, \dots, M\}$

<u>Parameters</u>

w_i	$Variable\ unit\ cost\ of\ tardiness\ per\ order\ i$
R_i	$Release\ time\ of\ order\ i$
D_i	$Due\ date\ of\ order\ i$
P_{im}	$Unit\ production\ time\ of\ order\ i\ on\ machine\ m$

Q_i	$Demand\ quantity\ of\ order\ i$
S_{ji}	$Sequence\ dependent\ setup\ time\ needed\ when\ order\ j$ $precedes\ the\ order\ i$
U_{im}	$1\ if\ order\ j\ can\ be\ processed\ by\ machine\ m, 0\ otherwise$
N	$Number\ of\ orders$
K	$Number\ of\ positions$
M	$Number\ of\ machines$
BM	$A\ big\ positive\ number$

<u>Variables</u>

C_i	$Completion\ time\ of\ order\ i\ in\ hour$
$Cday_i$	$Completion\ time\ of\ order\ i\ in\ day$
T_i	$Number\ of\ tardy\ days\ of\ order\ i$
X_{ikm}	$1\ if\ order\ i\ is\ assigned\ to\ position\ k\ of\ machine\ m; 0\ o/w$

According to the notation given in Section 1.5 and conceptual model defined in Section 5.3, the mathematical model that includes the objective function and constraints determined for the problem are given as follows.

Objective Function:

$$Minimize\ \sum_i^N (w_i * T_i) \tag{1}$$

Subject to:

$$\left\lceil \frac{C_i}{24} \right\rceil \leq Cday_i \qquad \forall\ i \epsilon N \tag{2}$$

$$\sum_{m=1}^{M} \sum_{k=1}^{K} x_{ikm} = 1 \qquad \forall\ i \in N \mid U_{im} = 1 \tag{3}$$

$$\sum_{i=1}^{N} x_{ikm} \leq 1 \qquad \forall\ k \epsilon K, \forall\ m \epsilon M \tag{4}$$

$$\sum_{i=1}^{N} x_{ikm} - \sum_{i=1}^{N} x_{i(k-1)m} \leq 0 \qquad \forall\ k \in K \mid k > 1, \forall\ m \in M \tag{5}$$

$$C_i + BM\left(2 - x_{ikm} - x_{j(k-1)m}\right) \geq \frac{P_{im} * Q_j}{60} + S_{ji} + C_j$$

$$\forall\ i,j\ \epsilon N\ , m\ \epsilon\ M, k\ \epsilon\ K\ |k > 1, i = !\,j \quad (6)$$

$$C_i \geq\ R_i + \sum_{m=1}^{M}\sum_{k=1}^{K}\frac{P_{im} * Q_i}{60} * x_{ikm} \qquad \forall\ i\epsilon N \quad (7)$$

$$T_i \geq Cday_i - \sum_{k=1}^{K}\sum_{m=1}^{M} x_{ikm} * D_i \qquad \forall\ i\epsilon N \quad (8)$$

$$x_{ikm}\ \epsilon \{0,1\} \qquad \forall\ i\ \epsilon\ N, m\ \epsilon\ M, k\ \epsilon\ K \quad (9)$$

$$C_i, Cday_i, T_i \geq 0\ and\ Cday_i\ is\ integer \qquad \forall i\ \epsilon\ N$$

(10)

Constraint (1) minimizes the total weighted tardiness cost. Constraint (2) calculates the completion day of each order. Constraint (3) guarantees that the orders can be assigned to only one machine in its eligible machine set. Constraint (4) ensures that only one order can be processed on a position of a machine. Constraint (5) forces as a full position among all positions over a machine. To be clear, it forces to assign jobs to positions respectively to avoid having an empty position. To illustrate, position 2 is empty while position 3 is busy. Completion times of the orders that are assigned to positions except position 1 are calculated in Constraint (6). Constraint (7) guarantees that orders cannot start to be processed before their release dates. Constraint (8) calculates the number of tardy days for each order. Finally, constraint (9) and constraint (10) define binary variables and nonnegative variables, respectively.

5.4.2 Experimental Design

The mathematical model given in the previous section is run for different data sets. The results together with statistics of mathematical model in OPL are given in Table 3. The model is run for 12, 16 number of orders for five machines respectively. Then, the number of machines is increased to 6 machines, and the model is run for 16, 20, 22 orders respectively. The objective function values and the machines sequences for each iteration are also given in Table 3. It appears that computation time increases by increasing the number of intances. According to this situation, it is concluded that the problem is NP-hard. Therefore, a heuristic algorithm is required to solve the problem with large problem instances.

Table 3 Results of the Computational Experiments for the Mathematical model

M	N	Comp. Time (min.)	Obj. Func.	Machine Sequences	Tardy Orders
5	12	0.07	12	M1= {8,10,12} M2= {6,3} M3= {2,1} M4= {4,9} M5= {5,7,11}	1,2,4,9,11
	16	4.5	19	M1= {2,6} M2= {8,10,12,14,1} M3= {16,3} M4= {4,13,15} M5= {5,7,9,11}	11,1,15,9,6,4,3,2
6	16	0.26	16	M1= {2,1} M2= {8,16,12} M3= {10} M4= {5,7,4} M5= {11,9,13,15} M6= {6,3,14}	1,2,4,9,15
	20	9	26	M1= {2,20} M2= {16,1} M3= {8,10,12,19,18} M4= {5,11,13,4} M5= {9,7,17,15} M6={6,3,14}	4,17,15,2,1
	22	51	29	M1= {2,20,21} M2= {8,10,1} M3= {16,12,18,19} M4= {5,11,13,22,4} M5= {9,7,17,15} M6= {6,3,14}	4,17,15,2,1

Apart from the results of computational experiments of mathematical model given in Table 3, the input data of the developed mathematical model are explained in Tables 3, 4, 5, 6 an 7 in the following.

Table 4 Order Numbers, Product Types and Qantities for the Mathematical Model

Order No (*i*)	Product Type of Order *i*	Quantity of Order *i* (Q_i) (kg)	Order No (*i*)	Product Type of Order *i*	Quantity of Order *i* (Q_i) (kg)
1	Type 1	500	12	Type 1	400
2	Type 1	391	13	Type 2	250
3	Type 1	350	14	Type 1	250
4	Type 2	1000	15	Type 2	700
5	Type 2	500	16	Type 1	650
6	Type 1	450	17	Type 2	800
7	Type 2	300	18	Type 1	450
8	Type 1	275	19	Type 1	500
9	Type 2	300	20	Type 1	600
10	Type 1	500	21	Type 1	500
11	Type 2	500	22	Type 2	350

The order set has two types of products, each order stands for a particular product type. The orders (product) types and order quantities used in this experimental design are shown in the Table 4.

The unit processing times of orders on their eligible machine set are shown in Table 5. In matrix P_{im}, processing times are equal to zero if the order cannot be produced on the machine *m*. U_{im} is equal to 1 if order *i* can be produced on machine *i*.

Table 6 represents the sequence dependent setup times between the orders. The sequence dependent setup times (S_{ij}) are equal to zero if the order *i* and *j* cannot be assigned to same machines because of their eligible machine sets.

Table 7 represents the release time and due date of each order that are used in the experimental design of the mathematical model.

Table 5 Processing Times (P_{im}) and (U_{im})

		Processing Times (P_{im}) in Minutes						U_{im}					
Order No (*i*)	Product Type of Order *i*	M1	M2	M3	M4	M5	M6	M1	M2	M3	M4	M5	M6
1	Type 1	3.4	3	3.2	0	0	3.1	1	1	1	0	0	1
2	Type 1	3.6	3.2	3.4	0	0	3.5	1	1	1	0	0	1
3	Type 1	3.5	3.2	3.1	0	0	3.3	1	1	1	0	0	1
4	Type 2	0	0	0	3.25	3.7	0	0	0	0	1	1	0
5	Type 2	0	0	0	3.5	3.8	0	0	0	0	1	1	0
6	Type 1	3.6	3	4	0	0	3.2	1	1	1	0	0	1
7	Type 2	0	0	0	3.4	3.6	0	0	0	0	1	1	0
8	Type 1	3	2.5	2.9	0	0	3.5	1	1	1	0	0	1
9	Type 2	0	0	0	3.5	3.8	0	0	0	0	1	1	0
10	Type 1	3	2.5	3.2	0	0	3.4	1	1	1	0	0	1
11	Type 2	0	0	0	3.5	3.8	0	0	0	0	1	1	0
12	Type 1	3	2.5	3.4	0	0	3.2	1	1	1	0	0	1
13	Type 2	0	0	0	3.5	3.8	0	0	0	0	1	1	0
14	Type 1	3	2.5	3.3	0	0	3.2	1	1	1	0	0	1
15	Type 2	0	0	0	3.5	3.8	0	0	0	0	1	1	0
16	Type 1	3	2.5	3	0	0	3.1	1	1	1	0	0	1
17	Type 2	0	0	0	3.5	3.8	0	0	0	0	1	1	0
18	Type 1	3	2.5	2.9	0	0	3.1	1	1	1	0	0	1
19	Type 1	3	3.4	3	0	0	3.2	1	1	1	0	0	1
20	Type 1	3	3.5	2.9	0	0	3.5	1	1	1	0	0	1
21	Type 1	3	3.5	3	0	0	3.3	1	1	1	0	0	1
22	Type 2	0	0	0	3.5	3.8	0	0	0	0	1	1	0

Table 6 Setup Times (S_{ij}) in hours

S_{ij}	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**
1	0	5	3	0	0	2.5	0	6	0	2	0	1	0	1	0	1	0	1	5	1	10	0
2	5	0	2	0	0	3	0	2	0	4	0	3	0	3	0	3	0	3	6	4	3	0
3	3	2	0	0	0	2.5	0	3.5	0	2.5	0	6	0	6	0	6	0	6	6	6	7	0
4	0	0	0	0	2	0	2.5	0	4	0	5	0	5	0	5	0	5	0	0	0	0	5
5	0	0	0	2	0	0	4	0	10	0	2.5	0	2.5	0	2.5	0	2.5	0	0	0	0	2.5
6	2.5	3	2.5	0	0	0	0	2.5	0	3	0	2	0	2	0	2	0	2	2	2	2	0
7	0	0	0	2.5	4	0	0	0	4	0	2	0	2	0	2	0	2	0	0	0	0	2
8	6	2	3.5	0	0	2.5	0	0	0	2	0	1	0	1	0	1	0	1	4	1	5	0
9	0	0	0	4	10	0	4	0	0	0	2	0	2	0	2	0	2	0	0	0	0	2
10	2	4	2.5	0	0	3	0	2	0	0	0	1	0	1	0	1	0	1	1	1	1	0
11	0	0	0	5	2.5	0	2	0	2	0	0	0	3	0	3	0	3	0	0	0	0	3
12	1	3	6	0	0	2	0	1	0	1	0	0	0	2	0	2	0	2	2	5	10	0
13	0	0	0	5	2.5	0	2	0	2	0	3	0	0	0	4	0	4	0	0	0	0	4
14	1	3	6	0	0	2	0	1	0	1	0	2	0	0	0	5	0	5	5	5	5	0
15	0	0	0	5	2.5	0	2	0	2	0	3	0	4	0	0	0	9	0	0	0	0	5
16	1	3	6	0	0	2	0	1	0	1	0	2	0	5	0	0	0	5	6	3	2	0
17	0	0	0	5	2.5	0	2	0	2	0	3	0	4	0	9	0	0	0	0	0	0	3
18	1	2.5	6	0	0	2	0	1	0	1	0	2	0	5	0	5	0	0	1	1	3	0
19	5	6	6	0	0	2	0	4	0	1	0	2	0	5	0	9	0	7	0	4	2	0
20	1	4	6	0	0	2	0	1	0	1	0	2	0	5	0	3	0	1	4	0	1	0
21	5	3	7	0	0	2	0	5	0	1	0	5	0	5	0	2	0	3	2	1	0	0
22	0	0	0	5	2.5	0	2	0	2	0	3	0	4	0	5	0	3	0	0	0	0	0

Table 7 Release Time (R_i) and Due Date (D_i) of Order *i*

Order No (*i*)	Release Time in Hours	Order No (*i*)	Release Time in Hours	Order No (*i*)	Due Date in Days	Order No (*i*)	Due Date in Days
1	10	12	24	1	2	12	3
2	12	13	32	2	1	13	4
3	12	14	24	3	2	14	3
4	12	15	12	4	2	15	5
5	8	16	12	5	3	16	2
6	0	17	24	6	2	17	3
7	24	18	48	7	3	18	5
8	5	19	48	8	1	19	6
9	24	20	24	9	3	20	3
10	18	21	24	10	2	21	4
11	24	22	24	11	3	22	5

CHAPTER 6: AN ACO APPLICATION DEVELOPMENT

6.1 Introduction

In this chapter, an application of ant colony optimization (ACO) is explained. Steps of the proposed ACO heuristic algorithm are shown with a flow chart in Figure 6, and they are explained in Section 6.2. Then, the pseudo-code of the proposed ACO heuristic algorithm is given in Section 6.3. Steps and pseudo-code of local search algorithm are explained step-by-step in Section 6.4. Finally, the complexity of the proposed ACO heuristic algorithm and concluding remarks about this chapter are given in Sections 6.4 and 6.5, respectively.

6.2 Ant Colony Optimization (ACO)

Ant colony optimizations are first mentioned by the studies of Dorigo (see Dorigo et al. 1999) as an approach to solve combinatorial optimization problems. Ant algorithms are generated by using observations of real ant colonies. Ants are social insects. That is, they work for the benefit of colony for which they belong to, they do not work for individual benefits. For this reason, the behavior of social insects gain much attention of the scientists (Dorigo et al. 1999; p. 1). The most attractive behavior of ant colonies is their ability to find the shortest path from their nest to the food source and this behavior is named as foraging behavior (Dorigo et al. 1999; p.1). The ants deposit pheromone trail. That is, a chemical substance that they deposit while they move between nest and food source. This behavior is known as stigmergy (Dorigo et al. 1999; p. 3). Stigmergy enables ants to perform their foraging behavior (Dorigo et al. 1999; p. 3). Ants smell the pheromone and they find their way by selecting the way that has strong pheromone concentrations (Dorigo et al. 1999; p. 1). Even a single ant can find a way from nest to food source, a colony of ant can find the shortest path from nest to food source.

