VARIABLE NEIGHBORHOOD SEARCH FOR ORDER ACCEPTANCE AND SCHEDULING PROBLEM

by

Ayşegül Altındağ

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

Master of Science

in

Industrial Engineering

Koç University

August, 2013

Koç University

Graduate School of Sciences and Engineering

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

Ayşegül Altındağ

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. Dr. Ceyda Oğuz (Advisor)

_____

Assist. Prof. Dr. Onur Kaya

_____

Assoc. Prof. Dr. Kerem Bülbül

Date: _____

to democracy...

ABSTRACT

Order acceptance is one of the important decisions to make while dealing with satisfaction of customers, risk of delays and overloaded production in competitive environments. A company can increase its profit, satisfy demands of the customers and utilize its capacity at its best with a proper management of the incoming orders through making acceptance-rejection decisions on the orders and simultaneously scheduling the accepted orders. This problem is known as the order acceptance and scheduling (OAS) problem.

In this study, we examine two different OAS problems on a single machine environment. In the first problem, each order is characterized with a processing time, a due date, a weight and a revenue. Each accepted order which is delivered to the customer before its due date brings maximum profit to the manufacturer. Late delivery of an order causes tardiness cost which decreases the profit. The manufacturer can reject the order without a penalty cost. Sometimes customers may specify deadlines for their orders. Deadlines are the preferred latest time for the customers to accept the orders. If the completion time of an order exceeds the deadline, the customer refuses the order and does not pay for it. Moreover, some orders coming from the customers may be defined with release dates to be ready for the processing. In the first problem, we ignore sequence dependent setup times (preparation time necessary between two successive orders), deadlines and release dates. The second problem includes these properties. The objective function for both of the problems is to maximize total profit that is a function of total revenue and total tardiness.

We propose a Variable Neighborhood Search (VNS), which is a metaheuristic solution approach, to solve this NP-hard problem. The VNS is developed by using two neighborhood structures with a local search in a compact form. We analyze the performance of the VNS for both of the problems by using a benchmark data set. We present the computational experiments in which the VNS is compared with the most competitive metaheuristic algorithms from the literature. We conclude with the insights gained regarding the strengths and weaknesses of the proposed algorithm and that of the algorithms from the literature.

ÖZET

Günümüz rekabet ortamında, bir firma için sipariş kabul etme ya da reddetme kararı oldukça önem kazanmıştır. Bu problemin ortaya çıkmasının asıl nedeni, bir taraftan her firmanın belirli bir üretim kapasitesinin ve kıt kaynaklarının olması diğer taraftan da sipariş veren her müşterinin ilgili firmadan belirli bir beklentisinin olmasıdır. Bu nedenlerle, firma özellikle çok fazla sipariş aldığı zamanlarda, gelen siparişlerin bir kısmını reddetmek durumunda kalabilir. Bu noktada firmanın hangi siparişi kabul edeceğine ve kabul ettiği siparişleri nasıl çizelgeleyeceğine dair önemli bir karar vermesi gerekmektedir. Bu problem yazında sipariş kabul etme ve çizelgeleme problemi (SKEÇ) olarak bilinir.

Bu tezde, tek makine üzerinde iki farklı SKEÇ problemi ele alınmıştır. İlk problemde her siparişin teslim tarihi, işlem süresi ve getirisi vardır. Kabul edilen ve teslim zamanından önce tamamlanıp, müşteriye teslim edilen her sipariş üreticiye en büyük kazancı sağlar. Geç teslim edilen siparişler kazançta bir düşüş yaratır. Herhangi bir siparişin reddedilmesi mümkündür ve hiçbir ek maliyet getirmez. Bazen müşteriler siparişleri için son teslim tarihi belirleyebilirler. Son teslim tarihi bir siparişin kabul edilebilmesi için müşteri tarafından üreticiye verilen en son zamandır. Eğer bir sipariş son teslim tarihinden sonra müşteriye ulaştırılırsa, müşteri siparişi reddeder ve satın almaz. Ek olarak, bazı siparişler işlenmeye başlamaya hazır olmak için serbest bırakılma zamanlarına ihtiyaç duyabilirler. İlk incelenen problem bir siparişin son teslim tarihini, serbest bırakılma zamanını ve siparişler arasında sıraya bağlı hazırlık sürelesini yok sayarak problemi ele alırken bu özellikler ikinci problemde kapsanmıştır. Her iki problemin amaç fonksiyonu elde edilen kazancı en büyüklemektir.

Bu tezde, incelenen problemler için değişken komşuluklu arama (DKA) algoritması önerilmiştir. DKA algoritması etkili yerel arama yöntemi ile iki komşuluk yapısının kullanılması yoluyla geliştirilmiştir. Önerilen algoritmanın performansı, yazında bulunan sezgisel yöntemlerle kıyaslanmıştır. SKEÇ problemine uygulanan DKA algoritmasının yazında bulunan sezgisel yöntemlere göre güçlü ve zayıf yönleri verilmiştir.

ACKNOWLEDGMENTS

Foremost, I am heartily thankful to my advisor, Prof. Ceyda Oğuz, for her encouragement; guidance and support from the initial to the final level of this thesis. She provided valuable suggestions for believing in my abilities.

I would like to thank Assist. Prof. Onur Kaya and Assoc. Prof. Dr. Kerem Bülbül for taking part in my thesis committee, insightful comments and hard questions.

I offer my regards and blessings to my friends Aysu, Duygu and Güneş for their unforgettable, valuable,warm and enjoyable friendships during my master study at Koç University.

I cannot find words to express my gratitude to Serhan who gave me encouragement during the completion of this thesis with a big patience.

I would like to thank my sister Fatma Gül, my brother Onur and my cat Behlül, my best friends, for their help, advice and love during my life.

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

My special thanks are to my parents Emine and Ismet for their endless support and love in case of all events. To them I dedicate this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

ABC	Artificial Bee Colony
ATCS	Apparent Tardiness Costs with Setups
B&B	Branch and Bound
d-RFSB	Dynamic Release First Sequence Best
GA	Genetic Algorithm
GVNS	General Variable Neighborhood Search
ISFAN	Iterative Sequence First - Accept Next Algorithm
LP	Linear Programming
m-ATCS	Modified Apparent Tardiness Costs with Setups Heuristic
MILP	Mixed Integer Linear Programming
MTO	Make-to-order
OAS	Order Acceptance and Scheduling
$RLR1$	Revenue-Load-Ratio1
$RLR2$	Revenue-Load-Ratio2
RVNS	Reduced Variable Neighborhood Search
R&M	Rachamadagu and Morton
SVNS	Skewed Variable Neighborhood Search
SA	Simulated Annealing
TS	Tabu Search
TSP	Travelling Salesman Problem
UB	Best Upper Bound
UB_{LPVI}	Upper Bound obtained by LP Relaxation of MILP with Valid Inequalities
UB_{MILP}	MILP Upper Bound at Termination
VNDS	Variable Neighborhood Decomposition Search
VNS	Variable Neighborhood Search
WPST	Weighted Shortest Processing Time

Chapter 1

INTRODUCTION

In today's competitive environment, customers show great interest in more diversified and unique products. Furthermore, they expect on time delivery performance from manufacturers. As a result, manufacturers need to satisfy the expectations of the customers to survive in the competitive environment. Make-to-order (MTO) firms allow customers to customize products according to their special needs. Thus, MTO firms have the capability to produce custom-made products.

The MTO production system is faced with capacity management problems when their resources have limited flexibility and orders coming from customers have tight delivery requirements. The capacity management problems generally occur in a period of high demand for MTO production system. In such periods, if the firm accepts all prospective orders, it takes a significant risk in performing on-time delivery. On the other hand, refusing some orders because of concerns about capacity results in another risk about losing customers who can turn to other firms. Both of these situations result in loss of business and revenue. In these circumstances, the firm should develop a decision strategy which deals with which orders should be accepted and how these orders should be processed to use the capacity efficiently.

Order acceptance is one of the important decisions to make while dealing with some situations as customer satisfaction, delayed risk and production overload in competitive environments. These situations generally arise in different MTO production systems since they generally service customized products. A firm can increase its profit, satisfy the demands of the customers and utilize its capacity at its best with a proper management of the incoming orders through making acceptance-rejection decisions on the orders and simultaneously scheduling the accepted orders. This problem is known as the order acceptance and scheduling (OAS) problem. In this thesis, we analyze the OAS problem in a single machine

environment under two different scenarios. The first scenario deals with a set of independent orders characterized by due dates, maximum revenues, processing times and weights. The weight of an order is used to denote the importance of the customer. The maximum revenue of an order is gained if it is accepted and completed before its due date. Early delivery is not penalized but there is a discount for tardy delivery on behalf of the customer. All orders are available at time zero and more than one order cannot be processed on the machine at the same time. Scheduling problems can be modeled without sequence dependent setup times, when the assumption that the setup times are negligible in comparison to processing times is valid. However, for some industries this assumption is not valid and they require sequence dependent setup times when two orders are processed consequently. Some customers may also specify deadlines for their orders. Deadline is the latest time given by the customers to accept an order. If the completion time of an order exceeds its deadline, the customer refuses the order and does not pay for it. The release date may also be important in some systems. It is the time when an order arrives at the system. In the first scenario, we ignore sequence dependent setup times, deadlines and release dates (OAS-1). The second scenario involves the sequence dependent setup times, deadlines and release dates (OAS-2).

The three major contributions of this study are as follows:

- We propose a variable neighborhood search (VNS) algorithm to solve two OAS problems with different properties taking the success of VNS algorithm into account for difficult optimization problems. The VNS algorithm is not used as a solution method for the OAS problem in the literature so far. Although we observe the weaknesses of the VNS algorithm on some benchmark instances, solving two problems with the same method facilitates a robust algorithm.

- Using nested neighborhood structures serves searching the solution space more effectively. The VNS algorithm allows searching the solutions in different search spaces. Since this situation exploits the hidden information in the solutions, we find more optimal solutions in the VNS algorithm than the number of optimal solutions found in some competitive metaheuristics.

- Computational tests are utilized to select a local search procedure. We decide using a compound move including three operators: drop, add and insert as local search

procedure in this study due to the best results among the tests. In the local search, we utilize add operator both deterministically and probabilistically. In order to select the order to be added into the sequence, we use ratio values for the deterministic case while we employ cumulative probabilities in the probabilistic case. We claim that the compound move is more successful as a local search according to the test results since it is compatible with the properties of the problem and concerns the combination of algorithm with strong specialization in intensification and diversification.

The structure of the thesis is as follows. In Chapter 2, we survey the studies related with the OAS-1 and the OAS-2 problems. Moreover, we summarize the papers in which VNS is used as a solution method for their problems. In Chapter 3, we give the definition of two OAS problems formally and present the mixed integer linear programming (MILP) formulation of the problems which are developed in Oğuz et al. (2010) and Talla Nobibon and Leus (2011). In Chapter 4, we present the proposed VNS algorithm. We give the computational results for the study in Chapter 5. Finally, in Chapter 6, we give conclusions and remarks for future work.

Chapter 2

LITERATURE SURVEY

In this chapter, we survey studies related with the OAS problem. Furthermore, we point out the differences of our problems with the related studies. We also address the studies which use VNS algorithm in their analyses. We emphasize the success of the VNS algorithm and give the details of the algorithms for the related studies.

2.1 Order Acceptance and Scheduling problem

Studies on OAS can be differentiated in terms of their objective function, solution methods and problem characteristics. In this study, we investigate two different OAS problems. The first problem is order acceptance and scheduling problem without sequence setup times, release dates and deadlines. In the second problem, these properties are included in the problem. In the remaining parts of the thesis, the first problem is referred to as OAS-1 problem and the second one as OAS-2 problem. In this section, we investigate the most related papers with both of the problems.

2.1.1 OAS-1 problem

Slotnick and Morton (1995) investigate the order acceptance and scheduling problem in a single machine environment with static arrivals, that is when all jobs available at time zero. They assume that each order is associated with a weight, a processing time, a due date and a revenue. Their objective is to maximize the total net profit. The net profit is the revenue of each order minus the weighted lateness. Here, the lateness refers to the completion time minus the due date. Since completion time is a time dependent variable, they successfully present the objective of the problem with a time related penalty. They propose one optimal and two heuristic algorithms to handle the OAS problem. They use a branch and bound algorithm that provides bounds by a linear relaxation of the problem at each node to find optimal solutions. They initiate their algorithms with weighted shortest processing time

(WSPT) method. The WSPT method suggests to place jobs in a sequence according to the ratio of processing time to the weight. They sort the ratio values from the highest to the lowest and jobs are positioned with respect to these sorted values. Since they realize that the branch and bound algorithm has a disadvantage about searching solution space ineffectively, they alternatively develop heuristic algorithms to solve the problem. The beam search is one of the heuristic algorithms employed by Slotnick and Morton (1995). The beam search provides limitations on the generation of the search space with a predetermined number of nodes at each level in contrast with the branch and bound algorithm. The second proposed heuristic algorithm is the myopic algorithm. Following the currently most promising path within the search is the one of the distinctive property of the myopic algorithm. The other important property of the myopic algorithm is the generation of a smaller tree during the search. They measure the performance of the proposed algorithms on 400 test instances (100 each with $12, 17, 22$ and 27 orders). The optimal branch and bound algorithm is only used to obtain optimal solutions of 12 and 17 order problems. For remaining large sized test instances, branch and bound algorithm consumes too much time so they do not apply the tests for them. Their computational tests show that branch and bound algorithm is computationally very expensive to solve the OAS problem and solutions obtained from beam search and myopic algorithm are optimal or very close to the optimal. Furthermore, the beam search and myopic algorithm are much more effective in terms of computational time.

Ghosh (1997) proves that the problem investigated by Slotnick and Morton (1995) is NP-hard. Then, he develops two dynamic programming algorithms for the exact solution of job selection problem investigated by Slotnick and Morton (1995). The algorithms demonstrate that the problem is solvable in polynomial time if either the job processing times or the job weights for the lateness penalty are equal.

