COMPARISON OF TWO PROCESSING APPROACHES FOR SOLVING A CUSTOMER ORDER SCHEDULING PROBLEM

PINAR YELOĞLU

FEBRUARY 2020

COMPARISON OF TWO PROCESSING APPROACHES FOR SOLVING A CUSTOMER ORDER SCHEDULING PROBLEM

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES OF
ÇANKAYA UNIVERSITY

BY

PINAR YELOĞLU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

THE DEPARTMENT OF INDUSTRIAL ENGINEERING

FEBRUARY 2020

STATEMENT OF NON PLAGIARISM

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : Pınar Yeloğlu

Signature :

Date :

ABSTRACT

COMPARISON OF TWO PROCESSING APPROACHES FOR SOLVING A CUSTOMER ORDER SCHEDULING PROBLEM

YELOĞLU, Pınar

M.Sc., Department of Industrial Engineering

Supervisor: Prof. Dr. Ferda Can ÇETİNKAYA

February 2020, 83 pages

This study considers a customer order scheduling (COS) problem in which each customer requests a variety of products (jobs) processed on a single machine. A sequence-independent setup for the machine is needed before processing each product. All products in a customer order are delivered to the customer when the processing of these products is completed. The completion time of the product processed as the last product in a customer order defines the completion time of the customer order. We aim to find the best schedule of the customer orders and the products to minimize the total completion time of the customer orders. We have studied this customer order scheduling problem with order-based and job-based processing approaches. We have developed two mixed-integer linear programming models, which are capable of solving the small and medium-sized problem instances optimally for the job-based processing approach, which has not been studied in the literature, and a heuristic algorithm for large-sized problem instances. The results of our empirical study show that our tabu-search based heuristic algorithm gives optimal or near-optimal solutions in a very short time. In addition, we have compared the order-based, and job-based processing approaches for both setup and no-setup cases.

Keywords: Customer order scheduling; order-based processing, job-based processing, total completion time; mixed integer linear programming; tabu search

ÖZ

MÜŞTERİ SİPARİŞLERİNİ ÇİZELGELEME PROBLEMİ İÇİN İKİ İŞLEME YAKLAŞIMININ KARŞILAŞTIRILMASI

YELOĞLU, Pınar

Yüksek Lisans, Endüstri Mühendisliği

Anabilim Dalı Tez Yöneticisi: Prof. Dr. Ferda Can ÇETİNKAYA

Şubat 2020, 83 sayfa

Bu çalışma, her bir müşterinin tek bir makinede işlenen çeşitli ürünleri (işleri) talep ettiği siparişlerin çizelgelenmesi problemini ele almaktadır. Her bir ürünü işlemeden önce makine için sıra-bağımsız bir hazırlık (kurulum) gereklidir. Bir müşterinin siparişindeki tüm ürünler, bu ürünlerin işlenmesi tamamlandığında müşteriye teslim edilir. Bir müşteri siparişinde son ürün olarak işlenmiş ürünün tamamlanma süresi müşteri siparişinin tamamlanma süresini belirler. Amacımız, müşteri siparişlerinin toplam tamamlanma süresini en aza indirmek için müşteri siparişleri ve ürünlerinin en iyi çizelgelemesini belirlemektir. Bu müşteri siparişlerini çizelgeleme problemini sipariş bazlı ve ürün bazlı işleme yaklaşımları ile çalıştık. Literatürde çalışılmamış olan ürün bazlı işlem yaklaşımı için küçük ve orta ölçekli problemleri en iyi şekilde çözebilen iki tane karışık tamsayılı doğrusal programlama modeli ile büyük ölçekli problemler için bir tabu arama esaslı sezgisel bir algoritma geliştirdik. Ayrıca, hazırlık sürelerinin olduğu ve olmadığı durumlar için sipariş ve iş bazlı işleme yaklaşımlarını karşılaştırdık.

Anahtar Kelimeler: Müşteri siparişlerini çizelgeleme; sipariş bazlı işleme, iş bazlı işleme, toplam tamamlanma süresi; karışık tamsayılı doğrusal programlama; tabu arama

ACKNOWLEDGEMENTS

There are several people that I would like to acknowledge for their help and support throughout my MSc. Program. First, I would like to express my sincere gratitude to my supervisor Professor Ferda Can Çetinkaya. He has patiently advised and helped me stay on track. His careful editing contributed enormously to the production of this thesis.

I would also like to express my appreciation to Hale Akkocaoğlu Çatmakaş for her constant help and advice, which were determinant for the work presented in this thesis.

I am also grateful to İbrahim Ediz for his help in computer programming issues, which made my computational studies smoother. He has been there, providing his heartfelt support and helping out whenever questions of doubt arose during the process.

Words are inadequate to express my gratitude to my friends, Melike Ekren, Sevginur Bostan, Zeynep Korkmaz, and Eylül Öykü Demir for their endless support, motivation, and unforgettable friendships.

Last but not least, I have limitless thanks to my parents. Their support and love was my driving force. To them, I dedicate this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

COS	Customer Order Scheduling
GAMS	General Algebraic Modelling System
GS	Group Scheduling
GT	Group Technology
JBP	Job-based Processing
MILP	Mixed Integer Linear Programming
NEH	Nawaz, Enscore, Ham
NP	Non-deterministic Polynomial-time
OBP	Order-based Processing
SCO	Sequence of Customer Orders
SPT	Shortest Processing Time
STT	Shortest Total Time
TS	Tabu Search
TT	Total Time

CHAPTER 1

INTRODUCTION

Most of the existing research on classical scheduling problems, except the *customer order scheduling* (COS) problem, assumes that there is a single customer that orders multiple different products (jobs), or there are multiple orders, each of which consists of only a single product (job). However, in a real-world make-to-order manufacturing system, there are multiple customer orders, in which each order is a collection of several products (jobs) that are often produced in a job lot consisting of many customer orders demanding the same product. In such a system, an order is shipped as a group to the customer, but only on the completion time of the last job of that order (Liu, 2009). In the COS, the problem is to satisfy the demand of several customers, who give orders with a set of several products (jobs) having different quantities, by optimizing the scheduling performance (objective).

In manufacturing environments, there are two extreme processing approaches for producing the products: the *order-based processing* (OBP), and the *job-based processing* (JBP). In the order-based processing, which is most frequently used in previous COS studies in the literature, all different products in a customer order form an order lot (group) and all products in this order lot are processed successively without intermingled with products of other customer orders (Yang, 2011). In other words, if the processing of a product in a customer order starts on a machine, then all different products within that customer order should be processed before switching the machine to process the products of another customer order. This processing approach follows the so-called *group technology* (GT) assumption. Decisions in the order-based processing are made to simultaneously determine the sequence of the customer order lots and the sequence of products (jobs) in each customer order lot. However, in job-based processing, the same products from different customer orders form a product lot and are processed successively without intermingled with other products. In other words, all customer orders for a product should be processed before switching the machine to process the customer orders for another product. Decisions in the job-based processing are made to simultaneously determine the sequence of the products (jobs) and the sequence of customer orders in each job. While the order-based processing aims to manage the customer orders on the shop floor easily, the job-based processing aims to reduce the negative effect of the job setups, especially when setup times required before processing the products are significantly large.

The optimal solution of the COS problem with order-based processing in a single-machine environment is easy and polynomial-time solvable, as shown in Section 2.2 when the scheduling performance is to minimize the total completion time, which is the sum of the completion times of the customer orders and is equivalent to minimizing total work-in-process inventory focusing on increasing the customer satisfaction. For the same scheduling performance, the COS problem with job-based processing is, however, not as easy as the problem with order-based processing. Thus, in our study, we will focus on the job-based processing problem, in which the aim is to determine a schedule that gives both the sequence of products (jobs) and the sequence of customer orders in each job sequence to minimize the total completion time of the customer orders. Furthermore, it is clear that the objective function values of the COS problem with these two extreme processing approaches are expected to be different so that, in our study, the order-based and job-based processing approaches for a single machine with both setup and no-setup cases will also be compared.

There are several contributions of our study. First, to the best of our knowledge, no previous research has considered our particular COS problem with the job-based processing for a single machine to minimize the total completion time, and we aim to contribute to the customer order literature in this direction. Second, we formulate a mixed-integer linear programming (MILP) model to solve the COS problem under consideration optimally. Third, our proposed heuristic algorithm for solving our COS problem is easy to implement for finding optimal and near-optimal solutions for medium and large-sized problem instances in which a solution cannot be obtained by solving the MILP model. Finally, we compare the order-based and job-based processing approaches.

The rest of this thesis is organized as follows. Chapter 2 defines the customer order scheduling problems with the order-based and job-based processing on a single-machine in detail and presents some structural properties of the optimal schedules for both problems. Chapter 3 provides a brief review of the works most relevant to our study on customer order scheduling. An MILP model and a tabu-search based heuristic algorithm for solving the COS problem with job-based processing are presented in Chapter 4. We give our empirical studies to evaluate the performances of the MILP model and the heuristic algorithm, as well as the comparison of the order-based and job-based processing approaches, in Chapter 5. Finally, in Chapter 6, we discuss the main findings of our study and several directions for future research.

CHAPTER 2

PROBLEM DEFINITION AND PRELIMINARY RESULTS

In this chapter, we first define our order-based and job-based processing problems in a joint statement in detail. Then, we present a numerical example to illustrate two extreme processing approaches, and finally establish some preliminary results that provide the basis for our analysis.

2.1. Problem Definition

For a planning period, consider a scheduling problem of K customers $(i = 1,2, \dots, K)$ in which each customer i gives an order O_i with one or more products (jobs) from a set of N jobs. Each customer order O_i has a demand for $D_{i,j}$ units of identical items of product j. A sequence-independent setup with s_j time units is needed to set up the machine before processing the product j. Sequence-independent setup means that the setup time is dependent only on the product to be processed next and is independent of the previous product. Each job has only one operation to be processed by a single machine, and the unit-processing time of the product j on the machine is p_j time units. All products (jobs) ordered by the same customer must be processed consecutively if the order-based processing approach is used. However, when the job-based processing approach is used, all customer orders for the same product must be processed consecutively. The following additional assumptions will be considered in describing our problem:

- All customer orders are available for processing at the same time, say time 0.
- The machine is available continuously from time zero onwards, with no breakdowns or maintenance delays, to process the products.
- The setup cannot be performed while the machine is processing a job.
- The machine can process, at most, one job at a time.

- No precedence relations among the jobs exist.
- No priorities among the customers exist.
- Job processing cannot be interrupted; i.e., no preemption is allowed.
- All parameters are known with certainty and not subject to any change; i.e., the scheduling problem is deterministic and static.

A completed product within a customer order has to wait until all finished products are being combined with other products belonging to the same customer order and shipped as a complete order. That is, each order is delivered to the customer when the processing of all products within that customer order is completed. Thus, the completion time of the product processed as the last product in a customer order defines the completion time of the customer order. Our goal is to find:

- a schedule with a sequence of customer orders and the sequence of jobs in each customer order when order-based processing approach is used, and
- a schedule with a sequence of the jobs and the sequence of the customer orders in each job when job-based processing approach is used,

so that the total completion time of the customer orders is minimized to increase the customer satisfaction in both processing approaches.

The order-based processing approach can be investigated in two forms:

- "Order-based processing without setup saving" in which setup time is required for each transition between products (jobs) while processing customer orders successively on a machine, and
- "Order-based processing with setup saving" in which setup times are eliminated between products (jobs) while processing customer orders successively on a machine.

2.2. An Illustrative Example

Before we proceed with our analysis, it seems appropriate to illustrate two extreme processing approaches by a numerical example. Consider a simple instance of the problem in which there are three customer orders and four products (jobs). Each customer gives order with a set of several products (jobs) having setup and unit processing times, as in Table 1. For example, Customer 1 gives an order with 10, 5, and 20 units of Products 2, 3, and 4, respectively.

Table 1 Customer orders, setup times, and unit-processing times

Jobs (Products)	Demand (in units) of the customer orders			Setup time	Unit-processing time
	O_1	O_2	O_3		
J_1	-	5	15	10	1
J_2	10	-	-	10	4
J_3	5	15	5	10	2
J_4	20	-	15	10	1

In Figures 1(a) and 1(b), the optimal schedules are illustrated for the order-based processing approach when there is no setup saving and a setup saving, respectively. In Figure 1(c), the optimal schedule for the job-based processing approach is given, and the setup and processing times are illustrated by the gray and blank blocks, respectively.

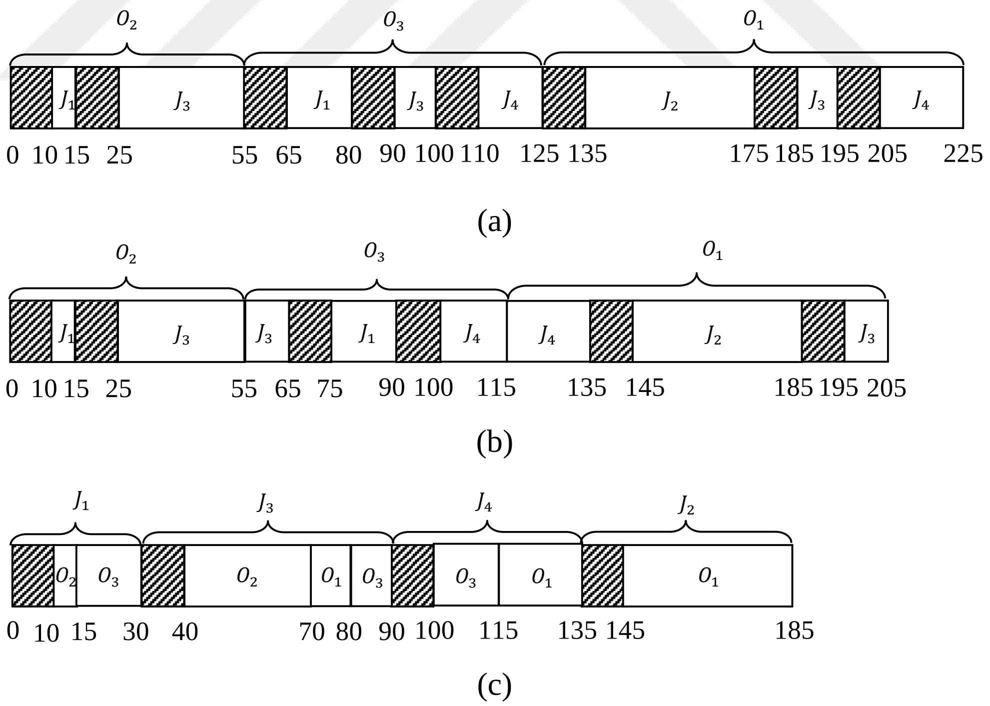


Figure 1 Gantt chart for the example problem: (a) with order-based processing having no setup savings; (b) with order-based processing having setup savings; and (c) with job-based processing

The optimal sequence of the customer orders is $O_2 - O_3 - O_1$, in which the customer orders 1, 2, and 3 are completed at 225, 55, and 125 time units, respectively, and the total completion time of the customer orders is 55 + 125 + 225 = 405 time units. However, if we allow processing common jobs successively when switching over customer orders on a machine, the total completion time is decreased due to setup-time savings of jobs. The optimal sequence of the customer orders is $O_2 - O_3 - O_1$, in which the customer orders 1, 2, and 3 are completed at 205, 55, and 115 time units, respectively, and the total completion time is reduced to 55 + 115 + 205 = 375 time units.

On the other hand, when we solve the problem with the job-based processing approach, the optimal job sequence is $J_1 - J_3 - J_4 - J_2$, in which the customer orders 1, 2, and 3 are completed at 185, 70, and 115 time units, respectively, and the total completion time of the customer orders is 185 + 70 + 115 = 370 time units.

2.3. Preliminary Results

We now give some definitions and theorems to investigate the complexities of the problems with different processing approaches, and derive some structural properties of the optimal solutions for these problems.

Definition 1. Let P_{JBP}, P_{OBP} and P'_{OBP} denote the problems with job-based processing, order-based processing with setup savings, and order-based processing without setup savings, respectively.

Definition 2. *Total Time* (TT) of a customer order is the sum of the setup, if any, and processing times of all products (jobs) in this customer order.

Definition 3. The *Shortest Total Time* (STT) sequence is a sequence in which customer orders are sequenced in non-decreasing order of their total time (TT).

Since there are no restrictions that delay setups, jobs, and customer orders, we have the following result.

Lemma 1. *For all problems* P_{JBP}, P_{OBP} and P'_{OBP}, *there exists an optimal schedule in which the machine has no idle time; that is, the machine is busy for either processing a customer order of a job or being set up.*

The optimal schedule for problem P'_{OBP} is given by the following theorem.

Theorem 1. *The Shortest Total Time (STT) sequence gives the optimal schedule for problem* P'_{OBP}.

Proof. It is clear that a sequence characterized by a string-based version of STT becomes optimal when each customer order may be treated as a pseudo-string of jobs, as it is given by Pinedo (2008). □

Remark 1. It is obvious that the minimum total completion time of the problem P_{OBP} is always less than or equal to the minimum total completion time of the problem P'_{OBP}. Furthermore, the problem P_{OBP}, when there is no-setup time, turns into the problem P'_{OBP}, which is optimally solved by Theorem 1.

The relevant definitions and theorems for the optimal solution of the problem P_{OBP} can be seen in Akkocaoğlu (2014), and the mathematical model for solving the problem P_{OBP} is given in Appendix A.

Remark 2. When each customer gives an order consisting of only one product different from those ordered by the other customers, we observe that the problem P_{JBP} reduces to the scheduling problem of multiple products (jobs) to minimize the total completion time of the customer orders, which is equivalent to the sum of the job completion times. This reduced problem is equivalent to the classical single-machine problem $1/\sum C_j$ in which there are multiple jobs. In this reduced problem, the STT rule minimizes the total completion time of the customer orders. On the other hand, when there is a single customer order, as an extreme case, with multiple products, we observe that the problem P_{JBP} reduces to the scheduling problem of multiple products (jobs) to minimize the maximum completion time (makespan) of the jobs in that customer order. In this reduced problem, the makespan minimization becomes trivial, and the arbitrary sequence of the products (jobs) is the optimal solution. Therefore, to investigate the complexity of the problem P_{JBP}, we assume that the number of customer orders is more than one, and at least one of the customers gives an order with more than one product (job) from a set of several jobs.

Definition 4. The *Smallest Demand* (SD) sequence is a sequence in which customer orders of a job are sequenced in non-decreasing order of their demand for this job.

