ORDER ACCEPTANCE AND SCHEDULING DECISIONS IN MAKE-TO-ORDER SYSTEMS

by

Zehra Bilgintürk

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, 2007

Koç University

Graduate School of Sciences and Engineering

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

Zehra Bilgintürk

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. Sibel Salman (Advisor)

Prof. Selçuk Karabatı

Assist. Prof. Metin Türkay

Assist. Prof. Onur Kaya

Date:

To my parents and my fiance

ABSTRACT

In this thesis, we examine simultaneous order acceptance and scheduling decisions in a make-to-order system. We model the manufacturing environment as a single machine environment, with a set of orders coming from customers. In this pool of orders, of which we know the release dates, due dates, processing times, deadlines, sequence dependent setup times and revenues, manufacturer has to decide on the subset of orders to select, considering the limited production capacity in order to maximize the profit. The tardiness of orders is penalized and revenue gained from an order decreases with tardiness, until the deadline, after which no revenue can be obtained. The problem generalizes some well-known scheduling problems with the objective of minimizing total tardiness and has the sequence dependent setup times as a further complicating property, as well as the order acceptance decisions. One special case of the problem is total weighted tardiness with sequence dependent setups, which is proven to be strongly NP-hard. As a result, the OAS problem is strongly NP-hard. In the thesis, first, we give an MILP model for the problem, and solve it with a commercial solver over a range of generated data sets. We draw some conclusions on the factors making the problem harder. Next, we propose a heuristic algorithm called ISFAN to solve the large-size problems. We compare the performance of the algorithm with respect to the optimal solution, when the problem can be solved optimally, and with several constructive heuristics we designed. We use upper bounds generated by LP relaxation of the MILP model for comparison when optimal objective values are not available. LP relaxation bound is strengthened with valid inequalities that are effective for some instances. By extensive computational tests, the proposed algorithms are found to provide high quality feasible solutions, even for large-scale instances, with modest computational effort.

ÖZET

Bu tezde, siparişe dayalı üretim yapan sistemlerde eşzamanlı sipariş kabul ve çizelgeleme problemi ele alınmıştır. Üretim ortamı müşterilerin sipariş verdiği tek makineli bir ortam olarak modellenmiştir. Bu problemde üreticinin, gelen siparişlerin üretimine başlanabilecek zamanları, işleme sürelerini, termin ve son teslim zamanlarını, dizilime bağlı hazırlık zamanlarını ve her bir siparişin getirebileceği en fazla geliri bildiği varsayılmaktadır. Üretici, bu siparişlerden oluşan havuzdan belli siparişleri seçip kısıtlı üretim kapasitesini de göz önüne alarak karını eniyilemelidir. Gecikme, bir siparişten kazanılan gelirin gecikmeyle orantılı olarak azalması şeklinde cezalandırılmaktadır. Tamamlanma zamanı son teslim tarihini aşan ürünlerden hiç gelir kazanılamamaktadır. Problem, toplam ağırlıklı gecikmenin en küçüklendiği, dizilime bağlı hazırlık zamanları ve sipariş kabul kararları içeren yapısıyla bilinen birçok çizelgeleme probleminin genel halidir. Problemin bir özel durumu polinom zamanda çözülemeyeceği ispatlanmış (NP-zor) toplam ağırlıklı gecikmenin en küçüklendiği ve dizilime bağlı hazırlık zamanları içeren çizelgeleme problemidir. Bu nedenle ele aldığımız sipariş kabul ve çizelgeleme (SKÇ) problemi NP-zor'dur. Bu tezde öncelikle SKÇ problemi için karışık tamsayılı doğrusal programlama modeli verilmekte ve bu matematiksel model ticari bir çözücü ile çeşitli veri setleri için çözülmektedir. Ayrıca, problemi zorlaştıran faktörler üzerine çıkarımlarda bulunulmaktadır. Sonrasında, daha büyük boyutlu problemleri çözebilmek için ISFAN sezgisel algoritması önerilmektedir. ISFAN algoritmasının performansı problemin eniyilenebildiği durumlarda en iyi çözümle, diğer durumlarda önerdiğimiz diğer yapısal algoritmalarla karşılaştırılmaktadır. Problemin en iyi değerini bulamadığımız durumlarda en iyi değere üst sınır olarak karışık tamsayılı doğrusal modelin doğrusal gevşetilmesi ile elde edilen değerler kullanılmaktadır. Bu üst sınır bazı durumlarda etkili olan geçerli eşitsizliklerle güçlendirilmiştir. Kapsamlı sayısal testlerde, önerilen algoritmaların büyük çaptaki problemler için bile makul hesaplama zamanlarında yüksek kalitede çözümler üretebildiği görülmüştür.

ACKNOWLEDGMENTS

First I would like to thank my supervisors Prof. Ceyda Oğuz and Prof. Sibel Salman who have been a great source of inspiration and who provided the right balance of suggestions, criticism, and freedom during my thesis studies.

I would like to thank to Prof. Selçuk Karabatı, Prof. Metin Türkay and Prof. Onur Kaya for taking part in my thesis committee, for critical reading of this thesis and for their valuable suggestions and comments.

I am also thankful to Seda for being a wonderful homemate and classmate, Dilek, Sibel, and Kenan for being helpful and joyful officemates, Ayşegül, Can, Burak, Taha, Fadime, Pınar, Uğur, Ali and Bora for their valuable friendship and Suat for his particular suggestions and comments on my studies.

Last but not the least, I thank my family for providing me a morale support that helps me in hard days of my research. I am especially grateful to my parents and my fiance for their patience and understanding. I owe everything I have today to them. Finally I thank my fiance, Fırat, for encouraging me for my graduate studies, being always near me and helping me not to lose my hope for the future. I really could not imagine today without him.

TABLE OF CONTENTS

List of Tables **x**

List of Figures **xii**

Nomenclature **xiii**

Chapter 1: Introduction **1**

1.1 Outline of the Thesis 4

1.2 Contributions to the Literature 4

Chapter 2: Literature Survey **6**

2.1 Total Tardiness and Total Weighted Tardiness Problems (with and without sequence dependent setups) 6

2.1.1 Sequence Dependent Setups 7

2.1.2 Metaheuristic methods applicable to TTSDS problem 8

2.2 Simultaneous Pricing and Scheduling Problems 11

2.3 Simultaneous Order Selection and Scheduling Problems 13

Chapter 3: Order Acceptance and Sequencing (OAS) Problem **18**

3.1 Problem Definition 18

3.2 Connection to the Literature 20

3.3 The Computational Complexity of the Problem 20

3.4 Mathematical Model 22

3.5 Finding Upper Bound for the Problem 26

3.5.1 LP Relaxation 27

3.5.2 LP Relaxation with Valid Inequalities 27

3.5.3 Lagrangean Relaxation 28

3.6 Dominance Relations for the Problem 28
3.6.1 Dominance properties dealing with acceptance decisions 29
3.6.2 Dominance property dealing with sequencing decisions 29

Chapter 4: Heuristic Solution Approaches 31

4.1 Iterative Sequence First- Accept Next(ISFAN) Algorithm 31
4.2 Constructive Heuristics 33
4.2.1 Dynamic Release First- Sequence Best (d-RFSB) Heuristic 34
4.2.2 Modified ATCS (m-ATCS) Heuristic 35
4.3 MILP Heuristic (MILPH) 37
4.4 Local Improvement Methods 38
4.4.1 Order Insertion Algorithm 38
4.4.2 Order Exchange Algorithm 39

Chapter 5: Computational Studies 45

5.1 Data Generation 45
5.1.1 Random Data 45
5.1.2 Slotnick & Morton's (S&M) Data 46
5.2 Computational Experiments 46
5.2.1 Computational Platform 46
5.2.2 Upper Bound Comparisons 46
5.2.3 Experiments with MILP model 47
5.2.4 Experiments for the ISFAN Algorithm 48
5.2.5 Experiments for d-RFSB and m-ATCS Heuristics 52
5.2.6 Experiments for MILPH 54
5.2.7 Conclusions on Computational Experiments 55

Chapter 6: Further Studies 74

6.1 Exact Method Development 74
6.2 Heuristic-Metaheuristic Method Development 79

Chapter 7: Conclusions 81

Bibliography **84**

LIST OF TABLES

5.1 The average % improvement of UB_{LPVI} over UB_{LP} for sizes of 10 and 15. . . 58

5.2 The average % improvement of UB_{LPVI} over UB_{LP} for sizes of 20 and 25. . . 59

5.3 The average % improvement of UB_{LPVI} over UB_{LP} for sizes of 50 and 100. . 60

5.4 The average % improvements of UB_{MILP} and UB_{LPVI} over UB_{LP} for sizes of 10 and 15. 61

5.5 Evaluation of different cooling functions for n=50, $\tau = 0.3$, $T_{initial} = 800$ . . . 63

5.6 % Gap of the ISFAN algorithm results from UB_{LP} for initial temperatures of 800, 1000 and 1500 for n=50, $\tau = 0.3$, $CF = 3$ (Compared with UB_{LP}) . . 63

5.7 ISFAN and MILP % deviation comparison from min(UB_{LPVI},UB_{MILP}) for the OAS problem with sizes 10 and 15. 65

5.8 CPU time comparison of MILP and ISFAN for the OAS problem with sizes 10 and 15. 66

5.9 ISFAN objective value % deviations from UB_{LPVI} and the correponfing CPU times for the problem with sizes of 20 and 25. 67

5.10 ISFAN objective value % deviations from UB_{LPVI} and the correponfing CPU times for the problem with size of 50. 68

5.11 ISFAN, d-RFSB and m-ATCS heuristics comparison for the problem with sizes of 10 and 15. 69

5.12 ISFAN, d-RFSB and m-ATCS heuristics' average % deviations from UB_{LPVI} for the problem with sizes of 20 and 25. 70

5.13 ISFAN, d-RFSB and m-ATCS heuristics' average % deviations from UB_{LPVI} for the problems with size of 50. 71

5.14 Number of rejected orders for ISFAN, d-RFSB and m-ATCS heuristics for $n = 50$. 71

5.15 d-RFSB and m-ATCS heuristics' average % deviations from UB_{LPVI}) for the problems with size of 100 72
5.16 MILPH gaps from UB_{LPVI} and CPU times for problem sizes of 15 and 20. . 73

LIST OF FIGURES

4.1 Flow chart for ISFAN algorithm. 42

4.2 Flow chart for Insertion algorithm. 43

4.3 Flow chart for Exchange algorithm. 44

5.1 Cooling functions 2, 3 and 4. 62

5.2 Convergence behavior of the ISFAN algorithm at a rejection iteration for a 50-order problem having τ=0.3, R=0.3, initial temperature=800 and cooling function 3. 64

6.1 Inserting job $[i]$ after job $[k]$. 76

6.2 Interchanging of positions of jobs $[i]$ and $[k]$. 77

6.3 Interchanging of positions of adjacent jobs $[i]$ and $[i+1]$. 79

NOMENCLATURE

MILP	Mixed Integer Linear Programming
OAS	Order Acceptance and Scheduling Problem
SA	Simulated Annealing
ACO	Ant Colony Optimization
TTSDS	Total Tardiness Problem with Sequence Dependent Setups
EDD	Earliest Due Date
LP	Linear Programming
LR	Lagrangean Relaxation
UB_{LP}	LP Relaxation Bound
UB_{LPVI}	LP Relaxation Bound with Valid Inequalities
UB_{MILP}	MILP Upper Bound at Termination
PCTSP	Prize Collecting Traveling Salesman Problem
S&M	Slotnick and Morton
ISFAN	Iterative Sequence First- Accept Next Algorithm
CF	Cooling Function
d-RFSB	Dynamic Release First- Sequence Best Heuristic
m-ATCS	Modified Apparent Tardiness Costs with Setups Heuristic

Chapter 1

INTRODUCTION

In today's competitive environment, firms increasingly present more customized and unique products in order to attract the customers to have a greater share from the market in which they are players. Additionally, the positive effect of make-to-order working principle on inventories and on the vulnerability of the firms in case of possible economical crisis increase the popularity of the make-to-order systems. The success stories of the principle in the world-wide known firms such as Dell Computers [12] attracted many manufacturing firms to offer a number of diversified products on demand in recent years.

The companies working on make-to-order operating principle require a high level of decision making practice and farsightedness since the incoming orders would have no standards on their features that change the time and the cost parameters. In addition, the customers may and probably would have constraints on the delivery time of the orders as a further complicating issue for the decision making process. In a profit maximizing make-to-order firm, all of these issues must be carefully considered and handled for a feasible and beneficial decision.

In this study we focus on make-to-order firms, such as a custom-made furniture shop or a boat builder, which manufacture customized products, and in which the production is initiated by a customer order. Typically, no work-in-process inventory is carried for this type of products. The manufacturer gains a revenue if a particular order is accepted and manufactured by using the production resources. Since the main production resource, i.e. the capacity, is limited, manufacturing one order may delay another one. If there are no deadlines, then the manufacturer will just incur some penalty for the orders which are delayed. If there are deadlines for the orders, such delays may lead to rejecting customer orders. In a competitive make-to-order environment, a manufacturer should use the capacity efficiently, satisfy the expectations of customers at a high level and gain the maximum revenue from the incoming orders. Therefore, the manufacturer should find a balance between the revenue that can be gained from the accepted orders and the cost incurred by manufacturing these orders. Hence, the question is which orders to accept to maximize the profit considering the limited production capacity of the firm. Accepting an order whose deadline cannot be met causes a loss in the reputation of the firm as well as in the revenue. To avoid such circumstances, the order acceptance decisions should be made very carefully. When high utilization levels are present, firms accept the orders that will bring high revenues and have comparably less production capacity requirements only. If an order will be tardy, that is, if its completion time will exceed its due date, the manufacturer has to sacrifice from some of its revenue, if this order is accepted.

The scope of this study is restricted to a single machine environment. Single machine environment is one of the most widely used environments in scheduling literature and it is easily justified: Consider a multi-machine environment which has a bottleneck machine significantly influencing the total performance of the system. Then this machine can be isolated and considered alone as a single machine.

Moreover, in the problem, sequence dependent setup times among the orders to be processed are present. Setups generalize the work needed on the machine or on the order before the order becomes available for the main processing. Inspecting materials, cleaning, dye changes, tool changes are some examples of setups. Although setup times or costs are negligible or can be taken within the processing time for make-to-stock products, make-to-order systems require separate consideration of setup times/costs. Inclusion of sequence dependent setup times increases the hardness of the the underlying scheduling problem.

In this study, we consider the problem from the manufacturer's point of view and try to maximize the profit gained from the selected orders. The profit is defined as the total revenues minus the total weighted tardiness. The manufacturing environment consists of a single machine with a limited capacity and a set of independent orders O ($|O| = n$) at the beginning of the planning period. For each order $i \in O$, we know the data including the release date r_i after which the order is available, the processing time p_i, the due date d_i up to which order i can be produced without incurring a penalty, the deadline $\bar{d}_i$ after which order i cannot be produced ($\bar{d}_i \geq d_i$), the sequence dependent setup time s_{ij} which is required when order i is scheduled before order j, and the revenue e_i to be gained if the order is accepted. The revenue e_i may also reflect the level of priority or the importance of order i. The decisions to be made in this environment include determining the set of accepted orders and the corresponding optimal schedule for this set of orders to minimize the weighted tardiness. The tardiness of order i is defined as $T_i = \max\{0, C_i - d_i\}$ where C_i is the completion time of order i.

As our aim is to maximize the profit gained from the accepted orders, which is defined as the total revenues minus the total weighted tardiness, our objective function is then defined by $\max \sum\limits_{i \in A} R_i = \sum\limits_{i \in A} (e_i - T_i \times w_i)$ where A is the set of accepted orders, w_i is the weight for the tardiness of order i and R_i is the profit gained from the accepted order i. The profit R_i reflects the fact that if an order i is tardy, the customer will receive a discount. However we note that the early delivery is not penalized as the manufacturer is assumed to have unlimited capacity for finished product storage. In this setting, the manufacturer has the right to reject an order that will not be profitable. We will denote this problem as OAS (Order Acceptance and Scheduling) problem throughout the thesis.

OAS problem is a common real-life problem faced at the production companies as well as the service companies. One such example to manufacturing companies facing this problem is a packaging firm that has varying sequence dependent setup times for each order. Companies working on project basis, such as law, accounting, and consulting firms are the examples of service companies working on a make-to-order basis that face the OAS problem.

The OAS problem can be stated as a single machine total tardiness scheduling problem with sequence dependent setup times (TTSDS) when some of the parameters are fixed. TTSDS problem is important in practice and in scheduling literature, since it is essential to minimize the total time each order/customer waits to assure customer satisfaction and retain a good reputation in today's competitive business environment. Consideration of sequence dependent setup times increases the reality of the problem for the make-to-order case.

The objective of the total tardiness problem is to find a schedule that minimizes the total tardiness of the jobs and is defined as $\sum\limits_{i=1}^{n} T_i = \sum\limits_{i=1}^{n} max\{0, C_i - d_i\}$, where C_i is the completion time of order i, and T_i is the tardiness of job i, $i = 1, \ldots, n$ [11]. Emmons [6] was one of the first researchers who studied this problem in the late sixties and from that time on, it has been found attractive to work on by various researchers. Dynamic programming methods and branch and bound methods were used for solving moderately small sized problems to optimality [26]. For relatively long time, the important question whether the total tardiness problem is polynomially solvable or NP-hard remained open. Finally, the problem was shown to be NP-hard by Du and Leung in 1990 [4]. Due to this result, reaching optimality in reasonable time limits is impossible for the moment, except for problems with a very limited number of jobs. However, the total tardiness problem can be solved in polynomial time under some special settings [11]. Furthermore, we can solve the problem up to some degree by using some dominance relationships and by combining them with a branch-and-bound method. When we add the sequence dependent setup times, the characteristics of the problem change and the dominance relations of the total tardiness problem become invalid and useless. For this reason, there are many applications of heuristics and metaheuristics for this problem, which yielded quite good results. In the content of the thesis, the main aim is to find time efficient and high quality solution methods to maximize the make-to-order manufacturer's revenues by incorporating the order acceptance decisions into scheduling decisions.

1.1 Outline of the Thesis

In Chapter 2, we give an extensive literature review for the OAS problem. In this review, some special cases of the problem and the solution methods proposed for these problems are also addressed. In Chapter 3, first, we define the OAS problem in detail, and give information on its hardness. Next, we present our MILP model for the problem. In Chapter 4, we propose a hybrid algorithm, ISFAN, that joins local search procedures, Simulated Annealing metaheuristic and a rejection rule in it. Next, we give local search methods and several constructive heuristics for the OAS problem and compare the performances of each. Finally in Chapter 7, we give our conclusions and mention future study areas on the problem.

1.2 Contributions to the Literature

In this thesis, we consider an order acceptance and scheduling problem that was not fully dealt in the literature before. The OAS problem is a hard problem that has many applications in real life. First of all, we developed benchmark data for this problem since no complete data was available in the literature. Throughout this study, we first formulated a MILP model for the exact solution of the problem. Since the problem is strongly NP-hard, we proposed an efficient hybrid metaheuristic algorithm called ISFAN which is much less time-consuming compared to the exact method. ISFAN algorithm proved to perform very well for the OAS problem in the computational experiments. Throughout the experiments with our data and Slotnick and Morton's data [28], whose problem is a special case for the OAS, ISFAN is a good performing algorithm in terms of time and solution quality. Moreover, we proposed two very time-efficient constructive heuristics for the problem, dynamic release-first sequence-best (d-RFSB) algorithm and a modified version of this algorithm which uses a selection index similar to the ATCS (m-ATCS) algorithm. Both algorithms perform very well in terms of the solution quality as well as time considerations. m-ATCS performs the best as n gets larger than 50. Furthermore, we employed two local search methods, order insertion and order exchange, in each of the algorithms in order to improve the solutions locally.

Chapter 2

LITERATURE SURVEY

In this chapter, we investigate the problems that are related to the OAS problem and give the solution methods proposed for these if applicable.

The papers discussed in this literature survey can be categorized in four subtitles : 1) studies concerning the total tardiness/ total weighted tardiness problem (with or without sequence dependent setups), 2) studies concerning the order selection problem, 3) studies concerning the simultaneous pricing and scheduling problem, 4) studies concerning simultaneous order selection and scheduling problem.

2.1 *Total Tardiness and Total Weighted Tardiness Problems (with and without sequence dependent setups)*

The total tardiness problems have received great attention in the operations research literature, and have been studied since 1960s. Koulamas [10] provides an extensive literature review about the total tardiness problems, and Keskinocak and Tayur [8] give research papers focusing on different aspects of the problem. Emmons is one of the first researchers who studied the total tardiness problems. Emmons [6] neglects the job setup times, and considers the problem of sequencing n jobs on one machine, provided that all jobs are available at time zero, and their processing times and due dates are known in advance. He proposes an enumerative algorithm to solve the problem.

Although the total tardiness problem had been popular in scheduling literature, most of the studies were on proposing solutions to the problem rather than considering the computational complexity of the problem until 1970s. The proposed methods mainly concentrated on finding efficient enumerative algorithms. Lawler's study [13] is one of the first researches emphasizing the complexity of the problem. In his study, Lawler proposes a pseudo-polynomial algorithm for the total tardiness problem where jobs have agreeable weights (that is, if processing times p_i and p_j satisfy $p_i < p_j$, then weights satisfy $w_i \geq w_j$), implying that the total tardiness problem cannot be NP-hard in the strong sense. In addition, the weighted tardiness problem is proved to be NP-hard in the strong sense in this study. Even though they could not develop a method outperforming Lawler's pseudo-polynomial algorithm [13], Potts and Wassenhove [23] present a well-performing branch and bound algorithm for this problem in which they use a Lagrangian relaxation method in order to obtain lower bounds in the subproblems. This relaxation decomposes the problem into subproblems with weighted total completion time objective. Instead of computationally time-consuming subgradient optimization technique, authors use a multiplier adjustment method to compute bounds faster in the relaxation. This method allows calculating fast bounds as intended, but the resulting bounds are not very tight. The study is important in the sense that it incorporates diverse tools for verifying dynamic programming dominance in the tree.

Although the computational complexity of the weighted tardiness problem was determined, the issue whether the total tardiness problem is NP-hard or not was not resolved until 1990s. Du and Leung [4] prove that minimizing total tardiness on one machine is NP-hard in the ordinary sense by reducing a restricted even-odd partition problem to this problem. Since McNaughton [18] proved that preemption does not decrease the tardiness of the problem, this proof can be extended to the preemptive case as well. Although the classical single machine total tardiness problem is NP-hard, the problem can be reduced to polynomially solvable problems with a set of special settings[11].

2.1.1 Sequence Dependent Setups

In the literature, single machine weighted total tardiness has been studied with setups which are either sequence dependent or independent. Setups have practical meaning in make-to-order manufacturing systems, since each order has its own characteristics, and thus require different setups. In scheduling problems where sequence dependent setup times are present, the sequence in which the jobs are processed on a machine affects the required time for completion of that job. Comprehensive literature reviews on setup considerations in scheduling problems are studied by Allahverdi et al.[1] and by Yang et al.[35]. One study dealing with impacts of sequence dependent setup times and variation of setup times in scheduling environments is Kim et al.'s[9]. This study shows that high variation on sequence dependent setup times has a negative effect on shop performance in job shop environments. While the addition of sequence dependent setup times increases the veracity of the system, it also increases the computational complexity of the underlying problem. For instance, single machine total weighted tardiness problem with sequence dependent setups is proven to be NP-hard in the strong sense [14]. A very time-efficient heuristic that produces high quality solutions and designed for the problems having tardiness objective is Apparent Tardiness Cost with Setups (ATCS) heuristic [15]. ATCS rule is a modified version of the ATC rule developed by Vepsalanien and Morton [31] and Ow and Morton [22]. ATCS rule is a dynamic rule that evaluates the orders/jobs' priority in the schedule in terms of processing time, weight, due date, setup time and the current time, and used for efficiently scheduling the orders.