Slotnick and Morton (2007) extend their previous work and use the tardiness for the problem instead of lateness, where tardiness is the positive part of the lateness. They focus on the order acceptance decision with static arrivals, deterministic processing times, customer weights and revenues. The objective is to maximize the profit which is a function of the tardiness. Intuitively, when an order is delivered after its due date, the customer satisfaction level has a downward trend. The tardiness represents this decrease in the problem and depends on the completion time and the due date of an order. Slotnick

and Morton (2007) provide several solution methods, which are based on the decision of the sequencing and accepting orders jointly. In the first part of the study, beam search, Rachamadagu and Morton (R&M) dispatch heuristic and Montagnes method are developed and tested with 7- job problems. Branch and bound (B&B) algorithm is also used to obtain optimal solutions for small sized benchmark solutions. All three heuristics are shown to be not better than the B&B algorithm. Beam search and Montagnes heuristic perform similarly and the R&M heuristic gives the worst among three of them. Then, the performances of Montagne and R&M are tested with 50 jobs problem. Here, the beam search is used as the benchmark because it is not practical to find an optimal solution to problems for large sized problems with the B&B algorithm. The results indicate that Montagne performs as well as beam search and it is concluded that R&M is not an effective method. In the second part of their study, Slotnick and Morton (2007) develop new methods which allow to decide jointly accepting of the orders and sequencing them using relaxation. They observe that this new joint approach is really stronger than the first one. They develop myopic algorithm that performs a search with more intensive techniques for large sized problems in a fast way. The beam search procedure runs faster than the benchmark and give low average deviations from the optimal solutions. In addition, they develop a beam search by using Vogels method for bounding. This method provides more qualified solutions for small sized problems. For the larger ones, they observe that myopic heuristic definitely is faster than the benchmark and gives comparable results with the benchmark in terms of quality.

Rom and Slotnick (2009) present a genetic and myopic algorithm for the order acceptance and scheduling problem with tardiness penalties. Their study deals with static arrivals, deterministic processing times, customer weights and revenues associated with orders. Maximization of the total revenue is the objective function and includes the tardiness penalty. They propose a myopic algorithm as one of the solution methods in their study. Basically, idea of the developed myopic algorithm is to use the solution of the relaxed problem with reassembling of the accepted orders in the sequence. Reassembling of the accepted orders depends on inserting orders that are sequenced heuristically to minimize the weighted tardiness. The second proposed algorithm is the genetic algorithm. They model the chromosomes as sample solutions and use a two point crossover to generate the offspring. The crossover is based on switching the part of the sequence randomly to generate offspring.

They use the local search procedure to improve the solutions. The local search is employed by interchanging successively the positions of a limited number pairs of jobs randomly. They compare the performance of the genetic algorithm with the myopic heuristic, in terms of objective function value and computation time. They test the algorithms with the problems including 50, 75 and 100 jobs and obtain the upper bounds with relaxation of the original problem. They get really impressive results from the GA. The GA always dominates the myopic algorithm in terms of the objective function value, at the cost of increased CPU times.

Talla Nobibon and Leus (2011) approach the problem from a different angle. They handle the problem by assuming that a firm has both planned and optional orders. Optional orders can be accepted or rejected by the firm, but the planned orders must be processed in the system. All jobs are available for the processing at the beginning of the planning period and every order has a due date, a processing time, a revenue and a weight. The objective is the maximization of the total profit. The total profit is the summation of total revenue of the accepted orders minus the tardiness incurred for those jobs. First, they show that a constant factor approximation algorithm cannot be developed for this problem. Then, they model two different mixed integer linear formulations for the problem. Moreover, they develop two B&B algorithms to find optimal solutions. One of the B&B algorithms has two phases which work separately for selection and scheduling procedures. In contrast with the B&B algorithm, the other procedure integrates both selection and scheduling. According to the experimental results, the two-phase algorithm gives the best overall performance.

In the OAS problem literature, some researchers examine the OAS problems in a single machine environment with the orders are identified by their release dates, due dates, deadlines, processing times, sequence dependent setup times and revenues. The objective function of the problems can be different from each other as maximization of total profit, minimization of total weighted tardiness, and minimization of makespan. In the next section, we focus on the related studies about the OAS-2 problem.

2.1.2 OAS-2 problem

The setup times are defined as the required time before the start of processing an order or between processing of two orders. The setup times are not negligible in some practical

applications. Panwalkar et al. (1973) emphasize that nearly 75% of the managers face sequence dependent setup times at least in some operations in their production system.

Allahverdi et al. (1999) present a comprehensive review of the literature on scheduling problems with setup times. They give an effective classification for scheduling problems. They categorize problems according to some properties which are batch and non batch, sequence independent and sequence dependent setup. They define batch problems as the problems which include setup when jobs in different batches are swapped and a certain time is needed to start processing of the jobs. Properties of the sequence independent and dependent setup time are explained in the study as follows: sequence independent setup time depends on the current job, but sequence dependent setup time is related to both the current job and the previous job that is executed just before the current one.

Yang and Liao (1999) focus on static and deterministic scheduling problems concerning setup times. They classify the problems into four groups. The classes are inseparable and sequence independent setup times, inseparable and sequence dependent setup times, separable and sequence independent setup times, separable and sequence dependent setup times. Yang and Liao (1999) distinguish the seperable and inseperable setups as follows: inseparable setups which must be executed immediately before an activity starts whereas separable setups may be executed before earlier or later than an activity.

Oğuz et al. (2010) investigate order acceptance and scheduling problems in a single machine environment where orders are characterized by their release dates, due dates, deadlines, processing times, weights, sequence dependent setup times and revenues. Their objective is to maximize the total profit. Total profit is defined as a function of the revenue and the tardiness. The manufacturer obtains a revenue from an order, if the order is delivered before its deadline. Here, the deadline is the latest time given by a customer to accept the order. The customer rejects the order if it is delivered after its deadline, i.e, this order brings zero revenue. A penalty occurs if processing of an order is completed after its due date. Due date is the planned time for delivering the order to the customer. Intuitively, when an order is delivered after its due date, the customer gets disappointed and wants to pay less than the original revenue of an order. The manufacturer gives a discount to the customer in this case and the discount is related with the weight of the customer and the tardiness amount. Oğuz et al. (2010) give a mixed integer model for the problem. They test the performance of

the mixed integer programming model with randomly generated instances. In order to solve large sized problems, they propose heuristic algorithms. Upper bounds are obtained with linear relaxation method to measure the performance of the heuristic algorithms. In the paper, they present three heuristic algorithms: iterative sequence first-accept next (ISFAN) algorithm, dynamic release first-sequence best (d-RFSB) heuristic and modified apparent tardiness cost with setups (m-ATCS) heuristic. ISFAN is an iterative heuristic method that uses a priority rule to make acceptance and rejection decisions. In addition, a simulated annealing (SA) algorithm is designed to schedule the accepted orders. Major idea of d-RFSB is to check the feasibility of the orders according to their release dates and deadlines and to make acceptance and sequence decisions according to a ratio of the revenue to processing time and sequence dependent setup time. The last one, m-ATCS heuristic is very similar to d-RFSB. It also uses a ratio to schedule the orders, where average processing time and setup time together with due dates are used. They observe that the ISFAN algorithm is effective for small and medium sized instances and m-ATCS outperforms other heuristics for large sized instances.

Cesaret et al. (2012) study the same problem of Oğuz et al. (2010). They develop a tabu search (TS) algorithm to solve the problem. Swap is the selected move operator for TS. In the algorithm, a feasibility check procedure is required to pick and remove the non-profitable orders from the sequence. The tabu list keeps the most recently performed move to avoid cycling. A local search procedure is employed in the algorithm to find out improved results. The local search is based on drop-add-insert operations and is applied to the best solution found in the neighborhood structure. The performance of the TS algorithm is analyzed by comparing the results with that of m-ATCS and ISFAN heuristics developed in Oğuz et al. (2010). It is observed that the TS algorithm outperforms m-ATCS and ISFAN heuristics in terms of solution quality and computational time.

Ling and Ying (2013) propose a new algorithm based on artificial bee colony (ABC) for the problem investigated in Oğuz et al. (2010) and Cesaret et al. (2012). The ABC algorithm is initiated with a population which is a combination of randomly generated solutions. They use a permutation based representation for the solutions. The neighborhood solutions are obtained in two phases: destruction and construction. In the destruction phase, a random number of orders are deleted from the sequence. In contrast with the destruction phase,

the deleted orders are inserted again to the sequence in the construction phase. Reinsertion process is made step by step and all possible positions of the remained subsequence are tried for each order in deleted order set to generate a new solution. If the new solution is not better than the best solution so far, another new solution is generated with a different procedure. The procedure is a kind of crossover that uses a random substring in the best order. If this new solution is better than the best solution so far, the solution becomes a member of the population. A local search is applied to each generated solution. The local search exchanges the positions of successive orders in the current sequence. A probabilistic approach is employed in the ABC algorithm while selecting a solution from the population. After the local search, a random part of the population is modified by this probabilistic approach. To understand the performance of the ABC algorithm, comparisons are made with the benchmark problems in Oğuz et al. (2010) and Cesaret et al. (2012). They observe that the ABC algorithm outperforms the benchmarks with respect to the computational and analytical results.

2.2 Variable Neighborhood Search

Mladenovic and Hansen (1997) claim that metaheuristics, such as SA, TS and GA, avoid to get stuck in local optimum but these heuristics do not allow changing neighborhood structures. They believe that this reason can be an explanation for inefficient results. They focus on changing neighborhood structure systematically and develop a new metaheuristic approach. They call this approach as variable neighborhood search (VNS). The idea of VNS is to jump from one neighborhood to the next one if an improvement is not observed in the objective function value. Mladenovic and Hansen (1997) present a basic VNS algorithm including a local search procedure and test the performance of the VNS algorithm on traveling salesman problem. They observe that the VNS algorithm is effective and can be applicable for many combinatorial optimization problems.

Hansen and Mladenovic (2001) implement VNS algorithm on several classical combinatorial or global optimization problems (e.g. travelling salesman problem (TSP), continuous location-allocation problems). They observe that the basic VNS is definitely useful as a solution method for several problems but it is inefficient to solve large sized instances. They consider three extensions of the VNS to overcome this shortcoming. The first, reduced VNS

(RVNS) aims to reduce the most time consuming part of the basic VNS which is local search. The solutions are selected randomly from different neighborhood structures and updated if a better solution is obtained. The RVNS is very useful when a quick solution, which does not have to be close to optimum, is needed. Variable neighborhood decomposition search, VNDS, is the second proposed extension of the basic VNS. This method includes a successive approximation decomposition method. The difference between basic VNS and VNDS is that the local search method is not applied in VNDS within the whole solution space. VNDS provides solving a subproblem at each iteration. Skewed variable neighborhood search (SVNS) enhances exploration of far away valleys by enabling modification of objective function value of a solution with an evaluation function.

Liu and Zhou (2012) examine a single machine rescheduling problem of minimizing the maximum lateness under an expected arrival of new jobs. Major assumption for the problem is that the total sequence disruption cannot be larger than a given upper limit. Jobs have been scheduled optimally in original sequence with respect to the earliest due date first rule. New jobs arrive together after the optimal original scheduling. All jobs need to rescheduled after the arrival. The rescheduling objective is to find an optimal schedule to minimize the maximum lateness over all jobs. They adopt a variant of VNS algorithm to solve the problem and establish some dominance rules to apply the local search. Three neighborhood structures are defined: single job insertion(insertion), insertion of two consecutive jobs(Or_Opt) and interchanging of two random jobs with reversing order of jobs identified by them(2_Opt). If a solution obtained from the neighborhood structures is better than the incumbent solution, the local search is employed to the solution. It is based on interchanging adjacent or non adjacent jobs. They develop a branch and bound algorithm with the depth-first strategy for the problem to make a comparison with the VNS algorithm. They conduct some preliminary experiments to decide the order of neighborhood structures in order to have the best performance. According to the results gathered from the experiments, they choose one of the VNS algorithms with the order of neighborhood structures as insertion-Or_Opt and 2_Opt.

Kirlik and Oğuz (2012) propose a VNS algorithm to solve the single machine scheduling problem for minimizing total weighted tardiness with sequence dependent setup times. General variable neighborhood search (GVNS) is proposed to solve the problem. Different

neighborhoods and distributions, induced from different metrics are ranked and used to get random points in the shaking step in GVNS. Three neighborhood structures are employed swap, edge-insertion and insertion. The neighborhood structures are decided through computational experiments. The GVNS is tested with benchmark problems and its effectiveness is established.

Chapter 3

ORDER ACCEPTANCE AND SCHEDULING (OAS) PROBLEM

In this thesis, we study the order acceptance and scheduling problem in MTO systems. In MTO systems, the production is initiated when the order arrives to the system. We assume that there is a single machine. In the following sections, we state the properties and the assumptions of the problem with details.

3.1 Problem Definition

The limited capacity of the companies may force the manufacturers to make a selection among the orders coming from the customers. OAS problem is the combination of two joint decisions of which orders should be accepted and how these orders should be scheduled.

Moreover, the following assumptions are made for the problem.

- All orders are available at time zero.
- All input data are known in advance.
- Order arrival times are deterministic.
- More than one order cannot be processed on the machine at the same time.
- The machine is always available for the processing.
- Early delivery is not penalized, but there is a discount for tardiness delivery.
- When an order starts to be executed, it cannot be interrupted before its completion.
- There is no precedence relation among the orders.
- There is no limit on the number of accepted orders.

- Holding and shipping costs are not incurred in the problem.
- The manufacturer is not punished or awarded for a rejected order.

This study examines two OAS problems on a single machine environment: OAS-1 problem and OAS-2 problem. Next subsections explain the characteristics of them.

3.1.1 Definition for the OAS-1 problem

Each incoming order i is characterized with a processing time p_i, a due date d_i, a weight w_i, and a maximum revenue e_i. Processing time is the required time for an order to be executed on the machine. The date which the order is promised to the customer is represented with a due date. An order i can be delivered after its due date, but the order i becomes tardy in this case. Otherwise, when the tardiness of an order is zero, the manufacturer obtains the maximum revenue from the order. If the manufacturer faces tardiness, she needs to pay a penalty cost for each time unit beyond its due date. The penalty is constructed by incurring w_i which is related with the importance of the corresponded customer. Tardiness is observed if an order is delivered after its due date, i.e. $T_i = w_i \max\{0, (C_i - d_i)\}$, where C_i is the completion time of order i. The profit is a function of the revenue and the weighted tardiness. The objective is to maximize the total profit, i.e.

$$\max \sum_{i=1}^{n} I_i[e_i - w_i T_i]$$

where i is the index for orders $J_1, \ldots, J_n$, I_i is a binary variable to represent whether order i is accepted or not.

3.1.2 Definition for the OAS-2 problem

OAS-2 problem involves release dates, sequence dependent setup times and deadlines. The earliest time at when order i starts its processing is termed as release date of order i, r_i. Sequence-dependent setup time, s_{ji}, is used when order j is immediately executed after order i in the processing sequence. The setup times do not need to be symmetric which means that s_{ij} does not have to be equal to s_{ji}. In addition, if an order i is the first in the sequence, s_{0i} denotes the setup time for order i to start processing. The earliest completion time of an order i can be expressed as the summation of release date, sequence dependent setup time and processing time of an order i. If an order is completed after its deadline,

$\bar{d}_i$, the customer refuses the order and does not pay for the order. The customer pays a discounted amount for her order when she receives the order after exceeding its due date. Total profit generated from order i is calculated by considering the revenue and the tardiness of order i which is the same with the OAS-1 problem.