Another significant property of real ants is autocatalytic mechanism (positive feedback) that works with implicit evaluation of solutions (Dorigo et al. 1999; p. 4). In the shorter paths, ants complete the path earlier than the ants on the longer paths. That is, pheromones are deposited by the ants sooner in the shorter path, and it results that more ants choose the shorter path. In addition, pheromones are evaporated over the time, and by this way the ant colony forgets the past history and work on the search space without being over-restriction by past decisions (Dorigo et al. 1999; p. 6).

In the ant colony optimization, artificial ants are used instead of real ants, and artificial ant colonies are used to find good solutions for difficult discrete optimization problems (Dorigo et al. 1999; p. 5). Both real and artificial ants have the same purpose, finding the shortest path. However, artificial ants have more abilities than the real ants (Dorigo et al. 1999; p.5). The artificial ants have the all characteristic of real ants and as addition of these characteristic, they can deposit pheromone with respect to the quality of the solution that they find. Moreover, the time to deposit pheromone can be arranged according to the problem. The other most significant additional characteristic of an artificial ant is deamon actions which are used to improve efficiencies of the artificial ants and give them extra capabilities (Dorigo et al. 1999; p. 6). These deamon actions are lookahead, local optimization and backtracking (Dorigo et al. 1999; p. 6). Local optimization is commonly used for them when it is compared to lookahead and backtracking (Dorigo et al. 1999; p. 6).

In the ACO meta-heuristic, incremental constructive approach is used to find feasible solution. In other words, the algorithm generates a solution by adding individual components of the problem. For instance, in TSP, the solution is constructed by adding or selecting a node and finally a feasible solution is generated at the end with the all predefined nodes. The main procedure of the ACO is composed of three main functions: ant generation and activity, pheromone evaporation, deamon actions (Dorigo et al. 1999). Deamon actions are optional and they depend on the construction of the algorithm and the problem. The first ACO algorithm in the literature was Ant System (AS) and it was built to solve Traveling Salesman Problem (TSP) (Dorigo et al. 1999). The algorithm executes a number of iterations. In each iteration, each ant finds a solution for the problem. That means, after an iteration, number of ant solutions are generated. To illustrate, a solution is a tour which includes all nodes and arcs once without any sub-tours in a TSP.

An ant visits the nodes (cities) step by step, and finally, it constructs a tour. Ant-decision table (a_{ij}) is used to decide which node is added to the tour. This table is obtained by the pheromone trail values and the heuristic values. In this problem, pheromone trail value is defined for an arc (i,j) and heuristic value is calculated by using the distance between node i and node j. With this table, the probability of an ant selects to go from node i to node j in an iteration (p_{ij}). α and β are used for the favorability between pheromone trail and heuristic value. After all ants complete their tour, the pheromone update on all arcs is occured. The pheromone update includes an addition (increase) and evaporation (decrease). Addition means that each ant deposits an amount of pheromone on each arc that it is visited. The amount is proportional with performance of the solution of this ant. In the evaporation part, the pheromone amount on each arc is decreased with using evaporation rate. This rate is also defined as a parameter like α and β (Dorigo et al. 1999).

In this section, the summary of developed ACO algorithm is given briefly. In next section, steps of the ACO algorithm applied in this study are presented and explained briefly.

6.3 Steps of Proposed ACO Algorithm

Proposed ACO heuristic algorithm is composed of four main parts. In the Part 1, the vector S1 that represents machine assignment of each order, is found. S1 is a vector that has N (Number of Orders) components. The components in this vector should be populated with the machine numbers. For instance, if the problem set has 8 orders and 3 machines S1 can be like S1= [1 1 2 3 1 2 3 1]. It means, orders 1,2,5 and 8 are assigned to machine 1, orders 3 and 6 are assigned to machine 2, and orders 4 and 7 are assigned to machine 3. In the flow chart, the steps between 4 and 6 are used to find S1. Part 2 includes steps 7 and 8. In this part, S2 matrix that has M (number of machines) number of rows and N (number of orders) number of columns (positions) is found. S2 matrix presents a machine sequence. More clearly, each row represents sequence of orders on a machine. $S2 = \begin{bmatrix} 1 & 8 & 5 & 2 & 0 & 0 & 0 & 0 \\ 6 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 7 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$ is can be given as an example for the same example above. For instance, machine 1 has a sequence as 1-8-5-2, machine 2 has a sequence as 6-3, and machine 3 has a sequence as 4-7 on this example.

In the Part 3, TWT_k (Total Weighted Tardiness), where $k \epsilon$ NbANT (Maximum Number of Ants), is calculated according to S2, and BESTSOL (best solution of the algorithm) is updated if an ant can find a better solution than current BESTSOL. Part 3 includes the steps 10, 11, 12 and 13. In the Part 4, the local search algorithm runs for each S2 that is found by an ant, and TWT_k and BESTSOL are updated if the local search algorithm can find better solution. After that, the pheromone is updated with pheromone deposit and evaporation. Part 4 contains the steps 14, 15, 16, 17, 18 and 19. These steps are represented in detail in Figure 6.

In this algorithm, two different pheromone trails are used, τ_{im}^{I} for machine selection and τ_{ij}^{II} for order sequence, where $i,j \ \epsilon$ N and m ϵ M. In equation 1, the probability of assigning an order to a machine (Prob1) is calculated. η_{im}^{I} suggests a machine which has the processing time for that order. It is used as heuristic information and calculated by equation 2. Equations 3 and 4 give the formula to calculate the probability of order *i* that precedes order *j* (Prob2) and heuristic information (η_{ij}^{II}) that is proportional to setup times between orders *i* and *j*, respectively. α_1, β_1 and α_2, β_2 are used to determine the importance of pheromone amount over the heuristic value while Prob1 and Prob2 are calculated, respectively. After all ants find a solution, the pheromones evaporate for each τ_{im}^{I} and τ_{ij}^{II}. Evaporation formula is given in equation 5.

$$Prob1 = \frac{(\tau_{im}^{I})^{\alpha 1} * (\eta_{im}^{I})^{\beta 1}}{\sum_{l}^{M} (\tau_{il}^{I})^{\alpha 1} * (\eta_{il}^{I})^{\beta 1}} \tag{1}$$

$$\eta_{im}^{I} = \frac{1}{ProcessTime_{im}} \tag{2}$$

$$Prob2 = \frac{(\tau_{ij}^{II})^{\alpha 2} * (\eta_{ij}^{I})^{\beta 2}}{\sum_{l}^{NB} (\tau_{il}^{II})^{\alpha 2} * (\eta_{il}^{I})^{\beta 2}} \tag{3}$$

$$\eta_{ij}^{II} = \frac{1}{Setup_{ij}} \tag{4}$$

$$\tau_{im}^{I} = (1-\rho) * \tau_{im}^{I}, \qquad \tau_{ij}^{II} = (1-\rho) * \tau_{ij}^{II} \tag{5}$$

After finding S1 and S2, TWT is calculated for each solution. In this problem, the constraints are the same with the mathematical model applied in Chapter 5. The orders can be assigned only to the machine in their eligible machine set, and orders can not be processed before their release dates. Therefore, the completion time of each order is computed with respect to release date of that order, or the completion time of the order which precedes that order. The completion time is not enough to calculate TWT since due dates are in days, therefore it is necessary to convert them into days. After finding the completion day of each order, the TWT is calculated by subtracting the completion day of an order from the due date of an order if the value is negative.

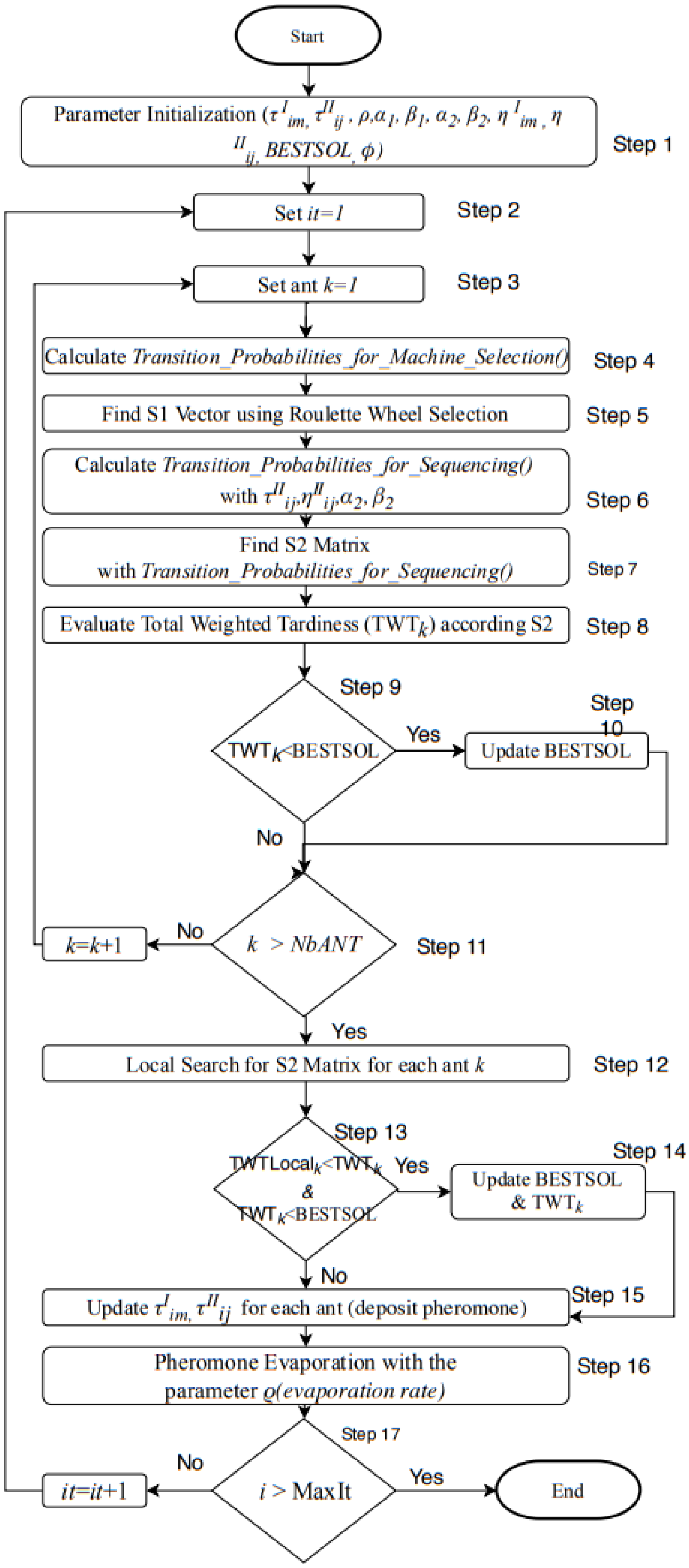


Figure 6 Flow Chart of ACO Algorithm

The explanations of steps of the ACO algorithm are presented in rest of this section.

Step 1: The algorithm initializes the ACO parameters and assings BESTSOL as infinitive. Since this is a minimizing problem, heuristic tries to find a smaller solution than the best solution.

Step 2: The first loop of the algorirthm starts by setting the it=1.

Step 3: The second loop, which is for ant generation, starts setting k=1.

Step 4: By using the parameters τ_{im}^{I}, η_{im}^{I} , α_1, β_1 and the equation 1 and 2 in Section 6.3, transition probabilities of assigning the order *i* to the machine *m are* calculated.

Step 5: By roulette wheel selection method, a machine is selected to assign for each order and vector S1 is found.

Step 6: By using the parameters, τ_{ij}^{II}, η_{ij}^{II}, α_2, β_2 and equation 3 and 4 in section 6.2, the probality to process the order *i* before order *j* is calculted.

Step 7: S2 matix is found by using the probability matrix that is calculated in step 6.

Step 8: Evaluate Total Weighted Tardiness (TWT_k) according to S2 matrix. This calculation is shown in the pseudo code of the ACO algorithm in Procedure Evaluate_TWT (*k*,*i*) in Figure 7.

Step 9: The algorithm checks for the solution if it is better than the best solution of the algorithm (TWT_k < BESTSOL). If it is better, the algorithm continues with step 10, otherwise continues with Step 11.

Step 10: Update BESTSOL with TWT_k and go Step 11.

Step 11: The algorithm checks for the ant number. If Ant > NbAnts (maximum number of ants), then algorithms continues with step 12; otherwise the algorithms go to step 3 by updating the ant number as (k=k + 1).

Step 12: After all ants find indiviualt solutions, these solutions enter to the local search algorithm described in Section 6.4.

Step 13: The algorithm checks for the solution that is found in the local search, is better than the solution of the ant that is tried to be improved. If the local search finds a better solution than the solution of that ant (TWT_k), the algorithm updates the ant's solution with the value TWT_Local_k in step 14. If the solution is also better than the best solution (BESTSOL) of the algorithm, then the best solution is also updated in step 14. If local search cannot find better solution, the algorithm continues with the step 15.

Step 14: Update BESTSOL or TWT_k, then go Step 15

Step 15: The algorithm updates the pheromone trails $\tau_{ij}^{II}, \tau_{im}^{I}$ as shown in the pseudo code of ACO in the procedure *Pheromone_Deposit* in Figıre 7.

Step 16: The algorithm updates the pheromone trails by the evaporation of them with the parameter ρ as shown in the procedure *Pheromone_Evaporation()* in Figure 7.

Step 17: The algorithm checks for the iteration number is smaller than the MaxIt (maximum number of iterations). If it is greater than the MaxIt, the algorithm stops. Otherwise, the iteration number *it* is updated by the formula *it* (*it*= *it*+ 1) and the algorithm goes back to the step 2.

The Steps for the ACO Algorthm is listed below.