Since the total tardiness problem with sequence dependent setup times (TTSDS) is NP-hard, and thus, very hard to solve for large instances; application of metaheuristics is very common as a solution approach as well as various branch and bound algorithms.

2.1.2 Metaheuristic methods applicable to TTSDS problem

One of the first applications of the metaheuristics to TTSDS problem was by Rubin and Ragatz [25]. This study leaded many of the researchers to use different metaheuristics for single machine total tardiness problem with sequence dependent setup times. In the study, authors design and apply a genetic search for the problem. They use a permutation of n jobs to represent a solution. Since the aim is to minimize total tardiness, reproduction scheme is designed so that absolute position of each job is relevant to the total cost of the schedule. Then, jobs with early due dates should be appearing early in the schedule, and vice versa. Therefore, a parent should pass along both information on the adjacency of pairs of jobs and information on the absolute positioning of the jobs for a good reproduction. RRX(Rubin Ragatz crossover) is designed and used especially for this problem. Besides, authors tried different mutation operators while searching the best improving mutation operator. At the early phases, they tried the job exchange method, where two jobs are interchanged randomly, but this method displayed a slow convergence. Then, they tried a different mutation operator which tries all possible interchanges of adjacent pairs of jobs and accepts the interchange if it improves the current schedule, until no improvements are obtained. Alternatively, if this operator does not achieve any improvement by adjacent job exchanges, randomly chosen pair of jobs is swapped. This specially designed mutation operator, compared to job exchange method, ensures that mutation improves most of the individual schedules. Although it takes more time to create a generation with this operator, the convergence progress is found satisfactory. Genetic search was promising since it did not create solutions too far from the optimal or best-known values, and it used less computational time than branch-and-bound method did. The study is important in the literature in the sense that, it encouraged and inspired other researchers to use metaheuristics in TTSDS problems, and the solution approach had significant computation time advantage over branch-and-bound algorithms proposed by that time.

Simulated Annealing (SA) method is another well-known and well performing metaheuristic algorithm for combinatorial optimization problems. The successful implementation of SA algorithm motivated researchers to apply this method to TTSDS problem as well after genetic search application. Tan and Narasimhan [29] propose a simulated annealing approach to TTSDS problem. Simulated annealing technique found to be outperforming random search method (which continuously improves objective function and has no procedure to escape from local minima), which outperformed genetic search of Rubin and Ragatz [25], in all but three out of 18 instances in terms of solution quality. Although random search method has a better computation time, this advantage is offset by solution quality of simulated annealing technique described by Tan and Narasimhan [29].

Another metaheuristic efficiently applied to TTSDS problem is ant colony optimization (ACO). ACO is most successfully applied to TSP problem in the literature because of its nature, but with some arrangements, it can be used to solve TTSDS problem efficiently as well. Gagné et al. [7] propose an ACO algorithm for TTSDS problem. In this algorithm, two distance matrices are used: one for setup times and one for slack times. Setup time matrix is of size $(n+1)\times n$, where additional column stands for an initial solution. By using the relative values of setup times and slack times, authors exploit the trade-offs that the algorithm deals. For each constructed sequence, last job processed during the previous period is taken to be the initial one in the next sequence construction, to maintain continuity. Authors also include local improvement methods in ACO algorithm in order to improve the performance further. One such method, restricted 3-opt model, which has an important aspect of not inverting the whole sequence and is quite useful for sequence dependent setup environment; and random start pairwise interchange (RSPI) method, which comprises inverting each pair of adjacent jobs in turn. Authors randomly select and use one of these local improvement methods to improve the route each ant forms in the algorithm. Moreover, authors add a candidate list formed by cities not yet assigned (orders which are not processed), which consists of orders having smallest $'cl'$ slack times. τ_{ij}, the trail intensity of edge (i, j), which gives information on the frequency of using this edge, is initialized as $(n \times L_{nn}) - 1$ for each edge where n is the problem size and L_{nn} is the result of the solution obtained by any simple and quick method. Authors propose a different look-ahead information idea, which consists of estimating the probable quality of partial solution by taking the actual value produced by the present partial solution and adding the consequences of one of the candidate choices as well as a lower bound on the remaining unfinished part. Performance of the ACO algorithm is compared with and without look-ahead information in the study. Although the computation time increases when look ahead information is added, authors show that, it significantly improves the quality of the solution obtained. ACO algorithm offers solving the TTSDS problem up to a size encountered in industrial cases. Gagné et al. [7] tested their algorithm on set of problems generated by Rubin and Ragatz [25] and on some randomly generated test data. A comparison of the efficiency of four heuristic techniques developed for TTSDS problem with respect to Ragatz's algorithm is given by Tan et al. [30]. Unfortunately, the comparison of the computational times is invaluable since the nature of the computers used and the nature of the algorithms are different from each other. From test results, authors concluded that ACO algorithm is competitive in solution quality and has shorter computation times than the RSPI (Random search with pairwise interchange) method [30] which is found to be best performing method especially for large size problems. Resultantly, ACO provides computation time advantages over other methods particularly for industrial-size problems.

Liao et al. [16] proposed an ACO algorithm for the weighted and unweighted TTSDS problem. This study incorporates the heuristic information by introducing a new parameter for the initial pheromone trail to hinder premature convergence. Additionally, local search runs are set according to the iteration results rather than applying local search at each iteration unlike the other ACO algorithms. The algorithm is found to be competitive with the existing metaheuristics and efficient in computation time.

Miller et al. [20] propose a hybrid genetic algorithm (HGA) for the single machine scheduling problem in a make-to-stock environment with setup costs where the objective is to minimize the sum of setup cost, inventory cost and backlog cost each associated by time. The proposed method combines the standard genetic algorithm with Wagner-Whitin algorithm used for local improvement and TSP modeling used for sequencing of the jobs. The algorithm is found to be efficient both in terms of time considerations and solution quality. HGA is promising for use in make-to-order systems as it outperforms the standard GA.

2.2 Simultaneous Pricing and Scheduling Problems

When price or cost of a product is affected from the position of the product in the schedule, simultaneous pricing and scheduling problems arise. When timing of placing an order affects the cost incurred by the customer (or the price quoted by the manufacturer), placing the order at the right time has critical value to minimize the customer's cost. If the delivery time of the product affects the revenue gained from that product, manufacturer has to schedule the orders very efficiently to maximize her profit. Therefore, simultaneous pricing and scheduling problem is an important problem for make-to-order systems from viewpoints of both manufacturer and the customer.

Webster [32] studies a simple model of make-to-order firm modifying the price and the capacity to adapt to the market changes. Webster defines the corresponding make-to-order system so that the price $p(t)$ and the lead time $l(t)$ of an arriving order are known and they are the determinants of the demand rate at time t. Resultantly, the demand is time-dependent. In this study, price is defined as net of delivery costs, so it is indirectly related to the margin and the capacity. Therefore, demand rate is defined as a function of lead time, margin and capacity. Webster intends to provide insights for pricing, capacity and lead time policies for make-to-order products. The paper is related to different literature streams such as how to dynamically price a product over time given the demand and the capacity, how to quote reasonable lead times in a make-to-order environment given the demand and the capacity, and connected to queueing literature dealing with capacity given the demand costs and the performance criteria. The study deals with a problem that was not investigated before and the model presented can be used as a base for further study. Webster provides a range of properties that are used for illustrating policy performance to help finding policies which can work well in complex real-life systems in the study. This study relates the pricing and scheduling decisions in one problem, mainly emphasizing on the capacity decisions.

Charnsirisakskul et al. [2] combine the pricing and scheduling decisions in one model in their recent study. The authors investigate the problem faced by a manufacturer who has the ability to set the prices to influence the demand, reject some of the orders, set the lead times or the due dates for selected orders. Simultaneous pricing, order acceptance, scheduling and lead time decisions are studied under the assumption of having the flexibility to charge either equal or different prices to each customer. The main objective in the study is to provide insights regarding the benefits of the price customization, the lead time, and the inventory flexibilities in various market settings. The problem is formulated as a mixed integer program. A multiple price model (p^i_j per unit price where manufacturer can charge for order i) and a single price model (where manufacturer has to select the best single price to quote to all customers) are considered in the paper. Charnsirisakskul et al. assume a single machine environment having 100% reliability. Setup times and setup costs between jobs are neglected and preemption of jobs is allowed. In the problem setting, customer orders differ in arrival times, demand, quantities, preferred and acceptable due dates and unit production, holding and tardiness costs. Final prices should be selected from a set of acceptable prices for each order and the products should be delivered altogether when finished. In this problem, manufacturer has the option of accepting/ rejecting the customer orders and she can assign different prices to each of the customers. Once the order is accepted, the manufacturer decides on the production schedule. The aim of the manufacturer is to maximize the profit subject to capacity, delivery time and demand constraints. In addition, a special case is identified, where the manufacturer has a few information about arriving orders that allows her to start the production of the order before the order commitment time. This case is generally not applicable for make-to-order systems. For computational issues, two LP-based rounding heuristics are proposed for the multiple and the single price model. Consequently, authors evaluate the priority of flexibilities that a manufacturer may choose to have such as price, inventory and lead time flexibilities. The consideration of the latest acceptable lead time is one of the distinctions of this model from the other studies in the literature.

2.3 Simultaneous Order Selection and Scheduling Problems

Research on the order selection problem has grown rapidly in the literature over the past quarter, parallel to the increasing popularity of the make-to-order environments. Wester et al. [33] consider a manufacturing environment where different items are produced on a single machine in batches. The underlying problem is to find acceptance strategies to make good acceptance decisions where producer has the flexibility of accepting or rejecting an incoming order. Monolithic, hierarchic and myopic approaches are compared using simple scheduling techniques in simulation experiments. Poisson arrivals with parameter λ_j are assumed in all of the experiments. In the monolithic approach, when a new order is received, schedule is updated for the orders accepted but not yet processed, including the newest one. In this approach, performance highly depends on the scheduling procedure. If no lateness occurs, new schedule is maintained; otherwise incoming order is rejected. In the hierarchic approach, readily available schedule is used to select the next order to process. A simple priority rule is used for the next order to be processed in the myopic approach, instead of forming a detailed schedule. The results indicate that the approaches do not generate drastic performance changes, and none of the them found to be dominating for every case to lead better order acceptance decisions.

Duenyas and Hopp [5] investigate the problem of dynamically quoting customer lead times in a production system where different modeling assumptions are considered. In the study, given the price, authors examine the cases where capacity is infinite, finite and processed in FCFS (first-come-first-serve) order, and the case in which jobs are finite and processed in a sequence other than FCFS order with scheduling considerations. Assuming the production time function and order placement probabilities to be continuous, authors derive a closed form expression for the optimal lead time quotation under infinite capacity assumption. Authors develop optimal control-limit policies for the case where price and acceptable lead time are fixed. In this setting, the manufacturer is able to reject a job by offering a lead time exceeding the acceptable latest delivery time for this job. Additional policies are proposed for the case where firms are free to rival on offered lead times where customer's probability of accepting an order depends on the quoted lead time. Finite capacity is assumed and sequencing is handled with FCFS processing principle for these policies. When the assumption of sequencing under FCFS processing principle is relaxed, problem becomes more complex. For this case, authors state the conditions in which processing jobs in EDD (earliest due date) sequence is optimal for the due-date quoting/order scheduling process. The research utilizes queuing theory and successfully fits the lead time quoting process and scheduling into queuing concepts. In addition, the study gives an example on how to consider order acceptance and scheduling concepts simultaneously in one problem.

Wouters [34] mainly discusses the economic considerations for order acceptance decisions. In the paper, the question how managers can base their order selection decisions on opportunity cost and incremental cost is resolved. Furthermore, it is shown that the production planning and control can provide useful information for finding the impact of current order selection decisions on the planned activities and the planned level of capacity costs.

Charnsirisaksul et al. [3] study an integrated problem of order selection and scheduling on a single machine. Unlike Wester et al. [33] who build a trivial suboptimal sequence and analyze the situation with simulation, Charnsirisaksul et al. [3] give a mixed integer programming formulation and pursue the optimal acceptance and sequence decisions in a preemptive environment. The authors assume deterministic demand and neglect the setup times and setup costs in the problem. In this problem, the manufacturer selects the lead time for each order. The main objective is to maximize the manufacturer's profit which is defined as revenue minus the manufacturing, the holding and the tardiness costs. This objective is pursued by coordinating the order selection, scheduling and lead time decisions. Authors determine the appropriate due date negotiation strategies, order acceptance and scheduling decisions with the numerical study performed. This study primarily concentrates on comparing the advantages of partial order fulfillment flexibility, inventory flexibility and lead time flexibility in order to determine efficient strategies. The study develops insights on the benefits of lead time setting flexibility for the manufacturer, and combines order selection and scheduling problems into one problem.

Another study concerning order acceptance decisions is by Roundry et al. [24]. In the problem, the firm has to decide on how to partition the JIT (Just-in-time) shipments into a number of production batches in a job shop environment. The batch order sizes and the batch due dates are determined to constitute the sequence of small JIT shipments. Next, it is examined whether the capacity is sufficient or not to produce these batches. In the modeling phase, authors assume that processes are deterministic, and raw material cost of the product dominates the holding cost. The discrete-time version of the problem is modeled as a mixed integer program with cost minimization objective and it is shown to be NP-hard. In the study, authors design and test viable algorithms for the model. In the problem under consideration, manufacturer starts with a feasible schedule for the current production load. When a new order is received, a feasible schedule that contains the new order and all of the orders that were previously accepted is investigated. Authors consider each machine in isolation, and consider each machine separately for the study. As a result, the authors prove that, for any machine m, a feasible schedule is present if and only if there is a feasible schedule that contains the operations in EDD (earliest due-date) order sequence. By exploiting this property, they tried to determine the flexibility limits: how far left and right the current operations can be pushed. Next, the heuristic solution approaches are introduced to obtain good computational results since the problem is found to be NP-hard. Each approach is experimented on real data as well as on randomly generated data, and finally, performances of the algorithms are evaluated and compared. The study characterizes valid feasible schedules for the underlying problem.

In the literature, simulation-based solution approaches are successfully applied to order acceptance/rejection problems as well. One example is due to Nandi and Rogers [21]. In this study, simulation is used as an effective tool to give the right decisions on order selection, that is, whether a specific order should be accepted or not by the manufacturer considering the limited capacity of the production facility and the profit to be maximized. Nandi and Rogers propose a simple rule to evaluate both order's acceptance and rejection cases in the simulation process: A simulation experiment is run both for the coming order's acceptance and rejection scenarios. In these simulation runs, authors assume that no additional orders arrive until the accepted ones are completed. If the outcome of the first scenario run exceeds the outcome of the second scenario run by some predefined proportion of the maximum possible revenue for that order, the order is accepted, otherwise it is rejected. This rule is customized for the problem in the sense that it uses information from the order itself and from the status of the shop floor. Although simulation needs simplifying assumptions to solve the complex order selection and scheduling problem, the study gives insights on how a make-to-order manufacturing system can be controlled using simple acceptance rules and how performance measures are affected by these rules.

A related study on order acceptance and scheduling problem is presented recently by Slotnick and Morton [28]. In the study, the revenue, the processing time, the due date and the weights reflecting the importance of the customer/order are assumed to be known in advance for each order. In this problem setting, customers receive a discount proportional to the time duration the order is late if the orders are delivered after their due dates. The paper examines order acceptance decisions where the production capacity is limited, and where the long-term capacity expansion and the early delivery options are not taken into consideration. Manufacturing facility is modeled so that there exists a set of orders, and the decision maker has to choose a subset of orders that results in the highest profit and determine the corresponding optimal sequence. In the paper, first the problem is considered by separating order acceptance and sequencing decisions to utilize the solution approach of a similar problem, weighted lateness [27]. Authors then propose an optimal branch-and-bound method with an LP relaxation for simultaneous order acceptance-sequencing problems since the computational results for the separated problem was not promising. For this approach, a few alternative heuristic methods are also developed since the branch-and-bound method is usable for a very limited number of orders. In order to improve branch-and-bound solutions by reducing the search space, authors propose a modified removable set procedure (RSP), which is developed and applied to the lateness problem before. Slotnick and Morton basically state that, when some conditions hold, the orders that will be in the optimal set can be addressed. Authors prove that for a fixed sequence S, RSP remains valid for the weighted tardiness acceptance problem. Although RSP is valid, in order to exploit the procedure, one first has to find the optimal sequence to the weighted tardiness problem which is NP-complete. To deal with this sequencing issue, two dispatch heuristics, Rachamadagu and Morton heuristic and Montagne's method, are used. These two methods and a beam search heuristic are compared with the branch-and-bound method, for the problem size of 10. A version of beam search that performs well in scheduling problems is used as a benchmark for the rest of the analysis because the optimal method is intractable for problems of size 20. In this analysis, beam search and Montagne heuristic performs very similar in terms of the solution quality, but the latter exhibits a much better computation time. Resultantly, separating order acceptance and scheduling decisions is proved not to work well for the simultaneous order acceptance and sequencing problems.

Chapter 3

ORDER ACCEPTANCE AND SEQUENCING (OAS) PROBLEM

The system under consideration along this thesis is a make-to-order system where each of the orders has different characteristics such as the maximum revenue that can be gained, the required processing time to complete a specific order and the required sequence dependent setup time on the machine for that order. In the problem, as in majority of the make-to-order systems, products are sophisticated and specially designed for each customer. Thus, no work-in-process inventory can be carried for the incoming orders. However, it is assumed that the manufacturer has no inventory holding limit for the completed orders, and she incurs no holding cost.

The organization of this chapter is as follows: In section 3.1 we give a definition for Order Acceptance and Sequencing Problem. In section 3.2, we relate the study with the existing literature. Then in section 3.3, we evaluate the computational complexity of the problem. Finally, in section 3.4, we introduce our mixed integer linear programming model and formulation.

3.1 Problem Definition

The order acceptance and scheduling (OAS) problem is defined as follows: In a single machine environment, we are given a set of independent orders O, at the beginning of the planning period. For each order $i \in O$, we have the data including release dates r_i, processing times p_i, due dates d_i, deadlines $\bar{d}_i$ which are greater than or at least equal to due dates, sequence dependent setup times where each element s_{ij} is the time that has to be incurred before order j is processed if order i precedes order j (s_{ij} is not necessarily equal to s_{ji}), revenues e_i which demonstrate the maximum revenue the manufacturer can gain from each of the selected orders, and weights w_i which are used while the revenue earned from an order is discounted by the amount of tardiness. If an order i is tardy for T_i units, the revenue one can gain from an order i decreases by $-w_i \times T_i$. The objective of the OAS problem is to find the optimal set of the selected orders and the optimal schedule for them that maximizes the manufacturer's total revenue, TR.

Suppose we have n orders, independent from each other, which will be processed by a single machine that can handle only one order at a time. We can define the elements of the OAS problem as follows:

Given a sequence σ_s of the selected order set S, we can calculate the completion time C_i of each order i, $i \in S$. Using the completion time C_i and the due date d_i of an order, we can calculate the tardiness of order i, $i \in S$, with the formula $\max\{0, C_i - d_i\}$. Let $T_i(\sigma_s)$ be the tardiness of job i, $i \in S$, in sequence σ_s. Manufacturer can process order i until its deadline $\bar{d_i}$, but for any time unit beyond the due date of an accepted order, she incurs an amount of penalty cost. The indirect penalty cost is modeled as follows: As the tardiness T_i of an order increases, the revenue we can gain from that order decreases with a linear function. The revenue we gain from the order becomes zero at its deadline. The completion time C_i of an order cannot exceed its deadline $\bar{d_i}$, and such an order cannot be accepted. Therefore $R_i(\sigma_s)$ denotes the revenue generated from order i, given that its tardiness is $T_i(\sigma_s)$ in the sequence σ_s, that is $R_i(\sigma_s) = e_i - T_i(\sigma_s) \times w_i$. Then, the total revenue gained from processing orders in S in the sequence σ_s is $TR(\sigma_s) = \sum\limits_{i \epsilon s} R_i(\sigma_s)$.

Intuition for this problem can be given as follows: The due date can be considered as the latest time preferred by the customer for delivery of his order. As an order becomes tardy, customer satisfaction from this order begins to decrease. Therefore, the customer accepts to pay only a discounted amount for his order when the due date is exceeded. The deadline is the time after which the customer refuses the delivery and to pay for his order.

In this problem setting, if the manufacturer is better off rejecting all of the orders, she can act so. There is no limit on the number of orders to be accepted. It is assumed that the production environment does not break down, and the preemption is not allowed. In addition, the machine can process only one order at a time. All input data are assumed to be known in advance. Then the order acceptance and scheduling (OAS) problem is to find the set S, and the sequence σ_s, when $TR(\sigma_s)$ is maximum. That is, the problem can be defined as $\max\limits_{\sigma_s, S \subseteq O} TR(\sigma_s)$.

3.2 Connection to the Literature

The study by Webster [32] relates the pricing and scheduling decisions in one problem, mainly emphasizing on capacity decisions. In OAS problem, we adjust the revenue gained by the manufacturer from a completed order according to the time it is completed, and we decide on our total revenue by considering the capacity constraints, and selecting the appropriate orders for processing. Therefore, total revenue depends on time and capacity as well as the demanded orders in our model.

The consideration of the latest acceptable lead time is one of the distinctions of the Charnirisaksul et al. [2] model from the other studies in the literature. Similarly, we have deadline for each of the customer orders defined as the latest time that the completed order should be delivered to the customer in the OAS problem. When make-to-order systems are considered, neglecting the setup times and allowing preemption are the weaknesses of the problem in Charnsirisakskul et al. [2]. In the OAS problem both of the concepts are handled.

The order acceptance problem with weighted tardiness objective which Slotnick and Morton [28] undertake in their study is similar to the OAS problem in the way it is defined. In addition to generalizing the Slotnick and Morton's problem, the OAS problem contains sequence-dependent setup times, release dates and deadlines as distinctive and complicating issues, which increase the reality of the problem especially when make-to-order systems are considered.

3.3 The Computational Complexity of the Problem

Obviously, the OAS problem is a combinatorial one, and even for very limited number of orders, the number of potential solutions is large. The OAS problem generalizes many well-known scheduling problems. Reduction of the OAS problem to the single machine total tardiness problem with sequence dependent setup times can be obtained as follows: Suppose all of the orders in set O will be accepted, and the deadlines are sufficiently large. Further suppose that the revenues e_i, $i \in O$, are fixed to 0, and the weights w_i, $i \in O$, are fixed to 1. Resultantly, the special case becomes a single machine total tardiness problem with sequence dependent setup times with the objective $\max \sum_{i=1}^{n} -T_i$, implying $\min \sum_{i=1}^{n} T_i$. Therefore, the problem reduces to a single machine scheduling problem with sequence dependent setup times with the objective of minimizing total tardiness. As mentioned earlier, the single machine total tardiness problem is proved to be NP-hard [4], and including the sequence dependent setup times brings more complicating issues to the problem. Hence the OAS problem is NP-hard.

