Mathematical Models and Heuristic Algorithms for the Order Acceptance and Scheduling Problems

by

İstenç Tarhan

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Doctor of Philosophy

in

Industrial Engineering

KOÇ ÜNİVERSİTESİ

January 6, 2020

Mathematical Models and Heuristic Algorithms for the Order Acceptance and Scheduling Problems

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a doctoral dissertation by

İstenç Tarhan

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:

_____

Prof. Ceyda Oğuz (Advisor)

_____

Prof. Sibel Salman

_____

Assist. Prof. Bahriye Cesaret

_____

Prof. Selçuk Karabatı

_____

Prof. Necati Aras

Date: _____

To my beloved family

ABSTRACT

Mathematical Models and Heuristic Algorithms for the Order Acceptance and Scheduling Problems
İstenç Tarhan
Doctor of Philosophy in Industrial Engineering
January 6, 2020

In make-to-order production systems, manufacturers often do not keep inventory of final goods beforehand which necessitates utilizing the production capacity more efficiently to be able to complete orders in more timely manner. Yet, it may not be possible to complete all orders on time when the order delivery time requirements of customers get more tight under a limited production capacity. In this case, manufacturers have to reject some orders and should simultaneously decide which orders to accept and how to schedule accepted orders since both decisions strictly depend on each other. The corresponding problem is called as the order acceptance and scheduling problem in the literature. In this thesis, we study the order acceptance and scheduling problem with release times and sequence dependent setup times that determines the set of accepted orders and their schedule so as to maximize the total revenue.

In this thesis, a mixed integer and a constraint programming model, and a matheuristic algorithm along with a new relaxed time-indexed model formulation, a variable neighborhood and a tabu search algorithm to be used within are presented for the generalized order acceptance and scheduling problem. Computational results show that the proposed models achieve smaller optimality gaps than the existing models in the literature and the proposed matheuristic algorithm outperforms both the proposed models and the state-of-the-art algorithms for the order acceptance and scheduling problem in the literature. New optimal solutions are also identified by the proposed models and the matheuristic algorithm.

Then, the order acceptance and scheduling problem is extended to include batch delivery of orders. In this extension, orders are not delivered individually (i.e., upon their completion) but rather are delivered in batches which is observed more often

in real-life practices. Mathematical models developed for the original problem are adapted to this extension. To tackle large size problem instances in which these models do not perform well, we propose two different iterated local search (ILS) algorithms. The first ILS employs a variable neighborhood search algorithm with alternating objective functions and the second ILS utilizes a tabu search algorithm as its local search mechanism. Computational results show that the proposed models achieve small optimality gaps for the small size problems. However, their performances deteriorate significantly as problem size increases. For medium and large size instances, the first ILS algorithm using the proposed alternating objective functions achieves smaller optimality gaps than both the proposed models and the second ILS algorithm with tabu search.

ÖZETÇE

Sipariş Kabulü ve Çizelgeleme Problemleri için Matematiksel Modeller ve Sezgisel Algoritmalar
İstenç Tarhan
Endüstri Mühendisliği, Doktora
6 Ocak 2020

Sipariş üzerine çalışan üretim sistemlerinde, üreticiler genellikle nihai ürün için önceden envanter tutmazlar; bu da, tüm siparişleri zamanında tamamlayabilmek için üretim kapasitesinin verimli bir şekilde kullanılmasını gerektirir. Ancak müşterilerin, siparişlerin dağıtım süresi ile ilgili talepleri sıkılaştıkça, kapasitenin kısıtlı olmasından dolayı tüm siparişleri zamanında tamamlamak mümkün olmayabilir. Bu durumda, üreticiler bazı siparişleri reddetmek zorunda kalır ve hangi siparişlerin kabul edileceğine ve kabul edilen siparişlerin çizelgelemesine eşzamanlı olarak karar vermek gerekir. İlgili problem literatürde, sipariş kabulü ve çizelgeleme problemi olarak adlandırılmaktadır. Bu tezde, siparişlerin serbest kalma zamanlarını ve siparişler arasındaki sıraya bağlı hazırlama sürelerini de içeren ve kabul edilen siparişleri ve bunların çizelgelemesini belirleyerek geliri enbüyükleyen genelleştirilmiş sipariş kabulü ve çizelgeleme problemini çalıştık.

Genelleştirilmiş sipariş kabulü ve çizelgeleme problemi için, karışık tamsayı programlama modeli, kısıt programlama modeli ve yeni bir gevşetilmiş zaman indeksli model formülasyonu, değişken komşu arama ve tabu arama algoritmalarını bünyesinde barındıran bir mat-sezgisel algoritma sunulmuştur. Sayısal deneyler önerilen modellerin literatürdeki modellerden daha az eniyilik boşluğu ile sonuçlandığını gösterirken, önerilen mat-sezgisel algoritmanın da önerilen modellerden ve literatürdeki en gelişkin algoritmalardan daha iyi sonuç verdiğine işaret ediyor.

Tezin bir sonraki aşamasında, sipariş kabulü ve çizelgeleme problemi, ürünlerin öbekler halinde dağıtılmasını da dahil ederek genişletilmiştir. Bu genişletilmiş problemde, siparişler tek tek gönderilmek yerine, pratikte daha sıklıkla gözlenildiği gibi, öbekler halinde gönderilmektedir. Bu problemin çözümü için, ele alınan ilk problem için geliştirilen matematiksel modeller bu probleme de uyarlanmıştır. Modellerin

başarısız olduğu büyük boyutlu problemlerin üstesinden gelebilmek için, iki farklı yinelemeli yerel arama algoritması önerilmiştir. Birinci algoritmada yerel arama algoritması olarak birbirini izleyen amaç fonksiyonlarını kullanan değişken arama algoritması uygulanırken, ikinci algoritmanın yerel arama mekanizması olarak tabu arama algoritması kullanılmıştır. Sayısal sonuçlar önerilen modellerin küçük boyutlu problemler için küçük eniyilik boşluğuna ulaştığını ancak büyük boyutlu problemlerde bu modellerin performansının önemli ölçüde düştüğünü göstermektedir. Orta ve büyük boyutlu problemlerde, birbirini izleyen amaç fonksiyonları yaklaşımını kullanan yinelemeli yerel arama algoritmasının hem önerilen modellerden hem tabu arama algoritmasını kullanan yinelemeli yerel arama algoritmasından daha küçük eniyilik boşluğuna ulaştığı görülmüştür.

ACKNOWLEDGMENTS

I would like to express my gratitude to many people who so generously contributed to the work presented in this thesis.

Special mention goes to my esteemed supervisor Ceyda Oğuz for the continuous support of my Ph.D study and research, for her motivation and immense knowledge that she is always enthusiastic to share unless you are not as motivated as she is. Her guidance helped me over the entire course of research and writing of this thesis. I am deeply grateful to her for enabling me to complete the Ph.D and most importantly, make me enjoy it.

Similar, profound gratitude goes to Sibel Salman and Bahriye Cesaret, who are the other members of the thesis monitoring committee and who helped me go through this process without any stress thanks to their academic support, valuable feedbacks and estimable courtesy. I also give my sincere gratitude to Selçuk Karabatı and Necati Aras for kindly participating to my Ph.D. thesis jury.

I have felt support of many friends at each single moment of the Ph.D. process. They are so many and I cannot name each of them herein. I owe them a lot and wish to remunerate their unconditional support in future.

Finally, but by no means least, thanks go to my dear family. They are the most important people in my world and I dedicate this thesis to them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

ABC	Artificial Bee Colony
CP	Constraint Programming
DP	Dynamic Programming
GOAS	Generalized Order Acceptance and Scheduling
GOASB	Generalized Order Acceptance and Scheduling with Batch Delivery
GOASB_ILS	An ILS Algorithm for the GOASB Problem
GOASB_ILS^{GTS}	GOASB_ILS That Uses a GTS Algorithm for the Local Search
GOASB_ILS^{VNSA}	GOASB_ILS That Uses a VNSA Algorithm for the Local Search
GOAS_I2PM	Iterative Two-Phase Matheuristic for the GOAS Problem
GOAS_MSGTS	Multi-Start GTS Algorithm for the GOAS Problem
GOAS_RTBM	Relaxed Time-Bucket Model Formulation for the GOAS Problem
GTS	Granular Tabu Search
ILS	Iterated Local Search
MILP	Mixed Integer Linear Programming
OAS	Order Acceptance and Scheduling
PL	Path Relinking
SA	Simulated Annealing
TS	Tabu Search
VNS	Variable Neighborhood Search
VNSA	Variable Neighborhood Search Based on Alternating Objective Functions

Chapter 1

INTRODUCTION

Scheduling is concerned with the allocation of scarce resources to activities with the objective of optimizing some performance measures. Resources and activities can take on many different forms depending on the context of the scheduling problem. Resources may be machines in an assembly plant, CPU, memory and I/O devices in a computer system, runways at an airport, mechanics in an automobile repair shop, etc. Activities may be various operations in a manufacturing process, execution of a computer program, landings and take-offs at an airport, car repairs in an automobile repair shop, and so on. There are also many different performance measures to optimize such as the minimization of the makespan (i.e., the completion time of the latest job), the minimization of the number of late jobs, the mean of the completion time of the jobs, etc. (Leung, 2004).

Initial efforts for scheduling headed towards managing activities occurring in a workshop in the 1950s. Companies can stay competitive by using scheduling algorithms to lower the production cost in a manufacturing process. The diversity of scheduling problems has increased considerably later on though the principal objective of them has remained same: Satisfying customers' orders by utilizing the scarce resources as efficiently as possible. However, in a firm that strives to align its functions so that profit is maximized, the coordination of capacity with demand may require business sometimes to be turned away (Slotnick, 2011). This case, that is declining some of the demand, is mostly observed in make-to-order systems. As firms operating on a make-to-order basis do not keep the inventory of final products beforehand and customers have delivery time requirements for their demand, satisfaction of the entire demand may not be possible due to capacity limitations faced

by the firm. When limited production capacity and tight delivery requirements necessitate selecting orders to maximize total revenues, simultaneous order acceptance and scheduling decisions should be made. As the firms increasingly offer more customized and unique products in order to attract customers, manufacturers operate on a make-to-order basis to realize benefits in terms of reduced inventory cost and decreased risk to vulnerabilities. In this case, limitations on manufacturing capacity force these firms to be more selective on customer orders (Oğuz et al., 2010).

To tackle the necessity of handling order acceptance and scheduling decisions simultaneously, "order acceptance and scheduling problems" have arisen. The problem of order acceptance and scheduling (OAS) is defined as the joint decision of which orders to accept for processing and how to schedule them. Decision-maker is faced with a collection or stream of orders whose combined processing requirements would exceed available capacity, and has the option of rejecting some of those orders. If all orders can be accepted, the problem reduces to the scheduling decision.

Although the first study considering the order acceptance and scheduling problem together was published in the late 1980s, escalation of the capacity limitations against delivery requirements has boosted the relevant studies considerably in the last decade. One of these studies is carried out by Oğuz et al. (2010). They propose a generalized order acceptance and scheduling (GOAS) problem that extends the original problem by incorporating sequence dependent setup times between orders and their release times.

The GOAS problem can be stated as $1\ |r_j, d_j, e_j, w_j, s_{i,j}|\sum R_j$ problem by the notation introduced by Graham et al. (1979) and it can be formally defined as follows: In a non-preemptive single machine environment, we are given a set of independent orders O with cardinality n at the beginning of the planning period. Each order $i \in O$ is characterized with release time r_i, processing time p_i, due date d_i, deadline $\bar{d}_i$ such that $d_i \leq \bar{d}_i$, sequence dependent setup time $s_{i,j}$ to be incurred before order j is processed if order i immediately precedes order j, revenue e_i which denotes the maximum gain from order i, and unit tardiness penalty cost w_i to denote the dissatisfaction of the customer toward the manufacturer in case the due date of

order i is violated. The manufacturer may violate the due date d_i until the deadline $\bar{d}_i$, but for each time unit beyond the due date, she incurs the tardiness penalty cost. Given a schedule σ_A of the accepted orders $A \subseteq O$, revenue generated from order i, denoted by $R_i(\sigma_A)$, is calculated as $R_i(\sigma_A) = max\{0, e_i - w_i * \text{tardiness of order } i\}$. Consequently, the total revenue gained from processing orders in schedule σ_A is $TR(\sigma_A) = \sum_{i \in A} R_i(\sigma_A)$. The generalized order acceptance and scheduling problem described as above aims to find a set A and a schedule σ_A when the total revenue is maximum.

This thesis centralizes on the GOAS problem due to mainly three reasons:

1. Opportunities for stronger alternative mathematical models.
2. Possibility to develop heuristic algorithms superior to the state-of-art-algorithms in the literature.
3. Variants of this problem awaiting consideration.

To this end, in this thesis, new mathematical models and heuristic algorithms are developed for the GOAS problem and for its extension, GOAS with batch delivery (GOASB), which is proposed in this thesis that includes the delivery of orders in batches.

The GOASB problem integrates the delivery of the orders to the customers with the scheduling decisions so as to decrease the delivery cost by batching the orders of same customer together. In the context of the GOASB problem, batch represents a group of orders that are loaded to the same vehicle and delivered together; therefore, it should be distinguished from the scheduling problems with batch processing. In the latter one, each order is associated with a family and a machine setup is required between two consecutively scheduled jobs from two different families (Zdrzałka, 1996) but not within the same family; groups of orders that are processed consecutively without any setups are called batches in relevant problem setting. However, in the GOASB problem, there is no pre-given families of orders. Orders that are delivered by the same vehicle are called batches and the batch delivery time is defined as the completion time of the latest scheduled order in a batch. Accordingly, the delivery time of an order corresponds to the delivery time of the batch containing this order.

In the current literature, the majority of the studies ignore the delivery of the orders scheduled as if it is independent of the scheduling decisions. On the other hand, in general, orders are delivered in batches consisting of multiple orders. Thus, the delivery cost and time of an order directly depend on the other orders in its batch. For example, an early completion of an order often does not bring an advantage to the firm if the other orders in its batch are completed late. Since hierarchical handling of scheduling and delivery decisions would not capture the relation between them, their simultaneous consideration is crucial. Besides, the delivery cost affects decisions on which orders to be accepted since the orders causing a high delivery cost are less likely to be accepted.

Although Shobrys and White (2002) state that companies pursuing integration among different decision levels in production management environments report substantial economic benefits, as pointed out by Chen (2004), most existing production-inventory distribution models study strategic or tactical levels of decisions, and few have addressed integrated decisions at the detailed job-scheduling level. The objective of GOASB problem is to maximize the net revenue which is equal to the gross revenue gained from accepted orders minus tardiness and delivery costs. Thus, the GOASB problem combines order acceptance, scheduling and delivery decisions that are significant parts of the supply chain management.

Major contributions of this thesis are outlined as follows:

- We exploit the strict link between travelling salesman problems and scheduling problems and develop a new mixed integer linear programming model for the GOAS problem by referring to a model in the travelling salesman literature. The proposed model can also be modified so as to represent travelling salesman problems with time windows or prize collecting problems.

- There are no studies, to the best of our knowledge, proposing a constraint programming model in the GOAS literature although it is proven to be an efficient method for many scheduling problems. Therefore, we develop a constraint programming model for the GOAS problem and show that its perfor-

mance is highly competitive compared to the state-of-the-art algorithms in the literature.

- A new matheuristic algorithm is developed for the GOAS problem. However, it is not peculiar to the GOAS problem and can be used to tackle any scheduling problems with a long planning horizon.

- Relaxation of time-indexed formulations is generally used to obtain strong dual bounds. However, these formulations can give key insights about the problem apart from the dual bounds. We show that it is a promising approach to use corresponding formulations within algorithms so as to take advantage of the information they provide.

- A new diversification scheme based on path relinking approach is proposed that can be promising for the problems including sequencing decisions. It achieves intensification and diversification simultaneously due to the exploitation of elite solutions.

- The GOAS problem is extended to the GOASB problem so that orders are delivered in batches, rather than individually. Flexibility of the mathematical models proposed for the GOAS problem is shown by modifying these models for the GOASB problem.

- A new heuristic approach is developed and applied to the GOASB problem. Most of the heuristic / metaheuristic algorithms can be interpreted as an iterative process between intensification and diversification phases. If the local search in the intensification phase is not sufficiently extensive, it is likely to resort to the diversification phase too early. In such cases, the proposed approach is promising to be used within the intensification phase due to its capability of steering the search to promising regions close to the current local optimum and to avoid an early jump to a large scale diversification.

The next chapter involves a literature review for both the order acceptance and scheduling problems and the scheduling problems with batch delivery. A mixed integer linear programming model and a constraint programming model along with a matheuristic algorithm for the GOAS problem are proposed in Chapter 3. In Chapter 4, mathematical models developed earlier for the GOAS problem are adapted to the GOASB problem. A new heuristic approach is developed and applied to the GOASB problem. Finally, Chapter 5 covers concluding remarks of this thesis.

Chapter 2

LITERATURE REVIEW

Both the GOAS problem and its extension, the GOASB problem, are addressed in this thesis. Thus, this chapter reviews the literature of order acceptance and scheduling problems, scheduling problems with batch delivery as well as their integration under different headings, to provide an organized review.

2.1 Order acceptance and scheduling problems

The OAS problem has been researched for almost three decades in the literature taking different forms regarding the order characteristics and the production environment. Wester et al. (1992) propose different heuristics for the OAS problem with setup times that use the number of accepted orders as the performance measure. The validity of the heuristics that produce good acceptance strategies is shown via simulations. Slotnick and Morton (1996) do not consider only the number of accepted orders but also study the maximization of the revenue of the accepted orders minus the lateness penalty. A branch-and-bound, a beam search and a myopic heuristic algorithm are proposed for this OAS problem. Ghosh (1997) shows that the corresponding order acceptance problem with lateness penalty is NP-hard. Lewis and Slotnick (2002) extend the OAS problem with lateness penalty to a multi-period environment such that each customer submits one job at each period until one of her jobs is rejected. Slotnick and Morton (2007) adjust the problem studied in Slotnick and Morton (1996) by considering the weighted tardiness instead of lateness and modify previously proposed algorithms of the former study in this respect. Rom and Slotnick (2009) study the same OAS problem, develop a genetic algorithm and show that it outperforms the myopic heuristic proposed by Slotnick and Morton (2007).

While these studies do not consider deadlines, Charnsirisakskul et al. (2004) make the distinction between a due date and a deadline. They define an order acceptance problem with leadtime flexibility. Customer does not place an order if the manufacturer cannot complete the order by the latest acceptable due date. Thus, an accepted order must be completed and shipped to the customer between the commitment time and the latest acceptable due date specified by the customer. If an order is shipped after the preferred due date, it is considered late and incurs a tardiness penalty proportional to the number of late periods. Tardiness penalty may include a discount offered to the customer, a penalty due to loss of goodwill, and the potential loss of future business. They provide the mathematical formulation of the problem defined. Increase in profit deriving from the leadtime flexibility is shown by the computational results.

Although Wester et al. (1992) study the OAS problem with setup times, other studies do not consider the setup times until Oğuz et al. (2010) who extend the OAS problem by including sequence dependent setup times and release times along with the deadlines (GOAS). After proposing a mathematical formulation for the GOAS problem and generating valid inequalities to strengthen its linear relaxation, Oğuz et al. (2010) develop a simulated annealing (SA) based heuristic and two constructive heuristics. Constructive heuristics produce high quality solutions -with 8% average deviation from the upper bound- for instances with 100–300 orders whereas the SA-based heuristic outperforms them in smaller instances.

Cesaret et al. (2012), Park et al. (2013), Lin and Ying (2013a), Chaurasia and Singh (2016), Nguyen (2016), Silva et al. (2018) and He et al. (2019) propose tabu search algorithm, genetic algorithm, artificial bee algorithm, hybrid evolutionary approach, hyper-heuristic using genetic programming, iterated local search and memetic algorithm, respectively, for the GOAS problem and each study improves the results of the previous one.

Cesaret et al. (2012) propose a tabu search (TS) algorithm for the GOAS problem. The vector encoding of solutions represents the acceptance and the sequencing decisions simultaneously. Neighborhood of the current solution is generated by

swapping two elements of the solution vector which enables to change both the set of accepted orders and their sequence. Due to the swapping move, an iteration of the TS algorithm may cause the number of accepted orders to decrease. If such a case is often observed, the algorithm is likely to converge to a poor local optimum. To overcome this obstacle, after each iteration, a probabilistic local search procedure is performed by applying iterative drop–add–insert operations. In comparison to the heuristics developed in Oğuz et al. (2010), the proposed TS algorithm gives significantly better solutions in all instances. Furthermore, run time of the TS algorithm is very small even for large instances.

Lin and Ying (2013b) develop an artificial bee colony (ABC) algorithm for the same problem. Unlike the solution encoding of Cesaret et al. (2012), they prefer a permutation representation which does not show the rejected jobs directly. However, following the calculation of the completion times of a particular solution representation, the orders that are completed after the deadline are not accepted. The destruction and construction phases of the iterated greedy algorithm (Ruiz and Stutzle, 2007) are applied to generate neighborhood solutions of the current solution. Following the generation of neighborhood solutions, the local search procedure improves them by iteratively exchanging orders in the solution with orders positioned after them. Computational results show that even for problem sizes involving up to 100 orders, the proposed ABC-based algorithm obtained good solutions with acceptable deviations from the upper bound.

Chaurasia and Singh (2016) develop two hybrid metaheuristic approaches, a hybrid steady-state genetic algorithm and a hybrid evolutionary algorithm with guided mutation, for GOAS problem. The proposed solution encoding is the same as the one in Lin and Ying (2013b). Although Chaurasia and Singh (2016) use a similar local search procedure like Cesaret et al. (2012), different solution representations make the solution methodologies in these studies differ significantly. The computational results show the superiority of the proposed approaches in terms of solution quality when compared to the TS algorithm proposed by Cesaret et al. (2012) and the ABC algorithm of Lin and Ying (2013b). However, in terms of computational times, the

TS algorithm is faster than the proposed approaches on most of the instances.

Although the hyper-heuristic using genetic programming proposed by Nguyen (2016) can give better solutions for some of the small size benchmark instances, iterated local search algorithm and memetic algorithm proposed by Silva et al. (2018) and He et al. (2019), respectively, outperform other algorithms in the literature for medium and large size instances. Iterated local search algorithm proposed by Silva et al. (2018) uses randomized variable neighborhood descent algorithm and double bridge mechanism in its local search and perturbation phase, respectively. In the local search phase, swap, block insertion and reverse block moves are used to generate new solutions. Swap move simply exchanges positions of two orders. While block insertion move shifts a block of orders to a new position, reverse block move reverses the sequence of the orders in a block. In the perturbation mechanism, double-bridge mechanism interchanges positions of two disjoint blocks randomly. The proposed iterated local search algorithm can find reasonable solutions for the instances with loose time windows. However, its performance deteriorates significantly when time windows get tighter.

He et al. (2019) propose a memetic algorithm that is a biased random key genetic algorithm using an adaptive large neighborhood search algorithm for local search. The relevant genetic algorithm uses a chromosome representation, having as many genes as the number of orders, such that each gene is encoded by a real value between 0 and 1. To decode a chromosome, all orders in it are sorted according to the ascending gene values that produces the sequence of the orders; following this sequence, orders are started to be processed as early as possible. Such an encoding scheme enables to search a large solution space efficiently while it is difficult to reach high quality solutions due to its problem independent structure. Therefore, He et al. (2019) use adaptive large neighborhood search algorithm, that employs problem specific insertion and removal moves, to overcome the corresponding drawback. Computational results show that the proposed algorithm can find competitive solutions in short time for large size instances.

Zandieh and Roumani (2017) study a similar problem to the GOAS problem by

excluding release times and propose biogeography-based optimization algorithm. Xie and Wang (2016) study the OAS problem with setup times between families, instead of sequence dependent setup times, and develop an enhanced ABC algorithm.

While the majority of the OAS literature is composed of heuristic algorithms, there is a limited number of studies developing exact algorithms. Slotnick and Morton (1996) propose a branch-and-bound algorithm that fixes the accepted solutions at each node of the search tree. A similar approach has been applied by Slotnick and Morton (2007), Nobibon and Leus (2011), Geramipour et al. (2017) and Cordone et al. (2017) for the variants of the OAS problem or prize collecting single machine scheduling which is akin to the OAS problem. Mestry et al. (2011) extend the OAS problem to the multi-resource environment that maximizes the total revenue of the accepted orders minus the resource consumption cost and develop a branch-and-price algorithm for it. Silva et al. (2018) propose the only exact algorithm for the GOAS problem which is an arc-time-indexed formulation based branch-and-price algorithm as in Mestry et al. (2011). Due to the high dimension of the corresponding formulation, it can only be solved for small size instances by off-the-shelf solvers and it cannot even find a feasible solution for large size instances. On the other hand, proposed branch and price algorithm can produce optimal / near optimal solutions, mostly due to tight linear relaxation of the arc-time-indexed formulation and pricing subproblem of the proposed algorithm that takes linear time, in the square of the number of orders multiplied by the length of planning horizon, to be solved.

2.2 Scheduling problems with batch delivery

The batch delivery brings forth the distinction between the completion time and the delivery time which is firstly proposed by Cheng and Kahlbacher (1993) who study a single machine scheduling problem with batch delivery to minimize the sum of the delivery cost and total weighted earliness cost; earliness of a job is defined as the difference between its delivery and completion time. They show that the problem is NP-complete and the equal weight case is polynomially solvable. Although there had been studies focusing on batch scheduling before Cheng and

Kahlbacher (1993), these studies examine 'batch production' in which similar orders / family of orders are processed consecutively to reduce the setup cost (Zdrzałka, 1996). However, Cheng and Kahlbacher (1993) are interested in 'batch delivery' that concerns the orders delivered together rather than processed successively. Since batch production is not in the scope of this thesis, only the studies similar to Cheng and Kahlbacher (1993) are reviewed in the following paragraphs of this section. Relevant studies are grouped with respect to the number of customers considered.

2.2.1 Single customer

Cheng and Gordon (1994) propose pseudopolynomial algorithms for a similar version of the problem defined by Cheng and Kahlbacher (1993) where the number of total batches is bounded and delivery cost is an increasing function of the total number of the batches. This problem is altered by Cheng et al. (1996) so that orders have arbitrary processing times depending on both the order and the batch to which it is assigned and the objective is the minimization of the delivery cost plus total tardiness. Cheng et al. (1996) show that this problem is polynomially solvable. Cheng et al. (1997) introduce the constant setup times before each batch for a similar problem with only job dependent processing times (i.e., processing times are independent of batches they are assigned) and the objection function including weighted earliness cost and the mean delivery time of the batches. They propose a pseudopolynomial algorithm along with a set of heuristics that are shown to obtain near optimal solutions. Ji et al. (2007) consider the weighted flow times instead of the earliness cost in the problem proposed by Cheng and Gordon (1994) and develop a pseudopolynomial dynamic programming (DP) algorithm under the same upper bound assumption on the number of total batches. Wang and Cheng (2000), for the first time, handle the parallel machine scheduling problem with batch delivery costs and propose a pseudopolynomial DP algorithm. They prove that two special cases, that is the optimal batching problem where the job assignment is predetermined and the original problem with identical processing times, are polynomially solvable.

The DP algorithms are prominent in most of the studies since they can efficiently

exploit the property that, for many scheduling and batching models with a single machine, the sequencing and batching decisions of the problem can be decoupled (Potts and Kovalyov, 2000). More precisely, analysis may show that there exists an optimal schedule in which the jobs appear in a certain sequence. Following this observation, Hall and Potts (2005) provide DP algorithms for a diverse class of scheduling problems with batch delivery. The objection function is defined as the minimization of delivery cost plus scheduling cost. They study both single and parallel machine problems while considering the total weighted completion time of the jobs, the maximum lateness of the jobs, the total weighted number of late jobs and the total tardiness of the jobs as the scheduling cost. Different types of batch delivery scheduling that are characterized by the number of machines and the objective function are defined. Either a DP algorithm is developed for each problem or it is shown that the problem is NP-complete. Ji et al. (2007) develop a DP problem for a different objective function, minimization of weighted flow time. Due to the curse of dimensionality, the proposed DP algorithms often cannot cope with the large size problems. Therefore, Mazdeh et al. (2011) propose a branch and bound algorithm for the weighted flow time problem and computational experiments show that the proposed algorithm outperforms the DP algorithm by Ji et al. (2007) as the number of orders increases.

