A TABU SEARCH ALGORITHM FOR ORDER ACCEPTANCE AND SCHEDULING PROBLEM

by

Bahriye Cesaret

A Thesis Submitted to the
Graduate School of Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Industrial Engineering

Koç University

July, 2010

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Bahriye Cesaret

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Assoc. Prof. Ceyda Oğuz (Advisor)

_____

Assist. Prof. F. Sibel Salman (Advisor)

_____

Assist. Prof. Onur Kaya

_____

Assist. Prof. Deniz Aksen

_____

Assist. Prof. Kerem Bülbül

Date: _____

to my family...

ABSTRACT

In this thesis, we consider a make-to-order production system, where limited production capacity and order delivery requirements necessitate selective acceptance among incoming orders. In the considered system, each accepted order that is completed and delivered to the customer before its quoted due date brings a revenue to the manufacturer. Late delivery of an order incurs tardiness costs and therefore causes a decrease in the revenue that is proportional to its tardiness. If an order cannot be completed before its deadline, then it brings no revenue. It is possible to reject an order and this incurs no additional penalty cost. Accepted orders should be scheduled after their corresponding release times on a single machine. A sequence-dependent setup time elapses between the processing of each order. Since revenue from an accepted order is a function of its completion time and processing of an order may delay the subsequent orders beyond their due dates, order acceptance and scheduling decisions should be taken jointly. In order to maximize the total revenue, the manufacturer has to determine which orders to accept and how to schedule them. For this NP-hard combinatorial optimization problem, we propose a new heuristic solution approach. Namely, we develop a tabu search algorithm which is supported with an effective probabilistic local search and diversification mechanism. We analyze the performance of the tabu search algorithm on an extensive set of test instances with up to 100 orders and compare it with two heuristics from the literature. In the comparison, we report optimality gaps which are calculated with respect to upper bounds generated from a mixed integer programming formulation. The results of our computational study show that the tabu search algorithm gives near optimal solutions that are significantly better compared to the solutions given by the two heuristics. Furthermore, the run time of the tabu search algorithm is very small, even for 100 orders. The success of the proposed heuristic largely depends on its capability to incorporate in its search the acceptance and scheduling decisions simultaneously, and to provide effective diversification mechanisms.

ÖZET

Bu tezde, sipariş teslim zamanlarının ve kısıtlı üretim kapasitesinin gelen siparişlerin seçilerek kabul edilmesini gerektirdiği, bir siparişe dayalı üretim sistemi ele alınmıştır. Ele alınan sistemde, kabul edilen ve teslim zamanından önce tamamlanıp, müşteriye teslim edilen her sipariş üreticiye kazanç sağlar. Geç teslim edilen siparişler teslim gecikmesi ile orantılı olarak kazançta bir düşüş yaratırken, termin zamanından önce tamamlanamayan işler kazanç getirmez. Herhangi bir siparişin reddedilmesi mümkündür ve hiçbir ek maliyet getirmez. Kabul edilen siparişler tek makine üzerinde, serbest bırakılma zamanlarından sonra olacak şekilde çizelgelenebilirler ve siparişler arasında sıraya bağlı hazırlık süreleri vardır. Kabul edilen siparişin kazancı, o siparişin tamamlanma zamanının bir fonksiyonu olduğu ve bir siparişin işlenmesi sonraki siparişlerin gecikmesine neden olabileceği için sipariş kabul etme ve çizelgeleme kararları birlikte ele alınmalıdır. Elde edilen kazancı enbüyüklemek için, üretici hangi siparişleri kabul edeceğini ve bu siparişleri hangi sırada işleyeceğini belirlemelidir. Bu NP-zor eniyileme problemi için, etkili olasılıksal yerel arama ve çeşitlendirme mekanizmaları ile desteklenmiş bir tabu arama algoritması önerdik. Önerilen algoritmanın performansı, çeşitli parametre değerleri ile rassal olarak oluşturulmuş, çok sayıda örnek problem ile çözülerek incelenmiştir. Ayrıca bu algoritma, literatürde var olan iki ayrı sezgisel yöntem ile kıyaslanmıştır. Karşılaştırmalarda performans ölçütü olarak çözümlerin kazançlarının en yüksek kazançtan ne kadar uzak olduğu ele alınmıştır. Karışık tamsayılı programlama kullanılarak en yüksek kazanç için bir üst sınır elde edilmiş ve bu üst sınıra dayanarak hesaplanmış ortalama optimalite aralığı verilmiştir. Yaptığımız hesaplamalı deneylerin sonuçlarına göre, tabu arama algortiması amaç fonksiyonu açısından test edilen örneklerde kıyasladığımız diğer iki sezgisel yöntemden çok daha iyi sonuçlar vermiştir. Buna karşın, tabu arama algortimasının çalışma zamanı 100 adet sipariş verildiği örnekler için bile oldukça kısadır. Önerilen sezgisel yöntemin başarısı, büyük ölçüde, arama sürecine sipariş kabul etme ve çizelgeleme kararlarını birlikte dahil edebilmesine ve etkili çeşitlendirme mekanizmaları kullanmasına bağlıdır.

ACKNOWLEDGMENTS

First I would like to thank my supervisors Assoc. Prof. Ceyda Oğuz and Assist. Prof. F. Sibel Salman for their hard work and guidance throughout this entire thesis process and for believing in my abilities. They provided a great source of inspiration and valuable suggestions during my thesis studies.

I would like to thank Assist. Prof. Onur Kaya, Assist. Prof. Deniz Aksen and Assist. Prof. Kerem Bülbül for taking part in my thesis committee, for critical reading of this thesis and for their valuable suggestions and comments.

I would like to acknowledge financial support from TUBITAK during my master thesis.

I am grateful to Tarık Kobalas and Bekir Yenilmez for their help in computer programming issues which made my computational studies smoother.

I would like to express my appreciation to my friends, Burcu Inci, Eylem Yalçınkaya, Nalan Lom, Serkan Kara, Ibrahim Aşkar, Alper Şahin, Onur Çoban, Bahtiyar Yıldırım, Aydın Akay, Habip Tiryaki and again Tarık Kobalas for their support and unforgettable friendships.

I would like to thank my friends Yeliz Akça and Selin Özdinç for their valuable friendships during my master study at Koç University.

And special thanks to Güliz Akkaymak and Hilal Şen for their warm, enjoyable, pleasant friendships and thanks again for them not merely being homemates.

Last but not the least, I thank my parents Selbiye and Hikmet, my sister Sibel and her husband Coşkun, for their never-ending love and support in all my efforts. To them I dedicate this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

ACO	Ant Colony Optimization
ATC	Apparent Tardiness Costs
ATCS	Apparent Tardiness Costs with Setups
DPSO	Discrete Particle Swarm Optimization
DSKP	Dynamic Stochastic Knapsack Problem
EDD	Earliest Due Date
GA	Genetic Algorithm
GRASP	Greedy Randomized Adaptive Search Procedure
ISFAN	Iterative Sequence First - Accept Next Algorithm
LP	Linear Programming
LR	Lagrangean Relaxation
m-ATCS	Modified Apparent Tardiness Costs with Setups Heuristic
MILP	Mixed Integer Linear Programming
MTO	Make-to-order
OAS	Order Acceptance and Scheduling
PCTSP	Prize Collecting Traveling Salesman Problem
$RLR1$	Revenue-Load-Ratio1
$RLR2$	Revenue-Load-Ratio2
SA	Simulated Annealing
TS	Tabu Search
TSP	Travelling Salesman Problem
UB	Best Upper Bound
UB_{LPVI}	Upper Bound obtained by LP Relaxation of MILP with Valid Inequalities
UB_{MILP}	MILP Upper Bound at Termination

Chapter 1

INTRODUCTION

In make-to-order (MTO) production systems, production decisions are initiated by customer orders. Hence, inventory holding and obsolescence costs can be kept at minimum. Moreover, the MTO approach provides the flexibility of either offering a greater variety of products or making products that are unique to the customers. On the other hand, fast response to production orders and the pressure to meet customer demand on time are important issues in MTO production systems. When customers called tight due dates and cannot tolerate late deliveries, it becomes critical for the MTO manufacturer to selectively accept and schedule customer orders.

In a make-to-order production system, simultaneous order acceptance and scheduling decisions arise when limited production capacity and tight delivery requirements necessitate selecting orders to maximize total revenues. Accepting orders without considering their impact on capacity may create an overload at certain time periods, that may in turn delay completion of other orders and thus incur penalty costs or a reduction in revenues from other orders. In some cases, this may also cause customer dissatisfaction that leads to loss of revenue in the long run, especially in competitive industries. These circumstances necessitate the acceptance decisions to be made not on an individual order-basis, but by considering the set of orders to be processed and their scheduling at the same time. To be able to use the production capacity more efficiently and maintain customer satisfaction, the manufacturer should determine which orders to accept and how to schedule them to maximize the total revenue.

In some manufacturing settings, such as a custom made furniture shop or printing and painting operations, having sequence-dependent setup times is an important factor in scheduling. In this thesis, we consider an order acceptance and scheduling problem in a single machine environment where the orders are processed without preemption and a sequence-

dependent setup time is incurred between processing of every two consecutive orders. When a customer places an order, a due date and a deadline are quoted. A set of orders is available with the following data: release dates, processing times, sequence-dependent setup times, due dates, deadlines, revenues and weights. Knowing this information in advance at the beginning of planning horizon, the manufacturer must select a subset of incoming orders and schedule these accepted orders to generate maximum total revenue. The revenue from an order depends on the following situations. If an order is accepted and completed before its due date, the pre-specified revenue is generated. If it is completed after its due date but before the deadline, it incurs a reduced revenue due to tardiness. On the other hand, an order completed after the deadline results in zero revenue. Hence, rejecting such orders, by considering the scheduling aspect a priori, leads to better utilization of the capacity. In the rest of this thesis, we refer to this problem as the Order Acceptance and Scheduling (OAS) problem, as in [80].

The OAS problem arises in different MTO production systems and was first introduced in [80]. A motivating example of a printing and laminating company that works on MTO principle and faces the OAS problem is given in that study. The production process of this company includes printing, lamination, slitting and packaging operations. The printing process requires different setup times depending on the orders. The required setup time of an order also changes depending on the predecessor of the order. Because fixing raw materials and apparatus related to the order differs from one order to another the setup time depends on the order as well as the predecessor of this order. Since, the company cannot handle simultaneous acceptance/rejection and scheduling decisions efficiently, it faces difficulties in meeting the customer due dates.

The OAS problem also arises in logistics systems which require customized services. In this setting, the distance between customers can be viewed as sequence-dependent setup time and each customer may require the service within a time window. The customer prefers to be served until a specified date. In case this is not possible, the customer also accepts to be served until a final quoted date, i.e. a deadline. However, after the preferred date, customer satisfaction begins to decrease and therefore he only accepts to pay for a reduced amount. Furthermore, he refuses to pay after the final quoted date. The service provider has to select which customers to serve based on the revenue he will get and the available

capacity. The service provider completes a tour by sequencing the selected customer visits. Hereby, the service provider also faces the OAS problem.

The contribution of this study is two-folds. Firstly, we develop a tabu search (TS) heuristic that performs significantly better in terms of both solution quality and run time compared to existing heuristics in the literature. The novel characteristics of this solution approach can be summarized as follows. The TS algorithm uses a representation that holds information about both the acceptance/rejection status and the processing sequence of the orders, which enables the algorithm to make the acceptance and scheduling decisions simultaneously rather than sequentially. This, together with a new neighborhood definition and the use of memory, facilitates searching the solution space much more effectively. TS also uses a local search procedure intelligently by compound moves. One compound move includes iterative drop, add and insertion operators utilizing problem specific knowledge. The local search procedure also serves to diversify probabilistically. Secondly, we provide a wide range of instances by changing the values of problem parameters such as due date range and tardiness factor. The proposed TS algorithm shows consistently superior performance over different types of instances that could be encountered in practice. Therefore, TS can also be suggested as a robust algorithm for the OAS problem.

The organization of the thesis is as follows. In Chapter 2, we review studies related to the OAS problem and point out their differences from the OAS problem. In this chapter, we also review the related studies which implemented TS as a solution method for their problems. In Chapter 3, we define the OAS problem formally and give some insights on computational complexity. We also provide the mixed integer linear programming (MILP) formulation of the problem which was developed in [80] for the sake of completeness. We describe the proposed heuristic algorithm in Chapter 4. We give the details of data generation and results of computational studies in Chapter 5. In this chapter, we also discuss the difficulty of the test instances, analyze the upper bounds and the number of rejected and tardy orders. Finally in Chapter 6, we give concluding remarks and future research directions.

Chapter 2

LITERATURE SURVEY

In the scheduling literature, researchers mainly focus on constructing a schedule for a given set of orders to optimize a particular criterion. These types of scheduling problems can be called pure sequencing problems, since only the sequencing of given orders is considered in the problems. On the other hand, when a subset among incoming orders should be selected and the orders should be sequenced, we can classify these problems as selection and sequencing problems. In this chapter, we examine both pure sequencing problems and selection and sequencing problems whose objectives and settings are close to ours.

2.1 Pure Sequencing Problems

The usual approach in scheduling literature is to schedule n orders in a specific machine environment while trying to optimize some established optimality criterion. In pure sequencing problems, the decision of accepting or rejecting an order is not conceived, that is, all orders must be accepted and scheduled on some machines. Widely studied optimality criteria of pure sequencing problems include minimizing the makespan, the total completion times, the number of early/tardy orders, the setup times/setup costs, the maximum lateness, the total earliness and the total tardiness /total weighted tardiness or sometimes maximizing profits. Some of the most widely studied criteria are related to order tardiness.

In today's competitive business conditions, meeting customer due dates, namely on time delivery, is designated as the most important requirement for retaining customers [**?**]. Late delivery of the orders leads to dissatisfaction of the customer and therefore may cause a reduced revenue and furthermore may result in loss of the customer for the manufacturer in the long term. These are among important reasons why tardiness based objectives are one of the most studied cases. On the other hand, finding optimal solutions for tardiness based objectives are known to be very difficult [77].

Two important and very closely related subtypes of pure sequencing problems addressing

the tardiness issue are total tardiness and total weighted tardiness problems which are widely studied in the literature. These problems are mostly studied in static order arrival cases. In static order arrival case, all the orders are available at time zero, may either be released at time zero or at a known release date for processing. Moreover, all order parameters are attainable in advance for the decision maker at the beginning of the planning horizon. Researchers also discuss dynamic order arrival cases where the orders arrive randomly over time and the parameters of the orders are not attainable before the arrival. Static order arrival cases are also referred to as off-line scheduling whereas dynamic order arrival cases are referred to as on-line scheduling in the literature. Since our study falls into the static order arrival case, we do not cover the dynamic order arrival cases for pure sequencing problems.

2.1.1 Total Tardiness and Total Weighted Tardiness Problems

We can simply state the total tardiness problem as follows: n orders, each with a given processing time and a due date, should be processed without preemption. If an order is completed after its due date, it becomes tardy and the amount by which the completion time exceeds the due date is its tardiness. The objective is to determine a schedule that minimizes the total tardiness of the orders. In terms of the tardiness criterion, the early completion of the orders is neither rewarded nor penalized; whereas, the late completion of the orders brings an associated delay penalty. In the total weighted tardiness problem, the only difference is that there are positive weights associated with the tardiness of the orders and accordingly the objective is to determine the optimal schedule that minimizes the total weighted tardiness. The weights act as priority indicators for the orders. The higher the weights the more important the orders, implying that the manufacturer does not want these orders to be late compared to the others.

Sen et al. [92] give a complete survey about weighted and unweighted tardiness scheduling problems. They survey multi-machine environments as well as single machine environments. Most of the techniques in the literature which aim to solve the scheduling problems are focused on single machine environments. Because in practice, all the machine environments can be reduced to a single machine environment by assuming that there is a bottleneck machine in the production system. Since this bottleneck machine will restrict

the production system to its maximum capacity, scheduling of the bottleneck machine in multiple machine shops is equivalent to solving the single machine case. Once the basic single machine case is studied, it may be possible to extend the results to other machine environments. In this study, we also consider the single machine case. Therefore, the type of machine environment we review in this survey for pure sequencing problems is restricted to a single machine environment.

McNaughton [74] completed one of the earliest studies addressing the tardiness criterion in 1959. He gives a formulation for deferral cost associated with scheduling problems [74]. Shild and Fredman [94] generalized Mc Naughton's results and showed how to solve this problem to optimality in certain special cases which are quite restrictive. Lawler introduced a dynamic programming formulation to solve the same problem to optimality for small number of orders [64]. However, the total tardiness problem as stated above was first defined in Emmons' study [36] in the late sixties. In fact, the total tardiness problem defined by Emmons [36] is a special case of Mc Naughton's more general scheduling problem [74]. In his study [36], Emmons points out some relationships among processing times and due dates. This approach has also been adopted by other researchers, however Emmons is the first who utilizes these relationships and can reduce the size of the problems considerably. Later on, Rinnooy Kan et al. [54] and Fisher [39] make some slight extensions to Emmons' results [36]. Baker and Schrage [11] propose a technique which combines Emmons' results [36] with a dynamic programming scheme. This combined technique has a superior performance to Rinnooy Kan et al.'s technique [54]. However, Baker and Schrage's [11] method is dominated in terms of computation time by Lawler's method [66] which is also a dynamic programming method.

All of the previous studies on this problem until Lawler [65] were focused on designing fast enumerative algorithms to solve the problem to optimality. Lawler is the first researcher who investigated the complexity of the problem. In his study [65], although Lawler proves that the total weighted tardiness problem is NP-hard in the strong sense, he leaves the complexity of the total tardiness problem unresolved. In 1990, Du and Leung proved that single machine total tardiness problem when preemption is not allowed is NP-hard in the ordinary sense [34]. This proof is also extended to the preemptive case based on McNaughton's study [74] in which he proves that the tardiness of the problem is not decreased when preemption

is allowed. Although the classical single machine total tardiness problem is NP-hard, in some cases by utilizing some particular set of settings suggested by Koulamas [60] some polynomially solvable problems can be obtained. More recently, Koulamas [59] surveys the related literature for the total tardiness problems under different machine environments and provides an integrated framework for the problem.

Single Machine Total Weighted Tardiness Problems

Most of the studies related to single machine total tardiness problems have been extended to those with weighted tardiness [92]. An excellent survey of papers on the weighted tardiness problem until 1990 can be found in Abdul-Razaq and van Wassenhove [1]. Sen et al. [92] provide a more recent review of studies on both weighted and unweighted tardiness problems. Only some of the noteworthy studies among those and a few more recent ones which are closely related to our study, are reviewed below. All of the papers below studies static, single stage and single machine total weighted tardiness problems.