Step 1: Parameter Intialization τ_{im}^{I} , τ_{ij}^{II} , ρ, α_1, β_1, α_2, β_2, η_{im}^{I}, η_{ij}^{II}, φ, *BESTSOL=inf*

Step 2: Initiate Iteration *it*

Step 3: Initiate Ant k

Step 4: Calculate *Transition_Probabilities_for_Machine_Selection()*with τ_{im}^{I}, η_{im}^{I} , α_1, β_1

Step 5: Find S1 vector using Roulette Wheel Selection

Step 6: Calculate *Transition_Probabilities_for_Sequencing()* with τ_{ij}^{II}, η_{ij}^{II}, α_2, β_2

Step 7: Find S2 Matrix with *Transition_Probabilities_for_Sequencing()*

Step 8: Evaluate Total Weighted Tardiness (TWT_k) according to S1 and S2

Step 9: If TWT_k < BESTSOL, go Step 12; otherwise go Step 13

Step 10: Update BESTSOL and go Step 13

Step 11: If Ant > NbAnts (maximum number of ants), then go to Step 14; otherwise update Ant number (k=k + 1) and go to Step 3

Step 12: Execute the local search algorithm described below

Step 13: If $TWT\_Local_k < TWT_k$ & TWT_k < BESTSOL, go Step 16; otherwise go

Step 14: Update BESTSOL and TWT_k, then go Step 17

Step 15: Update τ_{ij}^{II}, τ_{im}^{I} for each ant (deposit pheromone)

Step 16 : Pheromone Evaporation with the parameter ρ

Step 17 : If i > MaxIt (maximum number of iterations), STOP; Otherwise update it (it= it+ 1) and go to Step 2

6.4 Pseudo-Code of Proposed ACO Algorithm

This section represents the pseudo-code for the proposed ACO heuristic algorithm explained in the previous section. Figure 7 represents the pseudo-code of proposed ACO heuristic algorithm.

In the algorithm, three main activities of ACO algorithm exist. In the first activity, ants are generated and each ant find a solution that includes machine assignment and machine sequencing. The second activity is Local search Procedure. Local search activity is placed in the algorithm to improve the solution of each ant in each iteration.The third one is applied for pheromone deposit and the last one is for pheromone evaporation.

```
Procedure ACO_Meta_Heuristic()
    While (it ≠ MaxIteration)
        Ant_Generation_and_solution finding();
        Local_Search_Procedure();
        Pheromone Deposit();
        Pheromone_Evaporation();
    end while
end procedure

procedure Ant_Generation_and_solution finding()
    generate new ant;
    While (k ≠ NbANT)
        While (i ≠ NbOrders)
            Prob1= calculate_machine_selection_probabilities_with_ eq2;
            S1= RouletteWheelSelection();
            Prob2=calculate_sequencing_probabilities_for_all_orders_
            assigned_same_machine_with_eq4;
            S2=apply_ant_decision_policy_sequencing();
            Evaluate_TWT(k,i);
            Update_Best_Solution;
        end while
    end while
End procedure

Procedure Evaluate_TWT(k,i)
    While (m ≠ NbMachines)
        If i is the first order on a machine
            CompTime(k,S[m,i])=Release_time(S[m,i])+ProcessingTime(S[m,i],
            m)
        ElseIf  CompTime(k, S[m,i])< Release_time(S[m,i])
            CompTime(k, S[m,i])= Release_time(S[m,i]) +
            ProcessingTime(S[m,i],m)
        Else
            CompTime(k, S[m,i])= CompTime(k, S[m,i-1]) +
            ProcessingTime(S[m,i],m)
        End If

        TWT(k,i)= CompTime(k, S[m,i]) – d(S[m,i])

End procedure

Procedure Pheromone Deposit()
    While (k ≠ NbANT)
        While ( i ≠ NbOrders)
            While ( m ≠ NbMachines)
                If S1[i]=m
                τ^I_im=τ^I_im+φ / TWT_k
                End if

            end while
            While ( m ≠ NbMachines)
                While (y ≠ NumberOfAssignedOrderOnMachine m-1)
                τ^II_S2[m,y] S2[m,y−1]= τ^II_S2[m,y] S2[m,y−1] + φ / TWT_k
                end while
```

```
                    end while
                end while
            end while
end procedures

procedure Pheromone_Evaporation()
            τ^I =(1- ρ) τ^I
            τ^II =(1- ρ) τ^II
end procedures

procedure Roulette_Wheel_Selection()
            generate a random number between[0,1];
            C=CumulativeSum(Prob1);
            Selected_Machine=find(r<=C);
end procedures
```

Figure 7 Pseudo-code for ACO Meta-heuristic

6.5 Steps and Pseudo Code of Local Search Algorithm

In this section, local search algorithm applied in the ant colony optimization algorithm is represented and explained step-by-step. The flow chart of local search algorithm is presented in Figure 8. Then, pseudo code of the local search is shown in Figure 9.

Local search algorithms are one of deamon actions that can be used in ant colony optimizations to improve efficiency of the algorithm (Dorigo et al. 1999; p. 6). Ant colony optimization algortihms generally provide good solution when it is applied with a local search algortihms (Arnout et al. 2009; p. 696).

The local search algorithm is performed in each iteration for each ant solution. Therefore, the first loop of the algorithm repeats as the number of ants. Then, the second loop repeats with the number of MaxItLoc (maximum number of local iteration). In each iteration, a machine is randomly selected, and two orders assigned on the selected machine are chosen. Then, the algorithm swaps these two orders and calculates the TWT_Local(Ant), where it is the objective function value of the local search, with new sequence (S2). If the local search algorithm improves the solution of the ant, then it updates the S2 matrix and objective function value of the ant. If the solution of the local search is also better than the best solution of the algorithm (BESTSOL), then it is also updated by the solution found in local search.

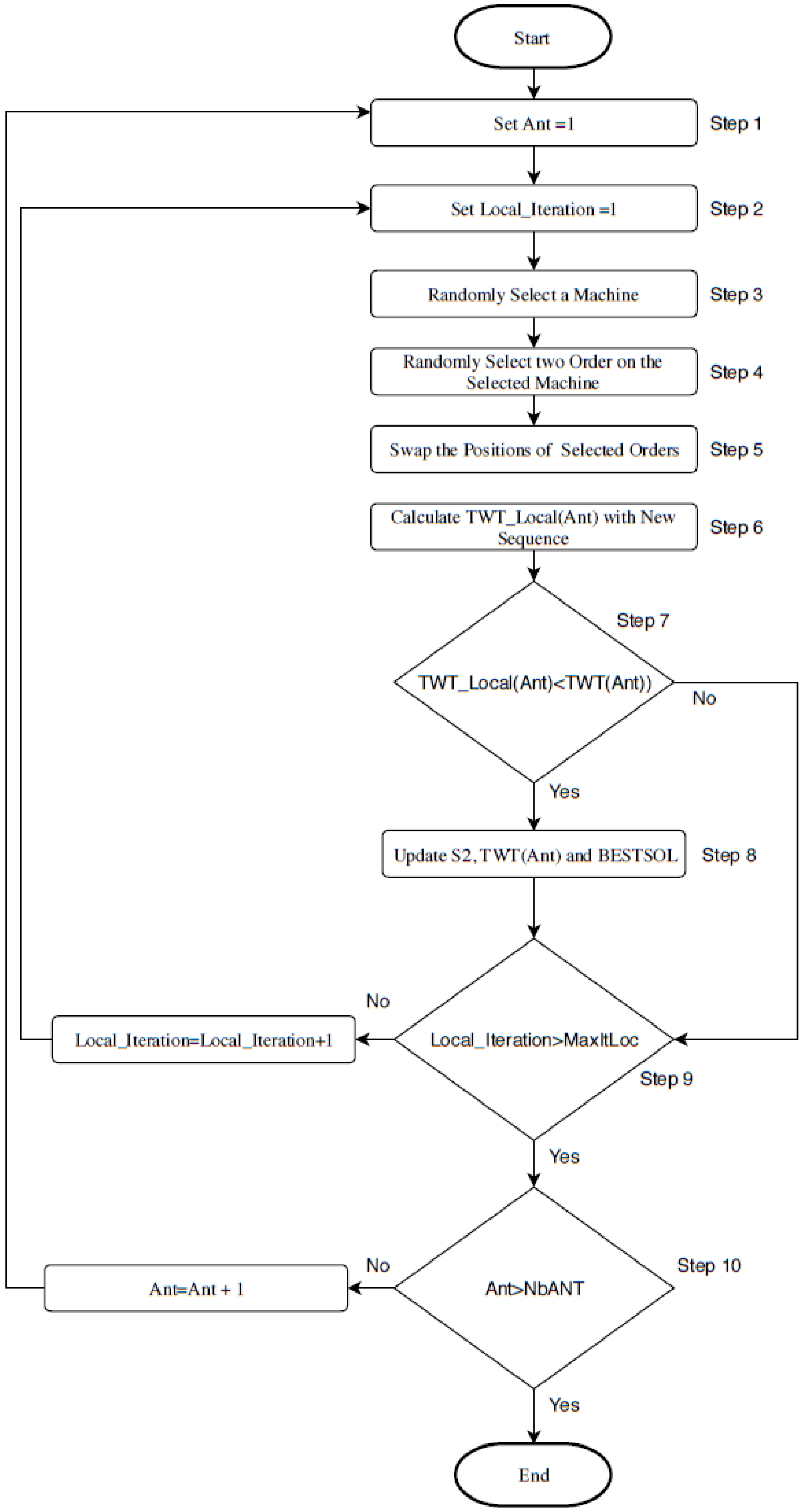


Figure 8 Flow Chart of Local Search Algorithm

The Steps for the Local Search Algorthm is listed below.

Step 1: Set Ant=1

Step 2: Set Local_Iteration=1

Step 3: Randomly select a machine

Step 4: Randomly select two orders on the selected machine

Step 5: Swap the positions of selected orders

Step 6: Calculate TWT_Local(Ant) with new sequence(S2) /* where TWT_Local is the objective function value found in local search algortihm */

Step 7: If TWT_Loc(Ant)≤TWT(Ant), then go Step 8; Otherwise go step 9

Step 8: Replace S2 with the ones found in local search and update TWT(Ant) and BESTSOL

Step 9: If Local_Iteration > MaxItLoc, go step 10; Otherwise, LocalIteration = LocalIteration + 1 and go back to step 2

Step 10: If Ant > NbANT, End the local search; Otherwise, Ant=Ant +1 and go back to step 1

```
Procedure Local_Search_Procedure()
    While (ant ≠ Max_Number_of_Ant)
        While (Current State2 ≠ Max_Iteration_Number_of_Local_Search)
            Randomly_select_a_machine;
            Randomly_select_two_orders;
            Swap_the_positions_of_selected_orders_on_selected_machine
        s;
            Evaluate_ TWT_Local(ant);
            If TWT_Local(ant)<TWT(ant)
                Update_S2(Ant);
                Update_TWT(Ant);
                Update_best_solution(it);
            end if
        end while
    end while
end procedures
```

Figure 9 Pseudo-code of Local Search Algorithm

6.6 Complexity of ACO Algorithm

The complexity of the developed ACO algorithm is explained in two parts in this section. First, the complexity of the thesis problem on which the ACO algorithm is applied is discussed with the scheduling literature. Then, the complexity of the ACO algorithm is discussed by considering the parameters of the algorithm, loops in the algorithm and the computation times from the experimental design of mathematical model and the algorithm under optimized parameters.

With regard to the complexity of the thesis problem, it is regarded as NP-hard by Pinedo (1995; pp. 51), and the other studies in scheduling literature such as Karp et al. (1972) and Ho and Chang et al. (1995). Pinedo (1995; pp. 51) points out that the problem with the objective of tardiness is NP-hard, therefore the weighted tardiness is also considered as NP-hard given in its complexity classes in scheduling.

With regard to the complexity of the developed ACO algorithm, ACO_Meta_Heuristic() that is given in Figure 6 encapculates an outer while loop that is defined by the NbANT parameter. Inside of this loop, there is an another while loop that is defined by the NbOrders parameter, and then this procedure calls the Evaluate_TWT(k,i) procedure has a loop defined by the parameter of NbMachines. Therefore, the complexity of the ACO algorithm can be represented by O(NbANT* NbOrders* NbMachines). In this complexity representation, it is noted that the complexity is linearly proportional to by the change in the number of orders (NbOrders) or number of machines (NbMachines). In addition, computation times oin Table 3 in Section 5 show significant increases under certain number of machines when the number of orders is increased.

6.7 Concluding Remarks

In this chapter, the ant colony optimization algorithm and reasons to apply it for optimization problems are explained briefly in Section 6.2. Then, the proposed ACO algorithm is explained step-by-step with using the flow chart and the pseudo code (see Figures 6 and 7). The local search algorithm integrated with the ACO is explained together with the flow chart and the pseudo code (see Figure 8 and 9).

CHAPTER 7: RESULTS

7.1 Introduction

In this chapter, results of ACO algorithm and experimental design for the problem are given. The ACO parameters which are used in experimental design is given in Section 7.2. Experimental design and the parameters that are used as the factors of experimental design are explained in Section 7.3.

7.2 Optimization of ACO Parameters

Taguchi method is applied to determine the proper values of the ACO parameters which minimize the total weighted tardiness (TWT). Taguchi method is a statistical method which is developed for improving the quality of goods in manufacturing (Atherya et al., 2012; p.13). It is used to determine proper values of control factors to optimize the results of the process. Orthogonal Arrays (OA) are used to define set of experiments in this method.

In this study, the factors and their levels that are considered in the experimental design are shown in Table 8. In the application of Taguchi method for this problem, smaller objective function values represent better types, and the objective function of the analysis is based on S/N$= -10\log_{10}\frac{1}{n}\sum_{i=1}^{n} y_i^2$, where n=Sample Size and y_i^2= Objective Function Value of the ACO Heuristic Algorithm in the run i. Standart L_{27} orthogonal array is used in Table 9.

Table 8 Selected Factors and Their Levels

Factors	Levels		
	0	**1**	**2**
NbANT	40	60	80
ρ	0.01	0.1	0.5
φ	0.01	0.5	1

Table 9 Orthogonal Array (OA) L_{27}

Experiment No	*ρ*	*φ*	*NbANT*
1	0	0	0
2	0	1	1
3	0	2	2
4	1	0	0
5	1	1	1
6	1	2	2
7	2	0	0
8	2	1	1
9	2	2	2
10	0	0	1
11	0	1	2
12	0	2	0
13	1	0	1
14	1	1	2
15	1	2	0
16	2	0	1
17	2	1	2
18	2	2	0
19	0	0	2
20	0	1	0
21	0	2	1
22	1	0	2
23	1	1	0
24	1	2	1
25	2	0	2
26	2	1	0
27	2	2	1

After applying 10 iterations for each experiment, the values used in experiment 11 give a better S/N ratio as shown in Table 10. Therefore, ρ: 0.01, φ: 0.5 and NbANT: 80 are chosen as the best ACO parameters.