As well as generalizing many single machine scheduling problems, The OAS problem is also a generalized form of Prize Collecting Traveling Salesman Problem with Time Windows (PCTSP with time windows). Reduction to PCTSP with time windows is done as follows: Suppose the weights w_i, $i \in O$, in the OAS problem are fixed to 0 so that the total revenue does not depend on the amount of tardiness any more. Next, the due dates are set to the corresponding deadlines, which makes the tardiness concept totally disappear. Since the due dates and deadlines are the same, we can get rid of the due dates. Therefore, we end up with a set of jobs having time windows $(r_i, \bar{d}_i)$ and revenues e_i. Let the processing times of all orders are set to 0. Then, the sequence dependent setup times between jobs are the same as the distances between the nodes in a TSP problem. Therefore the resulting special case of the problem is a PCTSP with time windows. A formal definition of the PCTSP with time windows can be found in [36].

The number of solutions in the solution space for the OAS problem can be calculated in the following manner: Suppose we have n incoming orders. Further suppose that we have an acceptance set A that contains i orders. The number of different sequences possible for this acceptance set is $i!$. The number of different selections for set A is $C(n, i)$. Therefore the total number of solutions in the solution space is $\sum_{i=1}^{n} C(n, i) \times i!$, including both feasible and infeasible solutions. This indicates that as i grows, applying an exact algorithm to this problem will be very time consuming.

A special case of the OAS problem can be obtained by fixing the processing sequence of the incoming orders. Suppose that S is the sequence that fixes the position of each order. Let $[i]$ be the order to be scheduled at position i. Therefore, the decision to be made is whether to accept this order or not. We have 2 different solutions for each order and hence the total number of solutions in the solution space is 2^n, including both feasible and infeasible solutions. Resultantly, this special case also has an exponentially growing solution space.

In the next section we develop a mathematical model for the OAS problem.

3.4 Mathematical Model

We formulate the order acceptance and sequencing problem as a mixed integer problem below:

Indices:

n: number of orders;

i, j, k: orders;

0, $n+1$: artificial orders at the first and the last positions respectively;

Problem Parameters:

O: the set of coming orders;

r_i: the release date of order i, $i \in O$;

$s_{k,i}$: the sequence dependent setup time for order i, given that the immediate predecessor of order i is order k, $i, k \in O$;

p_i: the processing time of order i, $i \in O$;

d_i: the due date of order i, $i \in O$;

$\bar{d}_i$: the deadline of order i, $i \in O$;

e_i: the maximum revenue that can be gained from order i, $i \in O$;

w_i: the weight for order i, $i \in O$.

Decision variables:

S: the set of selected orders;

σ_s: processing sequence of the selected order set S;

C_i: completion time of order i, $i \in S$;

T_i: tardiness of order i, $i \in S$;

R_i: revenue gained from order i, $i \in S$;

$TR(\sigma_S)$: The total revenue gained from processing set S, w.r.t. sequence σ_s;

$$y_{ij} = \begin{cases} 1 & \text{if order } i \text{ precedes order } j \\ 0 & \text{otherwise} \end{cases}$$

$$I_i = \begin{cases} 1 & \text{if order } i \text{ is selected} \\ 0 & \text{otherwise} \end{cases}$$

MILP:

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

$$s.t \quad \sum_{j=1, j\neq i}^{n+1} y_{ij} = I_i \qquad \forall i = 0, ..., n \tag{3.1}$$

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

$$C_i + (s_{ij} + p_j) \times y_{ij} + \bar{d}_i \times (y_{ij} - 1) \leq C_j \quad \forall i = 0, ..., n, \forall j = 1, ..., n+1, i \neq j \tag{3.3}$$

$$(r_j + p_j) \times I_j + s_{ij} \times y_{ij} \leq C_j \qquad \forall i = 0, ..., n, \forall j = 1, ..., n+1, i \neq j \tag{3.4}$$

$$T_i \geq C_i - d_i \qquad \forall i = 0, ..., n+1 \tag{3.5}$$

$$C_i \leq \bar{d}_i \times I_i \qquad \forall i = 0, ..., n+1 \tag{3.6}$$

$$T_i \leq (\bar{d}_i - d_i) \times I_i \qquad \forall i = 0, ..., n+1 \tag{3.7}$$

$$T_i \geq 0 \qquad \forall i = 0, ..., n+1 \tag{3.8}$$

$$R_i \leq e_i \times I_i - T_i \times w_i \qquad \forall i = 1, ..., n \tag{3.9}$$

$$R_i \geq 0 \qquad \forall i = 1, ..., n \tag{3.10}$$

$$C_0 = 0, \ C_{n+1} = \max_{i=1,...,n} \{\bar{d}_i\}, \qquad \forall i = 1, ..., n \tag{3.11}$$

$$I_0 = 1, \ I_{n+1} = 1 \tag{3.12}$$

$$I_i \in \{0, 1\}, y_{ij} \in \{0, 1\} \qquad \forall i = 1, ..., n \tag{3.13}$$

In this model, we define dummy orders, order 0 and order $n + 1$, where definitely, order 0 is assigned to the first position, and order $n + 1$ is assigned to the last position, whatever the sequence of orders in between is. Artificial orders 0 and $n + 1$ 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 maximum of deadlines of all jobs. Constraint sets (3.1) and (3.2) enforce that, if an order is accepted, this order precedes only one job and, is succeeded by only one job. If it is not accepted, it does not take place in the sequence. These constraints also prohibit the preemption of the jobs. Constraint set (3.3) implies that, if order j is preceded by order i, then the completion time of the order j should be larger than the completion time of order i plus the sequence dependent setup time between orders i and j, plus the processing time of order j. If order i does not precede order j in the sequence, $C_j \geq 0$ should be the only limit, and this constraint ensures this outcome. This set of constraints prevents any cycle formation in the sequence of the orders accepted. Constraint set (3.4) dictates, if an order j is accepted, and is preceded by order i in the schedule, then the completion time of order j should be greater than the release date of that order plus the sequence dependent setup time between orders i and j, plus the processing time of order j; and the completion time of the order j should be less than its deadline. In case where order i does not precede order j, the completion time of order j has looser bounds which are given by the release time of order j and its deadline. In case where order j is not accepted, it will not be in process, hence, $C_j = 0$. These constraints enable us to calculate the correct completion times of the orders. Constraint sets (3.5) and (3.6) are very general constraints and they put bounds on C_i and T_i. Constraint set (3.7) calculates the maximum tardiness for an accepted order. Constraint sets (3.8) and (3.10) ensure nonnegativity of P_i and T_i, implied by the definitions of these decision variables. Constraint set (3.9) calculates the revenue manufacturer can gain, when order i is accepted and incurs tardiness of T_i. Here, the revenue from an accepted order decreases as this order becomes tardy and becomes 0 at its deadline. Weight w_i is chosen accordingly to satisfy this assumption. For one unit time of tardiness, revenue decreases by w_i units. w_i is calculated by the formula $e_i/(\bar{d}_i - d_i)$ for each order i. Constraint sets (3.11) and (3.12) include some necessary initializations for dummy nodes 0 and $n+1$.

A feasible solution to our problem is represented as a chain of orders $0 \rightarrow O_1 \rightarrow O_2 \rightarrow \cdots \rightarrow O_l \rightarrow (n+1)$, where O_i represents the order scheduled at position i, in model MILP. Here, order 0 and order $(n+1)$ are dummy orders and $O_1, \ldots, O_l$ are the orders accepted such that O_i is scheduled as the i^{th} order to be processed.

Proposition 1 *A solution to model MILP is a feasible sequence for the OAS problem.*

Proof Suppose we have a solution S which is feasible to the model. Let the solution have k selected orders among the whole set of n orders. Further suppose that this solution has m

separate cycles. Since we assume that we have cycles in our solution S, we have a sequence $O_1 \to O_2 \to \cdots \to O_l \to (n+1)$ and additional m cycles formed by $k-l$ orders such as

$< O_{l+1} \to O_{l+2} \to \cdots \to O_{l+r} \to O_{l+1} >$

$< O_{l+r+1} \to O_{l+r+2} \to O_{l+r+1} >$

This is the case since by constraint sets (3.1) and (3.2), each selected order except order 0 is preceded by one order, and each selected order except order $(n+1)$ is succeeded by one order.

Since S is a feasible solution to the model, it should satisfy all of the constraints in the model. Now let's pick one of the cycles in solution S, and see whether it satisfies the constraint set (3.3) in the model shown below:

$$C_i + (s_{ij} + p_j) \times y_{ij} + \bar{d}_i \times (y_{ij} - 1) \leq C_j, \forall i = 0, \dots, n, \forall j = 1, \dots, n+1, i \neq j \tag{3.14}$$

Let the cycle under consideration be $O_{l+1} \to O_{l+2} \to \cdots \to O_{l+r-1} \to O_{l+r} \to O_{l+1}$. We write the constraints for $i = p+1,\dots,p+r$ and $j = i+1$. For any of the consecutive orders O_{l+i} and O_{l+i+1} in this cycle, the variable $y_{l+i,l+i+1} = 1$. Then the following constraints should be satisfied:

$$C_{l+1} + s_{l+1,l+2} + p_{l+2} \leq C_{l+2} \tag{3.15}$$

$$C_{l+2} + s_{l+2,l+3} + p_{l+3} \leq C_{l+3} \tag{3.16}$$

$$\vdots$$

$$C_{l+r-1} + s_{l+r-1,l+r} + p_{l+r} \leq C_{l+r} \tag{3.17}$$

$$C_{l+r} + s_{l+r,p+1} + p_{l+1} \leq C_{l+1} \tag{3.18}$$

Since the setup times $s_{ij} > 0$ and the processing times $p_i > 0$, we have the inequalities $C_{l+2} > C_{l+1}$, $C_{l+3} > C_{l+2}$,..., $C_{l+r} > C_{l+r-1}$,$C_{l+1} > C_{l+r}$ to hold. These inequalities can be given as $C_{l+1} > C_{l+r} > C_{l+r-1} > \cdots > C_{l+3} > C_{l+2} > C_{l+1}$, where we have $C_{l+1} > C_{l+r}$ and $C_{l+r} > C_{l+1}$ at the same time. This is a contradiction implying that this cycle is not feasible to the constraint set (3.1). This will also be valid for any cycle. Therefore, we conclude that any of the k cycles in solution S violates the constraint set (3.1). As a result, since we assumed that S satisfies all of the constraints of the model, this is a contradiction.

Hence, any solution feasible to model MILP contains no cycles and a solution to model MILP is a feasible sequence for the OAS problem. ■

Proposition 2 *Constraint set (3.3) of MILP model implies ATSP-MTZ subtour elimination constraints [19].*

Proof ATSP-MTZ subtour elimination constraints for the OAS problem can be written as follows:

$$I_j \leq u_j \leq (n-1)I_j \qquad \forall j = 1, \dots, n \tag{3.19}$$

$$u_j \geq (u_i + 1) - (n-1)(1 - y_{ij}) \qquad \forall i, j \geq 1, i \neq j. \tag{3.20}$$

where we have n orders, u_0=0 and the values of u_j variables represent the rank-order of the orders, that is, the value of u_1 is the number of order at the first position in the sequence.

Suppose we take any feasible solution to MILP model. Then, we know the values of decision variables y_{ij}'s, I_i's, C_i's, T_i's and R_i's for each order i. The corresponding decision variables for the rejected orders are known to be 0. Suppose C_i variables are ignored for rejected orders, and the remaining C_i variables which are known to be greater than 0 are sorted in increasing order. We call this sorted list as Q. Next, we can define u_j variables for all orders where $u_j = 0$ for each of the rejected orders. Index set of C_i variables in Q is equal to the processing sequence of accepted orders in the solution we take initially. Then, we can assign u_j variables so that, u_1=k , where k is the index of the first C_i in Q, $u_2 = l$, where l is index of the second C_i in Q and so on. Therefore the assigned u_j variables satisfy the constraint sets (3.19) and (3.20). Hence, the constraint set (3.3) of MILP model implies ATSP-MTZ subtour elimination constraints. ■

3.5 Finding Upper Bound for the Problem

As mentioned earlier, the OAS problem is NP-hard and we need to develop different methods to solve the problem for large instances. In this section, we propose 3 methods for generating upper bounds for the OAS problem.

3.5.1 LP Relaxation

An easy-to-find upper bound for the OAS problem is the LP relaxation solution of the MILP model. Major problem in the MILP model is that LP relaxation solution of the problem is not tight, and hence the calculated upper bounds are large. When we relax the integrality constraints for both I_i and y_{ij} variables in the MILP model and solve the problem, the resulting problem is easy and y_{ij}'s, the sequencing variables, are divided into many parts summing up to 1 for each row and each column in the solution. This is because the model has a tendency in choosing the sequence dependent setup times as small as possible and setting the completion times small to avoid from being tardy or rejecting some of the orders because of the deadlines. Resultantly, the model is generally able to accept all of the orders in the LP relaxation. The upper bound generated in this solution is simply the sum of the maximum revenues of all orders and named as UB_{LP} in the following sections.

3.5.2 LP Relaxation with Valid Inequalities

In order to improve the LP relaxation upper bound, we first remove the constraint $C_{n+1} = \max\limits_{i=1,\ldots,n} \bar{d}_i$ and we add the following three valid inequalities to the problem:

$$C_{n+1} \leq \max_{i=1,\ldots,n} \bar{d}_i; \tag{3.21}$$

$$C_{n+1} \geq \min_{i=1,\ldots,n} r_i + \sum_{i \in O} [(p_i + minsetup_i) \times I_i]; \tag{3.22}$$

$$C_i \geq (r_i + minsetup_i + p_i) \times I_i, \qquad \forall i = 1, \ldots, n; \tag{3.23}$$

where $minsetup_i$ is the minimum sequence dependent setup time for order i in case of processing it. Valid inequalities (3.21), (3.22) and (3.23) improve the upper bound and prevent the model to break the variables y_{ij} into too many small pieces. When we add the integrality constraint for I_i's, the acceptance variables, problem is still easy to solve, and the bound is further improved or remains the same for the OAS problem. The upper bound generated by this method is called UB_{LPVI} in the next sections.

We present the % improvements achieved by employing the proposed valid inequalities in Tables 5.1, 5.2 and 5.3 and analyze the improved cases in detail in Chapter 5, where we describe the computational experiments.

3.5.3 Lagrangean Relaxation

An observation on the original model is that the constraint set (3.3) includes the complicating constraints that undertake the most difficult job: subtour elimination. When we remove the constraint set (3.3), the model can be solved easily. However, the resulting solution includes subtours and hence it is infeasible to the original problem. Below, a Lagrangean relaxation (LR) approach is proposed for the OAS problem where constraint set (3.3) is relaxed and a penalty term is added to the objective function for violation.

Relaxation is handled as follows: LR problem is formed by multiplying the constraint set (3.3) with dual variables λ_{ij}, where $\lambda_{ij} \geq 0$, and by adding these expressions as penalty terms to the objective function.

LR:

$$z(\lambda) = \max(\sum_{i=1}^{n} R_i - \sum_{i=0}^{n} \sum_{j=1, j\neq i}^{n+1} \lambda_{ij}[(C_i + (s_{ij} + p_j) \times y_{ij} + \bar{d}_i \times (y_{ij} - 1)) - C_j]) \tag{3.24}$$

$$\text{s.t } (1), (2), (4) - (13) \text{ and } \lambda_{ij} \geq 0 \ \forall i = 0, \ldots, n, j = 1, \ldots, n+1$$

Then, the Lagrangian dual problem is

$$z^* = \min z(\lambda)$$

$$\text{s.t } \lambda \geq 0$$

The problem of applying the Lagrangean relaxation method to the MILP model by relaxing constraint set (3.3) is that, constraint set (3.3) becomes redundant when $y_{ij} = 0$. When $y_{ij} = 0$; we end up with additional positive term $\lambda_{ij} \times (C_j - C_i + \bar{d}_i)$ in the objective function for each (i, j) pair. This results in superoptimal infeasible solutions for the OAS problem. Hence, we cannot use Lagrangean relaxation method directly for the OAS problem because of the Big-M problem.

3.6 Dominance Relations for the Problem

Dominance relations are established by using the problem characteristics to make inferences about the optimal solution of the underlying problem. Researchers use dominance properties widely in proposing efficient solution methods to the scheduling problems by eliminating some parts of the solution space, or by characterizing the optimal solutions. In this section, we propose some dominance properties on order acceptance and sequencing that are designed for the OAS problem. The rest of the properties that we developed for order sequencing decision are discussed in Chapter 6.

Definition 1 *Minimum required time for completing order i is equal to $\bar{C}_i = r_i + \min\limits_j s_{j,i} + p_i$. The difference between $\bar{d}_i$ and $\bar{C}_i$ is defined as the slack time of order i, δ_i.*

Property 1 *Suppose we have two orders, i and m. If $\bar{C}_i > \bar{d}_m$, then order m cannot be placed after order i.*

Now, we present the dominance properties that we applied for the OAS problem. In this scope, we generate a new sequence S' by changing some of the order positions in a feasible sequence S. Next, we decide on which sequence dominates the other one by comparing two sequences. Note that, in S', $[i]$ still represents the order index of the i^{th} position in S.

3.6.1 Dominance properties dealing with acceptance decisions

1. Let L be the set of orders in the candidate acceptance list. For each of the orders $l \in L$, if $\max\limits_{l \in L} \bar{d}_l - \min\limits_{l \in L} r_l < \sum\limits_{l \in L} (\min\limits_{k \in J} s_{kl} + p_l)$, then not all of the orders in the set L can be accepted at the same time. In order to get a feasible set from this set L, one or more of the orders in set L should be rejected. This property applies to the cases with two as below:

2. **Two-order case of property 1:** Assume we have two orders j and k. If $\max\limits_{j,k}(\bar{d}_j, \bar{d}_k) < \max\limits_{j,k}(r_j + minsetup_j + p_j, r_k) + p_k + minsetup_k)$ and $\max\limits_{j,k}(\bar{d}_j, \bar{d}_k) < \max\limits_{j,k}(r_k + minsetup_k + p_k, r_j) + p_j + minsetup_j)$ then orders j and k cannot be accepted in the same feasible schedule.

   Additionally, if $\max\limits_{j,k}(\bar{d}_j, \bar{d}_k) - \min\limits_{j,k}(r_j, r_k) < minsetup_j + minsetup_k + p_j + p_k$, then order j and order k cannot be accepted in the same feasible schedule.

3.6.2 Dominance property dealing with sequencing decisions

Suppose we want to place order m right after order i. Minimum completion time for order i is $\bar{C}_i = r_i + \min\limits_{k \in J} s_{k,i} + p_i$. If $\max(\bar{C}_i, r_m) + s_{i,m} + p_m > \bar{d}_m$, then order m cannot be scheduled right after order i. A similar, but looser dominance property is as follows: Suppose we want to place order m not right after, but after order i. If $\max(\bar{C}_i, r_m) + \min\limits_{k \in J} s_{k,m} + p_m > \bar{d}_m$, then order m cannot be scheduled after order i.

These three dominance properties were coded in Java and fed into the MILP model as additional constraints. Next, both the original MILP model with dominance properties added and the LP relaxation of the MILP model with dominance properties added were solved with the data generated as mentioned in Chapter 5. However, the upper bounds for the original MILP model (UB_{MILP}) and the LP relaxation (UB_{LP}) were not improved for both of the trials. Therefore, the proposed dominance properties are not useful for the OAS random data described in detail in Chapter 5, but they are still promising to be used in other types of data or in an exact algorithm such as branch and bound algorithm.

Chapter 4

HEURISTIC SOLUTION APPROACHES

As mentioned in Chapter 3, the OAS problem is NP-hard. For this reason, CPLEX is capable of solving only small size OAS problem instances via MILP model in reasonable time limits. For industrial-size problems, we need to develop other methods that efficiently solve the problem requiring considerably small CPU times. Below we present the algorithms and heuristics developed for the OAS problem.

4.1 Iterative Sequence First- Accept Next(ISFAN) Algorithm

In this section, we describe the ISFAN algorithm for solving the OAS problem. ISFAN is an iterative heuristic algorithm that uses a rule designed for the OAS problem to give acceptance-rejection decisions and Simulated Annealing (SA) algorithm concepts to handle the sequencing of the selected orders. SA is a metaheuristic method that uses thermodynamic concepts. The reason why we use its ideas in ISFAN is an important aspect in its nature: it is good at diversifying and less likely to get stuck in the local optimum than other metaheuristics. Also, SA is proven to be succesful in solving combinatorial optimization problems in the literature [29]. Similar to SA algorithm, ISFAN accepts a new solution even its objective value is worse than before, with a probability given by $\exp(-\delta f/T)$, where δf is the increase in the objective function value and T is the current temperature. By this procedure, ISFAN handles the sequencing part of the problem. For order acceptance-rejection decisions, ISFAN uses the revenue-load ratio which is equal to $e_j/(p_j+minsetup_j)$. At each ISFAN iteration, first the sequencing of orders is handled with SA and the order with the lowest revenue-load ratio is rejected at the end of the iteration. The algorithm is run until a feasible schedule is found. Once a feasible schedule is found for the set of accepted orders, algorithm tries to find the best schedule for this set. In these attempts to find the best schedule, local search methods such as the best order exchange and the best order insertion are applied. Below we give the ISFAN and the local search algorithms in detail.

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 jobs 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.
```

The flow chart for the ISFAN algorithm is given in Figure 4.1.

Since ISFAN is a hybrid metaheuristic, the computational complexity of the algorithm cannot be expressed in terms of $O(n)$ notation. The complexity of ISFAN depends on many developer-defined parameters such as the number of iterations, number of iterations without improvement and the initial temperature of the annealing procedure. In the worst case, ISFAN algorithm may have exponential run time depending on these parameters.

4.2 Constructive Heuristics

In addition to the hybrid-metaheuristic algorithm ISFAN, we propose two constructive heuristics considering the special characteristics of the problem. Constructive heuristics are designed for generating high quality solutions in a short time compared to other more sophisticated exact or metaheuristic algorithms. Below we describe the procedures for each of the constructive algorithms.

4.2.1 Dynamic Release First- Sequence Best (d-RFSB) Heuristic

The main idea of the d-RFSB heuristic is to evaluate the availability of the orders by checking the release dates and deadlines, and then accepting and scheduling the order that has the highest $\frac{e_i}{(p_i+s_{previousJob,i})}$ ratio among the available orders. In d-RFSB heuristic, acceptance decisions are given dynamically. Time is evolved by one unit until an available order is found. Once an available order is scheduled, the completion time of this order is calculated and time is evolved up to this time. An unscheduled order whose deadline is passed is automatically counted as rejected by the heuristic. Hence, the d-RFSB heuristic has a procedure for accepting the orders rather than rejecting them as in ISFAN algorithm. Below we give the algorithm for the d-RFSB heuristic.