Lawler [65] considers the total weighted tardiness problem as well as the total tardiness problem. He gives a pseudo-polynomial time algorithm to solve the unweighted tardiness case to optimality, which also provides a basis for a fully polynomial time approximation scheme for the problem. Rinnooy Kan et al. [54] extend the Emmons' results [36] to the weighted tardiness case. Potts and van Wassenhove [84] develop a branch and bound algorithm for the problem. They introduce a new relaxation of the problem and obtain lower bounds using a Lagrangian Relaxation (LR) approach in which they use a multiplier adjustment method for updating the Lagrange multipliers instead of the well known subgradient optimization technique. This replacement leads to an extremely fast bound calculation. By combining this LR approach and the branch and bound algorithm, they find optimal solutions to problems with up to 50-orders. Bigras et al. [62] and Tanaka et al. [99] develop a time-indexed formulation for the single machine total weighted tardiness problem. Bigras et al. [62] suggest a temporal decomposition for the column generation algorithm. They also benefit from branching strategies and dominance rules for solving the problem to optimality. The resulting branch and bound algorithm performs quite well. Tanaka et al. [99] present a Successive Sublimation Dynamic Programming method by which they were able to solve instances with up to 300 jobs for the same problem. This algorithm outperforms the

existing algorithms proposed for the single machine total weighted tardiness problem and the single machine total weighted earliness-tardiness problem without machine idle time.

Recently Pessoa et al. [83] focus on both single and identical parallel machine environments of total weighted tardiness problem. They have suggested an arc-time-indexed formulation which is shown to be stronger than the time-indexed formulation and describe techniques that can solve instances with up to 100 jobs for the single machine case.

Since exact algorithms such as those studied by Potts and van Wassenhove [84] assure optimality but are computationally inefficient when the number of orders is large, heuristic methods and dispatching rules gained much popularity. Scheduling heuristics can provide good quality solutions in a reasonable time and can be broadly classified as constructive and improvement heuristics. Usually constructive heuristics obtain a feasible solution quickly but the solution quality is inferior; on the other hand, improvement heuristics yield better solutions at the expense of increased computational time.

Vepsalainen and Morton [104] propose an efficient constructive heuristic which they call the apparent tardiness costs (ATC) for the single machine total weighted tardiness problem with order-specific due dates and delay penalties. The ATC rule uses a dynamic, time-dependent dispatching rule and a look ahead parameter whose value is interrelated with the number of competing critical orders. Holsenback et al. [48] develop an improvement heuristic for the problem. Potts and van Wassenhove [85] propose several heuristics and dispatching rules. Subsequently, Crauwels [31] extends this work [85] by giving a thorough comparison of single and multi-start versions of different heuristics. In this study [31], Crauwels also defines a new solution representation which includes a binary encoding scheme and gives a heuristic to decode these binary solutions into actual sequences. Additionally, a Greedy Randomized Adaptive Search Procedure (GRASP) algorithm is also proposed for the problem by Gupta and Smith [47].

The iterated dynasearch algorithm introduced by Congram et al. [30] has been recognized to perform very well in the literature. This algorithm is a local search technique which allows a series of moves at each iteration unlike traditional only-a-single-move-allowing techniques. The most important feature of this algorithm is the capability to search the exponential size neighborhood in polynomial time. Later, Grosso et al. [44] improve the iterated dynasearch algorithm by adopting the generalized pairwise interchange operators

instead of a series of independent interchange moves in [30]. Angel and Bampis further extend [30] by developing a multi-start local search algorithm.

In a very recent study, for the same problem Wang and Tang [112] develop a population based variable neighborhood search technique which combines path-relinking, variable depth search and tabu search to improve the search intensification and uses a population of solutions to improve the search diversification. In another recent study, Altunc and Keha [7] proposed an interval-indexed formulation which can be solved much faster than the reduced time-interval formulation and they reduce size of an integer programming model by fixing some binary variables at zero.

Setup Times

In practical applications, there exists some unavoidable and non-negligible amount of time before beginning to process an order or between processing of two orders which is called setup time. A setup time is roughly, an amount of time required for an order and/or a machine to be ready for processing. The importance of considering setup times has been underlined in many studies. One of these studies belongs to Panwalkar et al. [81]. According to this study, nearly 75% of the managers indicated they face sequence-dependent setup times at least in some operations in their production system.

In the literature, the single machine total weighted tardiness problem is also studied with setup times which are either dependent or independent of the sequence. The sequence independent setup time is only affected by the order processed, however the sequence dependent setup time is affected both by the order processed and the immediate predecessor of this order. Most researchers have assumed that setup times are either negligible or could be added to the processing times which is the sequence independent case.

Potts and van Wassenhove [84] develop a branch-and-bound algorithm for the single machine total weighted tardiness problem with sequence independent setup times. Aktürk and Özdemir [3, 4], Chu [25] and Vepsalainen and Morton [104] also adopt the sequence independent setup time assumption in their studies. However, in our study we consider the sequence-dependent setup case, therefore we examine this type of problems further in this chapter.

Sequence-dependent Setup Times

In some cases, depending on the production characteristics, the sequence-dependent

setup times are indispensable. Making a sequence independent setup time assumption may cause good optimization models to turn out to be very poor models when applied to sequence dependent setup time problems [72]. Additionally, Wortman [108] emphasizes the entailment of the sequence-dependent setup consideration while trying to manage a manufacturing capacity effectively. Since the total weighted tardiness problem is known to be NP-hard and sequence-dependent setup times act as a further complicating property, the single machine total weighted tardiness problem with sequence-dependent setup times is also NP-hard. Rubin and Ragatz [90] examine how the total tardiness scheduling on a single machine gets more difficult when the sequence-dependent setups are added.

As usual, the first attempt to solve the single machine total weighted tardiness problem with sequence-dependent setup times is also a branch and bound algorithm developed by Rinnooy Kan et al. [54]. Lee et al. [68] proposed the best known constructive heuristic for the single machine total weighted tardiness problem in the presence of sequence-dependent setup times. They suggest a priority rule by adapting the idea of Vepsalainen and Morton [104] for the sequence-dependent case which then they called ATC with setups (ATCS). ATCS is a very time efficient dispatching rule that produces high quality solutions and is designed for problems having the tardiness objective. This rule calculates a priority index among unscheduled orders which takes into account both the slack and setup times to select the next order to be sequenced and it then performs a local search to increase the quality of the solution. They also present a simulated annealing (SA) heuristic which selects a neighborhood solution randomly with a high acceptance probability. Cicirello [29] develops five different improvement-type heuristics: the limited discrepancy search (LDS), the heuristic-biased stochastic sampling (HBSS), the value-biased stochastic sampling (VBSS), the value-biased stochastic sampling seeded hillclimber (VBSS-HS), and a SA approach to schedule orders in a sequence-dependent environment.

Many metaheuristics are also proposed for the problem. Armentano and Mazzini [10] develop a genetic algorithm (GA) to minimize total tardiness on one machine where the sequence-dependent setup times exist and compared the results of this GA with MILP formulation solved by CPLEX software for small-sized problems and with the ATCS results for large-sized problems. According to their results, GA outperforms ATCS heuristic for all test problems. A memetic algorithm which reduces the neighborhood and structures

the population hierarchically is proposed by Franca et al. [40] for the single machine total weighted tardiness problem including setups. Tan et al. [98] present a comparison of the methods of branch-and-bound, genetic search, random-start pairwise interchange, and SA, for the same problem. Lin and Ying [71] develop a SA approach with an insertion search, a GA approach with a greedy local search, TS approach with an insertion tabu list to solve the problem whose best result are compared with Cicirello's best known ones [29]. The authors report that all three algorithms improved the previous best known results. Gagne et al. [41], Agniholfi and Paolucci [8] and Liao and Juan [70] use ant colony optimization (ACO) methods in their studies. Gagne et al. [41] describe an efficient ACO algorithm which uses a look-ahead aspect in selection of the upcoming order for the single machine total tardiness problem with sequence-dependent setup times. Additionally, a discrete particle swarm optimization (DPSO) algorithm with excellent results is also presented by Anghinolfi and Paolucci [9]. Bozejko [18] proposes a parallel path relinking method and Valante and Alves [103] present a beam search for the same problem. Cicirello [28] develops a GA algorithm for the problem applying a new variation of the well-known order crossover (OX) which they call nonwrapping order crossover (NWOX) operator. The proposed GA algorithm performs well for the problem by providing an improvement in the best known results for the benchmarks proposed by Cicirello given in [27]. And finally, in their study Tasgetiren et al. [100] present a discrete differential evolution algorithm for the problem. They introduce newly designed speed up methods by inserting orders into the algorithm more easily which is their novel contribution to the literature. They verify the effectiveness of their algorithm by comparing their results with [8] and [9] and improve the best known solutions known in the literature so far for the benchmarks they used. An excellent survey of scheduling with a setup consideration can be found in Allahverdi et al. [5, 6] and in Yang and Liao [115].

Arbitrary Release Dates

Deterministic scheduling with release dates and due dates has received considerable attention throughout the years. Although finding studies with unequal release dates associated with other optimality criteria in different machine shops is easier in the literature, limited studies exist concerning the single machine total weighted tardiness problem with arbitrary release dates. Most of the studies which have been done on the single machine total weighted tardiness problem consider equal release dates among all orders such as the study

of Koulamas [59]. In another study [61], Koulamas investigates the effect of introducing nonzero release dates.

One of the earliest studies assuming unequal release dates is given by Rinnooy Kan et al. [53]. He shows that the total tardiness problem with unequal release dates is NP-hard. Lawler [65] proves that the total weighted tardiness problem is strongly NP-hard, hence the unequal release dates problem with total weighted tardiness objective is also strongly NP-hard.

There has been no exact algorithm for the single machine total weighted tardiness problem with release dates until Aktürk and Özdemir's study [3]. They develop a branch and bound algorithm for the problem. Furthermore, they present a set of dominance properties for the problem which can be utilized in any exact approach. This is the first study [3] in the literature which considers the weighted tardiness and unequal release date problems simultaneously on a single machine. They solved up to 150-orders sized problems in this study.

Sequence-dependent Setup Times and Arbitrary Release Dates

Finally, we want to focus on the papers which aim to solve the problem of minimizing the single machine total weighted tardiness problem considering unequal release dates and sequence-dependent setup times simultaneously. This problem is still a special case of our problem, since we consider order acceptance decisions additionally.

Sun et al. [97] considered the problem in the late nineties with a slightly different objective function which minimizes the total weighted squared tardiness. A TS approach, a SA approach, the earliest due date (EDD) and ATCS dispatching rules, and a fourway swap local search methods are compared with a LR approach. The proposed LR method outperforms all of the other methods except SA, and dominates the SA approach in terms of computation time and has a comparable solution quality with SA. Chang et al. [20] propose a heuristic algorithm with the complexity of $O(n^3)$. To see how effective the proposed method works, they also formulated a mathematical programming model with logical constraints. According to their computational results, the proposed heuristic method can handle this problem efficiently.

2.2 Selection and Sequencing Problems

As we stated in Section 2.1, traditional scheduling problems assume all orders must be accepted and therefore do not address the accept/reject decisions for the orders. However, in real-world scenarios, this may not hold. The manufacturer or scheduler might have the option of rejecting some orders, due to limited resources. An order accept/reject decision implies selecting a subset of the available orders when processing all orders exceeds current capacity. The manufacturer has to first decide which orders to select for processing and then decide in which sequence to process these selected orders which can be referred to as a selection and sequencing problem. Here, the order selection aspect provides a clear distinction from the pure sequencing problems found in the traditional scheduling literature.

We can examine this group under two subgroups: scheduling with rejections and scheduling with order acceptance. Since little prior research exists on these problems, we do not restrict these papers as we do for pure sequencing problems. We examine them under different objectives as well as different settings.

2.2.1 Scheduling with Rejections

In this class of problems, there is a set of independent orders that are generally characterized by their processing times and rejection penalties. The manufacturer has the option of rejecting some of the orders, in which case it must pay for each rejected order. An order can be either rejected or scheduled on one of the processors. The manufacturer pays the corresponding rejection penalties in the former case and pays the makespan of the constructed schedule in the latter case. The idea of scheduling with rejection is not so old and little prior research exists related to this type of problems. The problem is studied both in on-line and off-line settings. In an on-line setting, when the orders arrive, the information about the orders becomes known and the manufacturer has to decide whether to accept or reject the current order without any knowledge of the future system. However, in the off-line version, order arrivals are static and all the order data are known a priori. The manufacturer has to decide which orders to reject at the beginning of the planning horizon.

Bartal et al. [15] are the first researchers who considered rejection penalties in machine scheduling. They consider both on-line and off-line versions of the problem on identical parallel machines where preemption is not allowed. For the off-line problem, they suggest a

fully polynomial-time approximation scheme for fixed m and a polynomial-time approximation scheme for arbitrary m, where m represents the number of identical parallel machines. After that, the machine scheduling with rejection gained an increasing attention. Assuming preemption is allowed, which means orders may be arbitrarily interrupted and resumed later, Seiden [91] gives a better algorithm than Bartal et al. [15] for the on-line version of the problem. Hoogeven et al. [49] study the off-line version of the problem again in a multi-processor environment as in [15, 91]. They consider the problem where preemption is allowed.

Lu et al. [73] extend the problem with release dates in an unbounded parallel batch machine environment. In this study, the processing time of the batch is determined by the order which has the longest processing time in that batch. They show that minimizing the makespan on a single machine with release dates and rejections is NP-hard and develop a pseudo-polynomial-time algorithm and a fully polynomial-time approximation scheme for this problem. In all four studies above, the objective is to minimize the makespan of the accepted orders plus the total rejection penalties.

Engels et al. [37] address only the off-line setting in a single machine environment and, differing from previous studies mentioned above, they focus on minimizing the weighted sum of completion times plus the penalties of the rejected orders rather than the sum of the makespan plus the penalties of the rejected orders. They illustrate several techniques which show how to reduce a scheduling problem with rejection to a problem without rejection. The on-line version of the problem in [37] is studied by Epstein et al. [38], in which each order has a unit processing time. The objective function of both studies is to minimize the sum of the total completion time of the scheduled orders and the sum of the penalties of the rejected orders.

Sengupta [93] considers the objective function of minimizing the maximum lateness/tardiness of the scheduled orders plus the total rejection penalty of the rejected orders. They show that without considering the rejection decision the problems are polynomially solvable by using the EDD rule. However, with inclusion of the rejection option the problem turns out to be NP-complete. For the problem, they propose dynamic programming based pseudo-polynomial time algorithms and also develop a fully polynomial time approximation scheme.

Dosa and He [33] add the machine costs to the scheduling problem with rejection. In

this problem setting, initially there is no machine and for each newly purchased machine there is associated machine cost that has to be paid. When a new order arrives, there are three cases that can occur. First, the order can be rejected by paying its penalty, second the order can be processed non-preemptively on an existing machine by contributing to the machine load and third, the order can be processed on a newly purchased machine. Here, the aim is to minimize the sum of the makespan, the machine purchasing cost, and the total rejection penalty.

Cheng and Sun [23], consider several single machine off-line scheduling problems with rejection and deteoriating orders under the objectives of minimizing the makespan, the total weighted completion time and the maximum lateness/tardiness plus the total penalty of the rejected orders. In this study, they prove that all these problems are NP-hard, and to solve them they develop dynamic programming based algorithms. Zhang et al. [116] focused on minimizing the sum of the makespan of the accepted orders and the total rejection penalty of the rejected orders on a single machine environment where release dates and rejection decision exist. They prove that the problem is NP-hard in the ordinary sense and propose two pseudo-polynomial-time algorithms, a 2-approximation algorithm and a fully polynomial-time approximation scheme for the problem.

2.2.2 Order Acceptance and Scheduling Problems

In the second type of selection and sequencing problems, which we can call order acceptance and scheduling problems, only accepted items contribute to the objective function, whereas in scheduling with rejections types both accepted and rejected orders are considered in objective function. Although there exists no penalty for the rejected orders, the set of accepted orders must be scheduled at some cost in scheduling with order acceptance problems. The order acceptance and scheduling problem is studied with deterministic order arrivals in most of the cases.

Deterministic Order Arrival Cases

Order acceptance with lateness penalty, which is known to be NP-hard, is studied by Slotnick and Morton [96], Ghosh [42], and later Lewis and Slotnick [69]. The problem with the tardiness penalty instead of the lateness penalty is studied by Slotnick and Morton [95]

and Rom and Slotnick [88]. Slotnick and Morton [96], address static order arrivals with given deterministic processing times, due dates, profits and a customer weight. Ghosh [42] reconsiders the order selection problem introduced by Slotnick and Morton [96] and proves that the order acceptance problem with a lateness penalty is NP-hard in the ordinary sense. Lewis and Slotnick [69] focus on a multi-period deterministic version of the problem which is studied by Slotnick and Morton [96] in a one-period setting. The studies of Slotnick and Morton [95] and Rom and Slotnick [88] are extensions of Slotnick and Morton's work [96] with tardiness instead of lateness as the time related penalty. All of these works study an order acceptance and scheduling problem in a single machine environment without preemption. And in all these studies, the objective is to maximize the total profit same with our study, which is defined as sum of revenues minus total weighted tardiness.

Charnsirisakskul et al. [21] address the order selection and scheduling decisions in a preemptive single machine environment. In this study demand is deterministic, setup costs are negligible and customer orders differ in their arrival times. Completing orders after due-dates generate a penalty cost, while producing before due dates generates a holding cost. Charnsirisakskul et al. [22], add a pricing decision to the problem. In both studies the objective is to maximize the manufacturer's profit defined as revenue minus manufacturing, holding and tardiness costs.

Apart from these studies, Gupta [46] considers simultaneous selection and sequencing of projects among a certain number of projects in such a way that the total net present value is maximized. Chuzhoy et al. [26] consider the order interval selection problem both in single and multiple machine environments. They try to schedule as many orders as possible between their release dates and deadlines without preemption. Yang and Geunes [114] address a single machine scheduling problem with acceptance decisions, tardiness costs and controllable processing times. A recent study, again with deterministic order arrivals, is given by Talla Nobibbon et al. [78]. They study the order acceptance and scheduling problem in a single machine environment where the orders are characterized by known processing times, delivery dates, revenues and weight. Finally, Oğuz et al. [80] study a generalization of Talla Nobibbon et al.'s problem [78] including release dates, deadlines and sequence-dependent setup times, which represents the same problem settings and the same objective function as our study. They introduce a mixed integer linear programming

(MILP) formulation for the problem and propose a SA based heuristic algorithm to solve the problem. They also modify the ATSC rule for their problem and call it modified ATCS (m-ATCS). According to their results, they solved problems up to 50 orders with their proposed heuristic algorithm and can solve much larger problems using the m-ATCS constructive rule.

Dynamic Order Arrival Cases

The dynamic order arrival case of the order acceptance and scheduling problem has also been studied under different settings and with different objectives. Some of the studies considering dynamic order arrival cases are given below.

Wester et al. [106] study order acceptance and scheduling strategies in a single machine production environment where customer orders arrive randomly. In this study there are setup times due to batching of orders with the same type and due dates but a deadline is not specified for an order, thus tardy orders are rejected. In Akkan et al. [2] each order comes with a required due date and an earliest release time; preemption is not allowed. The objective function is to minimize the present-value of the cost of rejecting orders and the inventory holding cost due to early completion. De et al. [32] examine a single machine scheduling problem without preemption and with random processing times and deadlines. Ten Kate [101], concentrates on order acceptance in a single resource case with deterministic processing times. Balakrishnan et al. [12] aim to maximize overall profit of the firms by selectively rejecting some of the orders for lower products. Wu and Chen [109, 110] propose a model for justifying the acceptance of rush orders. They focus on the objective of minimizing the cost instead of profit maximizing. In Guerrero and Kern's study [45], there is an assemble to order environment and the quantity of the product needed, the earliest due date and the latest due date of each order is specifies by the customer.