The following lemma describes the sequence for the customer orders of the job in the last position of the optimal schedule for the problem P_{JBP}.

Lemma 2. *For the problem P_{JBP}, there exists an optimal schedule, in which all customer orders of the job in the last position of the job sequence are processed by the SD rule.*

Proof Note that the total completion time of the customer orders having no demand for the product (job) processed in the last position of the job sequence does not depend on the sequence of the customer orders in the job processed as the last in the job sequence. Thus, the problem of finding the sequence of the customer orders of the job in the last position of the job sequence can be considered as the single-job case, which is equivalent to the classical single-machine scheduling problem $1/\sum C_j$. Smith (1956) showed that processing the jobs in the shortest processing time (SPT) rule minimizes the total completion time for the basic single-machine problem in which there are multiple jobs. In our problem, all customer orders for the last job in the job sequence can be thought of as the jobs in the classical single-machine scheduling problem, and they should be sequenced by the SD rule. □

The following theorem gives the optimal sequence of the customer orders in each job when a product (job) sequence is given for the problem P_{JBP}.

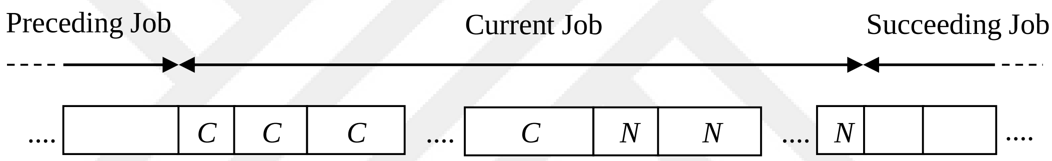
Theorem 2. *For a given product (job) sequence for the problem P_{JBP}, there is an optimal sequence of the customer orders in each job with the following properties:*

(a) In each job, the customer orders completed with this job precede all the customer orders completed with the succeeding jobs, as illustrated in Figure 2.

(b) In each job, the set of customer orders completed with this job is scheduled in SD sequence, whereas the set of customer orders completed with the succeeding jobs is sequenced in any order.

Proof. The proof of the first part is straightforward. Suppose that a customer order completed with the current job is preceded by a customer order completed with the succeeding job. The quality of the schedule does not decrease by moving this customer order completed with the succeeding job to the end of the sequence of customer orders in the current job and shifting forward all the customer orders currently succeeding it. The repetition of this argument shows the correctness of the first property.

The proof of the second property follows from the result by Smith (1956), who showed that processing the jobs with SPT rule minimizes the total completion time in a single-machine scheduling problem $1/\sum C_j$. Starting from the last minus one position of the job sequence, repetitive use of Lemma 2 in the job shows the correctness of the second property. □



C : Customer order completed in the current job

N : Customer order completed in the succeeding job

Figure 2 Positions of the customer orders in a job

Based on Theorem 2, the algorithm SCO below gives the optimal sequence of the customer orders in each job when a job sequence is given for the problem P_{JBP}.

Algorithm SCO

Step 1 For the given job sequence, generate the initial sequence of the customer orders in each job by sorting the customer orders of each job in non-descending order of their demand for this job.

Step 2 a Set $l = N - 1$.

b Let the job in position l of the given job sequence as the current job.

c Starting from the customer order in the first position of the customer orders sequence in the current job, check whether the customer order will be completed in the succeeding jobs. If the answer is yes, then sent this customer order to the last position of the customer orders in the current job; otherwise, keep this customer order in its current position. Repeat this step for all customer orders in the current job.

d Set $l = l - 1$. If $l > 0$, then go to Step 2b; otherwise, stop.

CHAPTER 3

LITERATURE REVIEW

Within the context of scheduling, a customer order and a product ordered by a customer in the COS problem may correspond to a group and a job in the group, respectively, in the *Group Scheduling* (GS) problem. Thus, the COS and GS are two closely related problems, and the problems under study fall in the intersection of these two main areas of research in the literature of scheduling.

In this chapter, we provide a brief overview of the COS and GS studies with a focus on the single-machine problems to facilitate the proper positioning of our study in the literature.

3.1. Customer Order Scheduling

Although the concept of customer order scheduling was first introduced nearly three decades ago by Julien and Magazine (1990), Ahmadi and Bagchi (1990), COS problems are scarce in the literature. Julien and Magazine (1990) considered multiple customer orders containing several products (jobs) processed on a single machine with a job-dependent setup time between two different types of jobs. They provided a dynamic programming algorithm for minimizing the total completion time of orders when there exist only two types of jobs, and the batch processing order is fixed. Subsequently, Bagchi et al. (1994) considered the COS problem on a single machine in which they aimed to determine the due dates of the customer orders and to schedule all jobs to minimize penalty function. Other early research efforts for COS problems of the single machine case are carried out by Baker (1988), Coffman et al. (1989), and Vickson et al. (1993). For more recent studies, Erel and Ghosh (2007) considered a single machine COS model in which orders consisted of various quantities of products coming from different families. Family dependent setup time is incurred between different families of products. They discussed the complexity of the problem and proposed a dynamic programming algorithm for solving the problem.

One of the other recent studies carried out by Hazır et al. (2008) investigated the COS problem on a single facility, which aimed to minimize the average customer order flow time, and they proposed four metaheuristics: simulated annealing, genetic algorithm, tabu search, and ant colony, respectively. Then, they evaluated the performance of these heuristics.

As we can see from the previous studies, there are several variants for the COS related problems under different scheduling criteria such as maximum completion time (makespan), total completion time, and maximum lateness, and under different machine environments such as parallel machines and job shop environments. However, COS problems for the single machine case are quite a few in the literature. We review the most related and recent works done in the remainder of this section.

Yang (2017) addresses a similar COS problem on a single machine of which the lot description is considered as job in our study. Orders are indivisible, and each order has to be processed on the same lot. He provided the complexity of the problem, a binary integer programming model, and four efficient heuristics to minimize the makespan and the total completion time objectives, respectively. The main difference between the problem studied by Yang (2017) and the one studied in this thesis is the processing approach. He assumed that all orders in the same lot have the same processing times and same completion times. Furthermore, each lot has the capacity, and there are no setup times between different lots in his study, whereas our study tackles sequence-independent setup times that exist between different products (jobs).

The study that has similar characteristics to our problem belongs to Akkocaoğlu (2014) which considers a COS problem with order-based processing approach, and there is a sequence-independent setup time between jobs in a customer order. It aims to avoid frequent product (job) switchovers, which aims to minimize the makespan and the total completion time. Hence, it is accomplished by combining the first job of a customer order with the last job of the immediately preceding customer order if these jobs are the same.

There is also one more recent study, namely by Yozgat (2018), which considers the job-based processing approach for the two-machine flow shop environment to find a sequence of the job lots as well as the sublots (customer orders) in each job, thereby minimizing the total completion time of the customer orders.

3.2. Group Scheduling

In the past two decades, the job grouping idea has received considerable attention. Family scheduling and group technology are prevalent aspects of recent scheduling problems. The main idea of these approaches is classifying the jobs that share similar properties into the same groups or families, which helps to improve the efficiency of operations and save time. The studies that incorporate benefits from job grouping, the reader is referred to the survey papers done by Webster and Baker (1995), Potts and Kovalyov (2000), Allahverdi et al. (2015), and Neufeld et al. (2016).

Group scheduling problems date back to the pioneering work of Gupta (1988). He studied a single-machine scheduling problem where jobs are divided into diverse classes of jobs, and setup time is required between different classes. A heuristic algorithm is proposed to minimize mean flow time. Similarly, Gupta et al. (1997) extended the scheduling problem under two different objective criteria: minimization of makespan and total carrying costs of the customer orders, respectively. Edwin et al. (1996) also provided a good framework for the problem of grouping jobs. In their study, jobs are classified into several groups, and the jobs within the same group processed contiguously. Sequence-independent setup time is defined. A schedule is determined by a sequence of the groups and a sequence of the jobs in each group. A polynomial-time algorithm is proposed to minimize maximum cost and total weighted completion time.

Another well-defined study for grouping jobs on a single machine is carried out by Liao and Chuang (1996). The various jobs of the customer orders are clustered into several groups, and setup time between different groups is required to process on a machine. Branch and bound algorithms are proposed to minimize the two objective criteria: number of tardy orders and the maximum tardiness, respectively. Gerodimos et al. (1999) addressed a similar problem of family scheduling model in which jobs consist of multiple operations that belong to different families. Their study covered three objective criteria: the maximum lateness, the weighted number of late jobs, and the sum of job completion times, respectively. Karabati and Akkan (2006) presented a branch and bound algorithm for minimizing the total completion time in a single-machine where jobs can be grouped into families, and a sequence-dependent family setup is incurred if the sequence requires a switch from a job in a particular family to a job in a different family. Wu and Lee (2006) focused on the same problem and determined total setup time and the total earliness as measures of performance for their problem. Gupta and Chantaravarapan (2008) considered the group scheduling problem with a sequence-independent setup time between families of jobs. A mixed-integer linear programming model and a simulated annealing algorithm are developed to minimize total tardiness.

On the other hand, job grouping problems are prevalent in the field of process industries and electronics manufacturing. One of the studies belongs to Sabouni and Logendran (2013) that considered a single machine group scheduling problem in the PCB manufacturing environment with carryover sequence-dependent setup times, and they proposed a branch-and-bound algorithm to minimize the makespan.

Several recent studies introduce new concepts of job deteriorating and learning effects into the group scheduling problems. We reviewed some of the them which are the most relevant to our study. One of the studies, which is done by Wang et al. (2012), considered a single machine problem under makespan minimization with the group technology assumption and the deterioration effect of jobs. Fixed group setup times and ready times of the jobs are assumed in this problem. In addition, the problem studied by He and Sun (2012) considered a single machine group scheduling with deterioration without ready times to minimize the total completion time. They showed that their problem could be polynomially solvable only under some conditions. In the case of jointly compressible setup and processing times, a polynomial-time algorithm to find the optimal solution to minimize the total job completion time on a single machine is presented in Ng et al. (2004).

The concepts of group technology and time-dependent processing times are also introduced in the study of Wang and Wang (2014). They proved that the problem is solvable in polynomial-time and attempted to minimize the makespan when ready times of the jobs are available. The reader can find thorough surveys on related works that are mentioned in the study of Wang and Wang (2014). Moreover, He and Sun (2015) similarly studied the problem with deterioration and learning effect with the group technology assumption. They showed that the total completion time minimization could be solved in polynomial time. More recently, Liu et al. (2019) addressed a single-machine group scheduling problem with deterioration effect and job-ready times. An efficient heuristic and two exact algorithms are developed to minimize the makespan objective. Their study also covers the related works done in this area so that the reader can refer to the study of Liu et al. (2019) for the comprehensive review.

Apart from the above studies, single machine batch delivery problems also resemble the problem undertaken in this thesis. Batch delivery, especially in a single machine case, was first introduced by Santos and Magazine (1985). Mazdeh et al. (2007) adopted the concept of batch delivery on a single machine and aimed to minimize maximum tardiness and delivery costs.

There are several variants of studies in the scheduling literature that deal with customer order scheduling and group scheduling problems. However, to the best of our knowledge, the job-based processing approach for the single machine case is considered in our study for the first time.

CHAPTER 4

PROPOSED SOLUTION APPROACHES: MIXED INTEGER PROGRAMMING MODELS AND A TABU-SEARCH BASED HEURISTIC ALGORITHM

In this chapter, our two solution approaches, which are the mathematical programming model and the tabu-search based heuristic algorithm, for the job-based processing problem are explained in detail.

4.1. Mixed Integer Programming Models

In this section, we present two mixed-integer linear programming (MILP) models to solve the problem P_{JBP} optimally. Our models provide the optimal schedule with the job sequence (i.e., the sequence of the products) and the sequence of the customer orders within each job to minimize the total completion time of the customer orders.

For developing our models, we first introduce the following parameters, indices and sets, which are commonly used in both MILP models.

Parameters, indices and sets

K Number of customer orders.

o, m, u Indices for customer orders ($o, m, u = 1,2, \dots, K$).

N Number of jobs.

j, k, l Indices for jobs ($j, k, l = 1,2, \dots, N$).

$D_{o,j}$ Demand (number of identical items) for job j in customer order o.

SC_j Set of customer orders having demand for job j.

L_j Lot size (total demand) for job j, where $L_j = \sum_{o \in SC_j} D_{o,j}$.

t_j Unit processing time for job j.

s_j Setup time for job j.

4.1.1. The First Model (MILP-1)

In this first model, we use the sequence-position variables. Our additional parameters indices and sets are as follows:

Additional parameters and indices

$H_{o,j}$ 1, if customer order o has demand for job j; 0, otherwise.

$\|SC_j\|$ Cardinality of the set of customer orders having demand for job j. That is, the number of customer orders having demand for job j.

Q Sufficiently large positive number.

p Index for positions in the job sequence ($p = 1,2, \dots, N$).

r Index for positions in the sequence of customer orders having demand for job j ($r = 1,2, \dots, \|SC_j\|$).

Decision variables

$$Y_{j,p} = \begin{cases} 1 & \text{if job } j \text{ is assigned to position } p \text{ of the job sequence} \\ 0 & \text{otherwise} \end{cases}$$

$$X_{o,j,p,r} = \begin{cases} 1 & \text{if customer order } o \text{ in job } j \text{ at position } p \text{ of the job sequence is} \\ & \text{assigned to position } r \text{ of the customer orders sequence in job } j \\ 0 & \text{otherwise} \end{cases}$$

$C_{o,j,p,r}$ Completion time of customer order o assigned to position r of the customer orders sequence in job j at position p of the job sequence.

$CT_{j,p}$ Completion time of job j assigned to position p of the job sequence.

T_o Completion time of the customer order o.

The MILP-1 model for solving the problem P_{JBP} can be modeled as follows:

Minimize $\sum_{o=1}^{K} T_o$ (1)

Subject to $\sum_{j=1}^{N} Y_{j,p} = 1$ for $p = 1,2, \dots, N$ (2)

$\sum_{p=1}^{N} Y_{j,p} = 1$ for $j = 1,2, \dots, N$ (3)

$\sum_{o=1}^{K} H_{o,j} X_{o,j,p,r} = Y_{j,p}$ for $j, p = 1,2, \dots, N$; $r = 1,2, \dots, \|SC_j\|$ (4)

$\sum_{r=1}^{\|SC_j\|} H_{o,j} X_{o,j,p,r} = Y_{j,p}$ for $j, p = 1,2, \dots, N$; $o = 1,2, \dots, K$ (5)

$C_{o,j,1,1} \geq s_j + t_j D_{o,j} X_{o,j,1,1} - Q(1 - X_{o,j,1,1})$

for $o = 1,2, \dots, K$; $j = 1,2, \dots, N$ (6)

$C_{o,j,p,1} \geq CT_{k,p-1} + s_j + t_j D_{o,j} X_{o,j,p,1} - Q(1 - X_{o,j,p,1})$

$$\text{for } o = 1,2, \dots, K;\ j, k = 1,2, \dots, N;$$
$$p = 2,3, \dots, N;\ j \neq l \quad (7)$$

$$C_{o,j,p,r} \geq C_{m,j,p,r-1} + t_j D_{o,j} X_{o,j,p,r} - Q(1 - X_{o,j,p,r})$$
$$\text{for } o, m = 1,2, \dots, K;\ j = 1,2, \dots, N;$$
$$p = 2,3, \dots, N;\ r = 2,3, \dots, \|SC_j\|;$$
$$m \neq o \quad (8)$$

$$CT_{j,p} \geq C_{o,j,p,r} - Q(1 - Y_{j,p})$$
$$\text{for } o = 1,2, \dots, K;\ j = 1,2, \dots, N;$$
$$p = 2,3, \dots, N;\ r = 1,2, \dots, \|SC_j\| \quad (9)$$

$$T_o \geq C_{o,j,p,r} \qquad \text{for } o = 1,2, \dots, K;\ j, p = 1,2, \dots, N;$$
$$r = 1,2, \dots, \|SC_j\| \quad (10)$$

$$C_{o,j,p,r},\ CT_{j,p},\ T_o \geq 0 \qquad \text{for } \forall\ o, j, p, r \quad (11)$$

$$X_{o,j,p,r}, Y_{j,p} \in \{0,1\} \qquad \text{for } \forall\ o, j, p, r \quad (12)$$

In the above MILP-1 model, the objective in (1) is to minimize the total completion time of customer orders. Constraint sets (2) and (3) ensure that each position in the sequence of jobs is occupied by one job only, and each job is assigned to one position only, respectively. Constraint set (4) guarantees that each position in the sequence of customer orders in a job is occupied by one customer order only. Constraint set (5) ensures that each customer order in a job is assigned to a position in the sequence of customer orders in this job. Constraint sets (6) and (7) determines the completion time of the customer order assigned to the first position of customer orders in the job assigned to the first and remaining positions of the job sequence, respectively. Constraint set (8) defines the completion times of the customer orders assigned to the remaining positions of the sequence of customer orders in a job. Constraint set (9) determines the completion time of each job in each position of the job sequence. Constraint set (10) defines the completion time of each customer order. Constraint sets (11) and (12) impose the non-negativity and binary restrictions, respectively, on the decision variables.

In our MILP-1 model, there are three sets of continuous variables, and the number of these variables are $N^2(K^2 + 1) + K$. Also, there are two sets of binary variables, and the number of binary decision variables is $N^2(K^2 + 1)$. This means that there is a total of $2N^2(K^2+1)+K$ decision variables. On the other hand, the MILP-1 model has a total of $N^3-2N^2-3N+NK(K^2-2K+3)$ constraints.

4.1.2. The Second Model (MILP-2)

In the second model, we rely on the precedence variables.

Decision variables

$$Y_{j,k} = \begin{cases} 1 & \text{if job } j \text{ precedes job } k \\ 0 & \text{otherwise} \end{cases}$$

$$X_{o,m,j} = \begin{cases} 1 & \text{if customer order } o \text{ in job } j \text{ precedes customer order } m \text{ in job } j \\ 0 & \text{otherwise} \end{cases}$$

$C_{o,j}$ Completion time of customer order o in job j.

T_o Completion time of the customer order o.