```
d-RFSB Algorithm
Step 1. Read the input data.
Step 2. Set the time, $t_{current} = 0$, $previousJob = 0$, $tempAccept = 0$, $count = 1$,
$ratioCount = 1$, $maxDue = 1$, $ind = true$;
Step 3. while ($ratioCount \neq 0$), do the following:
    3.1. If $t_{current} \geq maxDue$, terminate the algorithm.
    3.2. Count the number of unscheduled jobs.
     3.2.1. if number of unscheduled jobs=0, terminate the algorithm
    3.3. Find the maximum due date from the due dates of the set of
    unscheduled orders $U$.
    3.4. If $((\max(t_{current}, r_{tempAccept}) + s_{previousJob,tempAccept} + p_{tempAccept}) \geq maxDue)$,
    terminate the algorithm.
    3.5. while ($x = true$), do the following (while1):
     3.5.1. for each order $i$ in set $U$, if $r_i \leq t_{current}$ and $d_i \geq t_{current}$,
     calculate the ratio $ratio_i = \frac{e_i}{(p_i+s_{previousJob,i})}$, $ratioCount = ratioCount + 1$;
     3.5.2. If $ratioCount = 0$, $t_{current} = t_{current} + 1$, $x = true$; else, $x = false$;
    3.6. while $acceptedJob = 0$ (while2),
     3.6.1. If $t_{current} \geq maxDue$, terminate the algorithm,
3.6.2. Find the order $j$ having the maximum ratio, and assign $tempAccept = j$,
3.6.3. if $((\max(t_{current}, r_{tempAccept}) + s_{previous_job,tempAccept} + p_{tempAccept}) \geq maxDue)$
break while2;
3.6.4. if $(\max(t_{current}, r_{tempAccept}) + s_{previousJob,tempAccept} + p_{tempAccept} \leq \bar{d}_{tempAccept})$,
then do the following assignments:
    $acceptedJob = tempAccept$,
    $Acceptance[acceptedJob] = 1$,
    $t_{current} = \max(t_{current}, r_{tempAccept}) + s_{previousJob,tempAccept} + p_{tempAccept}$,
    $Completion_{acceptedJob} = t_{current}$,
    $previousJob = acceptedJob$,
    $sequence[count] = acceptedJob$;
    $count = count + 1$;
3.6.5. Else, $ratio_{tempAccept} = 0$,
    $Acceptance_{tempAccept} = 1$,
    $tempAccept = 0$, break while2;
3.7. $ratioCount = 1, x = true, acceptedJob = 0$;
3.8. return to while.
```

The computational complexity of the d-RFSB heuristic is $O(n^2)$ since we make at most n passes in the algorithm, and at each pass, we deal with $k-1$ orders where k is the number of unscheduled orders beginning from n.

4.2.2 Modified ATCS (m-ATCS) Heuristic

m-ATCS heuristic uses a slightly modified version of ATCS priority index mentioned in [15]. m-ATCS heuristic has a similar structure as d-RFSB heuristic. The priority index that m-ATCS heuristic uses is $\frac{e_i}{p_i} \times \exp(\frac{\max(d_i - p_i - t_{current}, 0)}{\bar{p}}) \times \exp(\frac{-s_{previousJob,i}}{\bar{s}})$ to select the order that will be sequenced next. $\bar{p}$ and $\bar{s}$ represent the average processing time and average setup time.

m-ATCS heuristic considers the information on setup times and processing times of other orders by adding $\bar{s}$ and $\bar{p}$ terms to the priority index, and on approximate potential tardiness's of orders by the term $\max(d_i - p_i - t_{current}, 0)$. Below we give the algorithm for the m-ATCS heuristic.

Algorithm

```
Step 1. Read the input data.
Step 2. Calculate the average processing time and average setup time.
Step 3. Set the time, $t_{current} = 0$, $previousJob = 0$, $indexCount = 1$, $tempAccept = 0$,
$count = 1$, $maxDue = 1$, $ind = true$;
Step 4. while ($indexCount \neq 0$), do the following:
    4.1. If $t_{current} \geq maxDue$, terminate the algorithm.
    4.2. Count the number of unscheduled jobs,
     4.2.1. if number of unscheduled jobs=0, terminate the algorithm.
    4.3. Find the maximum due date from the due dates of the set of unscheduled
    orders $U$.
    4.4. If $((\max(t_{current}, r_{tempAccept}) + s_{previousJob,tempAccept} + p_{tempAccept}) \geq maxDue)$,
    terminate the algorithm.
    4.5. while ($x = true$), do the following (while1):
     4.5.1. For each order $i$ in set $U$, if $r_i \leq t_{current}$ and $d_i \geq t_{current}$,
            calculate the index $index_i = \frac{e_i}{p_i} \times \exp(\frac{\max(d_i - p_i - t_{current}, 0)}{\bar{p}}) \times \exp(\frac{-s_{previousJob,i}}{\bar{s}})$,
            $indexCount = indexCount + 1$;
     4.5.2. If $indexCount = 0$, $t_{current} = t_{current} + 1$, $x = true$; else, $x = false$;
    4.6. While $acceptedJob = 0$ (while2),
     4.6.1. If $t_{current} \geq maxDue$, terminate the algorithm,
     4.6.2. Find the order $j$ having the maximum ratio, and assign $tempAccept = j$,
     4.6.3. if $((\max(t_{current}, r_{tempAccept}) + s_{previous_job,tempAccept} + p_{tempAccept}) \geq maxDue)$
            break while2;
     4.6.4. if $(\max(t_{current}, r_{tempAccept}) + s_{previousJob,tempAccept} + p_{tempAccept} \leq \bar{d}_{tempAccept})$,
            then do the following assignments:
           $acceptedJob = tempAccept$,
           $Acceptance[acceptedJob] = 1$,
           $t_{current} = \max(t_{current}, r_{tempAccept}) + s_{previousJob,tempAccept} + p_{tempAccept}$,
           $Completion_{acceptedJob} = t_{current}$,
           $previousJob = acceptedJob$,
           $sequence[count] = acceptedJob$;
```

```
        $count = count + 1$;
    4.6.5. Else, $ratio_{tempAccept} = 0$,
        $Acceptance_{tempAccept} = 1$,
        $tempAccept = 0$, break while2;
  4.7. $indexCount = 1, x = true, acceptedJob = 0$;
  4.8. return to while.
```

The computational complexity of the m-ATCS heuristic is $O(n^2)$ calculated as the complexity of d-RFSB heuristic.

4.3 MILP Heuristic (MILPH)

MILPH is a heuristic method proposed to generate good solutions to the OAS problem. In MILPH, we feed a sequence (either feasible or infeasible) to the mathematical model and run the model for a specific amount of time. The resulting solution is a lower bound for the OAS problem. The sequence is added in the following manner: In addition to the MILP model, constraint set

$$y_{ij} \geq I_i + I_j - 1 \tag{4.1}$$

is added for each pair of consecutive orders in sequence S. This constraint set assures that if the consecutive orders i and j in input sequence S are both accepted, they have to be scheduled consecutively in the optimal sequence. If all of the orders are eligible for acceptance at the same time, then the remaining problem is easy: Given the sequence, calculate the tardiness and the total profit. Otherwise, if one of each consecutive orders is rejected, then the remaining problem is again the OAS problem with $n/2$ orders. In this case, the remaining orders have no sequencing restrictions among them since the constraint set (4.1) puts limits on only the consecutive orders in sequence S.

The input sequence for the MILPH is found as follows: First of all, the complicating constraint set of the model, constraint set (3.3) is relaxed and the model is solved by CPLEX. If the resulting solution does not contain any subtours, then this sequence is directly input into the MILP model in which the constraint set (4.1) is added and solved by CPLEX solver with 1 hour time limit. Otherwise if the solution contains subtours; then subtours are identified and eliminated. The subtour elimination phase is handled by adding the constraints,

$$\sum_{i \notin S} I_i - z_S \geq 0; \tag{4.2}$$

$$\sum_{i \in S, j \notin S} y_{ij} - z_S \geq 0; \tag{4.3}$$

$$z_S - I_i \geq 0, \forall i \notin S \tag{4.4}$$

to the MILP model where constraint set (3.3) is relaxed and z_S is a binary variable generated for each subtour elimination iteration. These constraints force each order i such that $i \notin S$ to enter S, so that a sequence S that contains all of the orders is formed at the end. The elimination phase is terminated when a single sequence S of all orders is found. The sequence S is then fed into the original MILP model by adding the constraint set (4.1) for all consecutive orders and the problem is solved by CPLEX with 1 hour time limit. Next, local improvement methods mentioned in Section 4.4 are applied to the resulting solution. We present the results for the MILPH in Chapter 5.

4.4 Local Improvement Methods

In ISFAN algorithm, orders are rejected one by one in each rejection iteration after a number of neighborhood moves are performed to improve the current sequence. Once an order is rejected, there is no procedure for adding that rejected order back to the final solution in ISFAN algorithm. For this reason, ISFAN may yield to low-quality results when a few orders are rejected mistakenly. In order to overcome this issue and to improve the solutions obtained by the ISFAN algorithm, we employ order insertion and order exchange algorithms. We apply the local improvement methods to d-RFSB, m-ATCS and MILPH heuristics as well as ISFAN algorithm for a fair comparison of the methods. Below we give the algorithms for local improvement methods in detail.

4.4.1 Order Insertion Algorithm

The idea of the order insertion algorithm employed in this study is as follows: Given a feasible sequence of accepted orders, and the set of rejected orders we try to insert these rejected orders into the existing feasible schedule. Each orders in rejected orders set are inserted in each position and the feasible insertion that results in highest revenue is performed. Once an insertion move is performed, the existing feasible schedule is updated and procedure is continued until no improvement is achieved. Order insertion method allows us to insert some rejected orders that improve the current objective function value when added to the current sequence without violating feasibility. The algorithm steps are given below:

Order Insertion Algorithm

```
Step 1. Read the initial sequence obtained by the ISFAN algorithm.
Step 2. Read the set of rejected jobs in that solution.
Step 3. While an improvement on the objective value is achieved at the last iteration:
    3.1. Perform the following loop for $k$=number of rejected jobs times:
     3.1.1. Perform the following loop for $m$=number of positions at the initial sequence times:
        3.1.1.1. Insert the $i^{th}$ ($i$=0,...,$k-1$) element of the rejected jobs to the $j^{th}$ (j=0,...,$m-1$) position of the initial sequence.
        3.1.1.2. Calculate $C_l$'s, $T_l$'s and $R_l$'s for this new sequence.
        3.1.1.3. If (new sequence is feasible) and ($newrevenue >$ best revenue):
          3.1.1.3.1. initial sequence=new sequence.
          3.1.1.3.2. best revenue= new revenue.
    3.2. Update the set of rejected jobs by extracting the inserted job.
    3.3. Return to Step 3.
```

The flow chart for the insertion algorithm is given in Figure 6.1.

The computational complexity of the insertion heuristic is $O(n^2)$ since we have $n-a$ rejected orders to insert to a positions where a is the number of accepted orders, and in the worst case we have to make $(n-a)$ passes.

4.4.2 Order Exchange Algorithm

In the order exchange algorithm, given a feasible sequence of accepted orders, and the set of rejected orders we try to exchange these rejected orders by one of the orders in the existing schedule. Each order in rejected orders set is exchanged by each order in the existing feasible sequence and the feasible exchange that results in highest revenue is performed. Once an exchange move is performed, the existing feasible schedule is updated and procedure is continued until no improvement is achieved. Order exchange method allows us to exchange some of the rejected orders that improve the current objective function value when exchanged with some of the orders currently in the sequence without violating feasibility. The algorithm steps are given below in detail:

Order Exchange Algorithm