Table 10 OA with Control Factors and Results of the Experiments

Experiment No	***ρ***	***φ***	***NbANT***	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Mean**	**S/N**
1	0.01	0.01	40	35	34	33	33	35	34	33	33	36	34	34	-30.6333
2	0.01	0.5	60	36	35	34	34	36	34	33	33	35	34	34.4	-30.735
3	0.01	1	80	33	34	35	32	34	32	34	34	33	34	33.5	-30.5042
4	0.1	0.01	40	36	33	38	35	35	34	36	34	34	35	35	-30.8877
5	0.1	0.5	60	35	36	36	34	35	32	36	33	36	34	34.7	-30.8131
6	0.1	1	80	35	33	35	34	34	32	34	34	34	34	33.9	-30.6066
7	0.5	0.01	40	36	35	34	35	34	35	35	35	34	32	34.5	-30.7602
8	0.5	0.5	60	33	33	33	35	35	36	33	34	33	34	33.9	-30.6081
9	0.5	1	80	34	35	35	34	35	34	33	36	35	36	34.7	-30.8095
10	0.01	0.01	60	34	34	35	33	33	35	34	32	36	34	34	-30.6341
11	0.01	0.5	80	32	31	31	33	34	33	33	33	32	34	32.6	-30.2686
12	0.01	1	40	36	32	34	35	32	35	34	35	36	36	34.5	-30.7639
13	0.1	0.01	60	34	34	33	35	33	36	33	35	34	35	34.2	-30.6841
14	0.1	0.5	80	32	35	33	35	33	34	34	33	34	34	33.7	-30.5557
15	0.1	1	40	35	35	34	36	34	35	37	35	36	36	35.3	-30.9583
16	0.5	0.01	60	34	35	34	32	35	34	34	33	33	36	34	-30.6341
17	0.5	0.5	80	34	33	33	35	35	34	35	32	34	34	33.9	-30.6074
18	0.5	1	40	35	36	35	34	35	36	34	35	34	35	34.9	-30.8583
19	0.01	0.01	80	31	34	34	34	36	34	34	35	34	33	33.9	-30.6096
20	0.01	0.5	40	35	35	34	34	35	36	36	36	35	34	35	-30.8835
21	0.01	1	60	34	34	35	35	34	34	34	34	34	35	34.3	-30.7067
22	0.1	0.01	80	34	34	33	35	34	33	35	35	34	34	34.1	-30.6569
23	0.1	0.5	40	36	35	35	33	35	35	35	36	34	34	34.8	-30.8343
24	0.1	1	60	36	35	34	35	31	34	33	35	33	34	34	-30.6363
25	0.5	0.01	80	34	34	34	32	33	34	34	36	34	33	33.8	-30.582
26	0.5	0.5	40	37	35	34	35	35	35	34	34	35	36	35	-30.8842
27	0.5	1	60	34	36	34	34	33	34	35	33	36	32	34.1	-30.6606

7.3 Comparison of ACO Algorithm to Mathematical Model

In this section, results of the mathematical model (see Section 5.4.2), and the results of the ACO algorithm are compared. The ACO algorithm is run by 5 machines with 12 orders and 6 machines with 22 orders, which are the smallest and the biggest parameter sets in the mathematical model (see Section 5.4.2). Accordingly, the ACO algorithm can find the optimal solution with the smallest parameter set, and it can find near optimal solution with the biggest parameter set.

This disparity is represented in Table 8. Furthermore, when the mathematical model can find the optimal solution in 51 minutes with the biggest parameter set, the proposed ACO algorithm can find near optimal solution in 1.5 minutes. The sequence of the machines, objective function values and the computation times in minutes are given in Table 8. These results represent that the proposed algorithm performs better than mathematical model when the problem size gets bigger.

Table 11 Comparison of Mathematical Model and ACO Algorithm

M-N	TWT of Math. Model	TWT of ACO	Sequences of Math. Model	Sequences of ACO	Computation Time of Math. Model (in min.)	Computation Time of ACO (in min)
5-12	12	12	M1= {8,10,12} M2= {6,3} M3= {2,1} M4= {4,9} M5= {5,7,11}	M1= {8,10} M2= {6,3,12} M3= {2,1} M4= {4,9} M5= {5,7,11}	0.07	0.1
6-22	29	31	M1= {2,20,21} M2= {8,10,1} M3={16,12,18,19} M4={5,11,13,22,4} M5={9,7,17,15} M6={6,3,14}	M1={10,3,19} M2={8,2,12,14,18} M3={1,20,21} M4={5,11,13,17,15} M5={9,7,22,4} M6={16,6}	51	1,5

7.4 Experimental Design

This section explains the experimental design for the ACO algorithm developed. It employes three factors and two levels. Therefore, 2^2*3^1 general multilevel experimental design is performed. Accordingly, this section first defines the factors in Subsection 7.4.1. Then, the results of experimental design are given in Subsection 7.4.2.

7.4.1 Definition of Factors

The proposed ACO heuristic algortihm is implemented by Matlab R2018a running on Windows 8.1 with a Intel Core i5 processor with 4 GB of RAM. The Matlab code is given in Appendix F. The algorithm is run by different combinations of the factors of the experimental design by using the best ACO parameters which are obtained in the Section 7.2. These factors are number of orders, number of machines and number of order (product) type. Each order has one type of product, this why an order type defines a product type.

Number of orders is one of the factors for this experimental design. It impacts the performance of the algorithm. When the number of order is increased, the complexity of the problem is also increased. This characteristic is shown in the mathematical model results for this problem in Section 5.4.2.

Number of machines is an another factor of the experimental design. According to the results of the mathematical model given in Section 5.4.2, the number of machines affects the performance of the scheduling. For this reason, it is selected to see the impact of this parameter.

Number of product types is also a significant factor for both the scheduling problems and the industry. As it is discussed in Section 1.4, it is one of the challenges of the scheduling. For this reason, the impact of this parameter is inquired using the experimental design.

7.4.2 Results of Experimental Design

The proposed ACO algorithm is run with the experiments as shown in Table 9. According to the 2^2*3^1 general multilevel factorial experimental design, 12 number of different combinations are observed by the ACO algorithm in the experimental design and results of them are shown in Appendix G.

The ACO algorithm is also run with a real factory data and this experiment was added to Table 12 as experiment 13. In this experiment the number of machines is 42 and the number of orders is 218. This is the real number of machines that is capable to produce the product type focused in this thesis in the company. 218 is chosed as the number of orders because it is the average number of orders for the product type focused in this thesis in the company.

The deviation ratio between the experiments are calculated by the equation below. The deviation ratio of the experiments are shown in the Table 12. It is obvious that, when the production capacity increases, total weigted tardiness decreases and therefore, deviation ratio also decreases.

$MWTC_c^{Local}$ = The minimum total weighted tardiness cost in experiment c obtained from ACO

$MTWC^{Global}$ = The minimum total weighted tardiness cost in among all combinations obtained from ACO

$$Deviation\ Ratio = \left(\frac{MWTC_c^{Local} - MTWC^{Global}}{MTWC^{Global}}\right) \Big/ 100$$

Table 12 Deviation Ratio and Average Computation Times of Experiments

Experiment No	Number of Machine	Number of Order (Product) Type	Number of Orders	Deviation Ratio	Average Computation Time (in minutes)
1	6	4	60	0.0275	1.40
2	6	4	80	0.115	1.93
3	6	4	100	0.0725	2.42
4	6	7	60	0.1	1.78
5	6	7	80	0.125	2.06
6	6	7	100	0.12	2.61
7	8	4	60	0	1.85
8	8	4	80	0.075	2.10
9	8	4	100	0.095	2.50
10	8	7	60	0.0275	1.7
11	8	7	80	0.0725	2.00
12	8	7	100	0.0875	2.23
13	42	22	218	0.7575	16.5

Figure 10 and 11 represent the average computation time of experimental design in minutes and minimum total weighted tardiness of the experiments, respectively.

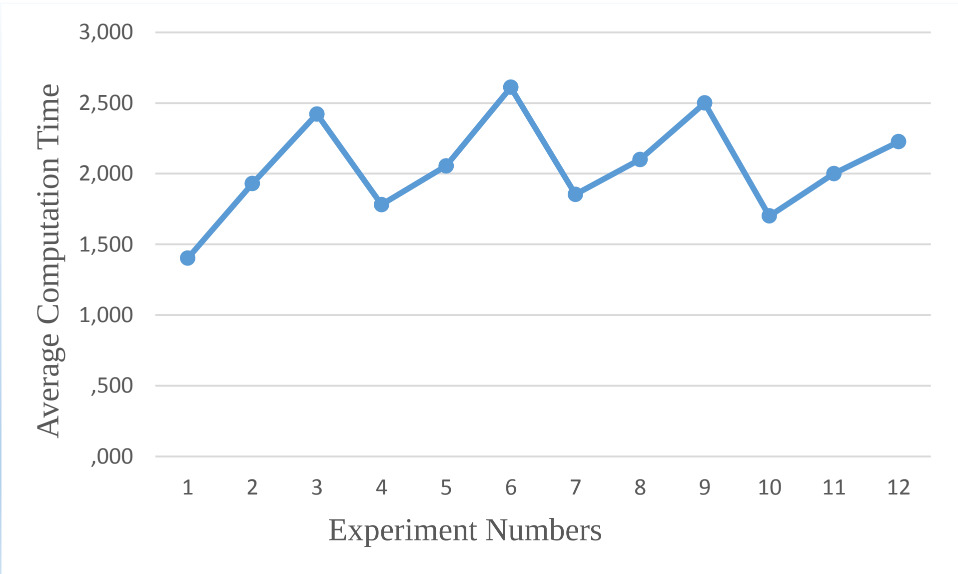


Figure 10 Average Computation Time of Experimental Design in Minutes

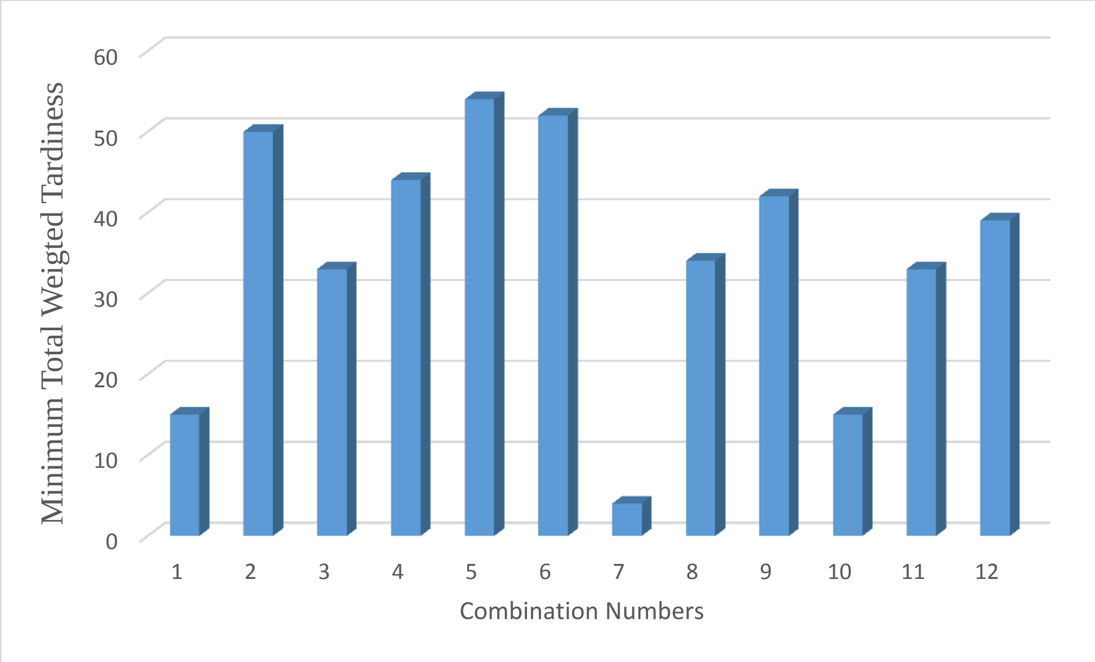


Figure 11 Minimum Total Weighted Tardiness of the Experiments

7.4.2.1 Results of Experimental Design by Minimum Objective Function

Experimental design is employed first by using the minimum objective function value. Accordingly, the results of experimental design are examined first by the main effects plot that is given in Figure 12. According to the results, as number of machines is increased from 6 to 8, the weighted tardiness drops significantly below 30. With regard to number of product types, as number of product varieties is increased from 4 to 7, the weighted tardiness increases significantly beyond 40. These increases are linearly proportional as in the cases of number of machines and number of product types. However, the changes in the weighted tardiness when the number of orders is increased from 60 to 100, it presents a non-linear behavior, it increases from 60 to 80, however at this point, it starts to decrease until 100. It is noted that the weighted tardiness is still beyond 40 when the number of orders is 100. Also, it shows the minimum objective value, that is 20, when the number of orders is 60.