Kingsman [56] studies a capacity oriented order acceptance problem with stochastic order arrivals in a job shop environment. Each order quotes either a price or a delivery lead time or both and the objective is to process the orders so as to meet the promised delivery dates. Roundy et al. [89] solve an order acceptance problem in a job shop environment rather than a popular single machine problem. The incoming orders, if accepted, are inserted into the current schedule by forming production batches. The objective is to minimize setup and holding costs, due to the batch-sizing decision. Raaymakers et al. [86, 87] and Ivanescu et

al. [50, 51], study batch manufacturing where Raaymakers et al. [86, 87] with deterministic processing times and Ivanescu et al. [50, 51] with stochastic processing times. Ebben et al. [35] examine some approaches to combine the order acceptance and the resource capacity loading in a job shop environment with stochastic processing times.

Kleywegt and Papastavrou [57, 58] address the case that customer orders arrive dynamically over time. Papastavrou et al. [82] study the dynamic stochastic version of the knapsack problem (DSKP). In this problem setting, the items arrive over time, the rewards and/or sizes are unknown before arrival and they do not consider holding costs while they are studying the problem in a finite horizon. Kleywegt and Papastavrou [57] study the continuous-time version of the DSKP with holding costs. They consider both the finite and infinite horizon cases and in this study all items have the same size; in other words, all demands require the same amount of resource. In [58], Kleywegt and Papastavrou extend the results to the case where demands require random amounts of resources. A holding cost that depends on the amount of resources allocated is incurred until the process is stopped. The aim is to decide which demands to be accepted in order to maximize expected profit.

Acceptance Decisions

We can also classify the literature based on how acceptance decisions are made. In Roundy et al. [89] the acceptance decision is made for each order based on the feasibility of the current schedule. Accepting the order may require rescheduling. However in study [2], the orders are accepted only if they are completed before their due dates and if inserting them is possible without changing the current schedule for already accepted orders. A workload control system is also widely used in the literature as an acceptance decision. In that case, the acceptance decision takes into account the available sufficient capacity in order to complete a set of orders before their due dates. In studies [106, 101, 86, 87, 50, 51, 35], the order acceptance decision is based on the total workload of the set of already accepted orders. Similarly, in Guerrero and Kern [45], accept/reject decisions should consider levels of finished goods and work in process (WIP) inventories as well.

In the remaining studies, a set of orders are available at the time of the decision making and simultaneous acceptance and scheduling decisions are made for all orders. This is also the approach we take in this thesis.

Solution Methods

These studies can further be analyzed according to the solution methods. Different mathematical programming models have been proposed for the order acceptance and scheduling problems. Charnsirisakskul et al. [21] suggest a time-indexed MILP formulation for their preemptive problem. Roundy et al. [89] develop a discrete time integer programming formulation, Wu and Chen propose a MILP formulation [109] and propose multiple objective programming [110] for their associated problems.

The majority of researchers propose both exact and heuristic methods in their studies; therefore, we do not prefer to categorize these studies under exact and heuristic approaches. Instead, we prefer to summarize the solution methods of the papers.

Slotnick and Morton [96] provide a branch and bound algorithm together with a beam-search heuristic and a myopic heuristic for the single machine problem without preemption. Later Ghosh [42] gives two pseudo-polynomial time algorithms and a fully polynomial time approximation scheme. Slotnick and Morton [95] extend their previous study [96] and provide similar algorithms. Lewis and Slotnick [69] propose exact dynamic programming algorithm with several myopic heuristics. Rom and Slotnick [88] develop a genetic algorithm for the same problem and compare it with the myopic heuristic given in [95]. Talla Nobibbon et al. [78] develop two branch-and-bound algorithms to solve small sized instances and six different heuristics to solve large sized instances. Oğuz et al. [80] propose a SA approach for their problem to solve small sized instances and they modify the ATCS rule using problem characteristics to solve large sized instances.

In his study [2], Akkan develops heuristic methods for his problem and Yang and Geunes [114] propose a two-phase heuristic procedure for their problem. Differing from these studies, De et al. [32] propose dynamic programming techniques. They also develop a fully-polynomial time approximation scheme and conduct simulation experiments to see how order characteristics are related to the order selection procedure. Gupta et al. [46] also provide a dynamic programming approach for solving their problem.

Other approaches to the order acceptance problem include simulation techniques studied by [101, 102, 106, 35, 50, 51] and decision theory studied by [12, 13, 14].

2.3 Related Tabu Search Algorithms

TS has been successfully implemented for different kinds of scheduling problems. Since we propose a TS algorithm in this study, we find it beneficial to provide a summary of the studies which are related to our problem and in which a TS algorithm has been implemented as a solution procedure.

A TS algorithm using a hybrid neighborhood consisting of both swap and insertion moves is implemented for a single machine scheduling problem with the objective of minimizing the setup costs plus the delay penalties by Laguna et al. [63]. James and Buchanan [52] focus on early/tardy scheduling on single machine and developed advanced TS strategies to solve the problem. However, the problems with different release dates as well as due dates and sequence-dependent set up times are sparsely considered in the literature. Nowicki and Zdrzalka [79] develop a tabu search approach for minimizing the maximum weighted lateness and total weighted tardiness in single machine scheduling problem with batch and order setups. They adopt insertion as the move operator. In [17] and [16] Bilge et al. propose a deterministic TS algorithm with hybrid neighborhood and dynamic tenure structures. In [17], they consider minimizing total tardiness in a parallel machine environment where orders have different arrival times and sequence-dependent setup times. In [16], they examine a single machine case and take into account the weights associated with tardiness; however, they do not consider different arrival and sequence-dependent setup times anymore. In both studies, they investigate candidate list strategies to restrict the neighborhood. According to the results they reported both the quality of the results and the computation time obtained under candidate list strategies are superior to the case when no candidate list strategy is employed.

Wan and Yen [105] focus on minimizing the weighted earliness and tardiness on a single machine where orders have distinct due windows. To obtain final order sequences, they propose a TS heuristic with an optimal timing algorithm. They use adjacent pairwise interchange as the move operator to construct the neighborhood and employ a multi-start procedure with several different initial solutions to escape from the local minima. They employ a fixed size of circular tabu list and set the termination criterion predetermined number of non-improving iterations. Another TS algorithm designed for multi-mode resource-constrained project scheduling with the schedule-dependent setup times problem where the aim is to

minimize the project duration is implemented by Mika et al. [75].

Choobineh et al. [24] employ a multi-objective TS for single machine scheduling problem with sequence-dependent setup times. This heuristic keeps independent parallel tabu lists associated with each objective and employs a swap move with a maximum allowable distance. The idea of this move distance is to narrow the neighborhood by allowing to swap only the orders which have a smaller distance than the maximum allowable distance. Bozejko et al. [19] propose a TS algorithm with specific neighborhood and compound move techniques for single machine total weighted tardiness problem. In compound move techniques, they perform swap and insert operators simultaneously in the same iteration. They keep cyclic tabu list with dynamic length. They examine the order blocks in their study which enable the solution space to restrict and to reduce the size of the neighborhood.

Xu et al. [111] develop a two-layer-structured algorithm based on a TS for the single machine scheduling problem with arbitrary release and due dates where the processing times are controllable. Their objective is to minimize total resource consumption while meeting the due dates. In their algorithm they adopt a branch and bound algorithm to construct their initial solution. They only consider insert moves to construct their neighborhoods. They implement a constant sized tabu-list which was activated only after an improvement was not achieved for a number of iterations. They set a predetermined number of iterations as a termination condition.

Woodruff and Spearman [107] are the only researchers who develop a TS algorithm for an order acceptance problem, to the best of our knowledge. The main differences of their problem are the existence of family setups and the objective function that includes holding and setup costs rather than tardiness.

To the best of our knowledge, the proposed method in this study is the first TS algorithm proposed for the OAS problem in the literature.

Chapter 3

ORDER ACCEPTANCE AND SCHEDULING (OAS) PROBLEM

In this thesis, we focus on a MTO system where the production process is initiated by the customer. We consider deterministic order arrival case implying at the beginning of planning process different order characteristics specified by the customer are available for the manufacturer to realize the production process. In this problem setting, we assume that the machine can process one order at a time and no machine break downs occur. Preemption of the orders is not allowed and as the feature of MTO system no inventory is carried for incoming orders. Furthermore, no holding cost incurs for finished orders implying the early delivery is not penalized. However, late delivery of the orders results in tardiness penalties.

3.1 Problem Definition

The OAS problem is defined formally as follows. In a single machine environment where the production capacity is limited, a set of incoming orders is available at time zero. Each incoming order i is identified with a release date, r_i, a processing time, p_i, a due date, d_i, a deadline, $\bar{d}_i$, a maximum revenue, e_i, a weight (tardiness penalty), w_i, and a sequence-dependent setup time, s_{ji}, incurred when order j immediately precedes order i in the processing sequence. The setup times are not symmetric which implies that $s_{ij} \neq s_{ji}$ is possible.

In this problem setting, the manufacturer neither pays a cost nor gains a revenue from the rejected order. As can be seen in Figure 3.1, the revenue that can be gained from an accepted order depends on the following situations. Since sum of release date and processing time of an order i represents the earliest possible completion time for this order, no revenue can be gained until this time. The manufacturer gains the maximum revenue, e_i, if the tardiness of order i is zero. However, if the order is tardy, the revenue to be gained from this order, $Revenue_i$, decreases linearly with its tardiness, T_i. The manufacturer may complete order i until its deadline $\bar{d}_i$, but for each time unit beyond its due date d_i, a tardiness penalty cost

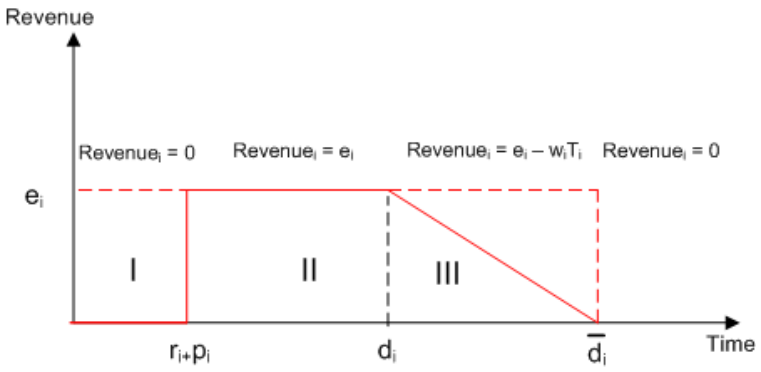

Figure 3.1: Revenue function for an accepted order

of w_i is incurred. No revenue can be gained from order i if its completion time exceeds its deadline, $\bar{d}_i$, and the revenue of order i equals to zero at its deadline. To ensure this, we set w_i to $e_i/(\bar{d}_i\text{-}d_i)$ for each order i. Tardiness of order i is $T_i = \max\{0, C_i - d_i\}$ where C_i is the completion time of order i. Then revenue from order i is $Revenue_i = e_i\, I_i - w_i\, T_i$. Here, I_i is an indicator that equals to 1 if order i is accepted, and 0 otherwise. Our aim is to find the optimal set of accepted orders and their schedule that maximize the manufacturer's total revenue, expressed as $\sum_{i=1}^{n} Revenue_i$.

3.2 MILP Formulation for the OAS Problem

We present the MILP formulation that was developed in Oguz et al. [80] below for the sake of completeness since the performance of the heuristics is compared to the upper bounds generated using MILP formulation.

In the MILP formulation, two sets of binary decision variables are defined. I_i equals to 1 if order i is accepted, and 0 otherwise, and y_{ij} equals to 1 if order i precedes order j immediately, and 0, otherwise. To represent sequence-dependent setup times accurately two dummy orders, order 0 and order $n+1$ are defined. Order 0 is assigned to the first position and order $n+1$ is assigned to the last position. These dummy orders are available at time zero, with r_0, r_{n+1}, p_0, p_{n+1}, d_0, $\bar{d}_0$, e_0 and e_{n+1} being 0; d_{n+1} and $\bar{d}_{n+1}$ being equal to the maximum deadline among all orders. The model is given below.

MILP:

$$\max \sum_{i=1}^{n} R_i$$

$$s.t. \tag{3.1}$$

$$\sum_{j=1,j\neq i}^{n+1} y_{ij} = I_i \qquad \forall i = 0,...,n \tag{3.2}$$

$$\sum_{j=0,j\neq i}^{n} y_{ji} = I_i \qquad \forall i = 1,...,n+1 \tag{3.3}$$

$$C_i + (s_{ij} + p_j) \times y_{ij} + \bar{d}_i \times (y_{ij} - 1) \leq C_j \qquad \forall i = 0,...,n, \forall j = 1,...,n+1, i \neq j \tag{3.4}$$

$$(r_j + p_j) \times I_j + s_{ij} \times y_{ij} \leq C_j \qquad \forall i = 0,...,n, \forall j = 1,...,n+1, i \neq j \tag{3.5}$$

$$T_i \geq C_i - d_i \qquad \forall i = 0,...,n+1 \tag{3.6}$$

$$C_i \leq \bar{d}_i \times I_i \qquad \forall i = 0,...,n+1 \tag{3.7}$$

$$T_i \leq (\bar{d}_i - d_i) \times I_i \qquad \forall i = 0,...,n+1 \tag{3.8}$$

$$T_i \geq 0 \qquad \forall i = 0,...,n+1 \tag{3.9}$$

$$R_i \leq e_i \times I_i - T_i \times w_i \qquad \forall i = 1,...,n \tag{3.10}$$

$$R_i \geq 0 \qquad \forall i = 1,...,n \tag{3.11}$$

$$C_0 = 0,\ C_{n+1} = \max_{i=1,...,n} \{\bar{d}_i\}, \qquad \forall i = 1,...,n \tag{3.12}$$

$$I_0 = 1,\ I_{n+1} = 1 \tag{3.13}$$

$$I_i \in \{0,1\}, y_{ij} \in \{0,1\} \qquad \forall i = 1,...,n \tag{3.14}$$

Constraint (3.2) and (3.3) together enforce that if an order is accepted, there is one preceding and one succeeding order. Constraint (3.4) enforces that if order i is followed by order j, then C_j should be at least C_i plus the setup time between i and j and the processing time of order j. If order i is not followed by order j, then the constraint states that C_i is at most its deadline plus a nonnegative term. Constraint (3.5) ensures that if order j is accepted, C_j should be at least its release time plus its processing time, and if order j is preceded by order i, the setup time between i and j is also added. In the case that order j is not accepted, the constraint reduces to $C_j \geq 0$. Constraints (3.6)-(3.9) calculate the tardiness of each order. Constraint (3.7) ensures that any order completed after the

deadline will not be accepted. Constraints (3.10) and (3.11) compute the revenue gained when order i is accepted and incurs a tardiness of T_i. Constraint (3.12) sets the completion times of the dummy orders and Constraint (3.13) states that dummy orders are accepted. Constraint (3.14) defines the binary variables.

3.3 The Computational Complexity of the Problem

The OAS problem is the generalization of many well-known scheduling problems such as single machine total weighted tardiness problems with sequence-dependent setup times, order acceptance problem with weighted tardiness objective and prize collecting traveling salesman problem (PCTSP) with time windows. The reduction of the OAS problem to these subproblems is given in [113]. Since one of its subproblems which is single machine total tardiness problem is known to be NP-hard [34], the OAS problem is also NP-hard.

Solving the OAS problem even for very small values of n is very troublesome. As n increases, the solution space of the OAS problem also increases rapidly. The solution space including both feasible and infeasible solutions can be calculated as follows. Assume that there exist n incoming orders from which the manufacturer is required to select. The selection of i orders among n generates $C(n, i)$ solutions and for this selected subset sequencing of these i orders generates $i!$ solutions. Hence, the solution space has $\sum_{i=1}^{n} C(n, i) \times i!$ solutions which implies an exact algorithm will be very time consuming to find the optimal solutions.

Due to this computational difficulty, we develop a heuristic algorithm as described in the next section to find near optimal solutions in a reasonable amount of time.

Chapter 4

A HEURISTIC SOLUTION APPROACH

As we stated in Section 3.3, the OAS problem is strongly NP-hard. In fact, an exact algorithm cannot solve the problem even for very small values of n in reasonable time limits. This reason inspires us to develop a heuristic algorithm which can produce high-quality solutions in a reasonable amount of time.

The travelling salesman problem (TSP) displays similar characteristics with the OAS problem due to sequence-dependent setup times. Since TS is known to be successful for solving the TSP compared to other metaheuristics, we choose to develop a TS algorithm for solving the OAS problem. Furthermore, the OAS problem which has same problem settings in thesis is only studied in [80] so far. In this study, a simulated annealing (SA) based heuristic which is called Iterative Sequence First-Accept Next Algorithm (ISFAN) is proposed to solve the problem. This implies that there is no TS method applied to this problem so far. These are the reasons why we choose TS heuristic as a solution procedure in this thesis.

4.1 Tabu Search

TS, which is originally proposed by Glover [43], is known to perform quite well for scheduling and routing problems. The general idea of TS can be summarized as follows. TS starts with an initial feasible solution and tries to improve it iteratively. It keeps the recently visited solutions in tabu list and forbids to move toward them to avoid cycling. Therefore, the solutions that can be reachable from the current solution is restricted to those which are not belong to tabu list and these solutions are referred to the allowed set. In TS, all neighborhood solutions are evaluated and then the best solution from the allowed set is chosen as the new current solution at each iteration. Then, this solution is added to tabu list and one of the solutions that were already in the tabu list is removed according to a

determined rule. The algorithm stops when a termination condition is met.

In rest of this chapter, we propose an efficient TS heuristic to solve the OAS problem which enables to make the acceptance and scheduling decisions simultaneously.

4.2 Proposed TS Algorithm

For solving the OAS problem, we develop a TS algorithm which is supported with probabilistic local search after each iteration. The success of TS in studies [17, 105, 24, 19] motivated us to develop a TS algorithm for the OAS problem. However, we observed that the TS algorithm cannot search the solution space efficiently when acceptance and sequencing decisions have to be made simultaneously. Hence, we incorporated a probabilistic local search into TS algorithm to complement its strengths. The pseudocode of the algorithm is provided on next page.

In the following subsections, we describe each element of the TS algorithm in detail.

4.2.1 Solution Representation

We represent a solution by a vector in which the i^{th} entry indicates the position of order i in the sequence. If order i is not accepted, the corresponding entry is set to zero. As an example, the representation of a solution with orders $i = 1, \ldots, 10$ is [1 0 5 0 0 0 2 4 3 0]. This representation indicates that orders $2, 4, 5, 6, 10$ are rejected.

The 1^{st} order is processed in the first position, the 3^{rd} order in the fifth position, the 7^{th} order in the second position, the 8^{th} order in the fourth position and the 9^{th} order in the third position; hence, the order processing sequence is $1-7-9-8-3$. This representation keeps both the acceptance and the sequencing decisions efficiently as it uses a vector of size n, where n is the number of orders.