```
Step 1. Read the initial sequence obtained by the ISFAN algorithm.
Step 2. Read the set of rejected jobs in that solution.
Step 3. While an improvement on the objective value is achieved at the last iteration:
    3.1. Perform the following loop for $k$=number of rejected jobs times:
     3.1.1. Perform the following loop for $m$=number of positions at the initial sequence times:
        3.1.1.1. Exchange the $i^{th}$ ($i$=0,...,$k-1$) element of the rejected orders
        with the order at $j^{th}$ (j=0,...,$m-1$) position at the initial sequence.
        3.1.1.2. Calculate $C_l$'s, $T_l$'s and $R_l$'s for this new sequence.
        3.1.1.3 If (new sequence is feasible) and ($newrevenue >$ best revenue):
          3.1.1.3.1. initial sequence=new sequence.
          3.1.1.3.2. best revenue= $newrevenue$.
    3.2. Update the set of rejected jobs by exchanging the newly added order by the
    extracted order.
    3.3. Return to Step 3.
```

The flow chart for the exchange algorithm is given in Figure 4.3.

The computational complexity of the exchange heuristic is $O(n^2)$ calculated as the complexity of the insertion heuristic.

The local improvement methods can be applied to existing feasible schedules sequentially as well as separately. In our test experiments, we found that the best combination of the local improvement methods is local insertion-local exchange performed consecutively. The result is not surprising since when an insertion move is performed, we add a new order to the acceptance list, and potentially increase the total revenue. However, we can add an order to the list only if we reject another one with the exchange move, thus the exchange move has a lower potential of increasing the total revenue than the insertion move. Still, if the exchange moves are performed after the possible insertion moves are performed, we may have the possibility of adding a mistakenly rejected order and increase the total revenue. Hence, we applied the local improvement methods to all of the heuristics' solutions in this prespecified order.

We coded the proposed heuristics and tested them on two types of data structure. We give the results in Chapter 5.

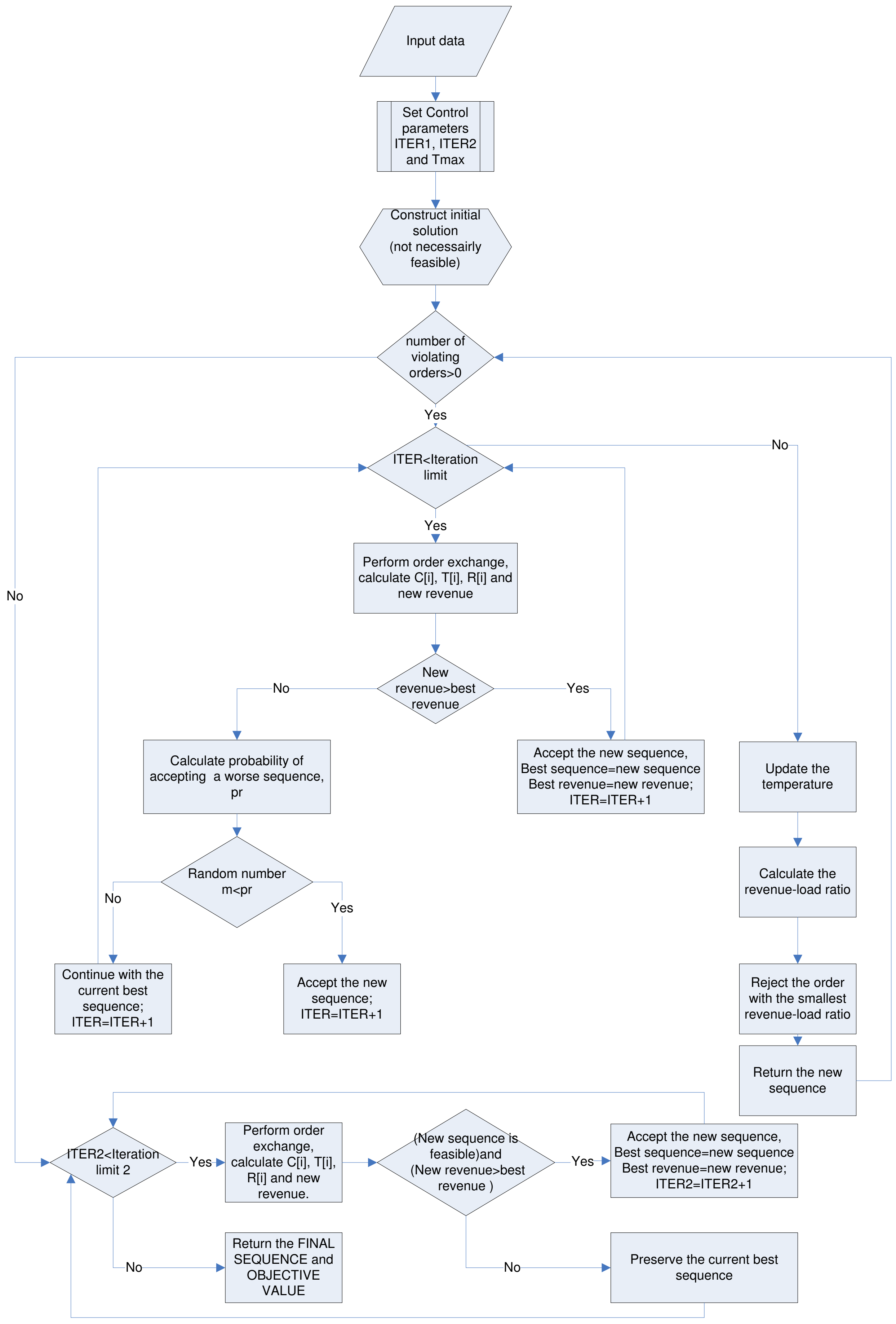
Input data
Set Control parameters ITER1, ITER2 and Tmax
Construct initial solution (not necessairly feasible)
number of violating orders>0
Yes
No
ITER<Iteration limit
Yes
No
Perform order exchange, calculate C[i], T[i], R[i] and new revenue
New revenue>best revenue
No
Yes
Calculate probability of accepting a worse sequence, pr
Accept the new sequence, Best sequence=new sequence Best revenue=new revenue; ITER=ITER+1
Update the temperature
Random number m<pr
No
Yes
Calculate the revenue-load ratio
Continue with the current best sequence; ITER=ITER+1
Accept the new sequence; ITER=ITER+1
Reject the order with the smallest revenue-load ratio
Return the new sequence
ITER2<Iteration limit 2
Yes
Perform order exchange, calculate C[i], T[i], R[i] and new revenue.
(New sequence is feasible)and (New revenue>best revenue)
Yes
Accept the new sequence, Best sequence=new sequence Best revenue=new revenue; ITER2=ITER2+1
No
Return the FINAL SEQUENCE and OBJECTIVE VALUE
No
Preserve the current best sequence

Figure 4.1: Flow chart for ISFAN algorithm.

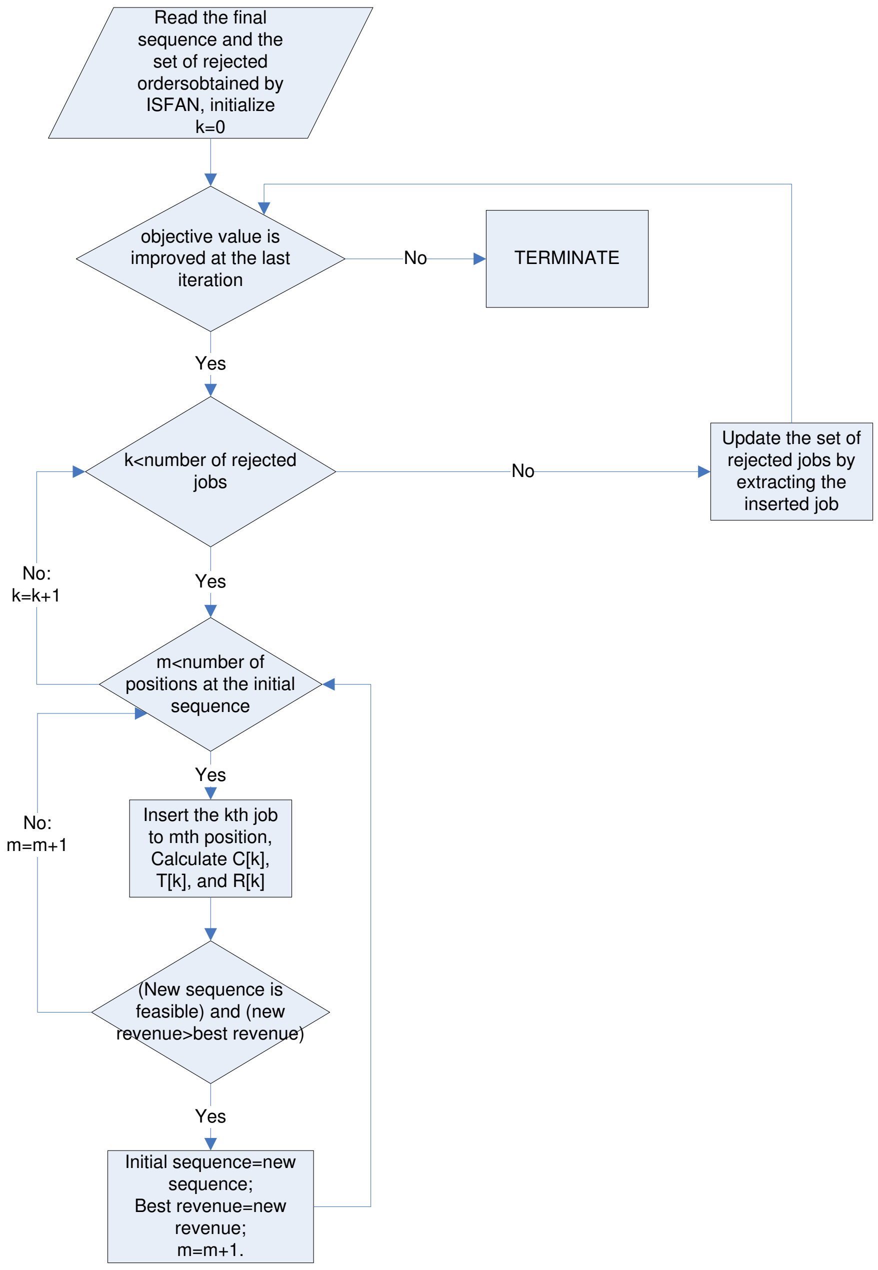
Read the final sequence and the set of rejected ordersobtained by ISFAN, initialize k=0
objective value is improved at the last iteration
No
TERMINATE
Yes
k<number of rejected jobs
No
Update the set of rejected jobs by extracting the inserted job
Yes
No: k=k+1
m<number of positions at the initial sequence
Yes
No: m=m+1
Insert the kth job to mth position, Calculate C[k], T[k], and R[k]
(New sequence is feasible) and (new revenue>best revenue)
Yes
Initial sequence=new sequence; Best revenue=new revenue; m=m+1.

Figure 4.2: Flow chart for Insertion algorithm.

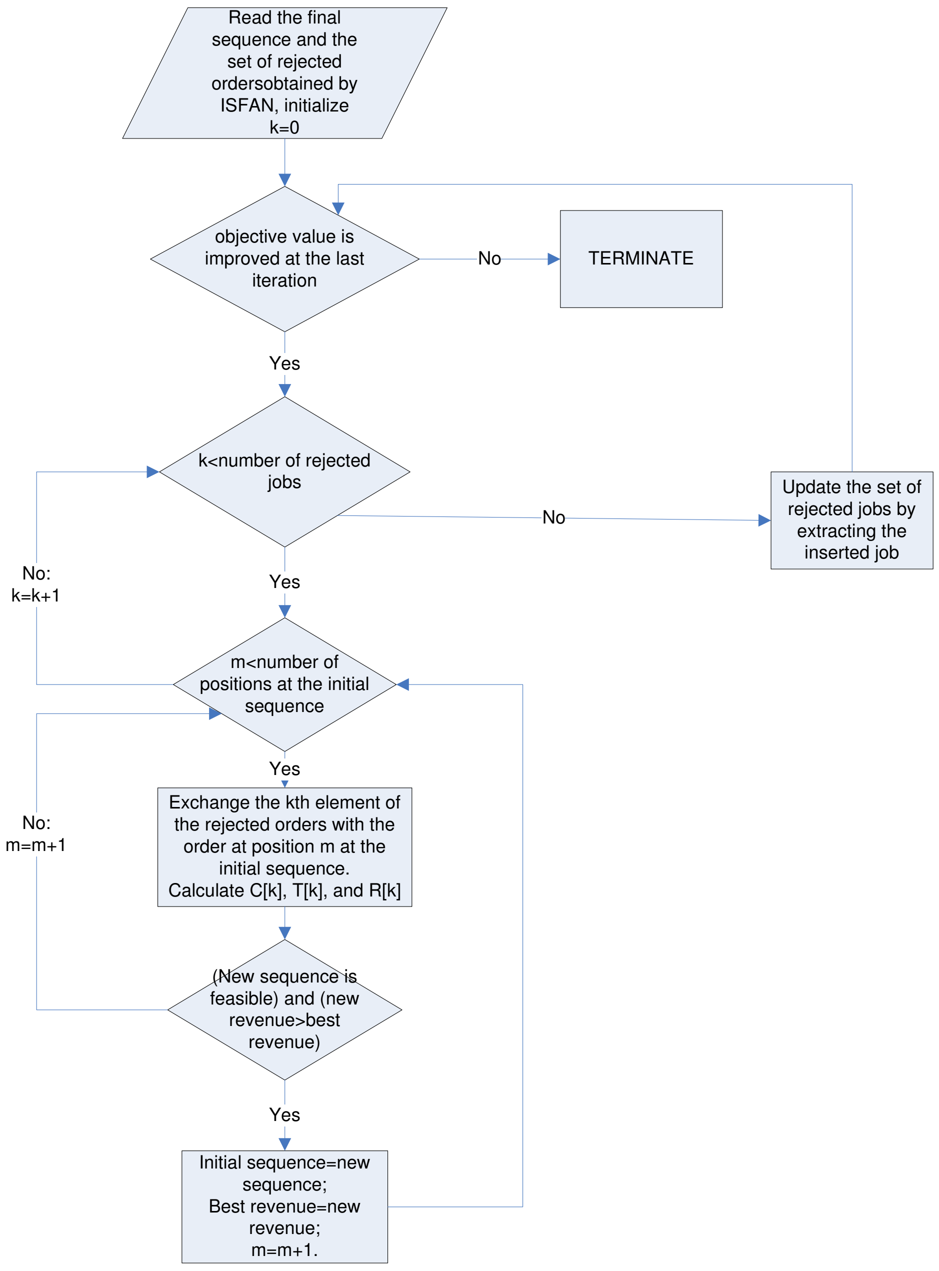
Read the final sequence and the set of rejected ordersobtained by ISFAN, initialize k=0
objective value is improved at the last iteration
No
TERMINATE
Yes
k<number of rejected jobs
No
Update the set of rejected jobs by extracting the inserted job
No: k=k+1
Yes
m<number of positions at the initial sequence
Yes
Exchange the kth element of the rejected orders with the order at position m at the initial sequence. Calculate C[k], T[k], and R[k]
No: m=m+1
(New sequence is feasible) and (new revenue>best revenue)
Yes
Initial sequence=new sequence; Best revenue=new revenue; m=m+1.

Figure 4.3: Flow chart for Exchange algorithm.

Chapter 5

COMPUTATIONAL STUDIES

We performed computational experiments to analyze the performance of the the proposed algorithms. In this chapter, we first describe the random data generation phase of the experiments since the OAS problem is not studied before in the literature, and hence there is no available data library for use in our computational experiments. Next, the data of a special case, Slotnick & Morton's data [28], is described. In Section 5.2, the computational experiments are discussed in detail.

5.1 Data Generation

5.1.1 Random Data

In order to test the proposed model and the heuristics, random test instances are generated varying in parameter values and problem sizes. The following problem parameters are integer numbers which were generated randomly from a uniform distribution in the following intervals; release dates r_i: [0,$p_T \times \tau$] where p_T is the total processing time of all orders and τ is the tardiness factor, processing times p_i: [1,20], sequence dependent setup times s_{ij}: [1,10], and revenues e_i: [1,20]. The due dates are generated so that for each order i, due date d_i is in the interval of $(p_T + \max\limits_j s_{ij} + r_i) \times [1 - \tau - R/2, 1 - \tau + R/2]$ and integer where R represents the due date range respectively. The deadlines are generated from the formula $\bar{d}_i$=$d_i + R \times p_i$. The weight w_i is calculated by the formula w_i=$e_i/(\bar{d}_i - d_i)$. In this setting, revenue gained from an order i becomes 0 at its deadline $\bar{d}_i$. The values used for τ and R are 0.1, 0.3, 0.5, 0.7 and 0.9. For each possible combination of τ and R, 10 instances were generated and the average results are reported.

In this study, $\tau = \{0.1, 0.3, 0.5\}$ have been found compatible with $R = \{0.1, 0.3, 0.5, 0.7, 0.9\}$, and a total of $3 \times 5 \times 10$=150 test instances are generated for these parameter combinations for each n. Similarly, $\tau = 0.7$ is compatible with $R = \{0.5, 0.7, 0.9\}$, and hence 3×10=30 instances are generated for each n. Finally, $\tau = 0.9$ is compatible only with $R = 0.9$, and therefore 10 instances are generated for each n for $\tau = 0.9$, $R = 0.9$ combination. Resultantly, the number of the test instances for each n is 190, which sums up to 1140 for 6 different n values where $n = 10, 15, 20, 25, 50, 100$. The results of the computational experiment are presented in Section 5.2.

5.1.2 Slotnick & Morton's (S&M) Data

The S& M data is taken from the authors, which exist for only the problem sizes of 10 and 20. In S&M data, the weights and the processing times were generated from a uniform distribution (0, 1), then adjusted by a constant, in order to vary the difficulty of the problem in terms of profit margin and order size. Revenue was randomly generated from a lognormal distribution with an underlying normal distribution with mean 0 and standard deviation 1. Due dates were generated from a uniform distribution, adjusted to the magnitude of processing times in a particular problem.

5.2 Computational Experiments

5.2.1 Computational Platform

All of the runs throughout these computational experiments are performed on a workstation with an Intel Xeon processor, 3.40 Ghz speed, and 8 GB of RAM.

In MILP, LP relaxation, LP relaxation with valid inequalities and MILPH runs, we have used ILOG CPLEX 10.0 and Java API of ILOG CPLEX Concert Technology. Time limit is 3600 seconds for CPLEX runs except for the runs with S&M data whose time limit is 600 seconds. All of the solution methods developed within the scope of this thesis are coded in Java.

5.2.2 Upper Bound Comparisons

In Section 3.5, we propose three methods for finding an efficient upper bound for the OAS problem. However, the third method, Lagrangean relaxation, does not work well with the OAS problem. Below in Tables 5.1, 5.2 and 5.3, we compare LP relaxation bound (UB_{LP}) and LP relaxation bound strengthened with valid inequalities (UB_{LPVI}) for $n \in \{10, 15, 20, 25, 50, 100\}$. In Tables 5.1, 5.2 and 5.3, we see that valid inequalities strengthen the upper bound by 1 to 2% on the average. In addition, we observe that while there are minimal or no improvements for $\tau = 0.1, 0.3$ cases; the improvements are noticeable for $\tau = 0.5$, $R = 0.1$ case (7 to 12% on the average) and $\tau = 0.7$, $R = 0.5$ case (3 to 7% on the average). We can conclude that valid inequalities have more potential to strengthen the UB_{LP} for the instances having τ large and R small with this evidence. Since the highest improvement is achieved when $\tau = 0.5$, $R = 0.1$ case is tested, we can arrive at the conclusion that as the ratio between τ and R increases, the likelihood of the UB_{LPVI} to improve increases. $\tau = 0.3$, $R = 0.1$ and $\tau = 0.7$, $R = 0.5$ cases also support this conclusion.

As it is obvious from Tables 5.1, 5.2 and 5.3 that UB_{LPVI} dominates the UB_{LP}, we use UB_{LPVI} for measuring the heuristic algorithms' performances in the following sections.

5.2.3 Experiments with MILP model

For each (n, τ, R) parameter combination, we solved 10 instances of the OAS problem, generated randomly as explained above. The comparison of upper bounds generated by MILP (UB_{MILP}) and LP relaxation with valid inequalities (UB_{LPVI}) can be found in Table 5.4 for n=10, 15. We reported the maximum, minimum and average deviations of MILP objective function values from the UB_{LPVI} and the CPU times in Tables 5.7 and 5.8 for n=10, 15. From the CPU time results in Table 5.8, it is seen that the hardest problem instances for MILP are generated when $\tau = 0.1$ and $R = 0.1, 0.3$, and $\tau = 0.3$ and $R = 0.1, 0.3$. It is still possible to solve the OAS problem when $\tau = 0.5$, 0.7 and 0.9 when $n = 10$. For the problem sizes above 15, it is impossible to find the optimal solution within the time limit of 3600 seconds in this setting. In addition, when we keep τ constant, and increase R, we see that the problem becomes easier for the MILP for $\tau = 0.1$ and 0.3. This is because when $\tau = 0.1$ or 0.3, orders are unlikely to be tardy in the optimal schedule, so the rejection decision is not easy. However, as R grows, the due dates of the orders become looser, and this brings a scheduling flexibility to the system which makes the OAS problem easier. UB_{MILP} in Table 5.7 is defined as the upper bound that MILP model has reached at its termination point. Below in Table 5.8, we see that MILP model with CPLEX solver is not capable of solving the OAS problem for sizes $n > 15$ except for a very limited number of cases in the prespecified time limit.

5.2.4 Experiments for the ISFAN Algorithm

Experimental Design for the ISFAN Algorithm

In order to make the best use of the ISFAN algorithm, we fixed the best algorithm parameter values experimentally.

One of the most important parameters of Simulated Annealing (SA)-based algorithms is the initial temperature. Initial temperature should be chosen carefully so that the system temperature does not approach to 0 too quickly to prevent early convergence, or too slowly to prevent spending CPU time unnecessarily. For the initial temperature, we examined the effect of five different temperatures, 600, 800, 1000, 1200 and 1500.

Identifying the appropriate cooling function is another important aspect of SA-based algorithms. A badly-chosen cooling function can cool the system very quickly and hence the results become poor due to early convergence or very slowly and therefore, the algorithm spends more CPU time even if it successfully converges to a good objective value at an early stage. The cooling functions that we tested are:

1. $T_{i+1} = T_i \times \alpha$, where $\alpha \in (0, 1)$ and i is the iteration number,
2. $T_i = T_{initial} \times (\frac{T_N}{T_{initial}})^{\frac{i}{N}}$, where N is the maximum number of iterations and T_N is the final temperature,
3. $T_i = T_{initial} - i^A$, $A = \frac{\ln(T_{initial} - T_N)}{\ln(N)}$,
4. $T_i = T_{initial} \times \exp(-A_i^2)$, $A = (\frac{1}{N^2}) \ln(\frac{T_{initial}}{T_N})$.

We examined the effect of these four functions that differ in shape and parameters. Since the first cooling function is a frequently used one, the shape for this function is not given. The graphs for the cooling functions 2, 3 and 4 are presented in Figure 5.1. We present the average deviations of the ISFAN objective function values from the UB_{LP} for 50-order instances that have τ=0.3, and initial temperature=800 as ISFAN parameters with different cooling functions tested in Table 5.5. Since similar behaviors are observed with other combinations of the parameter values, the tables for these combinations are omitted. Resultantly, we chose the cooling function 3 (CF3) as it had the least average deviation from the upper bound. These experiments also show that the algorithm is robust to cooling function selection since the gaps were different by around 1-2 % in each cooling function.

We present the results for initial temperatures 800, 1000 and 1500, combined with the cooling function 3 in Table 5.6. It is observed that the least % deviations are achieved when $T_{initial} = 800$. Since this behavior is observed with other combination of the parameter values and thus the example is representative, we excluded the graphs for other parameter combinations. As a result, we chose 800 as ISFAN's initial temperature.

Additionally, we tested different initial sequences for the ISFAN algorithm. It is observed that the quality of the initial sequence does not affect the final solution and the required CPU time of the ISFAN algorithm.

Finally we examined the convergence behavior of the ISFAN algorithm in each rejection iteration to determine the maximum number of neighborhood operation iterations. We present the results of neighborhood operations for a rejection iteration of a 50-order problem having τ=0.3, initial temperature=800 and cooling function 3 in Figure 5.2. As shown in the figure, the results do not achieve stability until at least 3.5 millions of neighborhood operations have been reached. After that point, the results of the experiment show that, dramatic improvements may still be achieved when the maximum number is increased to 5 millions. Although the required CPU time is not very significantly affected when this number is increased for example for a couple of millions, it is rare that the objective function is improved after 5 millions of iterations. Similar behaviors are observed with all other combinations of parameter values. Hence the maximum number of neighborhood operations in a rejection iteration is fixed to 5 millions in the ISFAN algorithm.

Computational Results of ISFAN Algorithm with Random Data

The first part of this computational study compares the performance of ISFAN with respect to the MILP model's objective values for $n = 10$ and 15. Local improvement methods, insertion and exchange, are applied to ISFAN solutions sequentially, and the maximum, minimum, and average % deviation of the ISFAN objective function value from the $\min(UB_{MILP}, UB_{LPVI})$ bound in Table 5.7 and the corresponding CPU times for the ISFAN algorithm are presented in Table 5.8.

The results of the experiments in Table 5.8 confirm that, as τ increases, the problem gets less time consuming for the MILP, but more time consuming for the ISFAN algorithm. This is because the time-consuming part of the ISFAN algorithm is the rejection part. Namely, as more orders are rejected, ISFAN algorithm needs more CPU time. The reverse is true for MILP: As more orders are rejected, the search space is narrowed, and thus the optimal solution is found more quickly.

We found that the ISFAN objective function values are competitive with the MILP objective function values for $n = 10$ and 15 which can be observed from the average percentage deviations displayed in Table 5.7. ISFAN algorithm is clearly a fast and an efficient metaheuristic for $n = 10$ and 15 since it generated gaps of 6% on the average where MILP generates gaps of 8% on the average.

As it is not possible to obtain reasonable solutions by solving the MILP for the OAS problem when $n > 15$, we compared the performance of the ISFAN with respect to an upper bound in the second part of the computational study for $n \in \{20, 25, 50\}$. Since ISFAN becomes time consuming when $n = 100$, we did not run ISFAN for the data where $n \geq 50$. An easy to compute upper bound is the bound that is obtained by relaxing the integrality restrictions, UB_{LP}. However, as mentioned in Section 3.5, UB_{LP} is dominated by LP relaxation bound with valid inequalities, UB_{LPVI}. Hence, we used the UB_{LPVI} to evaluate the performance and to calculate the gaps of the ISFAN algorithm. For each parameter combination of the problem, we solved 10 instances generated randomly as described in Section 5.1.1 and reported the results. We present the maximum, the minimum, and the average % deviation of the ISFAN objective function values from UB_{LPVI} for $n > 15$, and the corresponding CPU times in Tables 5.9 and 5.10. UB_{LPVI} bound is observed to be relatively tight when tardiness factor τ is equal to 0.1 and 0.3 for n=10 (average ISFAN-LP gaps are around 1% and 9%, respectively).

UB_{LPVI} bound calculation solution times are not included in the table since the problem is solved within maximum of 3–4 seconds when $n = 50$.

We can make the following further observations from the results presented in Tables 5.7, 5.8, 5.9 and 5.10:

- The CPU times of ISFAN are reasonable even for large-size problem instances.

- The average % deviations of ISFAN solutions from the upper bounds are below 21%

in all cases. The average optimality gap is only 9% when all cases are considered. Hence, the performance of the algorithm with respect to the objective function value side is good.

- The observation we made for small-size problem instances regarding the difficult problem cases is not clear as in Tables 5.7 and 5.8 because when we analyze the CPU times of ISFAN, we cannot see much difference depending on different cases. However, we observe that the average % deviation of the difficult cases for small-size problems is also large in Table 5.7. When we examine % deviation results for different n values in Tables 5.7, 5.9 and 5.10, we see that the differences among the average % deviations for different cases are not so large. Hence, we may conclude that for large-size problems, the difficulty of the problem remains the same for different R values.

- We observe that as τ increases, the problem gets more difficult for the ISFAN algorithm. This can be explained as follows: As, τ, the tardiness factor, increases, the time window for each order gets narrower, and hence the probability of being tardy increases for each order. Therefore, more of the orders are rejected when τ is large compared to when it is small. Since the most time-consuming part of the ISFAN is the rejection part, ISFAN requires more time for the instances having large τ.