3.2 MILP Formulation for the OAS Problem

We present the MILP formulation for the OAS-1 problem and the OAS-2 problem in this section.

3.2.1 MILP Formulation for the OAS-1 problem

Talla Nobibon and Leus (2011) present an assignment formulation for the OAS problem. They consider the OAS problem with both planned and optional orders. We use their formulation as the MILP formulation of the OAS-1 problem with some modifications since only optional orders are available in the OAS-1 problem. Talla Nobibon and Leus (2011) define three binary decision values. First one, $y_i \epsilon \{0,1\} (i \epsilon N)$, where N is the set for the incoming orders, is denoted to demonstrate that whether order i is accepted or not. It becomes 1, when order i is accepted and otherwise get a value of 0. The second decision variable $x_{it} \epsilon \{0,1\} (i \epsilon N, t \epsilon \{1,\ldots,n\})$ equals to 1, if the order i is accepted and executed at the tth position in the sequence, and 0 otherwise. The last decision variable z_{ji} where $i, j \epsilon N$ and $i \neq j$ becomes 1, if both order i and j are accepted and order j is processed just before order i, else z_{ji} takes the value of 0. The tardiness of order i is represented by T_i where $i \epsilon N$ and it should be noted that $T_i \geq 0$. This situation is ensured with the following definition $T_i = w_i \max\{0, (C_i - d_i)\}$, where C_i is the completion time of order i.

The modified MILP formulation for the OAS-1 problem is given below.

MILP for the OAS-1 problem:

$$\max \sum_{i=1}^{n}(Q_i y_i - w_i T_i) \tag{3.1}$$

s.t.

$$y_i = \sum_{t=1}^{n} x_{it} \qquad \forall i \epsilon N \tag{3.2}$$

$$\sum_{t=1}^{n} x_{it} \leq 1 \qquad t = 1, ..., n \tag{3.3}$$

$$z_{ji} \leq y_i \, and \, z_{ji} \leq y_j \qquad i, j \epsilon N, \ i \neq j \tag{3.4}$$

$$\sum_{q<t} x_{jq} + \sum_{q \geq t} x_{iq} \leq 1 + z_{ji} \qquad i, j \epsilon N, \ t = 2, ..., n, \ i \neq j \tag{3.5}$$

$$T_i \geq \sum_{j=1}^{n} p_j z_{ji} + p_i y_i - d_i \qquad i, j \epsilon N \tag{3.6}$$

$$T_i \geq 0 \qquad i \epsilon N \tag{3.7}$$

$$y_i \in \{0, 1\}, z_{ji} \in \{0, 1\}, x_{it} \in \{0, 1\} \qquad t = 1, ..., n, \ i, j \epsilon N \tag{3.8}$$

Constraint (3.2) demonstrates that an accepted order can be scheduled at exactly one position. It is ensured that each position can have at most one order with constraint set (3.3). Constraint (3.4) guarantees that when z_{ji} becomes 1, both order i and order j are accepted. Constraint set (3.5) enables that when order i is accepted and order j is executed before order i, z_{ji} becomes 1. The value of T_i is represented by constraint (3.6) and constraint (3.7) makes sure that the value of T_i cannot be negative. Constraint (3.8) is given to define three binary variables in the formulation.

3.2.2 MILP Formulation for the OAS-2 problem

In this section, we present the MILP formulation for the OAS-2 problem, which was given by Oğuz et al. (2010).

Two sets of binary decision values are defined, I_i and y_{ij} in the MILP formulation where i and j indicate the index for the orders $(i, j = 1, ..., n)$ in the incoming order set N. I_i is used to represent whether order i is accepted or not by the firm and it equals to 1, if order i is accepted, otherwise it becomes 0. The second binary decision value, y_{ij}, in the formulation

is used to define the orders which are processed consecutively. If order i is executed before order j, y_{ij} becomes zero, else it takes 1. There are also two dummy orders to represent the first and the last position of the sequence as order 0 and order $n+1$. Order 0 is processed in the first position while order $n+1$ is executed in the last position of the sequence. The MILP formulation developed by Oğuz et al. (2010) is given below. We should note that objective function is the same for both the OAS-1 problem and the OAS-2 problem.

MILP for the OAS-2 problem:

$$\max \sum_{i=1}^{n} (Q_i y_i - w_i T_i) \tag{3.9}$$

s.t.

$$\sum_{j=1, j\neq i}^{n+1} y_{ij} = I_i \qquad \forall i = 0, ..., n \tag{3.10}$$

$$\sum_{j=0, j\neq i}^{n} y_{ji} = I_i \qquad \forall i = 1, ..., n+1 \tag{3.11}$$

$$C_i + (s_{ij} + p_j) \times y_{ij} + \bar{d}_i \times (y_{ij} - 1) \leq C_j \quad \forall i = 0, ..., n, \forall j = 1, ..., n+1, i \neq j \tag{3.12}$$

$$(r_j + p_j) \times I_j + s_{ij} \times y_{ij} \leq C_j \qquad \forall i = 0, ..., n, \forall j = 1, ..., n+1, i \neq j \tag{3.13}$$

$$T_i \geq C_i - d_i \qquad \forall i = 0, ..., n+1 \tag{3.14}$$

$$C_i \leq \bar{d}_i \times I_i \qquad \forall i = 0, ..., n+1 \tag{3.15}$$

$$T_i \leq (\bar{d}_i - d_i) \times I_i \qquad \forall i = 0, ..., n+1 \tag{3.16}$$

$$T_i \geq 0 \qquad \forall i = 0, ..., n+1 \tag{3.17}$$

$$R_i \leq e_i \times I_i - T_i \times w_i \qquad \forall i = 1, ..., n \tag{3.18}$$

$$R_i \geq 0 \qquad \forall i = 1, ..., n \tag{3.19}$$

$$C_0 = 0,\ C_{n+1} = \max_{i=1,...,n} \{\bar{d}_i\}, \qquad \forall i = 1, ..., n \tag{3.20}$$

$$I_0 = 1,\ I_{n+1} = 1 \tag{3.21}$$

$$I_i \in \{0, 1\}, y_{ij} \in \{0, 1\} \qquad \forall i = 1, ..., n \tag{3.22}$$

Constraint set (3.10) and (3.11) ensure that if an order is processed on the machine (not in the first or the last position in the sequence), the order has one preceding and one succeeding order. Constraint set (3.12) guarantees that if order i is processed before order

j, the completion time of the order j, C_j, cannot be smaller than the summation of the completion time of order i, the processing time of order j with the setup time between i and j. In contrast, when order i is not executed before order j, the same constraint enforces that C_i cannot be greater than its deadline plus a non-negative term. Constraint set (3.13) sets that if order j is accepted, C_j should take the value at least the summation of its release time and its processing time, and if order i is followed by order j, the setup time between i and j should be added to this summation. The tardiness value of each order is defined with the constraint sets (3.14)-(3.17). In the problem, a customer refuses to pay for the order when the order is delivered after its deadline. In the formulation, this situation is given in constraint set (3.15). Constraint sets (3.18) and (3.19) calculate the revenue gained from the order i when the order i is accepted with a tardiness of T_i. This case is observed when an order is completed after its due date but before its deadline. Constraint set (3.20) explains the completion times of the dummy orders. Constraint set (3.21) ensures that dummy orders are accepted. Finally, constraint set (3.22) is used to define two binary variables.

Chapter 4

A HEURISTIC SOLUTION APPROACH

In this thesis, we investigate two OAS problems: OAS-1 and OAS-2. These two problems cannot be solved with an exact algorithm in a reasonable computational time since they are NP-hard problems.

We observe from the literature that the GA is very successful to solve the OAS-1 problem. Furthermore, the TS and the ABC algorithms are very efficient to handle the OAS-2 problem. VNS algorithm is not used to solve the OAS problem in the literature so far. However, it is very effective to solve total tardiness scheduling problems with sequence dependent setup times. Since systematic change of neighborhoods in the search space can be an efficient approach to solve the OAS problems. We were motivated to develop a VNS algorithm as a solution method in this thesis. Following sections give the details of the VNS algorithm.

4.1 Variable Neighborhood Search (VNS)

Mladenovic and Hansen (1997) analyze properties of the general metaheuristics. They state that although general metaheuristics are very successful, they employ just one neighborhood structure. By observing that changing neighborhood structure in a path can be an efficient way to obtain improved results, they develop a new metaheuristic and name it as variable neighborhood search. VNS is a trajectory based metaheuristic and works with the systematic change of neighborhood structures within the search. VNS algorithm is summarized as follows. In the algorithm, there is a finite set of predetermined neighborhood structures. The algorithm starts with an initial solution. It visits the first neighborhood structure to generate a new solution. Then, a local search procedure is applied to the solution. Aim of the local search is to get a local optimum solution by using the gathered search knowledge and narrower solution subspace. Then, obtained local optimum solution

is compared with the incumbent solution. If the local optimum solution is better than the incumbent solution, we continue the search in the first neighborhood. Otherwise, we visit the next neighborhood structure. In the next neighborhood structure, a new solution is generated. The local search procedure is employed again. If an improvement is observed, we visit the first neighborhood structure, else we move to the next neighborhood structure. This procedure is repeated until a termination criteria is met.

This is the basic definition for VNS algorithm. In the rest of this chapter, the details of the proposed VNS algorithm is elaborated.

4.2 Proposed VNS Algorithm

The papers studied by Mladenovic and Hansen (1997), Hansen and Mladenovic (2001), Liu and Zhou (2012) and Kirlik and Oğuz (2012) prove the success of the VNS algorithm on NP-hard problems. They provide insights about how we can conceptualize the VNS algorithm with respect to properties of the OAS problem.

4.2.1 Solution Representation

We use the same representation for both the OAS-1 problem and the OAS-2 problem. We employ a representation which is based on the indication of the position of order i in the sequence with the corresponding i^{th} entry of a vector. Furthermore, if order i is not accepted, the entry value is represented with zero in the vector. We explain the representation with the following example which considers a problem with 10 orders ($n = 10$) where order $i = 1, \ldots, 10$. The vector with the values such that [3 0 5 0 0 1 2 4 0 0] means that order 1 is executed in the third position whereas order 2 is rejected. Furthermore, order 3 is put into operation at the fifth position. Since the entry values are zero in the vector, order 4, order 5, order 9 and order 10 are rejected orders by the firm. Order 6 is processed order in the first position of the sequence. Order 6 is followed by order 7. Finally, order 8, the last positive entry in the vector, is executed at the fourth position on the machine.

4.2.2 Initial Solution

We employ a greedy rule to acquire an initial solution for the OAS-1 problem and the OAS-2 problem in the study, but the employed greedy rule is not the same for the problems.

These rules are described below for the OAS-1 problem and the OAS-2 problem. The initial feasible solution is represented with s_0 in both of the problems.

Initial solution for the OAS-1 problem

For the OAS-1 problem, each order is defined with a processing time, a revenue, a due date and a weight. The greedy rule is based on the processing time and the revenue of an order. Basically, it calculates the ratio of the revenue of an order to its processing time. The reason to employ the rule is to give a priority in the sequence to orders which bring more revenue while consuming less time to be processed. After the calculation of the ratio of revenue to processing time of each order i, *Revenue-load ratio (RLR)*, $RLR1_i = e_i/p_i$, the ratio values are sorted from the highest to the lowest. The orders are positioned in the sequence with respect to the corresponded ratio values. A feasibility check is utilized to delete non-profit orders whose tardiness values are greater than the revenues. For example consider a problem with $n = 10$, the initial sequence is generated with the greedy rule as [9 4 5 3 10 1 6 7 2 8]. If order 2, order 4 and order 5 are non-profit orders. In this situation, we redesign the sequence by deleting these orders as [7 0 3 0 0 1 4 5 2 6] to have a feasible solution.

Initial solution for the OAS-2 problem

We use the same greedy rule for the OAS-2 problem with the study of Cesaret et al. (2012). This rule depends on the revenue, the processing time and the sequence dependent setup time of each order. It is determined for order i with $RLR1_i = e_i/(p_i + s_{average,i})$, where $s_{average,i} = (s_{0,i} + s_{1,i} + ... + s_{n,i})/(n + 1)$. They sort ratio values in decreasing order to give priority to orders with a higher revenue and which consume less time to complete its processing with the effect of average dependent setup time. The orders are executed in the order determined by the ratio values. Next, a feasibility check is made as in OAS-1 problem for OAS-2 problem to remove non-profit orders from the sequence.

4.2.3 Neighborhood Structure Definition and Local Search Procedure

One of the most important step while generating a VNS algorithm is to decide on neighborhood structures and the local search procedure. We conduct preliminary tests to make

this decision. Details of the preliminary tests are presented in Section 5.2. We employ two neighborhood structures for VNS algorithm: swap and reverse order. The local search procedure relies on a compound move that is the combination of three move operators: drop-add-insert. We use the same neighborhood structure and the local search procedure for both the OAS-1 problem and the OAS-2 problem. In the following, we provide the details of the neighborhood structures and the local search procedure.

First Neighborhood Structure: Swap

Selection of the neighborhood structures depends on the efficiency of the operator in the problem. The first neighborhood structure employed for the VNS algorithm is the swap operator. We yield a new solution from the current one, s, by swapping two entries of the solution vector randomly. This move allows us alternatively to change the positions of orders from both of the accepted orders in the sequence and to exchange one of the accepted order with a rejected order in the sequence. Furthermore, we keep the number of accepted orders the same in the sequence in this move. For example, if the current solution is [7 0 3 2 0 0 4 5 1 6], the accepted orders in sequence is $9-4-3-7-8-10-1$ and the rejected orders are $2, 5, 6$. If we pick the order 3 and 9 for swap operator, randomly, we switch the positions of order 3 and order 9 in the sequence and new solution becomes [7 0 1 2 0 0 4 5 3 6] with the new sequence $3-4-9-7-8-10-1$. Alternatively, one selected order can be from the accepted set, while the other one can be from the rejected order set as for example orders 2 and 7. In this case, the new solution is [7 4 1 2 0 0 0 5 3 6] and it corresponds to the sequence $3-4-9-2-8-10-1$. This move enables us to change the accepted orders while keeping number of accepted orders the same. We realize that selection of two rejected orders for swap from the sequence can cause a cycling problem. Hence, we do not switch two rejected orders to avoid cycling.