The MILP-2 model for solving the problem P_{JBP} can be modeled as follows:

Minimize $\sum_{o=1}^{K} T_o$ (13)

Subject to $Y_{j,k} + Y_{k,j} = 1$ for $j,k = 1,2,\dots,N;\ j<k$ (14)

$Y_{j,k} + Y_{k,l} + Y_{l,j} \leq 2$ for $j,k,l = 1,2,\dots,N;\ j \neq k \neq l$ (15)

$X_{o,m,j} + X_{m,o,j} = 1$ for $o,m = 1,2,\dots,K;\ o<m\ ;\ j = 1,2,\dots,N;$ $D_{o,j} = D_{m,j} > 0$ (16)

$X_{o,m,j} + X_{m,u,j} + X_{u,o,j} \leq 2$ for $o,m,u = 1,2,\dots,K;\ j = 1,2,\dots,N;$ $o \neq m \neq u;\ D_{o,j} = D_{m,j} = D_{u,j} > 0$ (17)

$C_{o,j} = \sum_{\substack{k=1 \\ k\neq j}}^{N} (s_k + t_k L_k) Y_{k,j} + s_j + t_j D_{o,j} + \sum_{\substack{m=1 \\ m\neq o}}^{K} t_j D_{m,j} X_{m,o,j}$ for $o = 1,2,\dots,K;\ j = 1,2,\dots,N;\ D_{o,j} > 0$ (18)

$T_o \geq C_{o,j}$ for $o = 1,2,\dots,K;\ j = 1,2,\dots,N$ (19)

$C_{o,j},\ T_o \geq 0$ for $\forall\ o,j$ (20)

$X_{o,m,j}, Y_{j,k} \in \{0,1\}$ for $\forall\ o,m,j,k$ (21)

In the above MILP-2 model, the objective in (13) is to minimize the total completion time of customer orders. Constraint set (14) ensures the ordering of the jobs, and similarly, the constraint set (15) guarantees that for each pair of the orders, one of them should precede the other. Constraint sets (16) and (17) are triangular inequalities.

Constraint set (18) calculates the completion time of each customer order i that demands job j. Constraint set (19) defines the completion time of each customer order. Constraint sets (20) and (21) impose non-negativity and binary restrictions, respectively, on the decision variables.

In our MILP-2 model, there are two sets of continuous variables, and the number of these variables are $K(N+1)$. Also, there are two sets of binary variables, and the number of binary decision variables is $N(K^2+N)$. This means that there is a total of $K(N+1+KN)+N^2$ decision variables. On the other hand, the MILP-2 model has a total of $K+N+NK(1+N(1+N)+NK(NK+1))$ constraints.

According to number of decision variables and number of constraints, first model (MILP-1) is efficient than the second model (MILP-2). However, from our preliminary experiments, we observed that the solution time of MILP-1 took longer than MILP-2. Therefore, in the rest of our study, we considered our second model only, and called it MILP.

4.2. Heuristic Algorithm

The size of the MILP model increases tremendously as the number of products (jobs) and the number of customer orders increase. We observe from our experiments that the MILP model cannot provide optimal solutions for the large-sized problem instances in reasonable times. Therefore, we propose a heuristic algorithm that provides optimal or near-optimal solutions for the large-sized problem instances within relatively short times.

Our proposed heuristic algorithm consists of two phases: *finding an initial schedule by the insertion algorithm* and *improving the initial schedule by the tabu search algorithm*. The detailed descriptions of each phase in our heuristic algorithm are given below.

Phase 1 - Finding an initial schedule by the insertion algorithm

This phase finds an initial schedule of jobs by applying the *insertion algorithm,* which is a kind of neighborhood algorithm. It is also known as the NEH algorithm since it was proposed first by Nawaz et al. (1983) for the makespan minimization problem in a flow shop. The NEH algorithm has been widely used to solve various scheduling problems with different scheduling criteria other than makespan. The algorithm generates $(N(N+1)/2)-1$ different sequences of jobs, where N of them are complete, and the rest are partial sequences. The NEH algorithm is based on the assumption that a job with a long total processing time is given higher priority than the job with a small total processing time. In our algorithm, we have modified this assumption as the job with more number of customer orders is given a higher priority than the job with fewer customer orders.

The stepwise description of Phase 1 in our algorithm is given below.

Step 1 Generate an initial job sequence by sorting the jobs in descending order of their number of customer orders.

Step 2 In the initial job sequence, generate the initial sequence of the customer orders in each job by sorting the customer orders of each job in ascending order of their total demand.

Step 3
- a Select the jobs $J_{[1]}$ and $J_{[2]}$, which are in the first two positions of the initial job sequence obtained in Step 2.
- b Form two partial job sequences such that the first selected job $J_{[1]}$ is in the first and second positions in these partial sequences, respectively. That is,

  Partial sequence 1: $J_{[1]} - J_{[2]}$

  Partial sequence 2: $J_{[2]} - J_{[1]}$
- c Let the first partial sequence among all partial sequences be the current partial sequence.

Step 4
- a Let the job that is in the last position of the current partial sequence be the current job. Sort all customer orders of the current job in ascending order of their total demands.
- b Consider the previous job as the new current job.
- c Starting from the customer order in the first position of the customer orders sequence in the current job, check whether the customer order has the jobs

processed after the current job of the current partial job sequence. If the answer is yes, then sent this customer order to the last position of the customer orders in the current job; otherwise, keep this customer order in its current position. Repeat this step for all remaining customer orders in the current job.

d If the current job is in the first position of the current partial sequence, then compute the total completion time of the customer orders in the current partial sequence, and go to Step 4e; otherwise, go to Step 4b.

e If all partial sequences are considered, then select the best partial sequence giving the minimum total completion time and go to Step 5a; otherwise, consider the next partial sequence as the current partial sequence and go to Step 4a.

Step 5 a If all jobs of the initial job sequence obtained in Step 2 are not considered yet, then go to Step 4a; otherwise, go to Phase 2.

b Pick the job that is in the next position of the initial job sequence obtained in Step 2, generate all possible partial sequences by placing this new job in all possible positions (beginning, between and ending) in the best partial sequence developed so far, and go to Step 4a.

Phase 2 – Improving the initial schedule by the tabu search algorithm

Tabu Search (TS), which was first proposed by Glover (1989), is a local-search based metaheuristic algorithm for solving many combinatorial optimization problems. TS algorithm has attracted many researchers working on scheduling problems and widely used in the literature. It starts with an initial solution (schedule) generated randomly or obtained by a simple rule or a heuristic algorithm. The initial solution is considered as the best solution. Then a local search mechanism is applied to find a better solution in the *neighborhood* of the current solution, which is defined as all solutions (also called *mutations*) obtained by an alternative solutions generation mechanism using the current solution. This neighborhood generation mechanism can be an adjacent pairwise interchange of the jobs or inserting every job in every position in the current schedule. In our TS procedure, we use the schedule obtained by Phase 1 of our algorithm as the initial schedule, and the neighborhood is generated by *adjacent pairwise interchanges* of the jobs in this initial schedule. The mutation with the lowest objective function (total completion time of the customer orders) value is selected as a *candidate solution*. The local changes providing the candidate solution among the solutions in the neighborhood of a current solution is called a *move*. To keep the search history, a list called *tabu list* is used to avoid cycling (i.e., returning to a solution that has been visited before) and guide the search towards unexplored regions of the solution space of the problem. The move providing the candidate solution is put into the tabu list if this move is not tabu, and the candidate solution becomes the new best solution if the objective function value of the candidate solution is better than the objective function value of the current best solution. This is the *aspiration criterion* used in our TS procedure. Once a move is entered the tabu list, the oldest move in the tabu list is deleted since the tabu list has a fixed size, which is called *tabu list size*, say *l*. Tabu list size, which is also called *tabu tenure,* allows the new move added to the tabu list to remain in the list for the next *l* iterations.

Tabu-search iterations are conducted until one of the *stopping criteria* is reached. In the literature, there are several applications of the TS algorithm using different stopping criteria, which determine the length of the search. One approach is to set the number of iterations to a pre-specified value. That is, the TS procedure stops when no improvement can be obtained after several iterations. It is clear that setting the number of iterations to a large number may increase the search space and solution time. In our algorithm, we let the TS procedure run for $NI = 2 \times N$ iteration, where N is the number of jobs. Our second stopping criterion is that the TS procedure terminates if all possible mutations are worse than the parent.

Tabu tenure is also an important parameter that affects the performance of the TS procedure since the tabu list directs the search. The tabu tenure can be fixed (usually preferred in the literature) or variable. Setting the tabu tenure to a small number may cause an occurrence of cycling, i.e., returning to the solution already visited before. That is, it is very hard to escape from local optima when the tabu tenure is too small. However, setting the tabu tenure to a large number may result deterioration in the quality of the solutions found. In other words, the algorithm spends more time to compare with the current solution one by one. Tabu list size can be a variable or a fixed number. In our algorithm, we set the tabu list size to 5.
The stepwise description of Phase 2 in our algorithm is given below.

Step 1 Set the iteration counter ic to 1, i.e., set $ic = 1$. Set the initial schedule σ_1 to the schedule obtained in Phase 1 of the algorithm. Set the best schedule σ_B to σ_1, i.e., set $\sigma_B = \sigma_1$.

Step 2 a Generate the neighborhood of the schedule σ_{ic} by adjacent pairwise interchanges of the jobs in the schedule σ_{ic}.

b For each of the mutation in the neighborhood of the schedule σ_{ic}, apply the algorithm SCO.

c If the total completion time value of each mutation is bigger than the total completion time of the parent schedule σ_{ic}, then stop; otherwise, from the neighborhood of the schedule σ_{ic}, select the schedule with the lowest total completion time value as the candidate schedule σ_C.

Step 3 a If the move $\sigma_{ic} \rightarrow \sigma_C$ is prohibited by a mutation on the tabu list, set $\sigma_{ic+1} = \sigma_{ic}$ and go to step 4; otherwise,

i Delete the entry at the bottom of the tabu list.

ii Push all other entries in the tabu list one position down.

iii. Enter reverse mutation at the top of the tabu list.

iv. Set $\sigma_{ic+1} = \sigma_C$.

v. Set the new best schedule to the candidate schedule (i.e., set $\sigma_B = \sigma_C$) if the total completion time value of the candidate schedule is smaller than the total completion time value of the current best schedule, i.e., $TCT(\sigma_C) < TCT(\sigma_B)$.

vi. Go to step 4.

Step 4 a Increment the iteration counter ic by 1. i.e., set $ic = ic + 1$.

b If the iteration counter ic is equal to the pre-specified value NI for the number of iterations (i.e., $ic = NI$), then stop; otherwise, go to step 2.

4.3. A Numerical Example

We close this chapter with the following numerical example to demonstrate our proposed heuristic algorithm. Consider a problem instance in which five customers give orders for five products (jobs). Products demanded by the customer orders, the sequence-independent setup times, and the unit-processing times are given in Table 2.

Table 2 Data set for the numerical example

Jobs (Products)	Demand (in units) of the customer orders					Setup time	Unit-processing time
	O_1	O_2	O_3	O_4	O_5		
J_1	9	3	1	6	3	41	4
J_2	-	-	-	5	-	48	6
J_3	-	-	3	8	7	5	4
J_4	-	6	5	-	-	40	6
J_5	-	-	-	5	4	47	8

Phase 1 - Finding an initial schedule by the insertion algorithm

Step 1 Sorting the jobs in descending order of their number of customer orders gives the initial job sequence as:

$$J_1\{O_1, O_2, O_3, O_4, O_5\} - J_3\{O_3, O_4, O_5\} - J_4\{O_2, O_3\} - J_5\{O_4, O_5\} - J_2\{O_4\}$$

Step 2 In the initial job sequence, sorting the customer orders of each job in ascending order of their total demand yields the following initial sequence of jobs with the sorted customer orders:

$$J_1\{O_3[1], O_2[3], O_5[3], O_4[6], O_1[9]\} - J_3\{O_3[3], O_5[7], O_4[8]\} -$$
$$J_4\{O_3[5], O_2[6]\} - J_5\{O_5[4], O_4[5]\} - J_2\{O_4[5]\}$$

Step 3 From the initial job sequence $J_1 - J_3 - J_4 - J_5 - J_2$ obtained in Step 2, we select the first two jobs J_1 and J_3. We form two partial sequences $J_1 - J_3$ and $J_3 - J_1$.

Step 4 In the first partial sequence $J_1 - J_3$, the optimal sequence of the customer orders in each job is:

$$J_1\{\boldsymbol{O_2}[\mathbf{3}], \boldsymbol{O_1}[\mathbf{9}], O_3[1], O_5[3], O_4[6]\} - J_3\{\boldsymbol{O_3}[\mathbf{3}], \boldsymbol{O_5}[\mathbf{7}], \boldsymbol{O_4}[\mathbf{8}]\}$$

with the total completion time of customer orders

$$TCT(J_1 - J_3) = CT_2 + CT_1 + CT_3 + CT_5 + CT_4$$

$= 53 + 89 + 146 + 174 + 206$

$= 668.$

In the second partial sequence $J_3 - J_1$, the optimal sequence of the customer orders in each job is:

$J_3\{O_3[3], O_5[7], O_4[8]\} - J_1\{\boldsymbol{O_3[1]}, \boldsymbol{O_2[3]}, \boldsymbol{O_5[3]}, \boldsymbol{O_4[6]}, \boldsymbol{O_1[9]}\}$

with the total completion time of customer orders

$$TCT(J_3 - J_1) = CT_3 + CT_2 + CT_5 + CT_4 + CT_1$$
$$= 122 + 134 + 146 + 174 + 206$$
$$= 778.$$

We select the partial sequence $J_1 - J_3$ since its total completion time is smaller than that of the partial sequence $J_3 - J_1$.

Step 5 All jobs of the initial job sequence obtained in Step 2 are not considered yet. Thus, we go to Step 4.

Step 4 We select the next job, which is job J_4, from the initial job sequence obtained in Step 2, and form three partial sequences $\boldsymbol{J_4} - J_1 - J_3$, $J_1 - \boldsymbol{J_4} - J_3$, and $J_1 - J_3 - \boldsymbol{J_4}$.

Step 5 In the first partial sequence $\boldsymbol{J_4} - J_1 - J_3$, the optimal sequence of the customer orders in each job is:

$J_4\{O_3[5], O_2[6]\} - J_1\{\boldsymbol{O_2[3]}, \boldsymbol{O_1[9]}, O_3[1], O_5[3], O_4[6]\} - J_3\{\boldsymbol{O_3[3]}, \boldsymbol{O_5[7]}, \boldsymbol{O_4[8]}\}$

with the total completion time of customer orders

$$TCT(J_4 - J_1 - J_3) = CT_2 + CT_1 + CT_3 + CT_5 + CT_4$$
$$= 159 + 195 + 252 + 280 + 312$$
$$= 1{,}198.$$

In the second partial sequence $J_1 - \boldsymbol{J_4} - J_3$, the optimal sequence of the customer orders in each job is:

$J_1\{\boldsymbol{O_1[9]}, O_3[1], O_2[3], O_5[3], O_4[6]\} - J_4\{\boldsymbol{O_2[6]}, O_3[5]\} - J_3\{\boldsymbol{O_3[3]}, \boldsymbol{O_5[7]}, \boldsymbol{O_4[8]}\}$

with the total completion time of customer orders

$$TCT(J_1 - J_4 - J_3) = CT_1 + CT_2 + CT_3 + CT_5 + CT_4$$
$$= 77 + 205 + 252 + 280 + 312$$
$$= 1{,}126.$$

In the third partial sequence $J_1 - J_3 - \boldsymbol{J_4}$, the optimal sequence of the customer orders in each job is:

$J_1\{\boldsymbol{O_1}[\mathbf{9}], O_3[1], O_2[3], O_5[3], O_4[6]\} -$
$J_3\{\boldsymbol{O_5}[\mathbf{7}], \boldsymbol{O_4}[\mathbf{8}], O_3[3]\} - J_4\{\boldsymbol{O_3}[\mathbf{5}], \boldsymbol{O_2}[\mathbf{6}]\}$

with the total completion time of customer orders

$$\begin{aligned} TCT(J_1 - J_3 - J_4) &= CT_1 + CT_5 + CT_4 + CT_4 + CT_2 \\ &= 77 + 162 + 194 + 276 + 312 \\ &= 1{,}021. \end{aligned}$$

Among these three partial sequences, we select the partial sequence $J_1 - J_3 - J_4$ since its total completion time is smaller than those of other partial sequences.

Step 4 We select the next job, which is job J_5, from the initial job sequence obtained in Step 2, and form four partial sequences $\boldsymbol{J_5} - J_1 - J_3 - J_4$, $J_1 - \boldsymbol{J_5} - J_3 - J_4$, $J_1 - J_3 - \boldsymbol{J_5} - J_4$, and $J_1 - J_3 - J_4 - \boldsymbol{J_5}$.

Step 5 In the first partial sequence $\boldsymbol{J_5} - J_1 - J_3 - J_4$, the optimal sequence of the customer orders in each job is:

$J_5\{O_5[4], O_4[5]\} - J_1\{\boldsymbol{O_1}[\mathbf{9}], O_3[1], O_2[3], O_5[3], O_4[6]\} -$
$J_3\{\boldsymbol{O_5}[\mathbf{7}], \boldsymbol{O_4}[\mathbf{8}], O_3[3]\} - J_4\{\boldsymbol{O_3}[\mathbf{5}], \boldsymbol{O_2}[\mathbf{6}]\}$

with the total completion time of customer orders

$$\begin{aligned} TCT(J_5 - J_1 - J_3 - J_4) &= CT_1 + CT_5 + CT_4 + CT_3 + CT_2 \\ &= 196 + 281 + 313 + 395 + 431 \\ &= 1{,}616. \end{aligned}$$

In the second partial sequence $J_1 - \boldsymbol{J_5} - J_3 - J_4$, the optimal sequence of the customer orders in each job is:

$J_1\{\boldsymbol{O_1}[\mathbf{9}], O_3[1], O_2[3], O_5[3], O_4[6]\} - J_5\{O_5[4], O_4[5]\} -$
$J_3\{\boldsymbol{O_5}[\mathbf{7}], \boldsymbol{O_4}[\mathbf{8}], O_3[3]\} - J_4\{\boldsymbol{O_3}[\mathbf{5}], \boldsymbol{O_2}[\mathbf{6}]\}$

with the total completion time of customer orders

$$\begin{aligned} TCT(J_1 - J_3 - J_4 - J_5) &= CT_1 + CT_5 + CT_4 + CT_3 + CT_2 \\ &= 77 + 281 + 313 + 395 + 431 \\ &= 1{,}497. \end{aligned}$$

In the third partial sequence $J_1 - J_3 - \boldsymbol{J_5} - J_4$, the optimal sequence of the customer orders in each job is:

$J_1\{\boldsymbol{O_1}[\mathbf{9}], O_3[1], O_2[3], O_5[3], O_4[6]\} - J_3\{O_3[3], O_5[7], O_4[8]\} -$
$J_5\{\boldsymbol{O_5}[\mathbf{4}], \boldsymbol{O_4}[\mathbf{5}]\} - J_4\{\boldsymbol{O_3}[\mathbf{5}], \boldsymbol{O_2}[\mathbf{6}]\}$

with the total completion time of customer orders

$$TCT(J_1 - J_3 - J_5 - J_4) = CT_1 + CT_5 + CT_4 + CT_3 + CT_2$$
$$= 77 + 285 + 325 + 395 + 431$$
$$= 1{,}513.$$

In the fourth partial sequence $J_1 - J_3 - J_4 - \boldsymbol{J_5}$, the optimal sequence of the customer orders in each job is:

$$J_1\{\boldsymbol{O_1}[\mathbf{9}], O_3[1], O_2[3], O_5[3], O_4[6]\} - J_3\{O_3[3], O_5[7], O_4[8]\} -$$
$$J_4\{\boldsymbol{O_3}[\mathbf{5}], \boldsymbol{O_2}[\mathbf{6}]\} - J_5\{\boldsymbol{O_5}[\mathbf{4}], \boldsymbol{O_4}[\mathbf{5}]\}$$

with the total completion time of customer orders

$$TCT(J_1 - J_3 - J_4 - J_5) = CT_1 + CT_3 + CT_2 + CT_5 + CT_4$$
$$= 77 + 276 + 312 + 391 + 431$$
$$= 1{,}487.$$

Among these four partial sequences, we select the partial sequence $J_1 - J_3 - J_4 - J_5$ since its total completion time is smaller than those of other partial sequences.