- When we increase the problem size from 25 to 50, the optimality gap increases slightly, from 10% to 12%. Therefore the ISFAN algorithm is not dramatically affected from the size of the problems.

Overall we can say that ISFAN is an efficient solution method for the OAS problem even for large-size problems as its average % deviation from the UB_{LPVI} is below 21% in all cases, and 9% on the average.

Computational Results of MILP and ISFAN algorithm for S&M Data

As mentioned earlier in Chapter 2, Slotnick and Morton [28] study a special case of the OAS problem. 100 S&M instances for each of $n = 10$ and 20 problems were provided by the authors. We arranged the ISFAN algorithm and the original MILP model to handle this special case in order to evaluate if MILP and ISFAN work for this special data. The

ISFAN and MILP objective functions' values are compared to the UB_{MILP} at termination. Original MILP model was run for 10 minutes for each 10-order and 20-order problems. Since the MILP model is solvable for all 10-order problems optimally by CPLEX in 10 minutes, ISFAN algorithm was run for 20-order problems only. Average CPU time that CPLEX needed to solve the MILP for 10-job problems is 186.05 seconds whereas the average CPU time is 6154.12 seconds in [28]. Although this huge difference can be attributed to the evolution of computers, the result shows that MILP model solves the instances generated by SM data efficiently for $n = 10$.

Slotnick and Morton have also applied beam search using Vogel bound, Vogel heuristic and assignment heuristic to the S&M problem. Beam search algorithm with assignment bound and assignment heuristic dominated other methods, hence we used beam search algorithm with assignment bound and assignment heuristic to measure our methods' performances. We observe the followings for the case when $n = 20$: 12% of the 20-order problems are solved optimally in 10 minutes time limit with MILP. % deviation from the UB_{MILP} is less than 15% for 46% of the problems. Average % deviation of 100 instances of $n = 20$ size from UB_{MILP} is 20% for MILP. ISFAN algorithm was also run for these problems. ISFAN solved 31% of the problems optimally. Average CPU time needed for the ISFAN algorithm for $n = 20$ is 31 seconds. The average CPU time for Slotnick and Morton's beam search algorithm with assignment bound is 8474 seconds and the corresponding average optimality gap is 2.4% when compared to UB_{MILP}. Average gap between UB_{MILP} and the ISFAN objective function value is 4.7%, which shows that the ISFAN algorithm solves this special case efficiently with respect to time and objective function value considerations.

5.2.5 Experiments for d-RFSB and m-ATCS Heuristics

Computational Results of d-RFSB and m-ATCS Heuristics for Random Data

The constructive heuristics d-RFSB and m-ATCS are tested on the same problems generated for the tests of MILP model. Local improvement methods, insertion and exchange, are applied to both of the methods after the algorithms were run. In Tables 5.11, 5.12, 5.13, and 5.15, we present % deviation of the algorithms from the UB_{LPVI} for $n \in \{10, 15, 20, 25, 50, 100\}$. Since the required runtime is very small (varying in 0 to 5 seconds interval) for both d-RFSB and m-ATCS heuristics, the CPU times are not reported.

In Tables 5.11 and 5.12, we see that the ISFAN algorithm performs better than both d-RFSB and m-ATCS algorithms when n= 10, 15, 20 and 25. However, time requirement for ISFAN increases as the problem size increases. We can make the following further observations from Tables 5.11, 5.12 and 5.13:

- The average % deviation from the upper bound is below 23% in all cases for d-RFSB heuristic and below 26% in all cases for m-ATCS heuristic where $n \leq 50$. Therefore, both of the constructive algorithms generate good solutions in a very short time.

- d-RFSB heuristic and m-ATCS heuristic are prone to have significantly larger deviations than ISFAN algorithm for small n. As n gets larger, % deviations from the upper bound become closer for all of the three methods.

- Although both of the algorithms perform similarly in the tests for $n \leq 25$, when n=50, m-ATCS heuristic dominates the ISFAN and d-RFSB heuristic. This can be attributed to two main reasons:

  - Because the m-ATCS heuristic index includes information on release dates, processing times and setup times to the order selection process, it potentially generates more efficient solutions. However, this property is compensated by ISFAN's neighborhood moves for $n < 50$.
  - As the problem size increases, ISFAN needs more neighborhood operations to reach better solutions, however the number of neighborhood moves is fixed for all sizes.

  For the problem sizes greater than 50, m-ATCS heuristic is promising since it generates better objective value % deviations from UB_{LPVI} than d-RFSB heuristic for $n = 100$.

- When $\tau = 0.1$, gaps are smaller in all of the methods since fewer orders are rejected and the bound is considerably tighter in $\tau = 0.1$ case. This evidence shows that all of the proposed algorithms are good at sequencing the orders.

- % deviations of the d-RFSB and m-ATCS heuristics from the UB_{LPVI} values are independent of the problem size. While deviations are 8% for both of the algorithms for $n = 10$, 12 and 9% deviations are calculated respectively when $n = 50$.

As a representative experiment, we present and compare the number of rejected jobs for each three algorithms for $n = 50$ in Table 5.14. It is observed that the heuristic that rejects the least number orders is m-ATCS algorithm (9 rejected orders) on the average. Although this result agrees with our previous results stating that m-ATCS heuristic dominates other heuristics for large n, the number of rejected orders does not necessarily imply that a heuristic is the clear winner of a comparison since our revenues, e_i for each i, takes different values in [1,20] interval.

Since the required time for ISFAN algorithm increases as n increases, we only ran d-RFSB and m-ATCS heuristics for the case when $n = 100$. The results for this experiment are presented in Table 5.15. It is observed that both of the heuristics perform well for large n. The average % deviation of d-RFSB heuristic is 11% while it is 7% for m-ATCS heuristic for $n = 100$. Therefore for $n = 100$, m-ATCS heuristic dominates d-RFSB heuristic on the average % deviations from UB_{LPVI}. This domination can be attributed to m-ATCS heuristic's index. It includes more information on the OAS problem to the heuristic's procedures. An improved version of the d-RFSB heuristic may generate better solutions for the OAS problem.

Computational Results of d-RFSB and m-ATCS Heuristics for S&M Data

We arranged d-RFSB and m-ATCS heuristics to solve the S&M data and evaluate the performances of the heuristics. In the computational experiments, it is observed that m-ATCS heuristic solved the 56% of the 20-job problems optimally while the corresponding value for the Slotnick and Morton's best performing heuristic (assignment heuristic) is 44%. However the average % deviation from the UB_{MILP} has been found as 3.9% for the m-ATCS heuristic and 10.87% for the d-RFSB heuristic while the average % deviation is 2.4% for the assignment heuristic. The required times for the assignment heuristic, the m-ATCS and d-RFSB heuristics are competitive. Therefore, m-ATCS heuristic can efficiently be applied to S&M problem to obtain optimal or near-optimal solutions.

5.2.6 Experiments for MILPH

MILPH method has MILP model solved by CPLEX solver with a sequence that contains all of the jobs as input as mentioned in Section 4.3. In addition, local improvement methods, insertion and exchange, are sequentially applied to the solutions of MILPH. Due to solving the MILP model in it, MILPH is promising for small size problems only. We limit our MILPH runs for order sizes of 15 and 20 since for $n > 20$, MILPH becomes very time consuming for the OAS problem instances.

MILPH generates an average of 12% and 13% deviations from UB_{LPVI} for 15-order and 20-order problems respectively. We give a summary of numerical experiments for MILPH method in Table 5.16.

As observed from Tables 5.11, 5.12 and 5.16, the MILPH is dominated by ISFAN, d-ATCS and m-ATCS heuristics with respect to CPU time and objective function value considerations.

5.2.7 Conclusions on Computational Experiments

In this chapter, we performed computational experiments to compare the results obtained by solving the model using CPLEX 10.0 and the results of the proposed heuristics.

We explain the random data generation procedure used in creating our test instances in Section 5.1. Next, we solve these instances using MILP model by CPLEX 10.0 with one hour CPU time limit using the generated data, and we observed that MILP is not solvable for n >15. We also relaxed the integrality restrictions on y_{ij} variables in MILP and solved the relaxed model by CPLEX. We used the resulting LP bound to evaluate the results obtained from solving the MILP. The corresponding % deviations of MILP solutions from the min(UB_{LP}, UB_{MILP}) bound can be found in Table 5.7's righthand side and CPU times obtained by solving MILP can be found in Table 5.8's righthand side. It's observed from Table 5.8 that when τ and R are small and close to each other, the problem is harder to solve with the MILP model. The reason might be that when τ is small, the orders are not spread to different parts of the time line. Hence, both the rejection and the sequencing decisions are difficult since many orders are available at a time. To strengthen the upper bound, we added valid inequalities to the relaxed MILP model, and showed that UB_{LPVI} improves the UB_{LP} 0 to 12% on the average. When we compare the improvements of UB_{LPVI} and UB_{MILP} bounds in Table 5.4, it's found that UB_{LPVI} has higher average % improvements on the LP relaxation values on the average (UB_{LPVI} improvement is 2% and UB_{MILP} improvement is 1% on the average).It is observed that the largest improvements are achieved when $\tau = 0.5$ and $R = 0.1$. Hence UB_{LPVI} has the most improvement potential for τ large and R small. As discussed earlier, when τ is large, more orders are expected to be tardy. In addition, when R is small, due dates of the orders are close to each other, and there is more possibility of clashes. Therefore, more orders are expected to be rejected when τ is large and R is small. Valid inequalities efficiently identify such cases and improve the bounds more.

Next, we proposed our simulated annealing-based algorithm ISFAN. In order to find the best algorithm parameters, we performed an experimental design. As a result of the experiments, we find that the best combination of parameters is reached when initial temperature=800, cooling function=3 and number of iterations in simulated annealing phase is five millions. In addition we performed local search (insertion and exchange of the orders) to improve the results obtained by ISFAN.

ISFAN is found to be a good performing hybrid metaheuristic in the computational experiments. Even when $n = 50$, it produces efficient solutions (average % deviation from UB_{LPVI} bound is 12%) in less than 10 minutes of CPU time on the average. We observed that the problem gets harder for ISFAN when tardiness factor, τ, is large. This is because when τ is large, more orders are rejected in optimal sequence. Since the rejection operation is identified as the most time-consuming part of the ISFAN, more rejection causes greater CPU times.

We also ran the ISFAN algorithm for the S&M data. It solved these problems efficiently, especially in terms of CPU time. ISFAN required 31 seconds for solving an S&M problem of size 20 on the average. The average % deviation of ISFAN solutions from UB_{LPVI} bound is 4.7% and ISFAN solved 31% of the solutions optimally.

We further proposed two constructive heuristics, d-RFSB heuristic and m-ATCS heuristic, in Chapter 4 for the problem. Both of the heuristics are tested on the random data and the results are also improved by local search as in ISFAN. The constructive heuristics performed very well in terms of CPU time. For $n \leq 50$, ISFAN dominated d-RFSB and m-ATCS algorithms, however for $n = 100$, m-ATCS heuristic is proved to be the best performing method. Hence, m-ATCS heuristic has more potential to perform well for larger-size problems.

d-RFSB and m-ATCS heuristics are also tested on S&M data. While m-ATCS heuristic solves 56% of the S&M data of size 20, the best method proposed by Slotnick and Morton [28] solves 44% of the problems optimally. The average % deviation from UB_{LPVI} values is 3.9% for the m-ATCS heuristics while the best method in [28] results in average % deviation of 2.4%. Hence, the m-ATCS heuristic is an efficient method to solve the S&M problem when CPU time and average % deviations are considered simultaneously.

Finally we tested MILP heuristic with the generated random data. MILPH is dominated by ISFAN, d-RFSB and m-ATCS heuristics.

In Chapter 6, we give possible directions for future study, and our preliminary studies for these directions.

Table 5.1: The average % improvement of UB_{LPVI} over UB_{LP} for sizes of 10 and 15.

			UB_{LP} vs. UB_{LPVI}		
n	τ	R	Max	Min	Average
10	0.1	0.1	3%	0%	1%
		0.3	0%	0%	0%
		0.5	0%	0%	0%
		0.7	1%	0%	0%
		0.9	0%	0%	0%
	0.3	0.1	13%	1%	5%
		0.3	2%	0%	1%
		0.5	3%	0%	0%
		0.7	0%	0%	0%
		0.9	0%	0%	0%
	0.5	0.1	27%	5%	12%
		0.3	16%	1%	7%
		0.5	10%	0%	3%
		0.7	5%	0%	1%
		0.9	1%	0%	0%
	0.7	0.5	13%	1%	7%
		0.7	7%	0%	2%
		0.9	6%	0%	1%
	0.9	0.9	21%	0%	6%
		Avg	7%	0%	2%
15	0.1	0.1	3%	0%	1%
		0.3	0%	0%	0%
		0.5	0%	0%	0%
		0.7	0%	0%	0%
		0.9	0%	0%	0%
	0.3	0.1	5%	1%	2%
		0.3	2%	0%	1%
		0.5	3%	0%	0%
		0.7	0%	0%	0%
		0.9	0%	0%	0%
	0.5	0.1	20%	5%	11%
		0.3	9%	1%	4%
		0.5	4%	0%	1%
		0.7	1%	0%	0%
		0.9	0%	0%	0%
	0.7	0.5	11%	1%	4%
		0.7	10%	0%	2%
		0.9	1%	0%	0%
	0.9	0.9	5%	0%	1%
		Avg	4%	0%	2%

Table 5.2: The average % improvement of UB_{LPVI} over UB_{LP} for sizes of 20 and 25.

			UB_{LP} vs. UB_{LPVI}		
n	τ	R	Max	Min	Average
20	0.1	0.1	2%	0%	1%
		0.3	0%	0%	0%
		0.5	0%	0%	0%
		0.7	0%	0%	0%
		0.9	0%	0%	0%
	0.3	0.1	5%	2%	3%
		0.3	2%	0%	1%
		0.5	1%	0%	0%
		0.7	0%	0%	0%
		0.9	0%	0%	0%
	0.5	0.1	16%	8%	10%
		0.3	12%	2%	5%
		0.5	5%	0%	2%
		0.7	1%	0%	0%
		0.9	0%	0%	0%
	0.7	0.5	8%	1%	5%
		0.7	11%	0%	3%
		0.9	1%	0%	0%
	0.9	0.9	4%	0%	2%
		Avg	3%	1%	2%
25	0.1	0.1	1%	0%	0%
		0.3	0%	0%	0%
		0.5	0%	0%	0%
		0.7	0%	0%	0%
		0.9	0%	0%	0%
	0.3	0.1	3%	0%	2%
		0.3	1%	0%	0%
		0.5	0%	0%	0%
		0.7	0%	0%	0%
		0.9	0%	0%	0%
	0.5	0.1	12%	5%	8%
		0.3	5%	1%	3%
		0.5	2%	0%	1%
		0.7	0%	0%	0%
		0.9	0%	0%	0%
	0.7	0.5	6%	2%	3%
		0.7	2%	0%	1%
		0.9	2%	0%	0%
	0.9	0.9	5%	0%	2%
		Avg	2%	0%	1%

Table 5.3: The average % improvement of UB_{LPVI} over UB_{LP} for sizes of 50 and 100.

			UB_{LP} vs. UB_{LPVI}		
n	τ	R	Max	Min	Average
50	0.1	0.1	1%	0%	0%
		0.3	0%	0%	0%
		0.5	0%	0%	0%
		0.7	0%	0%	0%
		0.9	0%	0%	0%
	0.3	0.1	3%	1%	2%
		0.3	0%	0%	0%
		0.5	0%	0%	0%
		0.7	0%	0%	0%
		0.9	0%	0%	0%
	0.5	0.1	8%	5%	7%
		0.3	7%	2%	3%
		0.5	2%	0%	1%
		0.7	0%	0%	0%
		0.9	0%	0%	0%
	0.7	0.5	5%	2%	4%
		0.7	1%	0%	0%
		0.9	0%	0%	0%
	0.9	0.9	1%	0%	1%
		Avg	2%	0%	1%
100	0.1	0.1	1%	0%	0%
		0.3	0%	0%	0%
		0.5	0%	0%	0%
		0.7	0%	0%	0%
		0.9	0%	0%	0%
	0.3	0.1	3%	1%	2%
		0.3	1%	0%	0%
		0.5	0%	0%	0%
		0.7	0%	0%	0%
		0.9	0%	0%	0%
	0.5	0.1	9%	5%	7%
		0.3	3%	1%	2%
		0.5	1%	0%	0%
		0.7	0%	0%	0%
		0.9	0%	0%	0%
	0.7	0.5	5%	2%	4%
		0.7	1%	0%	0%
		0.9	0%	0%	0%
	0.9	0.9	1%	0%	1%
		Avg	1%	1%	1%

Table 5.4: The average % improvements of UB_{MILP} and UB_{LPVI} over UB_{LP} for sizes of 10 and 15.

			% Improvements					
			UB_{LP} vs. UB_{MILP}			UB_{LP} vs. UB_{LPVI}		
n	τ	R	Max	Min	Average	Max	Min	Average
10	0.1	0.1	0%	0%	0%	3%	0%	1%
		0.3	1%	0%	0%	0%	0%	0%
		0.5	0%	0%	0%	0%	0%	0%
		0.7	0%	0%	0%	1%	0%	0%
		0.9	4%	0%	0%	0%	0%	0%
	0.3	0.1	0%	0%	0%	13%	1%	5%
		0.3	19%	0%	5%	2%	0%	1%
		0.5	9%	0%	3%	3%	0%	0%
		0.7	7%	0%	2%	0%	0%	0%
		0.9	13%	0%	3%	0%	0%	0%
	0.5	0.1	33%	11%	22%	27%	5%	12%
		0.3	35%	6%	20%	16%	1%	7%
		0.5	27%	3%	12%	10%	0%	3%
		0.7	25%	2%	11%	5%	0%	1%
		0.9	21%	0%	7%	1%	0%	0%
	0.7	0.5	34%	13%	25%	13%	1%	7%
		0.7	24%	4%	16%	7%	0%	2%
		0.9	16%	2%	7%	6%	0%	1%
	0.9	0.9	38%	6%	21%	21%	0%	6%
		Average	16%	2%	8%	7%	0%	2%
15	0.1	0.1	0%	0%	0%	3%	0%	1%
		0.3	0%	0%	0%	0%	0%	0%
		0.5	0%	0%	0%	0%	0%	0%
		0.7	0%	0%	0%	0%	0%	0%
		0.9	0%	0%	0%	0%	0%	0%
	0.3	0.1	0%	0%	0%	5%	1%	2%
		0.3	0%	0%	0%	2%	0%	1%
		0.5	0%	0%	0%	3%	0%	0%
		0.7	0%	0%	0%	0%	0%	0%
		0.9	0%	0%	0%	0%	0%	0%
	0.5	0.1	0%	0%	0%	20%	5%	11%
		0.3	0%	0%	0%	9%	1%	4%
		0.5	0%	0%	0%	4%	0%	1%
		0.7	0%	0%	0%	1%	0%	0%
		0.9	0%	0%	0%	0%	0%	0%
	0.7	0.5	24%	0%	11%	11%	1%	4%
		0.7	4%	0%	1%	10%	0%	2%
		0.9	0%	0%	0%	1%	0%	0%
	0.9	0.9	30%	0%	16%	5%	0%	1%
		Average	3%	0%	1%	4%	0%	2%

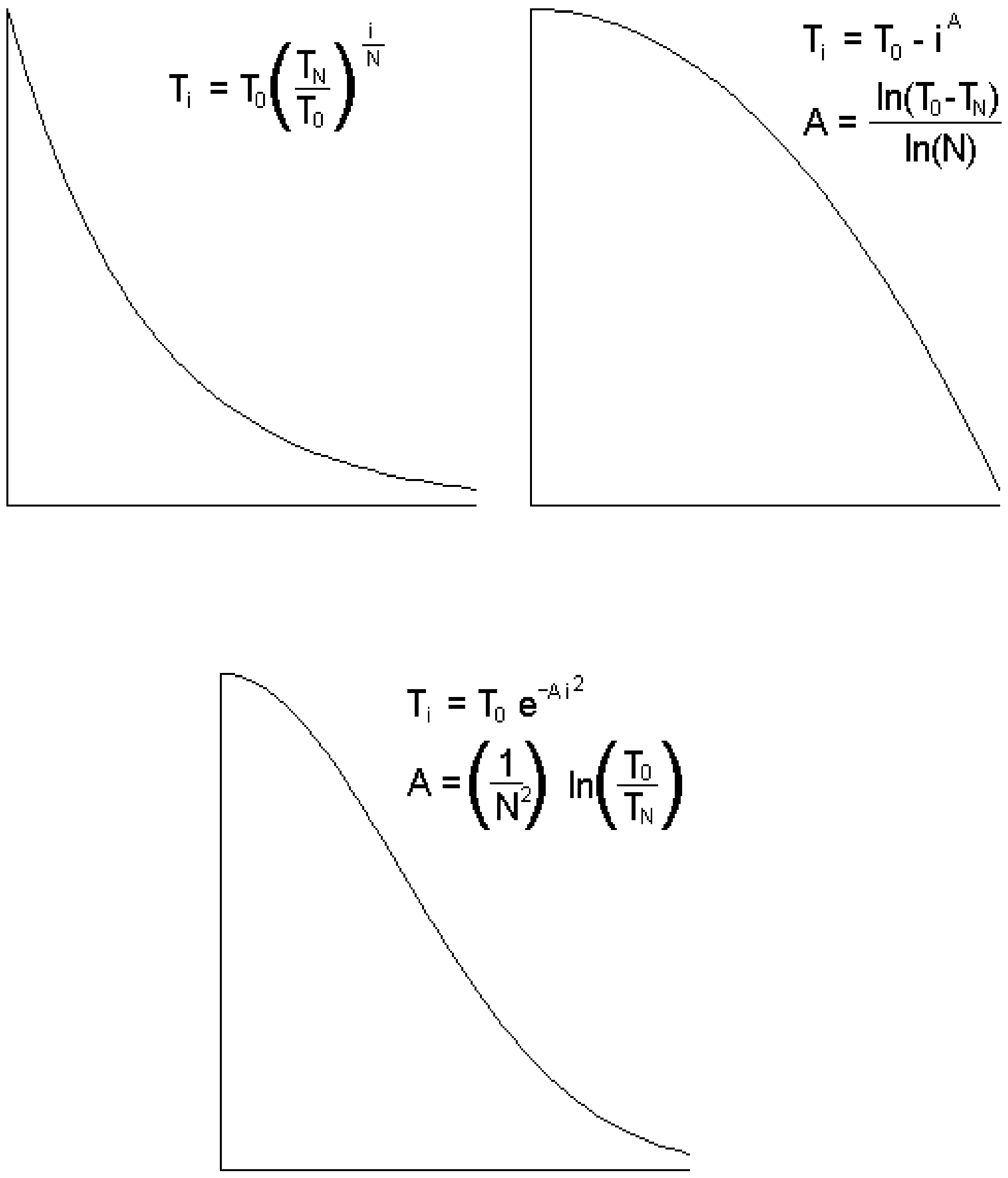

Figure 5.1: Cooling functions 2, 3 and 4.

Table 5.5: Evaluation of different cooling functions for n=50, $\tau = 0.3$, $T_{initial} = 800$

	CF1			CF2			CF3			CF4		
R	Max	Min	Average	Max	Min	Average	Max	Min	Average	Max	Min	Average
0.1	28%	19%	24%	29%	18%	24%	29%	20%	24%	30%	20%	24%
0.3	26%	16%	21%	26%	17%	21%	26%	17%	20%	26%	15%	20%
0.5	26%	17%	22%	25%	18%	22%	24%	15%	21%	26%	17%	22%
0.7	25%	16%	22%	27%	18%	22%	28%	15%	22%	30%	16%	23%
0.9	34%	14%	27%	31%	16%	26%	30%	16%	24%	34%	17%	25%
Average	28%	16%	23%	28%	17%	23%	27%	16%	22%	29%	17%	23%

Table 5.6: % Gap of the ISFAN algorithm results from UB_{LP} for initial temperatures of 800, 1000 and 1500 for n=50, $\tau = 0.3$, $CF = 3$ (Compared with UB_{LP})

	$T_{initial}$=800			$T_{initial}$=1000			$T_{initial}$=1500		
R	Max	Min	Average	Max	Min	Average	Max	Min	Average
0.1	29%	20%	24%	29%	19%	25%	29%	20%	24%
0.3	26%	17%	20%	25%	15%	21%	25%	16%	21%
0.5	24%	15%	22%	28%	16%	22%	25%	17%	22%
0.7	28%	15%	22%	27%	16%	22%	28%	17%	22%
0.9	30%	16%	24%	34%	15%	26%	34%	18%	26%
Avg	27%	17%	22%	29%	16%	23%	28%	18%	23%

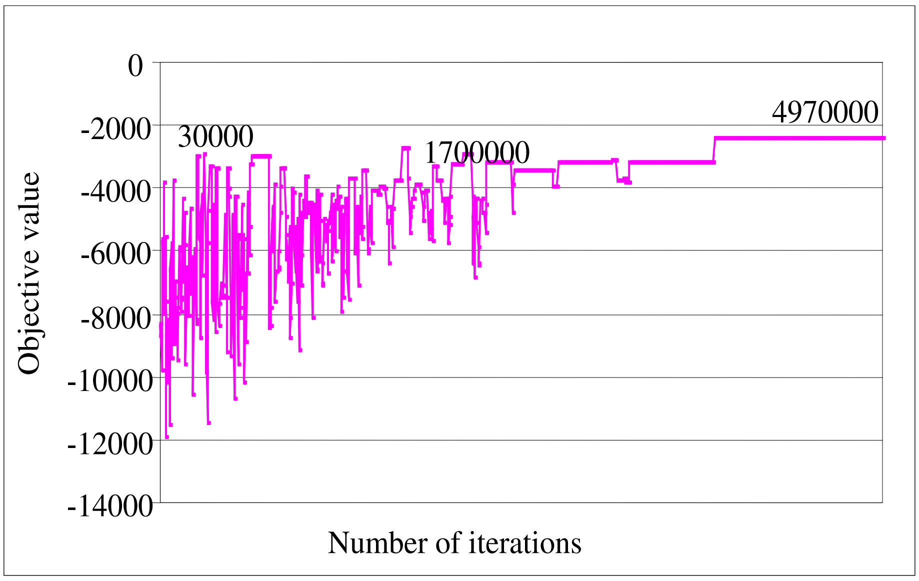

Figure 5.2: Convergence behavior of the ISFAN algorithm at a rejection iteration for a 50-order problem having τ=0.3, R=0.3, initial temperature=800 and cooling function 3.

Table 5.7: ISFAN and MILP % deviation comparison from min(UB_{LPVI},UB_{MILP}) for the OAS problem with sizes 10 and 15.

			% Deviation of ISFAN			% Deviation of MILP		
n	τ	R	Max	Min	Average	Max	Min	Average
10	0.1	0.1	14%	0%	6%	10%	0%	2%
		0.3	13%	0%	3%	6%	0%	1%
		0.5	6%	0%	1%	3%	0%	1%
		0.7	4%	0%	0%	4%	0%	0%
		0.9	9%	0%	1%	3%	0%	0%
	0.3	0.1	24%	2%	10%	9%	2%	5%
		0.3	10%	0%	4%	3%	0%	0%
		0.5	5%	0%	2%	0%	0%	0%
		0.7	6%	0%	2%	0%	0%	0%
		0.9	7%	0%	1%	0%	0%	0%
	0.5	0.1	20%	0%	8%	0%	0%	0%
		0.3	9%	0%	3%	0%	0%	0%
		0.5	9%	0%	4%	0%	0%	0%
		0.7	19%	0%	6%	0%	0%	0%
		0.9	18%	0%	3%	0%	0%	0%
	0.7	0.5	12%	0%	4%	0%	0%	0%
		0.7	9%	0%	4%	0%	0%	0%
		0.9	14%	0%	6%	0%	0%	0%
	0.9	0.9	16%	0%	4%	0%	0%	0%
		Avg	12%	0%	4%	2%	0%	1%
15	0.1	0.1	8%	1%	4%	22%	3%	13%
		0.3	12%	0%	3%	22%	2%	11%
		0.5	5%	0%	1%	17%	0%	7%
		0.7	6%	0%	1%	33%	0%	5%
		0.9	4%	0%	1%	20%	0%	5%
	0.3	0.1	11%	1%	6%	41%	4%	23%
		0.3	8%	2%	6%	42%	13%	26%
		0.5	8%	1%	4%	38%	8%	20%
		0.7	6%	1%	3%	38%	5%	22%
		0.9	16%	0%	6%	36%	0%	17%
	0.5	0.1	28%	5%	14%	49%	18%	33%
		0.3	20%	11%	15%	34%	6%	24%
		0.5	22%	8%	15%	45%	12%	26%
		0.7	22%	3%	10%	33%	4%	19%
		0.9	20%	0%	6%	44%	0%	15%
	0.7	0.5	27%	2%	11%	31%	0%	7%
		0.7	25%	6%	14%	30%	5%	16%
		0.9	16%	1%	9%	19%	2%	11%
	0.9	0.9	34%	0%	8%	26%	0%	4%
		Avg	16%	2%	7%	33%	4%	16%

Table 5.8: CPU time comparison of MILP and ISFAN for the OAS problem with sizes 10 and 15.

			CPU Time (ISFAN)			CPU Time (MILP)		
n	τ	R	Max	Min	Average	Max	Min	Average
10	0.1	0.1	32.75	13.91	21.95	3600.00	7.53	2828.35
		0.3	23.19	13.88	15.11	3600.00	0.10	2312.69
		0.5	32.25	13.89	15.82	3600.00	0.08	724.40