4.2.2 Initial Solution

TS starts with an initial feasible solution and tries to improve it iteratively. For the OAS problem, we use a greedy rule to construct the initial solution s_0. The rule takes revenues, processing times and setup times into account. It requires the calculation of the following *Revenue-load ratio (RLR)* for each order: $RLR1_i = e_i/(p_i + s_{average,i})$, where $s_{average,i} = (s_{0,i} + s_{1,i} + ... + s_{n,i})/(n+1)$.

Notation

s_0: the initial solution

s: the current solution

s^*: the best known solution,

f_s: revenue of s,

$N(s)$: the neighborhood of s,

$\bar{N}(s)$, the *admissible* (non-tabu or allowed by aspiration) subset of $N(s)$.

s^{iter}: the best solution $\in \bar{N}(s)$

Algorithm 1 TS algorithm

Require: r_i, p_i, s_{ij}, d_i, $\bar{d}_i$, e_i, w_i, n

Set $f(s^*) = 0$, $TabuTenure = k$. ▷ k depends on n
Generate s_0.
$s \leftarrow s_0$.
while termination criterion is not met **do**
    Construct $N(s)$ ▷ TS PART begins
    Find $s^{iter} \in \bar{N}(s)$
    $s \leftarrow s^{iter}$
    Update the $TabuList$
    Update s^* and $f(s^*)$. ▷ TS PART ends
    for m:1 to $\lceil n/6 \rceil$ **do** ▷ LOCAL SEARCH PART begins
        Select the order that gives minimum $RLR2$ (see p.31) to be dropped from s
        Find $reject$ (rejected orders array) for s
        while size of $reject > 0$ **do**
            Calculate $RLR1$ for each order $i \in reject$
            Construct cumulative probabilities for all orders proportional to their $RLR1$
            Find the order to be inserted from $reject$ randomly
            Set $l = 1$
            for $l \leq n$ **do**
                Insert selected order to l^{th} position
                Reject orders with zero revenue
                Set inserted solution as $s^{inserted}$
                if $f(s^{inserted}) \geq 0.998 f(s^{iter})$ **then**
                    $s \leftarrow s^{inserted}$ and set $l = n$
                else
                    l++
                end if
            end for
            Delete the order selected for insertion from $reject$
        end while
    end for ▷ LOCAL SEARCH PART ends
end while

We sort the orders with respect to the defined ratio, starting from the highest, to give priority to orders that potentially generate a higher revenue and take a small amount of time to process. After sorting the orders, we calculate the profit of the sequence while making a feasibility check simultaneously. We begin to calculate $Revenue_i$ for each order i, if $Revenue_i$ results with positive revenue ($Revenue_i > 0$), we keep the sequence of order i, however, if it results with negative revenue ($Revenue_i < 0$) we delete order i from its position and reschedule the orders coming after this order. When we reach end of the sequence, implying that all the orders in current sequence have positive revenues and the sequence is feasible, we obtain the feasible initial solution for the TS algorithm.

4.2.3 Move Operator and Neighborhood Definition

The move operators within TS algorithms developed for similar problems [17, 105, 24, 19, 107] are swap and insertion. These operators are preferred due to their efficiency. Therefore, we generate the neighborhood of the current solution, s, by swapping two entries of the solution vector. This seemingly simple pairwise exchange move allows us to change both the set of accepted orders and their sequence, while keeping the number of accepted orders the same.

For example, applying this operator to solution [1 0 5 0 0 0 2 4 3 0], whose processing sequence is $1-7-9-8-3$, the neighborhood of the solution includes the following solutions obtained by swapping the first entry with the others: [0 1 5 0 0 0 2 4 3 0], [5 0 1 0 0 0 2 4 3 0], [0 0 5 1 0 0 2 4 3 0], [0 0 5 0 1 0 2 4 3 0], [0 0 5 0 0 1 2 4 3 0], [2 0 5 0 0 0 1 4 3 0], [4 0 5 0 0 0 2 1 3 0], [3 0 5 0 0 0 2 4 1 0], [0 0 5 0 0 0 2 4 3 1]. The first neighboring solution, which is obtained by swapping the first and the second entries, corresponds to the processing sequence $2-7-9-8-3$. In this case, order 1 is rejected and order 2 is accepted in its position, thus the set of accepted orders changes. On the other hand, the second solution is obtained by swapping the first and the third entries and it corresponds to the processing sequence $3-7-9-8-1$. As it can be seen, the set of accepted orders is the same but the sequence has changed. For both of the solutions, the number of accepted orders remains same. Since exchanging the two entries that have the value of zero will result in same solution as current one and can cause the algorithm to cycling, we do not exchange if both of the entries that will be swapped in the current solution are zero. As a result, the

complete neighborhood of a given solution consists of at most $n\,(n-1)/2$ solutions.

We note that when we swap two orders, the completion time, hence the tardiness and the revenue of each order starting from the first swapped order will be affected due to release dates, sequence dependent setup times and deadlines. If any of the orders in new sequence attain zero revenue, they should be rejected to assure the feasibility. Thus, the number of accepted orders may decrease in this way. We handle this concern with a local search after a TS iteration as explained in Section 4.2.6.

4.2.4 Tabu List and Tabu Tenure

In TS, the tabu list ($TabuList$) keeps the most recent moves to avoid cycling while searching for a new solution. The tabu tenure parameter, i.e. size of the tabu list, helps to avoid cycling but also affects the search. As the tabu tenure increases, more moves will be restricted and the algorithm will explore a smaller neighborhood. In our implementation, the tabu list is formed with the k most recently performed swaps, where k is the tabu tenure ($TabuTenure$). In the tabu list, the swapped pairs are kept so that the same orders are not swapped again during the tabu tenure. For example, when the current solution is [2 0 5 0 0 0 1 4 3 0] and the best solution found in the neighborhood is [1 0 5 0 0 0 2 4 3 0], then the swap of the entries corresponding to order pairs (1,7) and (7,1) will be tabu during the tabu tenure.

4.2.5 Aspiration and Termination Criteria

In our implementation we used classical aspiration criterion. The classical aspiration criterion indicates that if the revenue of a tabu solution is better than the revenue of the best-known solution so far, this solution is accepted even though it is in the tabu list. We set the termination criterion as 50 iterations without an improvement in objective function.

4.2.6 Local Search Procedure

After each TS iteration, we also apply local search around the best TS solution. The reason why we apply a local search is that, as a consequence of our move operator the number of accepted orders remains same when moving from one solution to another. And when we reach the next solution, we are required to calculate its revenue and make a feasibility check which may cause the number of accepted orders in this solution to decrease. If the

iterations proceed this way, the algorithm is likely to converge to a poor local optimum. To remedy this, after each TS iteration, we perform a local search starting from the best solution obtained in the neighborhood by applying iterative drop-add-insert operations.

Drop operation: To be able to add orders with larger revenue, we first drop an order that brings low revenue and consumes a large amount of time to process. More specifically, we drop the order which has the minimum value of $RLR2_i = e_i/(p_i + s_{ji)}$, where order j is the immediate predecessor of order i in the current sequence. Note that, $RLR2$ is a modified version of $RLR1$. Since we know the required s_{ji} value for order i, we use this term instead of $s_{average,i}$.

Add operation: We try to add each of rejected orders to the current sequence iteratively. To select the order to be added first, we use the original revenue-load ratio ($RLR1$) to generate a probability distribution for the rejected orders similar to roulette wheel selection [76]. Briefly, we construct cumulative probabilities using $RLR1$ of orders. Then, we generate a random number and by checking in which interval this random number fits we decide which rejected order to add first. Note that we try to add each of the rejected orders but in which sequence these rejected orders will be added is an important decision for the algorithm. By applying the idea of roulette wheel selection, we assign a probability to each rejected order i that is proportional to its $RLR1_i$ so that orders with higher $RLR1$ are more likely to be selected. This allows us to bring in some randomness in the algorithm, providing a diversification mechanism.

Insert operation: Once the order to be added is selected, in which position to insert this order is decided as follows. We try each position from the beginning of the order processing sequence and calculate the new revenue. Since this move may result in deleting one or more orders to maintain the feasibility of the solution, we introduce a threshold value (*improvement threshold*) for accepting inferior solutions similar to the acceptance idea of simulated annealing. That is, we insert the order at the current position if this generates a feasible sequence with a total revenue at least 0.998 times the revenue of the best solution in the neighborhood, s^{iter}, and we do not consider the remaining positions. Otherwise, we continue with the next position until the end of the sequence. After the insertion operation of one rejected order is finished (it is also possible that a rejected order cannot be inserted into the sequence), we continue with the selection of the next rejected order which will be

added. This procedure is repeated until all rejected orders are considered for the addition operation.

This results in a compound move in which one drop move is followed by multiple add-insertion moves and each such move may necessitate dropping some orders to maintain feasibility. We note that this provides another diversification mechanism for our algorithm. We repeat the compound drop-add-insert move $\lceil n/6 \rceil$ times based on our preliminary tests. The iterative application of the compound move allows us to accept a profitable order, which was formerly rejected due to infeasibility, at a new position. After the local search, the algorithm returns back to the next TS iteration.

The flow chart of the TS algorithm is given in Figure 4.1.

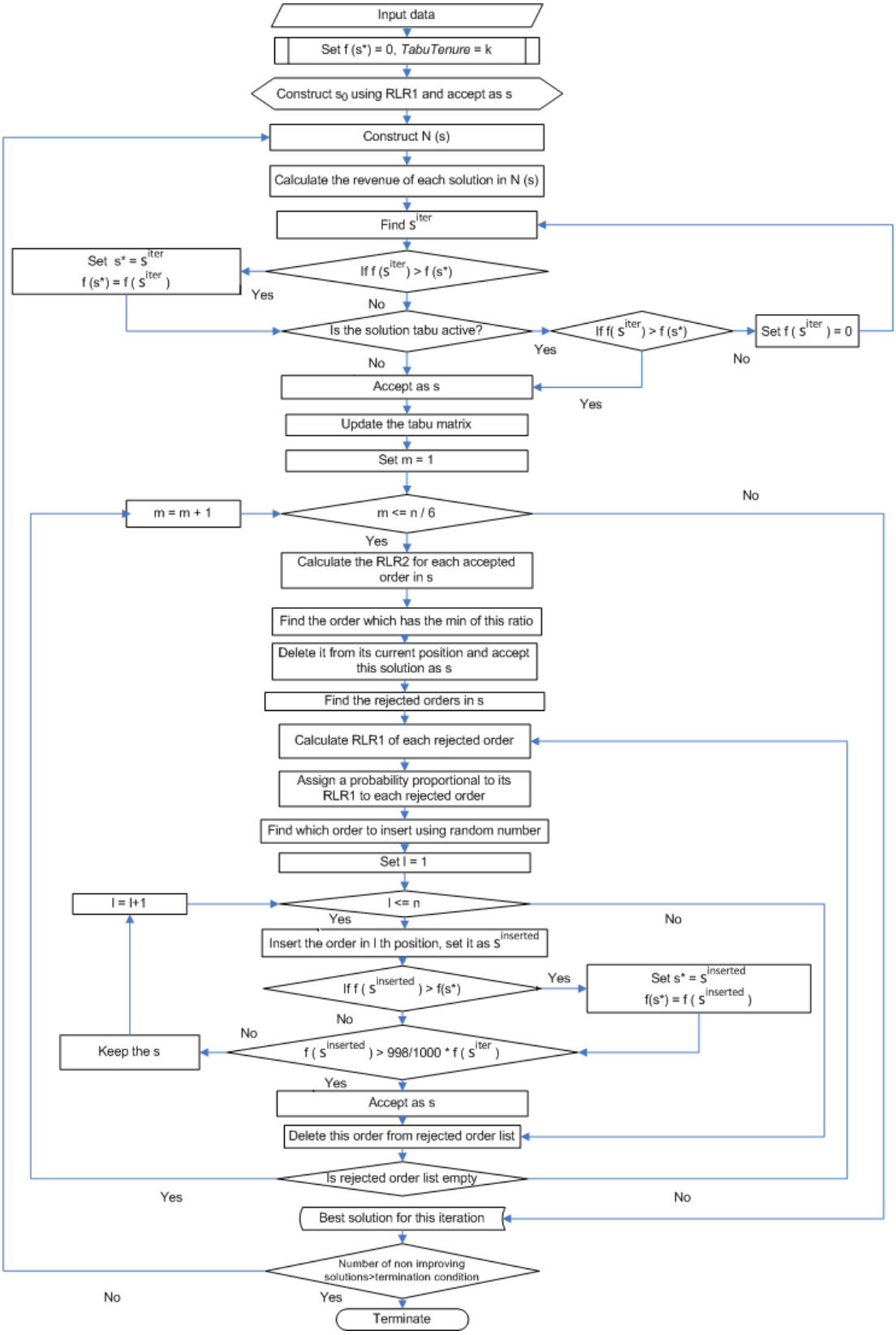

Figure 4.1: Flowchart of the TS algorithm

Chapter 5

COMPUTATIONAL STUDIES

In this chapter, we conducted computational experiments to analyze the performance of the TS algorithm. In Section 5.1 and 5.2, we describe how the test instances are generated and how the parameters of the TS algorithm are set, respectively. In Section 5.3, we give the results of our computational experiments and discuss these results in detail in Section 5.4.

5.1 Data Generation

In order to test the performance of the TS algorithm, we generated new test instances. There is data set generated for the OAS problem in [80], however this test bed was somewhat easier, especially for large sized instances. Therefore,we generated new test instances in varying parameter values and problem sizes. In the new data set, the tardiness factor τ and the due date range R take five different values: $0.1, 0.3, 0.5, 0.7, 0.9$ as in [84]. For each combination of τ and R, 10 problem instances are generated. Hence, the number of test instances for each problem size, n, is $10 \times 25 = 250$, which sums up to 1500 instances for six different n values; $n = 10, 15, 20, 25, 50, 100$.

A discrete uniform distribution is used to generate the following parameters: processing times and revenues from the interval [0,20]; setup times from [1,10]; release dates from $[0, \tau\, p_T]$, where p_T is the total processing time of all orders, as in Aktürk and Özdemir [3]. Due dates are generated as $d_i = r_i + \max\limits_{j=0,1,\ldots,n} s_{ji} + \max\{slack, p_i\}$, where $slack$ is drawn from a discrete uniform distribution in the interval $[p_T(1-\tau-R/2), p_T(1-\tau+R/2)]$ similar to the study by Potts and van Wassenhove [84]. Deadlines are generated from the formula $\bar{d}_i = d_i + R\, p_i$, as in Charnsirisakskul et al. [21]. All data parameters except the weights are integer numbers and the weights are calculated as $w_i = e_i/(\bar{d}_i - d_i)$ to ensure that the revenue gained from an order drops 0 at its deadline.

5.2 Parameter Settings for the TS Algorithm

In order to obtain the best performance of the TS algorithm, we performed preliminary tests to set the parameters of the algorithm. We used one special case of test instances selected randomly, where τ=0.3 and R=0.5 for all n values in preliminary tests. These parameters are: tabu tenure, improvement threshold, number of iterations of drop-add-insert operations in the local search procedure and termination criterion.

Tabu tenure: One of the most important parameters of the TS is tabu tenure which helps to avoid cycling but also affects the search. As the tabu tenure increases, a smaller neighborhood will be explored. On the other hand, the smaller the tabu tenure, the easier the algorithm to cycling. To decide the best value of tabu tenure, we examined the effect of four different tabu list sizes for every n value. We tested the following values of tabu tenure.

Tabu tenure = 3, 4, 5, 7 for n=10

Tabu tenure = 5, 8, 10, 15 for n=15

Tabu tenure = 7, 10, 12, 20 for n=20

Tabu tenure = 7, 9, 13, 25 for n=25

Tabu tenure = 7, 17, 25, 35 for n=50

Tabu tenure = 7, 34, 50, 75 for n=100

The affect of different tabu tenures can be seen in Table 5.1. In this table and in Tables 5.2, 5.3 the values presented are the run times and the objective values of the solutions output by the TS algorithm, averaged over 10 instances of the selected case, where τ=0.3 and R=0.5 and the bold values in all these tables show our selected parameter values.

According to the results in Table 5.1, the best performance was achieved when the tabu tenure is $\lceil 2n/3 \rceil$ and the $TabuTenure$ was set accordingly. This implies k is equal to 7 for 10 orders, 10 for 15 orders, 13 for 20 orders, 17 for 25 orders, 34 for 50 orders and 67 for 100 orders.

Improvement threshold: As mentioned in 4.2.6, we introduce a threshold value for accepting inferior solutions in the local search procedure similar to the acceptance idea of simulated annealing. We introduce an improvement threshold value because we first implement the drop operation and then try to add a rejected order. Since drop operation cause the revenue of the solution to drops concurrently, we necessitate to allow the acceptance of

Table 5.1: Preliminary test results for tabu tenure values where τ=0.3 and R=0.5

n	Tabu tenure	TS Time	TS Result
10	3	0.00	103.90
	4	0.00	104.04
	5	0.00	104.04
	7	**0.00**	**104.20**
15	5	0.00	162.94
	8	0.00	163.30
	10	**0.00**	**163.57**
	15	0.00	163.04
20	7	0.00	219.59
	10	0.00	219.96
	13	**0.01**	**220.03**
	20	0.01	219.62
25	7	0.01	256.67
	13	0.01	255.92
	17	**0.01**	**256.06**
	25	0.01	255.97
50	7	0.19	536.60
	17	0.17	539.70
	25	0.18	537.30
	34	**0.18**	**540.63**
100	7	13.85	1067.10
	34	14.84	1068.50
	67	**16.07**	**1072.10**
	75	15.04	1069.00

some inferior solutions.

We examined six different values, 0.990, 0.995, 0.996, 0.997, 0.998, 0.999, to set the improvement threshold value. According to the results given in Table 5.2, we set the improvement threshold value to 0.998.

Table 5.2: Preliminary test results for improvement threshold value where τ=0.3 and R=0.5

n	improvement threshold	TS Time	TS Result
10	0.990	0.00	105.40
	0.995	0.00	104.20
	0.996	0.00	104.20
	0.997	0.00	104.20
	0.998	**0.00**	**104.20**
	0.999	0.00	104.20
15	0.990	0.00	162.93
	0.995	0.00	163.50
	0.996	0.00	163.77
	0.997	0.00	163.57
	0.998	**0.00**	**163.74**
	0.999	0.00	163.68
20	0.990	0.00	219.52
	0.995	0.01	220.47
	0.996	0.00	220.03
	0.997	0.01	220.17
	0.998	**0.00**	**220.71**
	0.999	0.00	220.05
25	0.990	0.02	257.84
	0.995	0.01	254.90
	0.996	0.01	256.64
	0.997	0.01	257.60
	0.998	**0.07**	**255.90**
	0.999	0.08	257.20
50	0.990	1.15	538.50
	0.995	1.26	540.14
	0.996	1.15	538.50
	0.997	1.14	540.68
	0.998	**1.15**	**537.00**
	0.999	1.15	536.60
100	0.990	17.08	1070.40
	0.995	13.07	1067.30
	0.996	16.08	1070.00
	0.997	17.08	1070.00
	0.998	**16.18**	**1070.40**
	0.999	17.08	1065.80

Number of iterations of drop-add-insert operations in local search procedure: We explained in 4.2.6, we apply a compound move in which one drop move is followed by multiple add-insertion moves iteratively in local search. The iterative application of compound move allows a profitable order, which was formerly rejected due to infeasibility caused by adding an order, to schedule at a new position.