New schedule generated by the swap operator can be infeasible since changing the positions of the orders can affect the completion time of the orders in the sequence. We utilize the feasibility check for the new solution by deleting non-profit orders from the sequence. We observe that the number of accepted orders can change with respect to the feasibility check.

Second Neighborhood Structure: Reverse Order

Neighborhood structures used in the VNS algorithm should be nested while enlarging the search space. Hence, the second neighborhood structure, reverse order, is a good complement for the swap operator. It involves the selection of two accepted orders or one accepted order and one rejected order from the sequence randomly and the subsequence identified by these orders is reversed. For example, if we have a solution as [7 0 3 2 0 0 4 5 1 6], its processing sequence is $9-4-3-7-8-10-1$ with the rejected orders $2, 5$ and 6. Randomly, orders 4 and 9 are selected and the new solution is generated as [7 0 3 1 5 4 0 0 2 6] by reversing the orders in the subsequence identified by the selected orders. New solution corresponds to a new sequence as $4-9-3-5-6-10-1$. Alternatively, we can select one accepted order and one rejected order from the sequence as order 2 and 10 from the solution [7 0 3 2 0 0 4 5 1 6]. We obtain a new solution as [7 6 1 5 4 0 0 2 3 0] whose sequence is $3-8-9-5-4-2-1$. We observe from two examples that reverse order keeps the number of accepted orders the same in the sequence but it causes changing the members of the accepted order set. The feasibility check mechanism should be applied to the new solution generated by the reverse order as it can change the completion time of the orders in the sequence. After getting a new solution at each neighborhood structure, the local search procedure is employed to search improved solutions in a narrowed space.

Local Search Procedure: Compound move

The neighborhood structures do not allow changing the number of the accepted orders in the sequence, but they provide regeneration of accepted order set. After generation of a new solution, we check feasibility of the solution. We may delete one or more orders from the solution to get a feasible solution. This situation can cause to converge a local optimum solution. The local search is necessary to escape from local optimum solution. Compound move is selected as the local search procedure for the algorithm. It relies on iterative drop-add-insert operations which is similar to the local search in the study of Cesaret et al. (2012). The local search procedure is initiated from a random solution generated in the neighborhood structure for the proposed VNS algorithm. We explain the compound move operator of the local search procedure below.

Drop operation: Aim of this operation is to remove an order from the current sequence.

We drop the order with respect to the characteristics of the orders by using different ways for the OAS-1 problem and the OAS-2 problem. Order which brings less revenue but consumes a large amount of time while processing is removed from the sequence for the OAS-1 problem. We assign a ratio value for each accepted order as $RLR2_i = e_i/p_i$ and drop the order i with the minimum $RLR2_i$ value.

In the OAS-2 problem, we drop order i which is executed immediately after order j and has the minimum $RLR2_i$ where $RLR2_i = e_i/(p_i + s_{ji})$. A new sequence is generated when an order is removed from the sequence. An example is given as follows. If the the current solution is [7 0 3 2 0 0 4 5 1 6], whose processing sequence is $9-4-3-7-8-10-1$ with the rejected orders $2, 5, 6$. If order 7 is selected from the sequence with respect to minimum $RLR2$ value, we remove order 7 from the sequence. This corresponds to a new solution as [6 0 3 2 0 0 0 4 1 5], whose processing sequence is $9-4-3-8-10-1$.

Add and Insert operations: We calculate $RLR1$ values for all rejected orders. $RLR1$ values are sorted from the highest to the lowest to obtain a set ($SortReject$) with rejected orders. Add operation is applied to all rejected orders in this set. We select the first order in the set which has maximum $RLR1$ value to be added. After this selection, insertion move is utilized. In order to obtain a new solution, we try all possible positions in the sequence and decide the best position which brings the maximum profit when inserted. This procedure is called as insertion. Here, we make a decision to continue add and insertion operations according to the new solution. If the objective function value of the new solution is 0.998 times greater than the incumbent solution, we continue add and insertion operators with the following way. We constitute a probability based selection method to decide the next order to be added. $RLR1$ values are used to generate a probability distribution. We calculate cumulative probability values based on the distribution. A random number is generated and the interval corresponds to this random number is determined by checking cumulative probabilities to detect the order to be added. We insert the selected order to its best position. If the corresponding solution gives a total profit which is greater than 0.998 times the incumbent solution, we keep the solution and continue add-insertion procedure. Otherwise, we select the next order from the $SortReject$ to be added and we repeat the procedure until all rejected orders are tried to be added. We pick the solution which gives the maximum profit as local optimum solution.

For example, as given above we have the new solution [6 0 3 2 0 0 0 4 1 5], whose processing sequence is $9-4-3-8-10-1$ after dropping order 7. We generate *SortReject* set from this sequence. In the example, the rejected orders are order 2, order 5, order 6 and order 7. We calculate the $RLR1$ value for each rejected order. Suppose that $RLR1_2$=5.47, $RLR1_5$=10.5, $RLR1_6$=7.12 and $RLR1_7$=0.43. We sort $RLR1$ values from the highest to the lowest as $RLR1_5$, $RLR1_6$, $RLR1_2$ and $RLR1_7$. We get a *SortReject* set with the orders the sorted according to $RLR1$ values which is *SortReject*={5, 2, 6, 7} for this example. We select order 5 to be added since it is the first member of the *SortReject* set. Here, the insertion procedure starts. We find the best position for order 5 in the sequence. Assume that, the best position is determined as the third position in the sequence to be inserted. New solution becomes [7 0 4 2 3 0 0 5 1 6], whose processing sequence is $9-4-5-3-8-10-1$. If this solution brings a profit value which is 0.998 times greater than the incumbent solution, we keep the solution and continue add and insertion operators for this sequence. Herein, we continue the addition of orders by a probabilistic method. We construct cumulative probabilities using $RLR1$ of orders. Then, we create a random number and we decide the order to be added by checking in which interval this random number fits. For example, order 2 is selected to be added. Then, we insert the order to its best position by trying all possible positions in the sequence. If the objective function value of the new sequence is not 0.998 times greater than the incumbent solution, we continue with the the next rejected order from *SortReject* set (we continue with order 6 for this example) and repeat the procedure. The local search is ended when we try the addition of all orders in *SortReject* set to the sequence. We pick the solution with the maximum total profit obtained from the local search procedure.

During the local search, new solutions are checked for the feasibility. The compound move can cause deleting the orders while adding more than one order to keep feasibility. We note that the compound move allows to change both the accepted and the rejected orders.

We now present the pseudo code of the VNS algorithm.

Notation

k_{max}: maximum number of neighborhood structures

s_0: the initial solution

s: the current solution

$f(s)$: objective function value of s

s^*: the best known solution

$N(s)$: the neighborhood of s

s': random solution from the k^{th} neighborhood structure ($N_k(s)$)

$s^{inserted}$: solution generated after the insertion

s'': local optimum solution obtained from s'

$max(s^{inserted})$: maximum value among the $s^{inserted}$

$CumProb$: generated cumulative probability array

$SortReject$: sorted rejected orders array with respect to $RLR1$ values

$reject$: rejected order array

Algorithm 1 VNS algorithm for the OAS-1 problem and the OAS-2 problem

Require: p_i, d_i, e_i, w_i, n, k_{max} (for OAS-1)
Require: r_i, p_i, s_{ij}, d_i, $\bar{d}_i$, e_i, w_i, n, k_{max} (for OAS-2)
Generate s_0, $s \leftarrow s_0$, $f(s^*) \leftarrow f(s)$
$k \leftarrow 1$
while termination criterion is not met **do**
  while $k \leq k_{max}$ **do**
    Generate s' from $N_k(s)$
    ▷ LOCAL SEARCH PART begins
    Select the order with the minimum $RLR2$ to be dropped from s'
    Find $reject$ for s'
    for f:1 to size of $reject$ **do**
      $s_{local}(f) \leftarrow s'$
      Calculate $RLR1$ for each order $i \in reject$
      Generate $SortReject$ set
      Find the selected order to be added from the f^{th} member of $SortReject$
      Set $l = 1$ ▷ INSERTION PROCEDURE STARTS
      for $l \leq n$ **do**
        Insert selected order to l^{th} position
        Set inserted solution as $s'^{inserted}(l)$
        l++
      end for
      $s^{inserted} \leftarrow max(s'^{inserted})$
      if $f(s^{inserted}) \geq 0.998 f(s^*)$ **then**
        $s_{local}(f) \leftarrow s^{inserted}$
        Update $f(s^*)$, s^*
        $keepinsertion = 1$
      else
        $keepinsertion = 0$
      end if ▷ INSERTION PROCEDURE ENDS
      while $keepinsertion = 1$ **do**
        Calculate $RLR1$ for each order $i \in reject$ from $s_{local}(f)$
        Construct $CumProb$ for all orders proportional to their $RLR1$
        Find the order to be inserted from $reject$ randomly
        Repeat INSERTION PROCEDURE
      end while
    end for
    Generate s'' from $max(s_{local})$
    ▷ LOCAL SEARCH PART ends
    if $f(s'') \geq f(s^*)$ **then**
      Update s^*, s
      $k = 1$
    else
      $k = k + 1$
    end if
  end while
end while

Chapter 5

COMPUTATIONAL STUDIES

We test the performance of the VNS algorithm computationally. In this chapter, we present the details of the computational studies with the results. In Section 5.1, we describe the data set employed for the computational studies. In Section 5.2, we give the preliminary tests which are used to determine the neighborhood structures of the VNS algorithm described in Section 4.2.3. Tuning the parameters is explained in Section 5.3. In Section 5.4, we present the computational results. In Section 5.5, we analyze and discuss the results.

5.1 Data Sets

5.1.1 Data Set for the OAS-1 problem

For the OAS-1 problem, we use the data set generated by Rom and Slotnick (2009). Processing times and weights were created from a uniform distribution in the range of (0,1). Due dates were also drawn from a uniform distribution, but they were set according to the processing time values . The revenue values were produced from a log normal distribution which was obtained by the utilization of the normal distribution with mean zero and standard deviation one. They used the tardiness factor τ which was adjusted to 0.3, 2.0, 3.0 to generate the due dates. Moreover, they generated the revenue values for the half of the instances in correlation (Crl) with due dates and the remaining half was drawn without correlation. Hence, they gathered six data sets with different properties. There were three different sizes for the problems ($n = 50, 75, 100$) with six data sets which summed up to 360 instances. We use these 360 instances to test the VNS algorithm for the OAS-1 problem.

5.1.2 Data Set for the OAS-2 problem

In order to test the performance of the VNS algorithm on the OAS-2 problem, we use test instances generated by Cesaret et al. (2012). They generated processing times and revenues

by using the discrete uniform distribution on the interval [0,20]. The remaining properties of the data generated by Cesaret et al. (2012) are as follows. Release dates were drawn on the interval $[0, \tau\, p_T]$, where p_T was the total processing time of all orders while setup times were drawn on the interval [1,10] from the discrete uniform distribution. Due dates were obtained with the following formula $d_i = r_i + \max_{j=0,1,\ldots,n} s_{ji} + \max\{slack, p_i\}$, where $slack$ was drawn from a discrete uniform distribution in the range $[p_T(1-\tau-R/2), p_T(1-\tau+R/2)]$. Deadlines were generated with the formula as $\bar{d}_i = d_i + R\, p_i$. Moreover, they employed the tardiness factor τ and the due date range R which took five different values: $0.1, 0.3, 0.5, 0.7, 0.9$ for the test instances. 10 problem instances were created for each combination of τ and R values. They used six different sizes for the number of orders ($n = 10, 15, 20, 25, 50, 100$). Hence, they totally obtained 1500 instances. We use these 1500 instances to test the VNS algorithm for the OAS-2 problem.

5.2 Selection of the Neighborhood Structures

The major parts for developing the VNS algorithm are the definition of the neighborhood structures in their best order and finding an intelligent local search procedure. We conduct preliminary tests in which we vary the neighborhood structures and type of the local search procedure, including TS and SA. We design the tests as follows. For the neighborhood structures, we use swap, reverse order and compound move operator in different combinations. Definitions of the swap, the reverse order and the compound move operators were already given in Section 4.2.3. We employ two types of local search procedure: Local search-1 and Local search-2. Local search-1 improves the solutions from a random solution by utilizing pairwise interchange for the adjacent orders in the sequence. We perform local search-2 to a random solution obtained in the neighborhood by applying the compound move. When the compound move is employed in the algorithm as one of the neighborhood structures, TS is included in the move for some tests. TS keeps the most recent inserted orders to avoid cycling. We use SA with a constant temperature to make decision about moving to next neighborhood structure for some tests. For the OAS-1 problem, we use the data set described in Section 5.1.1 with n=50 for all preliminary tests. We perform the preliminary tests for the OAS-2 problem by using the data set explained in Section 5.1.1 with n=10. The results for preliminary tests are given in Tables 5.1 and 5.2 in terms of the solution

quality for the OAS-1 problem and the OAS-2 problem, respectively. We use the average percentage deviation (Avg.Dev) from the upper bound values as a performance measure. We show the calculation of average percentage deviation in Equation 5.1. We explain the upper bounds in Section 5.4.3.