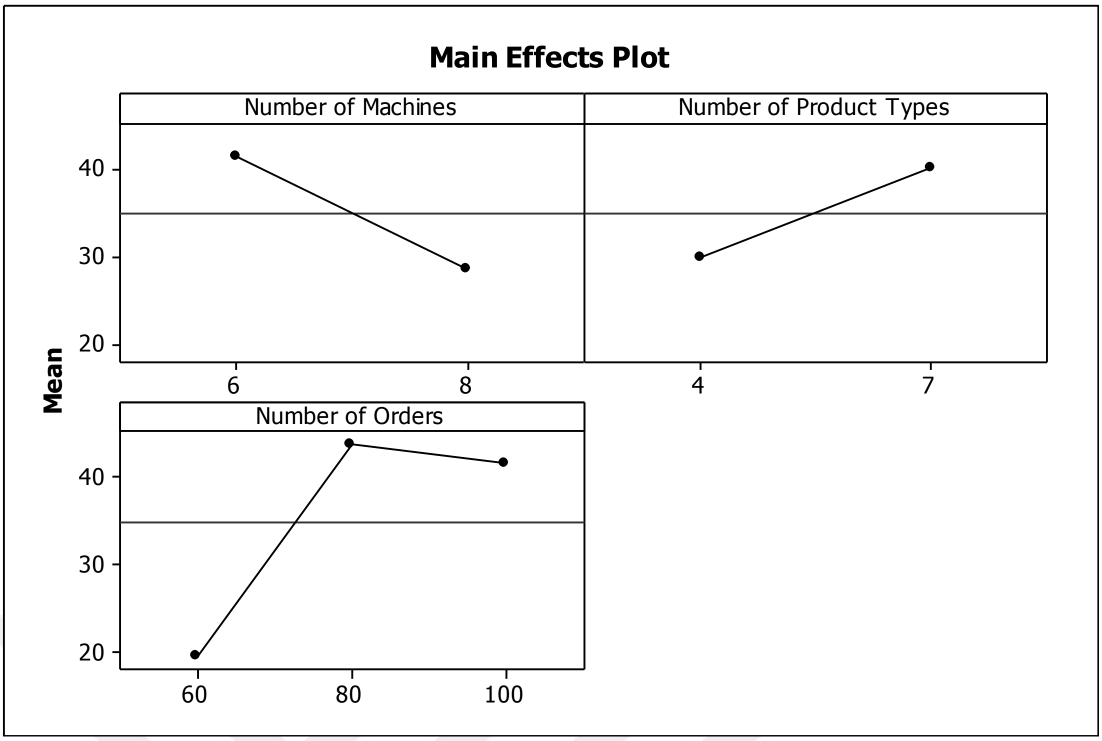


Figure 12 Main Effects Plot for Minimum Objective Function

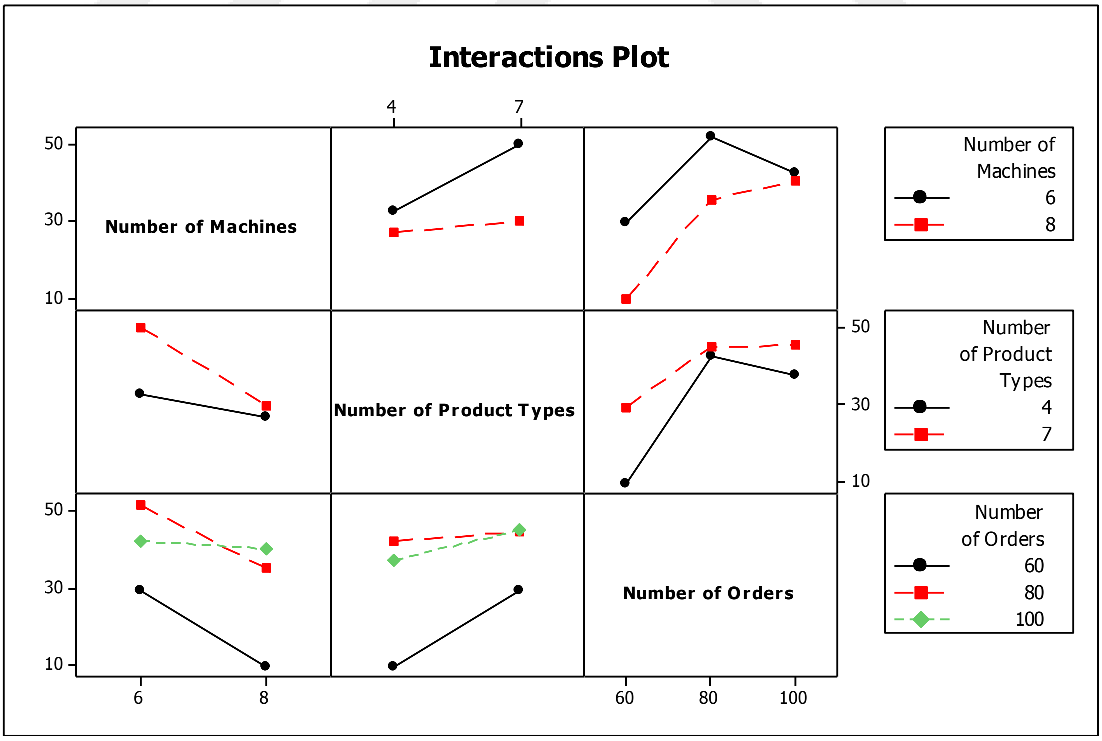


Figure 13 Interactions Plot for Minimum Objective Function

Second, the interactions plot is given in Figure 13. According to the results in Figure 11, in the case of interaction of number of machines and number product types, the increase is substantial when the number of machines is 6 and when the number of product types is increased from 4 to 7.

However, in the case of 8 machines, the increase in the weighted tardiness when the number of product types is increased is very little, and not substantial. The interaction between the number of machines and number of orders represents significant changes as number of orders is increased. The weighted tardiness increases significantly as number of orders is increased from 60 to 80 and it drops from 80 to 100. This behavior is similar to the main effects of number or orders given in Figure 10. The changes are similar when the number of machines is 8, however the weighted tardiness values are smaller. When the increaction is between number of product types and number of orders, the increase impact in the increase of number of orders under both product types are similar. It represents similar increase in the weighted tardiness until the number of orders is 80, then it represents decrease until the number of orders is 100.

7.4.2.2 Results of Experimental Design by Average Computation Time

Experimental design is employed first by using the average computation time value. Accordingly, the results of experimental design are examined first by the main effects plot that is given in Figure 14. According to the results, as number of machines is increased from 6 to 8, the average computation time increases significantly to above 1.95 minutes. With regard to number of product types, as number of product varieties is increased from 4 to 7, the average computation time increases significantly to above 1.95 minutes. With regard to number of orders, as number of orders increases from 60 to 100, the average computation time increases too from 1.55 minutes to almost 2 minutes. These increases are linearly proportional as in the cases of number of machines, number of product types and number of orders.

Second, the interactions plot is given in Figure 15. According to the results in Figure 15, in the case of interaction of number of machines and number product types, the increase when the number of machines is 6 and when the number of product types is increased from 4 to 7, is substantial. Then, in case of 8 machines, almost same amount of incease is observed when the number of product types is increased. The interaction between the number of machines and number of orders represents significant changes as number of orders is increased. The average computation time increases significantly as number of orders is increased from 60 to 100. The changes are similar when the number of machines is 8, however the average computation time values are smaller. When the interaction is between number of product types and number of orders, the increase impact in the increase of number of orders under both number of product types are similar for the number of orders between 60 to 80. However, after 80 number of orders, the average computation time decreases when the number of product types is 4 and the average computation time continue to increase when the number of product types is 7.

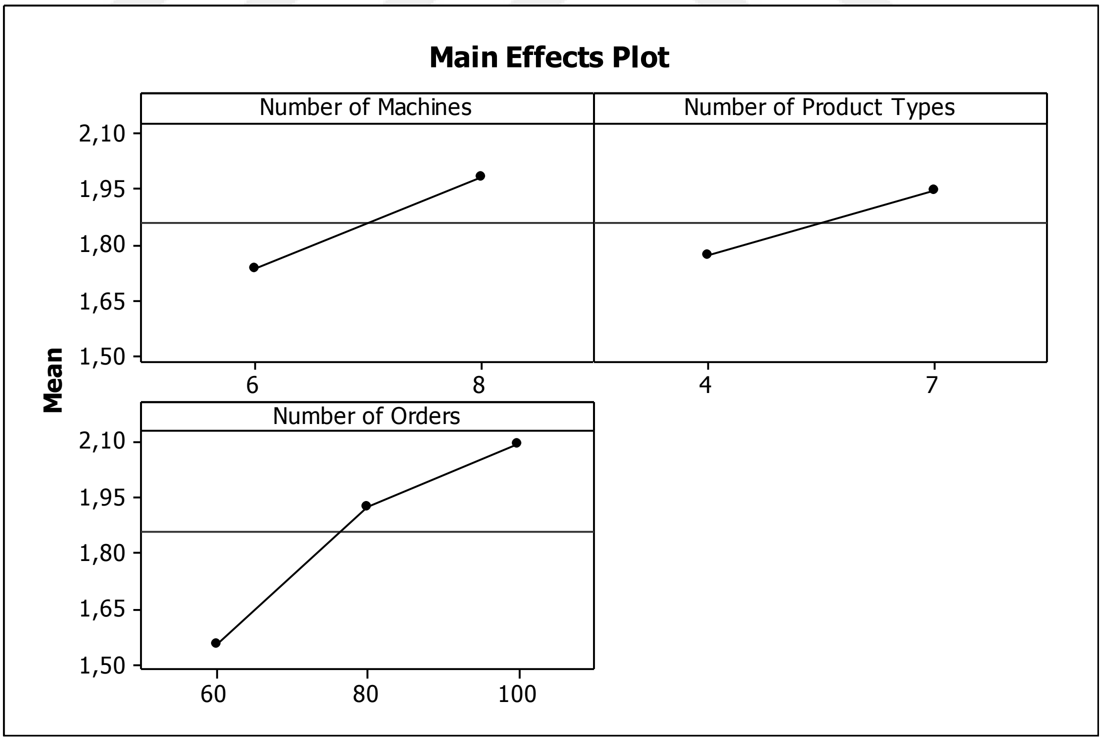


Figure 14 Main Effects Plot for Average Computation Time

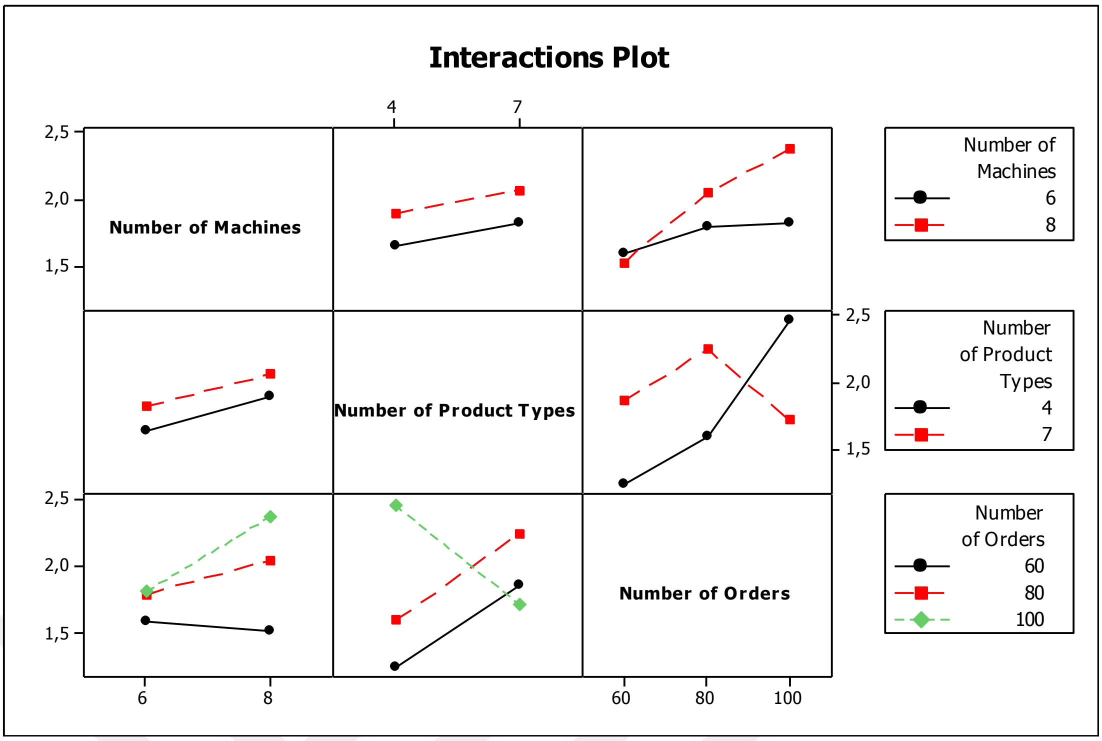


Figure 15 Interactions Plot for Average Computation Time

7.4.2.3 Regression Model by Average Computation Time

Based on the multilevel experimental design, the regression model of the design is computed to show the main effects. The regression model is calculated as in the following.

Min Obj Func = 17,2 - 6,42 Number of Machines + 3,39 Number of Product Types + 0,550 Number of Orders

The normal probability and residual plots for the regression model is also presented in Figure 14. However, it is noted that R square value is calculated less then 90%, that represents the fitting on the regression could be better with the increased number of problem instances (see Appendix L).

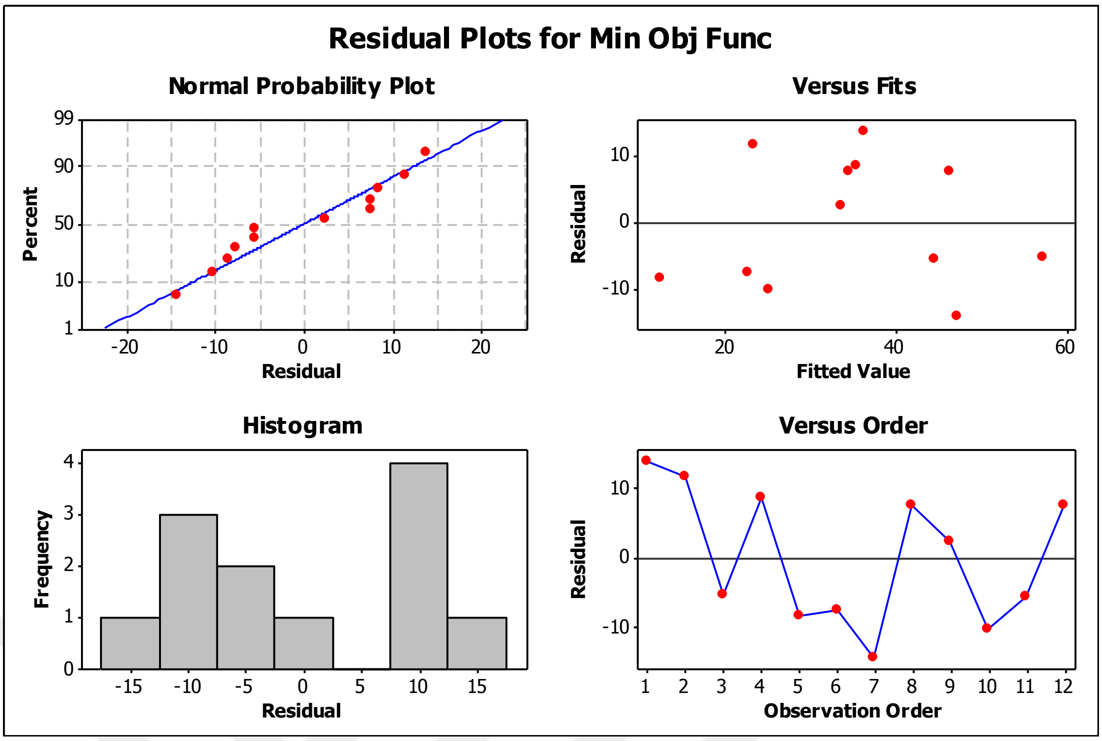


Figure 16 Residual Plots for Minimum Weighted Tardiness

7.5 Concluding Remarks

In this Chapter, the appropriate ACO parameters are found. Then, the mathematical model is compared with ACO algorithm and discussed. After that, the experimental design is performed with the appropriate ACO parameters. Furthermore, the developed ACO algorithm is also run for the real data of the company. The average computation time is found as 16.5 minutes for 42 number of machines, 22 number of product types and 218 number of orders. In this experiment, the objective function minimum objective function value is found as 307 and 35 number of tardy orders are observed. Finally, the results are represented in Section 7.4.2.