Although it is often assumed that the due dates are not negotiable, there are also various studies considering the due dates as decision variables. Chen (1996) develop a polynomial algorithm for the scheduling with batch delivery problem where the decision maker assigns a common due date to all orders and the objective is to minimize the sum of the due date assignment cost, delivery cost, earliness and tardiness cost. Yin et al. (2013a) study a similar problem in which a common due window is assigned to all orders and earliness and tardiness of an order is computed by considering the start and the end time of the due window assigned to it, respectively. They propose a polynomial algorithm for the general case of the problem and lower the degree of the proposed algorithm for special cases of the problem. Yin et al. (2013b) study an extended version of the problem in Yin et al. (2013a) by

considering common due date assignment instead of due window and assuming resource dependent processing times. It is shown that the corresponding problem can be solved in polynomial time when the resource-processing time function is linear or convex and special cases of the problem can be solved by lower degree algorithms.

While all studies reviewed up to this point assume infinite number of uncapacitated vehicles for the delivery, studies considering the delivery with limited capacity also exist in the literature. Lee and Chen (2001) propose polynomially solvable DP algorithms or a heuristic based on the derived optimal solution properties for multiple-stage scheduling problems such that the transportation of the jobs between the stages requires constant delivery time and there are limited number of capacitated vehicles. Agnetis et al. (2014) address different versions of the two-stage scheduling problem with batch delivery where the time between the completion of a job in the first stage and the second stage is constrained and the objective is to minimize the total delivery cost. The relevant versions vary in the number of vehicles, the capacity of them and the flexibility of the delivery times (i.e., fixed or not fixed). For each version of the problem, Agnetis et al. (2014) either propose a polynomial (or pseudo-polynomial) solvable algorithm or show that the problem is strongly NP-Hard. Karimi and Davoudpour (2017) develop a Benders decomposition algorithm to solve a multi-stage scheduling problems where transportation of the jobs between stages are carried out by capacitated vehicles. The master problem of the corresponding algorithm assigns jobs to the batches as the subproblem determines the schedule of the orders in the batches and delivery of them. It is shown that the proposed algorithm can achieve small gaps for large size problems for which a branch and bound algorithm fails in finding a feasible solution in a limited time.

2.2.2 Multiple customers

In the majority of the scheduling literature with batch delivery, the number of customers is assumed to be one which can be deceiving since the characteristics of customers like their locations are determinant in the delivery cost of the batches. Therefore, Hall and Potts (2003) study a two-stage scheduling problem where the

supplier delivers the orders to multiple manufacturers in batches. They develop DP algorithms for different objective functions defined from the viewpoint of the supplier, the viewpoint of the manufacturer or a combined / mutual one. Different scheduling measures such as the number of late jobs, the maximum lateness and the sum of flow times are used in the corresponding objective functions along with the delivery costs. The benefits of the cooperation between the supplier and the manufacturers are shown by empirical results.

As aforementioned, the DP algorithms can be computationally intractable as the problem dimension enlarges even though they are polynomially solvable. Apart from this, developing polynomially solvable algorithms for the scheduling problems with batch delivery can be more difficult in presence of multiple customers. Consequently, heuristic algorithms, approximate algorithms or exact algorithms alternative to the dynamic programming are developed more often in the problems with multi-customer environment. For instance, the DP algorithms proposed for the problem from supplier viewpoint with the objective of minimizing the delivery cost plus the sum of flow times and weighted flow times, respectively, are exponential in the number of the customers (Hall and Potts, 2003). Therefore, Mazdeh et al. (2007) propose a branch and bound algorithm for the same problem and show that the proposed algorithm outperforms the DP algorithm of Hall and Potts (2003) for large size instances. Similarly, for the general version of this problem that considers weighted flow times, Selvarajah and Steiner (2009) provide an approximation algorithm and prove that optimal batching of the weighted shortest processing time job sequence yields a 3/2-approximation. Rasti-Barzoki and Hejazi (2013) extend the scheduling problem with capacitated batch delivery by including the setup time for each batch and assignable due dates. The objective is to minimize the cost of due date alterations, tardy jobs and tardy batches (i.e. a batch (not) including a tardy job is called (early) tardy batch). They propose a mathematical model and a branch and bound algorithm that can solve moderate size problems and propose a heuristic, for larger size problems, comprising of three stages: assignment of orders to early or tardy batches, assignment of positions to customers in early batches and

displacement of the jobs in early and tardy batches.

Chen and Vairaktarakis (2005) study batch delivery scheduling problem with multiple customers for both single and parallel machine cases. It is assumed that there are unlimited number of capacitated vehicles to deliver the orders and for each shipment, a fixed cost and a variable cost dependent on the particular route taken by the vehicle are incurred. The problem is to find a joint production and distribution schedule to minimize a function measuring customer service level and total distribution cost. Two particular forms of the customer service measure are considered separately: mean delivery time and maximum delivery time. For each subset of the entire customer set, the route with minimum cost that traverses the customers in the corresponding subset is found preliminarily. Hence, Chen and Vairaktarakis (2005) develop DP algorithms for the defined problems with given route information. Algorithms developed are polynomially solvable for all problems defined when the number of customers is not arbitrary except parallel machine scheduling problem with maximum delivery time. However, computational time to solve parallel machine problems is high in practice that leads Chen and Vairaktarakis (2005) to develop two heuristic algorithms; for parallel machine scheduling problems with mean delivery time and maximum delivery time, respectively. Developed heuristic algorithms comprise of two steps: to solve an auxiliary problem that divides a given list of jobs into delivery batches and to schedule the jobs in each batch. The auxiliary problem is solved by a DP algorithm while job sequence in each batch is simply determined by using the longest processing time rule. Computational results show that the optimality gap of the first heuristic is reasonable. Since no exact algorithm is developed for the parallel machine scheduling problem with maximum delivery time, the results of the second heuristic are evaluated based on the lower bound generated for the test instances and solutions of the heuristic achieve small gap with respect to the lower bound. Both heuristics have exponential time complexity. Computational results show that the solution quality of the heuristics is mostly influenced by the number of machines, and the effect of the number of customers on the solution time is limited in comparison to that of the number of machines .

While Chen and Vairaktarakis (2005) take routing of the uncapacitated delivery vehicles into account, Cakici et al. (2014) vary this problem by limiting the capacity of vehicles and excluding routing decisions. They study parallel machine scheduling with batch delivery and single vehicle. Each job has an associated weight representing its holding cost and its size. Total size of the jobs in any batch to be delivered must satisfy the capacity limitation of the vehicle. The objective function is to minimize the sum of total weighted delivery time of all jobs which is dependent on both the completion time of batches and the jobs in them so that delivery time of each batch is the sum of the time parameters associated with the orders in the corresponding batch. Cakici et al. (2014) give the mathematical formulation of the problem defined and propose a heuristic approach following three steps: batch assignment (i.e., building delivery batches for each customer), sequencing the delivery batches and job sequencing. Different approaches for the relevant assignment and sequencing steps are proposed. Ultimately, twenty heuristics that use different combination of these approaches are developed. It is shown that each heuristic has a worst performance ratio that is less than 2 considering all problem instances. Moreover, they provide near optimal solutions for most of the problem instances. Gong et al. (2016) study a similar problem to Cakici et al. (2014) while they assume that there is a single customer. Gong et al. (2016) provide an exact algorithm, a polynomial time algorithm based on dynamic programming, for a special case with a given job assignment to the machines. For the general case, they develop a heuristic algorithm that is inspired from the heuristic developed by Chen and Vairaktarakis (2005). In this problem, there exists an optimal schedule such that jobs are scheduled in the shortest processing time order on each of the parallel machines. Thus, this problem can be reduced to partitioning jobs into subsets to be assigned to one of the machines. Gong et al. (2016) develop a fully polynomial-time approximation scheme that is based on a dynamic programming algorithm determining the possible departure times of jobs and batches for delivery.

A heuristic algorithm nesting a DP algorithm is developed by Pundoor and Chen (2005) who address a single machine scheduling problem with batch delivery where

there are multiple customers and delivery, of which size is restricted by a capacity limit, to each customer requires a constant delivery time, and incurs a fixed delivery cost. The proposed heuristic generates, for each customer, the optimal schedule for each possible number of total batches by using a DP algorithm. After determining the total number of batches to be delivered, the corresponding optimal schedules are sequenced based on the optimal solution properties derived.

2.3 Order acceptance and scheduling problems with batch delivery

Hamidinia et al. (2012) model a single machine batching delivery problem with multiple customers and develop a genetic algorithm for it. Batches are not allowed to include the orders of more than one customer and objective is to minimize the earliness, tardiness, holding and delivery cost. Since the model formulated cannot be solved by exact solvers for even smaller cases and no lower bounds are provided by them, the performance of the proposed algorithm is not evaluated properly. Ahmadizar and Farhadi (2015) extends this by taking into consideration job release dates and due windows. Hamidinia et al. (2012) claim that this problem cannot be solved by off-the-shelf optimizers even for small test instances. Although they have argued that this failure is due to the problem complexity, Ahmadizar and Farhadi (2015) show that their erroneous formulation is the main reason and present an appropriate mathematical model. They establish a set of dominance properties and propose a solution method by incorporating them within an imperialist competitive algorithm which is a population-based algorithm. The derived dominance properties are employed as a local search strategy enhancing the quality of a solution. Proposed algorithm is capable of finding near optimal solutions even for large instances. However, comparison analysis shows that success of the proposed algorithm is mostly by means of the dominance rules.

Khalili et al. (2016) extend the problem handled by Hamidinia et al. (2012) to an order acceptance scheduling problem with setup time, setup cost and batch delivery. This is the first study that integrates the OAS problem with the batch delivery. Objective function is to maximize revenues gained from the accepted orders minus

setup, holding and delivery cost. Khalili et al. (2016) use imperialist competitive algorithm like Ahmadizar and Farhadi (2015); however, unlike them, the proposed algorithm is straightforward that is not using any insights (i.e., dominance rules) related to the problem. Consequently, the gap between the best solution found by CPLEX solver in 3600s and found by the proposed algorithm reaches up to more than 150%.

Jiang et al. (2017) address the order acceptance and scheduling problem in a parallel machine environment with capacitated batch delivery and propose two approximating algorithms that minimize the weighted sum of the maximum lead time of the accepted orders, rejection cost of the orders and the delivery cost. Rasti-Barzoki et al. (2017) assume that there is limited number of capacitated vehicles and each order occupies different space in the vehicles. Regarding the corresponding conditions, they propose a mathematical model and a hybrid heuristic of genetic algorithm and particle swarm optimization. Noroozi et al. (2018) extend this problem by adding the third party logistics distribution and propose a genetic algorithm along with two mathematical models.

Chapter 3

NEW MATHEMATICAL MODELS AND A MATHEURISTIC FOR THE GENERALIZED ORDER ACCEPTANCE AND SCHEDULING PROBLEM

There are two mathematical models, both of which are mixed integer linear programming (MILP) models, for the GOAS problem in the literature. The first model is developed by Oğuz et al. (2010) that take benefit of Miller-Tucker-Zemlin constraints (Miller et al., 1960) to avoid subtours. Although the proposed model is amenable to be adapted to the variants of the GOAS problem (i.e., it is flexible), it gives a poor performance due to the loose upper bounds it generates. This performance is due to the Miller-Tucker-Zemlin constraints which are known not to provide tight bounds in comparison with alternative subtour elimination constraints (Öncan et al., 2009). On the other hand, Silva et al. (2018) propose an arc-time indexed model formulation for the GOAS problem which can generate tight upper bounds at the root node, at the expense of increasing number of decision variables and constraints. Since the proposed model can be solved for only modest size problems by the commercial solvers, Silva et al. (2018) develop a branch and price algorithm to tackle their extended model formulation. However, this algorithm cannot be easily adapted to the variants of the GOAS problem, mainly due to the special structure of its pricing problem. Thus, there is a lack of a model that combines good aspects of these two MILP models: flexible and strong.

In this chapter, we aim to develop an effective and efficient solution method for the GOAS problem:

1. to push the boundary of optimally solved instance size further, and

2. to obtain near optimal solutions in reasonable time for instances falling beyond

this boundary,

given its NP-hard status. To this end, we develop a new MILP and a constraint programming (CP) model, and then propose a matheuristic, which capitalizes on the strengths of the MILP model and two metaheuristics, for this complex GOAS problem. Both of the models developed can easily be modified for the extensions of the GOAS problem unlike the arc-time-indexed formulation by Silva et al. (2018) and they outperform the MILP model developed by Oğuz et al. (2010). However, they fail at solving large sized problem instances. Therefore, we exploit the ability of the new MILP model in solving this difficult and complex optimization problem and combine its strengths with two metaheuristics to propose an iterative two-phase matheuristic including a local search to improve the solution found at the end of the second phase (GOAS_I2PM).

The essence of GOAS_I2PM is decomposing the planning horizon into smaller time segments. Fragkos et al. (2016) propose a similar decomposition of the planning horizon and develop a branch-and-price algorithm such that pricing problem corresponds to solving a scheduling problem for each time segment independently. This horizon decomposition is different from the rolling horizon approach (Stauffer and Liebling, 1997) since it does not take into account the following time segments while considering a particular time segment. Elkamel et al. (1997) propose a temporal decomposition heuristic that divides the planning horizon into time segments, makes job assignments to them for once and solve the segments sequentially.

Unlike these decompositions, GOAS_I2PM, being iterative in nature, assigns orders to the time segments and solves the scheduling problem for these segments sequentially to obtain a feasible schedule in each iteration. For the first phase of GOAS_I2PM, we develop a relaxed time-bucket formulation to make the assignment of orders to the time segments while ensuring a new assignment of orders to the time segments in each iteration. In the second phase of GOAS_I2PM, the orders assigned to each segment in the first phase are scheduled via the MILP model sequentially. As the complete solution evolves from this sequential scheduling of orders starting from the first time segment, the partial solutions are improved with a variable neigh-

borhood search (VNS) algorithm. After the second phase, a local search is applied to the final solution which is a multi-start tabu search algorithm. GOAS_I2PM iteratively modifies the processing times used in the first phase based on the final solution obtained during this process and stops when the given time limit is exceeded. Performance of GOAS_I2PM is compared with both the proposed MILP and CP models and the state-of-the-art algorithms (Silva et al. (2018) and He et al. (2019)). It is shown that GOAS_I2PM outperforms both proposed mathematical models and the algorithm by Silva et al. (2018) in terms of both solution quality and run time. While GOAS_I2PM is superior to the algorithm by He et al. (2019) with respect to the solution quality, its run time is higher.

The remainder of this chapter is organized as follows. Section 3.1 presents new MILP and CP formulations while Section 3.2 describes the matheuristic algorithm, GOAS_I2PM. Computational results are given in Section 3.3 and a discussion of this chapter is provided in Section 3.4.

3.1 Two mathematical models

3.1.1 Mixed integer linear programming model

We develop a new precedence based model formulation for the GOAS problem motivated by the traveling salesman problem formulation developed by Gavish and Graves (1978). We consider this model formulation to include the precedence relations by using both binary and continuous variables which enables us to handle release time and deadline restrictions of the orders effectively.

In this formulation, two dummy orders, namely order 0 and order $n + 1$, are defined to be assigned to the first and the last positions of the schedule, respectively, and O' as a set that includes both set O and dummy orders. The model uses the following variables:

l_i: 1 if order i is accepted, 0 otherwise; $i \in O'$.

$y_{i,j}$: 1 if order j is processed immediately after order i, 0 otherwise; $i \in$

$O' \setminus \{n+1\}, j \in O' \setminus \{0\}$.

$g_{i,j}$: Completion time of order j if order i immediately precedes it, 0 otherwise; $i \in O' \setminus \{n+1\}, j \in O' \setminus \{0\}$.

$T_{i,j}$: Tardiness of order j if order i immediately precedes it, 0 otherwise; $i \in O' \setminus \{n+1\}, j \in O' \setminus \{0\}$.

Then the mixed integer linear programming (MILP) model is given as follows:

$$\text{Maximize} \quad \sum_{i \in O} e_i l_i - \sum_{i,j \in O'} T_{i,j} w_j \tag{3.1}$$

s.t.

$$\sum_{j=1, j \neq i}^{n+1} y_{i,j} = l_i \qquad \forall i \in O' \setminus \{n+1\} \tag{3.2}$$

$$\sum_{j=0, j \neq i}^{n} y_{j,i} = l_i \qquad \forall i \in O' \setminus \{0\} \tag{3.3}$$

$$\sum_{j=1}^{n+1} g_{i,j} - \sum_{j=0}^{n} g_{j,i} \geq \sum_{j=1}^{n+1} (s_{i,j} + p_j) y_{i,j} + \sum_{j=1, \bar{d}_i < r_j}^{n+1} (r_j - \bar{d}_i) y_{i,j} \qquad \forall i \in O \tag{3.4}$$

$$g_{i,j} \geq g_{i,j}^{lb} y_{i,j} \qquad \forall i \in O' \setminus \{n+1\}, \forall j \in O' \setminus \{0\} \tag{3.5}$$

$$g_{i,j} \leq g_{i,j}^{ub} y_{i,j} \qquad \forall i \in O' \setminus \{n+1\}, \forall j \in O' \setminus \{0\} \tag{3.6}$$

$$T_{i,j} \geq g_{i,j} - d_j y_{i,j} \qquad \forall i \in O' \setminus \{n+1\}, \forall j \in O' \setminus \{0\} \tag{3.7}$$

$$T_{i,j}, g_{i,j} \in R^+, \; l_i, y_{i,j} \in \{0,1\} \qquad \forall i \in O', \forall j \in O' \tag{3.8}$$

Objective function (3.1) is the net revenue that is equal to the gross revenue minus the tardiness cost. Constraints (3.2) and (3.3) ensure that each accepted order is preceded and succeeded by only one job, respectively. Constraint (3.4) represents the subtour elimination constraints. Constraints (3.5) and (3.6) avoid an order being processed before its release time and after its deadline by using

parameters $g^{lb}_{i,j}$ and $g^{ub}_{i,j}$ that are lower and upper bounds, respectively, on variable $g_{i,j}$. They are computed as follows.

$$g^{lb}_{i,j} = \max\{r_i + \min_{k:k\leq n, k\neq i,j} s_{k,i} + p_i + s_{i,j} + p_j, r_j + s_{i,j} + p_j\} \qquad \forall i,j \in O'$$

$$g^{ub}_{i,j} = \min\{\bar{d}_j, \max\{\bar{d}_i, r_j\} + s_{i,j} + p_j\} \qquad \forall i,j \in O'$$

As multiplying the $g^{lb}_{i,j}$ by $y_{i,j}$ in Constraint (3.5), it is guaranteed that $g_{i,j}$ cannot not be smaller than $r_j + s_{i,j} + p_j$, that is the completion time of order j if order i immediately precedes it and order j starts being processed at its release time, if $y_{i,j}$ is equal to 1. Similarly, $g_{i,j}$ cannot be bigger than the deadline of order j if $y_{i,j}$=1 by Constraint (3.6) and it must be equal to 0 if $y_{i,j}$=0. Constraint (3.7) sets the tardiness of the orders while Constraint (3.8) defines the bounds and integrality of the variables.

3.1.2 Constraint programming model

As Baptiste et al. (2012) state, constraint programming has become a paradigm that scheduling practitioners cannot afford to ignore. Moreover, Pesant et al. (1998) and Focacci et al. (2002) show that it is highly competitive when time windows are present. Therefore, the GOAS problem can be modelled by constraint programming easily due to mainly its two traits: *i*) the notion of 'activity', and *ii*) available specialized constraints which are the key elements of the constraint programming models to be successful (Kanet et al., 2004).

In CP, an 'activity' can be defined as a job that has a size and may have restrictions on its execution time. In our problem, order i can be defined as an activity with size p_i and the available period for its execution can be set by using r_i and $\bar{d}_i$, $\forall i \in O$. Once we define each order as an activity, constraint programming handles the release time and the deadline without any further constraints like constraints (3.5) and (3.6) in the MILP model.

Specialized constraints provide a similar advantage that allow complex constraints to be formulated easily while special algorithms developed particularly for these specialized constraints enable faster searches. 'NoOverlap' is one of those con-

straints that avoids a sequence of activities with time windows to overlap. Thus, the GOAS problem can be modelled by constraint programming as follows with the additional notation below:

A_i: An optional interval decision variable for order $i \in O$ where $[r_i, \bar{d}_i]$ is the range of the interval, p_i is the size of activity i and 'optional' means that it is not compulsory to process activity i, $\forall i \in O$.

seq: Sequence of the accepted activities.

Then, the CP model can be formulated as follows:

$$\text{Maximize} \quad (\sum_{i \in O} presenceOf(A_i)e_i - \max(0, endOf(A_i) - d_i)w_i) \tag{3.9}$$

s.t.

$$noOverlap(seq, s) \tag{3.10}$$

$$startOf(A_i) \geq presenceOf(A_i) * (r_i + s_{prev(seq,A_i),i}) \quad \forall i \in O \tag{3.11}$$

$presenceOf(A_i)$ returns true if activity i is performed, false otherwise. $endOf(A_i)$ returns the finish time of activity i and hence the objective function (3.9) is equal to the difference between the revenue and the tardiness cost. Constraint (3.10), by using $noOverlap(seq, s)$ function, ensures that considering the processing time and the setup time between adjacent activities in the sequence *seq*, none of those activities overlaps. $startOf(A_i)$ and $prev(seq, A_i)$ return the start time and the preceding activity of activity i, respectively. Constraint (3.11) guarantees that if activity i is executed, its start time is not less than its release time plus the setup time incurred just before it.

3.2 Matheuristic algorithm

The GOAS problem is NP-hard and the computational results in Section 3.3 show that the performance of the commercial solvers in solving the proposed MILP and

CP models significantly deteriorates as the number of orders increases. In order to achieve our goal in this study, we propose an iterative two-phase matheuristic algorithm utilizing the proposed time-bucket formulation and MILP model together with two metaheuristics.

The main idea of the proposed matheuristic algorithm GOAS_I2PM is to decompose the original problem into subproblems having manageable number of orders so as to exploit the good performance of the commercial solvers on moderate size problems. GOAS_I2PM is composed of two phases, ASSIGN and SCHEDULE, that are communicated iteratively. The original planning horizon T, that is equal to $\max_{i:i\in O} \bar{d}_i$, is decomposed into a set of segments, S, with a predefined cardinality and ASSIGN uses a relaxed time-bucket formulation for the GOAS problem without setup times to assign orders to those time segments; unaccepted orders are assigned to a dummy time segment. Given the assignments of orders to the time segments, SCHEDULE, starting from the first segment, solves the GOAS problem of each segment sequentially and generates a feasible solution to the original problem. During this process a VNS algorithm is used to improve partial solutions.

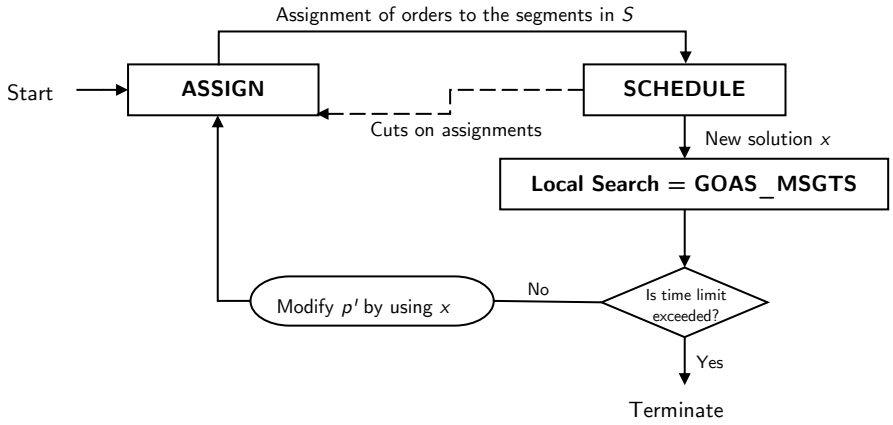

Figure 3.1: Flow chart of the proposed matheuristic GOAS_I2PM.

Once the second phase is finished, the complete solution obtained is improved by a local search in the form of a TS algorithm which also uses VNS for its local search. After this improvement step, the processing times are updated based on the final solution obtained during this process and fed into ASSIGN to generate a new

assignment of orders to the time segments to be used by SCHEDULE in an iterative manner until the termination condition is met. This process is depicted as a flow chart in Figure 3.1.

The details of two phases together with the mathematical models used in ASSIGN and SCHEDULE are presented in Section 3.2.1 and 3.2.2, respectively. The proposed TS algorithm is presented in Section 3.2.3 and the interaction between two phases, namely ASSIGN and SCHEDULE, is explained in Section 3.2.4. Implementation of GOAS_I2PM on a small example can be found in Appendix A.

3.2.1 Phase 1 of GOAS_I2PM: ASSIGN

Linear relaxation of the time-indexed formulations for scheduling problems generates tight bounds (Van Den Akker et al., 2000). However, they are not amenable to be called multiple times in an iterative manner because of their high run time. Therefore, we propose a relaxed time-bucket model (GOAS_RTBM) for the GOAS problem. Time-bucket formulations can yield tight lower bounds and high quality feasible or easily repairable solutions while requiring much shorter run times compared to the time-indexed formulations (Riedler et al., 2020). Wang and Regan (2009) show that time-buckets can be modified so as to ensure its convergence to the optimal solution. Boland et al. (2013) and Boland et al. (2016) develop time-bucket formulations that guarantee the optimal solution for all min-sum scheduling criteria. However, these formulations still have the run time obstacle to be called multiple times over the course of a heuristic. Therefore, instead of using buckets for which the length can be as large as the processing time of the shortest job as in Boland et al. (2013), we use the largest processing times as the length of time buckets in the GOAS_RTBM model to decrease the run time further. Preliminary runs show that it produces solutions with competitive quality when compared to the time-indexed model within much less run time.

The proposed time-bucket model is based on a time index set $T' = \{t_0, t_1, t_2, ..t_{|T'|}\}$. Intervals between adjacent two indices are denoted by index k such that the kth interval starts from $t_{k-1} + 1$ and ends at t_k where $k \in K = \{1, 2, .., |T'|\}$. We generate

the time indices such that an order that is started at the kth interval must be completed in the $k+1$st interval at the latest for any $k \in K$. Following parameters are used in generating the time index set T' in Algorithm 3.1 and in developing the GOAS_RTBM model:

$$minsetup_i = \min_{j \in O' \setminus \{i, n+1\}} s_{j,i} \qquad \forall i \in O$$

$$p'_i = p_i + minsetup_i \qquad \forall i \in O$$

$$minend_i = r_i + p'_i \qquad \forall i \in O$$

Algorithm 3.1 Generation of time index set T'

Initialize: $t_0 = 0, l = 1, t_l : \min_{i \in O} r_i + s_{0,i} + p_i$

while true **do**

$Next\_time\_index : \max_{i \in O}\{\min\{t_l + \max_{j \in O: r_i \leq t_l, \bar{d}_i \geq t_l, minend_j \leq t_l} p_i + s_{j,i}, \bar{d}_i\}\}$

if $Next\_time\_index \geq T$ **then**

$t_{l+1} = T, BREAK$

else

$t_{l+1} = Next\_time\_index, l++$

end if

end while

Output: $K = l + 1, T' = \{t_1, t_2, ..t_{|K|}\}$

As aforementioned, given the generation scheme of T', for an order that has started in interval k, there are two options: *i*) either it is completed in the same interval, or *ii*) it is completed in interval $k+1$ as shown in Figure 3.2.

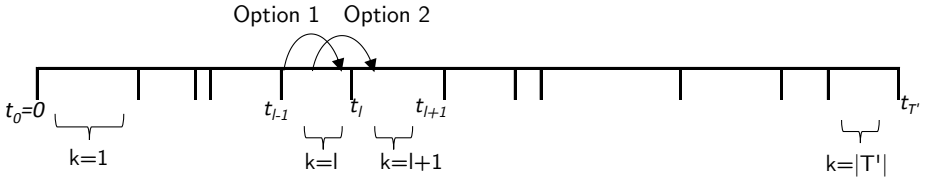

Figure 3.2: Planning horizon T with time indices.

We use the following decision variables to encompass these options within the GOAS_RTBM model:

$u_{i,k}$: 1 if order i has started and completed in interval k, 0 otherwise, $\forall i \in O, k \in K$.

$u'_{i,k}$: 1 if order i has started in interval $k-1$ and completed in interval k, 0 otherwise, $\forall i \in O, k \in K \setminus \{0\}$.