Step 4 We select the next job, which is job J_2, from the initial job sequence obtained in Step 2, and form five complete sequences $\boldsymbol{J_2} - J_1 - J_3 - J_4 - J_5$, $J_1 - \boldsymbol{J_2} - J_3 - J_4 - J_5$, $J_1 - J_3 - \boldsymbol{J_2} - J_4 - J_5$, $J_1 - J_3 - J_4 - \boldsymbol{J_2} - J_5$, and $J_1 - J_3 - J_4 - J_5 - \boldsymbol{J_2}$.

Step 5 In the first complete sequence $\boldsymbol{J_2} - J_1 - J_3 - J_4 - J_5$, the optimal sequence of the customer orders in each job is:

$$J_2\{O_4[5]\} - J_1\{\boldsymbol{O_1}[\mathbf{9}], O_3[1], O_2[3], O_5[3], O_4[6]\} -$$
$$J_3\{O_3[3], O_5[7], O_4[8]\} - J_4\{\boldsymbol{O_3}[\mathbf{5}], \boldsymbol{O_2}[\mathbf{6}]\} - J_5\{\boldsymbol{O_5}[\mathbf{4}], \boldsymbol{O_4}[\mathbf{5}]\}$$

with the total completion time of customer orders

$$TCT(J_2 - J_1 - J_3 - J_4 - J_5) = CT_1 + CT_3 + CT_2 + CT_5 + CT_4$$
$$= 1{,}877.$$

In the second complete sequence $J_1 - \boldsymbol{J_2} - J_3 - J_4 - J_5$, the optimal sequence of the customer orders in each job is:

$$J_1\{\boldsymbol{O_1}[\mathbf{9}], O_3[1], O_2[3], O_5[3], O_4[6]\} - J_2\{O_4[5]\} -$$
$$J_3\{O_3[3], O_5[7], O_4[8]\} - J_4\{\boldsymbol{O_3}[\mathbf{5}], \boldsymbol{O_2}[\mathbf{6}]\} - J_5\{\boldsymbol{O_5}[\mathbf{4}], \boldsymbol{O_4}[\mathbf{5}]\}$$

with the total completion time of customer orders

$$TCT(J_1 - J_2 - J_3 - J_4 - J_5) = CT_1 + CT_3 + CT_2 + CT_5 + CT_4$$
$$= 1{,}799.$$

In the third complete sequence $J_1 - J_3 - \boldsymbol{J_2} - J_4 - J_5$, the optimal sequence of the customer orders in each job is:

$J_1\{\boldsymbol{O_1}[\mathbf{9}], O_3[1], O_2[3], O_5[3], O_4[6]\} - J_3\{O_3[3], O_5[7], O_4[8]\} -$
$J_2\{O_4[5]\} - J_4\{\boldsymbol{O_3}[\mathbf{5}], \boldsymbol{O_2}[\mathbf{6}]\} - J_5\{\boldsymbol{O_5}[\mathbf{4}], \boldsymbol{O_4}[\mathbf{5}]\}$

with the total completion time of customer orders

$$TCT(J_1 - J_3 - J_2 - J_4 - J_5) = CT_1 + CT_3 + CT_2 + CT_5 + CT_4$$
$$= 1{,}799.$$

In the fourth complete sequence $J_1 - J_3 - J_4 - \boldsymbol{J_2} - J_5$, the optimal sequence of the customer orders in each job is:

$J_1\{\boldsymbol{O_1}[\mathbf{9}], O_3[1], O_2[3], O_5[3], O_4[6]\} - J_3\{O_3[3], O_5[7], O_4[8]\} -$
$J_4\{\boldsymbol{O_3}[\mathbf{5}], \boldsymbol{O_2}[\mathbf{6}]\} - J_2\{O_4[5]\} - J_5\{\boldsymbol{O_5}[\mathbf{4}], \boldsymbol{O_4}[\mathbf{5}]\}$

with the total completion time of customer orders

$$TCT(J_1 - J_3 - J_4 - J_2 - J_5) = CT_1 + CT_3 + CT_2 + CT_5 + CT_4$$
$$= 1{,}643.$$

In the fifth complete sequence $J_1 - J_3 - J_4 - J_5 - \boldsymbol{J_2}$, the optimal sequence of the customer orders in each job is:

$J_1\{\boldsymbol{O_1}[\mathbf{9}], O_3[1], O_2[3], O_5[3], O_4[6]\} - J_3\{O_3[3], O_5[7], O_4[8]\}$
$J_4\{\boldsymbol{O_3}[\mathbf{5}], \boldsymbol{O_2}[\mathbf{6}]\} - J_5\{\boldsymbol{O_5}[\mathbf{4}], O_4[5]\} - J_2\{\boldsymbol{O_4}[\mathbf{5}]\}$

with the total completion time of customer orders

$$TCT(J_1 - J_3 - J_4 - J_5 - J_2) = CT_1 + CT_3 + CT_2 + CT_5 + CT_4$$
$$= 1{,}565.$$

Among these five complete sequences, we select the sequence $J_1 - J_3 - J_4 - J_5 - J_2$ since its total completion time, which is 1,565 time units, is smaller than those of other complete sequences. Thus, the initial schedule obtained by Phase 1 is

$J_1\{\boldsymbol{O_1}[\mathbf{9}], O_3[1], O_2[3], O_5[3], O_4[6]\} - J_3\{O_3[3], O_5[7], O_4[8]\}$
$J_4\{\boldsymbol{O_3}[\mathbf{5}], \boldsymbol{O_2}[\mathbf{6}]\} - J_5\{\boldsymbol{O_5}[\mathbf{4}], O_4[5]\} - J_2\{\boldsymbol{O_4}[\mathbf{5}]\}$

Phase 2 – Improving the initial schedule by the tabu search algorithm

Step 1 Set $ic = 1$. We set the initial schedule σ_1 to the schedule obtained in Phase 3 of the algorithm, and set the best schedule σ_B to σ_1, i.e.,

$$\sigma_B = \sigma_1 = J_1 - J_3 - J_4 - J_5 - J_2$$

with $TCT(\sigma_B) = 1{,}565$.

Step 2 When we apply the adjacent pairwise interchanges of the jobs in the current schedule σ_1, we generate four mutations $\boldsymbol{J_3} - \boldsymbol{J_1} - J_4 - J_5 - J_2$, $J_1 - \boldsymbol{J_4} - \boldsymbol{J_3} - J_5 - J_2$, $J_1 - J_3 - \boldsymbol{J_5} - \boldsymbol{J_4} - J_2$, and $J_1 - J_3 - J_4 - \boldsymbol{J_2} - \boldsymbol{J_5}$. When we apply the algorithm SCO for each of these mutations, the candidate schedule σ_C becomes $J_1 - \boldsymbol{J_4} - \boldsymbol{J_3} - J_5 - J_2$ since $min\{ TCT(\boldsymbol{J_3} - \boldsymbol{J_1} - J_4 - J_5 - J_2), TCT(J_1 - \boldsymbol{J_4} - \boldsymbol{J_3} - J_5 - J_2), TCT(J_1 - J_3 - \boldsymbol{J_5} - \boldsymbol{J_4} - J_2), TCT(J_1 - J_3 - J_4 - \boldsymbol{J_2} - \boldsymbol{J_5}) \} = min\{1{,}642;\ 1{,}434; 1697; 1{,}643\} = 1{,}434$.

Step 3 The tabu list is updated with a pair of (J_3, J_4). We set $\sigma_2 = \sigma_C = J_1 - J_4 - J_3 - J_5 - J_2$, and set the new best schedule to the current schedule since the total completion time of the current schedule is smaller than that of the best schedule. That is, $\sigma_B = \sigma_C = J_1 - J_4 - J_3 - J_5 - J_2$ since $TCT(\sigma_C) = TCT(J_1 - \boldsymbol{J_4} - \boldsymbol{J_3} - J_5 - J_2) = 1{,}434 < TCT(\sigma_B) = TCT(J_1 - J_3 - J_4 - J_5 - J_2) = 1{,}565$.

Step 4 We set $ic = ic + 1 = 1 + 1 = 2$. Go to Step 2 since the iteration counter ic is smaller than the pre-specified value NI for the number of iterations, i.e., $ic = 2 < NI = 2 \times N = 2 \times 5 = 10$.

Step 2 When we apply the adjacent pairwise interchanges of the jobs in the current schedule $\sigma_2 = J_1 - J_4 - J_3 - J_5 - J_2$, we generate three mutations $\boldsymbol{J_4} - \boldsymbol{J_1} - J_3 - J_5 - J_2, J_1 - J_4 - \boldsymbol{J_5} - \boldsymbol{J_3} - J_2$, and $J_1 - J_4 - J_3 - \boldsymbol{J_2} - \boldsymbol{J_5}$. Note that the mutation $J_1 - \boldsymbol{J_3} - \boldsymbol{J_4} - J_5 - J_2$ is not possible since the pair of (J_3, J_4) is in the tabu list. When we apply the algorithm SCO for each of the three possible mutations, we observe that $min\big\{ TCT\big(\boldsymbol{J_4} - \boldsymbol{J_1} - J_3 - J_5 - J_2\big), TCT(J_1 - J_4 - \boldsymbol{J_5} - \boldsymbol{J_3} - J_2), TCT(J_1 - J_4 - J_3 - \boldsymbol{J_2} - \boldsymbol{J_5}) \big\} = \min\{1{,}530;$ $1{,}561;\ 1{,}512\} = 1{,}512 > TCT(\sigma_B) = TCT(J_1 - J_4 - J_3 - J_5 - J_2) = 1{,}434$. Thus, the TS algorithm terminates before reaching the tabu-search iteration-size of 10.

The total completion time of the schedule obtained by the heuristic algorithm is 1,434 time units and is equal to that of the optimal schedule found by solving the MILP model. Figure 3 illustrates the Gantt chart for this optimal schedule.

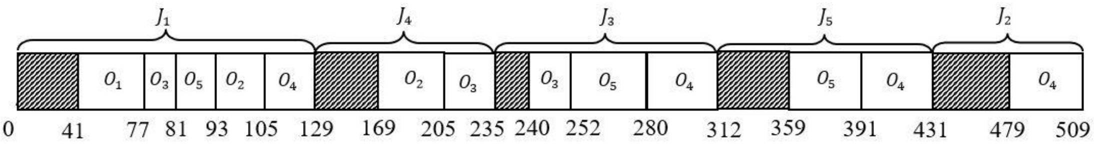


Figure 3 Gantt chart of the schedule for the numerical example problem

CHAPTER 5

COMPUTATIONAL STUDY

In this chapter, we describe our computational experiments to evaluate the performance of the mathematical programming model and the heuristic algorithm in finding optimal or near-optimal schedules. The MILP models for problems P_{JBP} and P_{OBP} are solved by using version 24.1 of the software package General Algebraic Modeling System (GAMS), the proposed heuristic algorithm for solving the problem P_{JBP} is programmed in Python in Visual Studio Code. In addition, the optimal schedule for the problem P'_{OBP} is obtained in Microsoft Excel VBA. All computations are conducted on a computer with Intel(R) Core(TM) i7-9750H processor running at 2.60GHz, with 16 GB of RAM under Windows 10 operating system.

5.1. Problem Instances Design

Problem size is mainly determined by the number of customer orders and the number of jobs (products). We generate the values of the parameters used in our experiments, as in Çetinkaya et al. (2019):

1. *Number of customer orders (K)*: They are taken as 5, 10, 15 and 20.
2. *Number of jobs (N)*: They are taken as 5, 10, 15 and 20.
3. *Number of customer orders having demand for each job ($\|SC_j\|$)*: They are randomly generated from a DU [1, K].
4. *Demand (number of identical items) for each job in each customer order ($D_{o,j}$)*: They are randomly generated from a discrete uniform distribution DU [1, 10].
5. *Unit processing times (t_j)*: They are randomly generated from a discrete uniform distribution DU [1, 10].
6. *Setup times (s_j)*: They are randomly generated from a discrete uniform distribution DU [0, $100f$], where f is taken as 0.5, 1.0, 1.5, and 2.0.

For each possible combination of the above parameters, 25 replicates (problem instances) are generated, and a total of 400 problem instances are tested for the setup case. In addition, 25 replicates are generated for each possible combination of the above parameters, excluding setup times, and a total of 400 problem instances are tested. Hence, the total number of problem instances for both setup and no-setup cases is 800.

5.2. Performance Measures

To measure the effectiveness of two solution approaches, we compared the objective function solutions obtained with the MILP model solved by GAMS and the proposed heuristic algorithm. For the problem instances in which the optimal solution is not obtained, but the best integer solution is achieved by the MILP model, we take the best integer solutions of the MILP to compare with the heuristic solutions. The average, the maximum and minimum deviations of objective values over the optimal solutions (or best integer solutions) are used as the performance measures. For a problem instance k, in which the optimal solution is obtained by the MILP model, we define the percent deviation PD_k of the total completion time obtained by the proposed heuristic algorithm from the total completion time of the optimal solution. That is,

$$PD_k = \frac{(TC_k^H - TC_k^O)}{TC_k^O} \times 100$$

where TC_k^H and TC_k^O are the total completion times of the solutions obtained by the heuristic algorithm and the MILP model, respectively. For the problem instances in which the optimal solution is not obtained (but the best integer solution exists) by the MILP-2 model, TC_k^O is replaced by TC_k^B where TC_k^B is the total completion time of the best integer solution obtained by the MILP model.

The computational time also serves as an efficient measure to compare performances of the MILP and the heuristic algorithm. The average computing time in CPU seconds is calculated in our experiments. The running time of the GAMS's CPLEX solver is set at 10,800 seconds (3 hours), and it exceeds the time limit for the large-sized problem instances. The running time of the proposed heuristic algorithm is recorded for all test problems, and it is relatively very small. The experiments in the following subsections demonstrate that the computational time of the heuristic algorithm increases as both the number of customer orders and jobs are increased. However, the computational time is very small in general, which is less than a minute. On the other hand, the MILP model has much longer computation time when it is compared to heuristic.

5.3. Discussion of the Results

In this section, the performances of the MILP-2 model and the heuristic algorithm for the setup and no-setup cases are presented. These solution approaches are examined concerning the number of customer orders and the number of jobs.

5.3.1. Performance of the MILP-2 Model for the Job-based Processing

5.3.1.1. Setup Case

This part investigates the performance of the MILP-2 model for the setup case when we solve problem instances with job-based processing approach. As shown in Table 3, the MILP-2 gives the optimal solutions for all problem instances up to 15 jobs. As the number of jobs increase, the mathematical model cannot find optimal solutions and exceeds the three-hour time limit. When the number of customer orders is 5, and the number of jobs is 15, there are 19 problem instances with optimal solutions, and there is no optimal solution is found when the number of jobs are increased to 20. For the problem instances with 10 customer orders and 15 jobs, there are 13 problem instances optimally solved. In the same problem set with 20 jobs; however, there is no optimal solution is found. Optimal solutions are obtained for only 2 problem instances when the number of customer orders is 15, and the number of jobs is 15, while no optimal solution is obtained when the number of jobs increased to 20. Lastly, any optimal solution is found for the problem instances of 20 customer orders with 15 jobs and 20 jobs, respectively.

Table 3 Performance of the MILP-2 model for the setup case

K	5				10				15				20			
N	5	10	15	20	5	10	15	20	5	10	15	20	5	10	15	20
Number of problem instances	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
Number of optimum solutions	25	25	19	0	25	25	13	0	25	25	2	0	25	25	0	0
Number of best integer solutions	0	0	6	25	0	0	12	25	0	0	23	25	0	0	25	25
Average gap (%)	0	0	12	24	0	0	12	32	0	0	22	45	0	0	48	58

To emphasize the performance of the MILP-2 model, we should investigate the quality of solutions that are not optimal. It is a common phenomenon that GAMS's CPLEX ends up with a gap between the best integer solution and the best possible solution. Therefore, we examined the gap values to investigate the percent deviation of the integer solution from the theoretical optimum. We analyzed the gap values for 166 non-optimally solved problems which are the problem instances with 15 jobs and 20 jobs. For these problems, so many iterations are done, and integer solutions found become closer to the theoretical optimum after each iteration. However, GAMS is terminated because of time limitation before reaching the optimum solution. Therefore, the gap values are considerable enough for these problem instances. When the number of customer orders are 5 and 10, respectively and the number of jobs are 5 and 10, respectively, all of the gap values equal to zero, which proves that the mathematical model can solve all problem instances optimally.

5.3.1.2. No-setup Case

In this part, we demonstrate the performance of the MILP-2 model for the no-setup case. As shown in Table 4, the mathematical model cannot find the optimal solutions for the problem instances with 15 jobs and 20 jobs regardless of the number of customer orders.