We tested three values, $\lceil n/6 \rceil$, $\lceil n/5 \rceil$ and $\lceil n/4 \rceil$ for determining how many times to apply the compound move. The results given in Table 5.3 suggest that the best value is

$\lceil n/6 \rceil$, therefore we set the number of iterations of drop-add-insert operations to this value.

Table 5.3: Preliminary test results for number of drop-add-insert operations where τ=0.3 and R=0.5

n	# of drop-add-insert operations	TS Time	TS Result
10	n/4	0.00	104.20
	n/5	0.00	104.20
	n/6	**0.00**	**104.30**
15	n/4	0.00	163.28
	n/5	0.00	163.67
	n/6	**0.00**	**164.00**
20	n/4	0.01	220.14
	n/5	0.01	220.17
	n/6	**0.01**	**218.40**
25	n/4	0.01	256.70
	n/5	0.01	256.50
	n/6	**0.01**	**257.80**
50	n/4	1.13	538.50
	n/5	1.14	540.68
	n/6	**1.14**	**536.92**
100	n/4	17.08	1072.20
	n/5	17.08	1070.00
	n/6	**17.08**	**1072.30**

Termination criterion: Our termination criterion is to stop after a number of non-improving iterations. After testing 30, 50 and 100 as the number of non-improving iterations, the convergence behavior of the algorithm suggested using 50 non-improving iterations. An example to the convergence of the TS algorithm for an instance with 50 orders is given in Figure 5.1.

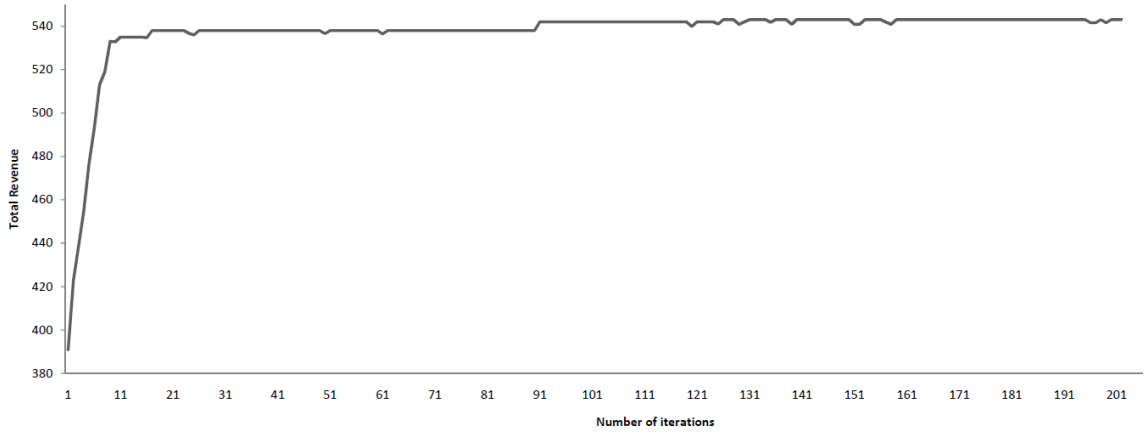

Figure 5.1: Convergence of the TS algorithm for an instance with 50 orders, $\tau = 0.3$ and $R = 0.7$

5.3 Results of the Computational Experiments

In this section, we explain how we conducted our computational experiments and how the results in Tables 5.4-5.9 are obtained. We analyze the obtained results in Section 5.4 in detail.

5.3.1 Computational Platform

All of the runs throughout these computational experiments are performed on a workstation with an Intel Xeon processor, 3.00 GHz speed, and 4GB of RAM. The TS algorithm was coded in C. For MILP runs and LP Relaxation runs, we solved test instances with ILOG CPLEX 11.2 and set a time limit of 3600 seconds for all MILP runs.

5.3.2 Benchmarks

To compare the performance of the TS algorithm, we used three benchmarks which are explained as follows.

m-ATCS: In the literature, Apparent Tardiness Cost (ATC) rule has been shown to be effective to construct a sequence for the total weighted tardiness scheduling problem [55]. Lee et al. [67] modified this rule to incorporate sequence-dependent setup times, named it Apparent Tardiness Cost with Setups (ATCS) and showed its effectiveness even for large

sized problems. Oguz et al. [80] adapted ATCS rule for the OAS problem and refer to it as m-ATCS. The m-ATCS heuristic is the modified version of the ATCS priority index $\frac{w_i}{p_i}\exp(\frac{\max(d_i-p_i-t_{current},0)}{k_1\bar{p}})\exp(\frac{-s_{previousOrder,i}}{k_2\bar{s}})$ by setting $k_1 = k_2 = 1$ and $w_i = e_i$. In this rule, $\bar{p}$ and $\bar{s}$ represent the average processing time and the average setup time, respectively. Instead of the weight term in the ATCS rule, the revenue term e_i is used in the m-ATCS rule since the aim of the OAS problem is to maximize the total revenue of the manufacturer. Hence, m-ATCS heuristic uses $\frac{e_i}{p_i}\exp(\frac{\max(d_i-p_i-t_{current},0)}{\bar{p}})\exp(\frac{-s_{previousOrder,i}}{\bar{s}})$ priority index to select the next order to schedule. After forming a sequence, orders with zero revenue are rejected until all accepted orders generate positive revenue. Similar to results of [67], m-ATCS was shown to be effective for large instances of OAS in [80] and has the advantage of very short run times. Hence, we selected m-ATCS as one of the benchmark heuristics. The details of the m-ATCS heuristic is given below.

Algorithm 2 m-ATCS algorithm

Require: r_i, p_i, s_{ij}, d_i, $\bar{d}_i$, e_i, w_i, n

Set $t_{current} = 0$, $profit = 0$, $sch = \{\}$, $unsch = \{\}$.
▷ sch: scheduled orders at time $t_{current}$.
▷ $unsch$: unscheduled orders at time $t_{current}$.
Calculate $p_{average} = (\sum_{i=1}^{n} p_i)/n$.
Calculate $s_{average} = (\sum_{i=1}^{n} \sum_{j=1}^{n} s_{ij})/(n+1)^2$.
$sch \leftarrow$ *order 0*, $unsch \leftarrow$ *all incoming orders.*
while *unsch* is not empty **do**
    for each $j \in unsch$ **do**
        Calculate
        $PI_j = (e_j/p_j) \times e^{-(max(d_j - p_j - t_{current}, 0)/p_{average})} \times e^{(-s_{ij}/s_{average})}$.
        ▷ PI: m-ATCS priority index
    end for
    Set $j^* \leftarrow$ *order with the highest PI ratio.*
    Calculate $t_{cur} = max(t_{current}, r_{j^*}) + s_{ij^*} + p_{j^*}$
    ▷ t_{cur}: completion time of order j^*
    if $t_{cur} \leq d_{j^*}$ **then** ▷ untardy order with max revenue
        Set $profit = profit + e_{j^*}$
        Set $t_{current} = t_{cur}$
        Remove j^* from *unsch*
        Add $sch \leftarrow j^*$ and set $i \leftarrow j^*$
    else if $t_{cur} \leq \bar{d}_{j^*}$ **then** ▷ tardy order with reduced revenue
        Set $profit = profit + e_{j^*} - w_{j^*} \times (t_{cur} - d_{j^*})$
        Set $t_{current} = t_{cur}$
        Remove j^* from *unsch*
        Add $sch \leftarrow j^*$ and set $i \leftarrow j^*$
    else ▷ order completed after its deadline
        Remove j^* from *unsch*
        $t_{current} = t_{current}$
    end if
end while

ISFAN Heuristic: ISFAN algorithm starts with a possibly infeasible initial solution in which all orders are accepted. Given a solution, the algorithm iterates between two main steps: order acceptance and sequencing. At the order acceptance step, it employs a priority rule similar to $RLR1$ in which the average setup time is replaced with minimum setup time. Thus, ISFAN uses $(e_i/(p_i + s_{min,i}))$ ratio for deciding which orders to keep in the sequence. At the sequencing step, ISFAN applies simulated annealing with exchange moves to find the best sequence for the current accepted order set. After a number of iterations, ISFAN terminates with a local search step. In the local search, an order insertion move is applied to add each rejected order back to the sequence, if possible. Next, an order exchange move

is applied to swap each rejected order with an accepted one. In both of these moves, the solution with the best improvement is kept.

ISFAN algorithm handles the order acceptance and scheduling decisions separately whereas our proposed TS algorithm considers these two decisions simultaneously. The second approach provides much better results as it can be observed from the Tables 5.4-5.9 and the analysis of these tables in Section 5.4. The pseudocode of the ISFAN algorithm is provided below.

Algorithm 3 ISFAN algorithm

Step 1. Read the input data

Step 2. Set the control parameters:

2.1. Initial temperature (T_{max})

2.2. Set $T_{current} = T_{max}$

Step 3. Set the parameters:

3.1. Total revenue=0;

3.2. best revenue=0;

Step 4. Construct the initial solution (not necessarily feasible)

4.1. Sequence the orders in descending order of $slacktime_j = \bar{d}_j - r_j - (minsetup_j + p_j)$;

4.2. Calculate the completion time C_j, the tardiness T_j and the gained revenue R_j for each j

4.3. Count the number of orders violating their deadlines d_j

4.4. Calculate the best revenue: best revenue=$\sum_j R_j$

Step 5. While number of violating orders > 0 do the following:

5.1. Perform the following SA-based loop ITER times:

5.1.1. Generate two different random integers a_1 and a_2 between 1 and n (n is the number of available orders)

5.1.2. Exchange the orders having the indices as a_1 and a_2

5.1.3. Calculate C_j's, T_j's and R_j's and the number of violating orders in this new sequence

5.1.4. Calculate the total revenue: total revenue=$\sum_j R_j$

5.1.4.1. If (Total revenue $>$ best revenue), accept the new sequence,

best revenue=Total revenue

5.1.4.2. If (Total revenue $\leq$ best revenue),

5.1.4.2.1. Calculate the probability of accepting, pr=exp(-(-Total revenue+best revenue)/$T_{current}$)

5.1.4.2.2. Select uniformly distributed random number m, from the interval (0,1)

5.1.4.2.2.1. If $m < pr$, accept the sequence, best revenue=Total revenue

5.1.4.2.2.2. If $m \geq pr$ reject the new sequence, and continue with the former best sequence

5.1.4.3. Return to Step (5.1.1)

5.1.5. Update the current temperature by using the selected cooling function

5.1.6. Calculate the revenue-load ratio $ratio_j = e_j/(p_j + minsetup_j)$ for violating orders

5.1.6. Reject the order having the smallest revenue-load ratio

5.1.7. Return to Step (4.2) with the new sequence

5.2. If the number of violating orders=0, i.e., a feasible solution is obtained, perform the following for ITER1 times:

5.2.1. Generate two different random integers a_1 and a_2 between 1 and n (n is the number of available orders)

5.2.2. Exchange the orders having the indices a_1 and a_2

5.2.3. Calculate C_j's, T_j's and R_j's and $newrevenue$ for this new sequence $newrevenue = \sum_j R_j$

5.2.3.1. If (new sequence is feasible) and ($newrevenue >$ best revenue):

5.2.3.1.1. best sequence=new sequence

5.2.3.1.2. best revenue= $newrevenue$

5.2.3.2. Else, preserve the current best sequence

MILP: To see how an exact method performs on test instances, we also decided to compare the results obtained with MILP formulation that was given in 3.2. Therefore, we report the best feasible solution found by the CPLEX solver within 3600 seconds time limit as the MILP solution and compare the MILP results with TS algorithm.

The solutions obtained by MILP, m-ATCS, ISFAN and TS are compared in terms of the solution quality, number of optimal solutions obtained and run time in Tables 5.4-5.9.

5.3.3 Upper Bounds

Since, solving test instances to optimality is impossible for large-sized problems we used an upper bound, UB, to measure the quality of the solutions obtained by benchmarks explained in 5.3.2. The UB is obtained as the best of the following two bounds. The first one, UB_{MILP}, is generated by solving MILP with a time limit of 3600 seconds and keeping the best upper bound obtained. The second one, UB_{LPVI}, is generated by solving the LP relaxation of MILP strengthened with the valid inequalities given below which are proposed in [80].

$$C_{n+1} \leq \max_{i=1,\ldots,n} \bar{d}_i; \tag{5.1}$$

$$C_{n+1} \geq \min_{i=1,\ldots,n} r_i + \sum_{i \in O} [(p_i + minsetup_i) \times I_i]; \tag{5.2}$$

$$C_i \geq (r_i + minsetup_i + p_i) \times I_i, \qquad \forall i = 1, \ldots, n; \tag{5.3}$$

where $minsetup_i = \min_{j=1,\ldots,n} s_{ji}$.

5.3.4 Performance Measures

Percentage Deviations: We use the percentage deviation of the objective function values, *objective*, obtained by each benchmark from the UB as one of the performance measures.

$$\frac{1}{n} \times \sum_{i=0}^{n} \frac{UB - objective}{UB} \tag{5.4}$$

The solutions obtained by MILP, m-ATCS, ISFAN and TS are compared in terms of the maximum, average and minimum percentage deviation from UB and the results are reported in Tables 5.4-5.9.

Run Times: Another important performance measure is average run time, thus we report the average run times of MILP, ISFAN and TS in CPU seconds, except for m-ATCS which takes less than a second for all test instances, in Tables 5.4-5.9.

Number of optimal solutions:The number of optimal solutions out of 10 instances of each type is also recorded as an auxiliary measure. We note that the percentage deviation is a better indicator for a company that faces the OAS problem due to time concerns in obtaining optimal solutions.

Table 5.4: Performance of MILP, m-ATCS, ISFAN and the TS algorithm for n=10

		% Deviations from UB																		
$n = 10$		MILP			m-ATCS			ISFAN			TS			# of optimal solutions				Run Times		
τ	R	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	MILP	m-ATCS	ISFAN	TS	MILP	ISFAN	TS
0.1	0.1	0%	0%	0%	25%	9%	0%	10%	3%	0%	3%	1%	0%	10	1	6	6	1123.90	22.93	0.00
	0.3	0%	0%	0%	19%	9%	2%	9%	3%	0%	3%	1%	0%	10	0	3	5	1119.90	21.32	0.00
	0.5	6%	1%	0%	27%	9%	0%	8%	3%	0%	8%	1%	0%	9	1	4	8	1037.20	16.35	0.00
	0.7	0%	0%	0%	23%	10%	0%	10%	3%	0%	3%	0%	0%	10	1	6	8	435.00	12.69	0.01
	0.9	0%	0%	0%	21%	8%	0%	0%	0%	0%	1%	0%	0%	10	1	9	7	289.63	10.63	0.00
0.3	0.1	0%	0%	0%	19%	10%	0%	12%	4%	0%	2%	0%	0%	10	1	2	9	466.20	28.90	0.00
	0.3	0%	0%	0%	33%	14%	0%	10%	2%	0%	2%	1%	0%	10	1	6	6	372.40	28.12	0.00
	0.5	0%	0%	0%	40%	16%	0%	19%	7%	0%	0%	0%	0%	10	1	2	6	257.30	24.24	0.01
	0.7	0%	0%	0%	36%	12%	1%	16%	6%	0%	2%	1%	0%	10	0	1	6	310.30	19.71	0.00
	0.9	0%	0%	0%	42%	21%	5%	29%	6%	0%	0%	0%	0%	10	0	5	4	118.20	16.84	0.00
0.5	0.1	0%	0%	0%	24%	14%	4%	20%	9%	0%	6%	1%	0%	10	0	3	9	22.10	33.33	0.00
	0.3	0%	0%	0%	34%	14%	0%	20%	5%	0%	5%	1%	0%	10	0	4	8	52.60	32.48	0.00
	0.5	0%	0%	0%	23%	15%	3%	9%	3%	0%	0%	0%	0%	10	0	4	9	13.90	30.99	0.00
	0.7	0%	0%	0%	40%	16%	1%	32%	6%	0%	2%	0%	0%	10	0	4	4	24.30	27.27	0.00
	0.9	0%	0%	0%	35%	15%	1%	28%	6%	0%	2%	0%	0%	10	0	3	7	26.70	23.85	0.00
0.7	0.1	0%	0%	0%	34%	14%	0%	23%	6%	0%	4%	0%	0%	10	1	4	9	1.30	42.10	0.00
	0.3	0%	0%	0%	27%	14%	0%	23%	7%	0%	0%	0%	0%	10	2	5	10	1.50	42.67	0.00
	0.5	0%	0%	0%	49%	24%	4%	12%	4%	0%	1%	0%	0%	10	0	3	9	1.60	37.37	0.00
	0.7	1%	0%	0%	26%	13%	0%	19%	8%	0%	0%	0%	0%	10	1	1	8	1.70	38.71	0.00
	0.9	0%	0%	0%	36%	20%	9%	19%	6%	0%	4%	0%	0%	10	0	1	6	1.90	28.31	0.00
0.9	0.1	0%	0%	0%	17%	8%	0%	11%	2%	0%	0%	0%	0%	10	3	7	10	1.00	59.88	0.00
	0.3	0%	0%	0%	29%	11%	0%	18%	7%	0%	0%	0%	0%	10	2	2	9	1.00	54.97	0.00
	0.5	0%	0%	0%	22%	9%	0%	27%	8%	0%	0%	0%	0%	10	3	5	8	1.00	46.37	0.00
	0.7	0%	0%	0%	34%	15%	6%	39%	11%	0%	0%	0%	0%	10	0	1	9	1.00	45.53	0.00
	0.9	0%	0%	0%	40%	19%	2%	27%	6%	0%	0%	0%	0%	10	0	2	8	1.00	43.45	0.00
	Avg.	0%	0%	0%	30%	14%	2%	18%	5%	0%	2%	0%	0%	-	-	-	-	227.31	31.56	0.00