CHAPTER 8: CONCLUSIONS AND FUTURE WORK

In this chapter, the conclusions are given in section 8.1 and the future research is discussed in Section 8.2.

8.1 Conclusions and Discussions

This thesis considers a real life problem in a textile company that produces knitted fabric. The problem is to minimize the total weighted tardiness on the knitted fabric production. In this study, a mathematical model is improved which minimizes total weighted tardiness based on recent literature. The model considers machine eligibility constraints and sequence-dependent setup times. The mathematical model is run for a number of machines and orders. The weighted total tardiness values are recorded for different scenarios under the mathematical model developed. It is observed that the unrelated parallel machine problem with weighted tardiness objective is an NP-hard problem and it takes significantly long time to find a solution using the optimization software.

An ant colony optimization algorithm is developed to solve the problem with real size dataset extracted from a real textile company and it shows acceptable computation times. To show the effectiveness of the developed ACO algorithm, the algorithm is first compared with the mathematical model and it is proven that the ACO algorithm performs better than the mathematical model. However, this results should be examined with more problem instances and data.

Furthermore, the best ACO parameters are found by several experiments. Then, design of experiments is performed for the proposed ACO algorithm and their results are analyzed by ANOVA, with the factors as follows: number of orders, number of machines and number of product types.

The interactions between these factors are shown by using ANOVA. It is shown that the best weighted tardiness values are obtained when the number of orders is 60, number of machines is 6 and the number of product types is 4. This is also verified by the average computational results. In additon to the main effects representation of these factors, a regression model is computed using statistical software and normality of results with residuals are verified. According to the regression model, number of machines has negative reducing impact on tardiness. An increase on the number of machines causes to decrease the total weigted tardiness. While the number of order number has low impact on the objective function, the number od product types has higher impact on the total weighted tardiness.

In short, this thesis contributes to the literature under the following points.

- It applies a real scheduling problem using real data in the textile industry as it lacks of such optimization studies.
- With a detailed literature, it is verified only a few population based heuristics are applies on scheduling of textile production systems, and this thesis adds a problem categories on a recent literature.
- The mathematical model applied incorporates machine eligibility constraints that adds additional complexity on running the model.
- The results of developed ACO algorithm are compared to the results of the mathematical model. Significant savings on computational times are observed.
- The ACO algorithm is validated on a large scale problem with real industry data and its efficiency is verified by obtaining an acceptable solution.

8.2 Future Research

This study can be further improved using a mix of further methods. Such methods could be hybrid heuristics that could improve the solution quality especially under larger size of real datasets. In addition, ant colony approach can be further combined specifically under daemon actions, local search. Further, lookahead and backtracking on the solutions could be added on obtaining better results. Moreover, the developed ACO algorithm should be examined with bigger data sets under real production environment. In addition, the algorithm can be improved by adding the lot sizing property so that the algorithm can schedule the orders by splitting them.

Apart from improvements regarding with the ACO algorithm, there can be further analyses with different sets of customer priorities and tardiness weights. The overall penalty costs can be correlated with weighted tardiness values. Further, experiments on product types can be improved.

REFERENCES

AEA, Sectorial Information Center, Aegean Textile and Raw Materials First Quarter Report of 2018, address: http://upload.eib.org.tr/20150512/00000000005036.pdf

Allahverdi, A., Aldowaisan, T., & Gupta, J. N. D. 1999. "A review of scheduling research involving setup considerations." Omega, Vol. 27, No. 2, pp. 219–239.

Anderson S.W. 1995. "Measuring the Impact of Product Mix Heterogeneity on Manufacturing Overhead Cost." The Accounting Review, Vol. 70, No. 3, pp. 363-387

Arnaout, J.-P., G. Rabadi, and R. Musa. 2009. "A Two-stage Ant Colony Optimization Algorithm to Minimize the Makespan on Unrelated Parallel Machines with Sequence-dependent Setup Times." Journal of Intelligent Manufacturing, Vol. 21, No. 6, pp. 693–701

Athreya, S., and Venkatesh, DR. Y. D. 2012. "Application Of Taguchi Method For Optimization Of Process Parameters In Improving The Surface Roughness Of Lathe Facing Operation." International Refereed Journal of Engineering and Science, Vol. 1, No. 3, pp. 13- 19

Behnamian, J., M. Zandieh, and S. Fatemi Ghomi. 2009. "Parallel-machine Scheduling Problems with Sequence-dependent Setup Times Using an ACO, SA and VNS Hybrid Algorithm." Expert Systems withApplications, Vol. 36 No. 6, pp. 9637–9644.

Bertrand, J. and Fransoo, J. 2002. Operations Management Research Methodologies Using Quantitative Modeling. International Journal of Operations & Production Management, Vol. 22, pp. 241-264.

Bilge, U., F. Kiraç, M. Kurtulan, and P. Pekgün. 2004. “ATabu Search Algorithm for Parallel MachineTotal Tardiness Problem.” Computers & Operations Research, Vol. 31, pp. 397–414.

Chen R.-C., P.-H. Hung, M.-C. Wu. 2007. “Scheduling Production Using Genetic Algortihm for Elastic Knitted Fabrics with Wide Ranges of Quantities Demanded.” Proceedings of the 7th WSEAS International Conference on Simulation, Modelling and Optimization, Beijing, China, September 15-17, pp. 182-187.

Dorigo, M., Di Caro, G. and Gambardella, LM. 1999. “Ant algorithms for distributed discrete optimization.” Artificial Life Vol. 5 No. 2, pp. 137–172

Garey, M. R., and Johnson, D. S. 1979. “Computers and Intractability: A Guide to the Theory of NP-Completeness.” San Francisco: W. H. Freeman and Company, Vol.5 No.1B,

Hamzadayi, A. and G. Yildiz. 2017. “Modeling and Solving Static m Identical Parallel Machines Scheduling Problem with a Common Server and Sequence-dependent Setup Times.” Computers & Industrial Engineering, Vol. 106, pp. 287–298

Joo, C.M. and B.S. Kim. 2015. “Hybrid Genetic Algorithms with Dispatching Rules for Unrelated Parallel Machine scheduling with Setup Time and Production Availability.” Computers & Industrial Engineering, Vol. 85, pp. 102–109

Karp, R. M. 1972. Reducibility among combinatorial problems. In R. E. Miller & J.W. Tatcher (Eds.), Complexity of computer computations. New York: Plenum Press, pp. 85–103.

Kayvanfar, V. and GH.M Komaki., A. Aalaei, M. Zandieh. 2014. “Minimizing Total Tardiness and Earliness on Unrelated Parallel Machines with Controllable Processing Times.” Computers & Operations Research, Vol. 41, pp. 31–43.

Kerkhove, L.P. and M. Vanhoucke. 2014. “Scheduling of Unrelated Parallel Machines with Limited Server Availability on Multiple Production Locations: A Case Study in Knitted Fabrics.” International Journal of Production Research, Vol. 52, No. 9, pp. 2630–2653

Koulamas, C. 1997. “Decomposition and Hybrid Simulated Annealing Heuristics for the Parallel-Machine Total Tardiness Problem.” Naval Research Logistics, Vol. 44, pp. 109-125

Lee J.-H, J.-M. Yu and D.-H. Lee. 2013.“A Tabu Search Algorithm for Unrelated Parallel Machine Scheduling with Sequence- and Machine-dependent Setups: Minimizing Total Tardiness” International Journal of Advanced Manufacturing Technology, Vol. 69, No. 9–12, pp. 2081–2089

Lin, S.-W., Z.-J. Lee, K.-C.Ying, and C.-C. Lu. 2011. “Minimization of Maximum Lateness on Parallel Machines with Sequence-dependent Setup Times and Job Release Dates.” Computers & Operations Research, Vol. 38, No. 5, pp. 809–815.

Lin C.-W., Y.-K. Lin and H.-T. Hsieh. 2013. “Ant Colony Optimization for Unrelated Parallel Machine Scheduling” International Journal of Advanced Manufacturing Technology, Vol. 67, pp. 35–45

Lin Y.-K and F.-Y. Hsieh. 2013. “Unrelated Parallel Machine Scheduling with Setup Times and Ready Times.” International Journal of Production Research, Vol. 52, No. 4, pp. 1200–1214,

Mendes, A., and F. Muller. 2002. “Comparing Metaheuristic Approaches for Parallel Machine Scheduling Problems with Sequence-dependent Setup Times.” Production Planning & Control, 2002, Vol. 13, No. 2, pp. 143-154.

Ngai E.W.T., S. Peng, P. Alexander, K.K.L. Moon, 2014.“Decision Support and Intelligent Systems in The Textile and Apparel Supply Chain: An academic Review of Research Articles.” Expert Systems with Applications Vol. 41, pp. 81–91

Pimentel C., F. Alveos, A. Duarte and J.M.V. Carvalho. 2006. “A Scheduling Model for a Knitted Planning Problem.” Manufacturing Fundamentals: Necessity and Sufficiency, Ch. 16

Pinedo, M. L. 2008. “Scheduling Theory, Algorithms and Systems.” (3rd ed.)

Radhakrishnan, S. and Ventura, J. 2000. “Simulated Annealing for Parallel Machine Scheduling with Earliness-Tardiness Penalties and Sequence-dependent Set-up Times.” International Journal of Production Research, Vol. 38, pp. 2233–2252

Tavakkoli-Moghaddam, R., F.Taheri, M. Bazzazi,M. Izadi, and F. Sassani. 2009. “Design of a Genetic Algorithm for Bi-objective Unrelated Parallel Machines Scheduling with Sequence-dependent Setup Times and Precedence Constraints.” Computers & Operations Research Vol. 36 No. 12, pp. 3224–3230

The Textile Hub. 2013. “Production Planning and Scheduling Software for the Textile Industry”. Accessed on https://textlnfo.wordpress.com/2013/01/23/production-planning-and scheduling-software-for-the-textile-industry/ [January 23, 2015]

TTM Machine, 2018. Online catalog for products. Accessed on https://www.ttmmakine.com/eng/products

Sen, A. 2014.“The US fashion industry: A supply chain review.” International Journal of Production Economics, Vol. 114, pp. 571-593

Vallada, E., and R. Ruiz. 2011. “A Genetic Algorithm for the Unrelated Parallel Machine Scheduling Problem with Sequence Dependent Setup Times.” European Journal of Operational Research Vol. 211, pp. 612–622

APPENDICES

Appendix A

List of Pus and Fein Values

Pus	Fein
10	**12**
11	**15**
12	**16**
13	**18**
14	**20**
15	**22**
16	**28**
17	**36**
18	**44**
19	
20	
22	
26	
30	
32	
34	
38	

Appendix B

Id Definitions of Machine Types According to the Knitting Types

TYPE_ID	TYPE
1	SINGLE JERSEY
3	VANIZE SINGLE JERSEY
4	PIQUE
5	2 THREAD
6	TOWEL
7	POLAR
8	3 THREAD
9	RIBB
10	TRICOT
11	INTERLOCK

Appendix C

Machine Type Id, Pus and Fein Values

Machine Type	Pus	Fein	Type
1	10	16	9
2	11	16	9
3	12	16	9
4	12	28	6
5	13	16	9
6	13	28	1
7	14	16	9
8	14	28	1
9	15	16	9
10	15	16	9
11	15	28	1
12	16	15	9
13	16	18	9
14	16	28	1
15	16	18	9
16	17	15	9
17	17	16	9
18	17	28	1
19	17	16	9
20	18	18	9
21	18	18	9
22	19	18	9
23	19	16	9
24	20	18	9
25	22	18	9
26	30	22	1-4-5
27	30	28	1-4-5
28	30	18	9-11
29	30	28	1-3-4-5
30	30	28	9-11
31	30-34	18-20	9-11
32	32	28	1-4-5
33	32	28	1-3-4-5
34	32	28	1-3

35	32	28-36	1-3
36	32	28	1
37	32	22	1-3-4-5
38	32	20	1-3-8
39	32	22	1-4-5
40	34-38	18-20	9-11
41	34	16	9
42	34	28	1-4-5
43	34-38	18-20-28	9-11
44	34	28	11
45	34	18	9-11
46	34	22-28	1-3