Table 5.1: Preliminary test results to select neighborhood structures for the OAS-1 problem where n=50

Test Number	Neighborhood Structures	Local Search	TS	SA	Avg. Dev. (%)
Test1	Compound Move-Swap	Local search-1	No	No	2.335
Test2	Compound Move-Swap	Local search-1	Yes	No	2.002
Test3	Compound Move-Swap	Local search-1	No	Yes	2.346
Test4	Compound Move-Swap	Local search-1	Yes	Yes	2.231
Test5	Swap-Compound Move	Local search-1	No	No	2.530
Test6	Swap-Compound Move	Local search-1	Yes	No	2.126
Test7	Swap-Compound Move	Local search-1	No	Yes	2.234
Test8	Swap-Compound Move	Local search-1	Yes	Yes	2.329
Test9	Compound Move-Swap-Reverse order	Local search-1	No	No	0.829
Test10	Compound Move-Swap-Reverse order	Local search-1	Yes	No	0.661
Test11	Compound Move-Swap-Reverse order	Local search-1	No	Yes	0.987
Test12	Compound Move-Swap-Reverse order	Local search-1	Yes	Yes	0.837
Test13	Compound Move-Reverse order-Swap	Local search-1	No	No	1.187
Test14	Compound Move-Reverse order-Swap	Local search-1	Yes	No	1.063
Test15	Compound Move-Reverse order-Swap	Local search-1	No	Yes	1.228
Test16	Compound Move-Reverse order-Swap	Local search-1	Yes	Yes	1.007
Test17	Swap-Compound Move-Reverse order	Local search-1	No	No	0.945
Test18	Swap-Compound Move-Reverse order	Local search-1	Yes	No	0.895
Test19	Swap-Compound Move-Reverse order	Local search-1	No	Yes	1.012
Test20	Swap-Compound Move-Reverse order	Local search-1	Yes	Yes	1.122
Test21	Swap-Reverse order-Compound Move	Local search-1	No	No	1.213
Test22	Swap-Reverse order-Compound Move	Local search-1	Yes	No	1.130
Test23	Swap-Reverse order-Compound Move	Local search-1	No	Yes	1.318
Test24	Swap-Reverse order-Compound Move	Local search-1	Yes	Yes	1.234
Test25	Reverse order-Compound Move-Swap	Local search-1	No	No	1.413
Test26	Reverse order-Compound Move-Swap	Local search-1	Yes	No	1.338
Test27	Reverse order-Compound Move-Swap	Local search-1	No	Yes	1.531
Test28	Reverse order-Compound Move-Swap	Local search-1	Yes	Yes	1.538
Test29	Reverse order-Swap-Compound Move	Local search-1	No	No	1.430
Test30	Reverse order-Swap-Compound Move	Local search-1	Yes	No	1.331
Test31	Reverse order-Swap-Compound Move	Local search-1	No	Yes	1.667
Test32	Reverse order-Swap-Compound Move	Local search-1	Yes	Yes	1.632
Test33	Swap-Reverse order	Local search-1	No	No	2.058
Test34	Swap-Reverse order	Local search-1	Yes	No	1.843
Test35	Swap-Reverse order	Local search-1	No	Yes	2.214
Test36	Swap-Reverse order	Local search-1	Yes	Yes	2.131
Test37	Swap-Reverse order	Local search-2	No	No	**0.503**
Test38	Swap-Reverse order	Local search-2	No	Yes	0.953
Test39	Reverse order-Swap	Local search-1	No	No	2.281
Test40	Reverse order-Swap	Local search-1	Yes	No	2.031
Test41	Reverse order-Swap	Local search-1	No	Yes	2.255
Test42	Reverse order-Swap	Local search-1	Yes	Yes	2.539
Test43	Reverse order-Swap	Local search-2	No	No	0.928
Test44	Reverse order-Swap	Local search-2	No	Yes	1.231

We select the tests which give better solutions in Table 5.1 and perform the selected preliminary tests on the OAS-2 problem. The results are given below.

Table 5.2: Preliminary test results to select neighborhood structures for the OAS-2 problem where n=10

Test Number	Neighborhood Structures	Local Search	TS	SA	Avg. Dev. (%)
Test9	Compound Move-Swap-Reverse order	Local search-1	No	No	0.522
Test10	Compound Move-Swap-Reverse order	Local search-1	Yes	No	0.534
Test11	Compound Move-Swap-Reverse order	Local search-1	No	Yes	0.538
Test12	Compound Move-Swap-Reverse order	Local search-1	Yes	Yes	0.550
Test13	Compound Move-Reverse order-Swap	Local search-1	No	No	0.838
Test14	Compound Move-Reverse order-Swap	Local search-1	Yes	No	0.830
Test15	Compound Move-Reverse order-Swap	Local search-1	No	Yes	0.862
Test16	Compound Move-Reverse order-Swap	Local search-1	Yes	Yes	0.833
Test17	Swap-Compound Move-Reverse order	Local search-1	No	No	0.945
Test18	Swap-Compound Move-Reverse order	Local search-1	Yes	No	0.995
Test19	Swap-Compound Move-Reverse order	Local search-1	No	Yes	0.912
Test20	Swap-Compound Move-Reverse order	Local search-1	Yes	Yes	0.952
Test21	Swap-Reverse order-Compound Move	Local search-1	Yes	Yes	1.013
Test22	Swap-Reverse order-Compound Move	Local search-1	Yes	No	1.030
Test23	Swap-Reverse order-Compound Move	Local search-1	No	Yes	1.018
Test24	Swap-Reverse order-Compound Move	Local search-1	Yes	Yes	1.034
Test37	Swap-Reverse order	Local search-2	No	No	**0.136**
Test38	Swap-Reverse order	Local search-2	No	Yes	0.483
Test43	Reverse order-Swap	Local search-2	No	No	0.428
Test44	Reverse order-Swap	Local search-2	No	Yes	0.631

According to the results given in Tables 5.1 and 5.2, we observe that the best performance is obtained when test37 is employed. Hence, we decide the neighborhood structures as the following. The first structure is swap and it is followed by reverse order with the selection of the compound move as a local search. The running times for each VNS algorithm are not reported in details here. It is due to that the majority of the instances are finished in similar times. After deciding neighborhood structures, we set the parameters of the algorithm to improve the results.

5.3 Tuning the Parameters

In this section, we set the parameters to obtain the best performance of the VNS algorithm with respect to the neighborhood structures and the local search procedure determined in Section 5.2. These parameters are threshold value and termination criterion.

5.3.1 Threshold Value

In a VNS algorithm, the local search procedure may result in getting stuck in a local optimum solution. In our algorithm, we decide to employ the compound move described in Section 4.2.3 as the local search procedure. We design to have a threshold value for accepting inferior solutions in the local search procedure to avoid from being trapped in a local optimum solution. In order to choose the neighborhood structures and the local search procedure in Section 5.2, we employ the threshold value as 0.998 in the preliminary tests. To tune the threshold value, we test the values 0.997, 0.998, 0.999 and 1.000 on the same data sets used in Section 5.2 for the OAS-1 problem and the OAS-2 problem. Tables 5.3 and 5.4 show the performance results with threshold values for the OAS-1 problem and the OAS-2 problem, respectively. We set the improvement threshold value to 0.998 based on the results for both of the problems.

Table 5.3: Preliminary test results to set the threshold values for the OAS-1 problem where $n = 50$

n	Threshold value	Avg.Dev. (%)
50	0.997	0.552
50	**0.998**	**0.503**
50	0.999	0.738
50	1.000	0.847

Table 5.4: Preliminary test results to set the threshold values for the OAS-2 problem where $n = 10$

n	Threshold value	Avg.Dev. (%)
10	0.997	0.154
10	**0.998**	**0.136**
10	0.999	0.262
10	1.000	0.383

5.3.2 Termination criterion

The algorithm is run for the preliminary tests given in Section 5.3.1 and Section 5.2 until a certain number of iterations. We decide to use the number of solutions without improvement generated as a termination criterion since it is more efficient. We set a counter in the algorithm to determine the number of the solutions without improvement. We need to set a number for the termination criterion with respect to the convergence behavior of the algorithm. Values between 300 and 2000 are tested. 750 and 1250 are selected as the number of non improving solutions for the OAS-1 problem and OAS-2 problem. We give two examples to visualize how the convergence is acquired for the OAS-1 and OAS-2 problems in Figure 5.1 and Figure 5.2, respectively.

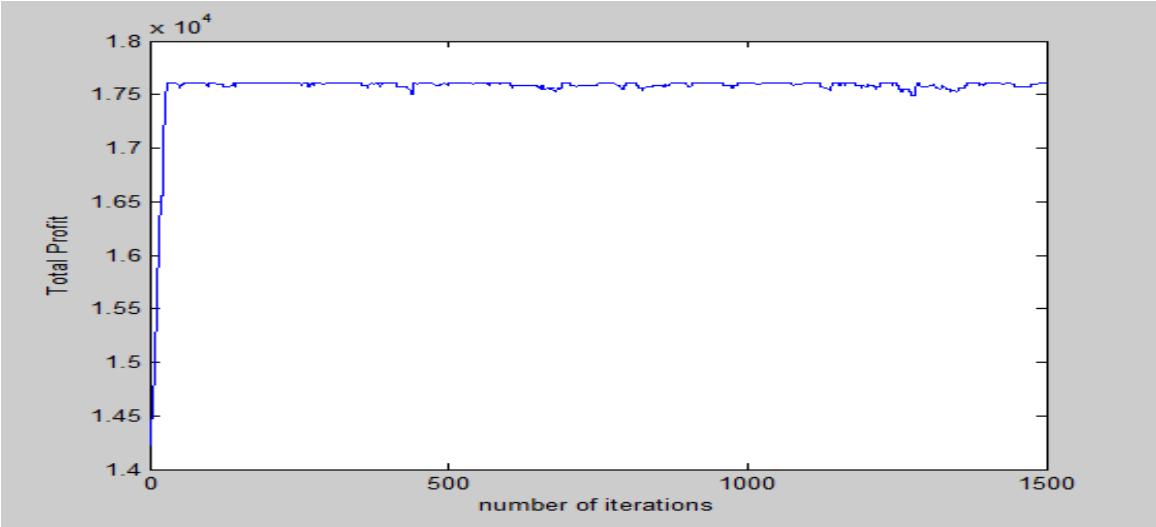

Figure 5.1: Convergence of the VNS algorithm for an instance with n=50, $\tau = 0.3$ and Crl=0.025

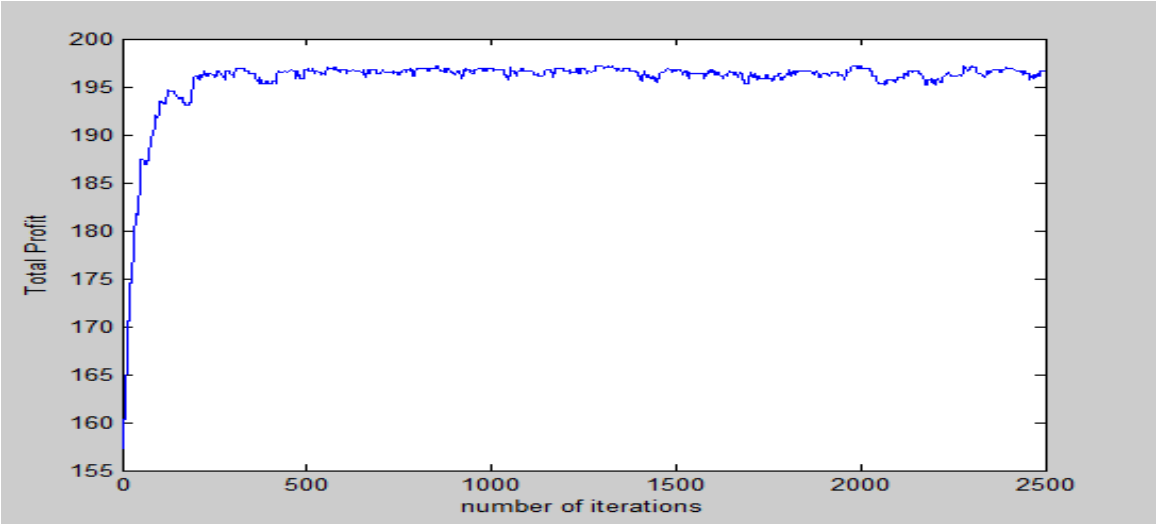

Figure 5.2: Convergence of the VNS algorithm for an instance with 25 orders, $\tau = 0.9$ and $R = 0.7$

5.4 Results of the Computational Study

After determining the best settings for the VNS algorithm, we perform a computational study which compares the VNS algorithm with the benchmarks.

5.4.1 Computational Platform

We conduct all the experiments on a computer with an Intel(R) Core(TM) processor, 2.50 GHz speed and 4GB of RAM. The VNS algorithm is coded in MATLAB R2010b.

5.4.2 Benchmarks

We use different heuristic algorithms from the literature as a benchmark to compare the performance of the VNS algorithm. We use the myopic and the genetic algorithm presented by Rom and Slotnick (2009) as benchmark for the OAS-1 problem. For the OAS-2 problem, the benchmarks are the tabu search studied by Cesaret et al. (2012) and artificial bee colony

developed by Ling and Ying (2013). We explain the properties of the benchmarks in the following.

Myopic Algorithm: It is a heuristic method based on the reassembling the accepted orders of the relaxation solution. Myopic algorithm is initiated by the calculation of the profit when all orders are placed in their original position. Each job is separated into joblets with respect to unit processing time, the weights and revenues after initiating. Joblets are assigned to unit duration time buckets to find an optimal sequence for the relaxed problem. The optimal sequence is searched by maximizing the return of accepting or rejecting each joblet. At least %75 of the joblets are accepted in the relaxed solution. Reassembling orders are sorted ascendantly with respect to the value obtained by the subtraction of the processing time from the completion time to modify the sequence. Remaining orders are sequenced after previously accepted orders with the heuristic described by Mohan et al. (1983) and Morton et al. (1993). The myopic algorithm is coded in FORTRAN 90 and run on a Gateway computer with an Intel Pentium M 1.6 GHz processor. Its performance is evaluated by the test problems generated by Rom and Slotnick (2009).

Genetic Algorithm (GA): GA is a population based heuristic method. GA is initiated with a population including random solutions. Each order in the solution is evaluated sequentially as follows. Return of each order to the sequence is checked and if this order increases the profit, it is added to the sequence until the end of the sequence is reached. The initial population is obtained by the evaluation of each solution. Then, GA starts the generation of the next population. All the solutions are sorted with respect to descending order of their profit value. Solutions whose profit values are within %10 of profit of the best solution so far are kept and sorted in descending order in the population. For the remaining solutions, each profit value is modified with the following formula: $ModProfit = (Profit - Rand \times 0.05 \times \frac{BestV}{GenNum})$, where $ModProfit$ is the modified profit value of the solution, $Profit$ is the original profit value of the solution, $Rand$ is the random number drawn from uniform distribution (0, 1), $BestV$ is the best profit found so far and $GenNum$ is the number of this generation. The remaining solutions are sorted as regarding $ModProfit$ values in decreasing order in the population. Two point crossover operation is performed between the better and worse half of the population. Two numbers are randomly picked between 1 and N to determine two positions in the sequence. If the first random number is

greater than the second one, the outer parts of the sequence identified by these numbers are switched and new offspring is generated. Else, the offspring is generated by interchanging orders between two numbers. Generated offspring does not have to be feasible so they are converted to feasible solutions. When two generated solutions have the same value, the algorithm performs a mutation with a probability on one of them to eliminate the duplication by interchanging randomly two successive orders. The GA employs a local search to improve the solutions generated. The local search procedure utilizes interchanging successively order pairs. Best solutions are kept and the search continues until a certain number of iterations are performed. The GA is coded in FORTRAN 90 and run on a Gateway computer with an Intel Pentium M 1.6 GHz processor. Its performance is evaluated by the test problems generated by Rom and Slotnick (2009). The GA becomes the state of the art for the OAS-1 problem with respect to the achieved results.