Table 5.5: Performance of MILP, m-ATCS, ISFAN and the TS algorithm for n=15

		% Deviations from UB												# of optimal solutions				Run Times		
$n = 15$		MILP			m-ATCS			ISFAN			TS									
τ	R	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	MILP	m-ATCS	ISFAN	TS	MILP	ISFAN	TS
0.1	0.1	13%	8%	1%	17%	10%	4%	15%	8%	1%	4%	3%	1%	0	0	0	0	3600.00	37.67	0.04
	0.3	17%	7%	1%	23%	16%	8%	12%	6%	2%	8%	3%	0%	0	0	0	1	3600.00	31.26	0.01
	0.5	10%	6%	1%	18%	11%	4%	8%	4%	2%	4%	2%	0%	0	0	0	1	3600.00	25.47	0.01
	0.7	21%	7%	1%	29%	11%	3%	9%	4%	0%	4%	2%	0%	0	0	1	1	3600.00	23.02	0.01
	0.9	15%	6%	0%	28%	10%	0%	11%	5%	0%	6%	1%	0%	1	1	3	4	3241.70	19.62	0.01
0.3	0.1	13%	10%	4%	24%	15%	7%	8%	6%	3%	8%	4%	3%	0	0	0	0	3600.00	44.64	0.01
	0.3	22%	11%	4%	41%	19%	6%	19%	9%	4%	11%	5%	2%	0	0	0	0	3600.00	41.25	0.01
	0.5	22%	11%	4%	29%	19%	9%	12%	7%	5%	8%	5%	1%	0	0	0	0	3600.00	34.50	0.01
	0.7	20%	11%	3%	36%	20%	6%	34%	11%	3%	8%	4%	2%	0	0	0	0	3600.00	32.61	0.01
	0.9	16%	9%	0%	31%	18%	3%	13%	6%	0%	7%	4%	0%	1	0	1	2	3444.50	30.32	0.01
0.5	0.1	18%	12%	2%	27%	20%	12%	19%	11%	2%	13%	7%	2%	0	0	0	0	3600.00	55.18	0.01
	0.3	15%	11%	7%	33%	22%	8%	18%	10%	4%	14%	8%	4%	0	0	0	0	3600.00	51.61	0.01
	0.5	26%	15%	7%	35%	26%	10%	20%	12%	4%	15%	9%	6%	0	0	0	0	3600.00	45.57	0.01
	0.7	12%	7%	0%	44%	26%	11%	25%	9%	4%	11%	6%	1%	0	0	0	0	3600.00	39.85	0.01
	0.9	18%	7%	0%	40%	21%	7%	21%	8%	0%	18%	6%	0%	1	0	1	1	3369.80	41.74	0.01
0.7	0.1	0%	0%	0%	26%	15%	0%	7%	3%	0%	4%	1%	0%	10	1	1	7	83.90	55.74	0.01
	0.3	0%	0%	0%	30%	14%	0%	5%	3%	0%	3%	1%	0%	10	1	1	7	88.60	58.06	0.01
	0.5	0%	0%	0%	44%	20%	12%	12%	4%	0%	2%	0%	0%	10	0	1	7	101.30	59.85	0.01
	0.7	8%	1%	0%	33%	19%	8%	20%	7%	0%	8%	2%	0%	9	0	2	3	883.90	64.45	0.01
	0.9	0%	0%	0%	29%	17%	6%	16%	7%	0%	0%	0%	0%	10	0	1	9	288.50	53.13	0.01
0.9	0.1	0%	0%	0%	27%	10%	0%	14%	4%	0%	3%	0%	0%	10	1	5	9	1.00	116.49	0.01
	0.3	0%	0%	0%	35%	17%	8%	10%	4%	0%	5%	0%	0%	10	0	3	9	1.00	87.96	0.01
	0.5	0%	0%	0%	24%	17%	2%	12%	4%	0%	0%	0%	0%	10	0	4	10	6.10	74.97	0.01
	0.7	0%	0%	0%	25%	16%	3%	33%	9%	0%	0%	0%	0%	10	0	2	10	4.40	69.81	0.01
	0.9	0%	0%	0%	39%	21%	3%	28%	9%	0%	0%	0%	0%	10	0	2	10	12.40	69.23	0.01
	Avg.	11%	6%	1%	31%	17%	6 %	16%	7%	1%	7%	3%	1%	-	-	-	-	2189.08	50.56	0.01

Table 5.6: Performance of MILP, m-ATCS, ISFAN and the TS algorithm for n=20

		% Deviations from UB												# of optimal solutions				Run Times		
$n = 20$		MILP			m-ATCS			ISFAN			TS									
τ	R	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	MILP	m-ATCS	ISFAN	TS	MILP	ISFAN	TS
0.1	0.1	19%	11%	6%	17%	11%	6%	10%	7%	2%	5%	3%	1%	0	0	0	0	3600.00	56.87	0.10
	0.3	19%	12%	2%	30%	17%	5%	13%	9%	3%	5%	3%	1%	0	0	0	0	3600.00	54.07	0.04
	0.5	22%	10%	3%	23%	17%	4%	8%	5%	1%	3%	2%	1%	0	0	0	0	3600.00	40.73	0.03
	0.7	23%	11%	0%	23%	12%	6%	10%	5%	0%	2%	1%	0%	1	0	0	5	3415.50	35.16	0.04
	0.9	15%	7%	1%	24%	10%	0%	17%	6%	0%	4%	1%	0%	0	2	1	3	3600.00	32.22	0.03
0.3	0.1	29%	18%	7%	28%	20%	9%	18%	11%	7%	8%	5%	2%	0	0	0	0	3600.00	69.44	0.03
	0.3	21%	14%	4%	27%	20%	12%	15%	9%	5%	6%	5%	3%	0	0	0	0	3600.00	63.65	0.04
	0.5	25%	16%	7%	26%	17%	9%	12%	8%	5%	8%	5%	2%	0	0	0	0	3600.00	59.10	0.03
	0.7	25%	12%	5%	31%	17%	5%	12%	7%	2%	7%	3%	1%	0	0	0	0	3600.00	55.62	0.04
	0.9	20%	10%	2%	28%	18%	5%	13%	7%	1%	5%	2%	0%	0	0	0	2	3444.50	53.07	0.04
0.5	0.1	30%	13%	6%	26%	20%	12%	13%	10%	7%	8%	6%	4%	0	0	0	0	3600.00	79.69	0.03
	0.3	26%	17%	7%	63%	26%	11%	18%	12%	5%	9%	6%	3%	0	0	0	0	3600.00	77.91	0.07
	0.5	23%	16%	9%	37%	27%	19%	17%	13%	9%	10%	7%	3%	0	0	0	0	3600.00	75.72	0.04
	0.7	27%	14%	8%	35%	23%	13%	17%	12%	7%	11%	5%	1%	0	0	0	0	3600.00	70.42	0.03
	0.9	21%	12%	5%	38%	26%	10%	23%	14%	5%	11%	6%	0%	0	0	0	1	3600.00	73.10	0.05
0.7	0.1	26%	12%	6%	30%	22%	17%	20%	14%	8%	15%	10%	5%	0	0	0	0	3600.00	92.53	0.05
	0.3	14%	10%	6%	33%	24%	10%	18%	13%	10%	14%	9%	6%	0	0	0	0	3600.00	97.47	0.04
	0.5	27%	10%	0%	47%	35%	19%	24%	14%	5%	20%	9%	0%	2	0	0	2	3048.20	105.50	0.04
	0.7	20%	8%	0%	40%	25%	9%	24%	13%	0%	12%	7%	0%	3	0	1	3	2954.40	100.07	0.07
	0.9	17%	10%	0%	44%	29%	7%	34%	17%	8%	13%	8%	0%	1	0	0	1	3242.80	104.19	0.04
0.9	0.1	0%	0%	0%	18%	10%	1%	8%	4%	0%	1%	0%	0%	10	0	3	0	1.70	138.66	0.06
	0.3	0%	0%	0%	26%	15%	7%	24%	7%	0%	1%	0%	0%	10	0	1	8	8.70	144.22	0.03
	0.5	0%	0%	0%	43%	20%	8%	24%	8%	1%	2%	0%	0%	10	0	0	8	80.10	127.94	0.03
	0.7	0%	0%	0%	46%	21%	11%	13%	6%	0%	1%	0%	0%	10	0	3	8	114.00	119.66	0.03
	0.9	13%	1%	0%	39%	27%	12%	18%	12%	0%	13%	2%	0%	9	0	1	6	467.00	118.01	0.03
	Avg.	18%	10%	3%	33%	20%	9%	17%	10%	4%	8%	4%	1%	-	-	-	-	2831.07	81.80	0.04

Table 5.7: Performance of MILP, m-ATCS, ISFAN and the TS algorithm for n=25

		% Deviations from UB																		
$n = 25$		MILP			m-ATCS			ISFAN			TS			# of optimal solutions				Run Times		
τ	R	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	MILP	m-ATCS	ISFAN	TS	MILP	ISFAN	TS
0.1	0.1	19%	12%	8%	20%	14%	9%	12%	8%	4%	6%	4%	1%	0	0	0	0	3600.00	83,62	0.08
	0.3	23%	13%	6%	32%	16%	6%	11%	7%	3%	6%	3%	2%	0	0	0	0	3600.00	65.37	0.06
	0.5	9%	7%	5%	21%	12%	3%	11%	7%	3%	4%	2%	1%	0	0	0	0	3600.00	61.45	0.06
	0.7	11%	5%	1%	23%	13%	5%	9%	5%	2%	4%	1%	0%	0	0	0	4	3600.00	53.97	0.06
	0.9	10%	4%	1%	24%	8%	2%	10%	6%	1%	2%	1%	0%	0	0	0	4	3600.00	56.69	0.05
0.3	0.1	24%	15%	9%	29%	19%	13%	12%	9%	5%	5%	4%	2%	0	0	0	0	3600.00	93.23	0.10
	0.3	20%	11%	7%	24%	18%	7%	12%	10%	7%	7%	5%	3%	0	0	0	0	3600.00	89.71	0.09
	0.5	8%	5%	2%	26%	18%	10%	10%	7%	5%	6%	3%	2%	0	0	0	0	3600.00	80.76	0.08
	0.7	9%	5%	1%	32%	17%	8%	24%	11%	6%	6%	2%	1%	0	0	0	0	3600.00	78.54	0.08
	0.9	27%	18%	11%	27%	16%	6%	18%	9%	5%	4%	2%	0%	0	0	0	2	3600.00	79.35	0.07
0.5	0.1	17%	10%	6%	29%	19%	12%	14%	11 %	7%	7%	6%	3%	0	0	0	0	3600.00	110.84	0.07
	0.3	13%	9%	4%	29%	21%	11%	17%	11%	6%	9%	5%	3%	0	0	0	0	3600.00	109.09	0.08
	0.5	17%	10%	3%	30%	20%	8%	20%	13%	8%	8%	5%	2%	0	0	0	0	3600.00	110.30	0.09
	0.7	13%	8%	2%	38%	26%	17%	17%	13%	7%	11%	6%	2%	0	0	0	0	3600.00	121.02	0.08
	0.9	42%	20%	6%	49%	28%	16%	25%	15%	6%	7%	4%	1%	0	0	0	0	3600.00	109.74	0.08
0.7	0.1	19%	10%	2%	26%	20%	12%	27%	18%	10%	18%	9%	3%	0	0	0	0	3600.00	142.09	0.06
	0.3	17%	12%	8%	33%	26%	18%	23%	17%	9%	14%	10%	7%	0	0	0	0	3600.00	141.36	0.08
	0.5	21%	13%	7%	43%	33%	26%	26%	19%	13%	15%	12%	7%	0	0	0	0	3600.00	143.57	0.08
	0.7	15%	9%	2%	40%	29%	15%	21%	15%	9%	14%	8%	2%	0	0	0	0	3600.00	140.45	0.07
	0.9	17%	10%	0%	53%	29%	8%	24%	17%	10%	15%	10%	3%	0	0	0	0	3600.00	152.11	0.08
0.9	0.1	0%	0%	0%	20%	12%	4%	25%	8%	0%	6%	1%	0%	10	0	1	5	5.41	188.99	0.06
	0.3	0%	0%	0%	22%	17%	10%	8%	5%	1%	0%	0%	0%	10	0	0	8	8.30	187.45	0.06
	0.5	12%	3%	0%	45%	27%	14%	44%	12%	1%	12%	4%	0%	6	0	0	3	1452.61	176.50	0.07
	0.7	22%	7%	0%	45%	29%	17%	31%	13%	0%	25%	8%	0%	6	0	1	4	1446.35	177.66	0.08
	0.9	21%	6%	0%	58%	30%	20%	32%	13%	2%	22%	7%	0%	4	0	0	4	2176.18	169.41	0.09
	Avg.	16%	9%	4%	33%	21%	11%	19%	11%	5%	9%	5%	2%	-	-	-	-	3083.55	116.93	0.07

Table 5.8: Performance of MILP, m-ATCS, ISFAN and the TS algorithm for n=50

		% Deviations from UB																		
$n = 50$		MILP			m-ATCS			ISFAN			TS			# of optimal solutions				Run Times		
τ	R	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	MILP	m-ATCS	ISFAN	TS	MILP	ISFAN	TS
0.1	0.1	81%	24%	9%	20%	15%	9%	12%	8%	6%	3%	2%	1%	0	0	0	0	3600.00	264.38	1.19
	0.3	22%	16%	7%	18%	15%	9%	12%	9%	7%	4%	2%	1%	0	0	0	0	3600.00	250.16	0.98
	0.5	25%	17%	11%	19%	13%	6%	10%	8%	5%	2%	2%	1%	0	0	0	0	3600.00	238.33	0.89
	0.7	18%	13%	9%	24%	14%	7%	22%	9%	4%	16%	3%	0%	0	0	0	1	3600.00	220.68	0.58
	0.9	27%	10%	2%	10%	4%	1%	11%	7%	4%	2%	1%	0%	0	0	0	2	3600.00	288.67	0.46
0.3	0.1	33%	24%	13%	24%	16%	11%	13%	11%	8%	3%	3%	2%	0	0	0	0	3600.00	338.06	1.61
	0.3	48%	28%	18%	30%	20%	11%	15%	12%	9%	5%	4%	3%	0	0	0	0	3600.00	343.90	1.49
	0.5	33%	24%	18%	33%	18%	9%	15%	11%	8%	5%	3%	1%	0	0	0	0	3600.00	328.47	1.16
	0.7	26%	20%	13%	24%	14%	5%	18%	11%	7%	3%	1%	0%	0	0	0	1	3600.00	346.78	0.85
	0.9	34%	21%	13%	26%	17%	3%	11%	9%	8%	3%	1%	0%	0	0	0	0	3600.00	345.54	0.68
0.5	0.1	31%	28%	19%	27%	20%	13%	19%	14%	10%	5%	4%	3%	0	0	0	0	3600.00	430.80	1.36
	0.3	86%	42%	25%	30%	25%	18%	20%	16%	11%	8%	6%	3%	0	0	0	0	3600.00	448.20	1.18
	0.5	44%	31%	17%	34%	25%	15%	21%	16%	7%	8%	4%	2%	0	0	0	0	3600.00	438.71	1.35
	0.7	38%	21%	10%	31%	24%	12%	18%	15%	9%	5%	3%	2%	0	0	0	0	3600.00	471.10	1.22
	0.9	22%	18%	8%	35%	27%	17%	19%	15%	6%	6%	4%	2%	0	0	0	0	3600.00	495.41	1.16
0.7	0.1	25%	21%	13%	30%	22%	17%	21%	16%	12%	9%	7%	4%	0	0	0	0	3600.00	561.75	1.44
	0.3	23%	18%	11%	33%	24%	16%	20%	16%	11%	11%	7%	4%	0	0	0	0	3600.00	565.54	1.52
	0.5	52%	29%	13%	46%	33%	23%	22%	19%	14%	13%	9%	7%	0	0	0	0	3600.00	582.52	1.59
	0.7	28%	21%	9%	35%	31%	24%	29%	19%	12%	18%	9%	2%	0	0	0	0	3600.00	609.67	1.27
	0.9	26%	19%	14%	41%	29%	19%	28%	21%	17%	18%	11%	6%	0	0	0	0	3600.00	617.63	1.15
0.9	0.1	18%	14%	8%	30%	24%	17%	22%	17%	12%	18%	13%	8%	0	0	0	0	3600.00	684.75	1.25
	0.3	22%	16%	11%	42%	32%	21%	28%	23%	18%	23%	17%	13%	0	0	0	0	3600.00	686.89	1.26
	0.5	21%	15%	5%	45%	39%	33%	26%	21%	18%	23%	17%	10%	0	0	0	0	3600.00	672.19	1.42
	0.7	25%	17%	9%	46%	37%	26%	72%	27%	17%	21%	16%	11%	0	0	0	0	3600.00	780.25	1.43
	0.9	30%	21%	14%	44%	36%	24%	29%	22%	16%	19%	16%	11%	0	0	0	0	3600.00	673.72	1.37
	Avg.	34%	21%	12%	31%	23%	15%	21%	15%	10%	10%	7%	5%	-	-	-	-	3600.00	467.36	1.19

Table 5.9: Performance of MILP, m-ATCS, ISFAN and the TS algorithm for n=100

		% Deviations from UB																		
$n = 100$		MILP			m-ATCS			ISFAN			TS			# of optimal solutions				Run Times		
τ	R	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	MILP	m-ATCS	ISFAN	TS	MILP	ISFAN	TS
0.1	0.1	56%	44%	37%	19%	15%	9%	13%	9%	8%	3%	2%	1%	0	0	0	0	3600.00	682.82	16.76
	0.3	88%	51%	42%	20%	17%	11%	10%	9%	7%	3%	2%	2%	0	0	0	0	3600.00	666.81	17.26
	0.5	60%	51%	43%	19%	14%	11%	11%	9%	6%	3%	1%	0%	0	0	0	0	3600.00	690.50	10.87
	0.7	60%	54%	46%	16%	11%	5%	12%	9%	6%	1%	0%	0%	0	0	0	1	3600.00	701.30	6.21
	0.9	70%	60 %	38%	11%	6%	0%	16%	12%	8%	1%	0%	0%	0	0	0	4	3600.00	744.60	3.53
0.3	0.1	68%	62%	52%	24%	19%	15%	13%	12%	10%	4%	3%	1%	0	0	0	0	3600.00	941,03	22.37
	0.3	75%	64%	52%	23%	18%	15%	17%	13%	11%	5%	3%	2%	0	0	0	0	3600.00	964.58	20.10
	0.5	76%	66%	56%	21%	18%	14%	17%	14%	10%	4%	2%	1%	0	0	0	0	3600.00	988.50	16.77
	0.7	84%	73%	63%	32%	20%	15%	15%	14%	12%	3%	2%	0%	0	0	0	0	3600.00	1028.52	9.48
	0.9	82%	66%	47%	18%	15%	10%	17%	13%	9%	2%	1%	0%	0	0	0	0	3600.00	1020.23	7.58
0.5	0.1	87%	77%	71%	24%	22%	17%	18%	16%	12%	5%	4%	2%	0	0	0	0	3600.00	1243.25	25.96
	0.3	86%	61%	47%	28%	23%	17%	18%	15%	12%	6%	4%	3%	0	0	0	0	3600.00	1288.58	28.87
	0.5	88%	70%	56%	31%	24%	20%	19%	17%	14%	5%	4%	3%	0	0	0	0	3600.00	1305.33	20.56
	0.7	92%	71%	55%	34%	23%	10%	21%	17%	13%	4%	3%	2%	0	0	0	0	3600.00	1385.12	15.57
	0.9	100%	66%	48%	36%	24%	15%	24%	18%	12%	5%	2%	1%	0	0	0	0	3600.00	1419.77	12.15
0.7	0.1	86%	67%	49%	28%	22%	17%	19%	17%	13%	6%	5%	3%	0	0	0	0	3600.00	1646.30	33.60
	0.3	66%	57%	45%	34%	24%	18%	21%	17%	13%	11%	7%	4%	0	0	0	0	3600.00	1659.33	26.62
	0.5	65%	55%	46%	37%	26%	20%	24%	18%	14%	13%	6%	4%	0	0	0	0	3600.00	1660.40	22.30
	0.7	76%	60%	42%	38%	31%	22%	23%	19%	16%	13%	7%	3%	0	0	0	0	3600.00	1690.41	26.36
	0.9	64%	53%	39%	37%	31%	23%	24%	18%	15%	13%	8%	5%	0	0	0	0	3600.00	1657.41	17.84
0.9	0.1	48%	37%	31%	25%	22%	18%	20%	17%	14%	12%	9%	7%	0	0	0	0	3600.00	1643.63	29.46
	0.3	50%	40%	28%	36%	31%	25%	26%	20%	16%	23%	15%	9%	0	0	0	0	3600.00	1807.60	26.32
	0.5	54%	38%	23%	39%	33%	25%	25%	21%	19%	18%	16%	13%	0	0	0	0	3600.00	1802.09	21.51
	0.7	48%	38%	27%	40%	36%	26%	24%	21%	15%	20%	16%	11%	0	0	0	0	3600.00	1798.96	22.70
	0.9	39%	35%	27%	40%	35%	28%	28%	21%	13%	22%	16%	12%	0	0	0	0	3600.00	1765.10	17.48
	Avg.	71%	57%	44%	28%	22%	16%	19%	15%	12%	8%	6%	4%	-	-	-	-	3600.00	1288.09	19.13

5.4 Analysis of the Results

The results presented in Tables 5.4 - 5.9 suggest that the TS algorithm is both efficient and effective for all problem sizes and types tested. The detailed analysis is as follows.