Table 4 Performance of the MILP-2 model for the no-setup case

K	5				10				15				20			
N	5	10	15	20	5	10	15	20	5	10	15	20	5	10	15	20
Number of problem instances	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
Number of optimum solutions	25	25	21	0	25	25	19	0	25	25	7	1	25	25	1	0
Number of best integer solutions	0	0	4	25	0	0	6	25	0	0	18	24	0	0	24	25
Average gap (%)	0	0	4	17	0	0	10	29	0	0	21	42	0	0	45	56

When the number of customer orders are 5, and the number of jobs are 15, there are 21 optimally solved problem instances while there is no optimal solution when the number of jobs is increased to 20. For the problem instances with 10 customer orders, 19 test problems are optimally solved when the number of jobs is 15, and there is no optimal solution is obtained when the number of jobs is 20 for the same problem set. The MILP-2 finds the optimal solution for 7 problem instances when the number of customer orders is 15, and the number of jobs is 15; however, only 1 optimal solution is obtained when the number of jobs is 20. Lastly, when the number of customer orders is 20, and the number of jobs is 15, the model finds the optimal solution for only 1 problem instance. However, any optimal solution is found when the number of jobs is 20. We also investigated the gap values for 151 non-optimally solved problems for the no-setup case. As can be seen in Table 4, the gap values for the no-setup case are also considerable when the number of jobs are increased, as in the setup case.

5.3.2. Performance Evaluation of the Proposed Heuristic Algorithm

In this section, we undertake computational tests in order to gauge the quality of solutions and computational time of the proposed heuristic algorithm. We also investigate a comparative study and solution improvement analysis for the proposed heuristic in the following subsections.

5.3.2.1. Computational Results of the Heuristic Algorithm for the Job-based Processing

5.3.2.1.1. Setup Case

In this part, we compare the computational solutions of the proposed heuristic algorithm with the MILP-2 for the setup case. The comparison of objective function values that are obtained by the proposed heuristic algorithm and the MILP-2 for problem instances when K =5 and N =15 are shown in Table 5. As it was explained before, best integer solutions are used for comparison when any optimal solution is found by the MILP-2.

Table 5 Heuristic solutions compared with MILP-2 solutions for the setup case when *K*=5 and *N*=15

Problem Instance	Total Completion Time		%
	HEURISTIC	MILP-2	
51	5645	5178	9,02
52	7474	7243	3,19
53	4642	4489	3,41
54	8906	8232	8,19
55	6987	6987	0,00
56	8495	8486	0,11
57	6794	6687	1,60
58	8587	8099	6,03
59	7706	7253	6,25
60	13501	13487	0,10
61	9661	9474	1,97
62	8354	8169	2,26
63	6239	6031	3,45
64	9028	8746	3,22
65	6871	6814	0,84
66	14240	13874	2,64
67	6226	5876	5,96
68	8099	7725	4,84
69	8337	8003	4,17
70	11536	11106	3,87
71	4758	4604	3,34
72	6241	6025	3,59
73	9858	9305	5,94
74	12660	12173	4,00
75	5454	5383	1,32
AVG	8251,96	7977,96	3,57
MAX	14240	13874	9,02
MIN	4642	4489	0,00

We can observe that the proposed heuristic has a good performance on finding near-optimal solutions for the problems when *K*=5 and *N*=15. The average deviation from the MILP-2 solutions is %3.57. Maximum and minimum percent deviations are %9.02 and %0.00, respectively.

Table 6 shows the total completion time values obtained by the heuristic and the MILP-2 model for the problem instances when *K* =5 and *N* =20, respectively.

Table 6 Heuristic solutions compared with MILP-2 solutions for the setup case when K=5 and N=20

Problem Instance	Total Completion Time		%
	HEURISTIC	MILP-2	
76	7449	7351	1,33
77	10941	10553	3,68
78	14177	14027	1,07
79	13068	12535	4,25
80	15257	14220	7,29
81	10878	10636	2,28
82	13998	13972	0,19
83	7514	7365	2,02
84	11456	11415	0,36
85	10128	9428	7,42
86	11765	11765	0,00
87	8366	8366	0,00
88	7651	7564	1,15
89	8805	8786	0,22
90	10850	10579	2,56
91	7997	7997	0,00
92	10016	10015	0,01
93	8146	7849	3,78
94	9607	9607	0,00
95	10822	10632	1,79
96	7794	7501	3,91
97	13550	13231	2,41
98	18115	18124	-0,05
99	10855	10112	7,35
100	10484	10368	1,12
AVG	10787,56	10559,92	2,17
MAX	18115	18124	7,42
MIN	7449	7351	-0,05

We can see that the proposed heuristic yields near-optimal solutions when the number of jobs are increased to 20. The average deviation of heuristic solutions from MILP-2 solutions is %2.17. Maximum and minimum percent deviations are %7.42 and %-0.05, respectively. We observed negative percent deviations for some of the non-optimal problems especially for the problem instances having 15 and 20 number of jobs which indicates the heuristic yields better solution than the best integer solution of the MILP-2.

Appendix B provides the remaining tables of our computational tests for the comparison of heuristic and MILP-2 for the setup case.

On the other hand, the summary table of the performance of heuristic for different problem sizes are presented in Table 7.

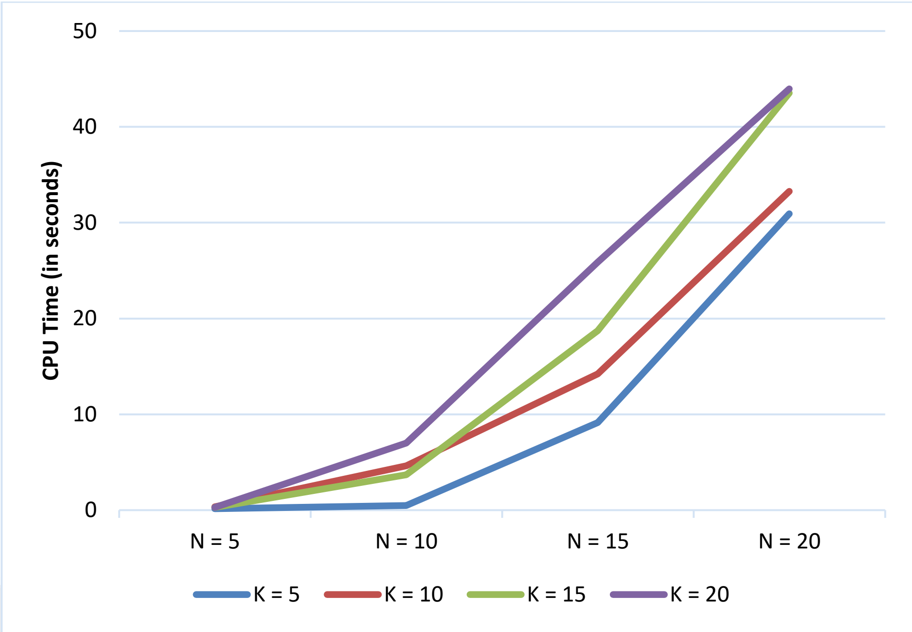
Table 7 The average, maximum and minimum percent deviations between solutions obtained by the heuristic and the MILP-2 for the setup case

K	*N*	Number of Problem Instances	Average Percent Deviation	Maximum Percent Deviation	Minimum Percent Deviation
5	5	25	1,84	11,60	0,00
	10	25	2,40	10,30	0,00
	15	25	3,57	9,02	0,00
	20	25	2,17	7,42	-0,05
Total & Averages		100	2,49	9,59	-0,01
10	5	25	1,15	8,01	0,00
	10	25	3,26	19,10	0,00
	15	25	3,28	28,03	0,00
	20	25	2,96	12,19	-1,34
Total & Averages		100	2,66	16,83	-0,33
15	5	25	1,77	8,28	0,00
	10	25	2,12	7,02	0,00
	15	25	3,35	10,07	0,17
	20	25	3,69	12,40	-0,69
Total & Averages		100	2,73	9,44	-0,13
20	5	25	1,14	2,80	0,00
	10	25	1,91	7,87	0,00
	15	25	1,45	7,41	-4,68
	20	25	2,48	6,07	-0,39
Total & Averages		100	1,74	6,04	-1,27
Total & Grand Averages		400	2,41	10,48	-0,44

The grand averages of the problems when K=5, K=10, K=15, and K=20 are %2.49, %2.66, %2.73, and %1.74, respectively. The average percent deviation for total of 400 problem instances is %2.41 which is relatively low and indicates that the heuristic is very practical for finding near-optimal solutions.

Figure 4 demonstrates the average computational time of the heuristic algorithm for different problem sizes for the setup case. Computational time tends to increase with respect to the number of jobs; however, it is relatively small in general. As a result, the heuristic algorithm yields significantly good results within a much lower computing time when we compare with the mathematical model.

Figure 4 Average CPU time of the heuristic algorithm for the setup case



5.3.2.1.2. No-setup Case

In this part, we compare the computational solutions of our heuristic algorithm with the proposed MILP-2 according to experiments for the no-setup case. Table 8 below shows the total completion time values that are obtained by the heuristic and the MILP-2 when K =5 and N =15, respectively.

Table 8 Heuristic solutions compared with MILP-2 solutions for the no-setup case when K=5 and N=15

Problem Instance	Total Completion Time		%
	HEURISTIC	MILP-2	
51_N	3971	3811	4,20
52_N	6031	5596	7,77
53_N	2276	2174	4,69
54_N	6255	5859	6,76
55_N	2622	2557	2,54
56_N	6828	6828	0,00
57_N	4489	4311	4,13
58_N	5461	5405	1,04
59_N	4339	4134	4,96
60_N	7906	7892	0,18
61_N	5622	5612	0,18
62_N	3776	3663	3,08
63_N	2456	2328	5,50
64_N	7183	7183	0,00
65_N	2098	2083	0,72
66_N	6744	6744	0,00
67_N	4968	4740	4,81
68_N	5020	4867	3,14
69_N	5400	5225	3,35
70_N	6384	6130	4,14
71_N	3384	3351	0,98
72_N	5073	4903	3,47
73_N	4618	4242	8,86
74_N	7254	7224	0,42
75_N	3959	3931	0,71
AVG	4964,68	4831,72	3,03
MAX	7906	7892	8,86
MIN	2098	2083	0,00

As can be seen from the Table 8, total completion time values that are obtained by the heuristic do not differ much than the values obtained by the mathematical model which indicates heuristic still provides near-optimal solutions when we ignore setup times between jobs. The average deviation from the MILP-2 solutions is %3.03. Maximum and minimum percent deviations are %8.86 and %0.00, respectively.

Table 9 illustrates the total completion time values obtained by the heuristic and the MILP-2 for the problem instances when K =5 and N =20.

Table 9 Heuristic solutions compared with MILP solutions for the no-setup case when K=5 and N=20

Problem Instance	Total Completion Time		%
	HEURISTIC	MILP-2	
76_N	5647	5641	0,11
77_N	6065	5977	1,47
78_N	8654	8650	0,05
79_N	7190	6900	4,20
80_N	6123	6122	0,02
81_N	5652	5419	4,30
82_N	6174	5982	3,21
83_N	5708	5613	1,69
84_N	9169	9160	0,10
85_N	5300	5156	2,79
86_N	6228	6228	0,00
87_N	4744	4591	3,33
88_N	3955	3955	0,00
89_N	6952	6737	3,19
90_N	8726	8516	2,47
91_N	6113	5940	2,91
92_N	5974	5950	0,40
93_N	5862	5494	6,70
94_N	7304	7304	0,00
95_N	8425	8347	0,93
96_N	6154	5912	4,09
97_N	7501	7437	0,86
98_N	9265	9226	0,42
99_N	6253	6143	1,79
100_N	8513	8417	1,14
AVG	6706,04	6592,68	1,85
MAX	9265	9226	6,70
MIN	3955	3955	0,00

We can say that the proposed heuristic yields satisfactory results comparing to those of the mathematical model when the number of jobs are increased to 20. The average deviation from the MILP-2's solutions is %1.85. Maximum and minimum percent deviations are %6.70 and %0.00, respectively.
The remaining tables for the solutions of the heuristic algorithm compared with the MILP-2 solutions for the no-setup case are provided in Appendix C.

Lastly, the summary table of the heuristic performance for different problem sizes are presented in Table 10.

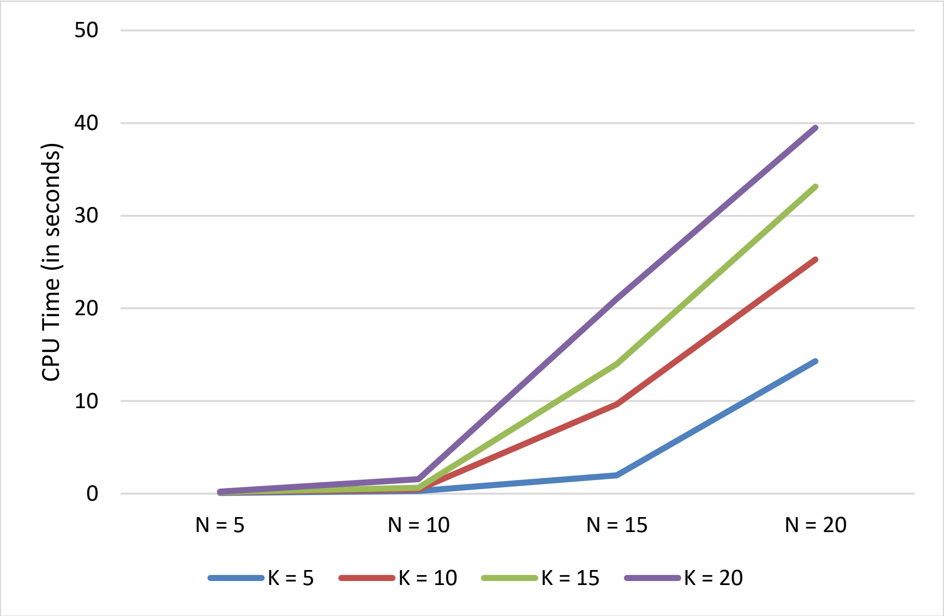
Table 10 The average, maximum and minimum percent deviations between solutions obtained by the heuristic and the MILP-2 for the no-setup case

K	*N*	Number of Problem Instances	Average Percent Deviation	Maximum Percent Deviation	Minimum Percent Deviation
5	5	25	2,06	19,20	0,00
	10	25	2,40	10,30	0,00
	15	25	3,57	8,86	0,00
	20	25	2,17	6,70	0,00
Total & Averages		100	2,55	11,27	0,00
10	5	25	2,58	8,95	0,00
	10	25	2,11	12,69	0,00
	15	25	3,28	10,58	0,00
	20	25	2,27	13,84	0,00
Total & Averages		100	2,56	11,52	0,00
15	5	25	2,10	9,42	0,00
	10	25	2,30	6,60	0,00
	15	25	3,34	9,86	-0,04
	20	25	4,08	16,43	-0,23
Total & Averages		100	2,95	10,58	-0,07
20	5	25	1,31	5,50	0,00
	10	25	1,78	8,49	0,00
	15	25	2,16	7,09	-2,56
	20	25	2,52	9,38	-6,65
Total & Averages		100	1,94	7,62	-2,30
Total & Grand Averages		400	2,50	10,24	-0,59

The grand averages of percent deviations are %2.55, %2.56, %2.95, and %1.94 for the problems when *K*=5, *K*=10, *K*=15, and *K*=20, respectively. The grand percent deviation of the heuristic for the no-setup case is %2.50. Minimum percent deviations are obtained in some of non-optimal problems solved by MILP-2; however, heuristic provides better solutions.

Figure 5 depicts the average computational time of the heuristic algorithm for different problem sizes for the no-setup case. Computational time tends to increase with respect to the number of jobs; whereas, it is small in general.

Figure 5 Average CPU time of the heuristic algorithm for the no-setup case



5.3.2.2 A Comparative Analysis of the Proposed Heuristic Algorithm

In this section, in order to analyze the behavior of the proposed heuristic algorithm, we investigated its search space and compared the results against complete enumeration technique. As we already know that a complete algorithm explores the whole search space however, computational effort raises exponentially. On the other hand, effective heuristic algorithms do not carry out a complete search on the solution space; instead, it explores some part of the solution space using heuristic information within a limited time. Therefore, it is important for us to report how effective our proposed heuristic on finding near-optimal solutions in whole search space. First, we analyzed the number of job sequences generated for each problem instance for setup and no-setup case, respectively. We obtained the number of job sequences generated for each problem instance when finding optimal sequence of jobs by the algorithm. Then, to analyse the search space used by the heuristics, the ratio of generated job sequences divided by all possible number of solutions ($N!$) is defined. It is obvious that there are $N!$ possible solutions of complete job sequences for each problem instance. For instance, there are $5!, 10!, 15!$ and $20!$ possible solutions when the number of jobs are 5, 10, 15 and 20, respectively.

The tables of the computational experiments for setup case and no-setup case are provided in Appendix D and Appendix E, respectively. We can easily deduce from the computational results that the proposed heuristic is very good on finding optimal solutions in a reasonable time. The heuristic carries out only small proportion of the solution space and finds optimal solutions for all problem instances. In other words, the algorithm reaches optimal solutions only within two or three more iterations since the last best solution was found after the Phase-1 of the algorithm.

5.3.2.3 Initial Solution Improvement for the Proposed Heuristic Algorithm

In order to assess the contribution of each phase of the proposed heuristic, we compared the solutions that are obtained from Phase-1 and Phase-2 (Tabu Search) of the algorithm separately. As we said the algorithm works in two phases: Phase 1 prepares heuristic solution which will be used in Phase 2 with tabu-search method thereafter. Table 13 shows the computational results for the improvement in the solutions in Phase-2 (Tabu Search) that are found in Phase-1 for the setup case.

Table 11 Objective Function Improvement in Phase-2 for the setup case

K	*N*	Number of Problem Instances	Number of Improved Solutions in Phase-2	Average Objective Improvement (%)
5	5	25	14	1,95
	10	25	14	0,98
	15	25	12	0,57
	20	25	16	0,51
Total & Averages		100	56	1,00
10	5	25	17	4,08
	10	25	8	0,65
	15	25	10	0,52
	20	25	15	0,59
Total & Averages		100	50	1,46
15	5	25	12	3,22
	10	25	16	0,79
	15	25	11	0,52
	20	25	10	0,22
Total & Averages		100	49	1,18
20	5	25	20	4,48
	10	25	13	0,66
	15	25	16	0,93
	20	25	13	0,43
Total & Averages		100	62	1,63
Total & Grand Averages		400	217	1,32

As expected, Phase-1 has a major impact in the quality of the obtained solutions due to generation of a richer neighborhood in Phase-2. There are 217 problems out of total 400 problem instances for the setup case are improved in Phase-2 of the algorithm. Average improvement percentages are relatively low due to strong initial solution provided in Phase-1 and the grand improvement percentage for a total of 400 problem instances is %1.32.