Tabu Search (TS): TS algorithm starts with a greedy rule based on the processing time, dependent sequence setup time and revenue of an order. The greedy rule utilizes the calculation of $Revenue - loadratio$ for each order i, $RLR1_i = e_i/(p_i + s_{average,i})$, where $s_{average,i} = (s_{0,i} + s_{1,i} + ... + s_{n,i})/(n+1)$. Initial sequence is obtained by locating the orders into the corresponding positions to the sorted $RLR1_i$ values from the highest to the lowest. Non-profitable orders are deleted from the initial sequence to have a feasible solution. After the generation of the initial feasible solution, the algorithm starts. Neighborhood solutions are produced by swap operator. This operator enables interchanging positions of two accepted orders or one accepted order with one rejected order to generate solutions in the search space. The best solution is determined in the search space. In the tabu list, the swapped pairs provide the best solution are kept and they become tabu during the tabu tenure. A local search procedure is employed starting from the best solution. In the local search procedure, an iteratively drop-add-insert operations are performed. It is explained as follows. An order with the minimum $RLR2_i$ value is dropped from the sequence where $RLR2_i = e_i/(p_i + s_{ji})$. Then, a probability distribution is constructed for all rejected orders in the sequence to detect the order to be added. The probability distribution is deal with $RLR1_i$ values which becomes the selection of the orders with higher $RLR1$ values more likely as in roulette wheel selection procedure. After the selection of the order to be added, the order is inserted to the position in which it generates a profit 0.998 times greater than

the profit of best solution found so far. Add-insertion procedure is applied in all rejected orders. The best solution is updated and recorded. The algorithm is repeated until the generation of predetermined number of solutions without improvement. The TS algorithm is coded in C by using a workstation with a 3.00 GHz Intel Xeon processor and 4 GB of RAM. Cesaret et al. (2012) test the TS algorithm with data sets created by them. The TS algorithm is competitive improvement heuristic for the OAS-2 problem with respect to the gathered results.

Artificial Bee Colony (ABC): ABC is a population based heuristic algorithm. The initial population is generated randomly. After the generation of the initial solution, the algorithm begins. The following procedure is repeated until a limit which is the number trials without improvement. First, the algorithm generates solutions in the neighborhood close the their current position as the following. The neighborhood solutions are found in two phases: the destruction and construction. Predefined number of orders is selected randomly and deleted from the sequence in the destruction phase. Deleted orders are added to a new sequence with respect to the selected order in the destruction phase. In the construction phase, the deleted orders are reinserted sequentially by trying all possible positions to find the best position. Then, a local search procedure is applied to improve the solutions by iteratively exchanging successive orders in the solution. If the improved solution is better than the best solution found so far, it becomes a member of the new population. Otherwise, a new solution is generated by employing a crossover operator. A substring from the best solution is taken randomly for the crossover. A proto-child is created by copying the substring into its corresponding positions. A second parent is selected randomly. Some orders which are already in the substring from the second parent are deleted for the feasibility. An offspring is created by placing the orders into unfixed positions from left to right according to the order of the sequence. If the solution of the offspring is better than a threshold value, it is employed as a member of the new population. Otherwise, the local search procedure is applied to improve the solution. After the local search, the best solution is selected among the solutions generated by the destruction and construction phases and the crossover operator as a new member of the population. Some members of the new population are changed with respect to a determined probability by applying the destruction and construction phases, if an improvement is observed. The best solutions are recorded. The ABC algorithm is coded

with C language and run on a PC with an Intel Premium 4, 3.0 GHz CPU and 4 GB of RAM. The performance of the ABC algorithm is evaluated with the study done by Cesaret et al. (2012). The ABC algorithm becomes the state of the art for the OAS-2 problem.

5.4.3 Upper Bounds (UB)

Since the OAS problem is NP-hard, solving large sized test instances to optimality in a reasonable time is very difficult. To measure the effectiveness of the proposed VNS algorithm, UB values are obtained from the studies presented by Rom and Slotnick (2009) and Cesaret et al. (2012) for the OAS-1 problem and the OAS-2 problem, respectively.

Upper Bound for the OAS-1 problem: An assignment algorithm is applied to a unit processing time relaxation of the problem to find UB. Rom and Slotnick (2009) observe in their study that the determined UB values are really tight.

Upper Bound for the OAS-2 problem: Cesaret et al. (2012) obtain the UB values as the best of two upper bounds. The first one, UB_{LPVI}, is obtained by linear programming relaxation of MILP with valid inequalities which are proposed by Oğuz et al. (2010). UB_{MILP} is the second upper bound and is found by solving the MILP in a predetermined time.

5.4.4 Performance Measures

We need performance measures to compare the results obtained from the VNS algorithm with the benchmarks. We define the percentage deviations, run times and number of optimal solutions which are used to present the computational results.

Percentage Deviations: We compare the average (Avg.Dev.), minimum (Min. Dev.) and maximum (Max.Dev) percentage deviation from the UB. Equation 5.1 is used to calculate the average percentage deviation. In Equation 5.1, n is the problem size and *objective* represents the objective function value for the proposed algorithm.

$$(\frac{1}{n} \times \sum_{i=1}^{n} \frac{UB - objective}{UB}) \times 100 \tag{5.1}$$

Run Times: We report run times on the average in seconds to see the efficiency of the algorithms.

Number of optimal solutions: We report number of optimal solutions found (# of optimal

solutions) to measure the solution quality from another dimension.

5.4.5 Results for proposed VNS algorithm

We perform a computational study which compares the proposed VNS algorithm with the benchmarks given in Section 5.4.2 for the OAS-1 problem and the OAS-2 problem. The results of the OAS-1 problem are tabulated in Tables 5.5, 5.6 and 5.7 for n =50, 75 and 100, respectively. Tables 5.8 - 5.13 present the results for the OAS-2 problem with $n = 10, 15, 20, 25, 50$ and 100, respectively.

Table 5.5: Performance Results of the Myopic, Genetic and VNS Algorithms for the OAS-1 problem when n =50

		% Deviations from UB														
$n = 50$		Myopic			Genetic			VNS			# of optimal solutions			Run Times		
τ	Crl	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Myopic	GA	VNS	Myopic	GA	VNS
0.3	0.025	1.104	0.333	0.000	0.685	0.059	0.000	0.685	0.093	0.000	2	11	11	2.96	10.88	165.71
0.3	-0.213	2.647	0.295	0.000	1.051	0.112	0.000	1.051	0.205	0.000	5	10	10	4.07	10.74	122.47
2.0	0.007	2.398	0.298	0.000	1.273	0.146	0.000	1.273	0.207	0.000	6	10	10	4.69	11.19	172.56
2.0	-0.161	2.558	0.588	0.011	1.137	0.313	0.001	1.837	0.946	0.369	0	0	0	3.24	18.50	181.62
3.0	0.009	4.076	0.893	0.004	3.010	0.527	0.000	3.053	0.814	0.051	0	1	0	4.19	20.18	196.61
3.0	-0.243	5.869	0.988	0.054	1.397	0.432	0.021	1.445	0.729	0.021	0	0	0	5.25	22.84	195.70
	Avg.	3.109	0.566	0.022	1.426	0.265	0.004	1.558	0.499	0.074	2.17	5.33	5.16	4.07	15.72	172.43
	Total										13	32	31			

Table 5.6: Performance Results of the Myopic, Genetic and VNS Algorithms for the OAS-1 problem when n =75

		% Deviations from UB														
$n = 75$		Myopic			Genetic			VNS			# of optimal solutions			Run Times		
τ	Crl	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Myopic	GA	VNS	Myopic	GA	VNS
0.3	-0.016	1.570	0.355	0.000	0.526	0.090	0.000	2.951	0.573	0.000	1	8	4	8.14	24.51	228.42
0.3	-0.168	1.934	0.338	0.000	1.151	0.175	0.000	2.500	0.460	0.000	2	6	2	16.58	20.67	253.11
2.0	0.034	6.715	1.247	0.070	1.632	0.612	0.037	4.111	1.059	0.093	0	0	0	9.81	67.68	279.16
2.0	-0.191	4.127	1.903	0.103	2.614	0.938	0.047	4.433	1.771	0.077	0	0	0	13.94	63.55	284.23
3.0	-0.005	6.408	1.405	0.071	1.406	0.535	0.032	5.887	1.944	0.104	0	0	0	9.30	52.32	234.13
3.0	-0.167	9.628	2.923	0.077	12.799	1.280	0.026	5.605	2.245	0.050	0	0	0	14.90	56.29	301.14
	Avg.	5.064	1.362	0.054	1.688	0.605	0.024	4.248	1.342	0.054	0.50	2.33	1.00	12.11	47.50	263.37
	Total										3	14	6			

Table 5.7: Performance Results of the Myopic, Genetic and VNS Algorithms for the OAS-1 problem when n =100

		% Deviations from UB														
$n = 100$		Myopic			Genetic			VNS			# of optimal solutions			Run Times		
τ	Crl	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	Myopic	GA	VNS	Myopic	GA	VNS
0.3	-0.011	1.501	0.451	0.000	1.124	0.199	0.000	1.501	0.362	0.000	2	9	3	19.85	55.37	525.23
0.3	-0.219	1.684	0.366	0.000	1.411	0.194	0.000	1.684	0.326	0.000	2	7	2	38.74	46.57	586.40
2.0	0.021	3.064	1.009	0.042	1.503	0.651	0.074	3.064	1.009	0.037	0	0	0	22.22	119.21	490.55
2.0	-0.133	8.354	2.357	0.134	1.956	0.936	0.069	8.354	2.357	0.134	0	0	0	33.87	132.43	550.45
3.0	-0.034	4.100	1.675	0.106	2.259	0.853	0.023	4.100	1.675	0.106	0	0	0	23.90	137.16	586.23
3.0	-0.187	7.757	2.608	0.085	2.421	1.055	0.027	7.577	2.608	0.085	0	0	0	33.41	116.54	511.12
	Avg.	4.410	1.411	0.108	1.779	0.648	0.032	4.410	1.390	0.108	0.67	2.67	0.83	28.66	101.21	541.66
	Total										4	16	5			

Table 5.8: Performance of the TS, ABC and VNS algorithms for the OAS-2 problem when n=10

		% Deviations from UB														
$n = 10$		TS			ABC			VNS			# of optimal solutions			Run Times		
τ	R	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	TS	ABC	VNS	TS	ABC	VNS
0.1	0.1	3	1	0	0	0	0	2	0	0	6	10	7	0.00	0.02	45.11
	0.3	3	1	0	2	0	0	2	0	0	5	9	8	0.00	0.02	43.08
	0.5	8	1	0	6	1	0	6	1	0	8	9	8	0.00	0.02	40.34
	0.7	3	0	0	0	0	0	3	0	0	8	10	9	0.01	0.03	35.20
	0.9	1	0	0	0	0	0	3	1	0	7	10	7	0.00	0.02	42.21
0.3	0.1	2	0	0	0	0	0	0	0	0	9	10	10	0.00	0.02	40.12
	0.3	2	1	0	0	0	0	2	0	0	6	10	9	0.00	0.02	45.07
	0.5	0	0	0	2	0	0	3	0	0	6	9	9	0.01	0.03	41.23
	0.7	2	1	0	0	0	0	1	0	0	6	10	9	0.00	0.02	45.35
	0.9	2	0	0	0	0	0	0	0	0	4	10	9	0.00	0.02	32.18
0.5	0.1	6	1	0	0	0	0	0	0	0	9	10	10	0.00	0.03	39.32
	0.3	5	1	0	0	0	0	0	0	0	8	10	10	0.00	0.02	45.17
	0.5	0	0	0	0	0	0	0	0	0	9	10	9	0.00	0.02	47.12
	0.7	2	0	0	0	0	0	2	0	0	4	10	10	0.00	0.03	35.45
	0.9	0	0	0	0	0	0	0	0	0	7	10	10	0.00	0.03	37.50
0.7	0.1	4	0	0	0	0	0	0	0	0	9	10	10	0.00	0.03	38.12
	0.3	0	0	0	0	0	0	0	0	0	10	10	10	0.00	0.02	35.52
	0.5	1	0	0	0	0	0	0	0	0	9	10	10	0.00	0.03	37.29
	0.7	0	0	0	0	0	0	0	0	0	8	10	10	0.00	0.03	45.29
	0.9	4	0	0	0	0	0	0	0	0	6	10	10	0.00	0.03	41.53
0.9	0.1	0	0	0	0	0	0	0	0	0	10	10	10	0.00	0.03	33.46
	0.3	0	0	0	0	0	0	0	0	0	9	10	10	0.00	0.03	37.26
	0.5	0	0	0	0	0	0	0	0	0	8	10	10	0.00	0.03	34.31
	0.7	0	0	0	0	0	0	0	0	0	9	10	10	0.00	0.00	29.12
	0.9	0	0	0	0	0	0	0	0	0	8	10	10	0.00	0.03	35.49
	Avg.	2	0	0	0	0	0	1	0	0	7.52	9.88	9.36	0.00	0.02	39.28
	Total										188	247	234			

Table 5.9: Performance of the TS, ABC and VNS algorithms for the OAS-2 problem when n=15

		% Deviations from UB														
$n = 15$		TS			ABC			VNS			# of optimal solutions			Run Times		
τ	R	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	TS	ABC	VNS	TS	ABC	VNS
0.1	0.1	4	3	1	4	2	0	4	3	1	0	1	0	0.04	0.04	81.12
	0.3	8	3	0	7	3	1	8	4	1	1	0	0	0.01	0.03	82.39
	0.5	4	2	0	4	2	0	4	2	1	1	1	0	0.01	0.04	80.38
	0.7	4	2	0	4	1	0	4	2	0	1	5	1	0.01	0.03	83.49
	0.9	6	1	0	6	1	0	6	1	0	4	5	5	0.01	0.04	82.55
0.3	0.1	8	4	3	6	3	0	7	5	3	0	1	0	0.01	0.04	80.43
	0.3	11	5	2	11	4	1	11	4	0	0	0	0	0.01	0.04	88.11
	0.5	8	5	1	8	4	1	8	4	1	0	0	0	0.01	0.03	84.25
	0.7	8	4	2	8	3	1	8	4	1	0	0	0	0.01	0.04	81.32
	0.9	7	4	0	7	3	0	7	3	0	2	2	2	0.01	0.04	83.35
0.5	0.1	13	7	2	13	7	2	13	8	2	0	0	0	0.01	0.04	79.23
	0.3	14	8	4	14	8	4	14	7	3	0	0	0	0.01	0.05	85.11
	0.5	15	9	6	15	9	4	15	9	5	0	0	0	0.01	0.04	82.19
	0.7	11	6	1	11	6	0	11	6	1	0	0	0	0.01	0.05	86.58
	0.9	18	6	0	18	6	0	18	6	0	1	1	1	0.01	0.05	88.12
0.7	0.1	4	1	0	2	0	0	2	1	0	7	9	6	0.01	0.04	79.33
	0.3	3	1	0	0	0	0	0	0	0	7	10	8	0.01	0.05	81.24
	0.5	2	0	0	0	0	0	1	0	0	7	10	7	0.01	0.05	77.46
	0.7	8	2	0	8	1	0	8	1	0	3	6	5	0.01	0.05	81.44
	0.9	0	0	0	0	0	0	1	0	0	9	8	7	0.01	0.05	80.53
0.9	0.1	3	0	0	0	0	0	3	0	0	9	10	9	0.01	0.05	80.11
	0.3	5	0	0	0	0	0	0	0	0	9	9	10	0.01	0.06	82.34
	0.5	0	0	0	0	0	0	0	0	0	10	8	10	0.01	0.05	84.13
	0.7	0	0	0	0	0	0	1	0	0	10	7	9	0.01	0.05	79.34
	0.9	0	0	0	0	0	0	0	0	0	10	8	10	0.01	0.05	81.45
	Avg.	7	3	1	6	3	1	7	3	1	3.64	4.04	3.60	0.01	0.04	82.24
	Total										91	101	90			