$q_{i,k}$: Amount of order i processed in interval k if $u'_{i,k} = 1$, 0 otherwise, $\forall i \in O, k \in K \setminus \{0\}$.

T_i: Tardiness of order i, $\quad \forall i \in O$.

In the preprocessing phase of the GOAS_RTBM model, the release time and the deadline of the orders are considered and if $u_{i,k}$ or $u'_{i,k}$ cannot be equal to one due to the release time and the deadline of order i, then the corresponding variable is set to zero. Initial version of the GOAS_RTBM model is as follows:

$$\text{Maximize} \quad \sum_{i \in O, k \in K} (u_{i,k} + u'_{i,k}) e_i - \sum_{i \in O} T_i w_i \tag{3.12}$$

s.t.

$$\sum_{i \in O, k \in K} ((u_{i,k} + u'_{i,k+1}) p'_i + q_{i,k} - q_{i,k+1}) \leq t_k - t_{k-1} \qquad \forall k \in K \tag{3.13}$$

$$\sum_{k \in K} u_{i,k} + u'_{i,k} \leq 1 \qquad \forall i \in O \tag{3.14}$$

$$\sum_{i \in O} u'_{i,k} \leq 1 \qquad \forall k \in K \tag{3.15}$$

$$\sum_{i \in O} q_{i,k} \leq u'_{i,k} p'_i \qquad \forall k \in K \tag{3.16}$$

$$T_i \geq \sum_{k \in K} (u'_{i,k}(t_{k-1} - d_i) + q_{i,k} + u_{i,k} max\{max\{t_{k-1}, r_i\} + p'_i - d_i, 0\}) \qquad \forall i \in O \tag{3.17}$$

$$u_{i,k}, u'_{i,k} \in \{0,1\}, q_{i,k}, T_i \in R^+ \qquad \forall i \in O, k \in K \tag{3.18}$$

Objective function (3.12) is the maximization of net revenue. Constraint (3.13) guarantees that the sum of processing times assigned to the intervals is not greater than the length of the corresponding interval. Each order can be processed at most once by Constraint (3.14). There cannot be more than one order that is in process at the beginning of any time interval by Constraint (3.15) and if there is such an order, its processing time beyond the start time of the corresponding time interval cannot be greater than its processing time by Constraint (3.16). Tardiness of each order is computed by Constraint (3.17). Integrality and bounds on the decision variables are defined by Constraint (3.18).

As aforementioned, the original planning horizon T is decomposed into a set of segments S. The end points of the segments, $t_{s_1}, t_{s_2}, .., t_{s_{|S|}}$, in set S is chosen from the set of time indeces T' such that the lengths of the segments, $t_{s_2} - t_{s_1}, .., t_{s_{|S|}} - t_{s_{|S|-1}}$ do not vary too much. Time segments are denoted by index s such that the sth segment spans the time period from $t_{s_{s-1}}$ to t_{s_s} where $s \in S = \{1, 2, .., |S|\}$. Following the definition of set S, the planning horizon depicted in Figure 3.2 can be altered as shown in Figure 3.3.

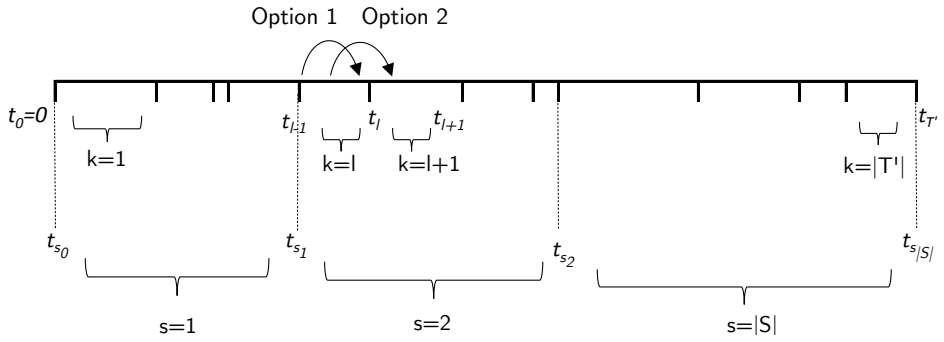

Figure 3.3: Decomposing the planning horizon T into time segments.

To assign the orders to one of the segments in S, a new binary variable $a_{i,s}$ is introduced as the integrality constraint (3.18) is relaxed. $a_{i,s}$ would be equal to 1 if the completion time of order i lies in time segment s, and 0 otherwise, $\forall i \in O, s \in S$. We add a dummy segment $s = 0$ to set S so that $a_{i,0}$ would be equal to 1 if order i is not accepted, $\forall i \in O$. Then, the final version of the GOAS_RTBM model is as

follows:

Maximize: Objective (3.12)

s.t.

Constraints (3.13), (3.15), (3.16), (3.17)

$$\sum_{k \in K: t_{s_{s-1}} < t_k <= t_{s_s}} (u_{i,k} + u'_{i,k}) = a_{i,s} \qquad \forall i \in O, s \in S \backslash \{0\} \tag{3.19}$$

$$\sum_{s \in S} a_{i,s} = 1 \qquad \forall i \in O \tag{3.20}$$

Cut set (3.21)

$$u_{i,k}, u'_{i,k} \in [0,1], a_{i,s} \in \{0,1\} \qquad \forall i \in O, k \in K, s \in S \tag{3.22}$$

Constraint (3.19) links u and a variables by defining each time segment as a set of adjacent time intervals. Constraint (3.20) ensures that an order is assigned to exactly one time segment. Constraint (3.14) is not included in the final version of the GOAS_RTBM model since it is implied by Constraints (3.19) and (3.20). Constraint (3.21) includes the cuts generated by the second phase of the matheuristic, SCHEDULE, while Constraint (3.22) includes the integrality and the bound restrictions. Overall, the steps taken in the first phase of GOAS_I2PM is presented in Algorithm 3.2.

Algorithm 3.2 ASSIGN

Input: Model GOAS_RTBM, $p'_i, i \in O$ (processing times modified after the local search), $\pi_s, s \in S$ (schedules generated by SCHEDULE)

if First iteration of the GOAS_RTBM **then**

Solve model GOAS_RTBM (p,.)

else

Solve model GOAS_RTBM (p', π) which uses the modified processing times and cuts derived by using the schedules of the time segments

end if

Output: $a_{i,s}$, $\forall i \in O, s \in S$

Based on the output of Algorithm 3.2, i.e., $a_{i,s}$, $\forall i \in O, s \in S$, we define the

following sets to be used in the second phase of GOAS_I2PM:

O_0 : Set of orders not accepted at the end of phase 1 after running ASSIGN, i.e., orders which are not assigned to any of the time segments, $O_0 = O \setminus \bigcup_{k=1}^{|S|} O_k$.

O_s : Set of orders assigned to time segment $s \in S$ by ASSIGN at the end of phase 1.

3.2.2 Phase 2 of GOAS_I2PM: SCHEDULE

At each iteration of GOAS_I2PM, the GOAS_RTBM model produces a new assignment of orders to time segments (i.e., a different value for variable a) in the first phase, which becomes an input to the second phase, that is SCHEDULE. Since SCHEDULE handles each time segment separately, it solves $|S|$ subproblems sequentially by using the MILP model.

As SCHEDULE solves the MILP model for a time segment $s \in S$, only the set O_s is considered. Since each time segment $s \in S$ has an associated beginning point and end point, release times and deadlines of the orders in set O_s must be in accordance with the corresponding points (i.e., $t_{s_{s-1}}$ and t_{s_s}). Let r_i^s and $\bar{d}_i^s$ denote the modified release time and the modified deadline of order i, respectively, to be used in the subproblem regarding time segment s, which are computed as follows:

$$r_i^s = \max\{t_{s_{s-1}}, r_i\} \qquad \forall i \in O, s \in S$$

$$\bar{d}_i^s = \min\{t_{s_s}, \bar{d}_i\} \qquad \forall i \in O, s \in S$$

$MILP(O_s, r^s, \bar{d}^s)$ denotes the MILP model considering the orders in set O_s with their modified r and $\bar{d}$ parameters. SCHEDULE runs $MILP(O_s, r^s, \bar{d}^s)$ for each time segment $s \in S$ sequentially and generates π_s which is the schedule of the accepted orders in O_s, $s \in S$. Although the length of the time segments are determined so that each subproblem can be solved quickly, it is still possible to have a subproblem requiring high computational time due to a large number of orders assigned to that time segment in the first phase of the matheuristic. To avoid such computational burdens, each run of SCHEDULE is limited to one second. Therefore, the subproblems may not be necessarily solved to optimality.

In order to differentiate the schedule created at each time segment, π_s, and the partial solutions constructed at the end of each time segment, we define x_s as the partial solution obtained at the end of time segment $s \in S$ in phase 2 so that it includes the schedule of the accepted orders in $\bigcup_{k=0}^{s} O_k$. Accordingly:

$x_1 = \pi_1$, and

$x_s = x_{s-1} + \pi_s$, $\forall s \in \{2, .., |S|\}$, where the operator $+$ is used to denote the concatenation of two (partial) schedules.

For the sake of convenience, we also define the set X_s as follows:

$X_s = \{i|$ order i belongs to partial solution $x_s\}$.

After SCHEDULE generates π_s at time segment $s \in S$, we apply a VNS algorithm to improve the partial solutions constructed throughout the second phase of GOAS_I2PM. We form the the initial solution of VNS by using x_s, which is explained in the next section.

Algorithm 3.3 SCHEDULE

Input: $O_s, r^s, \bar{d}^s, s \in S$

Initialize: $s = 1, x_0 = \emptyset$

while $s \in S$ **do**

  $\pi_s \leftarrow MILP(O_s, r^s, \bar{d}^s)$

  if $s < |S|$ **then**

    Create x'_s by adding rejected orders so far to the partial solution x_s

    $x_s \leftarrow VNS(x'_s)$

  else

    $x_s \leftarrow x_{s-1} + \pi_s$

  end if

  $s++$

end while

Output: $x_{|S|}$

Let $VNS(x_s)$ denote the execution of the VNS algorithm on solution x_s. Then, SCHEDULE can be interpreted as a process of solving the $MILP(O_s, r^s, \bar{d}^s)$ model and running the VNS algorithm recursively as shown in Algorithm 3.3.

Variable Neighborhood Search (VNS) algorithm

Since partial solution x_s include all accepted orders up to and including time segment $s \in S$, we want to exploit the benefits of accepting any rejected orders so far. Hence we run a VNS at the end of each time segment $s \in S$ by using partial solution x_s and all the rejected orders so far to form the initial solution of VNS. Let x'_s denote the initial solution to be improved by VNS, which is defined as:

$x'_s = x_s + [i|i \in U_s]$, where

$U_s = \{i|i \in \bigcup_{k=0}^{s} O_k \wedge i \notin X_s\}$.

As described above, when SCHEDULE generates π_s at time segment s, the matheuristic GOAS_I2PM will call VNS with the initial solution x'_s obtained from the partial solution x_s. We note that the set U_s denote the orders which are not accepted so far. When $s = 1$, the set U_s is identical to set O_0 which includes the orders that are not accepted by ASSIGN. For other segments $s \in S \setminus \{1\}$, U_s includes the orders that are in $\bigcup_{k=0}^{s} O_k$ but are not involved in the partial solution x_s, which is formed according to π_s after $MILP(O_s, r^s, \bar{d}^s)$ is run. The initial solution x'_s is encoded as the permutation of orders to represent the processing sequence of orders accepted and the sequence of orders rejected.

Using the VNS algorithm at the end of each phase 2 subproblem has three purposes:

i) SCHEDULE handles each time segment separately and the schedule created by it can be improved with a local search including the time segments preceding the current time segment,

ii) the schedule π_s produced by $MILP(O_s, r^s, \bar{d}^s)$ can be improved since it may not be solved to the optimality, and

iii) some of the orders assigned to a particular time segment by ASSIGN may not be accepted when $MILP(O_s, r^s, \bar{d}^s)$ is run for the corresponding subproblem in SCHEDULE, and these orders can be reconsidered in the following time segments through the VNS algorithm.

Another distinctive part of the proposed matheuristic is to consider a secondary criterion when evaluating the partial solutions at the second phase. Hence, solutions generated by the VNS algorithm are evaluated with respect to two criteria: the objective function value and the makespan. For any time segment $s \in S$, the makespan of the partial solution x_s should not exceed t_{s_s}; accordingly, the solutions for which the makespan exceeds the end of the corresponding time segment are ignored. It is also vital to be able to decrease the makespan of the partial solution x_s since it leads to having a longer time horizon for the succeeding time segments. Thus, among two solutions with equal objective function value, the one with the shorter makespan is preferred over the other one.

Algorithm 3.4 Variable neighborhood search algorithm

Input: x (initial solution), K (set of neighborhoods)

Initialize: k=1

$f(x)$: Objective value of solution x

$C_{max}(x)$: Makespan of solution x

repeat

$x' \leftarrow argmin_{y \in N_k(x)} f(y)$

if $f(x') > f(x)$ or $(f(x') = f(x)$ and $C_{max}(x') < C_{max}(x))$ **then**

$x \leftarrow x', k = 1$

else

$k++$

end if

until $k > |K|$

Output: x

Kirlik and Oğuz (2012) show that for minimizing total weighted tardiness with sequence dependent setup times of jobs on a single machine, VNS employing swap of jobs, job insertion and 2-successive job insertion moves performs better than VNSs that employ other commonly used moves in the literature. We adapt a similar version of the corresponding VNS, which is illustrated in Algorithm 3.4.

In Algorithm 3.4, $N_k(x)$ is the set of all feasible solutions in the neighborhood $k, k = 1, 2, 3$, of solution x. $N_1(x)$ defines a swap move that exchanges location of an accepted and an unaccepted order in solution x. $N_2(x)$ corresponds to an insertion move that picks a single order and inserts it into a different location in solution x, while $N_3(x)$ designates an insertion move that picks two adjacent accepted orders of solution x and inserts them into a different location. Algorithm 3.5 and Algorithm 3.6 show the swap ($N_1(x)$) and insertion ($N_2(x)$ and $N_3(x)$) moves, respectively.

Algorithm 3.5 Swap move

Input: x (current solution)
Initialize: $x^* \leftarrow x$
$Swap(x, i, j)$: Swap orders $x[i]$ and $x[j]$
for $i : 1; i \leq \bar{k} - 1; i++$ **do**
  for $j : \bar{k}; j \leq n; j++$ **do**
    $x' \leftarrow Swap(x, i, j)$
    if $f(x') > f(x^*)$ or ($f(x') = f(x^*)$ and $C_{max}(x') < C_{max}(x^*)$) **then**
      $x^* \leftarrow x'$
    end if
  end for
end for
Output: x^*

Algorithm 3.6 Insertion moves: $Insertion^1$ and $Insertion^2$

Input: x (current solution)

Initialize: $x^* \leftarrow x$

$Insertion^1(x, i, j)$: Shift order $x[i]$ to the jth position

$Insertion^2(x, i, j)$: Shift orders $x[i]$ and $x[i+1]$ to the jth and the $(j+1)$st position, respectively

$Insertion(x, i, j)$ corresponds to $Insertion^1(x, i, j)$ and $Insertion^2(x, i, j)$ for the second and the third neighborhood, respectively

for $i : 1; i \leq n; i++$ **do**
  for $j : 1; j \leq \bar{k}; j++$ **do**
    $x' \leftarrow Insertion(x, i, j)$
    if $f(x') > f(x^*)$ or $(f(x') = f(x^*)$ and $C_{max}(x') < C_{max}(x^*))$ **then**
      $x^* \leftarrow x'$
    end if
  end for
end for

Output: x^*

Regarding two insertion moves, we incorporate a mechanism to restrict the computational time as follows: Let $x[i]$ denote the ith element of solution (schedule) x to define an order at position i in schedule x. Now let $x[\bar{k}]$ be the first order in solution x for which the completion time exceeds its deadline, i.e., $C_{x[i]} \leq \bar{d}_{x[i]}$ for $\forall i < \bar{k}$ and $C_{x[\bar{k}]} > \bar{d}_{x[\bar{k}]}$ where $C_{x[i]}$ denotes the completion time of order $x[i]$. In this notation, orders $x[1], .., x[\bar{k}-1]$ are the accepted orders. Then, in the insertion moves, orders are not inserted beyond the $\bar{k}$th location since it is unlikely to observe any improvement by such moves.

3.2.3 A tabu search algorithm

A high quality solution is generated at the end of each run of SCHEDULE due to exploiting different set of assigned orders by MILP models. Thus, it is worth

to perform a local search to improve the quality of solutions further at the end of SCHEDULE. It is known that tabu search on one hand aggressively looks for local optima and can get stuck in a local optimum accordingly (Pirim et al., 2008), but on the other hand is an effective tool for fine tuning (Liaw, 2000). Hence, we select tabu search to fine tune the solutions generated and propose a multi-start granular TS algorithm using the VNS algorithm as the local search technique (GOAS_MSGTS).

The VNS algorithm explained in the previous section is adjusted to be embedded in the GOAS_MSGTS. Firstly, at this stage the makespan does not have any role in the evaluation of the solutions, so it is discarded. Secondly, we limit the size of the neighborhoods since it can be a computational burden to call VNS algorithm many times over the course of the GOAS_MSGTS. Hence, we exploit granular neighborhoods proposed by Toth and Vigo (2003) to limit the size of two insertion neighborhoods, $N_2(x)$ and $N_3(x)$. It is more favourable to insert an order into a new location such that either its new preceding or succeeding setup time is not high; otherwise, it is unlikely to observe any improvements via that insertion move. Therefore, we can enforce such promising insertions. Nonetheless, an insertion move that is promising with respect to setup times may lead to much worse solutions depending on the time windows of the orders. Thus, the time windows, along with the setup times, should be considered while generating granular neighborhoods as in the study by Schneider et al. (2017). To this end, for order $i \in O$, following scores are computed:

$$score^{pr}_{j,i} = \max\{r_i - \bar{d}_j, 0\} + st_{j,i} \qquad \forall i, j \in O$$

$$score^{sc}_{i,j} = \max\{r_j - \bar{d}_i, 0\} + st_{i,j} \qquad \forall i, j \in O$$

As the $score^{pr}_{j,i}$ ($score^{sc}_{i,j}$) gets higher, it becomes more unlikely to have order j as the predecessor (successor) of order i. For each order $i \in O$, we define sets PR_i and SC_i such that they store $\lceil n/3 \rceil$ orders with the lowest $score^{pr}_{.,i}$ and the lowest $score^{sc}_{i,.}$, respectively. Then, the insertion move $Insertion^1(x, i, j)$ is allowed to be performed if $x'[j-1] \in PR_{x[i]}$ or $x'[j+1] \in SC_{x[i]}$ where the solution x' is generated by the corresponding insertion move. Similarly, $Insertion^2(x, i, j)$ is allowed to be

performed if $x'[j-1] \in PR_{x[i]}$ or $x'[j+2] \in SC_{x[i+1]}$. When a new local solution x is generated by the VNS algorithm, $x[i-1]$ and $x[i+1]$ are added to the set $PR_{x[i]}$ and $SC_{x[i]}$, respectively, $\forall i \in O$ if they are not in the corresponding sets.

If an order i is swapped or inserted in a VNS call, that order becomes tabu and cannot be swapped or inserted in the course of the tabu tenure. The objective function value is used as the aspiration criterion.

If the GOAS_MSGTS fails at improving the incumbent solution for a given number of iterations, it is restarted by generating a new solution via *NewStart* that exploits the path relinking approach which is explained in the next section. After a certain number of restarts, the GOAS_MSGTS is terminated. General scheme of the proposed tabu search algorithm is given in Algorithm 3.7.

Algorithm 3.7 Tabu search algorithm: GOAS_MSGTS

Input: x (initial solution), $NewStartLimit$ and $IterationLimit$, E (set of elite solutions)

Initialize: $TabuList = \emptyset, x^* \leftarrow x, noimpr\_iter = 0$

for $i = 1; i \leq NewStartLimit; i++$ **do**

  repeat

    $x \leftarrow VNS(x)$

    if $f(x) > f(x^*)$ **then**

      $x^* \leftarrow x, noimpr\_iter = 0$

    else

      $noimpr\_iter++$

    end if

    $x \leftarrow$ best nontabu solution, update $TabuList$

  until $noimpr\_iter = IterationLimit$

  $x \leftarrow NewStart(x, E)$, reset $TabuList$, $noimpr\_iter = 0$

end for

Output: x^*

Generation of an initial solution for the restart of GOAS_MSGTS

Tabu search algorithms often suffer getting stuck in local optima which necessitates a diversification mechanism in this respect. Hence, we propose to use a path relinking (PL) strategy, that performs both intensification and diversification simultaneously, to restart the GOAS_MSGTS with the aim of altering the search space.

We store a set of elite solutions E, that is the best solutions generated in the course of the matheuristic algorithm, to be used in the PL approach. The cardinality of the set E is pregiven. At each call of PL, we set solution x as the starting solution and each solution of set E is used as the guiding solution (i.e., the number of paths generated at each call of PL is $|E|$). We employ swap moves to reach guiding solutions from the starting solution x.

Let $PL(x,k)$ be the set of solutions on the path between starting solution x and guiding solution $\epsilon_k \in E$ i.e., $PL(x,k) = \{x_1^k, x_2^k, \ldots, x_{|PL(x,k)|}^k\}$. Generation of set $PL(x,k)$ is presented in Algorithm 3.8.

Algorithm 3.8 Generation of set $PL(x,k)$

Input: x (local optimal solution), ϵ_k (kth elite solution)

$Swap(x,k)$: Set of all solutions that are one swap move away from solution x and an order in x is moved to its position in ϵ_k by the corresponding swap move

$PL(x,k) = \emptyset$

repeat

$x \leftarrow$ The solution in $Swap(x,k)$ that has the maximum objective function value

Add x to the end of set $PL(x,k)$

until $x = \epsilon_k$

Path relinking approach is employed to restart the GOAS_MSGTS and all solutions in $\bigcup_{k=1}^{|E|} PL(x,k)$ are candidates to be the initial solution upon the restart. We evaluate set $PL(x,k)$ separately for each $\epsilon_k \in E$ and pick a single solution from each set; the best solution, in terms of the objective function value, among the picked ones is set to be the initial solution of the restarted GOAS_MSGTS. New initial

solution should not be too close to neither x nor any $\epsilon_k \in E$ since it is likely to get stuck at a local optimum previously visited. Therefore, we set a distance measure Δ so that the solutions in $PL(x,k)$ that can be reached by at most Δ swap moves from either x or any $\epsilon_k \in E$, should not be chosen as the new initial solution unless they satisfy certain aspiration criteria. Figure 3.4 illustrates how to pick a solution from set $PL(x,k)$.

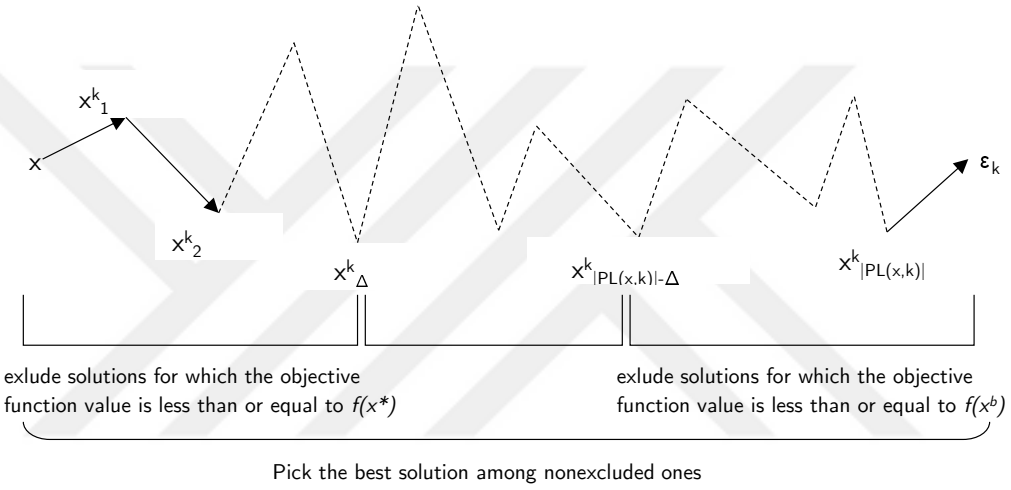

Figure 3.4: Path relinking illustration.

For each set $PL(x,k)$, solutions in the vicinity of x are omitted if their objective function value is not higher than that of solution x^* which is the best solution found by the GOAS_MSGTS in the corresponding iteration. Similarly, solutions in the vicinity of ϵ^k are omitted if their objective function value is not higher than that of solution x^b which is the best solution found throughout GOAS_I2PM. The one with the highest objective function value among the non-omitted solutions is set to be the new initial solution as shown in Algorithm 3.9.

Algorithm 3.9 *NewStart*(x, E) to generate a new initial solution for the GOAS_MSGTS

Initialize: $x^{new} \leftarrow x$, $candidate\_objective = 0$

for $k : 1; k \leq |E|; k++$ **do**

  for $i : 1; i \leq |PL(x,k)|; i++$ **do**

    if $i <\triangle$ **then**

      $threshold\_objective = max\{candidate\_objective, f(x^*)\}$

    else if $i < |PL(x,k)| - \triangle$ **then**

      $threshold\_objective = candidate\_objective$

    else

      $threshold\_objective = max\{candidate\_objective, f(x^b)\}$

    end if

    if $f(x_i^k) > threshold\_objective$ **then**

      $candidate\_objective = f(x_i^k)$, $x^{new} \leftarrow x_i^k$

    end if

  end for

end for

$x \leftarrow x^{new}$

Output: x

3.2.4 Interaction between ASSIGN *and* SCHEDULE

GOAS_I2PM is based on solving ASSIGN and SCHEDULE iteratively as the latter feeds the former in a twofold manner in each iteration as follows:

i) SCHEDULE generates new cut(s) to avoid ASSIGN allocating the same set of orders to the time segments, and

ii) the GOAS_RTBM model uses parameter p' as an input starting from the second iteration of GOAS_I2PM. Parameter p' is modified based on GOAS_MSGTS after SCHEDULE is run.

For each time segment $s \in S$, a new cut is generated if there are any orders, which are not accepted, in the corresponding segment. Using the set A_s, the set of accepted orders in O_s, $\forall s \in S$, new cuts can be generated as follows:

$$\sum_{i \in A_s} a_{i,s} |O_s \backslash A_s| + \sum_{i \in O_s \backslash A_s} a_{i,s} \leq |A_s||O_s \backslash A_s| \qquad \forall s \in S : O_s \not\equiv A_s. \tag{3.23}$$

Cut (3.23) ensures that if all orders in A_s are assigned to time segment s, none of the orders in $O_s \backslash A_s$ can be assigned to time segment s in future iterations. It is sufficient to add a cut of this type in any iteration to generate a new assignment of orders in the next iteration. However, if $O_s \equiv A_s$, $\forall s \in S$, in an iteration, we cannot derive such cuts. In this case, Cut (3.24) is generated.

$$\sum_{s \in S, i \in O_s} a_{i,s} \leq \sum_{s \in S} |O_s| - 1. \tag{3.24}$$

Addition of Constraint (3.23) or Constraint (3.24) to the GOAS_RTBM model is sufficient to generate a new solution by ASSIGN and to avoid entering into a loop during the iterations between ASSIGN and SCHEDULE. However, the convergence of GOAS_I2PM can be gradual due to the exclusion of setup times in ASSIGN. Therefore, GOAS_I2PM uses the local optimum obtained by GOAS_MSGTS after SCHEDULE in the current iteration, to update the processing times, denoted by p', to be used in the next iteration by the GOAS_RTBM model to include setup times implicitly. st'_i denotes the estimation of the setup time in the optimal solution before order i and initially, it is set to $minsetup_i$, $\forall i \in O$. As a new local optimum x is generated, st'_i is set to $0.9st'_i + 0.1s^x_i$ where s^x_i denotes the setup time executed before order i in solution x and it is equal to st'_i if order i is not accepted in solution x. Then, p'_i is set to $p'_i = p_i + st'_i$, $\forall i \in O$. As the new local optimum solutions are obtained, the setup time estimations are expected to converge to more accurate values, which leads to more precise assignments at the first phase, ASSIGN.