On the other hand, Table 14 shows the summary results for the improvement in the solutions in Phase-2 (Tabu Search) that are found in Phase-1 for the no-setup case.

Table 12 Objective Function Improvement in Phase-2 for the no-setup case

K	*N*	Number of Problem Instances	Number of Improved Solutions in Phase-2	Average Objective Improvement (%)
5	5	25	13	3,29
	10	25	11	0,77
	15	25	12	0,86
	20	25	13	0,57
Total & Averages		100	49	1,37
10	5	25	14	4,10
	10	25	8	0,87
	15	25	11	0,58
	20	25	14	0,58
Total & Averages		100	47	1,53
15	5	25	14	4,77
	10	25	11	0,53
	15	25	9	0,46
	20	25	6	0,12
Total & Averages		100	40	1,47
20	5	25	18	6,04
	10	25	12	0,72
	15	25	11	0,39
	20	25	12	0,25
Total & Averages		100	53	1,85
Total & Grand Averages		400	189	1,56

As can be seen, the scenario is similar for the no-setup case. There are 189 problems out of 400 problem instances are improved in Phase-2 of the algorithm. The grand improvement percentage is %1.56 which is relatively low again.

The detailed tables for the comparison of two phases of the algorithm for all problem instances for setup case and no-setup case are provided in Appendices F and G, respectively.

5.3.3. Comparison of the Job-based and Order-based Processing Approaches

In this section, we discuss the results of our experiments on the mathematical models of both job-based and order-based processing approaches.

5.3.3.1 Setup Case

First, we analyzed the results of the experiments with setup times. As it was mentioned in Lemma 3 in Chapter 2, the problem P_{OBP} gives optimal sequence of customer orders when there are setup times. Even though there is a significant amount of reduction in setup times in the problem P_{OBP}, the solutions of the problem P_{JBP} outperforms it from the results of the experiments. The job-based processing approach yields better solutions for 310 problems out of 400 test problems when the setup times are involved. Experiment results also demonstrates that large size problems are not optimally solved by the job-based approach, however provides the best integer solutions which are still smaller than the solutions obtained for the problems P_{OBP}. For example, for the problem set with 10 customer orders and 15 jobs, there are 12 non-optimal solutions found by the job-based approach and these solutions are smaller than the order-based approach. As shown in Table 7, job-based processing approach yields negative mean percent deviations which indicate the results obtained by the job-based processing approach is better than the results obtained by the order-based processing approach with setup saving. As the number of customer orders and jobs increase, the percent deviation gets larger and the job-based approach provides better solutions for the setup case.

Table 13 Comparison of the job-based and order-based processing approaches for the setup case

K	N	Number of problem instances	Average % difference
5	5	25	-6,69
	10	25	-7,70
	15	25	-8,68
	20	25	-7,63
Total & Averages		100	-7,67
10	5	25	-7,05
	10	25	-8,75
	15	25	-15,50
	20	25	-24,00
Total & Averages		100	-13,82
15	5	25	-9,53
	10	25	-11,35
	15	25	-15,88
	20	25	-17,08
Total & Averages		100	-13,46
20	5	25	-12,50
	10	25	-14,70
	15	25	-18,26
	20	25	-19,30
Total & Averages		100	-16,19
Total & Grand Averages		400	-12,79

5.3.3.2 No-setup Case

The results of the problem instances when there is no setup time between jobs differ significantly. As it was described in Remark 1 in Section 2, the problem P_{OBP} turns into the problem P'_{OBP} when we ignore setup times. Thus, both problems provide the same sequence of customer orders for the problem instances. As can be seen in Table 8, in contrast, the difference between job-based and order-based processing approaches now yields high positive average percent deviations between the solutions, which indicate that the order-based processing approach is better than the job-based processing approach when we ignore setup times in the same problem instances.

Table 14 Comparison of the job-based and order-based processing approaches for the no-setup case

K	*N*	Number of problem instances	Average % difference
5	5	25	23,18
	10	25	40,74
	15	25	47,98
	20	25	50,92
Total & Averages		100	40,71
10	5	25	32,31
	10	25	34,49
	15	25	54,91
	20	25	58,53
Total & Averages		100	45,06
15	5	25	29,73
	10	25	38,49
	15	25	69,94
	20	25	74,25
Total & Averages		100	53,10
20	5	25	30,52
	10	25	41,05
	15	25	79,76
	20	25	82,15
Total & Averages		100	58,37
Total & Grand Averages		400	49,31

We can deduce that customer order scheduling with the job-based processing approach yields better results when there is setup time. On the other hand, the order-based processing approach is more preferable when there is no setup time. However, the importance of setup times in production scheduling cannot be underestimated. Therefore, manufacturers or decision-makers should tailor processing methods to their needs for effective scheduling of customer orders.

CHAPTER 6

CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

In this study, we consider a customer order scheduling problem in a single machine to find a schedule with a sequence of jobs and the sequence of customer orders in each job when the job-based processing approach is used and compare this schedule with the schedule having the order-based processing approach. The total completion time of the customer orders is minimized in each processing approach.

We have proved that the problem P'_{OBP} with order-based processing in a single-machine environment is easy and polynomial-time solvable, and developed two MILP models and a tabu-search based heuristic algorithm that obtain optimal and near-optimal solutions, respectively, for the problem P_{JBP}. Our empirical study shows that the second model (MILP-2) finds optimal solutions for problems up to 10 jobs regardless of what the number of customer orders is in less than 3 hours of CPU time. However, there are problems with 15 and 20 jobs were not solved optimally. From these observations, it is clear that solving the problem with a standard MILP solver seems to be ineffective, especially for large-sized problem instances. The results also show that our proposed heuristic algorithm provides satisfactory solutions as it solves small and medium-sized problem instances optimally and finds near-optimal solutions for large-sized instances in a very short computational time.

We have also compared the order-based and job-based processing approaches, and observed that the job-based processing approach gives better results than the order-based processing approach when a setup on the machine is needed before starting to process each job (product). On the other hand, if there is no-setup, our observation was reversed towards the order-based processing as we expected.

We believe that there are several fruitful issues for future research in the customer order scheduling problem with job-based processing. First, it would be interesting to develop a branch and bound algorithm as another exact solution procedure for the job-based processing problem P_{JBP} considered in our study. Second, the complexity of the problem P_{JBP} is open for future investigation. Third, more elaborated metaheuristics, such as simulated annealing and genetic algorithm, could be developed and compared with our tabu-search algorithm. Fourth, total tardiness, maximum lateness, and the number of tardy customer orders could be other scheduling criteria to be investigated if there are due dates for the customer orders. Finally, considering the job-based processing approach on more complex machining environments, including parallel machines, flow shop, job shop, and open shop, would be other subjects of future study.

REFERENCES

Ahmadi, R.H., and Bagchi, U. (1990). Scheduling of multi-job customer orders in multi-machine environments. *ORSA/TIMS*, Philadelphia.

Akkocaoğlu, H. (2014). *A new customer order scheduling problem on a single-machine with job setup times*, MS Thesis, Çankaya University.

Allahverdi, A. (2015). The third comprehensive survey on scheduling with setup times/costs. *European Journal of Operational Research*, *246*(2), 345–378.

Bagchi, U., Julien, F.M., & Magazine, M.J. (1994). Note: due-date assignment to multi-job customer orders. *Management Science*, *40*(10), 1389–1392.

Baker, K.R. (1988). Scheduling the production of components at a common facility. *IIE Transactions*, *20*(1), 32–35.

Cheng, T.C.E, Kovalyov, M.Y. & Tuzikov, A.V. (1996). Single machine group scheduling with two ordered criteria. *The Journal of the Operational Research Society*, *47*(2), 315.

Coffman, E.G., Nozari, A., & Yannakakis, M. (1989). Optimal scheduling of products with two subassemblies on a single machine. *Operations Research*, *37*(3), 426-436.

Çetinkaya, F.C., Çatmakaş, H.A., & Görür, A.K. (2019). Single-machine scheduling of indivisible multi-operation jobs. *South African Journal of Industrial Engineering*, *30*(3), 78–93.

Erel, E., & Ghosh, J.B. (2007). Customer order scheduling on a single machine with family setup times: Complexity and algorithms. *Applied Mathematics and Computation*, *185*(1), 11–18.

Gerodimos, A.E., Glass, C.A., Potts, C.N., & Tautenhahn, T. (1999). Scheduling multi-operation jobs on a single machine. *Annals of Operations Research*, *92*(1–4), 87–105.

Glover F. (1989). Tabu search - Part I. *ORSA Journal of Computing*, *1*, 190–206

Gupta, J.N.D. (1988). Single facility scheduling with multiple job classes. *European Journal of Operational Research*, *8*, 42–45.

Gupta, J.N.D., Ho, J.C., & van der Veen, J.A.A. (1997). Single machine hierarchical scheduling with customer orders and multiple job classes. *Annals of Operations Research*, *70*(1–4), 127–143.

Gupta, J.N.D., & Chantaravarapan, S. (2008). Single machine group scheduling with family setups to minimize total tardiness. *International Journal of Production Research*, *46*(6), 1707–1722.

Hazır, O., Günalay, Y., & Erel, E. (2008). Customer order scheduling problem: A comparative metaheuristics study. *International Journal of Advanced Manufacturing Technology*, *3*(5-6), 589–598.

He, Y., & Sun, L. (2012). Single-machine group scheduling problems with deterioration to minimize the sum of completion times. *Mathematical Problems in Engineering*, 1–9.

He, Y., & Sun, L. (2015). One-machine scheduling problems with deteriorating jobs and position-dependent learning effects under group technology considerations. *International Journal of Systems Science*, *46*(7), 1319–1326.

Julien, F.M., & Magazine, M.J. (1990). Scheduling customer orders: an alternative production scheduling approach. *Journal of Manufacturing and Operations Management*, *3*, 177–199.

Karabati, S., & Akkan, C. (2006). Minimizing sum of completion times on a single machine with sequence-dependent family setup times. *Journal of the Operational Research Society*, *57*(3), 271–280.

Liao, C.J., & Chuang, C.H. (1996). Sequencing with setup time and order tardiness trade-offs. *Naval Research Logistics*, *43*(7), 971–984.

Liu, C.H. (2009). Lot streaming for customer order scheduling problem in job shop environments. *International Journal of Computer Integrated Manufacturing 22*(9), 890–907.

Liu, F., Yang, J., & Lu, Y.Y. (2019). Solution algorithms for single-machine group scheduling with ready times and deteriorating jobs. *Engineering Optimization*, *51*(5), 862–874.

Mason, S.J., & Chen, J.S. (2010). Scheduling multiple orders per job in a single machine to minimize total completion time, *European Journal of Operational Research*, *207*(1), 70–77.

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

Nawaz, M., Enscore, E., & Ham, I. (1983). A heuristic for the m-machine, n-job flow shop sequencing problem, *Omega*, *5*(11), 91.

Neufeld, J. S., Gupta, J.N.D., & Buscher, U. (2016). A comprehensive review of flowshop group scheduling literature. *Computers & Operations Research*, *70*(1), 56–74.

Ng, D., Cheng, E.T., & Kovalyov, M.Y. (2004). Single machine batch scheduling with jointly compressible setup and processing times. *European Journal of Operational Research*, *162*, 211–219.

Pinedo, M.L. **(**2008). *Scheduling: Theory, Algorithms and Systems*. Springer: New York.

Potts, C.N. (1991). Scheduling two job classes on a single machine. *Computers & Operations Research*, *18*(5), 411–415.

Potts, C.N., & Kovalyov, M.Y. (2000). Scheduling with batching: A review. *European Journal of Operational Research*, *120*(2), 228–249.

Sabouni, Y.M., & Logendran, R. (2013). A single machine carryover sequence-dependent group scheduling in PCB manufacturing. *Computers & Operations Research*, *40*(1), 236–247.

Santos, C. & Magazine, M. (1985). Batching in single operation manufacturing systems, *Operations Research Letters*, *4*(3), 99–103.

Smith, W.E. **(**1956). Various optimizers for single stage production. *Naval Research Logistics Quarterly*, *3*(1), 59–66.

Vickson, R.G., Magazine, M.J., & Santos, C.A. (1993) Batching and sequencing of components at a single facility. *IIE Transactions*, *25*(2), 65–70.

Wang, J.B., Huang, X., Wu, Y.B., & Ji, P. (2012). Group scheduling with independent setup times, ready times, and deteriorating job processing times. *International Journal of Advanced Manufacturing Technology*, *60*(5–8), 643–649.

Wang, J.B., & Wang, J.J. (2014). Single machine group scheduling with time dependent processing times and ready times. *Information Sciences*, *275*, 226–231.

Webster, S., and Baker, K.R. (1995). Scheduling groups of jobs on a single machine. *Operations Research*, *43*(4), 692–703.

Wu, C.C., & Lee, W.C. (2006). Single machine scheduling to minimize the setup time and the earliness. *Journal of Information & Optimization Sciences*, *27*(2), 499–510.

Xu, X., Ma, Y., Zhou, Z., & Zhao, Y. (2015). Customer order scheduling on unrelated parallel machines to minimize total completion time, *IEEE Transactions on Automation Science and Engineering*, *12*(1), 244–257.

Yang, D.L., Hou, Y.T., & Kuo, W.H. (2017). A note on a single-machine lot scheduling problem with indivisible orders. *Computers and Operations Research*, *79*, 34–38.

Yang, J. (2011). Customer order scheduling in a two machine flowshop, *International Journal of Management Science*, *17*(1), 1921–1939.

Yozgat, G. (2018). *Customer order scheduling with lot streaming to minimize the total order completion time in a two-machine flow shop*, MS Thesis, Çankaya University.

APPENDICES

APPENDIX A – MATHEMATICAL MODEL FOR THE PROBLEM P_{OBP}

Parameters, indices and sets

K Number of customer orders.

i Index for customer orders ($i = 1,2, \dots, K$).

j Index for jobs. ($j = 1,2, \dots, N$).

k Index for position of customer orders in the sequence ($k = 1,2, \dots, K$).

$D_{i,j}$ $D_{i,j} = 1$ if customer order O_i has job J_j; otherwise, $D_{i,j} = 0$

p_j Processing time for job J_j.

s_j Setup time for job J_j.

N_i Set of different jobs in customer O_i.

A Set of customer orders having more than one job to be processed.

TT_i Total (sum of setup and processing) time of all jobs in customer order O_i, where $TT_i = \sum_{J_j \in O_i} (s_j + p_j)$

ST_{hj} Setup time between jobs J_h and J_j if job J_j immediately follows job J_h, where $ST_{hj} = s_j$ if $j \neq h$; otherwise, $ST_{hj} = 0$.

Decision variables

$$X_{ik} = \begin{cases} 1 \text{ if customer order } O_i \text{ assigned to position } k \\ 0 \text{ otherwise} \end{cases}$$

$$\mathrm{F_{jik}} = \begin{cases} 1 \text{ if job } J_j \text{is the first job in customer order } O_i \text{ assigned to position } k \\ 0 \text{ otherwise} \end{cases}$$

$$L_{jik} = \begin{cases} 1 \text{ if job } J_j \text{is the last job in customer order } O_i \text{ assigned to position } k \\ 0 \text{ otherwise} \end{cases}$$

$$Y_{hijlk} = \begin{cases} 1 \text{ if both } L_{hik} \text{and } F_{jl,k+1} \text{are equal to 1 (i.e., last job of a customer order} \\ \quad \text{and the first job of the immediately following custmer order are not same.)} \\ 0 \text{ otherwise} \end{cases}$$

RT_{ik} Realized total (sum of setup and processing) time of customer orders O_i assigned to position *k*.

TC Total completion time of customer orders.