Table 5.10: Performance of the TS, ABC and VNS algorithms for the OAS-2 problem when n=20

		% Deviations from UB														
$n = 20$		TS			ABC			VNS			# of optimal solutions			Run Times		
τ	R	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	TS	ABC	VNS	TS	ABC	VNS
0.1	0.1	5	3	1	4	2	0	5	3	0	0	2	0	0.10	0.06	134.56
	0.3	5	3	1	2	2	1	5	4	1	0	0	0	0.04	0.06	136.87
	0.5	3	2	1	3	1	0	4	2	1	0	1	0	0.03	0.05	128.39
	0.7	2	1	0	1	0	0	1	0	0	5	6	6	0.04	0.05	125.89
	0.9	4	1	0	6	3	1	4	1	0	3	7	5	0.03	0.07	122.56
0.3	0.1	8	5	2	7	4	2	9	5	2	0	0	0	0.03	0.07	128.10
	0.3	6	5	3	6	4	1	7	4	2	0	0	0	0.04	0.07	118.39
	0.5	8	5	2	6	4	1	7	5	2	0	0	0	0.03	0.07	115.72
	0.7	7	3	1	7	2	1	8	3	1	0	1	0	0.04	0.07	127.94
	0.9	5	2	0	5	2	0	7	3	0	2	2	0	0.04	0.08	125.01
0.5	0.1	8	6	4	6	5	2	7	6	4	0	0	0	0.03	0.08	135.77
	0.3	9	6	3	9	5	1	12	7	2	0	0	0	0.07	0.07	124.34
	0.5	10	7	3	9	5	2	10	7	4	0	0	0	0.04	0.09	123.64
	0.7	11	5	1	11	5	1	11	5	1	0	0	0	0.03	0.09	125.91
	0.9	11	6	0	11	6	0	11	6	0	1	1	1	0.05	0.07	135.67
0.7	0.1	15	10	5	14	9	5	13	10	5	0	0	0	0.05	0.09	134.73
	0.3	14	9	6	13	8	6	12	9	6	0	0	0	0.04	0.10	141.29
	0.5	20	9	0	20	9	0	20	9	0	2	1	2	0.04	0.09	127.91
	0.7	12	7	0	12	6	0	13	7	0	3	2	1	0.07	0.10	118.58
	0.9	13	8	0	13	8	0	12	7	0	1	0	0	0.04	0.08	134.39
0.9	0.1	1	0	0	0	0	0	1	0	0	0	10	8	0.06	0.11	140.01
	0.3	1	0	0	1	0	0	3	0	0	8	8	8	0.03	0.12	135.45
	0.5	1	0	0	1	0	0	2	0	0	8	7	8	0.03	0.11	139.58
	0.7	2	0	0	0	0	0	3	2	0	8	6	5	0.03	0.10	144.34
	0.9	13	2	0	13	1	0	13	2	0	6	4	3	0.03	0.11	115.83
	Avg.	8	4	1	7	4	1	8	4	1	1.88	2.32	1.88	0.04	0.07	129.64
	Total										47	58	47			

Table 5.11: Performance of the TS, ABC and VNS algorithms for the OAS-2 problem when n=25

		% Deviations from UB														
$n = 25$		TS			ABC			VNS			# of optimal solutions			Run Times		
τ	R	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	TS	ABC	VNS	TS	ABC	VNS
0.1	0.1	6	4	1	4	3	1	5	3	2	0	0	0	0.08	0.09	211.09
	0.3	6	3	2	6	2	1	5	3	2	0	0	0	0.06	0.09	229.45
	0.5	4	2	1	2	1	0	4	2	1	0	1	0	0.06	0.08	219.34
	0.7	4	1	0	3	1	0	3	1	0	4	6	4	0.06	0.10	223.01
	0.9	2	1	0	2	0	0	3	1	0	4	7	4	0.05	0.10	221.98
0.3	0.1	5	4	2	5	3	1	4	4	3	0	0	0	0.10	0.11	205.90
	0.3	7	5	3	6	3	1	7	5	3	0	0	0	0.09	0.10	219.79
	0.5	6	3	2	5	2	2	5	2	1	0	0	0	0.08	0.12	221.80
	0.7	6	2	1	5	2	1	6	3	2	0	0	0	0.08	0.12	234.67
	0.9	4	2	0	3	1	0	4	2	0	2	2	1	0.07	0.12	210.40
0.7	0.1	7	6	3	7	5	3	7	5	3	0	0	0	0.07	0.12	204.09
	0.3	9	5	3	7	4	3	8	5	3	0	0	0	0.08	0.12	228.30
	0.5	8	5	2	6	4	2	9	4	1	0	0	0	0.09	0.16	245.54
	0.7	11	6	2	12	3	0	9	5	2	0	3	0	0.08	0.19	263.58
	0.9	7	4	1	7	3	1	7	4	1	0	0	0	0.08	0.11	206.45
0.7	0.1	18	9	3	16	8	1	17	8	2	0	0	0	0.06	0.14	210.49
	0.3	14	10	7	12	9	5	15	9	6	0	0	0	0.08	0.11	219.88
	0.5	15	12	7	14	10	5	14	11	6	0	0	0	0.08	0.18	215.61
	0.7	14	8	2	12	7	2	13	8	2	0	0	0	0.07	0.15	251.65
	0.9	15	10	3	14	8	0	14	9	1	0	1	0	0.08	0.15	239.10
0.9	0.1	6	1	0	5	1	0	5	1	0	5	8	6	0.06	0.18	230.17
	0.3	0	0	0	1	0	0	1	0	0	8	6	7	0.06	0.17	215.50
	0.5	12	4	0	12	3	0	12	3	0	3	3	5	0.07	0.19	244.58
	0.7	25	8	0	21	7	0	22	7	0	4	4	5	0.08	0.16	224.66
	0.9	22	7	0	19	6	0	21	6	0	4	2	3	0.09	0.17	201.51
	Avg.	9	5	2	8	4	1	9	4	1	1.36	1.72	1.40	0.07	0.13	223.94
	Total										34	43	35			

Table 5.12: Performance of the TS, ABC and VNS algorithms for the OAS-2 problem when n=50

		% Deviations from UB														
$n = 50$		TS			ABC			VNS			# of optimal solutions			Run Times		
τ	R	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	TS	ABC	VNS	TS	ABC	VNS
0.1	0.1	3	2	1	3	2	1	3	2	1	0	0	0	1.19	0.68	518.18
	0.3	4	2	1	2	2	1	4	3	1	0	0	0	0.98	0.60	603.55
	0.5	2	2	1	2	1	0	2	2	1	0	0	0	0.89	0.80	619.68
	0.7	16	3	0	16	2	0	17	2	0	1	4	1	0.58	0.73	672.19
	0.9	2	1	0	0	0	0	1	0	0	2	10	3	0.46	1.14	642.66
0.3	0.1	3	3	2	4	3	2	4	4	3	0	0	0	1.61	0.69	615.81
	0.3	5	4	3	4	3	2	5	4	2	0	0	0	0.09	0.10	641.80
	0.5	5	3	1	4	2	1	5	2	1	0	0	0	0.08	0.12	608.07
	0.7	3	1	0	2	1	0	3	1	0	1	1	1	0.08	0.12	621.13
	0.9	3	1	0	2	1	0	2	1	0	0	3	0	0.07	0.12	610.89
0.7	0.1	5	4	3	4	3	2	6	3	2	0	0	0	1.36	0.64	629.22
	0.3	8	6	3	6	4	3	6	5	3	0	0	0	1.18	0.48	630.55
	0.5	8	4	2	7	4	2	8	4	2	0	0	0	1.35	0.77	654.08
	0.7	5	3	2	4	2	1	5	2	1	0	0	0	1.22	1.02	689.80
	0.9	6	4	2	4	2	1	6	3	2	0	0	0	1.16	0.72	622.13
0.7	0.1	9	7	4	6	5	3	7	6	4	0	0	0	1.44	0.97	690.04
	0.3	9	6	4	6	4	5	8	4	5	0	0	0	1.52	0.95	619.23
	0.5	13	9	7	11	7	5	14	10	7	0	0	0	1.59	0.94	679.90
	0.7	18	9	2	15	8	2	16	8	3	0	0	0	1.27	0.95	651.85
	0.9	18	11	6	14	8	4	17	11	5	0	0	0	1.15	1.09	628.19
0.9	0.1	18	13	8	17	11	7	18	12	7	0	0	0	1.25	1.48	689.10
	0.3	23	17	13	18	14	9	21	17	9	0	0	0	1.26	1.32	701.67
	0.5	23	17	10	17	13	3	21	16	7	0	0	0	1.42	1.19	655.15
	0.7	21	16	11	18	12	6	19	15	4	0	0	0	1.43	1.24	724.16
	0.9	19	16	11	16	12	10	19	16	4	0	0	0	1.37	1.03	781.21
	Avg.	10	7	5	8	5	3	9	7	3	0.16	0.72	0.24	1.19	0.89	620.06
	Total										4	18	5			

Table 5.13: Performance of the TS, ABC and VNS algorithms for the OAS-2 problem when n=100

		% Deviations from UB														
$n = 100$		TS			ABC			VNS			# of optimal solutions			Run Times		
τ	R	Max	Avg.	Min	Max	Avg.	Min	Max	Avg.	Min	TS	ABC	VNS	TS	ABC	VNS
0.1	0.1	3	2	1	2	2	1	3	2	1	0	0	0	16.76	3.98	1034.32
	0.3	3	2	2	2	1	1	3	1	1	0	0	0	17.26	8.22	1090.55
	0.5	3	1	0	1	1	0	2	1	0	0	0	0	10.87	7.61	1055.11
	0.7	1	0	0	0	0	0	1	0	0	1	6	1	6.21	6.25	1072.19
	0.9	1	0	0	0	0	0	1	0	0	4	10	4	3.53	9.08	1042.66
0.3	0.1	4	3	1	3	2	1	4	3	2	0	0	0	22.37	4.89	1038.71
	0.3	5	3	2	4	2	5	3	3	2	0	0	0	20.10	5.29	949.04
	0.5	4	2	1	2	2	1	4	2	1	0	0	0	16.77	4.88	1010.11
	0.7	3	2	0	1	1	0	3	1	0	0	1	0	9.48	6.16	1981.56
	0.9	2	1	0	1	0	0	1	1	0	0	4	0	7.58	8.59	1060.01
0.7	0.1	5	4	2	4	3	3	6	3	2	0	0	0	25.96	5.27	1139.22
	0.3	6	4	3	4	3	2	7	3	2	0	0	0	28.87	6.94	1030.51
	0.5	5	4	3	4	3	2	5	3	2	0	0	0	20.56	5.86	954.08
	0.7	4	3	2	3	2	1	4	2	1	0	0	0	15.57	6.57	1129.37
	0.9	5	2	1	3	1	1	5	2	2	0	0	0	12.15	6.76	1122.91
0.7	0.1	6	5	3	5	4	3	6	4	3	0	0	0	33.60	6.04	1095.12
	0.3	11	7	4	7	5	2	10	7	5	0	0	0	26.62	6.25	1175.26
	0.5	13	6	4	12	5	3	13	6	3	0	0	0	22.30	6.77	1079.45
	0.7	13	7	3	7	5	3	11	7	3	0	0	0	26.36	6.78	1045.38
	0.9	13	8	5	8	6	4	13	9	5	0	0	0	17.84	7.42	1058.67
0.9	0.1	12	9	7	9	7	5	12	9	8	0	0	0	19.13	13.63	1199.62
	0.3	23	15	9	12	9	5	21	16	11	0	0	0	26.32	9.82	1234.23
	0.5	18	16	13	17	12	9	17	14	12	0	0	0	21.51	6.46	1155.56
	0.7	20	16	11	13	11	8	17	14	11	0	0	0	22.70	7.52	1215.30
	0.9	22	16	12	17	11	4	19	16	13	0	0	0	17.48	8.61	1326.27
	Avg.	8	6	4	6	4	2	7	6	4	0.20	0.84	0.24	19.13	7.03	1088.21
	Total										5	21	5			

5.5 Analysis of the Results

In this section, we summarize the results of our computational experiments, which are presented in Tables 5.5-5.13 for the OAS-1 and OAS-2 problem, and analyze them. Although the proposed VNS algorithm is run on a similar computer with the computers used in benchmarks, the run time may vary due to programming languages. Hence, in this section the run times are not reported for comparison purposes.

5.5.1 Analysis of the results for the OAS-1 problem

Comparison of the VNS algorithm with the Myopic Algorithm

Table 5.5 displays the results of the VNS and the myopic algorithm for the data set $n = 50$. The average run time for the VNS algorithm is 172.43 seconds. We observe that the VNS achieves 0.499 % deviation on the average whereas the myopic algorithm gives 0.566% deviation. The VNS also outperforms the myopic algorithm with respect to the number of optimal solutions found. Although the myopic algorithm finds the optimal solution in 13 out of 100 instances, the VNS algorithm improves this result by finding optimal solution in 31 out of 100 instances.

The results of the VNS and the myopic algorithm for $n = 75$ are tabulated in Table 5.6. The VNS algorithm gives the results in 263.37 seconds on the average. Although average deviation values are very close to each other (1.342% for the VNS and 1.362% for the myopic algorithm), the VNS is more effective than the myopic algorithm according to the number of optimal solutions found. The VNS algorithm finds the optimal solution in 6 out of 100 instances whereas the myopic algorithm finds the optimal solution in only 3 out of 100 instances.