3.3 Computational experiments

In this section, we compare the computational performance of GOAS_I2PM with both the proposed mathematical models and the state-of-the-art algorithms SSP (Silva et al., 2018) and SPARROW (He et al., 2019) for the GOAS problem. Since both algorithms use the problem instances generated by Cesaret et al. (2012), same instances are used in this study for the purpose of comparison. In data generation, Cesaret et al. (2012) use two predefined parameters, the due date range R, and the tardiness factor τ, to vary the problem instances covering a wide range of cases. Both τ and R take values in the set $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. The number of orders, n, is set to 10, 15, 20, 25, 50 and 100. For each possible combination of n, τ and R, there are 10 instances.

All the runs pertaining to the proposed mathematical models and GOAS_I2PM are performed on a workstation with an Intel Core i7 processor, 2.60 GHz speed, and 8 GB of RAM, through Visual Studio 2015 and ILOG CPLEX 12.9. We set a time limit of 3600s and 900s for the mathematical model and the matheuristic runs, respectively.

In a preliminary computational experiment, we tested the values for several parameters, namely $|S|$, $|E|$, Δ, tabu tenure, $NewStartLimit$ and $IterationLimit$, in different combinations of values for each parameter. We present the selected final values for these parameters in Table 3.1.

Table 3.1: The values for the parameters of GOAS_I2PM.

n	**|S|**	$\triangle$	**|E|**	**Tabu tenure**	***NewStartLimit***	***IterationLimit***
50	3	5	10	5	5	100
100	6	5	10	5	5	100

The most significant parameter is the size of set S as expected and SCHEDULE gives better solutions as the cardinality of set S decreases which would require higher run time in return. Therefore, the size of set S is determined so as to have a few

time segments but still can be solved quickly by the MILP model. When the number of segments are three and six for the instances with 50 and 100 orders, respectively, at most 20 orders are assigned to a single time segment in ASSIGN in the majority of the iterations. As shown in Table 3.4, the MILP model can solve the instances with 20 orders in a short time. Therefore, it is unnecessary to have larger number of time segments so that the scheduling problem for each time segment can be solved faster. On the other hand, solving the scheduling problem of a single time segment can consume too much time if the planning horizon is divided into smaller number of segments. Consequently, cardinality of set S given in Table 3.1 is used in the computational runs.

As Δ and tabu tenure take values less than 5, cycles in the tabu search are observed more often. As the values for *NewStartLimit* and *IterationLimit* increase, the tabu search occupies more of the total run time within GOAS_I2PM. Preliminary runs show that these parameters should be small to enable higher number of iterations within GOAS_I2PM.

The computational results of the comparisons of GOAS_I2PM with respect to the mathematical models and the state-of-the-art algorithms are presented in Tables 3.2-3.7 and Tables 3.10-3.11. In these tables, the column-headings denote the following:

i) Gap(%): For each instance group, denoted by (n, τ, R), 10 instances are solved and the optimality gap of an instance by a method (an algorithm or a commercial solver) is computed as

$$\text{Gap} = \frac{UB_{instance} - LB_{instance}^{method}}{UB_{instance}} \tag{3.25}$$

where $LB_{instance}^{method}$ is the objective function value of the best solution found by the corresponding method for the corresponding instance and $UB_{instance}$ is the upper bound on the objective function value of the instance. $UB_{instance}$ is taken as the minimum of three values: upper bound given by the MILP model, upper bound given by the CP model, and the objective function value of the optimal solution (or the upper bound if the optimal solution cannot be found) of the relaxed instances (i.e., when sequence dependent setup times are

omitted). Then Gap(%) denotes the percentage of the optimality gap, i.e., Gap(%) = Gap × 100. Finally, Min, Avg and Max columns under the Gap(%) heading show the minimum, the average and the maximum optimality gaps, respectively, of 10 instances falling in an instance group (n, τ, R) in each row.

ii) NO: Number of instances for which the optimal solution can be proven by the proposed mathematical models or matheuristic.

iii) NF: Number of instances for which the respective algorithm/solver finds the optimal solution out of NO instances.

3.3.1 Computational results of MILP *and* CP *for instances with* $n = 10, 15, 20, 25$

In this study, one of our aims is to push the boundary of optimally solved instance size further. To fulfil this aim, we test the performance of the proposed mathematical models, namely MILP and CP, with respect to whether the performance of the mathematical models is robust in terms of the solution quality and the run time. More specifically, we want to establish when the performance of these models deteriorates regarding the instance sizes, for which instances large optimality gaps are observed, and which instances can easily be solved to the optimality.

In Tables 3.2-3.5, we present the percentage optimality gap, the number of optimal solutions found, and the solution time of each model in comparison to the state-of-the-art results of Silva et al. (2018) for the instances with $n = 10, 15, 20, 25$. Silva et al. (2018) presented the best results of a branch and price algorithm and an iterated local search algorithm without differentiating which algorithm found the presented best results. We will refer to these results as SSP hereafter.

We can observe from Table 3.2 that for the instances with 10 orders, the performance of both the proposed mathematical models and SSP are indistinguishable since they can find the optimal solution of each instance in a second. When $n = 15$, as we see from Table 3.3, the proposed mathematical models become superior to SSP in terms of both the solution quality and the run time. Although the run time of SSP is higher than that of the proposed mathematical models, it cannot find the

optimal solution of one instance while both the MILP and the CP models can find the optimal solution of all instances. In terms of the run time, the MILP model is slightly better than the CP model.

We note a similar superior performance for the instances with $n = 20$, where the results are presented in Table 3.4.

Table 3.2: Comparison of the proposed mathematical models and SSP for $n = 10$.

			Gap (%)									NO	NF			Solution time (s)		
n	τ	R	SSP			MILP			CP				SSP	MILP	CP	SSP	MILP	CP
			Min	Avg	Max	Min	Avg	Max	Min	Avg	Max							
10	0.1	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.90	0.12	0.60
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.90	0.11	0.38
		0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	2.70	0.14	0.49
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	2.00	0.17	0.57
		0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.50	0.19	0.22
	0.3	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.30	0.13	0.53
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.20	0.14	0.48
		0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.30	0.17	0.47
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.80	0.17	0.83
		0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.10	0.19	0.41
	0.5	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.10	0.14	0.71
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.20	0.17	0.76
		0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.10	0.15	0.52
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.10	0.19	0.88
		0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.10	0.17	0.73
	0.7	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.10	0.10	0.34
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.10	0.08	0.27
		0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.10	0.11	0.44
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.10	0.11	0.39
		0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.10	0.12	0.38
	0.9	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.10	0.05	0.07
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.10	0.06	0.09
		0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.10	0.05	0.13
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.10	0.07	0.13
		0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.10	0.07	0.42
Avg:			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	250.00	250.00	250.00	250.00	0.41	0.13	0.45

On the other hand, we observe from Table 3.5 that, for instances with 25 orders, SSP performs better than the CP model, except for the ones with small τ values, with respect to the solution albeit requiring higher run time. The MILP model still prevails the CP model and SSP as it can solve all instances with 25 orders to the optimality in reasonable time. However, when n is larger than 25, the MILP model cannot maintain its competitiveness with neither the CP model nor SSP. Due to this observation, we run the proposed matheuristic for only instances with $n = 50$ and 100 and analyze its performance by comparing with the MILP and the CP models, SSP and SPARROW in Section 3.3.2.

Table 3.3: Comparison of the proposed mathematical models and SSP for $n = 15$.

			Gap (%)									NO	NF			Solution time (s)		
n	τ	R	SSP			MILP			CP				SSP	MILP	CP	SSP	MILP	CP
			Min	Avg	Max	Min	Avg	Max	Min	Avg	Max							
15	0.1	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	693.10	0.28	8.33
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	632.80	0.29	6.20
		0.5	0.00	0.02	0.16	0.00	0.00	0.00	0.00	0.00	0.00	10.00	9.00	10.00	10.00	519.60	0.60	2.19
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	500.00	0.94	1.52
		0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	10.90	0.69	1.70
	0.3	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	9.90	0.41	5.26
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	15.20	0.35	4.71
		0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	48.90	0.63	2.35
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	16.10	1.98	1.46
		0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	14.50	2.10	1.20
	0.5	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	2.70	0.77	2.70
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	1.70	0.78	2.49
		0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.90	0.75	3.84
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	1.40	1.10	1.50
		0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	1.70	1.17	2.28
	0.7	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.30	0.44	0.85
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.10	0.48	0.81
		0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.20	0.62	1.02
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.10	0.70	1.15
		0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.30	2.16	1.02
	0.9	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.10	0.10	0.18
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.10	0.13	0.46
		0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.10	0.20	0.70
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.10	0.18	0.56
		0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.10	0.37	1.48
Avg:			0.000	0.001	0.007	0.000	0.000	0.000	0.000	0.000	0.000	250.00	249.00	250.00	250.00	98.84	0.73	2.24

Table 3.4: Comparison of the proposed mathematical models and SSP for $n = 20$.

			Gap (%)									NO	NF			Solution time (s)		
n	τ	R	SSP			MILP			CP				SSP	MILP	CP	SSP	MILP	CP
			Min	Avg	Max	Min	Avg	Max	Min	Avg	Max							
20	0.1	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	1527.40	0.86	46.03
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	1440.00	0.73	54.75
		0.5	0.00	0.03	0.28	0.00	0.00	0.00	0.00	0.00	0.00	10.00	9.00	10.00	10.00	1653.50	3.47	14.17
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	1191.40	8.32	7.04
		0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	104.50	12.71	3.12
	0.3	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	576.10	1.62	79.61
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	688.70	1.82	33.20
		0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	89.70	2.32	12.76
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	1576.50	9.99	9.64
		0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	32.90	31.81	4.04
	0.5	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	15.20	2.70	18.56
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	28.80	2.73	12.12
		0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	18.20	5.00	7.09
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	40.40	7.22	3.62
		0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	4.70	13.04	2.15
	0.7	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	7.50	2.31	3.68
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	2.00	2.14	2.29
		0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.60	4.81	2.57
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	1.40	4.01	1.96
		0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.30	6.60	2.34
	0.9	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.10	0.16	1.09
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.10	0.22	1.17
		0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.10	0.39	1.17
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.30	0.58	1.42
		0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.10	1.08	1.59
Avg:			0.000	0.001	0.011	0.000	0.000	0.000	0.000	0.000	0.000	250.00	249.00	250.00	250.00	360.02	5.07	13.09

Table 3.5: Comparison of the proposed mathematical models and SSP for $n = 25$.

			Gap (%)									NO	NF			Solution time (s)		
n	τ	R	SSP			MILP			CP				SSP	MILP	CP	SSP	MILP	CP
			Min	Avg	Max	Min	Avg	Max	Min	Avg	Max							
25	0.1	0.1	0.00	0.08	0.77	0.00	0.00	0.00	0.00	0.05	0.46	10.00	9.00	10.00	9.00	2221.10	2.17	1378.25
		0.3	0.00	0.03	0.32	0.00	0.00	0.00	0.00	0.00	0.00	10.00	9.00	10.00	10.00	2219.00	3.05	340.52
		0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	1307.70	11.87	74.25
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	1090.00	24.03	8.91
		0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	1555.70	109.85	84.11
	0.3	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	1272.40	4.64	848.33
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	1901.80	10.53	168.86
		0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	2584.40	34.67	372.97
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	871.50	157.69	217.84
		0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.20	10.00	10.00	10.00	9.00	973.20	317.26	751.90
	0.5	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	286.80	13.29	1535.19
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.08	0.83	10.00	10.00	10.00	9.00	885.90	57.45	919.03
		0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	776.10	299.94	700.61
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	627.30	622.92	62.97
		0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	449.40	947.41	52.51
	0.7	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	12.40	15.53	87.75
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	11.10	14.68	49.46
		0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	3.30	32.54	10.62
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	9.50	435.53	4.77
		0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	12.30	824.59	3.46
	0.9	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.10	0.27	2.23
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.10	0.60	2.92
		0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.20	1.12	2.77
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	0.40	3.41	3.68
		0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00	10.00	10.00	10.00	1.10	26.03	5.19
Avg:			0.000	0.004	0.043	0.000	0.000	0.000	0.000	0.006	0.059	250.00	248.00	250.00	247.00	762.91	158.84	307.56

3.3.2 Computational results of MILP*,* CP *and* GOAS_I2PM *matheuristic for instances with* $n = 50, 100$

The results presented in Section 3.3.1 for small sized instances of GOAS problem show that the mathematical models developed are quite successful in obtaining the optimal solutions in reasonable run times. However, when we solve the medium and large sized instances of the GOAS problem with these mathematical models, we observe the following general trends for both the MILP and the CP models:

i) The performance of the mathematical models can deteriorate notably as the number of orders increases,

ii) the performance of the mathematical models is not robust (i.e., high optimality gaps can be observed for some instances while the others can easily be solved

to the optimality), and

iii) the run time of these models can be high even if they can find the optimal solution.

These observations motivated us to develop a matheuristic algorithm for large sized instances of the GOAS problem to achieve our goals for this study, which were stated at the beginning of Chapter 3 as:

1. to push the boundary of optimally solved instance size further, and

2. to obtain near optimal solutions in reasonable time for instances falling beyond this boundary,

The proposed matheuristic GOAS_I2PM capitalizes on the strengths of the time-bucket and MILP models, and two metaheuristics, which are known to perform better for scheduling problems, namely VNS and TS. We run GOAS_I2PM for medium and large sized instances of the GOAS problem and analyze its performance compared to the mathematical models, SSP and SPARROW when $n = 50$ and 100.

As it can be observed from Tables 3.6-3.7, the CP model and GOAS_I2PM outperform the MILP model in terms of both the optimality gap and the number of optimal solutions found when $n = 50$ and 100. The MILP model can find the optimal solution of all instances when both τ and R are equal to 0.1 (i.e., the instances with small release times, large due dates and deadlines) which is expected since the MILP model performs better as the constraints get looser. However, its poor performance in the most of the remaining instances makes the MILP model uncompetitive with the others (e.g., the average gap of the MILP model is 28.77% when $n = 100$, $\tau = 0.5$).

When $n = 100$, optimal solution of 11, 47, and 63 instances are found by the MILP model, the CP model and GOAS_I2PM, respectively, as Table 3.7 indicates. Optimality gaps of both the CP model and GOAS_I2PM deviate within a smaller range except for the instances where $n = 50$, $\tau = 0.7$ and $n = 100$, $\tau = 0.7$ or 0.9.

Regarding the relevant instances, it should be noted that the optimal solutions of none of these instances with 100 orders are known as shown in Table 3.7.

Table 3.6: Comparison of GOAS_I2PM with the proposed mathematical models for $n = 50$.

			Gap (%)									NO	NF			Solution time (s)		
n	τ	R	MILP			CP			GOAS_I2PM				MILP	CP	GOAS_I2PM	MILP	CP	GOAS_I2PM
			Min	Avg	Max	Min	Avg	Max	Min	Avg	Max							
50	0.1	0.1	0.00	0.00	0.00	0.00	0.45	0.80	0.00	0.02	0.19	10.00	10.00	2.00	9.00	145.04	3296.05	900.00
		0.3	0.00	0.00	0.00	0.00	0.07	0.34	0.00	0.02	0.17	10.00	10.00	7.00	9.00	166.12	1854.18	900.00
		0.5	0.00	0.36	2.03	0.00	0.06	0.55	0.00	0.00	0.00	10.00	5.00	9.00	10.00	2762.04	953.35	900.00
		0.7	0.00	0.14	0.56	0.00	0.00	0.00	0.00	0.00	0.00	10.00	7.00	10.00	10.00	2372.16	2.54	900.00
		0.9	0.00	1.21	4.13	0.00	0.00	0.00	0.00	0.00	0.00	10.00	2.00	10.00	10.00	3568.20	8.60	900.00
	0.3	0.1	0.00	0.02	0.16	0.00	0.22	0.72	0.00	0.07	0.19	9.00	9.00	4.00	6.00	1249.73	3051.26	900.00
		0.3	0.00	0.10	0.59	0.00	0.10	0.41	0.00	0.14	0.41	9.00	8.00	7.00	5.00	2178.41	2366.09	900.00
		0.5	0.00	0.61	1.50	0.00	0.10	0.64	0.00	0.08	0.44	8.00	3.00	8.00	8.00	3413.08	1796.21	900.00
		0.7	0.00	1.75	3.47	0.00	0.03	0.35	0.00	0.03	0.35	9.00	1.00	9.00	9.00	3521.54	977.26	900.00
		0.9	0.88	1.90	4.28	0.00	0.10	1.02	0.00	0.06	0.61	9.00	0.00	9.00	9.00	3600.00	599.99	900.00
	0.5	0.1	0.00	2.89	5.84	0.00	0.61	1.17	0.00	0.48	1.03	2.00	1.00	2.00	2.00	3600.00	2931.87	900.00
		0.3	0.56	2.67	4.73	0.00	0.94	1.74	0.00	0.69	1.27	2.00	0.00	2.00	2.00	3600.00	3031.47	900.00
		0.5	1.47	5.02	10.08	0.00	1.28	2.26	0.00	1.05	2.26	1.00	0.00	1.00	1.00	3600.00	3245.92	900.00
		0.7	2.46	4.06	7.46	0.00	0.83	1.87	0.00	0.79	1.87	2.00	0.00	2.00	2.00	3600.00	3041.95	900.00
		0.9	3.68	5.87	8.00	0.00	1.23	2.59	0.00	1.01	2.45	2.00	0.00	2.00	2.00	3600.00	3005.01	900.00
	0.7	0.1	1.17	2.15	3.33	0.64	1.55	2.79	0.42	1.13	2.11	0.00	0.00	0.00	0.00	3600.00	3600.00	900.00
		0.3	0.48	1.82	2.80	0.48	1.35	3.29	0.37	0.94	1.49	0.00	0.00	0.00	0.00	3600.00	3600.00	900.00
		0.5	0.72	2.96	4.89	0.91	2.08	3.33	0.91	1.99	3.13	0.00	0.00	0.00	0.00	3600.00	3600.00	900.00
		0.7	0.61	4.49	8.73	0.00	2.68	6.42	0.00	2.64	6.06	3.00	0.00	3.00	3.00	3600.00	2807.91	900.00
		0.9	0.41	5.16	9.47	0.00	3.19	8.00	0.00	3.02	8.00	4.00	0.00	4.00	4.00	3600.00	2233.09	900.00
	0.9	0.1	0.00	0.09	0.86	0.00	0.55	1.95	0.00	0.09	0.86	9.00	9.00	5.00	9.00	541.82	3241.00	900.00
		0.3	0.00	0.37	2.08	0.00	0.36	1.64	0.00	0.33	1.64	8.00	8.00	7.00	7.00	1682.76	1964.24	900.00
		0.5	0.00	1.96	3.45	0.00	2.29	7.32	0.00	1.42	3.45	4.00	2.00	4.00	4.00	3346.96	2739.28	900.00
		0.7	0.00	2.22	7.84	0.00	2.04	8.35	0.00	2.02	7.25	7.00	2.00	7.00	5.00	3600.00	1322.60	900.00
		0.9	0.00	4.97	8.81	0.00	4.52	9.73	0.00	4.15	8.65	4.00	2.00	4.00	4.00	3600.00	2320.33	900.00
Avg:			0.498	2.110	4.203	0.082	1.066	2.691	0.068	0.887	2.155	142.00	79.00	118.00	130.00	2869.91	2303.61	900.00

Similarly, for only seven instances out of 50, the optimal solution is known when $n = 50$ and $\tau = 0.7$. Thus, the optimality gap values of the instances, for which the optimality is not proven, are computed by using the upper bounds of the optimal objective function values, not by using their exact values.

We further investigate the number of optimal solutions found by the mathematical models and the GOAS_I2PM to show that we push the boundary of optimally solved instance size more than the literature. In Table 3.8, we compare the number of optimal solutions known in the literature and that we find in this study. We particularly note that our proposed solution methods are able to find the optimal solution of all instances when $n = 25$ and to find the optimal solution of many instances with $n = 50$ and $n = 100$ of which optimal solutions were not known before. Note that when $n = 50$ and $\tau \in \{0.7, 0.9\}$, available methods were better at finding

Table 3.7: Comparison of GOAS_I2PM with the proposed mathematical models for $n = 100$.

			Gap (%)									NO	NF			Solution time (s)		
n	τ	R	MILP			CP			GOAS_I2PM				MILP	CP	GOAS_I2PM	MILP	CP	GOAS_I2PM
			Min	Avg	Max	Min	Avg	Max	Min	Avg	Max							
100	0.1	0.1	0.00	0.00	0.00	0.10	0.57	1.28	0.00	0.04	0.26	10.00	10.00	0.00	8.00	1817.58	3600.00	900.00
		0.3	0.00	0.45	1.86	0.20	0.50	0.82	0.00	0.07	0.19	3.00	1.00	0.00	3.00	3561.59	3600.00	900.00
		0.5	2.44	6.91	11.04	0.00	0.05	0.27	0.00	0.00	0.00	10.00	0.00	8.00	10.00	3600.00	756.05	900.00
		0.7	5.75	9.34	12.95	0.00	0.00	0.00	0.00	0.00	0.00	10.00	0.00	10.00	10.00	3600.00	2.23	900.00
		0.9	12.34	16.27	25.68	0.00	0.00	0.00	0.00	0.00	0.00	10.00	0.00	10.00	10.00	3600.00	0.25	900.00
	0.3	0.1	2.38	5.10	7.65	0.36	0.57	0.89	0.10	0.23	0.36	0.00	0.00	0.00	0.00	3600.00	3600.00	900.00
		0.3	4.07	7.87	12.49	0.45	0.68	1.14	0.19	0.30	0.52	0.00	0.00	0.00	0.00	3600.00	3600.00	900.00
		0.5	12.04	16.10	23.28	0.45	0.64	1.11	0.20	0.36	0.52	0.00	0.00	0.00	0.00	3600.00	3600.00	900.00
		0.7	10.82	26.13	36.20	0.00	0.10	0.63	0.00	0.08	0.36	7.00	0.00	7.00	7.00	3600.00	1242.41	900.00
		0.9	23.26	31.14	40.01	0.00	0.00	0.00	0.00	0.00	0.00	10.00	0.00	10.00	10.00	3600.00	137.21	900.00
	0.5	0.1	17.43	23.38	36.90	0.56	1.04	1.93	0.34	0.52	0.88	0.00	0.00	0.00	0.00	3600.00	3600.00	900.00
		0.3	18.03	23.94	30.46	0.53	1.03	1.72	0.33	0.68	0.96	0.00	0.00	0.00	0.00	3600.00	3600.00	900.00
		0.5	18.70	31.26	45.03	0.51	0.83	1.16	0.28	0.56	0.80	0.00	0.00	0.00	0.00	3600.00	3600.00	900.00
		0.7	27.06	34.75	44.86	0.17	0.59	0.99	0.00	0.51	0.99	1.00	0.00	0.00	1.00	3600.00	3600.00	900.00
		0.9	20.26	30.53	42.12	0.00	0.43	1.47	0.00	0.34	1.47	4.00	0.00	2.00	4.00	3600.00	3191.52	900.00
	0.7	0.1	14.03	19.56	29.67	0.90	1.40	1.94	0.75	1.19	1.67	0.00	0.00	0.00	0.00	3600.00	3600.00	900.00
		0.3	9.26	18.71	26.80	0.88	2.11	4.67	0.69	1.90	4.49	0.00	0.00	0.00	0.00	3600.00	3600.00	900.00
		0.5	16.20	22.69	34.25	0.99	1.82	2.90	0.83	1.54	2.16	0.00	0.00	0.00	0.00	3600.00	3600.00	900.00
		0.7	15.58	21.35	30.74	1.01	2.79	4.41	1.10	2.67	4.56	0.00	0.00	0.00	0.00	3600.00	3600.00	900.00
		0.9	16.76	23.83	32.50	0.77	2.79	3.85	1.05	2.88	4.53	0.00	0.00	0.00	0.00	3600.00	3600.00	900.00
	0.9	0.1	1.81	3.28	5.24	1.08	2.67	4.34	1.08	2.18	3.59	0.00	0.00	0.00	0.00	3600.00	3600.00	900.00
		0.3	4.76	7.08	10.40	3.50	6.12	11.11	3.34	5.16	8.61	0.00	0.00	0.00	0.00	3600.00	3600.00	900.00
		0.5	6.28	9.50	12.00	5.22	6.74	9.23	4.95	6.21	8.75	0.00	0.00	0.00	0.00	3600.00	3600.00	900.00
		0.7	8.53	11.67	14.83	4.80	7.30	11.79	4.74	6.99	10.34	0.00	0.00	0.00	0.00	3600.00	3600.00	900.00
		0.9	9.81	14.68	21.46	2.18	7.27	10.27	2.68	6.64	10.27	0.00	0.00	0.00	0.00	3600.00	3600.00	900.00
Avg:			11.103	16.621	23.537	0.987	1.922	3.116	0.906	1.641	2.651	65.00	11.00	47.00	63.00	3527.17	2949.19	900.00

the optimals.

For the instances of which optimal solutions are not known, we use both the upper bounds given by the mathematical models and the optimal solutions (or the upper bounds if the optimal solutions cannot be found) of the relaxed instances (i.e., when sequence dependent setup times are omitted) to compute the optimality gap values; the minimum of the corresponding objective function values is assigned as the upper bound. When n is 100 and τ is 0.7 or 0.9, optimality gaps of 62 instances out of 100 are computed by using the upper bounds provided by the relaxed instances (i.e., the optimal value or the upper bound of the GOAS problem without sequence dependent setup times gives the tightest upper bounds). Thus, for the corresponding instances, it is likely to observe higher optimality gap values due to weak upper bounds. A similar explanation can be made for the instances with 50 orders where τ is 0.7; optimality gaps of these instances are higher than that of other instances with 50 orders because of the weak upper bounds - solutions of only 7 out of 50 instances

Table 3.8: Comparison of the number of optimal solutions found in the literature and in this study.

		n=25		n=50		n=100	
τ	R	Literature	This study	Literature	This study	Literature	This study
0.1	0.1	3	10	0	10	0	10
	0.3	4	10	0	10	0	3
	0.5	7	10	3	10	6	10
	0.7	10	10	10	10	10	10
	0.9	10	10	10	10	10	10
0.3	0.1	6	10	0	9	0	0
	0.3	3	10	0	9	0	0
	0.5	3	10	1	8	0	0
	0.7	9	10	6	9	5	7
	0.9	8	10	8	9	8	10
0.5	0.1	10	10	0	2	0	0
	0.3	8	10	1	2	0	0
	0.5	8	10	0	1	0	0
	0.7	9	10	0	2	0	1
	0.9	9	10	1	2	1	4
0.7	0.1	10	10	0	0	0	0
	0.3	10	10	1	0	0	0
	0.5	10	10	1	0	0	0
	0.7	10	10	7	3	0	0
	0.9	10	10	7	4	0	0
0.9	0.1	10	10	10	9	0	0
	0.3	10	10	10	8	0	0
	0.5	10	10	10	4	0	0
	0.7	10	10	10	7	0	0
	0.9	10	10	10	4	0	0
Sum:		207	250	106	142	40	65

can be proved to be optimal by the proposed mathematical models.

It is judicious to state that the order acceptance decisions become obvious if most of the orders can be accepted or many of them has to be rejected. Performance of both the MILP and the CP models improves if determining the right set of accepted orders becomes easier. Effect of the τ value on the instances with 50 orders justifies this statement. When τ is equal to 0.1, time windows of the orders are large which enables to accept the majority of the orders. On the other hand, when τ is equal to 0.9, the number of accepted orders decreases significantly due to the more restrictive time windows. In both cases, the MILP and the CP models perform better in comparison with the instances where τ is between 0.1 and 0.9 that makes the order acceptance decisions more difficult. However, it should be noted that as the number of orders increases to 100, both mathematical models fail at finding the optimal solutions even if τ value is high.

Both Table 3.6 and Table 3.7 show that performance of the CP model improves as R value increases. Although its average optimality gap for the instances with high τ value increases with R value, this is mostly due to loose bounds provided by the MILP model in the corresponding instances. The CP model can find more of the corresponding instances' optimal solutions as R value increases which supports the positive impact of high R value on the performance of the CP model. This is an expected outcome since the range of the due dates gets larger (i.e., due dates become more distinct) as the value of R increases which is in favor of the CP model unlike the MILP model that performs better as the due dates are closer to each other. Thus, for all τ values, these models show complementary behaviours with respect to the correlation between their performance and R value. We also observe that GOAS_I2PM surpasses the mathematical models in terms of the optimality gap values, the number of optimal solutions found and the run time. GOAS_I2PM can find the optimal solution of 130 out of 142 and 63 out of 65 instances of which optimal solutions are known when n is equal to 50 and 100, respectively. Regarding the number of optimal solutions found, GOAS_I2PM performs worse as the time windows get looser. Therefore, when $n = 50$ and, τ and R take values from the set $\{0.1, 0.3\}$, it can find the optimal solution of 29 instances out of 38 known optimals. We can explain this behaviour with the fact that when time windows are loose, setup times become more effective in determining the sequence of the orders. However, as they become tighter, the release times and the deadlines become instrumental in the sequencing decisions. Thus, since ASSIGN of GOAS_I2PM omits the setup times, we expect to observe better performance of GOAS_I2PM as the values of τ and R increase.