MILP model

Minimize $TC = \sum_{i=1}^{K}(K-k+1)\sum_{i=1}^{K}(RT_{ik})$ (A.1)

Subject to $\sum_{i=1}^{K} X_{ik} = 1$ for $k = 1,2,\dots,K$ (A.2)

$\sum_{k=1}^{K} X_{ik} = 1$ for $i = 1,2,\dots,K$ (A.3)

$\sum_{\boldsymbol{j \in N_i}} \sum_{i=1}^{K} F_{jik} = 1$ for $k = 1,2,\dots,K$ (A.4)

$\sum_{\boldsymbol{j \in N_i}} \sum_{i=1}^{K} L_{jik} = 1$ for $k = 1,2,\dots,K$ (A.5)

$F_{jik} \le D_{lj} X_{lk}$ for $j \in N_i; i = 1,2,\dots,K;\ k = 1,2,\dots,K$ (A.6)

$L_{jik} \le D_{lj} X_{lk}$ for $j \in N_i; i = 1,2,\dots,K;\ k = 1,2,\dots,K$ (A.7)

$L_{hik} + F_{jlk+1} - 1 \le Y_{hijlk}$ for $j \in N_i; h \in N_i; j \ne h; i = 1,2,\dots,K$

$l = 1,2,\dots,K; l \ne i;\ k = 1,2,\dots,K$ (A.8)

$F_{jik} + L_{jik} \le 1$ for $j \in N_i; i \in A; k = 1,2,\dots,K$ (A.9)

$RT_{i1} \ge TT_i X_{i1}$ for $i = 1,2,\dots,K$ (A.10)

$RT_{ik} \ge TT_i X_{ik} - \sum_{j \in N_i} s_j\, F_{jik} \sum_{h \in N_i} \sum_{j \in N_i} \sum_{l=1}^{K} ST_{hj} Y_{hijlk-1}$

for $i = 1,2,\dots,K; k \ge 2$ (A.11)

$RT_{ik} \ge 0$ for $\forall i, k$ (A.12)

$X_{ik}, F_{jik}, L_{jik}, Y_{hijlk} \in \{0,1\}$ for $\forall\ h, i, j, k, l$ (A.13)

“In the above MILP model, the objective in (A.1) is to minimize the total completion time. Constraint sets (A.2) and (A.3) ensure that each position in the sequence of customer orders is occupied by one customer only and each customer order is assigned to one position only, respectively. Constraint sets (A.4) and (A.5) guarantee only one job in each customer order can be processed as the first or last job in its customer order, respectively. Constraint sets (A.6) and (A.7) ensure that a job cannot be the first or last job of a customer order assigned to a position if this customer order does not include the job. Constraint set (A.8) satisfies the condition that no setup time is necessary before the processing of the first job of a customer order if this first job is same as the last job of the immediately preceding customer order. Constraint set (A.9) guarantees that each job in a customer order can be the first, immediate or last job of this customer order. Constraint sets (A.10) and (A.11) define the realized total (sum of setup and processing) time of the customer orders assigned to the first and other positions, respectively. Constraint sets (A.12) and (A.13) impose non-negativity and binary restrictions on the decision variables, respectively.” (Akkocaoğlu, 2014, p.22-25)

APPENDIX B – TOTAL COMPLETION TIME VALUES OBTAINED BY THE HEURISTIC AND THE MILP-2 FOR THE SETUP CASE

Table B.1 Total Completion Time Values when *K*=5

K = 5, N = 5			
Problem Instanc	Total Completion Time HEURISTIC	MILP	%
1	653	653	0,00
2	1780	1759	1,19
3	1760	1577	11,60
4	1907	1907	0,00
5	2812	2812	0,00
6	1434	1434	0,00
7	2892	2779	4,07
8	1338	1338	0,00
9	3946	3854	2,39
10	1573	1573	0,00
11	3198	3198	0,00
12	1809	1809	0,00
13	1596	1596	0,00
14	2579	2579	0,00
15	4497	4177	7,66
16	2833	2699	4,96
17	1174	1174	0,00
18	2842	2570	10,58
19	3531	3460	2,05
20	986	986	0,00
21	3354	3326	0,84
22	2005	1994	0,55
23	3163	3163	0,00
24	3700	3700	0,00
25	1958	1958	0,00
AVG	2372,8	2323	1,84
MAX	4497	4177	11,60
MIN	653	653	0,00

K = 5, N = 10			
Problem Instanc	Total Completion Time HEURISTIC	MILP	%
26	4683	4555	2,81
27	5507	5491	0,29
28	3554	3554	0,00
29	4615	4615	0,00
30	4179	4119	1,46
31	5712	5511	3,65
32	5031	4885	2,99
33	2988	2888	3,46
34	6701	6528	2,65
35	6179	6104	1,23
36	5300	5300	0,00
37	3631	3631	0,00
38	6690	6346	5,42
39	4008	4008	0,00
40	3300	3196	3,25
41	3526	3391	3,98
42	4919	4751	3,54
43	5881	5825	0,96
44	3684	3340	10,30
45	4287	4109	4,33
46	2653	2599	2,08
47	6097	6072	0,41
48	5049	4929	2,43
49	6853	6536	4,85
50	5414	5414	0,00
AVG	4817,64	4707,88	2,40
MAX	6853	6536	10,30
MIN	2653	2599	0,00

K = 5, N = 15			
Problem Instance	Total Completion Time HEURISTIC	MILP	%
51	5645	5178	9,02
52	7474	7243	3,19
53	4642	4489	3,41
54	8906	8232	8,19
55	6987	6987	0,00
56	8495	8486	0,11
57	6794	6687	1,60
58	8587	8099	6,03
59	7706	7253	6,25
60	13501	13487	0,10
61	9661	9474	1,97
62	8354	8169	2,26
63	6239	6031	3,45
64	9028	8746	3,22
65	6871	6814	0,84
66	14240	13874	2,64
67	6226	5876	5,96
68	8099	7725	4,84
69	8337	8003	4,17
70	11536	11106	3,87
71	4758	4604	3,34
72	6241	6025	3,59
73	9858	9305	5,94
74	12660	12173	4,00
75	5454	5383	1,32
AVG	8251,96	7977,96	3,57
MAX	14240	13874	9,02
MIN	4642	4489	0,00

K = 5, N = 20			
Problem Instance	Total Completion Time HEURISTIC	MILP	%
76	7449	7351	1,33
77	10941	10553	3,68
78	14177	14027	1,07
79	13068	12535	4,25
80	15257	14220	7,29
81	10878	10636	2,28
82	13998	13972	0,19
83	7514	7365	2,02
84	11456	11415	0,36
85	10128	9428	7,42
86	11765	11765	0,00
87	8366	8366	0,00
88	7651	7564	1,15
89	8805	8786	0,22
90	10850	10579	2,56
91	7997	7997	0,00
92	10016	10015	0,01
93	8146	7849	3,78
94	9607	9607	0,00
95	10822	10632	1,79
96	7794	7501	3,91
97	13550	13231	2,41
98	18115	18124	-0,05
99	10855	10112	7,35
100	10484	10368	1,12
AVG	10787,56	10559,92	2,17
MAX	18115	18124	7,42
MIN	7449	7351	-0,05

Table B.2 Total Completion Time Values when *K*=10

K = 10, N = 5

Problem Instance	Total Completion Time HEURISTIC	MILP	%
101	5002	5002	0,00
102	3201	3176	0,79
103	3069	3039	0,99
104	2929	2925	0,14
105	3157	3125	1,02
106	7031	6859	2,51
107	5350	5350	0,00
108	11701	11686	0,13
109	2163	2097	3,15
110	3063	3063	0,00
111	4060	4036	0,59
112	2476	2466	0,41
113	5756	5329	8,01
114	6728	6728	0,00
115	4666	4666	0,00
116	2194	2158	1,67
117	5063	5004	1,18
118	8405	8405	0,00
119	5050	5026	0,48
120	9494	9426	0,72
121	3563	3399	4,82
122	4419	4401	0,41
123	4728	4728	0,00
124	8269	8126	1,76
125	8543	8543	0,00
AVG	5203,2	5150,52	1,15
MAX	11701	11686	8,01
MIN	2163	2097	0,00

K = 10, N = 10

Problem Instance	Total Completion Time HEURISTIC	MILP	%
126	5857	5857	0,00
127	15574	15444	0,84
128	13850	13651	1,46
129	13749	13436	2,33
130	13353	12309	8,48
131	7317	7126	2,68
132	19765	19182	3,04
133	12468	12468	0,00
134	13643	11455	19,10
135	17220	16541	4,10
136	11310	10295	9,86
137	17797	17602	1,11
138	21184	20700	2,34
139	22871	22802	0,30
140	12380	11870	4,30
141	13571	13571	0,00
142	8840	8840	0,00
143	25375	23647	7,31
144	16480	16480	0,00
145	17372	16304	6,55
146	12590	12001	4,91
147	18381	18381	0,00
148	20725	20173	2,74
149	10669	10669	0,00
150	15182	15182	0,00
AVG	15100,92	14639,44	3,26
MAX	25375	23647	19,10
MIN	5857	5857	0,00

K = 10, N = 15

Problem Instance	Total Completion Time HEURISTIC	MILP	%
151	21102	21081	0,10
152	24131	24067	0,27
153	19291	19291	0,00
154	21935	21839	0,44
155	18206	18206	0,00
156	22111	21756	1,63
157	29343	29201	0,49
158	36339	34607	5,00
159	19355	19174	0,94
160	29330	29098	0,80
161	30704	28638	7,21
162	24555	23911	2,69
163	28514	28318	0,69
164	22045	21210	3,94
165	40240	38032	5,81
166	35315	34513	2,32
167	28213	28100	0,40
168	17524	17288	1,37
169	20241	20159	0,41
170	26240	25313	3,66
171	21904	17108	28,03
172	20817	20710	0,52
173	32861	31013	5,96
174	18913	18873	0,21
175	37731	34566	9,16
AVG	25878,4	25042,88	3,28
MAX	40240	38032	28,03
MIN	17524	17108	0,00

K = 10, N = 20

Problem Instance	Total Completion Time HEURISTIC	MILP	%
176	32683	29494	10,81
177	45725	40949	11,66
178	38426	37258	3,13
179	32709	32333	1,16
180	38009	36916	2,96
181	32848	32052	2,48
182	34268	34184	0,25
183	36233	35713	1,46
184	37863	37863	0,00
185	22141	21744	1,83
186	40072	37441	7,03
187	35726	35302	1,20
188	40543	39582	2,43
189	40329	40278	0,13
190	49245	48114	2,35
191	30112	30355	-0,80
192	32399	32320	0,24
193	30748	30547	0,66
194	51566	52266	-1,34
195	26032	26032	0,00
196	40118	37449	7,13
197	23791	23641	0,63
198	44191	44431	-0,54
199	21019	18736	12,19
200	33682	31476	7,01
AVG	35619,12	34659,04	2,96
MAX	51566	52266	12,19
MIN	21019	18736	-1,34

Table B.3 Total Completion Time Values when *K*=15

K = 15, N = 5

Problem Instance	Total Completion Time HEURISTIC	MILP	%
201	9032	9032	0,00
202	10092	9919	1,74
203	7897	7867	0,38
204	8100	8070	0,37
205	9016	8946	0,78
206	8631	8232	4,85
207	16792	16612	1,08
208	14728	14527	1,38
209	10795	10488	2,93
210	8319	8115	2,51
211	16513	16513	0,00
212	8188	8128	0,74
213	11507	11045	4,18
214	15094	14750	2,33
215	20760	19363	7,21
216	8992	8928	0,72
217	5385	5319	1,24
218	19169	19145	0,13
219	15193	15177	0,11
220	9168	9139	0,32
221	10425	10282	1,39
222	10011	9920	0,92
223	14301	14191	0,78
224	22268	20565	8,28
225	13637	13637	0,00
AVG	12160,52	11916,4	1,77
MAX	22268	20565	8,28
MIN	5385	5319	0,00

K = 15, N = 10

Problem Instance	Total Completion Time HEURISTIC	MILP	%
226	20312	19880	2,17
227	31861	31534	1,04
228	25249	24346	3,71
229	21442	20892	2,63
230	24298	24298	0,00
231	17517	17293	1,30
232	47009	44045	6,73
233	32246	32009	0,74
234	25015	24060	3,97
235	29301	28640	2,31
236	36780	36780	0,00
237	25703	25440	1,03
238	38854	38537	0,82
239	41008	40410	1,48
240	35465	34268	3,49
241	26463	26436	0,10
242	20226	18899	7,02
243	50105	50105	0,00
244	39918	38945	2,50
245	30734	30524	0,69
246	21591	21591	0,00
247	35315	34272	3,04
248	34529	33956	1,69
249	37871	36052	5,05
250	30672	30233	1,45
AVG	31179,36	30537,8	2,12
MAX	50105	50105	7,02
MIN	17517	17293	0,00

K = 15, N = 15

Problem Instance	Total Completion Time HEURISTIC	MILP	%
251	36063	35775	0,81
252	46021	44945	2,39
253	37403	35783	4,53
254	35149	34513	1,84
255	49996	49473	1,06
256	35872	35181	1,96
257	39895	39339	1,41
258	68526	65295	4,95
259	49618	48207	2,93
260	38687	36063	7,28
261	53844	50083	7,51
262	50265	50179	0,17
263	44343	43182	2,69
264	61127	58511	4,47
265	66086	63960	3,32
266	52527	48347	8,65
267	37220	34986	6,39
268	43991	43787	0,47
269	74889	73593	1,76
270	54415	54200	0,40
271	49413	44891	10,07
272	38276	36939	3,62
273	55332	53870	2,71
274	61458	60882	0,95
275	48045	47385	1,39
AVG	49138,44	47574,8	3,35
MAX	74889	73593	10,07
MIN	35149	34513	0,17

K = 15, N = 20

Problem Instance	Total Completion Time HEURISTIC	MILP	%
276	48353	47859	1,03
277	65876	58611	12,40
278	51720	48788	6,01
279	46698	45029	3,71
280	47972	46050	4,17
281	51716	50267	2,88
282	88109	87993	0,13
283	69086	67906	1,74
284	53477	49174	8,75
285	64112	61798	3,74
286	74841	72248	3,59
287	59174	54634	8,31
288	75966	75142	1,10
289	85177	82162	3,67
290	71141	69496	2,37
291	48890	48447	0,91
292	54544	53548	1,86
293	103059	97635	5,56
294	75601	73923	2,27
295	62208	60416	2,97
296	56002	49987	12,03
297	71476	70805	0,95
298	78655	79202	-0,69
299	81601	80339	1,57
300	69810	68960	1,23
AVG	66210,56	64016,8	3,69
MAX	103059	97635	12,40
MIN	46698	45029	-0,69

Table B.4 Total Completion Time Values when *K*=20

K = 20, N = 5			
Problem	Total Completion Time		%
Instance	HEURISTIC	MILP	
301	14592	14465	0,88
302	18124	17898	1,26
303	15123	15093	0,20
304	13071	12934	1,06
305	14384	14251	0,93
306	14670	14342	2,29
307	31381	31094	0,92
308	21539	21150	1,84
309	21333	20751	2,80
310	15337	14973	2,43
311	31723	31723	0,00
312	15626	15476	0,97
313	18117	17838	1,56
314	26046	26046	0,00
315	34394	33740	1,94
316	17385	17241	0,84
317	9472	9382	0,96
318	28941	28941	0,00
319	29743	29727	0,05
320	13195	13137	0,44
321	19988	19521	2,39
322	17656	17548	0,62
323	25814	25711	0,40
324	36001	35104	2,56
325	24824	24510	1,28
AVG	21139,16	20903,8	1,14
MAX	36001	35104	2,80
MIN	9472	9382	0,00

K = 20, N = 10			
Problem	Total Completion Time		%
Instance	HEURISTIC	MILP	
326	45505	44557	2,13
327	57103	57023	0,14
328	43362	42173	2,82
329	37616	37390	0,60
330	40356	40251	0,26
331	33645	32621	3,14
332	76916	74550	3,17
333	55961	54076	3,49
334	43813	43617	0,45
335	51366	51283	0,16
336	63910	63910	0,00
337	47004	46599	0,87
338	72653	71993	0,92
339	70104	69702	0,58
340	63192	61406	2,91
341	44541	41872	6,37
342	37742	34988	7,87
343	82326	82326	0,00
344	62040	60354	2,79
345	45041	45041	0,00
346	34337	34328	0,03
347	61241	59792	2,42
348	59050	57436	2,81
349	65534	63984	2,42
350	56334	55566	1,38
AVG	54027,68	53073,5	1,91
MAX	82326	82326	7,87
MIN	33645	32621	0,00

K = 20, N = 15			
Problem	Total Completion Time		%
Instance	HEURISTIC	MILP	
351	70292	68130	3,17
352	88190	83654	5,42
353	54369	57038	-4,68
354	61200	59074	3,60
355	84809	84421	0,46
356	63364	62445	1,47
357	70573	69974	0,86
358	117458	117187	0,23
359	86818	85727	1,27
360	72483	71245	1,74
361	94092	94275	-0,19
362	83853	83566	0,34
363	71893	69883	2,88
364	115843	113780	1,81
365	101816	101184	0,62
366	90747	90902	-0,17
367	67434	67269	0,25
368	79050	78607	0,56
369	123868	123820	0,04
370	96003	94758	1,31
371	82706	76997	7,41
372	61539	61539	0,00
373	88416	86070	2,73
374	117279	113886	2,98
375	85356	83664	2,02
AVG	85178,04	83963,8	1,45
MAX	123868	123820	7,41
MIN	54369	57038	-4,68

K = 20, N = 20			
Problem	Total Completion Time		%
Instance	HEURISTIC	MILP	
376	99349	95171	4,39
377	108955	103001	5,78
378	93625	93130	0,53
379	83128	80693	3,02
380	79499	79211	0,36
381	87626	85188	2,86
382	148070	148654	-0,39
383	121428	121583	-0,13
384	93766	89877	4,33
385	111563	106747	4,51
386	128868	124374	3,61
387	100658	98583	2,10
388	129241	128543	0,54
389	150364	146458	2,67
390	126898	119634	6,07
391	88219	88130	0,10
392	95808	95193	0,65
393	179506	174724	2,74
394	140425	133101	5,50
395	104990	103373	1,56
396	96531	93537	3,20
397	126770	125628	0,91
398	148261	147245	0,69
399	150070	143854	4,32
400	134845	132198	2,00
AVG	117138,52	114313	2,48
MAX	179506	174724	6,07
MIN	79499	79211	-0,39

APPENDIX C – TOTAL COMPLETION TIME VALUES OBTAINED BY THE HEURISTIC AND THE MILP-2 FOR THE NO-SETUP CASE

Table C.1 Total Completion Time Values when *K*=5

K = 5, N = 5

Problem Instance	Total Completion Time HEURISTIC	Total Completion Time MILP	%
1_N	159	137	16,06
2_N	1142	1121	1,87
3_N	1232	1232	0,00
4_N	615	615	0,00
5_N	603	603	0,00
6_N	912	912	0,00
7_N	2361	2323	1,64
8_N	617	617	0,00
9_N	1835	1737	5,64
10_N	1146	1146	0,00
11_N	2683	2683	0,00
12_N	1334	1334	0,00
13_N	776	776	0,00
14_N	1584	1568	1,02
15_N	2595	2177	19,20
16_N	1489	1455	2,34
17_N	649	649	0,00
18_N	1230	1230	0,00
19_N	2608	2608	0,00
20_N	325	323	0,62
21_N	1683	1653	1,81
22_N	1107	1107	0,00
23_N	1740	1740	0,00
24_N	1947	1947	0,00
25_N	1555	1535	1,30
AVG	1357,08	1329,12	2,06
MAX	2683	2683	19,20
MIN	159	137	0,00

K = 5, N = 10

Problem Instance	Total Completion Time HEURISTIC	Total Completion Time MILP	%
26_N	3787	3438	10,15
27_N	4315	4313	0,05
28_N	2118	2118	0,00
29_N	2546	2546	0,00
30_N	1979	1979	0,00
31_N	3483	3445	1,10
32_N	2981	2888	3,22
33_N	1792	1723	4,00
34_N	3321	3278	1,31
35_N	3690	3611	2,19
36_N	3470	3470	0,00
37_N	2602	2602	0,00
38_N	5525	5254	5,16
39_N	2068	2068	0,00
40_N	2191	2119	3,40
41_N	2573	2504	2,76
42_N	3013	2932	2,76
43_N	2645	2610	1,34
44_N	1323	1309	1,07
45_N	1622	1528	6,15
46_N	1878	1873	0,27
47_N	1846	1821	1,37
48_N	2265	2226	1,75
49_N	3236	3063	5,65
50_N	4388	4388	0,00
AVG	2826,28	2764,24	2,15
MAX	5525	5254	10,15
MIN	1323	1309	0,00