Comparison of the TS algorithm with MILP

Throughout the tables, we observe that $n = 10$ is the only case where MILP can solve all 250 but 1 instances to optimality. Although TS finds the optimal solution in 188 out of 250 instances in Table 5.4, it gives 0% deviation on the average, as it outputs solutions whose revenues are uniformly very close to optimal values. This performance is achieved with a run time of less than a second in all instances, whereas MILP runs in 228 seconds on the average and reaches the limit of one hour in one of the instances.

From Table 5.5, we see that the TS algorithm finds 91 optimal solutions out of 250 instances whereas MILP finds the optimal solution in 102 out of 250 instances. MILP gives 6% deviation on the average, the TS algorithm gives only 3% deviation on the average. Although MILP finds more optimal solutions than TS, it gives a higher percentage deviation than TS. This shows that TS algorithm find near-optimal solutions for most of the instances whereas MILP finds optimal solutions for some instances and far-optimal solutions for the rest of the instances. The average run time of MILP increases to 2189 seconds while the TS algorithm runs in still less than a second in all test instances for $n = 15$.

When the problem size increases to 20, the TS algorithm still outperforms MILP in terms of percentage deviation. The TS algorithm finds the optimal solution in 51 out of 250 instances and gives 4% deviation on the average. In contrast, the MILP finds the optimal solution in 55 out of 250 instances but gives 10% deviation on the average. It is notable that that the number of optimal solutions obtained by TS catching up the MILP as n increases and the average percentage deviation of MILP increases much faster than that of TS. The average run time of MILP increases to 2831 seconds while the TS algorithm runs in less than a second in all test instances.

For $n = 25$, the TS algorithm still runs in less than a second for all test instances whereas the run time of MILP increases to 3084 seconds on the average. TS finds 34 optimal solutions out of 250 instances whereas MILP finds the optimal solution in 36 out of 250 instances. MILP gives 9% deviation on the average, the TS algorithm gives only 5% deviation on the average.

When there is 50 incoming orders, although MILP cannot find any optimal solutions out of 250 instances, TS achieves to find the optimal solution in 4 out of 250 instances and gives 6% deviation on the average whereas MILP gives 23% deviation on the average. The average run time of MILP increases to 3600 seconds since the algorithm could not achieve to solve any of the instances while the TS algorithm runs in less than two seconds on the average.

From Table 5.9, we see that the TS algorithm finds 5 optimal solutions out of 250 instances whereas MILP still cannot find any optimal solution out of 250 instances. MILP gives 22% deviation on the average, the TS algorithm gives only 6% deviation on the average. The average run time of MILP is again 3600 seconds while the TS algorithm runs less than twenty second on the average.

We observe from Table 5.5 that MILP has difficulty in solving instances with $\tau = 0.1, 0.3, 0.5$ for $n = 15$. MILP faces difficulties in solving instances with $\tau = 0.7$ in addition to 0.1, 0.3 and 0.5 when $n = 20, 25$ (see Tables 5.6-5.7) and can not solve any of the instances when $n = 50, 100$ (see Tables 5.8-5.9). These results suggest that the problem is getting fairly easier for higher τ values and therefore becomes less time consuming for the exact algorithm. We discuss the difficulty of test instances in fortcoming subsections.

The results presented above indicate that the TS algorithm is competitive with the MILP for $n = 10$, and outperforms the MILP results when n equals 15, 20, 25, 50 and 100. Run times presented in Tables 5.4-5.9 further support the efficiency of the TS algorithm.

Comparison of the TS algorithm with m-ATCS and ISFAN

We see from Tables 5.4-5.9 that the TS algorithm outperforms m-ATCS and ISFAN heuristics in terms of solution quality and in terms of number of optimal solutions obtained out of 10 instances in all instances. While TS algorithm dominates the ISFAN heuristic in terms of run times for all problem types and sizes, m-ATCS heuristic runs less than a second for all test instances as a property of constructive heuristic.

m-ATCS gives 14% and ISFAN 5% deviations on the average whereas TS achieves 0% deviation on the average when problem size equals to 10. For $n = 15$, m-ATCS and ISFAN give 17% and 7% deviations on the average respectively and are dominated by TS which gives 3% deviation on the average. As n increases to 20, the percentage deviation of the m-ATCS also increases to 20. ISFAN gives 10% deviation and TS gives only 4% deviation

on the average. Respectively, m-ATCS, ISFAN and TS give 21%, 11% and 5% deviations on the average for $n = 25$, 23%, 15% and 6% deviations on the average for $n = 50$ and finally 22%, 15% and 6% deviations on the average for $n = 100$.

As n changes, the average performances of m-ATCS and ISFAN range from 14% to 23% and 5% to 15%, respectively, while that of the TS algorithm ranges from 0% to 6%. These ranges indicate that our TS algorithm is robust and has a smaller variation in solution quality comparing to other heuristic methods. Since the run time of the TS algorithm is less than 20 seconds on the average even for $n = 100$, we can conclude that the TS algorithm is a viable solution procedure for practical situations.

Effect of parameters τ and R on test instances

By definition, the parameter τ indicates due date tightness while the parameter R determines the due date range. In addition, τ determines the range of the release dates and R dictates the tightness of deadlines with respect to due dates.

As τ gets larger, the interval on which the release dates were drawn also gets larger. In other words, the orders are released on a wider time interval. At the same time, the values of *slack* get smaller. Hence, as τ increases, we obtain orders released at different time points with small *slack* values. This results in rejection of more orders, which in turn implies the reduction of the size of the solution space. Consequently, the optimal solution can be found in a short time as can be seen in Tables 5.4-5.7.

Given a τ value, the difference between the deadline and the due date increases, as R increases, which allows more time to complete a tardy order. Furthermore, the range of *slack* increases, while the release dates are generated in a constant interval. However, these data characteristics do not suggest any distinctive patterns on problem hardness.

Analysis of the upper bounds

As we mention in Section 5.3.3, we calculated two different upper bounds which are UB_{MILP} and UB_{LPVI}, refer minimum of these two bounds to UB and used it to measure the solution quality of the benchmarks. We observed that UB_{LPVI} is more effective for τ=0.1, 0.3, 0.5, whereas UB_{MILP} becomes more effective for τ=0.7, 0.9. The reasons of why UB_{MILP} can found better bounds for smaller τ values and UB_{LPVI} can found better bounds for large τ values can be explained as follows. We mention that the problems get easier for MILP when τ increases, therefore UB_{MILP} can found better bounds and even

optimal solutions for most of the cases of larger τ values. On the other hand, the valid inequality in Equation 5.2, which is the main effective valid inequality among Equations 5.1-5.3, accumulates the processing times and minimum setup times of the accepted orders. For small values of τ, we obtain closely released orders which have closer slack times. When we schedule the orders the idle time for a machine can not arise in most of the cases, therefore summing the processing and minimum setup times of all accepted orders can help to improve the upper bound for smaller τ values. Whereas, for larger τ values we obtain orders released on wider time interval which have smaller slack times, therefore in some cases, the machine can be idle and can wait for the next order to be released to continue the production process. Hereby, only summing the processing times and setup times of accepted orders can not help for larger τ values to improve the UB. As can be seen from Tables 5.4-5.7, in most of the cases we obtain the highest gap when τ=0.5 and 0.7. Because these are the cases which cannot benefited neither from UB_{MILP} nor from UB_{LPVI}.

It is notable that when $n = 50, 100$, which are the cases that MILP can not solve any of the instances to optimality and therefore UB cannot benefited from UB_{MILP} anymore, the highest percentage deviations was obtained in case of $\tau = 0.9$. We can infer that these higher percentage deviations are the results of poor upper bounds. In order to reinforce this inference, we compare the percentage deviation of TS calculated with respect to these two upper bounds for the instances with $n = 25$ and $\tau = 0.9$. The results are summarized in Table 5.10.

Table 5.10: Average % deviations of TS heuristic from UB_{LPVI}, UB_{MILP} and UB where n=25 and $\tau = 0.9$

			Dev. of TS from UB_{LPVI}			Dev. of TS from UB_{MILP}			Dev. of TS from UB		
n	τ	R	Max	Average	Min	Max	Average	Min	Max	Average	Min
25	0.9	0.1	27%	20%	11%	6%	1%	0%	6%	1%	0%
		0.3	28%	22%	17%	0%	0%	0%	0%	0%	0%
		0.5	24%	17%	11%	12%	4%	0%	12%	4%	0%
		0.7	26%	21%	13%	25%	8%	0%	25%	8%	0%
		0.9	27%	19%	11%	22%	7%	0%	22%	7%	0%
	Average		26%	20%	13%	13%	4%	0%	13%	4%	0%

We can see that for this case, the average percentage deviation of the TS algorithm drops to 4% from 20% when the UB is benefited from UB_{MILP}. Therefore the results in Table

5.10 asserts that the UB is weak where it can not benefited any of UB_{MILP} and UB_{LPVI}. This also verifies that the TS algorithm performs well.

Comparison of the number of accepted and tardy orders among solution methods: In order to understand the structure of the solutions generated by the compared heuristics, we analyzed the number of rejected and tardy orders and present the results in Table 5.11 for only several contrasting cases.

We notice that both the m-ATCS and ISFAN heuristics reject more orders and accepts more tardy orders than the TS algorithm over all instances. This can be attributed to the constructive nature of the m-ATCS heuristic and the separately consideration of acceptance and rejection decisions in ISFAN heuristic. In contrast, the TS algorithm considers acceptance and scheduling decisions simultaneously, hence rejects the tardy orders if there are more profitable orders. As a consequence, the TS algorithm can achieve both rejecting less orders and accepting less tardy orders compared to other heuristics.

When we analyze the number of rejected orders for different cases in Table 5.11, we see that for constant τ value as n increases, more orders are rejected but the results do not suggest any distinctive pattern on tardy orders. On the other hand, significantly more orders are rejected and also more tardy orders are accepted as τ increases for a constant value of n.

The latter one also helps to explain why UB gets weaker for higher τ values. Note that the LP relaxation solution accepts most of the orders since it relaxes the capacity constraint. As a result, the upper bound gets weaker when the number of orders to be rejected is high due to the overload at certain time intervals.

Table 5.11: Number of rejected and tardy orders of m-ATCS, ISFAN and TS heuristics for n=25 with $\tau = 0.1$ and $\tau = 0.9$, and for n=50 with $\tau = 0.9$

			# of rejected orders									# of tardy orders								
			m-ATCS			ISFAN			TS			m-ATCS			ISFAN			TS		
n	τ	R	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min
25	0.1	0.1	8	6	5	7	6	5	6	4	3	0	0	0	0	0	0	0	0	0
		0.3	6	5	4	4	4	3	4	3	2	3	1	0	4	1	0	0	0	0
		0.5	7	4	2	4	3	2	3	2	1	2	1	0	1	0	0	0	0	0
		0.7	5	3	2	4	3	1	3	1	0	3	2	0	4	1	0	0	0	0
		0.9	5	2	0	4	2	1	2	1	0	3	2	0	3	1	0	2	0	0
	Average		6	4	3	5	4	2	4	2	1	2	1	0	2	1	0	0	0	0
25	0.9	0.1	12	10	7	12	10	8	10	9	7	2	1	0	1	0	0	1	0	0
		0.3	14	10	8	12	10	6	11	9	6	4	2	1	3	1	0	2	1	0
		0.5	12	10	8	15	9	7	10	8	6	5	3	1	2	1	0	1	1	0
		0.7	12	11	9	12	9	7	10	8	6	6	3	0	3	1	0	2	1	0
		0.9	14	9	6	12	8	5	9	6	5	7	5	3	4	2	1	3	1	0
	Average		13	10	8	13	9	7	10	8	6	5	3	1	3	1	0	2	1	0
50	0.9	0.1	19	16	13	19	17	15	16	13	10	2	1	0	0	0	0	0	0	0
		0.3	24	20	16	22	19	16	17	15	13	7	4	0	2	1	0	0	0	0
		0.5	23	20	14	19	17	14	16	13	10	6	5	4	2	1	0	0	0	0
		0.7	21	19	16	39	18	14	15	12	11	10	7	4	3	1	0	0	0	0
		0.9	23	17	13	20	16	13	14	12	10	14	10	4	7	2	0	0	0	0
	Average		22	18	14	24	20	14	16	13	11	8	5	2	3	1	0	0	0	0

Chapter 6

CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

6.1 Conclusions

In this thesis, we consider an order acceptance and scheduling problem on a single machine. The differentiating aspects of the problem are the sequence-dependent setup times, release dates, due dates and deadlines regarding orders and inclusion of tardiness while computing the revenue in the objective function. The OAS problem is strongly NP-hard.

We provide a competitive improvement heuristic for the order acceptance and scheduling problem (OAS) in a single machine environment. The proposed improvement heuristic is a tabu search algorithm which is supported with a probabilistic local search procedure after each iteration. We provide a new data set for the OAS problem that can be used in further studies. In this data set, we used parameters such as the tardiness factor and the due date range to obtain different types of instances resulting in varying difficulty.

We compare the performance of the TS algorithm with that of two heuristic algorithms, namely m-ATCS and ISFAN, as well as with the solutions found by a mixed integer programming formulation. Since optimal solutions cannot be obtained for large instances, we used upper bounds based on the mixed integer programming formulation to measure the quality of the solutions. Our computational study shows that the TS algorithm gives significantly better solutions compared to other solution procedures in terms of revenue in all instances. Furthermore, the run time of the TS algorithm is very small even for large instances.

The success of the TS algorithm may be attributed to the following factors. First, the problem representation makes it possible to consider both acceptance and sequencing decisions simultaneously in a compact form and to create a wider neighborhood by efficient move operators. As a result, the solution space can be searched extensively without heavy computational effort. The seemingly simple pairwise exchange move of the TS algorithm allows us to change both the set of accepted orders and their sequence, while keeping the

number of accepted orders the same. We note that when we swap two orders, the completion time, hence the tardiness and the revenue of each order starting from the first swapped order will be affected due to release dates, sequence-dependent setup times and deadlines. Therefore, a feasibility check is required after each modification to the current solution. Second, the local search procedure involves a compound move in which one drop operation is followed by multiple add and insert operations. Each such move may necessitate deletion of some orders to maintain feasibility and this in turn adds diversification. Applying the compound move iteratively allows accepting a profitable order, which was formerly rejected due to infeasibility, at a new position. While both ISFAN and TS utilize a local improvement procedure at the end of the algorithm, TS guides the search more intelligently by compound moves. In spite of this complexity, TS runs very fast while obtaining much better results in terms of solution quality. Therefore, the proposed algorithm provides a viable solution method for real life problems. As a note, our efforts in developing dominance properties and new valid inequalities for the formulation did not result in stronger bounds confirming the computational complexity of the problem.

6.2 Future Research

As mentioned in previous section, we develop an efficient TS algorithm to solve the OAS problem.

6.2.1 Strengthening Upper Bounds

In Section 5.4, we indicated that the upper bounds which we tested the performance of the TS algorithm are weak for some cases. Therefore, the upper bounds may be strengthened as a further study. One approach for strengthening the upper bounds may be proposing a position based model formulation for the OAS problem. Solving the relaxation of new model or solving the test instances within 3600 seconds time limit with new model might be results in better upper bounds. Strengthening the existing MILP model with new valid inequalities may be another approach for improving the upper bounds.

6.2.2 Exact Method Development

A branch and bound algorithm might be developed for the OAS problem. Although we stated that the OAS problem is strongly NP-hard, thus an exact algorithm may be very time consuming for solving the problem, a well developed branch and bound algorithm can be promising for solving larger instances to optimality and may even promising for improving the upper bounds.

BIBLIOGRAPHY

[1] T.S. Abdul-Razaq and L.N. van Wassenhove. A survey of algorithms for the single-machine total weighted tardiness scheduling problem. *Discrete Applied Mathematics*, 26:235–253, 1990.

[2] C. Akkan. Finite-capacity scheduling-based planning for revenue-based capacity management. *European Journal of Operational Research*, 100:170–179, 1997.

[3] M.S. Aktürk and D. Özdemir. An exact approach to minimizing total weighted tardiness with release dates. *IIE Transactions*, 32:1091–1101, 2000.

[4] M.S. Aktürk and D. Özdemir. A new dominance rule to minimize total weighted tardiness with unequal release dates. *European Journal of Operational Research*, 135:394–412, 2001.

[5] A. Allahverdi, J.N.D. Gupta, and T. Aldowaisan. A review of scheduling research involving setup considerations. *Omega, International Journal of Management Science*, 27:219–239, 1999.

[6] A. Allahverdi, C.T. Ng, T.C.E. Cheng, and M.Y. Kovalyov. A survey of scheduling problems with setup times or costs. *European Journal of Operations Research*, 187:985–1032, 2008.

[7] A.B.C. Altunc and A.B. Keha. Interval-indexed formulation based heuristics for single machine total weighted tardiness problem. *Computers and Operations Research*, 36:2122–2131, 2009.

[8] D. Anghinolfi and M.A. Paolucci. A new ant colony optimization approach for the single machine total weighted tardiness scheduling problem. *accepted for publication on International Journal of Operations Research.*

[9] D. Anghinolfi and M.A. Paolucci. A new discrete particle swarm optimization approach for the single machine total weighted tardiness scheduling problem with sequence dependent setup times. *European Journal of Operational Research*, 193:73–85, 2009.

[10] V.A Armanento and R.A. Mazzini. A genetic algorithm for scheduling on a single machine set-up times and due dates. *Production Planning and Control*, 11(7):713–720, 2000.

[11] K.R. Baker and L.R. Schrage. Finding an optimal sequence by dynamic programming:an extension to precedence-related tasks. *Operations Research*, 26:111–120, 1978.

[12] N. Balakrishnan, J. Patterson, and V. Sridharan. Rationing capacity between two product classes. *Decision Sciences*, 27(2):185–214, 1996.

[13] N. Balakrishnan, J. Patterson, and V. Sridharan. An experimental comparison of capacity rationing models. *International Journal of Production Research*, 35(6):1639–1649, 1997.