As shown in Table 3.9, the MILP model gives the smallest optimality gap on average when $n = 50$ and τ and R take values from the set $\{0.1, 0.3\}$ or when $n = 100$ and both τ and R are equal to 0.1. On the other hand, SSP is the best solution approach, with respect to the average optimality gaps, when $n = 50$ and τ and R are equal to 0.7 or τ=0.9 and $R \in \{0.3, 0.7\}$. In the rest of the problem instances, GOAS_I2PM achieves smaller optimality gaps than both the proposed

models MILP and CP, and the state-of-the-art algorithms SSP and SPARROW.

Table 3.9: Best solution approach, with respect to average optimality gaps, for each (τ,R) pair when n=50 and 100

$\boldsymbol{n}$=50	$\boldsymbol{\tau}$	$\boldsymbol{R}$	**MILP**	**GOAS_I2PM**	**SSP**
	0.1, 0.3	0.1, 0.3	✓		
	0.1, 0.3	0.5, 0.7, 0.9		✓	
	0.5	0.1, 0.3, 0.5, 0.7, 0.9		✓	
	0.7	0.1, 0.3, 0.5, 0.9		✓	
	0.7	0.7			✓
	0.9	0.1, 0.5, 0.9		✓	
	0.9	0.3, 0.7			✓
$\boldsymbol{n}$=100	0.1	0.1	✓		
	0.1	0.3, 0.5, 0.7, 0.9		✓	
	0.3, 0.5, 0.7, 0.9	0.1, 0.3, 0.5, 0.7, 0.9		✓	

Although the MILP model performs better than GOAS_I2PM in some of the problem instances, we note that the MILP model approaches to the optimal solutions gradually, except for the instances with $n = 50$ and $\tau = 0.1$, while GOAS_I2PM can find near optimal solutions in short times, as shown in the convergence graphs of the corresponding instances depicted in Figure 3.5. Hence, due to the agility of GOAS_I2PM in reaching a good solution, we can conclude that it can be terminated early without sacrificing the quality of the solution immensely. In Figure 3.5, we plot the average gap of the corresponding instance groups, to show the convergence of the mathematical models and GOAS_I2PM. We note that the convergence line of GOAS_I2PM is straight from 900s to 3600s since it runs for 900s.

While the competitiveness of GOAS_I2PM is shown by Tables 3.6-3.7, it also outperforms the state-of-the-art algorithms SSP and SPARROW in terms of solution quality as can be seen in Tables 3.10-3.11. Since He et al. (2019) use the upper

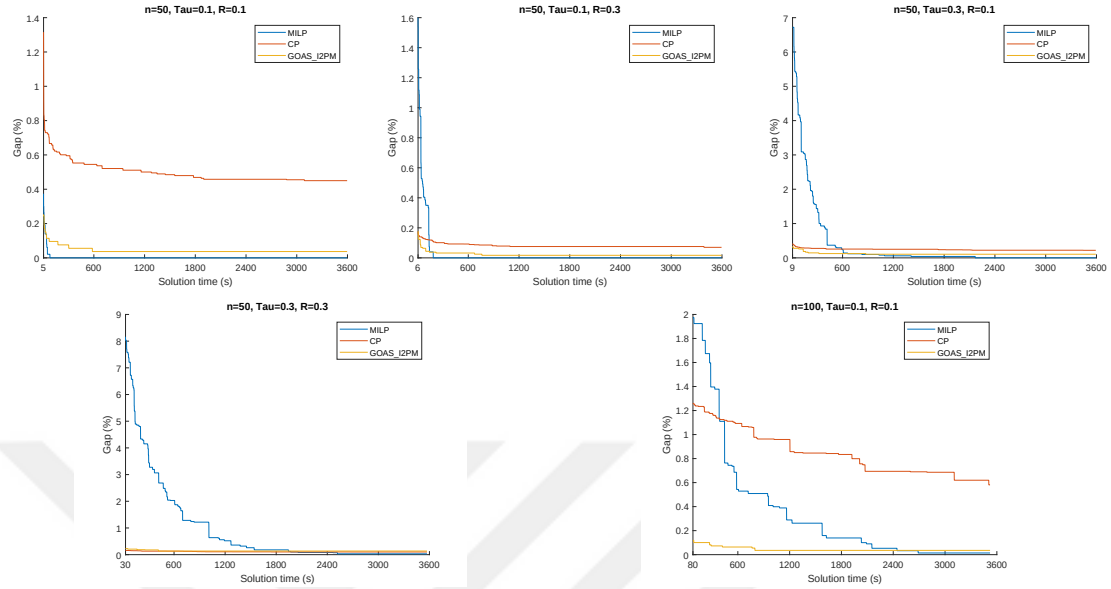

Figure 3.5: Convergence graphs for the mathematical models and GOAS_I2PM for particular instances.

bounds given by Cesaret et al. (2012) to evaluate the performance of SPARROW, same bounds are used in Tables 3.10-3.11 for the purpose of having a fair comparison. Accordingly, gap values in these tables are calculated by using the equation (3.25) with the upper bounds of Cesaret et al. (2012) instead of $UB_{instance}$ values. While SSP can achieve smaller optimality gaps than the other algorithms for a few instance groups with 50 orders, both GOAS_I2PM and SPARROW can generate better solutions when the number of instances rise to 100 regardless of τ and R values. It is worth of note that GOAS_I2PM can provide smaller optimality gaps than SPARROW for any combination of n, τ and R. On the other hand, the run time of SPARROW is much less than that of GOAS_I2PM. However, as Figure 3.5 implies, performance of GOAS_I2PM is slightly affected even if it is terminated after a few minutes, instead of 900s.

The results of our computational experiments show that each component of GOAS_I2PM has a significantly positive influence on its performance. As expected, the assignment of the orders to the time segments given by ASSIGN converges to the assignments found in the optimal solution. Also, SCHEDULE can find the optimal

Table 3.10: Comparison of GOAS_I2PM with the state-of-the-art algorithms SSP and SPARROW for $n = 50$.

			Gap (%)									Solution time (s)	
n	τ	R	SSP			SPARROW*			GOAS_I2PM			SSP	GOAS_I2PM
			Min	Avg	Max	Min	Avg	Max	Min	Avg	Max		
50	0.1	0.1	1.20	2.24	3.00	0.51	0.79	1.27	0.00	0.29	0.60	3415.80	900.00
		0.3	1.39	2.44	3.92	0.60	1.00	1.53	0.30	0.53	0.78	3600.00	900.00
		0.5	1.12	1.61	2.10	0.00	0.42	0.96	0.00	0.04	0.42	3217.10	900.00
		0.7	0.00	2.45	16.39	0.00	1.66	16.38	0.00	1.64	16.39	2661.80	900.00
		0.9	0.00	0.54	1.90	0.00	0.00	0.00	0.00	0.00	0.00	2139.00	900.00
	0.3	0.1	0.52	2.35	3.48	0.63	1.59	2.86	0.41	1.02	1.72	3600.00	900.00
		0.3	2.66	3.54	5.05	1.40	1.81	2.69	0.77	1.28	2.15	3600.00	900.00
		0.5	1.33	3.49	11.87	0.18	1.44	3.64	0.00	1.01	3.10	3600.00	900.00
		0.7	0.00	1.40	3.44	0.00	0.39	1.16	0.00	0.07	0.35	3312.10	900.00
		0.9	0.40	1.35	2.85	0.00	0.27	1.32	0.00	0.06	0.61	3250.10	900.00
	0.5	0.1	0.57	4.32	13.76	1.11	2.01	2.64	0.56	1.42	2.15	3558.70	900.00
		0.3	2.37	4.83	7.80	1.97	3.30	5.44	1.64	2.68	4.77	3017.60	900.00
		0.5	1.55	3.84	8.01	1.03	3.00	6.49	0.34	2.41	5.52	3600.00	900.00
		0.7	1.51	2.91	5.16	0.37	1.97	4.14	0.00	1.43	3.31	3600.00	900.00
		0.9	0.35	2.25	4.20	0.00	1.94	4.04	0.00	1.31	3.64	3484.60	900.00
	0.7	0.1	2.42	3.93	5.20	2.24	3.88	4.74	1.67	3.44	4.45	3600.00	900.00
		0.3	2.50	4.27	7.60	2.75	4.82	8.40	2.48	4.17	7.42	3160.50	900.00
		0.5	4.55	6.16	10.32	4.74	6.46	10.91	4.55	6.07	10.32	3356.50	900.00
		0.7	1.03	6.55	14.31	1.73	7.13	15.50	1.19	6.56	14.31	1971.70	900.00
		0.9	2.52	7.16	13.00	3.26	7.63	13.02	2.52	7.16	13.00	1374.30	900.00
	0.9	0.1	6.55	10.88	16.77	6.55	10.98	16.77	6.55	10.88	16.77	304.60	900.00
		0.3	9.28	13.25	17.40	9.28	13.43	17.80	9.28	13.27	17.60	15.70	900.00
		0.5	2.63	12.15	16.52	3.05	12.31	16.78	2.63	12.15	16.52	76.10	900.00
		0.7	5.62	11.13	17.64	5.62	11.50	18.42	5.62	11.21	17.64	54.00	900.00
		0.9	9.26	12.09	15.85	9.74	12.49	16.30	9.26	12.09	15.85	238.00	900.00
Avg:			2.454	5.085	9.101	2.270	4.489	7.728	1.991	4.087	7.175	2552.33	900.00

* He et al. (2019) do not share the run time of SPARROW. They state that it is under a minute on average for the instances with 50 orders.

solutions for each time segment very quickly. Although ASSIGN feeds SCHEDULE with a proper assignment of orders to the time segments, the optimal solution may not be found immediately since time segments are tackled sequentially. Therefore, the tabu search algorithm GOAS_MSGTS is vital to obtain the optimal solutions in such cases. Although the VNS algorithm performs well for minimizing total weighted tardiness with sequence dependent setup times (Kirlik and Oğuz, 2012), embedding it into a TS algorithm can be computationally ineffective. However, using granular neighborhoods enables the GOAS_MSGTS algorithm to cope with the corresponding inefficiency and, in turn, improves its performance. The role of the PL approach is also fundamental on the performance of GOAS_I2PM. To use elite solutions to diversify the search has been applied successfully by Glover et al. (2010) and Laguna et al. (1999). However, the PL approach outperforms other diversification mechanisms regarding the GOAS problem. The main advantage of the PL approach is to generate new sequences gradually unlike others that change the sequences instantly,

Table 3.11: Comparison of GOAS_I2PM with the state-of-the-art algorithms SSP and SPARROW for $n = 100$.

			Gap (%)									Solution time (s)	
n	τ	R	SSP			SPARROW*			GOAS_I2PM			SSP	GOAS_I2PM
			Min	Avg	Max	Min	Avg	Max	Min	Avg	Max		
100	0.1	0.1	1.18	2.44	3.28	0.39	0.57	0.80	0.09	0.24	0.43	3600.00	900.00
		0.3	1.51	2.10	3.06	0.34	0.61	0.88	0.09	0.26	0.53	3600.00	900.00
		0.5	0.36	1.39	2.74	0.00	0.13	0.28	0.00	0.00	0.00	3423.00	900.00
		0.7	0.00	0.43	0.80	0.00	0.00	0.00	0.00	0.00	0.00	2243.60	900.00
		0.9	0.00	0.22	0.89	0.00	0.00	0.00	0.00	0.00	0.00	2354.80	900.00
	0.3	0.1	1.47	2.65	3.96	0.53	0.99	1.44	0.20	0.58	1.26	3600.00	900.00
		0.3	1.94	2.94	5.39	0.57	1.24	2.49	0.37	0.77	1.93	3600.00	900.00
		0.5	1.14	2.21	3.81	0.65	1.07	1.75	0.36	0.62	1.02	3600.00	900.00
		0.7	0.37	1.65	2.75	0.00	0.30	0.79	0.00	0.08	0.36	3515.40	900.00
		0.9	0.19	0.91	2.41	0.00	0.11	0.42	0.00	0.00	0.00	2788.30	900.00
	0.5	0.1	2.49	3.96	5.27	1.07	1.79	2.71	0.49	1.23	2.12	3600.00	900.00
		0.3	2.68	3.95	5.99	1.49	2.16	2.63	0.86	1.57	2.09	3600.00	900.00
		0.5	2.57	3.54	4.53	1.31	2.33	3.35	0.70	1.69	2.66	3600.00	900.00
		0.7	1.61	2.71	4.04	0.35	1.49	2.38	0.00	0.82	1.80	3600.00	900.00
		0.9	0.67	2.10	4.65	0.19	0.92	2.40	0.00	0.41	1.79	3600.00	900.00
	0.7	0.1	2.70	4.98	6.42	1.69	2.47	3.21	1.27	1.95	2.74	3600.00	900.00
		0.3	3.93	6.66	10.83	1.35	3.66	5.80	0.97	3.04	5.17	3600.00	900.00
		0.5	4.04	6.45	12.85	2.23	4.34	10.65	1.60	3.78	10.03	3600.00	900.00
		0.7	3.36	7.36	13.20	1.92	4.81	7.33	1.60	4.28	6.72	3600.00	900.00
		0.9	5.30	8.40	12.95	2.97	5.07	6.55	2.62	4.52	6.05	3600.00	900.00
	0.9	0.1	4.59	8.70	11.95	3.44	5.46	8.47	3.45	5.27	8.29	3600.00	900.00
		0.3	5.68	12.45	17.41	4.90	8.65	11.20	4.53	8.25	10.77	3600.00	900.00
		0.5	10.80	15.15	18.38	7.90	10.76	15.67	7.48	10.58	15.52	3600.00	900.00
		0.7	10.70	14.97	19.95	7.38	10.73	13.02	7.03	10.53	12.57	3600.00	900.00
		0.9	11.23	14.95	20.47	3.32	10.67	16.53	3.36	10.54	16.32	3600.00	900.00
Avg:			3.221	5.330	7.920	1.760	3.213	4.830	1.482	2.841	4.407	3453.00	900.00

* He et al. (2019) do not share the run time of SPARROW. They state that it is under a minute on average for the instances with 100 orders.

which results in poor diversification in the presence of setup times. Trial runs also show that the PL approach performs better than the diversification mechanisms for multi-start tabu search by James et al. (2009) and continuous diversification by Taillard (1993), too.

3.4 Discussion

Main purpose of this chapter is to develop new mathematical models, that are flexible and strong, and a matheuristic algorithm so as to increase the number of optimally solved instances and obtain near optimal solutions of the instances for which optimal solutions cannot be obtained.

Considering the instances with less than 50 orders, both MILP and CP models can find more optimal solutions than the models developed by Oğuz et al. (2010) and Silva et al. (2018). More importantly, these models do not compromise on their flexibilities to have their successful performances. In other words, both models can

be easily adapted to the variants / extensions of the GOAS problem as you can see in the next chapter.

Although the proposed MILP model performs better than the model of Oğuz et al. (2010) for the instances with 50 and 100 orders, the model of Silva et al. (2018) performs better than the proposed MILP model when there are 50 instances. On the other hand, the model of Silva et al. (2018) cannot even solve the linear relaxation of the instances with 100 instances and cannot produce upper bounds, accordingly. Therefore, the MILP model is still capable of producing reasonable upper bounds for the corresponding instances albeit without finding near optimal solutions (it should be noted that it can still find optimal or near optimal solutions for the instances with loose time windows). While the MILP model consumes significant portion of its running time to solve the linear relaxations of the nodes on its search tree, the CP model aggressively looks for a feasible solution and improves it throughout its search. Thus, the significant increase in the difficulty of solving linear relaxations as the number orders increase does not directly affect the performance of the CP model. Consequently, it has a more robust performance than the MILP model with respect to the solution quality it achieves. However, it can take considerable time for the CP model to reach a satisfactory solution. In this respect, the proposed matheuristic GOAS_I2PM can conduct a very efficient search so that it obtains better solutions than the CP model in much less time.

GOAS_I2PM outperforms the state-of-the-art-algorithms regardless of the type of the problem instances while its run time can be higher. On the other hand, as the convergence graphs show, GOAS_I2PM can remain competitive even if it is terminated after a few minutes. Preliminary runs also show that it can still generate reasonable solutions if it is limited with single iteration (i.e., if it is not an iterative heuristic anymore). This observation signifies that GOAS_RTBM can provide a plausible assignment of orders to the segments, even before incorporating setup time estimations into it, so that a good solution, of which vicinity is worth to search for, can be obtained at the end of SCHEDULE. The aggressive and fast, due to its granular neighorhoods, search for the local optimum of GOAS_MSGTS

and the path relinking approach developed for the diversification enable to reach a near optimal solution at the end of the first iteration of GOAS_MSGTS. However, to increase the number of optimally solved instances, GOAS_MSGTS is allowed to be run for 900s for each instance. It can prove the optimality of the solutions for the instances which cannot be solved to the optimality by the mathematical models but the upper bound is equal to the objective function value of the incumbent solution found by GOAS_MSGTS.

Chapter 4

A NEW HEURISTIC APPROACH AND ITS APPLICATION TO THE GENERALIZED ORDER ACCEPTANCE AND SCHEDULING PROBLEM WITH BATCH DELIVERY

Scheduling problems integrating the decision of either order acceptance or batch delivery are widely studied as stated in Chapter 2. However, studies addressing these two decisions integrated in scheduling problems are highly limited, although both problems are observed in real-life practices and often simultaneously. While Chen (2004) emphasizes the lack of studies concerning integrated decisions at the detailed job-scheduling level, studies that consider integrated problems at job-scheduling level may disregard some crucial elements of the problems for the sake of convenience. In this respect, the majority of scheduling studies view setup times as negligible (Allahverdi et al., 1999). This praxis can be observed in the limited studies for the order acceptance and scheduling problem with batch delivery too. Among these limited studies stated in Chapter 2, only Khalili et al. (2016) consider the setup times which escalates the complexity of the scheduling problems significantly. As aforementioned, the DP algorithms are commonly used for the scheduling problems with batching decisions (Potts and Kovalyov, 2000) since they can take advantage of the property that sequencing and batching decisions can be decoupled in many of the corresponding problems. However, the decoupling property is rarely observed in the presence of setup times. Accordingly, most of the algorithms ignoring setup times, such as Jiang et al. (2017), cannot be applied to the scheduling problems when setup times are included.

To fill the gap of solution methodologies for the scheduling problems with batch

delivery and sequence dependent setup times, we extend the GOAS problem, by including the batch delivery decisions, to the GOASB problem. Khalili et al. (2016) do not consider release times, due dates or deadlines. Thus, to the best of our knowledge, the GOASB problem is the most generalized problem that consider both order acceptance, scheduling and batching decisions.

In the GOASB problem, unlike the individual deliveries in the GOAS problem, orders are delivered in batches to the customers (it is assumed that there are infinite number of uncapacitated vehicles for delivery). Each batch can only contain the orders belonging to a single customer and there is a fixed cost fc of delivering a batch. When all orders in a batch b, $b \in B$, are completed, batch b can be delivered. Therefore, delivery time of order i is equal to the delivery time of batch b_i that includes order i, C'_{b_i}, and computed as $\max_{j \in O: b_j = b_i} C_j$ where C_j is the completion time of order j, $j \in O$. We assume direct delivery such that each vehicle carries a single batch and directly goes to the associated customer. The transportation times between the manufacturer and the customers are assumed to be zero without loss of generality. The set B comprises of the disjoint sets B_q, $q \in Q$, such that the batches in set B_q can be used only for customer q. Since the manufacturer incurs a tardiness penalty cost for each tardy period if an order is delivered beyond its due date, tardiness of order i, T_i, is equal to $max\{0, C'_{b_i} - d_i\}$.

Given a sequence σ_s of the selected orders $S \subseteq O$ and the number of batches (in another sense, deliveries) of corresponding sequence, revenue generated from order i, $R_i(\sigma_s)$, is calculated as $R_i(\sigma_s) = max\{0, e_i - T_i w_i\}$. Consequently, the total revenue gained from processing orders in S in the sequence σ_s is $TR(\sigma_s) = \sum_{i \in S} R_i(\sigma_s)$ and net revenue is $TR(\sigma_s) - (\text{number of batches}) fc$. The GOASB problem is to find the set S, the sequence σ_s and its batching configuration when net revenue is maximum.

In this chapter, the models developed for the GOAS problem are adapted to the GOASB problem. To tackle large problem instances in which these models fail, we propose a new heuristic approach with alternating objective functions which enables to steer the search to a diversified but promising region following the generation of a new local optimum. By using the proposed approach, we develop an iterated local

search (ILS) algorithm for the GOASB problem (GOASB_ILS$^{\text{VNSA}}$) that employ VNS algorithms with different objective functions as its local search mechanism. We also develop an iterated local search algorithm using a TS algorithm for the local search, GOASB_ILS$^{\text{GTS}}$, for the purpose of comparison. Computational results show that the new heuristic approach can achieve better optimality gaps than the mathematical models in much shorter time and than GOASB_ILS$^{\text{GTS}}$ in comparable time.

The remainder of this chapter is organized as follows. Section 4.1 presents the models AMILP and ACP, that are adaptation / extension of the MILP and the CP models to the GOASB problem, while Section 4.2 describes the proposed heuristic approach, its application to the GOASB problem and GOASB_ILS$^{\text{GTS}}$. Computational results are given in Section 4.3 and the discussion of this chapter is provided in Section 4.4.

4.1 Two mathematical models

4.1.1 Mixed integer linear programming model

The AMILP model is generated by incorporating the batching decisions to the MILP model. Thus, we define the following variables regarding batching of the orders:

$\bar{z}_b$: 1 if batch b is used, 0 otherwise; $b \in B$.

$z_{i,b}$: 1 if order i is in batch b, 0 otherwise; $i \in O, b \in B_{q_i}$.

C'_b: Delivery time of batch b; $b \in B$.

T_i: Tardiness of order i; $i \in O$.

It should be noted that variable $T_i, i \in O$, is used to denote tardiness instead of $T_{i,j}, i, j \in O$ as in the MILP model to reduce the size of the AMILP model. Then

the AMILP model is given as follows:

$$\text{Maximize} \quad \sum_{i \in O}(e_i l_i - T_i w_i) - \sum_{b \in B} \bar{z}_b fc \tag{4.1}$$

s.t.

Constraints (3.2), (3.3), (3.4), (3.5), (3.6)

$$\sum_{b \in B_{q_i}} z_{i,b} = l_i \qquad \forall i \in O \tag{4.2}$$

$$z_{i,b} \leq \bar{z}_b \qquad \forall i \in O, b \in B_{q_i} \tag{4.3}$$

$$C'_b \geq \sum_{j \in O' \setminus \{n+1\}} g_{j,i} - \bar{d}_i(1 - z_{i,b}) \qquad \forall i \in O, b \in B_{q_i} \tag{4.4}$$

$$C'_b \leq z_{i,b}\bar{d}_i + (1 - z_{i,b}) \max_{q_i = q_j, j \in O} \bar{d}_j \qquad \forall i \in O, b \in B_{q_i} \tag{4.5}$$

$$T_i \geq C'_b - z_{i,b} d_i - (1 - z_{i,b}) \max_{q_i = q_j, j \in O} \bar{d}_j \qquad \forall i \in O, b \in B_{q_i} \tag{4.6}$$

$$\bar{z}_b \leq \bar{z}_{b+1} \qquad \forall q \in Q, b \text{ and } b+1 \in B_q \tag{4.7}$$

$$C'_b \geq \sum_{b' < b, b' \in B_{q_i}} z_{i,b'} d_i \qquad \forall i \in O, b \in B_{q_i} \tag{4.8}$$

$$T_i, g_{i,j}, C'_b \in R^+,\ l_i, y_{i,j}, \bar{z}_b, z_{i,b} \in \{0,1\} \qquad \forall i \in O', j \in O', b \in B_{q_i} \tag{4.9}$$

Objective function is simply the net revenue that is equal to gross revenue minus tardiness and delivery cost. Constraints inherited from the MILP model presented in Section 3.1.1 ensure that this model does not generate subtours or infeasible schedule with respect to the release time and deadline of the orders. If an order is accepted, it has to be contained in a batch by Constraint (4.2) and this batch is set as 'a used batch' by Constraint (4.3). Constraint(4.4) ensures that a batch cannot be delivered before the completion of an order in it while Constraint (4.5) guarantees orders to be delivered before their deadlines. Since Constraints (4.4)-(4.5) guarantee that an order cannot be completed beyond its deadline, Constraint (3.6) can be excluded from this model. However, we prefer to keep it to strengthen the model. Constraint (4.6) computes tardiness of the orders. Both Constraint

(4.7) and Constraint (4.8) are used to break symmetries pertaining to batches and since a different batch set is defined for each customer, these constraints handle each corresponding set independently. A batch b in set B_q, $q \in Q$, cannot be used before the following batch $b+1$ in the same set by Constraint (4.7). Therefore, given a certain number of batches used for a customer, Constraint (4.7) determines which specific batches to use. It is also aimed that if there are multiple batches that an order can be delivered with no tardiness, it should be delivered with the one having highest index among them. Accordingly, Constraint (4.8) ensures that if an order is contained in a batch b', the delivery time of other batches having higher index should not be earlier than the due date of the corresponding order because it should be contained in one of these batches having higher index than b', otherwise. Constraint (4.9) defines bounds and integrality of the variables.

4.1.2 Constraint programming model

Constraint programming is an appropriate approach to model the GOASB problem, mainly due to the notion of 'activity' and available specialized constraint to define subtour elimination constraint and sequence dependent setup times. In addition to these advantages, it also permits to use a decision variable as a subscript of another variable which enables to define batch related constraints more effectively. To generate the ACP model, we need to define new variables, apart from the activity and sequence variables in the CP model, for batching decisions and they are defined as follows:

C_i: Completion time of order i ; $i \in O$.

L_i: The latest order in the batch in which order i is contained; $i \in O$.

z_i: 1 if a batch is delivered immediately after the completion of order i, 0 otherwise; $i \in O$.

By defining the variable L_i, $i \in O$, the batch related constraints can be modelled efficiently. Although such a variable would result in non-linear constraints in a mixed

integer programming model, it can be easily introduced to the ACP model by using the variable L_i as the subscript of other variables. The GOASB problem can be modelled by constraint programming clearly as follows:

$$\text{Maximize} \quad \sum_{i \in O} presenceOf(A_i)(e_i - \max(0, endOf(A_i) - d_i)w_i - z_i fc) \quad (4.10)$$

s.t.

Constraints (3.10), (3.11)

$$C_i = endOf(A[i]) \qquad \forall i \in O \qquad (4.11)$$

$$C_{L_i} \geq C_i \qquad \forall i \in O \qquad (4.12)$$

$$z_{L_i} \geq presenceOf(A_i) \qquad \forall i \in O \qquad (4.13)$$

$$C_{L_i} \leq \bar{d}_i + (1 - presenceOf(A_i))(\max_{q_i = q_j, j \in O} \bar{d}_j - \bar{d}_i) \qquad \forall i \in O \qquad (4.14)$$

$$L_i \geq (1 - presenceOf(A_i) \max_{q_i = q_j, j \in O} j \qquad \forall i \in O \qquad (4.15)$$

$$C_i, L_i \in R^+, z_i \in \{0, 1\} \qquad \forall i \in O \qquad (4.16)$$

Objective (4.10) is the maximization of the net revenue. Constraints inherited from the CP model guarantee a feasible schedule in terms of the time windows of the orders. Constraint (4.11) is redundant but used for the purpose of clarification since $endOf(A_i)$ can be used in lieu of C_i in the rest of the model. For any order, the latest order of the batch it is sent with must be completed after it by Constraint (4.12) and z variable should be equal to 1 for the corresponding latest order by Constraint (4.13). Constraint (4.14) ensures that none of the orders are delivered after their deadlines. Constraint (4.15) is to break the symmetries such that unaccepted orders are assumed to be delivered with the order having highest index among the orders of its customer. Constraint (4.16) defines the bound and integrality constraints.