Appendix D

Mathematical Model in OPL

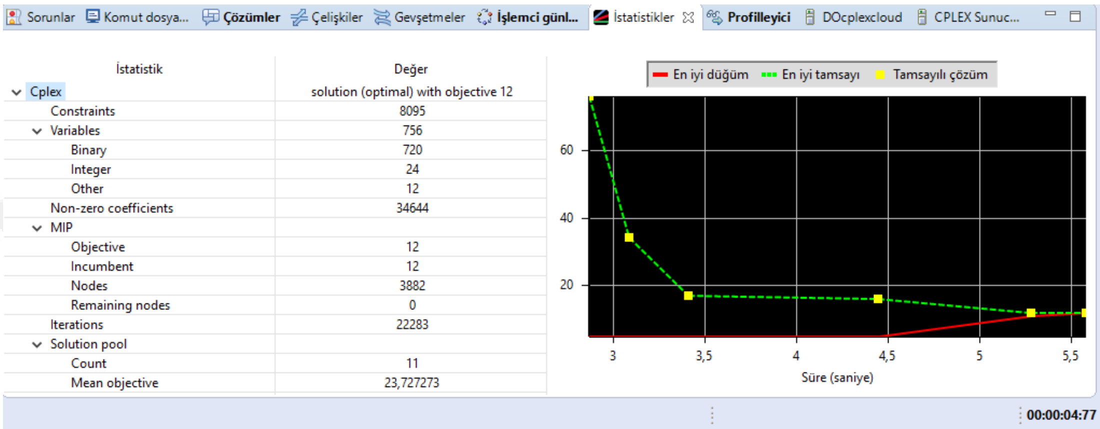
```
int nborders=22;
range order=1..nborders;

int nbmachines=6;
range machine=1..nbmachines;

int MPozitions=22;
range position=1..MPozitions;

int w[1..nborders]=...; // Weight of order j
int R[1..nborders]=...; //Release time of job j in day
int D[1..nborders]=...; //due date of oder in day
int Q[1..nborders]=...; // quantity of order
float P[i in 1..nborders,k in machine]=...; //machine dependent processing times of
order in day
float S[1..nborders,1..nborders]=...;// Time needed to switch from job j to job i in
hour
int M=100000; // big M
int U[1..nborders,1..nbmachines]=...; // machine eligibility set

dvar float+ C[1..nborders]; //Completion time of job j in hour
dvar int+ Cday[1..nborders]; // Completion time of job j in day
dvar int+ T[1..nborders]; // Number of days job j is late
dvar int+ X[order,position,machine] in 0..1; // 1 if job j is planned on position k of
machine m, otherwise 0



minimize sum(j in order)(w[j]*T[j]);
        subject to {

                forall(j in order){
                        cnst1:C[j]/24<=Cday[j];
                }


                forall(j in order){
                        cnst2:(sum(m in machine:U[j,m]>0)(sum(k in position)
X[j,k,m]))==1;
                }
```

```
forall(j in order){
	cnst3:(sum(m in machine)(sum(k in position) X[j,k,m]))==1;

}

forall(k in position, m in machine ){
	cnst4:sum(j in order) X[j,k,m]<=1;
}

forall(k in position,m in machine:k>1){
	cnst5:(sum(j in order) X[j,k,m])-(sum(j in order)X[j,k-
1,m])<=0;
}

forall(j in order,i in order, k in position, m in machine:k>1){
	cnst6:C[i]+M*(2-X[i,k,m]-X[j,k-
1,m])>=((P[i,m]*Q[i])/60)+S[j,i]+C[j];
}
forall(j in order){
	cnst7:C[j]>=R[j]+(sum(m in machine)(sum(k in
position)((P[j,m]*Q[j]/60)*X[j,k,m])));
}

forall(j in order){
	cnst8:T[j]>=Cday[j]-(sum(k in position)(sum(m in
machine)(X[j,k,m]*D[j])));
}

}
```

Appendix E

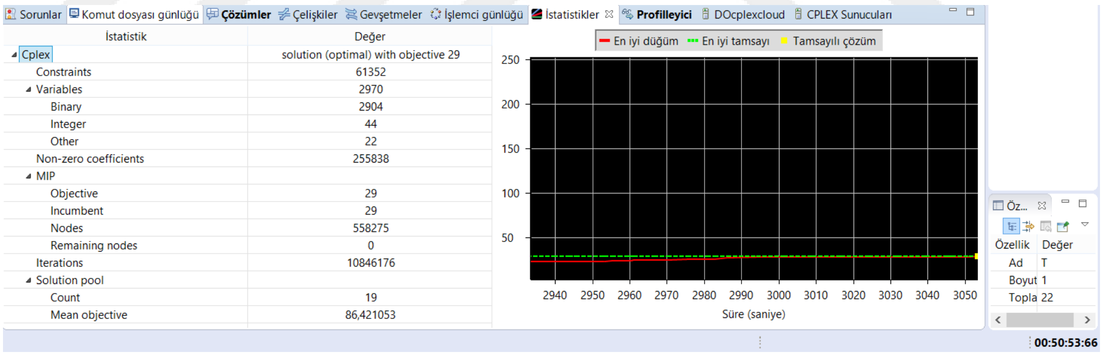
Statistics of Mathematical Models in OPL

5 Machines and 12 Orders:



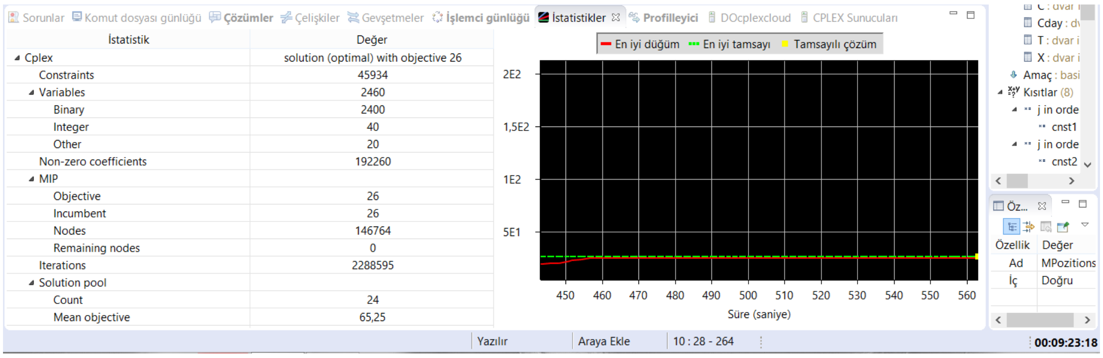
İstatistik	Değer
Cplex	solution (optimal) with objective 12
Constraints	8095
Variables	756
Binary	720
Integer	24
Other	12
Non-zero coefficients	34644
MIP	
Objective	12
Incumbent	12
Nodes	3882
Remaining nodes	0
Iterations	22283
Solution pool	
Count	11
Mean objective	23,727273

6 Machines and 22 Orders:



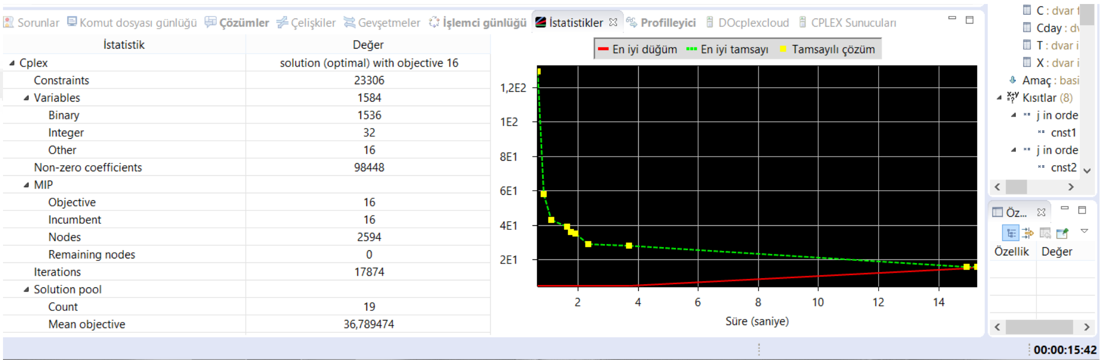
İstatistik	Değer
Cplex	solution (optimal) with objective 29
Constraints	61352
Variables	2970
Binary	2904
Integer	44
Other	22
Non-zero coefficients	255838
MIP	
Objective	29
Incumbent	29
Nodes	558275
Remaining nodes	0
Iterations	10846176
Solution pool	
Count	19
Mean objective	86,421053

6 Machines and 20 orders:



İstatistik	Değer
Cplex	solution (optimal) with objective 26
Constraints	45934
Variables	2460
Binary	2400
Integer	40
Other	20
Non-zero coefficients	192260
MIP	
Objective	26
Incumbent	26
Nodes	146764
Remaining nodes	0
Iterations	2288595
Solution pool	
Count	24
Mean objective	65,25

6 Machines and 16 Orders:



İstatistik	Değer
Cplex	solution (optimal) with objective 16
Constraints	23306
Variables	1584
Binary	1536
Integer	32
Other	16
Non-zero coefficients	98448
MIP	
Objective	16
Incumbent	16
Nodes	2594
Remaining nodes	0
Iterations	17874
Solution pool	
Count	19
Mean objective	36,789474

Appendix F

ACO Matlab Code with Local Search Algorithm

```
%% ACO Main Loop

for it=1:MaxIt
%% Ant generation
    for k=1:NbANT
%% Finding S1
      for i=1:NbOrders

         Prob1=tau1(i,:).^alfa.*eta1(i,:).^beta;
         NonzeroProb1=find(Prob1~=0);
         EligibleProb1=Prob1(NonzeroProb1);
         EligibleProb1=EligibleProb1/sum(EligibleProb1);
         x=RWSLocal(EligibleProb1);
         %x=find(Prob1 == max(Prob1(:)));
         %endd=x(end);
         S1(k,i,it)=NonzeroProb1(x);
      end
%% Finding S2
      for j=1:NbMachines
          Prob2=zeros(NbOrders,NbOrders);
          DummyS1=S1(k,:,it);
          NotAssignedOrders=find(DummyS1~=j);
          AssignedOrders=find(DummyS1==j);
          sizeofarray1=size(NotAssignedOrders,2);
          sizeofarray2=size(AssignedOrders,2);
          if isempty(AssignedOrders)
          else
          if sizeofarray2==1
          S2(j,1,k,it)=AssignedOrders;
          else

          for t=1:NbOrders

            for g=1:sizeofarray2
                if t==AssignedOrders(g)
                    Prob2(t,:)=tau2(t,:).^alfa2.*eta2(t,:).^beta2;
                    Prob2(t,:)=Prob2(t,:)/sum(Prob2(t,:));
                    Prob2(t,t)=0;
                    for r=1:sizeofarray1
                        index1=NotAssignedOrders(r);
                        Prob2(t,index1)=0;
                    end
                else
                end
            end
          end
```

```
            [M,N]=find(Prob2==max(Prob2(:)));
            S2(j,1,k,it)=M(1);
            S2(j,2,k,it)=N(1);
             for h=3:sizeofarray2

                 indx=S2(j,h-1,k,it);
                 DummyArray=Prob2(indx,:);
                 for z=1:NbOrders
                     if S2(j,z,k,it)~=0
                     DummyArray(S2(j,z,k,it))=0;
                     end
                 end

                 seq=find(DummyArray==max(DummyArray(:)));

                 S2(j,h,k,it)=seq(end);
             end
            end
            end
        end
%% TWT Calculation
        for l=1:NbMachines
            if S2(l,1,k,it)==0
            else

C(k,S2(l,1,k,it),it)=RT(S2(l,1,k,it))+((ProcessTime(S2(l,1,k,it),l)*
Quantity(S2(l,1,k,it)))/60);
            Cday(k,S2(l,1,k,it),it)=ceil(C(k,S2(l,1,k,it),it)/24);
            Tday(k,S2(l,1,k,it),it)=d(S2(l,1,k,it))-
Cday(k,S2(l,1,k,it),it);
               if Tday(k,S2(l,1,k,it),it)<0

Tday(k,S2(l,1,k,it),it)=abs(Tday(k,S2(l,1,k,it),it));
               else
                    Tday(k,S2(l,1,k,it),it)=0;
               end
            for f=2:NbOrders
                if S2(l,f,k,it)==0
                else
                    if (C(k,S2(l,f-1,k,it),it)+Setup(S2(l,f-
1,k,it),S2(l,f,k,it)))<RT(S2(l,f,k,it))

C(k,S2(l,f,k,it),it)=((ProcessTime(S2(l,f,k,it),l)*Quantity(S2(l,f,k
,it)))/60)+RT(S2(l,f,k,it));
                    else
                          C(k,S2(l,f,k,it),it)=(C(k,S2(l,f-
1,k,it),it)+(((ProcessTime(S2(l,f,k,it),l)*Quantity(S2(l,f,k,it))))/
60)+Setup(S2(l,f-1,k,it),S2(l,f,k,it)));
                    end

Cday(k,S2(l,f,k,it),it)=ceil((C(k,S2(l,f,k,it),it)/24));
                          Tday(k,S2(l,f,k,it),it)=d(S2(l,f,k,it))-
Cday(k,S2(l,f,k,it),it);
                if Tday(k,S2(l,f,k,it),it)<0

Tday(k,S2(l,f,k,it),it)=abs(Tday(k,S2(l,f,k,it),it));
                else
                    Tday(k,S2(l,f,k,it),it)=0;
                end
                end
            end
```

```
            end
        end

        CostFunction(k,it)=0;
        for s=1:NbOrders
        CostFunction(k,it)=CostFunction(k,it)+(Tday(k,s,it)*w(s));
        end

        if CostFunction(k,it)<BestSol
            BestSol=CostFunction(k,it);
            BestAnt=k;
            BestS1=S1(k,:,it);
            BestS2=S2(:,:,k,it);
            BestIt=it;
        end
      end

%% Local Search Algortihm
       for n=1:NbANT
          for ga=1:MaxItLoc
           CdayLocal=zeros(1,NbOrders);
           CLocal=zeros(1,NbOrders);
           TdayLocal=zeros(1,NbOrders);
           addlist=[];
           for yx=1:NbMachines
               S2Dummy=S2(yx,:,n,it);
               NonZeroS2Dummy=find(S2Dummy~=0);
               if size(NonZeroS2Dummy,2)>1
                   addlist=[addlist yx];
               else
               end
           end
           randomno=randi([1 size(addlist,2)]);
           ha=addlist(randomno);
           DummyS2Loc(:,:)=S2(:,:,n,it);
              indx1=size(find(DummyS2Loc(ha,:)~=0),2);
              RandNo1=randi([1 indx1]);
              RandNo2=randi([1 indx1]);
              dum=DummyS2Loc(ha,RandNo1);
              DummyS2Loc(ha,RandNo1)=DummyS2Loc(ha,RandNo2);
              DummyS2Loc(ha,RandNo2)=dum;
              for la=1:NbMachines
                  if DummyS2Loc(la,1)==0
                  else

CLocal(DummyS2Loc(la,1))=RT(DummyS2Loc(la,1))+((ProcessTime(DummyS2L
oc(la,1),la)*Quantity(DummyS2Loc(la,1)))/60);

CdayLocal(DummyS2Loc(la,1))=ceil((CLocal(DummyS2Loc(la,1))/24));
                    TdayLocal(DummyS2Loc(la,1))=d(DummyS2Loc(la,1))-
CdayLocal(DummyS2Loc(la,1));
                         if TdayLocal(DummyS2Loc(la,1))<0

TdayLocal(DummyS2Loc(la,1))=abs(TdayLocal(DummyS2Loc(la,1)));
                         else
                            TdayLocal(DummyS2Loc(la,1))=0;
                         end


                    for fa=2:NbOrders
```

```
                          if DummyS2Loc(la,fa)==0
                          else

                             if (CLocal(DummyS2Loc(la,fa-
1))+Setup(DummyS2Loc(la,fa-
1),DummyS2Loc(la,fa)))<RT(DummyS2Loc(la,fa))

CLocal(DummyS2Loc(la,fa))=((ProcessTime(DummyS2Loc(la,fa),la)*Quanti
ty(DummyS2Loc(la,fa))/60)+RT(DummyS2Loc(la,fa)));
                             else

CLocal(DummyS2Loc(la,fa))=(CLocal(DummyS2Loc(la,fa-
1))+(((ProcessTime(DummyS2Loc(la,fa),la)*Quantity(DummyS2Loc(la,fa))
))/60)+Setup(DummyS2Loc(la,fa-1),DummyS2Loc(la,fa)));
                             end

CdayLocal(DummyS2Loc(la,fa))=ceil((CLocal(DummyS2Loc(la,fa))/24));

TdayLocal(DummyS2Loc(la,fa))=d(DummyS2Loc(la,fa))-
CdayLocal(DummyS2Loc(la,fa));
                             if TdayLocal(DummyS2Loc(la,fa))<0

TdayLocal(DummyS2Loc(la,fa))=abs(TdayLocal(DummyS2Loc(la,fa)));
                             else
                                   TdayLocal(DummyS2Loc(la,fa))=0;
                             end
                          end
                      end
                 end
             end


              for sa=1:NbOrders

CostFunctionLocal(n,ga,it)=CostFunctionLocal(n,ga,it)+(TdayLocal(sa)
*w(sa));
              end

              if CostFunctionLocal(n,ga,it)<CostFunction(n,it)
                  CostFunction(n,it)=CostFunctionLocal(n,ga,it);
                  S2(:,:,n,it)=DummyS2Loc(:,:);
                  Tday(n,:,it)=TdayLocal;
                  Cday(n,:,it)=CdayLocal;
                  C(n,:,it)=CLocal;
                  if CostFunction(n,it)<BestSol
                  BestSol=CostFunction(n,it);
                  BestAnt=n;
                  BestIt=it;
                  BestS1=S1(n,:,it);
                  BestS2=S2(:,:,n,it);
                  else
                  end
              end
         end



      end
%% Pheromone deposit
      for antt=1:NbANT
      for a=1:NbOrders
```

```
          for v=1:NbMachines
              if S1(antt,a)==v

                  DeltaTau1(a,v)=0.5/CostFunction(antt,it);
                  tau1(a,v)=tau1(a,v)+DeltaTau1(a,v);

              end

          end
     end

     for x=1:NbMachines
         for y=1:size(find(S2(x,:,antt,it)~=0),2)-1

tau2(S2(x,y,antt,it),S2(x,y+1,antt,it))=tau2(S2(x,y,antt,it),S2(x,y+
1,antt,it))+(0.5/CostFunction(antt,it));
         end

     end

     end
%% Pheromone Evaporation
    tau1=(1-eva)*tau1;
    tau2=(1-eva)*tau2;
%% Best solution
    BestCost(it)=BestSol;
    disp(['Iteration ' num2str(it) ': Best Cost = '
num2str(BestCost(it))]);
end

%% Results

figure;
plot(BestCost,'LineWidth',2);
xlabel('Iteration');
ylabel('Best Cost');
grid on;
```