		0.7	32.16	13.94	15.81	3600.00	0.08	660.57
		0.9	23.16	14.00	14.97	3600.00	0.06	409.16
	0.3	0.1	42.61	23.94	36.00	3600.00	3600.00	3600.00
		0.3	50.38	14.81	27.35	3600.00	946.13	1960.58
		0.5	32.28	14.03	21.52	3016.41	399.97	1294.52
		0.7	32.70	13.98	18.05	2361.50	23.41	708.04
		0.9	32.95	14.02	17.03	785.01	0.66	228.34
	0.5	0.1	63.28	43.13	54.81	237.39	20.20	120.92
		0.3	59.66	33.27	45.93	339.52	41.59	105.54
		0.5	52.39	14.11	34.69	829.91	8.61	198.93
		0.7	41.88	23.25	30.82	450.95	3.66	148.47
		0.9	33.28	14.02	22.57	303.50	9.47	97.38
	0.7	0.5	68.70	32.73	52.32	15.30	0.08	3.48
		0.7	50.78	23.80	38.76	162.56	5.42	51.50
		0.9	32.48	14.00	21.88	180.11	14.33	52.16
	0.9	0.9	70.47	32.14	45.66	13.13	0.22	4.30
		Avg	42.49	20.04	29.00	1783.96	267.45	816.28
15	0.1	0.1	48.19	26.67	35.50	3600.00	3600.00	3600.00
		0.3	36.89	15.84	26.46	3600.00	3600.00	3600.00
		0.5	31.64	15.86	19.91	3600.00	0.89	3240.09
		0.7	40.14	15.94	19.62	3600.00	15.29	2515.50
		0.9	33.69	16.02	18.93	3600.00	0.31	2791.46
	0.3	0.1	59.75	39.06	49.56	3600.00	3600.00	3600.00
		0.3	59.30	28.58	45.74	3600.00	3600.00	3600.00
		0.5	47.69	27.42	34.98	3600.00	3600.00	3600.00
		0.7	47.06	16.75	34.45	3600.00	3600.00	3600.00
		0.9	55.66	16.24	36.39	3600.00	6.42	3240.64
	0.5	0.1	103.69	61.84	86.85	3600.00	3600.00	3600.00
		0.3	96.61	49.06	72.23	3600.00	3600.00	3600.00
		0.5	76.44	49.80	64.61	3600.00	3600.00	3600.00
		0.7	69.44	30.63	50.41	3600.00	3600.00	3600.00
		0.9	78.27	22.27	41.08	3600.00	2998.09	3521.96
	0.7	0.5	99.17	66.17	78.20	3600.00	803.82	2435.55
		0.7	91.06	51.42	68.03	3600.00	3600.00	3600.00
		0.9	63.50	31.30	51.88	3600.00	3600.00	3600.00
	0.9	0.9	123.97	67.94	84.48	3600.00	76.33	1709.82
		Avg	66.43	34.15	48.38	3600.00	2479.01	3297.63

Table 5.9: ISFAN objective value % deviations from UB_{LPVI} and the correponfing CPU times for the problem with sizes of 20 and 25.

			% Deviation of ISFAN			ISFAN CPU Time		
n	τ	R	Max	Min	Average	Max	Min	Average
20	0.1	0.1	16%	2%	6%	78.75	47.36	62.37
		0.3	8%	2%	5%	58.67	40.23	48.05
		0.5	6%	1%	3%	54.75	31.28	42.34
		0.7	5%	0%	2%	55.55	29.66	37.51
		0.9	4%	0%	1%	43.58	18.67	33.33
	0.3	0.1	13%	2%	8%	95.88	69.19	85.63
		0.3	12%	5%	9%	100.49	68.19	79.72
		0.5	12%	4%	7%	81.38	57.02	68.71
		0.7	16%	2%	8%	93.63	45.14	73.70
		0.9	17%	1%	8%	124.17	42.94	66.59
	0.5	0.1	26%	10%	15%	142.13	110.52	125.72
		0.3	22%	9%	16%	131.33	98.78	116.30
		0.5	23%	8%	16%	156.20	82.61	109.41
		0.7	14%	5%	10%	103.59	55.20	89.73
		0.9	18%	2%	10%	134.63	55.84	90.79
	0.7	0.5	23%	15%	18%	149.00	122.67	133.68
		0.7	20%	14%	17%	120.17	99.08	113.51
		0.9	24%	5%	14%	113.08	89.16	98.93
	0.9	0.9	25%	13%	18%	142.99	101.56	119.04
		Average	16%	5%	10%	104.21	66.58	83.95
25	0.1	0.1	11%	2%	5%	106.47	61.31	75.99
		0.3	8%	1%	4%	87.50	48.41	64.43
		0.5	10%	2%	4%	63.67	35.14	51.87
		0.7	8%	0%	3%	75.50	33.89	51.03
		0.9	3%	0%	2%	68.45	20.48	41.38
	0.3	0.1	13%	3%	8%	121.33	97.88	111.68
		0.3	13%	5%	8%	123.86	89.55	102.28
		0.5	9%	3%	6%	101.56	82.92	91.60
		0.7	13%	4%	8%	125.41	78.63	94.97
		0.9	16%	4%	7%	134.64	64.72	98.39
	0.5	0.1	19%	7%	12%	199.50	149.19	164.54
		0.3	29%	9%	16%	180.27	145.05	157.89
		0.5	21%	11%	16%	180.19	122.91	148.72
		0.7	24%	4%	12%	175.55	107.27	140.73
		0.9	15%	3%	8%	145.80	91.92	123.17
	0.7	0.5	25%	11%	18%	197.23	169.17	185.33
		0.7	27%	10%	17%	175.63	144.81	165.48
		0.9	20%	4%	12%	155.86	128.48	142.04
	0.9	0.9	37%	12%	21%	260.14	151.58	196.01
		Average	17%	5%	10%	140.98	95.96	116.19

Table 5.10: ISFAN objective value % deviations from UB_{LPVI} and the correponfing CPU times for the problem with size of 50.

			% Deviation of ISFAN			ISFAN CPU Time		
n	τ	R	Max	Min	Average	Max	Min	Average
50	0.1	0.1	9%	4%	6%	292.13	233.69	257.77
		0.3	8%	3%	6%	280.19	211.73	245.42
		0.5	7%	2%	4%	259.81	161.17	218.67
		0.7	7%	2%	4%	297.64	171.30	218.87
		0.9	8%	0%	5%	278.83	32.28	207.44
	0.3	0.1	16%	8%	11%	423.00	327.30	377.20
		0.3	14%	8%	11%	399.59	341.41	368.95
		0.5	15%	6%	10%	435.86	332.00	382.55
		0.7	17%	7%	10%	421.63	340.36	378.14
		0.9	13%	7%	10%	464.28	329.16	397.48
	0.5	0.1	19%	14%	16%	603.30	522.75	560.50
		0.3	22%	15%	19%	575.73	503.63	551.24
		0.5	28%	14%	20%	613.77	521.19	549.99
		0.7	20%	12%	16%	631.45	500.91	556.58
		0.9	23%	9%	14%	587.08	482.17	531.93
	0.7	0.5	24%	12%	19%	709.05	656.34	682.72
		0.7	22%	15%	17%	677.66	623.20	650.23
		0.9	19%	9%	13%	669.66	577.66	623.37
	0.9	0.9	21%	11%	15%	737.83	668.48	706.32
		Avg	16%	8%	12%	492.55	396.67	445.55

Table 5.11: ISFAN, d-RFSB and m-ATCS heuristics comparison for the problem with sizes of 10 and 15.

			% Deviation from min(UB_{LPVI}, UB_{MILP})								
			ISFAN Heuristic			d-RFSB Heuristic			m-ATCS Heuristic		
n	τ	R	Max	Min	Average	Max	Min	Average	Max	Min	Average
10	0.1	0.1	14%	0%	6%	13%	4%	7%	29%	2%	9%
		0.3	13%	0%	3%	16%	2%	6%	9%	0%	3%
		0.5	6%	0%	1%	12%	0%	5%	28%	0%	4%
		0.7	4%	0%	0%	11%	0%	5%	9%	0%	2%
		0.9	9%	0%	1%	20%	0%	10%	5%	0%	0%
	0.3	0.1	24%	2%	10%	21%	5%	13%	28%	3%	14%
		0.3	10%	0%	4%	17%	0%	8%	22%	0%	8%
		0.5	5%	0%	2%	28%	6%	14%	28%	0%	8%
		0.7	6%	0%	2%	12%	1%	7%	13%	0%	7%
		0.9	7%	0%	1%	29%	1%	13%	24%	0%	7%
	0.5	0.1	20%	0%	8%	15%	0%	5%	19%	0%	7%
		0.3	9%	0%	3%	29%	0%	11%	38%	0%	15%
		0.5	9%	0%	4%	21%	3%	7%	30%	0%	12%
		0.7	19%	0%	6%	18%	3%	8%	28%	1%	14%
		0.9	18%	0%	3%	27%	0%	9%	20%	0%	10%
	0.7	0.5	12%	0%	4%	15%	0%	4%	21%	0%	5%
		0.7	9%	0%	4%	24%	0%	8%	16%	0%	6%
		0.9	14%	0%	6%	10%	0%	6%	21%	0%	9%
	0.9	0.9	16%	0%	4%	18%	0%	7%	23%	0%	10%
		Average	12%	0%	4%	19%	1%	8%	22%	0%	8%
15	0.1	0.1	8%	1%	4%	13%	2%	8%	9%	2%	5%
		0.3	12%	0%	3%	13%	1%	6%	13%	1%	6%
		0.5	5%	0%	1%	9%	0%	5%	20%	0%	4%
		0.7	6%	0%	1%	13%	1%	6%	39%	0%	5%
		0.9	4%	0%	1%	12%	1%	4%	12%	0%	2%
	0.3	0.1	11%	1%	6%	16%	4%	10%	16%	4%	9%
		0.3	8%	2%	6%	22%	5%	12%	23%	6%	11%
		0.5	8%	1%	4%	21%	6%	13%	23%	1%	7%
		0.7	6%	1%	3%	21%	3%	12%	15%	0%	6%
		0.9	16%	0%	6%	28%	1%	11%	24%	0%	10%
	0.5	0.1	28%	5%	14%	26%	8%	18%	34%	8%	23%
		0.3	20%	11%	15%	28%	11%	22%	36%	8%	23%
		0.5	22%	8%	15%	26%	12%	23%	35%	11%	19%
		0.7	22%	3%	10%	37%	10%	22%	26%	3%	14%
		0.9	20%	0%	6%	23%	6%	14%	32%	0%	10%
	0.7	0.5	27%	2%	11%	34%	0%	14%	29%	0%	15%
		0.7	25%	6%	14%	30%	7%	17%	42%	10%	22%
		0.9	16%	1%	9%	27%	4%	13%	24%	1%	14%
	0.9	0.9	34%	0%	8%	33%	0%	10%	33%	2%	10%
		Average	16%	2%	7%	23%	4%	13%	26%	3%	11%

Table 5.12: ISFAN, d-RFSB and m-ATCS heuristics' average % deviations from UB_{LPVI} for the problem with sizes of 20 and 25.

			% Deviation from UB_{LPVI}								
			ISFAN Heuristic			d-RFSB Heuristic			m-ATCS Heuristic		
n	τ	R	Max	Min	Average	Max	Min	Average	Max	Min	Average
20	0.1	0.1	16%	2%	6%	21%	2%	8%	20%	1%	8%
		0.3	8%	2%	5%	9%	4%	7%	17%	2%	8%
		0.5	6%	1%	3%	17%	2%	8%	11%	0%	5%
		0.7	5%	0%	2%	12%	0%	7%	6%	0%	3%
		0.9	4%	0%	1%	19%	4%	9%	6%	0%	2%
	0.3	0.1	13%	2%	8%	14%	4%	9%	26%	2%	11%
		0.3	12%	5%	9%	17%	7%	11%	17%	6%	10%
		0.5	12%	4%	7%	18%	4%	12%	25%	4%	10%
		0.7	16%	2%	8%	29%	8%	15%	21%	1%	9%
		0.9	17%	1%	8%	25%	2%	16%	20%	1%	9%
	0.5	0.1	26%	10%	15%	25%	13%	19%	42%	16%	23%
		0.3	22%	9%	16%	29%	8%	20%	29%	17%	23%
		0.5	23%	8%	16%	35%	11%	21%	34%	10%	20%
		0.7	14%	5%	10%	21%	7%	15%	25%	4%	11%
		0.9	18%	2%	10%	33%	4%	17%	24%	2%	10%
	0.7	0.5	23%	15%	18%	27%	13%	20%	34%	16%	26%
		0.7	20%	14%	17%	24%	14%	19%	35%	11%	23%
		0.9	24%	5%	14%	26%	9%	18%	24%	9%	14%
	0.9	0.9	25%	13%	18%	30%	11%	21%	30%	10%	21%
		Average	16%	5%	10%	23%	7%	14%	23%	6%	13%
25	0.1	0.1	11%	2%	5%	12%	2%	7%	11%	3%	6%
		0.3	8%	1%	4%	6%	1%	3%	9%	0%	4%
		0.5	10%	2%	4%	9%	1%	5%	20%	1%	5%
		0.7	8%	0%	3%	13%	2%	6%	5%	0%	2%
		0.9	3%	0%	2%	20%	1%	8%	2%	0%	0%
	0.3	0.1	13%	3%	8%	14%	6%	9%	13%	5%	9%
		0.3	13%	5%	8%	16%	7%	11%	15%	4%	9%
		0.5	9%	3%	6%	13%	4%	8%	14%	2%	7%
		0.7	13%	4%	8%	22%	8%	15%	8%	2%	4%
		0.9	16%	4%	7%	22%	7%	13%	16%	0%	7%
	0.5	0.1	19%	7%	12%	20%	8%	13%	27%	10%	20%
		0.3	29%	9%	16%	28%	8%	19%	29%	11%	19%
		0.5	21%	11%	16%	29%	10%	17%	35%	8%	17%
		0.7	24%	4%	12%	23%	11%	17%	21%	7%	13%
		0.9	15%	3%	8%	23%	7%	13%	21%	2%	9%
	0.7	0.5	25%	11%	18%	27%	11%	20%	28%	17%	21%
		0.7	27%	10%	17%	23%	11%	17%	26%	11%	18%
		0.9	20%	4%	12%	23%	6%	15%	21%	2%	11%
	0.9	0.9	37%	12%	21%	32%	14%	22%	32%	13%	24%
		Average	17%	5%	10%	20%	7%	13%	19%	5%	11%

Table 5.13: ISFAN, d-RFSB and m-ATCS heuristics' average % deviations from UB_{LPVI} for the problems with size of 50.

			% Deviation from UB_{LPVI}								
			ISFAN Heuristic			d-RFSB Heuristic			m-ATCS Heuristic		
n	τ	R	Max	Min	Average	Max	Min	Average	Max	Min	Average
50	0.1	0.1	9%	4%	6%	6%	3%	4%	10%	3%	5%
		0.3	8%	3%	6%	7%	2%	4%	9%	2%	4%
		0.5	7%	2%	4%	6%	1%	4%	9%	0%	3%
		0.7	7%	2%	4%	8%	2%	5%	6%	0%	2%
		0.9	8%	0%	5%	9%	2%	6%	2%	0%	1%
	0.3	0.1	16%	8%	11%	12%	5%	8%	22%	5%	11%
		0.3	14%	8%	11%	13%	7%	10%	19%	5%	11%
		0.5	15%	6%	10%	13%	7%	10%	15%	3%	6%
		0.7	17%	7%	10%	18%	9%	12%	17%	0%	4%
		0.9	13%	7%	10%	18%	5%	12%	8%	0%	3%
	0.5	0.1	19%	14%	16%	16%	10%	14%	21%	12%	17%
		0.3	22%	15%	19%	21%	15%	18%	26%	13%	18%
		0.5	28%	14%	20%	25%	13%	20%	31%	7%	16%
		0.7	20%	12%	16%	26%	13%	16%	17%	4%	11%
		0.9	23%	9%	14%	21%	11%	14%	16%	2%	8%
	0.7	0.5	24%	12%	19%	21%	12%	17%	22%	13%	18%
		0.7	22%	15%	17%	27%	12%	16%	17%	11%	14%
		0.9	19%	9%	13%	24%	7%	13%	17%	3%	9%
	0.9	0.9	21%	11%	15%	22%	10%	16%	26%	14%	19%
		Average	16%	8%	12%	16%	8%	12%	16%	5%	9%

Table 5.14: Number of rejected orders for ISFAN, d-RFSB and m-ATCS heuristics for $n = 50$.

			Number of rejected orders out of 50 orders								
			ISFAN Heuristic			d-RFSB Heuristic			m-ATCS Heuristic		
n	τ	R	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg
50	0.1	0.1	11	7	9.1	10	6	7.6	8	5	7
		0.3	10	6	8.2	7	3	6	7	4	5.5
		0.5	7	5	6.1	7	3	5	7	2	4.2
		0.7	9	4	6.4	9	3	5.7	3	0	1.7
		0.9	7	0	5.4	7	2	5.6	4	0	0.7
	0.3	0.1	15	11	13.4	14	9	11.4	14	10	11.1
		0.3	16	11	12.4	13	10	11.2	12	8	9.3
		0.5	13	10	11.2	13	6	9.8	9	5	7.1
		0.7	12	8	9.5	11	6	9.1	10	2	5
		0.9	11	8	9.4	12	6	9	4	0	1.9
	0.5	0.1	21	17	18.2	19	16	17.5	19	15	16.9
		0.3	19	16	17.1	19	13	15.8	19	12	16.3
		0.5	20	13	16.5	19	13	15.3	16	11	13.2
		0.7	17	11	13.6	18	8	12.5	12	6	9
		0.9	15	9	12	13	9	10.8	10	2	6.3
	0.7	0.5	19	15	17.2	20	15	16	18	12	16.4
		0.7	17	12	14.8	18	11	14.2	15	11	11.8
		0.9	17	8	11.8	17	8	11.4	14	5	8.9
	0.9	0.9	18	13	15	17	11	14	17	12	14.4
		Avg	14	10	12	14	8	11	11	6	9

Table 5.15: d-RFSB and m-ATCS heuristics' average % deviations from UB_{LPVI}) for the problems with size of 100

			d-RFSB Heuristic			m-ATCS Heuristic		
n	τ	R	Max	Min	Avg	Max	Min	Avg
100	0.1	0.1	4%	2%	3%	12%	3%	6%
		0.3	6%	2%	4%	9%	2%	3%
		0.5	6%	2%	4%	4%	1%	2%
		0.7	6%	2%	5%	4%	0%	1%
		0.9	9%	3%	6%	1%	0%	0%
	0.3	0.1	12%	6%	9%	13%	6%	8%
		0.3	11%	7%	9%	15%	4%	7%
		0.5	12%	7%	10%	8%	3%	4%
		0.7	16%	9%	11%	9%	2%	4%
		0.9	15%	8%	12%	5%	1%	2%
	0.5	0.1	18%	12%	14%	20%	12%	15%
		0.3	18%	12%	16%	21%	8%	13%
		0.5	21%	14%	18%	14%	8%	11%
		0.7	21%	13%	17%	13%	5%	9%
		0.9	18%	10%	14%	6%	1%	4%
	0.7	0.5	22%	13%	17%	19%	11%	15%
		0.7	18%	11%	14%	13%	9%	11%
		0.9	19%	10%	13%	14%	5%	9%
	0.9	0.9	15%	10%	13%	19%	10%	14%
		Avg	14%	8%	11%	12%	5%	7%

Table 5.16: MILPH gaps from UB_{LPVI} and CPU times for problem sizes of 15 and 20.

			% Deviation of MILPH			MILPH CPU Time		
n	τ	R	Max	Min	Average	Max	Min	Average
15	0.1	0.1	14%	1%	7%	4.42	0.52	1.65
		0.3	15%	2%	7%	3600.02	0.33	362.24
		0.5	9%	0%	4%	5.36	0.23	1.55
		0.7	14%	0%	4%	7.13	0.33	2.52
		0.9	9%	1%	5%	11.59	0.75	3.97
	0.3	0.1	22%	5%	13%	765.66	0.67	80.92
		0.3	20%	7%	14%	18.98	1.38	5.72
		0.5	15%	1%	8%	60.03	0.58	10.58
		0.7	13%	3%	9%	17.28	1.13	4.32
		0.9	16%	4%	9%	7.67	0.97	2.99
	0.5	0.1	28%	8%	18%	74.83	2.66	20.14
		0.3	20%	13%	16%	18.72	2.34	7.39
		0.5	29%	13%	20%	59.94	2.02	12.30
		0.7	21%	3%	12%	11.86	0.67	4.56
		0.9	23%	7%	13%	43.06	0.52	7.15
	0.7	0.5	37%	10%	21%	8.91	0.34	1.73
		0.7	26%	7%	17%	31.84	0.50	4.67
		0.9	22%	4%	15%	426.41	0.36	44.70
	0.9	0.9	34%	12%	23%	1.64	0.31	0.72
	Average		20%	5%	12%	272.39	0.87	30.52
20	0.1	0.1	17%	2%	9%	2093.00	2.89	263.26
		0.3	12%	5%	7%	3600.03	4.51	393.91
		0.5	16%	2%	6%	3600.03	2.70	733.46
		0.7	11%	2%	5%	3600.03	3.45	737.14
		0.9	15%	2%	6%	3600.03	2.50	519.27
	0.3	0.1	18%	4%	11%	3600.13	38.86	1005.15
		0.3	16%	5%	11%	1191.72	3.38	453.98
		0.5	20%	5%	10%	1441.64	36.41	533.22
		0.7	21%	4%	13%	1731.63	10.55	856.38
		0.9	21%	2%	11%	2212.00	8.33	500.22
	0.5	0.1	31%	12%	20%	3600.09	394.91	2816.86
		0.3	33%	12%	21%	3601.44	261.84	2280.36
		0.5	33%	9%	18%	3600.05	8.27	1011.80
		0.7	17%	9%	14%	1092.84	22.31	341.04
		0.9	24%	4%	15%	3600.03	10.02	763.90
	0.7	0.5	24%	16%	20%	287.61	2.02	38.97
		0.7	25%	9%	16%	3600.03	13.58	458.20
		0.9	21%	7%	16%	3600.03	6.08	479.72
	0.9	0.9	27%	18%	21%	3600.03	4.61	378.65
	Average		21%	7%	13%	2802.76	44.06	766.60

Chapter 6

FURTHER STUDIES

In this study, we propose a mathematical model and several heuristics in order to solve the OAS problem which are verified to work efficiently. In this chapter, we give possible directions for future study, and preliminary results for these directions. First, an exact method is mentioned, and next, the ways to develop an efficient heuristic or metahuristic are described.

6.1 Exact Method Development

As a further study, a branch and bound algorithm that is specially designed for the OAS problem may be developed to solve larger problem instances exactly. As we discussed earlier, the OAS problem is NP-hard and thus solution space grows exponentially, which makes the exact solution methods are not promising to solve large size problems. However, an efficient problem-specific branch and bound algorithm may increase the limit of the solvable size to 20-orders. Below, we present several dominance properties mainly on order sequencing that can be used in a branch and bound algorithm to eliminate certain non-optimal branches of the tree. First, we introduce additional notation which will be used in dominance relations:

J: the set of all orders, $J = 0, 1, 2, ..., n+1$, where 0 and $n+1$ are artificial orders;

K: the partial sequence starting with the artificial order 0, corresponding to a node at level k in the research tree, in which orders in the first $k+1$ positions have been fixed.

$J(K)$: the set of orders in the partial schedule K,

$U(K)$: the set of unscheduled orders, given the partial schedule K,

$C(K)$: the completion time of the last order in K,

S: a full feasible schedule.

1. Dominance property that apply to partial schedules:

Suppose we have a partial schedule K, where k orders are scheduled and where $[j]$ denotes the order scheduled at the j^{th} position in this partial schedule. Suppose further that we want to add an order from the set $U(K)$ to the current schedule K.

Let $C_{[k]}$ be the completion time of the order scheduled at position k, and $s_{[k,u]}$ be the setup time required to schedule order u right after the order at position k, $[k]$. If $\exists\ u$ such that $C_{[k]} + s_{[k,u]} + p_u > \bar{d}_u$; then adding this order u to the partial sequence is infeasible since deadline of job u is violated. Hence, we can prune this branch.

2. Dominance properties to determine the first and the last elements of the optimal schedule:

(a) Determining the first order: Suppose we have an order j satisfying $d_j < \min\limits_{i \in J, i \neq j} r_i$. Since $r_j + s_{0,j} + p_j > d_j$ by data generation, there exists an optimal schedule, where j is the first job to be processed.

(b) Determining the last order: Suppose we have an order j satisfying $r_j > \max\limits_{i \in J, i \neq j} \bar{d}_i$. Since $r_j + maxsetup_j + p_j > d_j$ by data generation, there exists an optimal schedule, where j is the last job to be processed.

3. Dominance properties that apply to full schedules:

Dominance properties proposed in this section are the modified versions of the dominance properties proposed in [17]. We extend these dominance properties to apply on the OAS problem.

Theorem 1 *If $\exists i$, $[i] \in S$ such that n_1 is the number of jobs from the job at position i to job at position k, n_2 is the number of jobs from the job at position u to the job at the last position, and Δ_T is the change of the total tardiness, $\Delta_1 = s_{[i-1][i+1]} - s_{[i-1][i]} - s_{[i][i+1]} - p_{[i]} \leq 0$, $\Delta_2 = s_{[i-1][i+1]} + s_{[k][i]} + s_{[i][u]} - s_{[i-1][i]} - s_{[i][i+1]} - s_{[k][u]} \leq 0$, $\Delta_T = n_1\Delta_1 + n_2\Delta_2 + C_{[k]} - C_{[i-1]} + s_{[k][i]} - s_{[i-1][i]} \leq 0$, release date constraints are satisfied for all jobs in schedule S' and deadline constraint of job $[i]$ is satisfied in S'; then the schedule S' dominates the schedule S.*

Proof Consider a feasible schedule S as shown in Figure 6.1. We form another schedule, S', by inserting job $[i]$ after job $[k]$. Until the end of processing job $[i-1]$, $T_s = T_{s'}$ and $C_s = C_{s'}$. In this part, since each order is completed at exactly the same time in S' as it is in S, the release date and the deadline of any order are satisfied; and $C_{[i-1]} = C'_{[i-1]}$.

Since the orders in the $[i+1]$st to the $[k]$th position are processed at different times in schedules S and S', we need to compare the completion times $C_{[j]}$ and $C'_{[j]}$ for these orders. $C_{[i+1]} = C_{[i-1]} + s_{[i-1][i]} + w_{[i]} + s_{[i][i+1]} + p_{[i+1]}$ and $C'_{[i+1]} = C_{[i-1]} + s_{[i-1][i+1]} + p_{[i+1]}$ for order $[i+1]$. Therefore, since the triangular inequality is assumed to be satisfied among

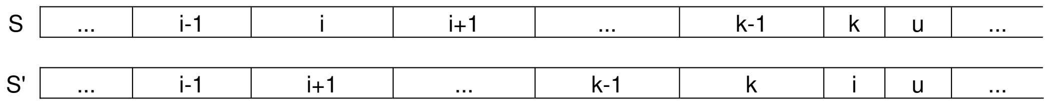

Figure 6.1: Inserting job $[i]$ after job $[k]$.

setup times, $C'_{[i+1]} \leq C_{[i+1]}$. $[i+1]$st job is completed at an earlier time in schedule S' then it is in schedule S. This relationship applies to all orders up to $[k]$. This condition provides that all orders from $[i+1]$ up to and including order $[k]$ are completed at Δ_1 time units earlier in schedule S' than they are in schedule S. This also ensures that deadlines of orders from order $[i+1]$ up to and including order $[k]$ are satisfied and the tardiness of these orders in schedule S' cannot be greater than the tardiness of these orders in schedule S. The tardiness gain is $(n_1 - 1)\Delta_1$.

Now consider the orders scheduled after order $[u]$. Because we have $\Delta_2 \leq 0$, $C_{[u]} \geq C'_{[u]}$. Then, each of the orders scheduled after $[u]$ are completed at Δ_2 time units earlier in schedule S' than they are in schedule S including order $[u]$. So, the change in the tardiness is Δ_2 for each of these orders. This ensures that deadlines of the orders from the order $[u]$ up to and including the order at last position are satisfied. The tardiness gain is $n_2\Delta_2$.

Now in order to dominate S, gain of the tardiness by scheduling orders $[i+1]$ to $[k]$ at earlier positions should be greater than the additional tardiness incurred by order $[i]$ by positioning it at a later position. Let the required time to complete the part of the schedule separately from order $[i+2]$ up to $[k]$ including order $[k]$ be F where F includes setup time for order $[i+2]$. The value of F is the same for both S and S'. Since $T'_{[i]} = C_{[i-1]} + s_{[i-1][i+1]} + p_{[i+1]} + F + s_{[k][i]} + p_{[i]} - d_{[i]}$ and $T_{[i]} = C_{[i-1]} + s_{[i-1][i]} + p_{[i]} - d_{[i]}$, $T'_{[i]} - T_{[i]} = s_{[i-1][i+1]} + p_{[i+1]} + F + s_{[k][i]} - s_{[i-1][i]}$. We can write F explicitly as $F = C_{[k]} - C_{[i-1]} - s_{[i-1][i]} - p_{[i]} - s_{[i][i+1]} - p_{[i+1]}$. When we plug the explicit form of F in the inequality, we get $T'_{[i]} - T_{[i]} = C_{[k]} - C_{[i-1]} + \Delta_1 + s_{[k][i]} - s_{[i-1][i]}$. The additional tardiness incurred by placing the order $[i]$ at a later position is $C_{[k]} - C_{[i-1]} + \Delta_1 + s_{[k][i]} - s_{[i-1][i]}$.

Then the change of the total tardiness is $\Delta_T = (n_1 - 1)\Delta_1 + n_2\Delta_2 + T'_i - T_i$, which is equal to $n_1\Delta_1 + n_2\Delta_2 + C_{[k]} - C_{[i-1]} + s_{[k][i]} - s_{[i-1][i]}$, where $\Delta_1 \leq 0$ and $\Delta_2 \leq 0$.

Therefore, $T_{s'} \leq T_s$ and the schedule S' dominates the schedule S. $\square$