4.2 Iterated local search algorithm

The GOASB problem is NP-Hard as it can be reduced to a weighted tardiness problem which is also an NP-Hard problem (Lenstra et al., 1977) and the computational

results in Section 4.3 show that commercial solvers cannot handle the large size problem instances. Hence, to cope with such instances, an iterated local search algorithm for the GOASB problem (GOASB_ILS) is developed in this study.

ILS algorithm is known to be fast and applicable to different optimization problems successfully. Lourenço et al. (2010) show that the ILS algorithm can be effective in terms of solution time and provides good solutions for many scheduling problems. Therefore, ILS is preferred as the solution approach for the GOASB problem

Algorithm 4.1 Iterated local search algorithm

Input: x (initial solution)

repeat

$x^* \leftarrow LocalSearch(x)$

$x \leftarrow Perturbation(x^*)$

until Termination condition met

Output: x^* (local optimum)

An ILS algorithm, typically, comprises of two phases: Local search and perturbation. Basic structure of an ILS algorithm is given in Algorithm 4.1. We propose a new variable neighborhood search algorithm based on alternating objective functions (VNSA) to be used as the local search algorithm of the GOASB_ILS as explained in Section 4.2.1. To compare the performance of the proposed local search algorithm with, we develop a granular tabu search algorithm (GTS) to be also employed as the local search algorithm in GOASB_ILS and present it in Section 4.2.2. To avoid the confusion, GOASB_ILS using VNSA is denoted as GOASB_ILS$^{\text{VNSA}}$ while the one with GTS is denoted as GOASB_ILS$^{\text{GTS}}$; GOASB_ILS is the general denotation of the proposed algorithm regardless of the kind of the local search algorithm. The perturbation mechanism and the objective function value computation of GOASB_ILS are described in Section 4.2.3 and Section 4.2.4, respectively.

4.2.1 A new local search algorithm for GOASB_ILS

Due to the promising performance of the VNS algorithm in GOASB_MSGTS, a similar VNS algorithm is used as the local search algorithm of GOASB_ILS$^{\text{VNSA}}$. However, the proposed VNS in this section, VNSA, distinguishes from other VNS algorithms in the literature due to its cyclic mechanism with different objective functions.

A typical VNS algorithm starts running with an initial solution x and terminates upon the generation of a local optimal solution x^* such that none of its neighborhood solutions can improve it in terms of the function $f(x)$, the objective function of the relevant problem, value. In the proposed VNSA algorithm, we use two VNS algorithms, VNS$^{\text{f}}$ and VNS$^{\text{u}}$, in a cyclic manner so that the local optimum of a VNS would be the initial solution of the other and so on until a termination condition is met. VNS$^{\text{f}}$ and VNS$^{\text{u}}$ use the same VNS algorithm and only differ in the evaluation of the neighborhood solutions; they return the local optimal solution with regard to the original objective function $f(x)$ and an artificial objective function $u(x)$, respectively.

Artificial objective functions are often used in the heuristic algorithms. They can be used to guide the search to the feasible region (e.g., lagrangean relaxation heuristic or other heuristics using penalty terms concerning the feasibility on the objective function), to differentiate the weights of the solution features based on the search history as in the guided local search algorithm (Voudouris and Tsang, 1999), to develop constructive heuristics that use look ahead factors on the objective function (Schaerf, 1997) or for other similar purposes. It is also very common to exploit the optimal solution properties in designing ad hoc heuristics. However, to the best of our knowledge, artificial objective functions, that measure the quality of a solution in terms of the proved or presumed optimal solution properties instead of the original objective function, have not been used in heuristic algorithms. In the GOASB problem, two different functions, apart from the original objective function, can be used to measure the quality of solutions:

i): the average setup time

$$\sum_{i,j \in O} y_{i,j} s_{i,j} / \sum_{i \in O} l_i \tag{4.17}$$

ii): the number of batches multiplied by the average earliness of the orders

$$\sum_{b \in B} \bar{z}_k (\sum_{i \in O} (C'_{b_i} - C_i) / \sum_{i \in O} l_i) \tag{4.18}$$

It can judiciously be assumed that as the setup times observed in a solution x decrease, its objective function value $f(x)$ tends to increase. Thus, Objective (4.17) can be conceived as an indicator of the solution quality. Similarly, for two particular solutions that have the same number of batches, the one avoiding the orders to wait for the delivery too much after their completions can be preferred over the other since it may lead to less tardiness cost or it can be more promising in the following iterations of the algorithm. Accordingly, Objective (4.18) is an another solution quality indicator that highlights the delivery feature of the solutions. The ILS algorithms that use limited neighborhood search (i.e., not large neighborhood search) can jump to the perturbation phase very quickly that may results in missing improving solutions in the vicinity of the current local optimum. In this case, using artificial functions to evaluate the solutions can help to change the direction of the search, in the local search phase, while the neighborhood of the solution does not change. Changing the search direction cannot directly lead to a better solution since the neighborhood of the solution does not change. However, it can generate a promising solution in the corresponding neighborhood that is worth to search for the vicinity of it. For the corresponding purposes, it is vital to have artificial functions that are not so identical to the original objective function -since it would not change the search direction sufficiently otherwise- while being reasonable with respect to the solution properties that are expected to be observed in an optimal solution to keep the search direction in promising regions.

Considering the GOASB problem, when we employ VNS$^{\text{f}}$ and obtain a local maximum x^*_f, the best solution in the neighborhood of x^*_f with respect to an artificial objective function $u(x)$ (that can be either Objective (4.17) or Objective (4.18) for

the GOASB problem) which is provided by VNS^{u} can be used to initiate a new search of VNS^{f}. Depending on the structure of the artificial objective function $u(x)$, VNS^{u} can return unfavourable solutions. For instance, if Objective(4.17), that is the average setup time incurred before the accepted orders, is used as function $u(x)$ for the GOASB problem, VNS^{u} can return a solution with only one accepted order requiring small setup time. To avoid such cases, if VNS^{u} generates a solution $x \ni f(x) < \alpha f(x^*_f)$, $\alpha = 0.95$ for the GOASB problem, corresponding solution x is not taken into account. If a solution $x \ni f(x) > f(x^*)$ is generated over the course of VNS^{u} where x^* is the best solution found in the corresponding iteration of $\text{OAS\_ILS}^{\text{VNSA}}$, it can be stopped and solution x can be set as the initial solution of a new search of VNS^{f}. Otherwise, it will return the solution x^*_u that can be used to restart the search VNS^{f}. If x^*_u is already used as the initial solution of VNS^{f}, this cyclic process is terminated. For the GOASB problem, depending on the problem instance, either Objective (4.17) or Objective (4.18) can be used as the artificial objective function $u(x)$. Thus, it is proposed to use adaptive function $u(x)$ in VNSA. Let's assume that we have a set of candidate functions U to be used as function $u(x)$ such that $U = \{u^1(x), u^2(x), ..., u^{|U|}(x)\}$. For each artificial objective function a, $\forall a \in U$, a weight w'_a is assigned. In each iteration of the $\text{GOASB\_ILS}^{\text{VNSA}}$, one of the functions in set U is chosen to be the function $u(x)$ by using the roulette wheel selection and an artificial function a' is chosen with probability:

$$\frac{w'_{a'}}{\sum_{a \in U} w'_a}.$$

Weights of the artificial objective functions are updated over the course of VNSA by using the adaptive weight adjustment (Ropke and Pisinger, 2006) so that weights of the artificial objective functions would be higher as they enable to improve the local optimum. To this end, the entire search is divided into segments. A segment is a number of iterations of GOASB_ILS; in our problem, a segment consists of 10 iterations. A score of 0 is assigned to each artificial objective function at the

beginning of each segment. During a certain segment, when an artificial objective function achieves to improve the local optimum given by VNS^{f}, the score of the corresponding artificial objective function would increase by 1 and at the end of each segment, weight of an artificial objective function a is updated as follows:

$$w'_a = 0.5w'_a + 0.5\frac{\pi_a}{\theta_a}$$

where π_a denotes the score of the artificial objective function a and θ_a is the number of times we use it during the relevant segment. The general scheme of VNSA is presented in Algorithm 4.2.

Algorithm 4.2 Variable neighborhood search with alternating objectives (VNSA)

Input: x (initial solution), x^{best} (incumbent solution), *noimpr_iter*, set U

Initialise: $x^* \leftarrow x$:

Choose the function $u(x)$ from the set U by roulette wheel principle

while Solution x is not previously visited **do**

$x_f^* \leftarrow VNS^{\text{f}}(x)$

if $f(x_f^*) > f(x^*)$ **then** $x^* \leftarrow x_f^*$

$x_u^* \leftarrow VNS^{\text{u}}(x_f^*)$

$x \leftarrow x_u^*$

end while

if $f(x^*) > f(x^{best})$ **then** $x^{best} \leftarrow x^*, noimpr\_iter = 0$ **else** $noimpr\_iter++$

Output: x^*

The proposed algorithm VNSA can be viewed as analogous to the TS algorithm using VNS^{f} as the local search algorithm in several aspects (note that this analogy is not limited to VNSA and applies for any local search algorithm with the proposed alternating objectives). The TS algorithm jumps to the best non-tabu solution with respect to the original objective function to avoid a cycle whenever a local optimum is obtained. As the TS algorithm aggressively looks for local optimum, one of its main drawbacks is the lack of diversification (Pirim et al., 2008). In VNSA, cycles are

not allowed since it is terminated whenever the initial solution to VNS^{f} is previously set to be its initial in that iteration of $\text{GOASB\_ILS}^{\text{VNSA}}$. When a local optimum x_f^* is generated, instead of choosing the best non-tabu solution to restart VNS^{f}, the best solution returned by $VNS^{\text{u}}(x_f^*)$ is used to restart VNS^{f}. The main advantage of this approach is to enable faster diversification during the local search phase in contrast to the gradual diversification of a typical TS algorithm while the proposed approach requires more time in a single iteration. In this respect, it can be comparable with the tabu search algorithm with continous diversification (Taillard, 1993) that uses search history information to continuously diversify the search whereas the alternating objective functions facilitate to diversify the search independently of the search history and it redirects the search only when a local optimum is found with respect to $f(x)$. It also enables to reach the promising solutions that may not be reachable due to the tabu status of them by the TS algorithm. The TS algorithms often use the objective function value of the incumbent solution as the aspiration criteria and VNSA uses the same criteria as it stops VNS^{u} whenever x^* is improved and returns to VNS^{f}.

The VNS algorithm used in this chapter is almost identical to the one described in Algorithm 3.4 in terms of the types of the neighborhoods and their granularity. Considering the artificial objective functions, Algorithm 3.4 can be modified as in Algorithm 4.3.

Solution x is encoded similar to the explanation given in Section 3.2.2. However, for the GOASB problem, $x[1]$,.., $x[\bar{k}-1]$ are not called accepted orders but potentially accepted orders since rejecting some of them may be more lucrative depending on the delivery cost they incur. Determination of the accepted orders in solution x is explained in detail in Section 4.2.4.

Algorithm 4.3 Variable neighborhood search algorithm VNS^f and VNS^u

Input: x (initial solution), M (set of neighborhoods)

Initialize: m=1

$f(x)$: Original objective value of solution x

$u(x)$: Artificial objective value of solution x

if VNS^f **then**

$h(x) \leftarrow f(x)$

else

$h(x) \leftarrow u(x)$

end if

repeat

$x' \leftarrow argmin_{y \in N_m(x)} h(y)$

if $h(x') > h(x)$ **then**

$x \leftarrow x', m = 1$

else

$m++$

end if

until $m > |M|$

Output: x

4.2.2 A TS algorithm for GOASB_ILS

We propose a tabu search algorithm using VNS^f with the granular neighborhoods as the local search algorithm (GTS). If an order i is swapped or inserted in a VNS^f call, that order becomes tabu and cannot be swapped or inserted in the course of tabu tenure. Objective function value is used as the aspiration criterion. Since GTS is used within $GOASB\_ILS^{GTS}$, when it fails at improving the local optimum solution for a given number of iterations, it is restarted by generating a new solution via the perturbation mechanism of GOASB_ILS. General scheme of GTS is shown in Algorithm 4.4.

Algorithm 4.4 Granular tabu search algorithm (GTS)

Input: x (initial solution),x^{best} (incumbent solution), $PerturbationLimit$, $noimpr\_iter$

Initialise: $TabuList = \emptyset$, $perturb\_iter = 0$, $x^* \leftarrow x$

repeat

$x \leftarrow VNS^f(x)$

if $f(x) > f(x^{best})$ **then**

$x^* \leftarrow x, x^{best} \leftarrow x, perturb\_iter = 0, noimpr\_iter = 0$

else if $f(x) > f(x^*)$ **then**

$x^* \leftarrow x, perturb\_iter = 0, noimpr\_iter++$

else

$perturb\_iter++, noimpr\_iter++$

end if

$x \leftarrow$ best nontabu solution, update $TabuList$

until $perturb\_iter = PerturbationLimit$

Output: x^*

4.2.3 Perturbation mechanism of GOASB_ILS

The other component of the ILS algorithm apart from the local search is the perturbation that enables to jump to a different point in the search space so that the search algorithm does not get stuck at a local optimum. Any perturbation mechanism, mainly, should have two properties: i) it should diversify the solution adequately so that the algorithm does not return to the set of local optima recently found, and ii) it should not diversify too much which would be equivalent to a random search. In this respect, at the end of each iteration of GOASB_ILS, a solution $\hat{x}$ is perturbed by shifting a pre-given number, $\hat{N}$, of orders of it. An order of solution $\hat{x}$ is selected by a roulette wheel principle and shifted to a new location and this process is performed for $\hat{N}$ times so that in each time, location of a different order changes. To this end, a weight $\hat{w}_i$, initially 1 for each order $i \in O$, is assigned to the orders and order i is chosen to be shifted with the probability:

$$\frac{\sum_{j \in O} \hat{w}_j - \hat{w}_i}{\sum_{j \in O} \hat{w}_j}.$$

Weight of an order increases by 1 when it is shifted in the perturbation. The orders of $\hat{x}$ are partitioned into two sets, S_1 and S_2, in the perturbation phase. S_1 is the set of orders from $\hat{x}[1]$ to $\hat{x}[\bar{k}-1]$ and S_2 includes the remaining orders of $\hat{x}$. This partition signifies that set S_1 includes potentially accepted orders in $\hat{x}$ whereas set S_2 includes the unaccepted orders. Let $\hat{x}[k]$ be the order selected to be shifted. If $\hat{x}[k]$ belongs to set S_1, it is shifted to its best insertion position among all available positions from 1 to n except its current position k. Otherwise, it is shifted to its best insertion position among the positions from 1 to $|S_1|$ to force the selected order to be accepted. As the number of unaccepted orders increases, it becomes more critical to change the set of accepted orders to sufficiently perturb a solution and due to the roulette wheel principle and the described shifting rule, the proposed perturbation mechanism can achieve a sufficient alteration in the set of accepted solutions.

Solution $\hat{x}$ is set to the first solution in set E that has a pre-defined upper limit on its cardinality and stores the elite solutions found over the course of GOASB_ILS in a descending order of the objective function values. When a new local optimum is obtained, it can be added to set E only if it is better than at least one solution in set E with respect to the original objective function. To avoid set E having too similar solutions, parameter Δ is defined. If there are solutions in set E of which hamming distance to the new local optimum is less than Δ, the new local optimum is added to set E if it is better than the worst one among the corresponding solutions in set E with respect to the original objective value. When it is included in set E, the worst one is removed from the set. In this setting, $\hat{x}$ is always set to the incumbent solution. However, to exploit other elite -and diverse- solutions in set E, we keep track of how many times each solution in this set is designated to be solution $\hat{x}$. When a solution is designated to be perturbed more than UE times, it is removed from set E. Correspondingly, $\hat{x}$ is set to a different and diverse solution as OASB_ILS iterates. Algorithm 4.5 illustrates the perturbation mechanism.

Algorithm 4.5 Perturbation

Input: Weights $\hat{w}_i$, $i \in O$, set E

$\hat{x} \leftarrow$ First solution in set E

repeat

Select a random order i from solution $\hat{x}$ using the weights of the orders and let it be the kth order of solution $\hat{x}$

if $k < \bar{k}$ **then**

Insert order i to the best location among 1st,...,nth positions

else

Insert order i to the best location among 1st,...,$\bar{k}$-1st positions

end if

until $\hat{N}$ orders are selected

$x \leftarrow \hat{x}$

Output: x

Both GOASB_ILS$^{\text{VNSA}}$ and GOASB_ILS$^{\text{GTS}}$ use the initial solution generated by using the priority ratio, $e_i/(p_i + s_{previousjob,i}), i \in O$, in the constructive heuristic DynamicReleaseFirst-SequenceBest developed by Oğuz et al. (2010). Initially, none of the orders are scheduled and *previousjob* is 0. Then, the order i having the highest ratio among the unscheduled orders is set to be the next order in the schedule and *previousjob* is set to i and so on until all orders are assigned. GOASB_ILS$^{\text{VNSA}}$ and GOASB_ILS$^{\text{GTS}}$ are terminated after successive *TerminationLimit_VNSA* and *TerminationLimit_GTS*, respectively, iterations during which the incumbent solution is not improved. Algorithm 4.6 shows the outline of GOASB_ILS.

Algorithm 4.6 Proposed iterated local search algorithm GOASB_ILS

Input: x (initial solution), $TerminationLimit\_VNSA$, $TerminationLimit\_GTS$, set E

Initialise: $x^{best} \leftarrow x$, $noimpr\_iter = 0$

if GOASB_ILS$^{\text{VNSA}}$ **then**

$LocalSearch(x) \leftarrow VNSA(x)$

$TerminationLimit = TerminationLimit\_VNSA$

else

$LocalSearch(x) \leftarrow GTS(x)$

$TerminationLimit = TerminationLimit\_GTS$

end if

repeat

$x^* \leftarrow LocalSearch(x)$

Update set E

$x \leftarrow Perturbation(x^*)$

until $noimpr\_iter = TerminationLimit$

Output: x^{best} (incumbent solution)

4.2.4 Objective function value computation

In the proposed solution encoding, we reckon that if there are k^* accepted orders in solution x, then they must be $x[1], x[2]..., x[k^*]$. Thus, k^* must be less than $\bar{k}$ since $x[\bar{k}]$ is the first order in solution x of which the completion time exceeds its deadline. On the other hand, all orders from $x[1]$ to $x[\bar{k}-1]$ are not necessarily accepted since the revenue of an order may not compensate the cost it incurs. Thus, each sub-solution of solution x, denoted by $s_{x,k}$ and starting from $x[1]$ and ending at $x[k], k = 1, ..., \bar{k}-1$, should be evaluated and the sub-solution that maximizes the net revenue (i.e., $k^* = \underset{k|k<\bar{k}}{\text{argmax}}\{f(s_{x,k})\}$) corresponds to the feasible schedule of accepted orders represented by solution x (i.e., $f(x) = f(s_{x,k^*})$).

Effective objective function value computation of sub-solutions is crucial to the performance of the proposed algorithm GOASB_ILS since it is likely to consume

a significant portion of the run time. The objective function value of $s_{x,k}$ is equal to the sum of the revenue of its orders minus their tardiness and delivery cost. Computation of the cost is straightforward if the delivery times of the orders are known. Yet, to know the delivery times, optimal batch configuration of the orders in $s_{x,k}$, that minimizes the tardiness plus delivery cost, should be found. It should be noted that the optimal batch configuration of $s_{x,k}$ can be determined for each customer independently since orders from different customers cannot be contained in the same batch. Herein, solution x is assumed, without loss of generality, to include orders from a single customer for the sake of simplicity. While it would be inefficient to compute the objective function value for each sub-solution, $s_{x,1}, ..., s_{x,\bar{k}-1}$, from scratch, the corresponding objective function values can be computed using a dynamic programming approach.

Proposition 1. *For each sub-solution $s_{x,k}$, a batch upon the completion of its last order must be delivered. Let $c(s_{x,k})$ be the total cost of the sub-solution $s_{x,k}$ and $l(s_{x,k})$ be the position of the last batch before the batch delivered at the end of this sub-solution (i.e., regarding the sub-solution $s_{x,k}$, the last batch is delivered upon the completion of $x[k]$ and the previous batch is delivered upon the completion of $x[l(s_{x,k})]$) in its optimal batch configuration. If $t(i,j)$ denotes the tardiness cost of $x[i]$ when it is delivered immediately after the completion of $x[j]$. Then,*

$$c(s_{x,k}) = \min_{k'<k, k' \geq l(s_{x,k-1})} \{c(s_{x,k'}) + \sum_{i=k'+1}^{k} t(i,k)\} + fc \quad \forall k \in \{2, .., n\} \tag{4.19}$$

Proof. By definition,

$$c(s_{x,k}) = \min_{k'<k} \{c(s_{x,k'}) + \sum_{i=k'+1}^{k} t(i,k)\} + fc \tag{4.20a}$$

$$= c(s_{x,l(s_{x,k})}) + \sum_{i=l(s_{x,k})+1}^{k} t(i,k) + fc \tag{4.20b}$$

$$\leq c(s_{x,k'}) + \sum_{i=k'+1}^{k} t(i,k) + fc \quad \forall k' \ni k' < k \tag{4.20c}$$

By subtracting $\sum_{i=l(s_{x,k})+1}^{k} t(i,k) + fc$ from both sides of the inequality (4.20c), it

can be rewritten as:

$$c(s_{x,l(s_{x,k})}) \leq c(s_{x,k'}) + \sum_{i=k'+1}^{l(s_{x,k})} t(i,k) \quad \forall k' \ni k' < l(s_{x,k}) \tag{4.21a}$$

$$\leq c(s_{x,k'}) + \sum_{i=k'+1}^{l(s_{x,k})} t(i,k+1) \quad \forall k' \ni k' < l(s_{x,k}) \tag{4.21b}$$

By inequality (4.21b), making $l(s_{x,k+1})$ equal to $l(s_{x,k})$ does not cost more than making it equal to $k' \ni k' < l(s_{x,k})$ which enables us to derive equality(4.19). □

Corollary 1. *By using equality (4.19), $f(x)$ can be found by Algorithm 4.7, starting from sub-solution $s_{x,1}$ and proceeding through $s_{x,\bar{k}-1}$ iteratively while finding $c(s_{x,k})$ and $l(s_{x,k})$ at each iteration k and using this information in subsequent iterations.*

The complexity of the algorithm $Objective(x)$, which assumes single customer, is $O(n^2)$. In case of multiple customers, since we can use the corresponding algorithm for each customer independently, the complexity of it becomes $O(Q_{max}^2)$ where Q_{max} is the highest number of orders belonging to a single customer. It should be noted that this complexity is only realized when all orders of the customer with the highest number of orders are accepted and they are sent in a single batch. Worst-case complexity is rarely observed and consequently, the objective function value computation can be handled more efficiently in practice.

Algorithm 4.7 *Objective*(x)

Input: x (current solution)

Initialize: $k = 1, f(x) = 0, tc = 0, l(s_{x,0}) = 0, c(s_{x,i}) = +inf \quad \forall i < \bar{k}$

repeat

$R(x,k) = \sum_{i=1}^{k} e_{x_i}$

for $i : k-1; i >= l(s_{x,k-1}); i--$ **do**

if $\bar{d}_{x[i+1]} < C_{x[k]}$ **then** STOP **end if**

$tc+ = t(x[i+1], k)$

if $c(s_{x,i}) + tc + fc < c(s_{x,k})$ **then**

$c(s_{x,k}) = c(s_{x,i}) + tc + f$ and $l(s_{x,k}) = i$

end if

end for

if $R(x,k) - c(s_{x,k}) > f(x)$ **then**

$f(x) = R(x,k) - c(s_{x,k})$

end if

$k++$

until $k = \bar{k}$

Output: $f(x)$

4.3 Computational experiments

In this section, we compare the computational performance of the proposed models, OASB_ILS$^{\text{VNSA}}$ and OASB_ILS$^{\text{GTS}}$. We used the benchmark instances generated by Cesaret et al. (2012) to test their performance. In the GOASB problem, the customer information of orders $q_i, \forall i \in O$, and the delivery cost fc are also required which are not provided for the GOAS problem. We assume that for all instances, each customer has 5 orders such that first 5 orders belong to the first customer, orders 6-10 belong to the second customer and so on. The delivery cost fc is equal to the average revenue of the orders, $\sum_{i \in O} e_i \backslash n$, for all instances. All of the runs pertaining to the proposed models and the metaheuristics are performed on a workstation with an

Intel Core i7 processor, 2.60 GHz speed, and 8 GB of RAM, through Visual Studio 2015 and ILOG CPLEX 12.9. We set a time limit of 3600s for all model runs.

The values of $|E|$, $\hat{N}$, Δ, UE, tabu tenure, *PerturbationLimit*, *TerminationLimit_VNSA* and *TerminationLimit_GTS* are determined by trying combinations of different values for each parameter preliminarily. Their final values are given in Table 4.1.

Table 4.1: The parameters required for OASB_ILS.

n	$\|E\|$	$\hat{N}$	Δ	UE	Tabu tenure	*PerturbationLimit*	*TerminationLimit_VNSA*	*TerminationLimit_GTS*
10	10	n/5	n/5	n/2	3	n	250	2500
15	10	n/5	n/5	n/2	3	n	250	2500
20	10	n/5	n/5	n/2	3	n	250	2500
25	10	n/5	n/5	n/2	3	n	250	2500
50	10	n/5	n/5	n/2	5	n	250	2500
100	10	n/5	n/5	n/2	5	n	250	2500

This study pursues to develop a heuristic that can find optimal / near optimal solutions quickly. Therefore, $TerminationLimit\_VNSA$ is determined so as to make GOASB_ILS$^{\text{VNSA}}$ reach a satisfactory solution in a short time. For the purpose of a fair comparison, $TerminationLimit\_TS$ is given such a value that the average running times of GOASB_ILS$^{\text{VNSA}}$ and GOASB_ILS$^{\text{GTS}}$ become comparable. Since the perturbation mechanism includes randomness, heuristics solve each instance 5 times and averages of them are used in this section to make more robust comparisons that alleviates the impact of the randomness on the solutions.

For each instance group, denoted by (n, τ, R), 10 instances are solved and the optimality gap of an instance by a method (an algorithm or a commercial solver) is computed as

$$\text{Gap} = \frac{UB_{instance} - LB_{instance}^{method}}{UB_{instance}}$$

where $LB_{instance}^{method}$ is the objective function value of the best solution found by the corresponding method for the corresponding instance and $UB_{instance}$ is the upper bound on the objective function value of the instance. $UB_{instance}$ is taken as the minimum of the upper bounds given by the AMILP and the ACP models. Then, in Tables 4.2 –

4.7, Gap(%) denotes the percentage of the optimality gap, i.e., Gap(%) = Gap × 100. Finally, Min, Avg and Max columns under the Gap(%) heading show the minimum, the average and the maximum optimality gaps, respectively, of 10 instances falling in an instance group (n, τ, R) in each row.

Table 4.2: Comparison of the proposed models, GOASB_ILS$^{\text{VNSA}}$ and GOASB_ILS$^{\text{GTS}}$ for n=10.

			Gap (%)												Solution time (s)			
n	τ	R	AMILP			ACP			GOASB_ILS$^{\text{GTS}}$			GOASB_ILS$^{\text{VNSA}}$			AMILP	ACP	GOASB_ILS$^{\text{GTS}}$	GOASB_ILS$^{\text{VNSA}}$
			Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max				
10	0.1	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.18	3.09	0.09	0.09
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.26	3.57	0.10	0.09
		0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.31	2.80	0.10	0.10
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.77	4.69	0.11	0.11
		0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.49	5.98	0.12	0.12
	0.3	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.36	2.74	0.07	0.06
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.27	3.52	0.07	0.07
		0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.39	3.84	0.08	0.07
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.72	4.75	0.10	0.10
		0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.54	4.24	0.09	0.09
	0.5	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.39	1.60	0.06	0.05
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.51	2.87	0.06	0.06
		0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.70	3.04	0.07	0.06
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.79	2.73	0.07	0.07
		0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	3.15	0.08	0.07
	0.7	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.33	1.86	0.04	0.03
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.37	2.34	0.05	0.03
		0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.41	1.52	0.05	0.05
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.46	1.74	0.05	0.04
		0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.62	2.48	0.06	0.05
	0.9	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.26	0.11	0.03	0.01
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.24	0.21	0.03	0.01
		0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.32	0.74	0.04	0.02
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.37	0.99	0.04	0.02
		0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.37	1.60	0.05	0.03
Avg:			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.54	2.65	0.07	0.06

Both the models and the metaheuristics can solve all smallest size instances with 10 orders to the optimality in a very short time. Similar performances for the models are observed as the number of orders increases to 15; they can solve all the corresponding instances to the optimality as shown in Table 4.3. On the other hand, GOASB_ILS$^{\text{VNSA}}$ and GOASB_ILS$^{\text{GTS}}$ can solve 244 and 230 instances, respectively, out of 250 to the optimality in each run for them and GOASB_ILS$^{\text{VNSA}}$ achieves smaller gaps than GOASB_ILS$^{\text{GTS}}$, accordingly, when the number of orders becomes 15. As τ value increases (i.e., the range of the release times gets larger

Table 4.3: Comparison of the proposed models, GOASB_ILS^{VNSA} and GOASB_ILS^{GTS} for n=15.