Appendix G

Results of the ACO Algorithm

Experiment No	Best Sequence of Machines	Obj. Value	Average Computation Time in Minutes
1	M1={59,35,1,48,10,11,3,39,56,12,51,4} M2={34,33,37,36,7,9,8,41,14,6,13} M3={2,5,15,46,45,5038,44,40,42} M4={21,54,27,22,26,29,23,57,58,47,32} M5={17,28,30,31,25,20} M6={18,24,19,16,55,43,60,49,52,53}	15	1,3
2	M1={7,8,3,15,60,14,12,11,10,75,67,77,65,68,48,49} M2={45,43,6,4,9,13,54,16,17,2,53} M3={1,51,47,46,5,52,20,63,74,76,72,61,,66,19} M4={62,26,37,27,28,25,41,37,57,79,78,71,59} M5={35,29,23,39,31,39,40,38,36,34} M6={44,33,55,21,22,48,70,73,69,64,24,50,80,56,42,58}	50	2
3	M1={20,18,6,87,8,98,91,16,84,71,22,97,59,85,74,10,80,70,54,23,67,69} M2={2,14,4,7,5,64,17,24,21,12,68,61,58,63,55,53,11} M3={9,1,3,15,19,25,13,56,62,65,60,57,95,73,83,96,77,72,75,86} M4={76,29,29,37,51,92,82,46,50,93,81,33,49,78,99,32,47} M5={28,34,36,41,43,44,48,52,35,45} M6={42,31,40,94,39,30,79,66,90,27,89,38,100,88}	33	2.5
4	M1={1,59,11,39,56,3,51,57,49,54,47,32} M2={7,4,9,5,3,33,34,41,42} M3={37,46,2,36,35,10,8,38,44,6,45,29} M4={58,48,16,14,15,12,27,50,23,43,13} M5={21,26,18,19,20,22,28} M6={55,24,17,30,31,52,25,60,53,40}	44	1.6
5	M1={62,46,48,47,49,6,38,40,79,76,60,77,63,52,53,2,13,65} M2={7,9,43,8,3,4,45,51,54,41,37} M3={1,10,80,5,14,12,59,71,68,64,11,61,42,56} M4={73,19,17,15,20,32,7070,57,18,67,58,78,16,27,34,33} M5={31,29,26,24,25,28,35,30} M6={21,69,23,22,75,44,55,50,66,74,36,39,72}	54	2

6	M1={2,5,3,13,4,91,57,66,59,97,71,88,80,51,46,74,98,53,86} M2={7,8,11,17,1,10,16,47,45,67,68} M3={77,6,14,15,9,12,78,94,100,90,87,48,72,73,75,79,50,61,58,56,69,70,62} M4={81,76,25,23,24,22,31,37,92,18,20,19,32,21,82} M5={40,26,28,30,3829,41,43,44,39,33,27,42} M6={52,36,34,35,60,63,65,55,64,54,93,49,83,96,84,89,95,85,99}	52	2.6
7	M1={1,3,10,2,57,39,49,42} M2={7,59,48,14,51,52,4,11,45} M3={41,33,6,37,5,38,35,13} M4={8,9,15,46,36,44,43,12,54} M5={17,21,30,20,47,58,53,56} M6={25,24,27,26,29} M7={18,55,60,28} M8={16,34,23,32,19,22,31,50,40}	4	2
8	M1={4,3,2,51,74,79,77,67,61,72} M2={1,73,68,10,19,18,16,11,13,54} M3={7,45,6,43,47,17,4} M4={46,9,55,15,8,48,5,20,12,14,65,60,64,63,71,58,49} M5={26,27,25,31,22,34,41,57,78} M6={38,30,33,36} M7={23,62,69,32,50,24,28,80,52,59,53} M8={21,35,40,39,42,29,37,56,66,76,70,75}	35	2.25
9	M1={98,91,97,57,4,13,1,20,19,2} M2={87,3,15,67,6,24,25,22,5,9,70,64,56,11,88,73,23} M3={53,68,14,16,7,18,63} M4={10,17,21,12,8,58,62,78,74,85,100,72} M5={30,75,93,71,96,90,41,86,37,32,80,79} M6={42,36,48,34,45,40,33,38} M7={76,49,92,94,95,82,84,77,29,47,89,27,69} M8={52,31,59,55,46,43,65,28,44,39,35,60,26,51,54,61,50,83,81,99,}	42	2.5
10	M1={59,5,43,45,31,52,11} M2={3,9,48,54,57,47,32,42} M3={1,4,6,7,34} M4={36,2,35,38,8,46,30,49,10,44} M5={15,17,14,12,27,25,13,22,23} M6={18,24,21,16,26} M7={29,60,55,58,39,53} M8={41,33,37,28,20,56,19,50,40,51}	15	1.7
11	M1={73,3,11,4,44,10,76,40,49,71} M2={2,5,8,7,69,62,37,57,41,14,38} M3={45,43,47,54,13,12,39} M4={1,9,6,60,75,70,80,61,78,59,68,65,58,56,55} M5={32,20,64,15,19,28,29,30,18,16,17,72,79} M6={23,34,31,24,36} M7={51,26,27,48,42,52,74,67,33} M8={21,35,22,25,77,66,46,63}	36	2
12	M1={50,93,87,80,46,97,2,9,58} M2={176,68,54,59,84,98,91,17,94,49,71,73,72} M3={4,8,16,7,1,13,12,61,55,64,47,51} M4={14,62,77,96,82,63,6,74,69,99,10}	39	2.225

	M5={18,25,23,37,22,19,21,90,89,35,85,86,75,24,81,88} M6={36,40,31,29,32,41} M7={30,44,28,45,48,38,33,43,27,70,56,95} M8={66,92,53,57,52,100,39,78,83,26,60}		

Appendix H

Structure of the Design of Experiments

StdOrder	RunOrder	PtType	Blocks	Number of Machines	Number of Product Types	Number of Orders	Min Obj Func	Avg Comp Time
2	1	1	1	6	4	80	50	1.1
8	2	1	1	8	4	80	35	2.1
6	3	1	1	6	7	100	52	1.2
4	4	1	1	6	7	60	44	1.78
7	5	1	1	8	4	60	4	1.1
10	6	1	1	8	7	60	15	1.95
3	7	1	1	6	4	100	33	2.43
5	8	1	1	6	7	80	54	2.5
11	9	1	1	8	7	80	36	2
1	10	1	1	6	4	60	15	1.4
12	11	1	1	8	7	100	39	2.25
9	12	1	1	8	4	100	42	2.5

Appendix I

Structure of the Design of Experiments -Multilevel Factorial Design

```
Factors:        3     Replicates:     1
Base runs:     12     Total runs:    12
Base blocks:    1     Total blocks:   1

Number of levels: 2. 2. 3
```

Design Table (randomized)

Run	Blk	A	B	C
1	1	1	1	2
2	1	2	1	2
3	1	1	2	3
4	1	1	2	1
5	1	2	1	1
6	1	2	2	1
7	1	1	1	3
8	1	1	2	2
9	1	2	2	2
10	1	1	1	1
11	1	2	2	3
12	1	2	1	3

Appendix J

Results of the Design of Experiments-Minimum Objective Function

General Linear Model: Min Obj Func

Factor	Type	Levels	Values
Number of Machines	fixed	2	6. 8
Number of Product Types	fixed	2	4. 7
Number of Orders	fixed	3	60. 80. 100

Analysis of Variance for Min Obj Func, using Adjusted SS for Tests

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Number of Machines	1	494,08	494,08	494,08	**	
Number of Product Types	1	310,08	310,08	310,08	**	
Number of Orders	2	1436,17	1436,17	718,08	**	
Number of Machines* Number of Product Types	1	154,08	154,08	154,08	**	
Number of Machines*Number of Orders	2	182,17	182,17	91,08	**	
Number of Product Types* Number of Orders	2	160,17	160,17	80,08	**	
Number of Machines* Number of Product Types* Number of Orders	2	50,17	50,17	25,08	**	
Error	0	*	*	*		
Total	11	2786,92				

** Denominator of F-test is zero or undefined.

Appendix K

Results of the Design of Experiments-Average Computation Time

General Linear Model: Avg Comp Time

Factor	Type	Levels	Values
Number of Machines	fixed	2	6. 8
Number of Product Types	fixed	2	4. 7
Number of Orders	fixed	3	60. 80. 100

Analysis of Variance for Avg Comp Time, using Adjusted SS for Tests

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Number of Machines	1	0,18501	0,18501	0,18501	**	
Number of Product Types	1	0,09187	0,09187	0,09187	**	
Number of Orders	2	0,60382	0,60382	0,30191	**	
Number of Machines* Number of Product Types	1	0,00021	0,00021	0,00021	**	
Number of Machines*Number of Orders	2	0,19532	0,19532	0,09766	**	
Number of Product Types* Number of Orders	2	1,25645	1,25645	0,62823	**	
Number of Machines* Number of Product Types* Number of Orders	2	0,85762	0,85762	0,42881	**	
Error	0	*	*	*		
Total	11	3,19029				

** Denominator of F-test is zero or undefined.

Appendix L

Results of the Regression Model- Minimum Ojective Function

Regression Analysis: Min Obj Func

```
The regression equation is
Min Obj Func = 17,2 - 6,42 Number of Machines + 3,39 Number of Product Types
               + 0,550 Number of Orders


Predictor                  Coef  SE Coef      T      P
Constant                  17,19    30,40   0,57  0,587
Number of Machines       -6,417    3,251  -1,97  0,084
Number of Product Types   3,389    2,167   1,56  0,157
Number of Orders         0,5500   0,1991   2,76  0,025


S = 11,2625   R-Sq = 63,6%   R-Sq(adj) = 49,9%


Analysis of Variance

Source          DF      SS     MS     F      P
Regression       3  1772,2  590,7  4,66  0,036
Residual Error   8  1014,7  126,8
Total           11  2786,9


Source                   DF  Seq SS
Number of Machines        1   494,1
Number of Product Types   1   310,1
Number of Orders          1   968,0
```