Table 5.7 contains the result of the VNS and the myopic algorithm for $n = 100$. The VNS runs in 541.66 seconds on the average. When the problem size gets larger, the performance of both the VNS and the myopic algorithm get worse which is an expected case because the problem gets harder. We see that the myopic algorithm gives 1.411% deviation on the average. Similarly, the VNS algorithm gives 1.390% deviation from the UB. The VNS algorithm is still better in terms of the number of optimal solutions found. While the myopic algorithm finds 4 optimal solutions out of 100 instances, the VNS finds 5 optimal solutions out of 100 instances.

The reasons for the success of the VNS on the myopic algorithm can be twofold:

- The nature of the VNS allows changing the neighborhood structures which facilitates searching the solutions in different spaces.

- The myopic algorithm solves the OAS-1 problem in two steps: determination of the accepted orders from the relaxation of the problems and reassembling the accepted orders, respectively, whereas the VNS algorithm handles the problem by considering both acceptance and sequencing decisions simultaneously.

Comparison of the VNS algorithm with the GA

Table 5.5 shows that the VNS algorithm is competitive with the GA in terms of the number of optimal solutions achieved by finding 31 optimal solutions out of 100 instances, while the GA finds 32 optimal solutions out of 100 instances for the data set $n = 50$. However, we see from the same table that the GA gets a better performance by giving 0.265% deviation, since the VNS gives 0.499% deviation.

Table 5.6 reports the results of the VNS and the GA applied on the data set $n = 75$. The GA outperforms the VNS algorithm in terms of both average deviation and the number of optimal solutions found. The GA finds 14 optimal solutions out of 100 instances with 0.605% average deviation whereas the VNS algorithm finds 6 optimal solutions out of 100 instances with 1.342% average deviation.

For n=100, we observe a similar performance for the VNS algorithm in Table 5.7. The VNS algorithm finds the optimal solution in 5 out of 100 instances and gives 1.390% deviation on the average. However, the GA still outperforms the VNS by finding the optimal solution in 16 out of 100 instances and gives 0.605% deviation on the average. We observe that when the problem size increases, the performances of both the VNS algorithm and the GA decrease since the difficulty of the problem increases with respect to its size.

We present the findings about the comparison of the VNS algorithm with the GA as follows. The GA is a population based metaheuristic method. It can use the possibility of recombining several solutions into a new one by achieving search space diversification. However, the VNS provides changing the search space, if an improvement is not observed during the search. This situation can lead getting stuck in local optimum solution for some instances solved in the VNS algorithm.

Effect of parameters τ and Crl on the test instances for the OAS-1 problem

For the OAS-1 problem, we use six data sets created with different tardiness factors which are set to 0.3, 2.0 and 3.0 and the generated revenues for the half of the problems correlating with due date and revenue, and for the other half, revenues are inversely correlated with each due date for each other. The value of τ implies the tightness of the due dates. When the τ value increases, the tightness of the corresponded due date also increases. We observe from Tables 5.5-5.7 that the VNS algorithm generally performs worse, when the τ value rises. The problem is more difficult with tighter due dates. This is an expected case since orders with tighter due dates are more difficult to meet than loose due dates. The performance of the VNS shows a similar pattern for the instances generated with correlation and no correlation so we cannot get an idea about the hardness of the problem with respect to correlation property.

5.5.2 Analysis of the results for the OAS-2 problem

Comparison of the VNS algorithm with the TS algorithm

Table 5.8 reports the results for the VNS and the TS algorithms for the data test $n = 10$. The VNS algorithm runs in 39.28 seconds to solve the instances with $n = 10$. Although both the VNS and the TS algorithms give 0% deviation on the average (it implies that the gathered solutions are very close to the optimal values), the VNS algorithm outperforms the TS algorithm in terms of the number of the optimal solutions found. The VNS algorithm obtains 234 optimal solutions out of 250 instances, while TS algorithm finds 188 optimal solutions.

From the Table 5.9, we see that the VNS and the TS algorithms perform similarly in terms of the average deviation and the number of solutions found for the data set $n = 15$. Both the VNS and the TS algorithms give 3% deviation on the average. The TS algorithm finds the optimal solution in 91 out of 250 instances whereas the VNS algorithm finds 90 optimal solutions. The VNS algorithm gives the solutions of the instances for the data set in 82.24 seconds on the average.

For the data set $n = 20$, we observe the same performance for the VNS and the TS algorithms. Both algorithms find the optimal solution in 47 out of 250 instances and produce a deviation of 3% on the average as can be seen from Table 5.10. The average time for the

VNS algorithm to find the solutions is 129.64 seconds.

For $n = 25$, the VNS algorithm runs in 223.94 seconds. We see from Table 5.11 that it gives 4% deviation on the average while finding 35 optimal solutions out of 250 instances. In contrast, the TS gives a higher percentage deviation (5%) than the VNS while obtaining 34 optimal solutions out of 250 instances.

When the problem size increases to 50, the VNS algorithm runs in 620.06 seconds on the average. From Table 5.12, we observe that the VNS and the TS algorithms show the same performance in terms of average deviation. Both of them achieve 7% deviation on the average. Although TS finds the optimal solution in 4 out of 250 instances, the VNS algorithm outperforms the TS algorithm by obtaining 5 optimal solutions.

When n=100, performance of the VNS and the TS algorithms are not different. We observe from Table 5.13 that the VNS and the TS algorithms obtain 5 optimal solutions out of 250 test instances. Furthermore, both of the algorithms perform 6% deviation on the average. In order to obtain the results, the VNS runs in 1088.21 seconds on the average.

The results presented above indicate that the VNS algorithm is competitive with the TS algorithm and outperforms the TS results when n equals $10, 25$ and 50 in terms of the number of optimal solutions found. They generally show the same performance on the average deviation. Moreover, when the problem size is increased, both of the algorithm do worse since the complexity of the problem increases with its size. The major differences between the TS and the VNS algorithms to explain the success of the VNS algorithm are as follows. The VNS algorithm uses an additional move operator: reverse order and employed local search procedure starting from a random solution in a given neighborhood. These properties may help finding optimal solutions through the search which are not reachable in the search space of the TS algorithm.

Comparison of the VNS algorithm with the ABC algorithm

From Table 5.8, we see that the ABC algorithm dominates the VNS algorithm in terms of the number of optimal solutions obtained for the data set $n = 10$. The ABC algorithm finds 247 optimal solutions out of 250 instances, while the VNS obtains 247 optimal solutions. However, they show the same performance by achieving 0% on the average deviation.

For $n = 15$, the ABC algorithm still outperforms the VNS algorithm with respect to the

number of optimal solutions found. We see from Table 5.9 that the VNS finds 90 optimal solutions out of 250 instances whereas the ABC finds the optimal solution in 101 out of 250 instances. They are still comparable since both algorithms give 3% deviation on the average.

Table 5.10 reports the results obtained from data set $n = 20$ for both the VNS and the ABC algorithms. We see that the VNS and the ABC algorithms give the same average deviation (4%) for the data set $n = 20$, even though the ABC obtains more optimal solutions than the VNS algorithm. The VNS algorithm finds 47 optimal solutions out of 250 instances whereas the ABC algorithm finds 58 optimal solutions.

From Table 5.11, we observe that the VNS algorithm finds 35 optimal solutions out of 250 instances, while the ABC finds 43 optimal solutions out of 250 instances. The average deviation is still 4% for both of the algorithms, although the problem size is increased to 25 from 20.

When there are 50 incoming orders, the ABC algorithm starts dominating the VNS algorithm in terms of the optimal solution and the average deviation. According to the results given in Table 5.12, the ABC algorithm achieves to find the optimal solution in 18 out of 250 instances and gives 4% deviation on the average whereas the VNS gives 7% deviation on the average by finding the optimal solution in 5 out of 250 instances.

We observe from Table 5.13 that the solution quality of the ABC algorithm is still better than that of the VNS algorithm by finding a greater number of optimal solutions and less deviation on the average. The ABC algorithm gives only 4% deviation on the average by obtaining 21 optimal solutions, while the VNS gives 6% on the average by obtaining 5 optimal solutions.

We make some inferences from the results explained above. They are given below.

- The range for the average deviation obtained by the VNS algorithm is between 0%-7% and this value is between 0%- 5% for the ABC algorithm. This case indicates that although the ABC algorithm outperforms the VNS algorithm, the proposed VNS algorithm is robust and has an acceptable variation in solution quality as the ABC algorithm.

- The reason for the ABC algorithm dominates the VNS algorithm is probably that the

ABC algorithm applies combination of swap and insertion operators simultaneously and iteratively, while the VNS applies the swap and reverse order separately. Although changing neighborhood structures enable diversification mechanism by searching the solutions in different search spaces, the gathered solutions from the VNS algorithm may be far from the optimal solutions for some instances.

- Notably, both of the ABC algorithm and the GA algorithm which dominate the proposed VNS algorithm for the OAS problems are population based and include crossover operators. We infer that more diversification can be included in the VNS algorithm to improve its efficiency, since population based algorithms on the OAS problem are more successful.

Effect of parameters τ and R on test instances

When τ gets larger, the release dates are generated on a wider interval and tightness of the due date values increase. R indicates the tightness of the deadlines with respect to the due dates. If R increases, the difference between the deadline and the due date increases. The performance of the algorithm is very changeable according to increasing of R values with a constant τ. Similarly, changing τ values with a constant R value does not effect the performance of the algorithm in a pattern. We observe from the results given in Tables 5.8-5.13 that varying τ and R values does not give an idea about the hardness of the problem.

Chapter 6

CONCLUSIONS AND FUTURE RESEARCH

6.1 Conclusions

In this thesis, we study two order acceptance and scheduling problems on a single machine: OAS-1 and OAS-2. We differentiate the problems with respect to the properties of the incoming orders. Each order is defined with a weight, a processing time, a revenue and a due date for the OAS-1 problem whereas an order has additional properties in the OAS-2 problem which are the sequence dependent setup time, the release date and the deadline. Tardiness is included in both of the problems. An order becomes a tardy order if it is delivered to the customer after its due date and the manufacturer gives a discount for the tardy order on behalf of the customer. Moreover, if the customer specifies a deadline for the order as in the OAS-2 problem, she refuses the order and does not pay for the order which is completed after its deadline. The objective of the problems is to maximize the total profit. The profit is a function of the total revenues and total weighted tardiness.

We propose a competitive improvement heuristic to solve the problems since they are strongly NP-hard. Namely, the proposed heuristic is variable neighborhood search (VNS). The proposed VNS gives acceptable results for both of the problems. We understand that the proposed VNS is a flexible algorithm which can be applicable to different problems. We conduct preliminary tests to decide on the parameters of the algorithm. We decide to develop the VNS with two neighborhood structures and an intelligent local search procedure according to the results of the preliminary tests. We explain the possible reasons why the preliminary test gives its best with the corresponded structure as follows. The first neighborhood structure, swap, provides a random exchange by changing both the set of accepted orders and their sequence. The second neighborhood structure, reverse order, enables swapping of two random orders in which the positions of all orders between swapped ones are reversed to have a new schedule. Randomness provides diversification to the algorithm. In contrast, intensification behavior is observed in the algorithm while making a decision

about moving to the next neighborhood structure. We move to the next neighborhood structure with a deterministic way in which we change the neighborhood structure if we have a better solution than the incumbent. The local search provides a compound move which is the combination of three moves: drop, add and insert. It starts to search from a random solution in the neighborhood structures. When we apply swap or reverse order operations, the completion time of the orders, hence the tardiness and the revenue of the order can be affected due to the release date, the sequence dependent setup time, the due dates and the deadline therefore a feasibility check is required after each modification to the current solution. The feasibility check enables deleting some orders from the sequence. Deleting the orders is critical to vary the number of accepted orders in the current sequence. This deletion can cause to converge a local optimum solution. The local search procedure provides a compact form to overcome this situation. The number of the accepted orders can be changed due to the local search since it continues adding the orders if an improvement is observed in the solution, while the move operators do not provide a change in the number of accepted orders. Moreover, we use a threshold value to accept an inferior solution through the search which enables generation of diversified solutions.

We compare the performance of the VNS algorithm with the myopic and the genetic algorithm (GA) for the OAS-1 problem and with the tabu search (TS) and the artificial bee colony (ABC) optimization for the OAS-2 problem. While the VNS algorithm is competitive with the myopic and the TS algorithm, the ABC and the GA outperforms the VNS.

We observe that the proposed VNS is a trajectory based algorithm which may cause missing some parts of the search space. However both of the GA and the ABC algorithms are population based metaheuristics involving crossover and probably overcome this problem.

6.2 Future Research

The following recommendations are made for the future search. An additional efficient metaheuristic can be developed for these problems. We infer from Section 5.5 that the VNS can be strengthened with a population based heuristic. The GA including the VNS as a move operator can be future search directions.

Furthermore, different neighborhood structures can be developed to improve the success of the VNS algorithm for OAS problems.

BIBLIOGRAPHY

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

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

J.B. Ghosh. Job selection in a heavily loaded shop. *Computers Operational Research*, 24: 141–145, 1997.

P. Hansen and N. Mladenovic. Variable neighborhood search: Principles and applications. *European Journal of Operation Research*, 130:449–467, 2001.

G. Kirlik and C. Oğuz. 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, 2012.

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

L. Liu and H. Zhou. Applying variable neighborhood search to the single-machine maximum lateness rescheduling problem. *Electronic Notes in Discrete Mathematics*, 39:107–114, 2012.

N. Mladenovic and P. Hansen. Variable neighborhood search. *Computers Operational Research*, 17:52–67, 1997.

R. Mohan, V. Rachamadugu, and T.E. Morton. Myopic heuristics for the weighted tardiness problem on identical parallel machines. *Computers and Operations Research*, 10:83–109, 1983.

T.E. Morton, R.V. Rachamadugu, and A. Vepsalainen. *Accurate myopic heuristics for tardiness scheduling*. University of Michigan. Graduate School of Business Administration, 1993.

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

S. Panwalkar, R. Dudek, and M. Smith. The lessons of flow shop scheduling research. *Symposium on the theory of scheduling and its applications*, 7:29–38, 1973.

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

S.A. Slotnick and T.E. Morton. Selecting jobs for a heavily loaded shop with lateness penalties. *Computers Operational Research*, 23:131–140, 1995.

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

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

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

VITA

Ayşegül Altındağ was born in Uşak, Turkey on May 26, 1988. She graduated from Uşak Science High School in 2006. She received his B.Sc. degree with double major in Chemical Engineering and in Systems Engineering from Yeditepe University, Istanbul,Turkey in 2011. Same year, she became M.Sc. student in Industrial Engineering at Koç University. She has recently presented this study for the workshop CSW 2013 in Istanbul, Turkey and the conference MAPSP 2013 in Nancy, France. Next year, she will be a PDEng at Eindhoven University of Technology in Eindhoven, Netherlands.