			Gap (%)												Solution time (s)			
n	τ	R	AMILP			ACP			GOASB_ILS^{GTS}			GOASB_ILS^{VNSA}			AMILP	ACP	GOASB_ILS^{GTS}	GOASB_ILS^{VNSA}
			Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max				
15	0.1	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.64	5.30	0.00	0.21	2.12	0.29	21.77	0.30	0.36
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.34	2.21	0.00	0.03	0.33	0.42	24.13	0.34	0.38
		0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.41	4.07	0.00	0.00	0.00	2.66	31.63	0.33	0.41
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.24	1.02	0.00	0.00	0.00	12.77	46.35	0.37	0.41
		0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.64	0.00	0.06	0.64	15.28	44.11	0.35	0.46
	0.3	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.44	0.00	0.00	0.00	1.96	29.11	0.25	0.25
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.53	0.00	0.00	0.00	5.80	40.99	0.28	0.30
		0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	12.22	50.99	0.29	0.35
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.38	0.00	0.00	0.00	18.33	56.32	0.32	0.35
		0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.49	0.00	0.00	0.00	52.29	84.79	0.33	0.37
	0.5	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	7.41	48.07	0.20	0.20
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	21.08	41.64	0.20	0.21
		0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.25	1.59	0.00	0.09	0.88	10.44	76.03	0.23	0.24
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	11.65	44.83	0.24	0.24
		0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.09	0.90	0.00	0.00	0.00	19.58	29.53	0.23	0.23
	0.7	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.24	3.71	6.04	0.15	0.14
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	3.88	8.95	0.15	0.15
		0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	8.76	7.77	0.15	0.14
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	7.73	12.18	0.15	0.14
		0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.08	0.81	10.38	13.85	0.18	0.20
	0.9	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.58	1.62	0.07	0.05
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.88	4.21	0.08	0.06
		0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.06	3.74	0.10	0.08
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.28	5.10	0.11	0.10
		0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	3.35	3.85	0.14	0.13
Avg:			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.089	0.703	0.000	0.020	0.201	9.43	29.50	0.22	0.24

and due dates get smaller), time windows of the orders get narrower which results in less number of accepted orders. Therefore, when τ is 0.9, the time windows of the orders get too narrow and distinct which makes determining the correct set of the accepted orders easier and the metaheuristics can reach the optimal solutions more frequently compared to the other instances. The impact of τ factor can also be observed in the instances with 20 orders as both metaheuristics can solve all instances to the optimality only when τ is equal to 0.9 as shown in Table 4.4. While all instances can be solved to the optimality by both the AMILP and the ACP model when the total number of orders is small, starting from the instances with 20 orders, the corresponding models fail at solving some of the instances. Consequently, since a model runs for 3600s unless it can prove the optimality of a solution during the run, the solution time of the models does not remain competitive with that

of the metaheuristics when n is at least 20. While both GOASB_ILS$^{\text{VNSA}}$ and GOASB_ILS$^{\text{GTS}}$ reach near optimal solutions in 1 second on average for the instances with 20 orders, average solution time of the models can be dramatically higher for particular instances although they achieve smaller gaps regarding the entire instances with 20 orders. For example, when τ is 0.3, average solution time of both the AMILP and the ACP model gradually increases as R increases and reaches up to 2000s and 1500s, respectively. A similar behaviour, a direct proportion between the run times and R, is also observed for the instances with more orders. This is an expected outcome for the AMILP model since the range of the due dates -and the windows between due dates and deadlines- gets larger and the problem becomes less constrained, which is in favor of the mixed integer programming, as R value gets higher. However, the constraint programming is more applicable to tightly constrained problems and the ACP model is supposed to perform better as R increases. For the GOASB problem, as R value increases and the due dates get distinct, higher number of batches would be needed which increases the complexity of the batching decision in return and makes the performance of the ACP model worsen in the relevant instances .The performances of the metaheuristics, in terms of solution gaps, also deteriorate as R increases in many of the instances. However, it is difficult to make judgement of the relationship between R value and their performances since the optimality gaps are computed using the upper bounds given by the models and it is not clear whether the deteriorating performance of the metaheuristics stems from their poor performance itself or the loose upper bounds. Nonetheless, in the instances that the AMILP model provides the optimal solutions but the metaheuristics fail at reaching them (e.g., the instances with 20 orders for which the AMILP model achieves zero optimality gap), the metaheuristics do not perform worse as R value increases. Hence, it is difficult to claim that the relevant deterioration originates from the poor performance of the metaheuristics on the instances with high R value.

While the performance of the mathematical models improves when τ is high (i.e., when τ=0.9) as aforementioned, it is also noticeable, as in Table 4.5, that they also

Table 4.4: Comparison of the proposed models, GOASB_ILS$^{\text{VNSA}}$ and GOASB_ILS$^{\text{GTS}}$ for n=20.

			Gap (%)												Solution time (s)			
n	τ	R	AMILP			ACP			GOASB_ILS$^{\text{GTS}}$			GOASB_ILS$^{\text{VNSA}}$			AMILP	ACP	GOASB_ILS$^{\text{GTS}}$	GOASB_ILS$^{\text{VNSA}}$
			Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max				
20	0.1	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.84	1.74	0.00	0.20	0.58	1.31	135.97	0.82	0.99
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.95	0.00	0.50	1.24	6.85	166.08	0.80	1.07
		0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.76	2.68	0.00	0.32	1.29	116.05	256.99	0.92	1.15
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.82	1.86	0.00	0.34	1.78	1450.64	628.50	1.03	1.55
		0.9	0.00	0.25	1.35	0.00	0.00	0.00	0.00	0.64	2.05	0.00	0.23	0.56	1578.69	950.02	1.12	1.62
	0.3	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.25	0.74	1.54	0.00	0.17	0.95	15.68	263.73	0.72	0.72
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.71	3.36	0.00	0.46	3.36	82.68	307.53	0.72	0.86
		0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.78	4.30	0.00	0.31	1.49	138.49	349.88	0.83	0.91
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.53	1.33	0.00	0.34	1.87	1589.04	1069.81	0.91	1.09
		0.9	0.00	1.25	7.82	0.00	0.76	7.57	0.00	0.92	7.82	0.00	0.82	7.77	2194.21	1662.99	0.78	1.02
	0.5	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.10	0.62	0.00	0.00	0.00	205.65	773.22	0.56	0.58
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.60	2.72	0.00	0.41	2.72	388.82	1054.26	0.66	0.68
		0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.30	0.00	0.13	0.99	256.97	1780.38	0.54	0.61
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.12	0.65	0.00	0.10	0.39	449.16	1490.57	0.60	0.67
		0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.22	2.20	0.00	0.24	2.36	1389.96	740.28	0.55	0.60
	0.7	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.61	0.00	0.08	0.76	23.40	216.70	0.30	0.36
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.45	0.00	0.00	0.00	21.02	180.48	0.33	0.36
		0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	36.02	144.78	0.29	0.31
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.24	2.39	58.50	130.66	0.37	0.42
		0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.12	0.86	0.00	0.04	0.37	130.76	158.48	0.35	0.39
	0.9	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	4.21	13.02	0.18	0.16
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	6.02	12.70	0.18	0.16
		0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.45	28.17	0.20	0.20
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	15.72	36.23	0.23	0.21
		0.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	27.30	68.44	0.27	0.30
Avg:			0.000	0.060	0.367	0.000	0.030	0.303	0.010	0.361	1.482	0.000	0.197	1.234	407.90	504.79	0.57	0.68

perform well for the instances with small τ. When τ is 0.1, the due dates become larger and enable to accept the majority of the orders. Consequently, it becomes easier to decide which orders to reject since there would be a few rejections. Therefore, both the AMILP and the ACP model achieve smaller gaps for the instances with τ=0.1 or 0.9 than that of the instances having τ value in-between when n is 25. The AMILP model outperforms the ACP model and the metaheuristics with respect to solution quality in the instances having less than 50 orders. However, as n gets higher than 25, the optimality gaps of both models increase drastically so that GOASB_ILS$^{\text{VNSA}}$ outruns both the mathematical models and GOASB_ILS$^{\text{GTS}}$ in the corresponding instances. Nevertheless, the optimality gaps of the metaheuristics can get as high as 20% as in the instances with n=50, τ=0.9 and R=0.7. It is a contrasting outcome since better optimality gaps are expected for the instances with high τ value. However, the mathematical models cannot cope with the instances with high number of orders even if τ is high and possibly provide loose bounds

Table 4.5: Comparison of the proposed models, the GOASB_ILS$^{\text{VNSA}}$ and the GOASB_ILS$^{\text{GTS}}$ for n=25.

			Gap (%)												Solution time (s)			
n	τ	R	AMILP			ACP			GOASB_ILS$^{\text{GTS}}$			GOASB_ILS$^{\text{VNSA}}$			AMILP	ACP	GOASB_ILS$^{\text{GTS}}$	GOASB_ILS$^{\text{VNSA}}$
			Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max				
25	0.1	0.1	0.00	0.00	0.00	0.00	0.19	1.30	0.00	1.38	2.97	0.00	1.10	2.04	5.51	1719.32	1.44	0.85
		0.3	0.00	0.01	0.07	0.00	0.01	0.07	0.94	1.53	2.06	0.00	0.74	1.94	379.88	1245.09	1.52	1.04
		0.5	0.00	0.42	3.06	0.00	1.05	4.17	0.00	1.45	3.57	0.00	1.19	3.31	1730.20	2114.55	1.85	1.25
		0.7	0.00	0.76	3.80	0.00	0.82	3.80	0.00	1.47	3.80	0.00	1.15	3.80	2622.59	2570.47	1.74	1.26
		0.9	0.00	5.74	11.09	0.00	5.87	13.06	0.38	5.84	11.93	0.34	5.70	12.54	3367.79	3333.13	1.72	1.38
	0.3	0.1	0.00	0.00	0.00	0.00	0.76	4.02	0.53	1.20	2.62	0.00	0.70	1.45	124.88	3011.50	1.41	0.92
		0.3	0.00	0.37	2.60	0.00	0.87	2.76	0.00	1.48	4.72	0.00	1.10	3.25	1037.78	3393.12	1.57	0.92
		0.5	0.00	1.67	4.81	0.00	1.78	6.28	0.45	2.42	5.31	0.34	2.08	6.10	2491.03	2966.71	1.51	0.90
		0.7	0.00	2.41	10.01	0.00	4.55	9.53	0.00	2.65	9.62	0.00	2.78	8.41	2488.88	3508.46	1.59	1.12
		0.9	3.80	7.18	13.00	4.80	8.04	13.20	5.27	7.16	13.00	4.72	7.10	13.00	3600.00	3600.00	1.59	1.09
	0.5	0.1	0.00	0.00	0.00	0.00	1.24	4.81	0.00	0.99	2.03	0.00	0.95	3.39	1131.92	3269.75	1.13	0.68
		0.3	0.00	1.40	7.21	0.00	2.01	7.00	0.00	1.92	6.38	0.00	1.62	6.38	2105.36	3412.83	1.03	0.73
		0.5	0.00	3.79	13.56	0.00	4.58	12.29	0.00	3.57	9.76	0.00	3.49	9.65	2623.94	3600.00	1.13	0.72
		0.7	0.00	4.56	11.64	0.00	5.58	12.32	0.00	3.98	9.02	0.00	4.17	9.02	2923.22	3442.54	1.02	0.65
		0.9	0.00	7.56	11.02	0.00	8.22	13.06	0.00	7.36	10.86	0.00	7.48	11.86	3390.38	3600.00	1.04	0.77
	0.7	0.1	0.00	0.47	4.75	0.00	1.77	4.75	0.00	0.52	5.04	0.00	1.40	5.34	830.71	3218.65	0.68	0.41
		0.3	0.00	0.51	5.12	0.00	2.10	13.97	0.00	0.63	5.12	0.00	0.67	5.65	997.09	3490.48	0.70	0.38
		0.5	0.00	0.87	5.39	0.00	2.33	13.24	0.00	1.08	6.08	0.00	1.07	6.08	1563.21	2772.64	0.64	0.41
		0.7	0.00	1.56	8.01	0.00	1.78	8.01	0.00	1.52	8.01	0.00	1.58	8.01	1851.48	1999.44	0.68	0.45
		0.9	0.00	1.92	14.88	0.00	4.47	25.84	0.00	1.92	14.88	0.00	2.02	14.88	2007.93	1982.08	0.63	0.40
	0.9	0.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.17	1.69	26.36	52.63	0.32	0.14
		0.3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	65.51	114.03	0.30	0.16
		0.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.09	0.93	0.00	0.56	4.67	119.82	164.25	0.36	0.17
		0.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.09	0.91	0.00	0.33	1.34	170.67	173.78	0.42	0.21
		0.9	0.00	0.00	0.00	0.00	0.04	0.39	0.00	0.00	0.00	0.00	0.11	0.70	823.93	1007.33	0.43	0.22
Avg:			0.152	1.648	5.201	0.192	2.323	6.955	0.303	2.010	5.546	0.216	1.969	5.781	1539.20	2390.51	1.06	0.69

considering the high optimality gaps of them. Therefore, although the metaheuristics have high optimality gaps for many instances when n is 50 or 100, it may be more purposeful to evaluate them by considering the best solutions found by each approach, instead of the optimality gaps. To this end, Incumbent Gap values can be used as a performance measure as in Figure 4.1. Incumbent Gap of an instance is computed as follows:

$$\text{Incumbent Gap} = \frac{LB_{best} - LB_{instance}^{method}}{LB_{best}} * 100$$

where LB_{best} is the objective function value of the best solution found by any solution approaches for the corresponding instance. In Figure 4.1, for each solution approach, average Incumbent Gap values are delineated for each τ value; we do not provide a figure for each $\tau - R$ pair to avoid high number of figures. It should be noted that average Incumbent Gap values for the instances with 10 orders are not included in Figure 4.1 since all solution approaches can find the optimal solutions in that

case. If a solution approach finds the best solution among all approaches for all instances with a particular n value, then it would a straight line on the x-axis in the corresponding figure. Since, both the AMILP and the ACP models can find the optimal solution of each instance with 15 orders, lines of the both models lie on the x-axis in Figure 4.1a. Similarly, the ACP model can solve all instances with 20 orders to the optimality and gives the best solutions among all approaches, accordingly, as can be seen in Figure 4.1b. On the other hand, for the instances with 50 and 100 orders, GOASB_ILS$^{\text{VNSA}}$ finds the best solutions on average regardless of the τ values, as shown in Figures 4.1d-4.1e. Although there is at least one solution approach dominating over GOASB_ILS$^{\text{VNSA}}$ (e.g., the AMILP model achieves less Incumbent Gap for any τ value when n=25) for the instances with less than 50 orders, Incumbent Gap of GOASB_ILS$^{\text{VNSA}}$ does not exceed 1% in these instances. This indicates the competitiveness of GOASB_ILS$^{\text{VNSA}}$; same reasoning applies for GOASB_ILS$^{\text{GTS}}$ too. However, the AMILP model does not bear this competitive, and robust, performance as Figures 4.1d-4.1e clearly demonstrate. Particularly, increasing Incumbent Gap values of the AMILP model through the middle of the figures show that it cannot produce reasonable solutions for the large instances when τ is in $\{0.3, 0.5, 0.7\}$. On the other hand, both the ACP model and GOASB_ILS$^{\text{GTS}}$ show a similar, and robust, performance for the large instances. Specifically, they reach (or get stuck at) the same local optimum in the many instances with 100 orders while GOASB_ILS$^{\text{VNSA}}$ can find better local optimum in many of these instances. This can be interpreted as the outcome of the inherent diversification in the proposed heuristic approach that enables to escape from the local optima in which the ACP model and GOASB_ILS$^{\text{GTS}}$ get stuck.

While both GOASB_ILS$^{\text{VNSA}}$ and GOASB_ILS$^{\text{GTS}}$ can solve all instances with 10 orders to the optimality and yield zero optimality gaps, GOASB_ILS$^{\text{VNSA}}$ achieves smaller gaps for the larger size problems in a comparable time. Moreover, as the number of orders increases, the performance difference becomes more considerable. GTS is embed into GOASB_ILS to diversify the search space through the perturbation mechanism. Although it enables the diversification to a certain extent,

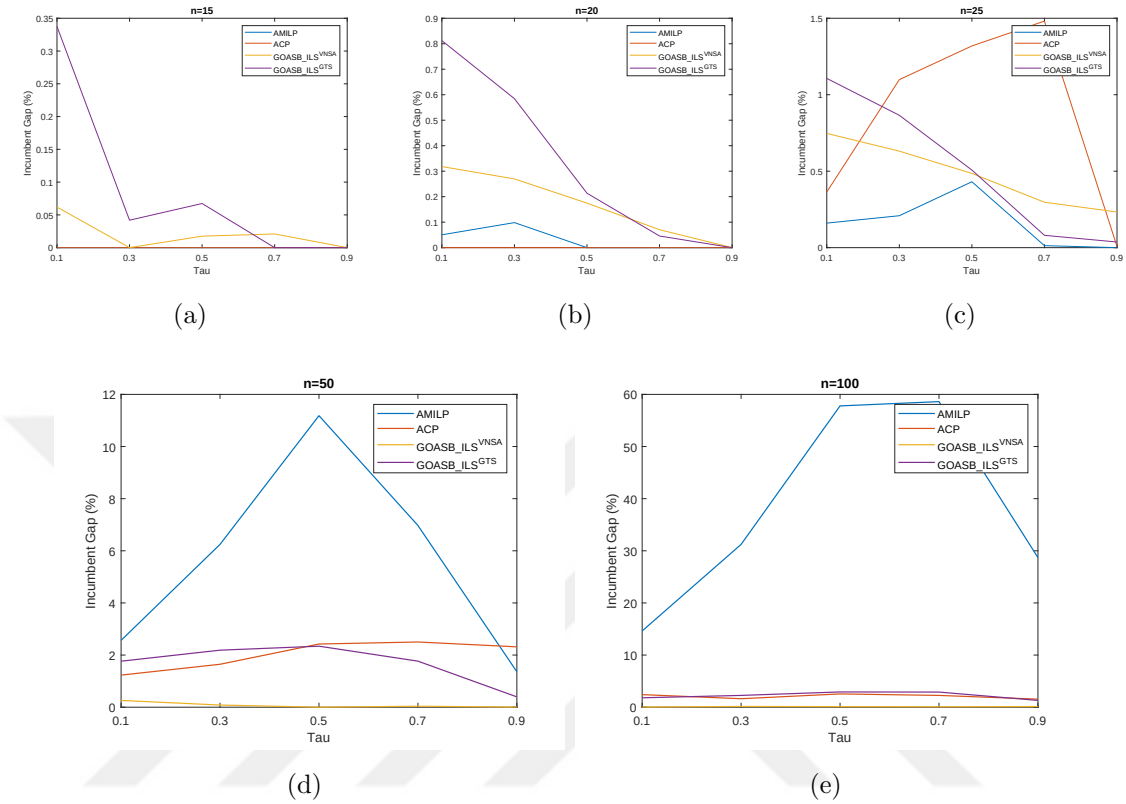

Figure 4.1: Incumbent gaps of the solution approaches for n=15, 20, 25, 50 and 100.

GOASB_ILS$^{\text{VNSA}}$ can continuously diversify and jump to different -but promising- search space more frequently than GOASB_ILS$^{\text{GTS}}$ by using alternating objective functions. As the search space expands in conjunction with the number of orders, it becomes more critical to be able to scan wider area, and increasing average optimality gap difference between GOASB_ILS$^{\text{VNSA}}$ and GOASB_ILS$^{\text{GTS}}$ in Tables 4.5 – 4.7 substantiates this remark.

As Table 4.7 shows, the AMILP cannot remain competitive when the number of orders rises to 100. It cannot find even a feasible solution for four of the instances with $\tau = 0.7$ and $R \in \{0.1, 0.3, 0.5\}$. Although the AMILP can find the optimal solution of 49 out of 50 instances with $\tau = 0.1$, $R = 0.1$, $n = 50$, it cannot find the optimal solution of any instances when n becomes 100. Moreover, GOASB_ILS$^{\text{VNSA}}$ and the ACP model can find better solutions than the AMILP model in nine and

six of the corresponding 10 instances, respectively. The ACP model can handle the instances with 100 orders better than GOASB_ILS^{GTS} unlike the ones with 25 or 50 orders. On the other hand, GOASB_ILS^{VNSA} can find, on average, better solutions than the ACP model in 218 of 250 instances with 100 orders that indicates how GOASB_ILS^{VNSA} prevails over the mathematical models as the number of orders increases.

Table 4.6: Comparison of the proposed models, the GOASB_ILS^{VNSA} and the GOASB_ILS^{GTS} for n=50.

			Gap (%)												Solution time (s)			
n	τ	R	AMILP			ACP			GOASB_ILS^{GTS}			GOASB_ILS^{VNSA}			AMILP	ACP	GOASB_ILS^{GTS}	GOASB_ILS^{VNSA}
			Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max				
50	0.1	0.1	0.00	0.02	0.24	0.22	0.82	2.09	1.27	2.09	2.77	0.66	0.99	1.46	877.15	3600.00	10.91	7.02
		0.3	1.23	3.16	5.20	1.31	3.12	4.91	2.57	3.91	6.06	1.76	2.71	4.53	3600.00	3600.00	12.13	8.23
		0.5	4.08	7.10	10.50	2.41	5.96	11.75	2.81	6.34	11.61	2.50	4.93	8.21	3600.00	3600.00	10.85	10.36
		0.7	4.95	10.52	17.42	5.05	8.34	12.22	4.83	8.51	10.92	3.39	6.68	9.72	3600.00	3600.00	13.61	11.78
		0.9	7.64	12.08	17.40	5.83	8.47	11.48	6.25	8.48	11.99	5.30	6.88	10.79	3600.00	3600.00	12.83	13.23
	0.3	0.1	0.00	4.41	7.62	0.27	3.22	5.07	2.46	4.86	6.73	0.32	3.27	4.78	3344.16	3600.00	9.26	5.44
		0.3	2.32	8.96	20.58	2.32	5.03	7.72	3.81	5.93	8.21	2.39	4.28	7.00	3600.00	3600.00	10.03	6.56
		0.5	6.03	10.82	19.23	4.04	7.67	10.41	5.97	8.01	10.11	4.04	6.23	8.25	3600.00	3600.00	10.88	8.52
		0.7	13.59	18.60	23.41	8.85	13.60	19.31	9.48	13.39	17.23	7.44	11.23	15.31	3600.00	3600.00	11.80	10.13
		0.9	12.52	20.90	33.01	9.51	13.25	16.52	10.26	13.24	17.07	8.20	10.72	13.78	3600.00	3600.00	10.58	9.94
	0.5	0.1	14.41	19.47	25.54	8.01	11.68	16.57	10.45	12.18	17.33	7.60	9.82	14.23	3600.00	3600.00	8.57	4.72
		0.3	11.05	20.10	27.96	6.76	12.37	15.36	7.35	12.54	15.78	6.33	10.50	13.34	3600.00	3600.00	8.21	4.70
		0.5	14.94	20.78	32.14	7.65	13.27	17.71	8.27	13.31	17.01	6.28	11.26	14.18	3600.00	3600.00	7.98	5.70
		0.7	13.49	20.20	26.48	8.73	13.72	17.17	8.72	12.98	17.85	6.37	11.19	15.54	3600.00	3600.00	8.37	6.09
		0.9	18.39	23.73	29.50	8.78	14.32	19.48	8.72	14.01	18.21	5.82	11.82	16.92	3600.00	3600.00	7.07	6.47
	0.7	0.1	3.45	12.34	20.51	5.81	11.39	17.98	5.76	10.88	18.34	3.85	8.87	15.00	3600.00	3600.00	5.12	2.80
		0.3	7.73	16.11	20.41	6.90	12.21	17.68	8.70	12.19	16.45	5.18	10.36	15.33	3600.00	3600.00	5.06	3.66
		0.5	15.47	18.79	23.82	11.83	16.13	22.62	12.60	15.53	19.41	10.65	14.14	18.94	3600.00	3600.00	4.82	2.91
		0.7	6.59	25.15	40.11	4.30	17.99	33.38	6.14	17.44	34.16	4.17	16.17	32.72	3600.00	3600.00	4.83	3.58
		0.9	18.13	24.25	28.73	13.78	20.05	26.87	12.69	18.66	23.68	11.48	17.59	23.14	3600.00	3600.00	4.07	3.29
	0.9	0.1	0.00	3.71	14.22	0.00	5.47	15.15	0.00	3.75	14.22	0.00	3.56	14.22	3600.00	3600.00	1.94	1.06
		0.3	0.00	7.52	19.73	3.32	8.65	21.66	0.94	7.64	19.73	0.00	7.26	19.73	3600.00	3600.00	2.04	1.02
		0.5	4.65	12.75	21.51	8.55	15.27	26.10	4.65	12.70	21.20	4.65	12.46	20.66	3600.00	3600.00	2.23	1.45
		0.7	9.67	22.02	31.12	6.61	22.16	37.60	5.71	20.63	30.47	5.13	20.22	29.81	3600.00	3600.00	2.51	1.74
		0.9	10.47	21.48	34.08	12.08	20.30	29.99	9.92	18.87	27.10	9.92	18.43	26.38	3600.00	3600.00	6.65	1.95
Avg:			8.032	14.599	22.019	6.116	11.379	17.472	6.413	11.124	16.545	4.937	9.662	14.960	3480.85	3600.00	7.69	5.69

Table 4.7: Comparison of the proposed models, the GOASB_ILS$^{\text{VNSA}}$ and the GOASB_ILS$^{\text{GTS}}$ for n=100.