[14] N. Balakrishnan, J. Patterson, and V. Sridharan. Robustness of capacity rationing policies. *European Journal of Operational Research*, 115:328–338, 1999.

[15] Y Bartal, S. Leonardi, A.M. Spaccamela, J. Sgall, and L. Stougie. Multi-processor scheduling with rejection. *SIAM Journal on Discrete Mathematics*, 13:64–78, 2000.

[16] U. Bilge, F. Kirac, M. Kurtulan, and P. Pekgun. A tabu search algorithm for parallel machine total tardiness problem. *Computers and Operations Research*, 31:397–414, 2003.

[17] U. Bilge, M. Kurtulan, and F. Kirac. A tabu search algorithm for single machine total weighted tardiness problem. *European Journal of Operational Research*, 176:1423–1435, 2007.

[18] W. Bozejko. Parallel path relinking method for the single machine total weighted tardiness problem with sequence-dependent setups. *Journal of Intelligent Manufacturing*, doi 10.1007/s10845-009-0253-2, 2009.

[19] W. Bozejko, J. Grabowski, and M. Wodecki. Block approach-tabu search algorithm for single machine total weighted tardiness problem. *Computers and Industrial Engineering*, 50:1–14, 2006.

[20] T.Y. Chang, F.D. Chou, and C.E. Lee. A heuristic algorithm to minimize total weighted tardiness on a single machine with release dates and sequence-dependent setup times. *Journal of Chinese Institute of Industrial Engineers*, 21(3):289–300, 2004.

[21] K. Charnsirisakskul, P. Griffin, and P. Keskinocak. Order selection and scheduling with leadtime flexibility. *IIE Transactions*, 36:697–707, 2004.

[22] K. Charnsirisakskul, P. Griffin, and P. Keskinocak. Pricing and scheduling decisions with leadtime flexibility. *European Journal of Operational Research*, 171:153169, 2006.

[23] Y.S. Cheng and S.J. Sun. Scheduling linear deteriorating jobs with rejection on a single machine. *European Journal of Operational Research*, 194:18–27, 2009.

[24] F.F. Choobineh, E. Mohebbi, and H. Khoo. A multi-objective tabu search for a single-machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research*, 175:318–337, 2006.

[25] C. Chu. A branch and bound algorithm to minimize total tardiness with unequal release dates. *Naval Research Logistics*, 39:265–283, 1992.

[26] J. Chuzsoy, R. Ostrovsky, and Y. Rabani. Approximation algorithms for the job interval selection problem and related scheduling problems. *IEEE Symposium on Foundations of Computer Science(FOCS)*, pages 348–356, 2001.

[27] V.A. Cicirello. Weighted tardiness scheduling with sequence-dependent setups: a benchmark library. *Technical Report, Intelligent Coordination and Logistics Laboratory, Robotics Institute, Carnegie Mellon University, USA*, 2003.

[28] V.A. Cicirello. Non-wrapping order crossover: An order preserving crossover operator that respects absolute position. *Proceeding of GECCO06 Conference, Seattle, Washington, USA*, pages 1125–1131, 2006.

[29] V.A. Cicirello and S.F. Smith. Enhancing stochastic search performance by value-based randomization of heuristics. *Journal of Heuristics*, 11:5–34, 2005.

[30] R.K. Congram, C.N. Potts, and S.L. Van de Velde. An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem. *Informs Journal on Computing*, 14(1):52–67, 2002.

[31] H.A.J. Crauwels and C.N. Potts. Local search heuristic for the single machine total weighted tardiness scheduling problem. *INFORMS Journal on Computing*, 10(3):341–350, 1998.

[32] P. De, J.B. Ghosh, and C. Wells. Job selection and sequencing on a single machine in a random environment. *European Journal of Operational Research*, 70:425–431, 1993.

[33] G. Dosa and Y. He. Scheduling with machine cost and rejection. *Journal of Combinatorial Optimization*, 12:337–350, 2006.

[34] J. Du and J.Y.T. Leung. Minimizing total tardiness on one machine is NP-hard. *Mathematics of Operations Research*, 15(3):483–495, 1990.

[35] M.J.R. Ebben, E.W. Hans, and F.M. Olde Weghuis. Workload based order acceptance in job shop environments. *OR Spectrum*, 27:107–122, 2005.

[36] H. Emmons. One-machine sequencing to minimize certain functions of job tardiness. *Operations Research*, 17:701–715, 1969.

[37] D.W. Engels, D.R. Karger, S.G. Kolliopoulos, S. Sengupta, R.N. Uma, and J. Wein. Techniques for scheduling with rejection. *Journal of Algorithms*, 49:175–191, 2003.

[38] L. Epstein, J. Noga, and G.J. Woeginger. On-line scheduling of unit time jobs with rejection: Minimizing the total completion time. *Operations Research Letters*, 30:415–420, 2002.

[39] M.L. Fisher. A dual problem for the one machine scheduling problem. *Mathematics Programming*, 11:229–251, 1969.

[40] P.M. Franca, A. Mendes, and P.A. Moscato. A memetic algorithm for the total tardiness single machine scheduling problem. *European Journal of Operational Research*, 132(1):224–242, 2001.

[41] C. Gagne, W. Price, and M. Gravel. Comparing an ACO algorithm with other heuristics for the single machine scheduling problem with sequence-dependent setup times. *Journal of the Operational Research Society*, 53:895–906, 2002.

[42] J.B. Ghosh. Job selection in a heavily loaded shop. *Computers and Operations Research*, 24(2):141–145, 1997.

[43] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13:533–543, 1986.

[44] A. Grosso, F. Della Croce, and R. Tadei. An enhanced dynasearch neighborhood for the single-machine total weighted tardiness scheduling problem. *Operations Research Letters*, 32:68–72, 2004.

[45] H.H. Guerrero and G.M. Kern. How to more effectively accept and refuse orders. *Production and Inventory Management Journal*, 29(4):59–63, 1988.

[46] S.K. Gupta, J. Kyparisis, and C.M. Ip. Project selection and sequencing to maximize net present value of the total return. *Management Science*, 38(5):751–752, 1992.

[47] S.R. Gupta and J.S. Smith. Algorithms for single machine total tardiness scheduling with sequence dependent setups. *European Journal of Operational Research*, 175:722–739, 2006.

[48] J.E. Holsenback, R.M. Russel, R.E. Markland, and P.R. Philipoom. An improved heuristic for the single machine, weighted-tardiness problem. *Omega*, 27(4):485–495, 1999.

[49] H. Hoogeveen, M. Skutella, and G.J. Woeginger. Preemptive scheduling with rejection. *Mathematics Programming*, 94:361–374, 2003.

[50] C. Ivanescu, J.C. Fransoo, and J.W.M. Bertrand. Makespan estimation and order acceptance in batch process industries when processing times are uncertain. *OR Spectrum*, 24:467–495, 2002.

[51] C. Ivanescu, J.C. Fransoo, and J.W.M. Bertrand. A hybrid policy for order acceptance in batch process industries. *OR Spectrum*, 28:199–222, 2006.

[52] R.J.W. James and J.T. Buchanan. Performance enhancements to tabu search for the early/tardy scheduling problem. *European Journal of Operational Research*, 106:254–265, 1998.

[53] A.H.G. Rinnoy Kan. Machine scheduling problems: Classification, complexity and computations. *Nijhoff, The Hague*, 1976.

[54] A.H.G. Rinnoy Kan, B.J. Lageweg, and J.K. Lenstra. Minimizing total cost in one-machine scheduling. *Operations Research*, 23(5):908–927, 1975.

[55] S.Y. Kim, Y.H. Lee, and D. Agnihotri. A hybrid approach to sequencing jobs using heuristic rules and neural networks. *Production Planning Control*, 6(4):445–454, 1995.

[56] B.G. Kingsman. Modelling input-output workload control for dynamic capacity planning in production planning systems. *International Journal of Production Economics*, 68:73–93, 2000.

[57] A.J. Kleywegt and J.D. Papastavrou. The dynamic and stochastic knapsack problem. *Operations Research*, 46(1):17–35, 1998.

[58] A.J. Kleywegt and J.D. Papastavrou. The dynamic and stochastic knapsack problem with random sized items. *Operations Research*, 49(1):26–41, 2001.

[59] C. Koulamas. The total tardiness problem: review and extensions. *Operations Research*, 42:1025–1041, 1994.

[60] C. Koulamas. Polynomially solvable total tardiness problems: Review and extensions. *Omega, International Journal of Management Science*, 25(2):235–239, 1997.

[61] C. Koulamas and G.J. Kyparisis. Single machine scheduling with release times, deadlines and tardiness objectives. *European Journal of Operational Research*, 133:447–453, 2001.

[62] M. Gamache L.-P. Bigras and G. Savard. Time-indexed formulations and the total weighted tardiness problem. *INFORMS Journal on Computing*, 20(1):133–142, 2008.

[63] M. Laguna, J.W. Barnes, and F. Glover. Tabu search methods for single machine scheduling problem. *Journal of Intelligent Mnufacturing*, 2:63–74, 1991.

[64] E.L. Lawler. On scheduling problems with deferral costs. *Management Science*, 11:280–288, 1964.

[65] E.L. Lawler. A pseudo-polynomial algorithm for sequencing jobs to minimize total tardiness. *Annals of Discrete Mathematics*, 1:331–342, 1977.

[66] E.L. Lawler. Efficient implementation of dynamic programming algorithms for sequencing problems. *Report BW 106. Mathematisch Centrum, Amsterdam*, 1979.

[67] Y.H. Lee, K. Bhaskaran, and M. Pinedo. A heuristic to minimize the total weighted tardiness with sequence dependent setups. *IIE Transactions*, 29:45–52, 1997.

[68] Y.H. Lee and M. Pinedo. Scheduling jobs on parallel machines with sequence dependent setup times. *European Journal of Operational Research*, 100:464–474, 1997.

[69] H.F. Lewis and S.A. Slotnick. Multi-period job selection: planning work loads to maximize profit. *Computers and Operations Research*, 29:1081–1098, 2002.

[70] C.J. Liao and H.C Juan. An ant colony optimization for single machine tardiness scheduling with sequence dependent setups. *Computers and Operations Research*, 34:1899–1909, 2007.

[71] S.W. Lin and K.C. Ying. Solving single-machine total weighted tardiness problems with sequence-dependent setup times by meta-heuristics. *Journal of Advanced Manufacturing Technology*, 34:1183–1190, 2007.

[72] A.G. Lockett and A.P. Muhlemann. Technical notes: A scheduling problem involving sequence dependent changeover times. *Operations Research*, 20(4):895–902, 1972.

[73] L.F. Lu, L.Q. Zhang, and J.J. Yuan. The unbounded parallel batch machine scheduling with release dates and rejection to minimize makespan. *Theoretical Computer Science*, 396:283–289, 2008.

[74] R. McNaughton. Scheduling with deadlines and loss functions. *Management Science*, 6:1–12, 1959.

[75] M. Mika, G. Waligora, and J. Weglarz. Tabu search for multi-mode resource-constrained project scheduling with schedule-dependent setup times. *European Journal of Operational Research*, 187:1238–1250, 2008.

[76] M. Mitchell. *An Intoduction to Genetic Algorithms*. MIT Press, Cambridge, MA, 1996.

[77] A. Nagar, J. Haddock, and S. Heragu. Multiple and bicriteria scheduling: A literature survey. *European Journal of Operational Research*, 81:88–104, 1995.

[78] F. Talla Nobibon, J. Herbots, and R. Leus. Order acceptance and scheduling in a single-machine environment: exact and heuristic algorithms. *Working paper KBI-0903,K.U.Leuven.*, 2009.

[79] E. Nowicki and S. Zdrzalka. Single machine scheduling with major and minor setup times: a tabu search approach. *Journal of Operational Research Society*, 47:1054–1064, 1996.

[80] C. Oğuz, F. S. Salman, and Z. B. Yalcin. Order acceptance and scheduling decisions in make-to-order systems. *International Journal of Production Economics*, 125(1):200–211, 2010.

[81] S. Panwalkar, R. Dudek, and M. Smith. Symposium on the theory of scheduling and its applications. *Springer-Verlag, New York*, pages 29–38, 1973.

[82] J.D. Papastavrou, S. Rajagopalan, and A.J. Kleywegt. The dynamic and stochastic knapsack problem with deadlines. *Management Science*, 42(12):1706–1718, 1996.

[83] A. Pessoa, E. Uchoa, M. Poggi de Arago, and R. Rodrigues. Algorithms over arc-time indexed formulations for single and parallel machine scheduling problems. Technical Report RPEP Vol. 8 no. 8, Universidade Federal Fluminense, Engenharia de Producao, Niteroi, Brazil, 2008.

[84] C.N. Potts and L.N. van Wassenhove. A branch and bound algorithm for the total weighted tardiness problem. *Operations Research*, 33(2):363–377, 1984.

[85] C.N. Potts and L.N. van Wassenhove. Single macine tardiness sequencing heuristics. *IIE Transactions*, 23(4):346–354, 1991.

[86] W.H.M. Raaymakers, J.W.M. Bertrand, and J.C. Fransco. The performance of workload rules for order acceptance in batch chemical manufacturing. *Journal of Intelligent Manufacturing*, 11:217–228, 2000.

[87] W.H.M. Raaymakers, J.W.M. Bertrand, and J.C. Fransco. Using aggregate estimation models for order acceptance in a decentralized production control structure for batch chemical manufacturing. *IIE Transactions*, 32:989–998, 2000.

[88] W. Rom and S.A. Slotnick. Order acceptance using genetic algorithms. *Computers and Operations Research*, 36:1758–1767, 2009.

[89] R. Roundry, D. Chen, P. Chen, and M. Cakanyildirim. Capacity-driven acceptance of customer orders for a multi-stage batch manufacturing system: Models and algortihms. *IIE Transactions*, 37:1093–1105, 2005.

[90] P.A. Rubin and G.L. Ragatz. Scheduling in a sequence dependent setup environment with genetic search. *Computers and Operations Research*, 22(1):85–99, 1995.

[91] S. Seiden. Preemptive multiprocessor scheduling with rejection. *Theoretical Computer Science*, 262:437–458, 2001.

[92] T. Sen, J.M. Sulek, and P. Dileepan. Static scheduling research to minimize weighted and unweighted tardiness: a state-of-the-art survey. *International Journal of Production Economics*, 83:1–12, 2003.

[93] S. Sengupta. Algorithms and approximation schemes for minimum lateness/tardiness scheduling with rejection. *Lecture Notes in Computer Science*, 2748:79–90, 2003.

[94] L. Shild and K.R. Fredman. On scheduling tasks with associated linear loss functions. *Management Science*, 7:280–285, 1961.

[95] S.A. Slotnick and T. E. Morton. Order acceptance with weighted tardiness. *Computers and Operations Research*, 34(10):3029–3042, 2007.

[96] S.A. Slotnick and T.E. Morton. Selecting jobs for a heavily loaded shop with lateness penalties. *Computers and Operations Research*, 23(2):131–140, 1996.

[97] X. Sun, J.S. Nobble, and C.M Klein. Single machine scheduling with sequence dependent setup to minimize total weighted squared tardiness. *IIE Transactions*, 31(2):113–124, 1999.

[98] K.C. Tan, R. Narasimhan, P.A. Rubin, and G.L. Ragatz. A comparison of four methods for minimizing total tardiness on a single processor with sequence dependent setup times. *Omega*, 28(3):313–326, 2000.

[99] S. Tanaka, S. Fujikuma, and A. Mituhiko. An exact algorithm for single-machine scheduling without machine idle time. *Journal of Scheduling*, 12(6):575–593, 2009.

[100] M.F. Tasgetiren, Q.K. Pan, and Y.C. Liang. A discrete differential evolution algorithm for the single machine total weighted tardiness problem with sequence dependent setup times. *Computers and Operations Research*, 36:1900–1915, 2009.

[101] H.A. ten Kate. Towards a better understanding of order acceptance. *International Journal of Production Economics*, 37:139–152, 1994.

[102] H.A. ten Kate. Order acceptance and production control. *Ph.D. thesis, University of Groningen*, 1995.

[103] J.M.S. Valante and R.A.F.S. Alves. Beam search algorithms for the single machine total weighted tardiness scheduling problem with sequence-dependent setups. *Computers and Operations Research*, 35:2388–2405, 2008.

[104] A.P.J. Vepsalainen and T.E. Morton. Priority rules for job shops with weighted tardiness costs. *Management Science*, 33(8):1035–1047, 1987.

[105] G. Wan and B.P.C. Yen. Tabu search for single machine scheduling with distinct due windows and weighted earliness/tardiness penalties. *European Journal of Operational Research*, 142:271–281, 2002.

[106] W. Wester, J. Wijngaard, and M. Zijm. Order acceptance strategies in a production-to-order environment with setup times and due-dates. *International Journal of Production Research*, 30:1313–1326, 1992.

[107] D.L. Woodruff and M.L. Spearman. Sequencing and batching for two classes of jobs with deadlines and setup times. *Production and Operations Mangement*, 1(1):87–102, 1992.

[108] D.B. Wortman. Managing capacity: getting the most from your company's assets. *Industrial Engineering*, 24(2):47–49, 1992.

[109] M.C. Wu and S.Y. Chen. A cost model for justifying the acceptance of rush orders. *International Journal of Production Research*, 34:1963–1974, 1996.

[110] M.C. Wu and S.Y. Chen. A multiple criteria decision-making model for justifying the acceptance of rush orders. *Production Planning and Control*, 8:753–761, 1997.

[111] K. Xu, Z. Feng, and K. Jun. A tabu search algorithm for scheduling jobs with controllable processing times on a single machine to meet due dates. *Computers and Operations Research*, doi:10.1016/j.cor.2009.11.012, 2010.

[112] X.Wang and L. Tang. A popultion-based variable neighborhood search for the single machine total weighted tardiness problem. *Computers and Operations Research*, 36:2105–2110, 2009.

[113] Z. Bilgintürk Yalcin. Order selection and scheduling decisions in make-to-order systems. *M.Sc. thesis, Koc University*.

[114] B. Yang and J. Geunes. A single resource scheduling problem with job-selection flexibility, tardiness costs and controllable processing times. *Computers and Industrial Engineering*, 53(2):420–432, 2007.

[115] W.H. Yang and C.J. Liao. Survey of scheduling involving setup times. *International Journal of Systems Science*, 30(2):143–155, 1999.

[116] L. Zhang, L. Lu, and J. Yuan. Single machine scheduling with release dates and rejections. *European Journal of Operational Research*, 198:975–978, 2009.

VITA

Bahriye Cesaret was born in Haskova, Bulgaria on April 20, 1985. She graduated from Lüleburgaz Anatolian High School in 2003. She received his B.Sc. degree in Industrial Engineering from Istanbul Technical University, Istanbul, in 2008. Same year, she joined the M.Sc. program in Industrial Engineering at Koç University and from September 2008 to August 2010, she worked as a teaching and research assistant at Koç University, Turkey. She has recently produced papers for the conferences PMS (Tours, France), YAEM (Istanbul, Turkey) and EURO XXIV (Lisbon, Portugal) in 2010 and submitted the results of this thesis to *Computers andOperations Research* which is under the second revision. Next year, she will be a Ph.D. candidate at School of Management of University of Texas at Dallas.