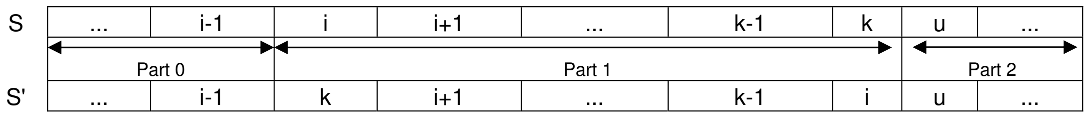

Figure 6.2: Interchanging of positions of jobs $[i]$ and $[k]$.

Theorem 2 *If $\exists i$, $[i] \in S$,such that n_1 is the number of jobs from the job at position i to job at position k, i.e. [i],...,[k], n_2 is the number of jobs from the job at position u to the job at the last position, i.e. [u],...,[last position] and Δ_T is the change of the total tardiness, $\Delta_1 = s_{[i-1][k]} + p_{[k]} + s_{[k][i+1]} - s_{[i-1][i]} - p_{[i]} - s_{[i][i+1]} \leq 0$, $\Delta_2 = s_{[i-1][k]} + s_{[k][i+1]} + s_{[k-1][i]} + s_{[i][u]} - s_{[i-1][i]} - s_{[i][i+1]} - s_{[k-1][k]} - s_{[k][u]} \leq 0$, $\Delta_1 + s_{[i-1][k]} + s_{[k-1][i]} - s_{[i-1][i]} - s_{[k-1][k]} \leq 0$, $\Delta_1(n_1 - 1) + \Delta_2 n_2 + s_{[i-1][k]} + s_{[k-1][i]} - s_{[i-1][i]} - s_{[k-1][k]} \leq 0$, the deadline constraint of order [i] is satisfied and if the release date constraints are satisfied for all orders beginning from order k, that is, if the constraints:*

$C_{i-1} \geq r_k,\ C_k \geq r_{i+1},\ \ldots,\ C_{k-2} \geq r_{k-1},\ C_{k-1} \geq r_i,\ \ldots$

are not violated, then the new schedule S' formed by exchanging orders $[i]$ and $[k]$ dominates the existing schedule S.

Proof Consider a feasible schedule S. We construct another schedule S' by exchanging the positions of orders at positions $[i]$ and $[k]$ as in Figure 6.2. In the new schedule, we assume that release date constraints are satisfied, that is, completion time of each order should be greater than or equal to the release date of its immediate successor. In such a schedule S', we will have three parts:

In Part 0: $T_s = T_{s'}$ and $C_s = C_{s'}$. In this part, since each order is completed at exactly the same time in S' as it is in S, the release dates and the deadlines of all orders are satisfied.

In Part 1: In this part, the completion time of each order $[l]$ has a change of $\Delta_1 \leq 0$, so $T'_{[l]} \leq T_{[l]}$, for indexes $i+1 \leq l \leq k-1$. Since none of the orders are completed at a later time due to $\Delta_1 \leq 0$, deadline constraints are satisfied in Part 1. The gain of tardiness for each of the orders between $[i+1]$ and $[k-1]$, including $[i+1]$ and $[k-1]$ is Δ_1 time units. Then, the total gain of tardiness is $(n_1-2)\Delta_1$ in part 1.

In Part 2: Change of completion time of order $[u]$ is Δ_2. Since $\Delta_2 \leq 0$, we have $C_u \geq C'_u$, which ensures that deadline constraints are not violated in this part. Therefore, any sequence comprised of all the jobs in $U(K)$ can be scheduled earlier in S' in part 2. That is, $T_{s'} \leq T_s$. Gain of tardiness by scheduling jobs $[u],\ldots,[lastposition]$ in part 2 is $\Delta_2 n_2$. Then the total gain of tardiness in part 1 and part 2 is $\Delta_1(n_1-2)+\Delta_2 n_2$.

The change of tardiness caused by interchanging $[i]$ and $[k]$:

$T'_{[i]} - T_{[i]} + T'_{[k]} - T_{[k]} = \Delta_1 + s_{[i-1][k]} + s_{[k-1][i]} - s_{[i-1][i]} - s_{[k-1][k]} \leq 0.$

Finally, the total change of tardiness of schedule S':

$\Delta_T = T_{s',Part0} + T_{s',Part1} + T_{s',Part2} - T_{s,Part0} - T_{s,Part1} - T_{s,Part2} + \Delta_1(n_1-2) + \Delta_2 n_2$ which is equal to $\Delta_1 + s_{[i-1][k]} + s_{[k-1][i]} - s_{[i-1][i]} - s_{[k-1][k]} + \Delta_1(n_1-2) + \Delta_2 n_2$, where $\Delta_1 \leq 0$ and $\Delta_2 \leq 0$.

Therefore, the new schedule S' dominates the existing schedule S. $\square$

Theorem 3 *If $\exists i$, $[i] \in S$,such that n_1 is the number of jobs from the job at position $i+2$ to the job at last position , i.e. $[i+2],\ldots,[lastposition]$, if $\Delta_1 = s_{[i-1][i+1]} + p_{[i+1]} + s_{[i+1][i]} - s_{[i-1][i]}$, $\Delta_2 = s_{[i-1][i+1]} - s_{[i-1][i]} - p_{[i]} - s_{[i][i+1]} \leq 0$, $\Delta_3 = s_{[i-1][i+1]} + s_{[i+1][i]} + s_{[i][i+2]} - s_{[i-1][i]} - s_{[i][i+1]} - s_{[i+1][i+2]} \leq 0$, $\Delta_1+\Delta_2+n_1\Delta_3 \leq 0$ and release date of each job is satisfied in the new schedule, then the new schedule S' formed by exchanging the adjacent orders $[i]$ and $[i+1]$ dominates the existing schedule S.*

Proof Consider a feasible schedule S. We construct another schedule S' by exchanging the positions of adjacent orders at positions $[i]$ and $[i+1]$ as in Figure 6.3. In the new schedule, we assume that release date constraints are satisfied, that is, completion time of each order should be greater than or equal to the release date of its immediate successor. In such a schedule S', we will have three parts:

In Part 0: $T_s = T_{s'}$ and $C_s = C_{s'}$. In this part, since each order is completed at exactly the same time in S' as it is in S, the release date and the deadline of any order are satisfied. (Part 0 ends after processing of $(i-1)$st job).

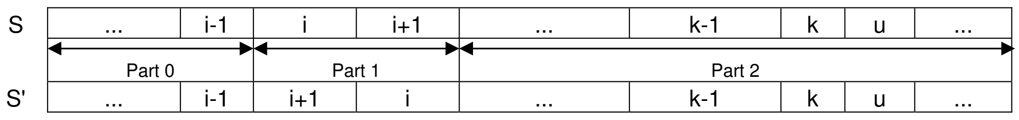

Figure 6.3: Interchanging of positions of adjacent jobs $[i]$ and $[i+1]$.

In Part 1: The extra tardiness incurred by order $[i]$ by positioning it to $i+1$st position is $\Delta_1 = s_{[i-1][i+1]} + p_{[i+1]} + s_{[i+1][i]} - s_{[i-1][i]}$. The tardiness of $[i+1]$ cannot increase, because it is completed at an earlier time than it is in schedule S. The gain of tardiness by scheduling $[i+1]$ to the ith position is $\Delta_2 = s_{[i-1][i+1]} - s_{[i-1][i]} - p_{[i]} - s_{[i][i+1]}$.

In Part 2: The completion time of order $[i+2]$ has a change of $\Delta_3 = s_{[i-1][i+1]} + s_{[i+1][i]} + s_{[i][i+2]} - s_{[i-1][i]} - s_{[i][i+1]} - s_{[i+1][i+2]} \leq 0$ in S'. Then, each of the orders scheduled after $[i+2]$ are completed at Δ_3 time units earlier in schedule S' than they are in schedule S including order $[i+2]$. Since order $[i+2]$ is not completed at a later time in S than it is in S' , deadline constraint of $[i+2]$ and the deadlines of all other orders coming after $[i+2]$ are also satisfied. Total gain of tardiness in part 2 is $n_1\Delta_3$.

Then, the change of total tardiness by exchanging the orders $[i]$ and $[i+1]$ is $\Delta_T = \Delta_1 + \Delta_2 + n_1\Delta_3$.

In the light of these data, we can summarize that $T_s = T_{s'}$ in part 0, and $T_s \geq T_{s'}$ in parts 1 and 2. Therefore, schedule S' dominates schedule S. $\square$

6.2 Heuristic-Metaheuristic Method Development

In this thesis we developed one hybrid metaheuristic (ISFAN), two constructive heuristics (d-RFSB and m-ATCS heuristics), and one MILP model based heuristic (MILPH) for finding efficient solutions to the OAS problem. ISFAN, d-RFSB and m-ATCS heuristics are proved to perform well in certain cases. For instance, ISFAN performs well on the average when $n \leq 50$, and m-ATCS performs well when n is large enough (i.e. $n > 50$). An idea for an efficient heuristic method development may be to combine the strengths of the proposed heuristics.

This combining issue may be handled by assigning a probability for each heuristic, and select the heuristics according to these probabilities for each instance.

Another approach may be using population based algorithms for the OAS problem. A metaheuristic such as Genetic algorithm (GA) may help combining two existing sequences' good parts and may yield a better sequence overall. Of course, any population algorithm requires a good representation of individuals. Currently, there is no population based algorithm developed and applied to simultaneous order acceptance and scheduling problems. This is because defining the elements of the population based algorithms appropriately is not an easy task.

As mentioned in Section 3.3, the OAS problem has a structure similar to TSP. The previous studies in the literature indicates that Tabu Search (TS) works well for TSP compared to other metaheuristics. Therefore an appropriately defined TS algorithm has a potential to improve the solutions found by the methods proposed in this thesis. The main challenge for the mentioned metaheuristics GA and TS is to have a proper design that will take the characteristics of the OAS problem into account. Hence we will consider this avenue as one of our future directions.

Chapter 7
CONCLUSIONS

In this thesis, we study the simultaneous order acceptance and single machine scheduling problem (OAS). The OAS problem that is undertaken throughout this study is a problem that is faced by make-to-order firms which produce or offer unique and customized products/services. In the OAS, we need to give two tough decisions: which orders to accept, and how to sequence the accepted orders. The revenues gained from each of the orders are affected from each order's weighted tardiness. Therefore, total revenue gained by the manufacturer is time-dependent since the tardiness amounts depend on completion times.

Throughout this thesis, we first define the OAS problem and examine the complexity of the problem. We show that the OAS problem is strongly NP-hard, as it can be reduced to the total weighted tardiness problem. Since the problem has not studied before to the best of our knowledge, no test data was available. However, Slotnick and Morton [28] studies a special case of the problem. Hence, we generated the OAS data randomly. We used parameters such as tardiness factor and due date range coefficients to obtain different types of instances.

We proposed a mixed integer linear programming model (MILP) for the OAS problem and solved it by CPLEX 10.0 solver. As it was expected, we observed that MILP is able to solve the OAS problem up to a very limited size, only when $n \leq 15$, in one-hour CPU time limit on our computational platform. It is observed that when the tardiness factor is small, the MILP model is harder to solve. The reason might be that, as this factor gets larger, more orders are rejected, hence size of the sequencing problem decreases. We developed new heuristic methods that work for larger instances. In order to compare and measure the performances of the newly designed methods we needed to develop upper bounds. An easy-to-find upper bound is the LP relaxation bound of the MILP model, where integrality restrictions on the sequencing variables are relaxed. It's observed that, the LP relaxation solution divides the orders into small partitions, and behaves as if we have unlimited number of parallel machines in the system. Hence the completion times of orders can not be calculated correctly and for this reason, no orders are rejected in the LP relaxation solution. Therefore, the LP relaxation can not provide tight bounds for the cases in which many orders should be rejected. In order to strengthen this upper bound, we forced the relaxed model to reject at least some of the orders by means of three valid inequalities. When these valid inequalities are added to the LP relaxation, the upper bound improves by 2% on the average. In some cases, the improvement is significant. For example for tardiness factor= 0.5 and due date range= 0.1, the average % improvement is 11% for 15-order problems. When the tardiness factor is large, more orders are expected to be tardy. In addition, when due date range is small, due dates of the orders are close to each other. Therefore, more orders are rejected when tardiness factor is large and due date range is small, and the valid inequalities may detect orders to be rejected. It is observed that the largest number of rejections occur for the cases where tardiness factor is large and due date range is small, on the average when solved by the proposed heuristics. When tardiness factor is small and due date range is large, the number of rejected orders are small. We can conclude that when there is a high potential for improving the bound by rejecting some orders, the LP relaxation of MILP model with valid inequalities improves the upper bound better.

We developed several heuristics to solve the large size instances. We proposed a hybrid metaheuristic algorithm that uses simulated annealing concepts in sequencing decisions and the revenue-load ratio to decide on which orders to reject in the order acceptance decisions. We named this algorithm ISFAN. We employed local search algorithms, order insertion and order exchange to improve the results of ISFAN. ISFAN is able to solve 50-order problems in less than 10 minutes on the average in our test problems. However, the CPU time increases as n gets larger. The most time consuming part of ISFAN is found to be the rejection iteration which includes finding a local optimal sequence for the set of non-rejected orders by SA. Hence, as the number of required rejection iterations increase, the required CPU time for ISFAN increases. It was also observed that order insertion and order exchange improve the ISFAN results up to 20% for 50-order problems. Overall, ISFAN is found to be a good performing heuristic by our computational experiments.

As alternatives to the static ISFAN algorithm, we developed two dynamic constructive heuristics, d-RFSB and m-ATCS heuristics. These heuristics differ from ISFAN in the following way: In ISFAN, we initially accept all of the orders, and reject one at each rejection iteration. In these dynamic heuristics, we have no accepted orders at the beginning, and by checking the availability of orders and the machine, and the feasibility of sequencing the order in a prespecified position, we accept one order at a time. To decide on which order to accept and sequence next, we use two different rules for each of the heuristics. These constructive heuristics may give better decisions on sequencing and order selection, compared to ISFAN, they use more information such as the sequence dependent setup time as we know which order is scheduled last. The objective function of the OAS problem is dependent on weighted tardiness of orders, and hence indirectly relates to time. Therefore giving the acceptance and sequencing decisions dynamically and adjusting them according to the time generates better results for larger instances of the OAS problem. The d-RFSB and m-ATCS heuristics are fast and produce high quality solution solutions. m-ATCS algorithm dominates both d-RFSB and ISFAN heuristics for $n \geq 50$. This is because ISFAN requires larger numbers of neighborhood operations as the problem size increases.

Finally, we developed a MILP heuristic in which we relax the constraint set (3.3) and solve the MILP model accordingly. As a result we end up with a solution that may consist of subtours. We eliminate these subtours and get a single tour by adding the constraints (3.21), (3.22) and (3.23). We then feed the single tour to the MILP model by means of inequalities and solve the model using CPLEX 10.0. The MILP heuristic did not perform well in terms of time and objective value considerations.

To summarize the contributions we made to the literature, we defined and considered simultaneous order acceptance and scheduling problem that was not dealt in the literature before. Since the problem is newly defined, we developed the benchmark data that is required for the computational tests. We showed that the OAS problem is strongly NP-hard, and proposed valid inequalities to improve the upper bound. We also proposed four methods, ISFAN, d-RFSB, m-ATCS and MILP heuristics in addition to developing a mixed integer linear programming model. ISFAN and the constructive algorithms, d-RFSB and m-ATCS heuristics, performed well both for the OAS problem and its special case (S&M problem) in the computational experiments. Finally, we presented future study directions and our findings including several dominance properties that may be used in these studies.

BIBLIOGRAPHY

[1] 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.

[2] K. Charnsirisakskul, P. Griffin, and P. Keskinocak. Pricing and scheduling decisions with lead-time flexibility. *European Journal of Operational Research*, 171:153–169, 2006.

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

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

[5] I. Duenyas and W.J. Hopp. Quoting customer lead times. *Management Science*, 41(1):43–57, 1995.

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

[7] 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.

[8] P. Keskinocak and S. Tayur. *Handbook of Quantitative Supply Chain Analysis: Modeling in the E-Business Era.*, chapter Due date management policies, page 485553. International Series in Operations Research and Management Science. Kluwer Academic Publishers, Norwell, MA, 2004.

[9] S.C. Kim and P.M. Bobrowski. Scheduling jobs with uncertain setup times and sequence dependency. *Omega, International Journal of Management Science*, 25(4):437–447, 1997.

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

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

[12] K.L. Kreamer, J. Dedrick, and S. Yamashiro. Refining and extending the business model with information technology: Dell computer corporation. *The Information Society*, 16:5–21, 2000.

[13] E.L. Lawler. A 'pseudopolynomial' algorithm for sequencing jobs to minimize total tardiness. *Annals of Discrete Mathematics*, 1:331–342, 1977.

[14] E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan. *Deterministic and Stochastic Scheduling*, chapter Recent developments in deterministic sequencing and scheduling: a survey, pages 35–73. Reidel, Dordrecht, 1982.

[15] 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.

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

[17] X. Luo and C. Chu. A branch-and-bound algorithm of the single machine schedule with sequence-dependent setup times for minimizing maximum tardiness. *European Journal of Operational Research*, 180(1):68–81, 2007.

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

[19] C. Miller, A. Tucker, and R. Zemlin. Integer programming formulations and travelling salesman problems. *Journal of the Association for Computing Machinery*, 7:326–329, 1960.

[20] D.M. Miller, H. Chen, J. Matson, and Q. Liu. A hybrid genetic algorithm for the single machine scheduling problem. *Journal of Heuristics*, 5:437–454, 1999.

[21] A. Nandi and P. Rogers. Using simulation to make order acceptance/rejection decisions. *Simulation*, 80(3):131–142, 2004.

[22] P.S. Ow and T.E. Morton. The single machine early/tardy problem. *Management Science*, 35(2):177–191, 1989.

[23] 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.

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

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

[26] L. Schrage and K. Baker. Dynamic problem solution of sequencing problems with precedence constraints. *Operations Research*, 26:444–449, 1978.

[27] S.A. Slotnick and T. E. Morton. Selecting jobs for a heavily loaded shop with lateness penalties. *Computers and Operations Research*, 23(3):131–140, 1996.

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

[29] K.C. Tan and R. Narasimhan. Minimizing tardiness on a single processor with sequence dependent setup times: A simulated annealing approach. *Omega, International Journal of Management Science*, 25(6):619–634, 1997.

[30] 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, International Journal of Management Science*, 28:313–326, 2000.

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

[32] S. Webster. Dynamic pricing and lead-time policies for make-to-order systems. *Decision Sciences*, 33(4):579–599, 2002.

[33] 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.

[34] M.J.F. Wouters. Relevant cost information for order acceptance decisions. *Production Planning and Control*, 8(1):2–9, 1997.

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

[36] R.B. Yehuda, G. Even, and S.(M.) Shahar. On approximating a geometric prize-collecting traveling salesman problem with time windows. *Journal of Algorithms*, 55:76–92, 2005.