			Gap (%)												Solution time (s)			
n	τ	R	AMILP			ACP			GOASB_ILS$^{\text{GTS}}$			GOASB_ILS$^{\text{VNSA}}$			AMILP	ACP	GOASB_ILS$^{\text{GTS}}$	GOASB_ILS$^{\text{VNSA}}$
			Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max				
100	0.1	0.1	2.55	4.12	8.09	2.08	3.49	4.58	3.14	4.22	4.77	1.98	3.11	3.80	3600.00	3600.00	129.54	93.92
		0.3	6.67	8.40	9.55	4.39	5.84	7.17	4.41	5.44	6.73	3.49	4.32	5.54	3600.00	3600.00	138.21	134.37
		0.5	13.65	18.34	23.71	5.89	8.40	10.00	6.03	7.63	9.60	4.11	5.73	8.22	3600.00	3600.00	154.83	176.73
		0.7	22.03	28.33	34.34	6.77	9.12	10.64	5.85	8.20	9.87	4.34	6.31	7.70	3600.00	3600.00	164.61	189.78
		0.9	30.08	35.70	44.83	7.73	10.91	14.59	5.65	9.45	12.42	4.10	7.06	10.21	3600.00	3600.00	180.21	203.59
	0.3	0.1	14.58	20.08	25.92	4.69	6.20	7.60	6.63	7.78	9.55	4.92	6.26	7.39	3600.00	3600.00	124.95	95.01
		0.3	22.49	26.47	28.51	5.77	7.71	11.01	7.27	8.63	10.20	5.49	7.21	9.23	3600.00	3600.00	137.98	104.54
		0.5	28.61	37.04	47.30	9.34	11.43	14.12	10.01	12.03	16.80	7.74	10.35	14.75	3600.00	3600.00	154.22	131.71
		0.7	39.11	46.24	57.84	11.61	13.41	15.33	11.49	13.41	15.31	9.69	11.14	12.61	3600.00	3600.00	153.15	151.24
		0.9	43.13	56.25	67.34	9.02	14.08	18.66	9.90	13.92	19.31	7.14	11.25	16.94	3600.00	3600.00	141.88	151.87
	0.5	0.1	48.20	58.78	67.33	7.25	12.50	15.12	9.55	14.15	16.65	8.52	11.89	14.52	3600.00	3600.00	107.30	78.22
		0.3	49.08	60.62	70.85	12.78	14.63	17.92	12.16	15.04	16.78	9.00	12.71	14.61	3600.00	3600.00	100.07	84.38
		0.5	54.49	64.02	76.54	10.66	15.78	19.58	13.55	15.97	18.65	10.55	13.40	16.21	3600.00	3600.00	99.69	85.34
		0.7	60.24	67.66	75.01	12.71	15.21	20.49	12.76	14.87	19.27	10.15	12.21	15.36	3600.00	3600.00	88.28	94.27
		0.9	49.99	63.18	80.36	7.97	13.14	15.65	9.19	12.89	16.88	6.69	10.33	14.50	3600.00	3600.00	89.54	92.45
	0.7	0.1	45.69	58.73	100.00	11.01	14.58	16.45	12.92	16.06	18.30	10.39	13.71	16.51	3600.00	3600.00	67.00	52.72
		0.3	47.16	66.96	100.00	11.70	16.13	20.30	14.79	17.34	20.89	11.85	15.13	18.27	3600.00	3600.00	72.78	50.25
		0.5	57.71	69.48	100.00	17.49	21.02	25.01	11.31	19.72	23.89	8.66	17.10	21.39	3600.00	3600.00	67.02	59.44
		0.7	46.78	68.01	79.44	14.48	21.60	27.86	13.01	22.43	29.59	11.12	19.86	26.67	3600.00	3600.00	54.88	55.11
		0.9	53.94	63.47	74.04	11.36	18.21	24.45	13.42	18.54	24.93	11.27	16.55	22.09	3600.00	3600.00	58.50	56.09
	0.9	0.1	19.66	31.63	49.45	10.79	22.98	35.14	9.25	22.07	35.41	9.11	21.47	34.29	3600.00	3600.00	30.88	20.40
		0.3	27.41	37.14	43.82	19.52	26.70	33.85	18.77	26.91	33.85	17.91	25.88	32.37	3600.00	3600.00	24.75	17.26
		0.5	37.07	48.24	62.99	16.60	26.24	31.56	18.31	26.43	32.64	17.35	25.39	31.37	3600.00	3600.00	27.98	25.10
		0.7	46.79	55.94	67.88	22.68	30.85	36.52	22.13	30.71	36.61	21.91	30.02	35.68	3600.00	3600.00	29.34	24.96
		0.9	52.48	62.45	79.36	24.92	30.02	36.32	23.57	29.67	35.43	22.29	28.75	35.00	3600.00	3600.00	35.24	29.74
Avg:			36.784	46.291	58.980	11.169	15.607	19.597	11.403	15.741	19.773	9.591	13.885	17.808	3600.00	3600.00	97.31	90.34

4.4 Discussion

Owing to the flexibility of both the MILP and the CP models, they are adapted to the AMILP and the ACP models, respectively, for the GOASB problem. However, unlike the MILP and the CP models, they can hardly handle the medium size problems (e.g., when n=20 and 25) due to their larger size in terms of both the decision variables and constraints. On the other hand, batching decisions can be incorporated to the CP model more easily and effectively than the MILP model due to the possibility of using the decision variables as subscripts. Thus, the ACP model can still produce reasonable solutions for the GOASB problem whereas the optimality

gap of the AMILP model can reach beyond the acceptable levels. More importantly, for four instances with 100 orders, the AMILP model fails in generating a feasible solution which is an unexpected result due to the properties of the GOASB problem. In terms of feasibility, the GOASB problem has a flexible structure so that we do not have to accept any order and there are infinite number of uncapacitated vehicles for delivery. Consequently, it is easy to find a feasible solution for this problem while finding the optimal solution poses a significant challenge. This outcome signifies that CPLEX solver is not efficient at producing feasible solutions in despite of employing heuristic algorithms aggressively to find a primal bound, especially at the root node. On the other hand, the CP model has a complementary property such that it can produce reasonable solutions even for the large size instances. However, computational results show that its performance still requires to be improved as GOASB_ILS$^{\text{VNSA}}$ finds a better solution than the CP model in almost all instances with 50 and 100 orders. This superior performance is mostly due to the continuous diversification inherent in GOASB_ILS$^{\text{VNSA}}$. This claim can be justified by making a comparison between GOASB_ILS$^{\text{VNSA}}$ and GOASB_ILS$^{\text{GTS}}$. Both algorithms have a similar structure (e.g., both use the same VNS algorithm in the local search phase and the same perturbation mechanism) and their run times are comparable. However, GOASB_ILS$^{\text{VNSA}}$ produces better solutions than GOASB_ILS$^{\text{GTS}}$ in the majority of the large size instances. As aforementioned, it is well known that tabu search can get stuck in a local optimum (Pirim et al., 2008) and this is observed more often when its local search algorithm is limited (i.e., not extensive). Thus, although it is embedded in an ILS algorithm, the tabu search algorithm in GOASB_ILS$^{\text{GTS}}$ fails in searching diversified regions in most of the instances. Therefore, the superior performance of GOASB_ILS$^{\text{VNSA}}$ encourages to maintain a continuous diversification in the absence of extended / sophisticated local search algorithms and the proposed heuristic approach is viable to achieve this diversification.

Chapter 5

CONCLUSION

In this thesis, we addressed two different scheduling problems: the GOAS problem and its extension with batch delivery, i.e., the GOASB problem. Although there are many heuristic algorithms for the GOAS problem in the literature, mathematical models for this problem are very limited. Particularly, there are two MILP models proposed by Oğuz et al. (2010) and Silva et al. (2018). We proposed a new MILP model and a CP model for the GOAS problem to fill the gap of the modelling efforts for this problem. It was shown that the travelling salesman problem formulation by Gavish and Graves (1978) can be altered for the GOAS problem by using the continuous variables to generate subtour elimination constraints and to handle time restrictions of the orders more effectively. Using the same alteration procedure, new mathematical models can be develop for both travelling salesman problems with time windows and prize collecting problems. Both proposed models perform better than the existing models in the literature. Additionally, they improve some of the best known solutions. However, as the number of orders increases, their performance deteriorates significantly. Therefore, a matheuristic algorithm that decomposes the original planning horizon into smaller segments was proposed to cope with the large scale problems. The proposed matheuristic GOAS_I2PM can be applied to any scheduling problem with long planning horizon. With the aim of having an effective and efficient solution method, we devised GOAS_I2PM in two phases, one for the assignment of the orders to the time segments and the other for solving the moderate size scheduling problem for each time segment. Assigning orders to the time segments optimally is a difficult problem by itself and solving the scheduling problem for each time segment sequentially does not guarantee the optimality even if the optimal assignment is provided. Thus, the proposed matheuristic GOAS_I2PM runs

iteratively, unlike the studies by Stauffer and Liebling (1997) and Elkamel et al. (1997), considering the implausibility of reaching the optimal solution in a single run. Consequently, each phase of the proposed matheuristic should not require high running times to have a viable solution methodology. To this end, we developed a new relaxed time-bucket formulation, where bucket sizes are not necessarily equal, to provide a proper assignment of orders to the time segments in a short time. This time-bucket formulation considers sequence dependent setup times implicitly so that estimation of the setup time observed before each order in the optimal solution, that is updated over the course of the matheuristic algorithm, is hold and estimated setup time is added to the processing time of the corresponding order in the relaxed time-bucket formulation. Although the relaxed time-bucket formulation can easily be adapted to include the sequence dependent setup times explicitly, proposed estimation approach enables to obtain a proper set of order assignments to the time segments in much shorter time. We exploited local search algorithms both for partial solutions generated through sequential solving of time segments and for the final solutions that are obtained at the end of each iteration. Computational results show that the proposed matheuristic outperforms both the proposed mathematical models and the state-of-the-art studies for the GOAS problem in most of the problem instances.

Owing to the flexibility of them, mathematical models proposed for the GOAS problem were adapted to the GOASB problem by incorporating new components regarding batch delivery decisions to these models. Although the adapted models can solve small size problem instances to the optimality, they cannot find satisfactory solutions in reasonable time as the number of orders increases. Therefore, we proposed a new heuristic approach that is based on alternating objective functions designated as a tool to escape from local optima and applied this approach to the GOASB problem within an iterated local search algorithm. This new approach enables to move through the promising but divergent search spaces more frequently, by using alternating objective functions in a cyclic manner, than the algorithms looking for the local optimum aggressively such as the tabu search algorithm. For the

purpose of comparison, an iterated local search algorithm that uses a granular tabu search algorithm for the local search, instead of the proposed heuristic approach, was developed and computational experiments showed that the proposed approach can reach better solutions than both the proposed models and the iterated local search algorithm with the granular tabu search algorithm as problem size enlarges. The proposed approach is capable to intensify and diversify the search simultaneously and to avoid performing large scale diversifications before examining the vicinity of the promising solutions.

Although both the matheuristic algorithm proposed for the GOAS problem and the new heuristic algorithm applied to the GOASB problem give satisfactory results for most of the instances, their optimality gap can be beyond reasonable for some of the instances with high number of orders. It is likely that this result stems from the loose upper bounds provided by the mathematical models considering much larger optimality gaps achieved by the mathematical models in the corresponding instances. Therefore, possibly weak upper bounds provided by the mathematical models for large scale problems necessitate further modelling studies for both the GOAS and GOASB problem.

Another future research direction is to apply the proposed matheuristic and heuristic approaches to different problems to show their broad applicability. Although we presented some insights such that the proposed matheuristic algorithm is promising for the scheduling problems with long planning horizon and the proposed heuristic approach can be useful to escape from local optimum, the proposed approaches must be tested on much more problems to evaluate the validity of the corresponding insights.

BIBLIOGRAPHY

Agnetis, A., Aloulou, M. A., and Fu, L.-L. (2014). Coordination of production and interstage batch delivery with outsourced distribution. *European Journal of Operational Research*, 238(1):130–142.

Ahmadizar, F. and Farhadi, S. (2015). Single-machine batch delivery scheduling with job release dates, due windows and earliness, tardiness, holding and delivery costs. *Computers and Operations Research*, 53:194–205.

Allahverdi, A., Gupta, J. N., and Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. *Omega*, 27(2):219–239.

Baptiste, P., Le Pape, C., and Nuijten, W. (2012). *Constraint-Based Scheduling: Applying Constraint Programming to Scheduling Problems*. Springer Science and Business Media.

Boland, N., Clement, R., and Waterer, H. (2013). A big bucket time indexed formulation for nonpreemptive single machine scheduling problems. *E-print 3773, Optimization Online*, http://www.optimization-online.org/.

Boland, N., Clement, R., and Waterer, H. (2016). A bucket indexed formulation for nonpreemptive single machine scheduling problems. *INFORMS Journal on Computing*, 28(1):14–30.

Cakici, E., Mason, S. J., Geismar, H. N., and Fowler, J. W. (2014). Scheduling parallel machines with single vehicle delivery. *Journal of Heuristics*, 20(5):511–537.

Cesaret, B., Oğuz, C., and Salman, F. S. (2012). A tabu search algorithm for order acceptance and scheduling. *Computers and Operations Research*, 39:1197–1205.

Charnsirisakskul, K., Griffin, P. M., and Keskinocak, P. (2004). Order selection and scheduling with leadtime flexibility. *IIE Transactions*, 36(7):194–205.

Chaurasia, S. N. and Singh, A. (2016). Hybrid evolutionary approaches for the single machine order acceptance and scheduling problem. *Applied Soft Computing*, 177:2033–2049.

Chen, Z.-L. (1996). Scheduling and common due date assignment with earliness-tardiness penalties and batch delivery costs. *European Journal of Operational Research*, 93(1):49–60.

Chen, Z. L. (2004). Integrated production and distribution operations: Taxonomy, models, and review. *International Series in Operations Research and Management Science*, 74:711–745.

Chen, Z. L. and Vairaktarakis, G. L. (2005). Integrated scheduling of production and distribution operations. *Management Science*, 51(4):614–628.

Cheng, T. and Gordon, V. (1994). Batch delivery scheduling on a single machine. *Journal of the Operational Research Society*, 45(10):1211–1216.

Cheng, T. C. E. and Kahlbacher, H. G. (1993). Scheduling with delivery and earliness penalties. *Asia-Pacific Journal of Operational Research*, 10:145–152.

Cheng, T. E., Gordon, V. S., and Kovalyov, M. Y. (1996). Single machine scheduling with batch deliveries. *European Journal of Operational Research*, 94(2):277–283.

Cheng, T. E., Kovalyov, M. Y., and Lin, B. M. (1997). Single machine scheduling to minimize batch delivery and job earliness penalties. *SIAM Journal on Optimization*, 7(2):547–559.

Cordone, R., Hosteins, P., and Righini, G. (2017). A branch-and-bound algorithm for the prize-collecting single-machine scheduling problem with deadlines and total tardiness minimization. *INFORMS Journal on Computing*, 30(1):168–180.

Elkamel, A., Zentner, M., Pekny, F., and Reklaitis, G. (1997). A decomposition heuristic for scheduling the general batch chemical plant. *Engineering Optimization*, 28(4):299–330.

Focacci, F., Lodi, A., and Milano, M. (2002). A hybrid exact algorithm for the TSPTW. *INFORMS Journal on Computing*, 14(4):403–417.

Fragkos, I., Degraeve, Z., and De Reyck, B. (2016). A horizon decomposition approach for the capacitated lot-sizing problem with setup times. *INFORMS Journal on Computing*, 28(3):465–482.

Gavish, B. and Graves, S. C. (1978). The travelling salesman problem and related problems. *Working paper GR-078-78*, Operations Research Center, Massachusetts Institute of Technology.

Geramipour, S., Moslehi, G., and Reisi-Nafchi, M. (2017). Maximizing the profit in customer's order acceptance and scheduling problem with weighted tardiness penalty. *Journal of the Operational Research Society*, 68(1):89–101.

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

Glover, F., Lü, Z., and Hao, J.-K. (2010). Diversification-driven tabu search for unconstrained binary quadratic problems. *4OR*, 8(3):239–253.

Gong, H., Tang, L., and Leung, J. Y. T. (2016). Parallel machine scheduling with batch deliveries to minimize total flow time and delivery cost. *Naval Research Logistics*, 63:492–502.

Graham, R. L., Lawler, E. L., Lenstra, J. K., and Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. In *Annals of Discrete Mathematics*, volume 5, pages 287–326. Elsevier.

Hall, N. G. and Potts, C. N. (2003). Supply chain scheduling: Batching and delivery. *Operations Research*, 51(4):566–584.

Hall, N. G. and Potts, C. N. (2005). The coordination of scheduling and batch deliveries. *Annals of Operations Research*, 135:41–64.

Hamidinia, A., Khakabimamaghani, S., Mazdeh, M. M., and Jafari, M. (2012). A genetic algorithm for minimizing total tardiness/earliness of weighted jobs in a batched delivery system. *Computers and Industrial Engineering*, 62:29–38.

He, L., Guijt, A., de Weerdt, M., Xing, L., and Yorke-Smith, N. (2019). Order acceptance and scheduling with sequence-dependent setup times: A new memetic algorithm and benchmark of the state of the art. *Computers and Industrial Engineering*, 138:106102.

James, T., Rego, C., and Glover, F. (2009). Multistart tabu search and diversification strategies for the quadratic assignment problem. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 39(3):579–596.

Ji, M., He, Y., and Cheng, T. E. (2007). Batch delivery scheduling with batch delivery cost on a single machine. *European Journal of Operational Research*, 176(2):745–755.

Jiang, D., Tan, J., and Li, B. (2017). Order acceptance and scheduling with batch delivery. *Computers and Industrial Engineering*, 107:100–104.

Kanet, J. J., Ahire, S. L., and Gorman, M. F. (2004). Constraint programming for scheduling. In Leung, J. Y. T., editor, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. CRC Press.

Karimi, N. and Davoudpour, H. (2017). A benders decomposition algorithm for multi-factory scheduling problem with batch delivery. *Scientia Iranica. Transaction E, Industrial Engineering*, 24(2):823.

Khalili, M., Esmailpour, M., and Naderi, B. (2016). The production-distribution problem with order acceptance and package delivery: Models and algorithm. *Manufacturing Review*, 3(18):194–205.

Kirlik, G. and Oğuz, C. (2012). A variable neighborhood search for minimizing total weighted tardiness with sequence dependent setup times on a single machine. *Computers and Operations Research*, 39:1506–1520.

Laguna, M., Marti, R., and Campos, V. (1999). Intensification and diversification with elite tabu search solutions for the linear ordering problem. *Computers and Operations Research*, 26(12):1217–1230.

Lee, C.-Y. and Chen, Z.-L. (2001). Machine scheduling with transportation considerations. *Journal of Scheduling*, 4(1):3–24.

Lenstra, J. K., Kan, A. R., and Brucker, P. (1977). Complexity of machine scheduling problems. In *Annals of Discrete Mathematics*, pages 343–362. Elsevier.

Leung, J. Y. T., editor (2004). *Handbook of Scheduling*. Chapman and Hall/CRC, Florida, 1 edition.

Lewis, H. F. and Slotnick, S. A. (2002). Multi-period job selection: Planning workloads to maximize profit. *Computers and Operations Research*, 29:1081–1098.

Liaw, C.-F. (2000). A hybrid genetic algorithm for the open shop scheduling problem. *European Journal of Operational Research*, 124(1):28–42.

Lin, S.-W. and Ying, K. (2013a). Increasing the total net revenue for single machine order acceptance and scheduling problems using an artificial bee colony algorithm. *Journal of the Operational Research Society*, 64(2):293–311.

Lin, S. W. and Ying, K. C. (2013b). Increasing the total net revenue for single machine order acceptance and scheduling problems using an artificial bee colony algorithm. *Journal of the Operational Research Society*, 64:293–311.

Lourenço, H. R., Martin, O., and Stützle, T. (2010). Iterated local search: Framework and applications. *International Series in Operations Research and Management Science*, 146:363–397.

Mazdeh, M. M., Sarhadi, M., and Hindi, K. S. (2007). A branch-and-bound algorithm for single-machine scheduling with batch delivery minimizing flow times and delivery costs. *European Journal of Operational Research*, 183(1):74–86.

Mazdeh, M. M., Shashaani, S., Ashouri, A., and Hindi, K. S. (2011). Single-machine batch scheduling minimizing weighted flow times and delivery costs. *Applied Mathematical Modelling*, 35(1):563–570.

Mestry, S., Damodaran, B., and Chen, C. S. (2011). A branch and price solution approach for order acceptance and capacity planning in make-to-order operations. *Computers and Operations Research*, 211(3):480–495.

Miller, C. E., Tucker, A. W., and Zemlin, R. A. (1960). Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, 7(4):326–329.

Nguyen, S. (2016). A learning and optimizing system for order acceptance and scheduling. *International Journal of Advanced Manufacturing Technology*, 86(5-8):2021–2036.

Nobibon, F. T. and Leus, R. (2011). Exact algorithms for a generalization of the order acceptance and scheduling problem in a single-machine environment. *Computers and Operations Research*, 38:367–378.

Noroozi, A., Mazdeh, M. M., Heydari, M., and Rasti-Barzoki, M. (2018). Coordinating order acceptance and integrated production-distribution scheduling with batch delivery considering third party logistics distribution. *Journal of Manufacturing Systems*, 46:29–45.

Öncan, T., Altınel, İ. K., and Laporte, G. (2009). A comparative analysis of several asymmetric traveling salesman problem formulations. *Computers and Operations Research*, 36(3):637–654.

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

Park, J., Nguyen, S., Zhang, M., and Johnston, M. (2013). Genetic programming for order acceptance and scheduling. In *2013 IEEE Congress on Evolutionary Computation*.

Pesant, G., Gendreau, M., Potvin, J.-Y., and Rousseau, J.-M. (1998). An exact constraint logic programming algorithm for the traveling salesman problem with time windows. *Transportation Science*, 32(1):12–29.

Pirim, H., Bayraktar, E., and Eksioglu, B. (2008). Tabu search: A comparative study. IntechOpen.

Potts, C. N. and Kovalyov, M. Y. (2000). Scheduling with batching: A review. *European Journal of Operational Research*, 120:228–249.

Pundoor, G. and Chen, Z.-L. (2005). Scheduling a production–distribution system to optimize the tradeoff between delivery tardiness and distribution cost. *Naval Research Logistics (NRL)*, 52(6):571–589.

Rasti-Barzoki, M. and Hejazi, S. R. (2013). Minimizing the weighted number of tardy jobs with due date assignment and capacity-constrained deliveries for multiple customers in supply chains. *European Journal of Operational Research*, 228(2):345–357.

Rasti-Barzoki, M., Noroozi, A., and Mahdavi Mazdeh, M. (2017). Coordinating order acceptance and batch delivery for an integrated supply chain scheduling. *International Journal of Engineering*, 30(5):700–709.

Riedler, M., Jatschka, T., Maschler, J., and Raidl, G. R. (2020). An iterative time-bucket refinement algorithm for a high-resolution resource-constrained project scheduling problem. *International Transactions in Operational Research*, 27:573–613.

Rom, W. O. and Slotnick, S. A. (2009). Order acceptance using genetic algorithms. *Computers and Operations Research*, 36(6):1758–1767.

Ropke, S. and Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472.

Ruiz, R. and Stutzle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal Operational Research*, 177(3):2033–2049.

Schaerf, A. (1997). Combining local search and look-ahead for scheduling and constraint satisfaction problems. In *IJCAI*, pages 1254–1259.

Schneider, M., Schwahn, F., and Vigo, D. (2017). Designing granular solution methods for routing problems with time windows. *European Journal of Operational Research*, 263(2):493–509.

Selvarajah, E. and Steiner, G. (2009). Approximation algorithms for the supplier's supply chain scheduling problem to minimize delivery and inventory holding costs. *Operations Research*, 57(2):426–438.

Shobrys, D. E. and White, D. C. (2002). Planning, scheduling and control systems: Why cannot they work together. *Computers and Chemical Engineering*, 26(2):711–745.

Silva, Y. L. T., Subramanian, A., and Pessoa, A. A. (2018). Exact and heuristic algorithms for order acceptance and scheduling with sequence-dependent setup times. *Computers and Operations Research*, 90:142–160.

Slotnick, S. A. (2011). Order acceptance and scheduling: A taxonomy and review. *European Journal of Operational Research*, 212:1–11.

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

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

Stauffer, L. and Liebling, T. M. (1997). Rolling horizon scheduling in a rolling-mill. *Annals of Operations Research*, 69:323–349.

Taillard, É. (1993). Parallel iterative search methods for vehicle routing problems. *Networks*, 23(8):661–673.

Toth, P. and Vigo, D. (2003). The granular tabu search and its application to the vehicle-routing problem. *INFORMS Journal on Computing*, 15(4):333–346.

Van Den Akker, J. M., Hurkens, C. A. J., and Savelsbergh, M. W. P. (2000). Time indexed formulations for machine scheduling problems: Column generation. *INFORMS Journal on Computing*, 12(2):111–124.

Voudouris, C. and Tsang, E. (1999). Guided local search and its application to the traveling salesman problem. *European Journal of Operational Research*, 113(2):469–499.

Wang, G. and Cheng, T. C. E. (2000). Parallel machine scheduling with batch delivery costs. *International Journal of Production Economics*, 68:177 – 183.

Wang, X. and Regan, A. C. (2009). On the convergence of a new time window discretization method for the traveling salesman problem with time window constraints. *Computers and Industrial Engineering*, 56(1):161–164.

Wester, F., Wijngaard, J., and Zijm, W. H. M. (1992). Order acceptance strategies in a production-to-order environment with setup times and due-dates. *International Journal of Production Research*, 30(6):1313–1326.

Xie, X. and Wang, X. (2016). An enhanced abc algorithm for single machine order acceptance and scheduling with class setups. *Applied Soft Computing*, 44:255–266.

Yin, Y., Cheng, T., Hsu, C.-J., and Wu, C.-C. (2013a). Single-machine batch delivery scheduling with an assignable common due window. *Omega*, 41(2):216–225.

Yin, Y., Cheng, T., Wu, C.-C., and Cheng, S.-R. (2013b). Single-machine common due-date scheduling with batch delivery costs and resource-dependent processing times. *International Journal of Production Research*, 51(17):5083–5099.

Zandieh, M. and Roumani, M. (2017). A biogeography-based optimization algorithm for order acceptance and scheduling. *Journal of Industrial and Production Engineering*, 34(4):312–321.

Zdrzałka, S. (1996). A sequencing problem with family setup times. *Discrete Applied Mathematics*, 66:161–183.

Appendix A

We can illustrate basic steps of GOAS_I2PM by implementing it on a small example. In this example, we are given six orders and their associated parameters are as follows:

Table A.1: Example parameters

i	r_i	p_i	d_i	$\bar{d}_i$	e_i	w_i
1	2	4	10	16	3	0.50
2	8	2	20	27	2	0.29
3	1	4	8	20	5	0.42
4	3	4	8	20	2	0.17
5	4	2	9	12	3	1.00
6	2	3	10	12	3	1.50

Following the time index generation in Algorithm 3.1 and the division of the planning horizon into segments, we obtain the planning horizon depicted in Figure A.1.

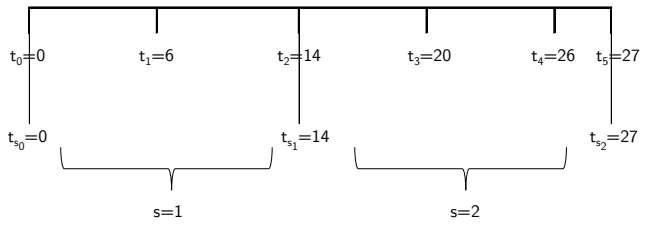

Figure A.1: Time indeces and segments on the planning horizon

GOAS_I2PM starts by solving the GOAS_RTBM model in ASSIGN to assign

orders to segment 0 (dummy segment to assign unaccepted orders), segment 1 or segment 2. After solving the relevant model, we obtain set $O_s, s \in \{0, 1, 2\}$, that is the orders assigned to segment s which are given at below.

$O_0 = 1$,

$O_1 = \{3, 5, 6\}$,

$O_2 = \{2, 4\}$.

Then, we can start SCHEDULE by solving the $MILP(O_s, r^s, \bar{d}^s)$ model for the first time segment, $s = 1$, which produces sequence $\pi_1 = \{3, 6\}$. This means that order 5 is unaccepted in the corresponding model although it is assigned to the first segment by ASSIGN. Partial solution x_1 is equal to π_1 at the end of first segment and its objective function value is 6.5.

After solving the first segment, the VNS algorithm can be applied which uses x'_1 as its initial solution. x'_1 is created by adding rejected orders to the end of x_1; $x'_1 = \{3, 6, 1, 5\}$. The VNS algorithm updates x_1 as $\{3, 1\}$ which means that order 6 is exchanged with order 1 and objective function value of the new partial solution is 7.

Once the VNS algorithm is employed for the first time segment, $MILP(O_2, r^2, \bar{d}^2)$ is solved which returns $\pi_2 = \{2\}$. Again, it means that order 4 is not accepted by the MILP model. Then, partial solution x_2 becomes $\{3, 1, 2\}$. At the end of the last segment, rejected orders are added to the partial solution and generates solution x which is $\{3, 1, 2, 4, 5, 6\}$.

After solving scheduling problem for all time segments, GOAS_MSGTS is employed to improve solution x. However, GOAS_MSGTS cannot improve solution x in this example since it is the optimal solution.

At the end of each iteration, we should generate cut(s) so that a new set of assignment is generated by ASSIGN in future iterations. Considering sets $A_1 = \{5\}$ and $A_2 = \{4\}$ which store the accepted orders by the MILP model of the corresponding time segments, Cut (23) can be generated for each time segment as follows:

$a_{5,1} + a_{3,1} + a_{6,1} \leq 2$ for the first time segment

$a_{4,2} + a_{2,2} \leq 1$ for the second time segment

These two cuts ensure that sets O_1 and O_2 cannot be assigned to the first and second segment, respectively, again. After adding these cuts to the GOAS_RTBM model and updating setup estimations st and modified processing times p', GOAS_I2PM moves to the next iteration and so on until termination condition is met.