K = 5, N = 15

Problem Instance	Total Completion Time HEURISTIC	Total Completion Time MILP	%
51_N	3971	3811	4,20
52_N	6031	5596	7,77
53_N	2276	2174	4,69
54_N	6255	5859	6,76
55_N	2622	2557	2,54
56_N	6828	6828	0,00
57_N	4489	4311	4,13
58_N	5461	5405	1,04
59_N	4339	4134	4,96
60_N	7906	7892	0,18
61_N	5622	5612	0,18
62_N	3776	3663	3,08
63_N	2456	2328	5,50
64_N	7183	7183	0,00
65_N	2098	2083	0,72
66_N	6744	6744	0,00
67_N	4968	4740	4,81
68_N	5020	4867	3,14
69_N	5400	5225	3,35
70_N	6384	6130	4,14
71_N	3384	3351	0,98
72_N	5073	4903	3,47
73_N	4618	4242	8,86
74_N	7254	7224	0,42
75_N	3959	3931	0,71
AVG	4964,68	4831,72	3,03
MAX	7906	7892	8,86
MIN	2098	2083	0,00

K = 5, N = 20

Problem Instance	Total Completion Time HEURISTIC	Total Completion Time MILP	%
76_N	5647	5641	0,11
77_N	6065	5977	1,47
78_N	8654	8650	0,05
79_N	7190	6900	4,20
80_N	6123	6122	0,02
81_N	5652	5419	4,30
82_N	6174	5982	3,21
83_N	5708	5613	1,69
84_N	9169	9160	0,10
85_N	5300	5156	2,79
86_N	6228	6228	0,00
87_N	4744	4591	3,33
88_N	3955	3955	0,00
89_N	6952	6737	3,19
90_N	8726	8516	2,47
91_N	6113	5940	2,91
92_N	5974	5950	0,40
93_N	5862	5494	6,70
94_N	7304	7304	0,00
95_N	8425	8347	0,93
96_N	6154	5912	4,09
97_N	7501	7437	0,86
98_N	9265	9226	0,42
99_N	6253	6143	1,79
100_N	8513	8417	1,14
AVG	6706,04	6592,68	1,85
MAX	9265	9226	6,70
MIN	3955	3955	0,00

Table C.2 Total Completion Time Values when *K*=10

K = 10, N = 5			
Problem Instance	Total Completion Time HEURISTIC	MILP	%
101_N	4428	4428	0,00
102_N	1583	1502	5,39
103_N	2315	2285	1,31
104_N	1779	1775	0,23
105_N	2168	2149	0,88
106_N	4317	4054	6,49
107_N	3898	3898	0,00
108_N	5808	5808	0,00
109_N	1452	1452	0,00
110_N	1968	1961	0,36
111_N	1873	1849	1,30
112_N	1516	1516	0,00
113_N	4699	4313	8,95
114_N	5975	5975	0,00
115_N	1751	1751	0,00
116_N	1392	1347	3,34
117_N	2189	2074	5,54
118_N	5760	5616	2,56
119_N	3358	3086	8,81
120_N	8111	7466	8,64
121_N	2808	2599	8,04
122_N	3633	3615	0,50
123_N	3442	3436	0,17
124_N	6754	6644	1,66
125_N	5448	5432	0,29
AVG	3537	3441,24	2,58
MAX	8111	7466	8,95
MIN	1392	1347	0,00

K = 10, N = 10			
Problem Instance	Total Completion Time HEURISTIC	MILP	%
126_N	4446	4446	0,00
127_N	9813	9813	0,00
128_N	10183	10009	1,74
129_N	10572	10572	0,00
130_N	11670	10615	9,94
131_N	5365	5275	1,71
132_N	17921	17394	3,03
133_N	10162	10162	0,00
134_N	9864	8753	12,69
135_N	10185	10185	0,00
136_N	9944	9196	8,13
137_N	15945	15750	1,24
138_N	18086	17761	1,83
139_N	17314	17128	1,09
140_N	5564	5564	0,00
141_N	10798	10798	0,00
142_N	6741	6715	0,39
143_N	15289	14832	3,08
144_N	13314	13314	0,00
145_N	10988	10988	0,00
146_N	8777	8188	7,19
147_N	9891	9839	0,53
148_N	12929	12894	0,27
149_N	7071	7071	0,00
150_N	13223	13223	0,00
AVG	11042,2	10819,4	2,11
MAX	18086	17761	12,69
MIN	4446	4446	0,00

K = 10, N = 15			
Problem Instance	Total Completion Time HEURISTIC	MILP	%
151_N	17271	17250	0,12
152_N	16574	16315	1,59
153_N	17232	17078	0,90
154_N	15232	15171	0,40
155_N	15173	15173	0,00
156_N	14309	14034	1,96
157_N	20751	20751	0,00
158_N	24756	23540	5,17
159_N	17062	16712	2,09
160_N	24694	24694	0,00
161_N	23465	21986	6,73
162_N	18967	18323	3,51
163_N	17847	17816	0,17
164_N	19370	17517	10,58
165_N	31001	30040	3,20
166_N	26723	25780	3,66
167_N	25181	25176	0,02
168_N	14260	13944	2,27
169_N	16474	15028	9,62
170_N	20231	19838	1,98
171_N	11128	11050	0,71
172_N	17432	17399	0,19
173_N	26015	25531	1,90
174_N	16404	16396	0,05
175_N	22176	21954	1,01
AVG	19589,12	19139,8	2,31
MAX	31001	30040	10,58
MIN	11128	11050	0,00

K = 10, N = 20			
Problem Instance	Total Completion Time HEURISTIC	MILP	%
176_N	24874	21850	13,84
177_N	28228	28082	0,52
178_N	26079	26011	0,26
179_N	28185	28018	0,60
180_N	21484	21339	0,68
181_N	20497	19643	4,35
182_N	26288	25300	3,91
183_N	23447	21792	7,59
184_N	24713	24713	0,00
185_N	15738	15588	0,96
186_N	29515	29290	0,77
187_N	26634	26322	1,19
188_N	27430	26501	3,51
189_N	31786	31638	0,47
190_N	32221	31142	3,46
191_N	26461	26325	0,52
192_N	24627	24580	0,19
193_N	19355	19230	0,65
194_N	34263	34082	0,53
195_N	21934	21934	0,00
196_N	20329	19943	1,94
197_N	19307	19307	0,00
198_N	33152	33152	0,00
199_N	12763	12169	4,88
200_N	29152	27527	5,90
AVG	25138,48	24619,1	2,27
MAX	34263	34082	13,84
MIN	12763	12169	0,00

Table C.3 Total Completion Time Values when *K*=15

K = 15, N = 5			
Problem	Total Completion Time		%
Instance	HEURISTIC	MILP	
201_N	8272	8272	0,00
202_N	8371	8145	2,77
203_N	6437	6407	0,47
204_N	3984	3982	0,05
205_N	2680	2560	4,69
206_N	7178	6560	9,42
207_N	15472	15292	1,18
208_N	9606	9391	2,29
209_N	6084	5832	4,32
210_N	7074	6870	2,97
211_N	15116	15116	0,00
212_N	6928	6868	0,87
213_N	8758	8380	4,51
214_N	12071	12071	0,00
215_N	14264	13594	4,93
216_N	5753	5609	2,57
217_N	3924	3852	1,87
218_N	13543	13519	0,18
219_N	12466	12450	0,13
220_N	5717	5571	2,62
221_N	6188	6057	2,16
222_N	7270	7225	0,62
223_N	8927	8817	1,25
224_N	15006	14804	1,36
225_N	12622	12459	1,31
AVG	8948,44	8788,12	2,10
MAX	15472	15292	9,42
MIN	2680	2560	0,00

K = 15, N = 10			
Problem	Total Completion Time		%
Instance	HEURISTIC	MILP	
226_N	18415	17943	2,63
227_N	22692	22692	0,00
228_N	19975	19601	1,91
229_N	15247	14832	2,80
230_N	15528	15501	0,17
231_N	14217	14217	0,00
232_N	43254	40576	6,60
233_N	25101	24593	2,07
234_N	16010	15296	4,67
235_N	23889	23304	2,51
236_N	34362	34362	0,00
237_N	22950	22578	1,65
238_N	32411	32185	0,70
239_N	33516	32760	2,31
240_N	25601	24263	5,51
241_N	21743	21743	0,00
242_N	16477	15511	6,23
243_N	36314	36314	0,00
244_N	34459	33150	3,95
245_N	22077	22077	0,00
246_N	13445	13445	0,00
247_N	24537	23438	4,69
248_N	24449	23165	5,54
249_N	27006	26854	0,57
250_N	28030	27234	2,92
AVG	24468,2	23905,4	2,30
MAX	43254	40576	6,60
MIN	13445	13445	0,00

K = 15, N = 15			
Problem	Total Completion Time		%
Instance	HEURISTIC	MILP	
251_N	30665	30345	1,05
252_N	35404	35404	0,00
253_N	27851	26724	4,22
254_N	21902	20706	5,78
255_N	33750	32763	3,01
256_N	22325	20322	9,86
257_N	34991	34495	1,44
258_N	57780	55282	4,52
259_N	35396	35288	0,31
260_N	28221	26167	7,85
261_N	44098	43216	2,04
262_N	46315	46130	0,40
263_N	38370	37402	2,59
264_N	51416	49309	4,27
265_N	52953	50935	3,96
266_N	36627	35326	3,68
267_N	30248	28716	5,34
268_N	34399	34222	0,52
269_N	57309	55927	2,47
270_N	43146	43165	-0,04
271_N	34887	32031	8,92
272_N	27501	25734	6,87
273_N	38789	37713	2,85
274_N	42966	42641	0,76
275_N	43571	43232	0,78
AVG	38035,2	36927,8	3,34
MAX	57780	55927	9,86
MIN	21902	20322	-0,04

K = 15, N = 20			
Problem	Total Completion Time		%
Instance	HEURISTIC	MILP	
276_N	42081	41727	0,85
277_N	51364	47792	7,47
278_N	40366	38133	5,86
279_N	28832	26819	7,51
280_N	28723	24669	16,43
281_N	44955	43843	2,54
282_N	78489	75951	3,34
283_N	50173	49926	0,49
284_N	37452	34324	9,11
285_N	57411	53697	6,92
286_N	69110	66060	4,62
287_N	47850	46783	2,28
288_N	63796	62933	1,37
289_N	68784	66497	3,44
290_N	52354	51456	1,75
291_N	38687	38013	1,77
292_N	42816	41493	3,19
293_N	78023	74482	4,75
294_N	61341	60875	0,77
295_N	43799	43136	1,54
296_N	39386	34690	13,54
297_N	52001	50798	2,37
298_N	54663	54674	-0,02
299_N	64391	64117	0,43
300_N	57469	57602	-0,23
AVG	51772,64	50019,6	4,08
MAX	78489	75951	16,43
MIN	28723	24669	-0,23

Table C.4 Total Completion Time Values when *K*=20

K = 20, N = 5

Problem Instance	Total Completion Time HEURISTIC	Total Completion Time MILP	%
301_N	13544	13478	0,49
302_N	15367	15163	1,35
303_N	13087	13057	0,23
304_N	7644	7434	2,82
305_N	5771	5470	5,50
306_N	12476	12148	2,70
307_N	29446	29257	0,65
308_N	14809	14728	0,55
309_N	14190	13838	2,54
310_N	13674	13379	2,20
311_N	29811	29811	0,00
312_N	13891	13741	1,09
313_N	14394	14115	1,98
314_N	22426	22426	0,00
315_N	26502	25971	2,04
316_N	12691	12547	1,15
317_N	7483	7393	1,22
318_N	21975	21975	0,00
319_N	25980	25964	0,06
320_N	8148	7972	2,21
321_N	13752	13508	1,81
322_N	14007	13945	0,44
323_N	18072	17959	0,63
324_N	28033	27949	0,30
325_N	22812	22636	0,78
AVG	16799,4	16634,6	1,31
MAX	29811	29811	5,50
MIN	5771	5470	0,00

K = 20, N = 10

Problem Instance	Total Completion Time HEURISTIC	Total Completion Time MILP	%
326_N	42487	41786	1,68
327_N	45076	45026	0,11
328_N	36298	35751	1,53
329_N	28599	28599	0,00
330_N	27435	27040	1,46
331_N	28571	28168	1,43
332_N	72188	69908	3,26
333_N	45877	44520	3,05
334_N	31656	30846	2,63
335_N	43915	43835	0,18
336_N	60222	60222	0,00
337_N	42981	42589	0,92
338_N	63805	63056	1,19
339_N	59827	59425	0,68
340_N	48224	46815	3,01
341_N	37640	35627	5,65
342_N	32567	30019	8,49
343_N	64844	63950	1,40
344_N	53082	52252	1,59
345_N	33241	33211	0,09
346_N	23408	23399	0,04
347_N	45756	44297	3,29
348_N	42178	42131	0,11
349_N	52476	51902	1,11
350_N	52684	51874	1,56
AVG	44601,48	43849,9	1,78
MAX	72188	69908	8,49
MIN	23408	23399	0,00

K = 20, N = 15

Problem Instance	Total Completion Time HEURISTIC	Total Completion Time MILP	%
351_N	64746	61931	4,55
352_N	74235	70485	5,32
353_N	43401	44541	-2,56
354_N	41625	39522	5,32
355_N	61208	60474	1,21
356_N	43160	40301	7,09
357_N	64461	63316	1,81
358_N	104424	103005	1,38
359_N	69456	67063	3,57
360_N	58618	57475	1,99
361_N	85501	85022	0,56
362_N	78603	77446	1,49
363_N	63588	62002	2,56
364_N	102492	100291	2,19
365_N	83894	83165	0,88
366_N	72874	71758	1,56
367_N	58560	58031	0,91
368_N	65779	65472	0,47
369_N	98759	100911	-2,13
370_N	80834	81328	-0,61
371_N	62455	58858	6,11
372_N	46968	46514	0,98
373_N	68821	64611	6,52
374_N	90677	91022	-0,38
375_N	79250	76835	3,14
AVG	70575,56	69255,2	2,16
MAX	104424	103005	7,09
MIN	41625	39522	-2,56

K = 20, N = 20

Problem Instance	Total Completion Time HEURISTIC	Total Completion Time MILP	%
376_N	90545	87682	3,27
377_N	89970	88434	1,74
378_N	78132	76783	1,76
379_N	56947	54371	4,74
380_N	46829	46829	0,00
381_N	78624	76508	2,77
382_N	133260	142754	-6,65
383_N	94911	94627	0,30
384_N	74568	69202	7,75
385_N	99605	96766	2,93
386_N	121154	116478	4,01
387_N	90636	88717	2,16
388_N	112493	110111	2,16
389_N	128151	124171	3,21
390_N	99286	93190	6,54
391_N	74726	73743	1,33
392_N	79584	78308	1,63
393_N	145659	140830	3,43
394_N	120718	114896	5,07
395_N	79469	78573	1,14
396_N	74196	71561	3,68
397_N	99961	98358	1,63
398_N	110921	111429	-0,46
399_N	129229	118152	9,38
400_N	115074	115632	-0,48
AVG	96985,92	94724,2	2,52
MAX	145659	142754	9,38
MIN	46829	46829	-6,65

APPENDIX D – COMPARATIVE ANALYSIS OF THE PROPOSED HEURISTIC ALGORITHM FOR THE SETUP CASE

Table D.1 The number of job sequences generated by the heuristic when *N*=5

K = 5			*K = 10*			*K = 15*			*K = 20*		
Problem Instance	Number of Job Sequences	%	Problem Instance	Number of Job Sequences	%	Problem Instance	Number of Job Sequences	%	Problem Instance	Number of Job Sequences	%
1	4	0,03	101	4	0,03	201	4	0,03	301	4	0,03
2	4	0,03	102	4	0,03	202	8	0,07	302	8	0,07
3	8	0,07	103	4	0,03	203	4	0,03	303	8	0,07
4	4	0,03	104	4	0,03	204	4	0,03	304	8	0,07
5	4	0,03	105	8	0,07	205	8	0,07	305	8	0,07
6	8	0,07	106	8	0,07	206	8	0,07	306	8	0,07
7	4	0,03	107	4	0,03	207	8	0,07	307	8	0,07
8	8	0,07	108	8	0,07	208	8	0,07	308	8	0,07
9	4	0,03	109	8	0,07	209	4	0,03	309	8	0,07
10	4	0,03	110	8	0,07	210	8	0,07	310	8	0,07
11	8	0,07	111	8	0,07	211	4	0,03	311	4	0,03
12	4	0,03	112	4	0,03	212	8	0,07	312	8	0,07
13	8	0,07	113	4	0,03	213	4	0,03	313	8	0,07
14	4	0,03	114	8	0,07	214	4	0,03	314	8	0,07
15	8	0,07	115	8	0,07	215	4	0,03	315	8	0,07
16	8	0,07	116	8	0,07	216	8	0,07	316	8	0,07
17	4	0,03	117	8	0,07	217	4	0,03	317	8	0,07
18	8	0,07	118	8	0,07	218	4	0,03	318	8	0,07
19	8	0,07	119	8	0,07	219	8	0,07	319	4	0,03
20	8	0,07	120	8	0,07	220	8	0,07	320	8	0,07
21	8	0,07	121	8	0,07	221	8	0,07	321	8	0,07
22	8	0,07	122	8	0,07	222	8	0,07	322	8	0,07
23	4	0,03	123	4	0,03	223	4	0,03	323	4	0,03
24	8	0,07	124	8	0,07	224	4	0,03	324	4	0,03
25	8	0,07	125	8	0,07	225	4	0,03	325	8	0,07
AVG	6,24	0,05	AVG	6,72	0,06	AVG	5,92	0,05	AVG	7,2	0,06

Table D.2 The number of job sequences generated by the heuristic when *N*=10

K = 5			*K = 10*			*K = 15*			*K = 20*		
Problem Instance	Number of Job Sequences	%	Problem Instance	Number of Job Sequences	%	Problem Instance	Number of Job Sequences	%	Problem Instance	Number of Job Sequences	%
26	18	4,96E-06	126	18	4,96E-06	226	18	4,96E-06	326	18	4,96E-06
27	18	4,96E-06	127	9	2,48E-06	227	9	2,48E-06	327	18	4,96E-06
28	9	2,48E-06	128	9	2,48E-06	228	18	4,96E-06	328	9	2,48E-06
29	18	4,96E-06	129	18	4,96E-06	229	18	4,96E-06	329	9	2,48E-06
30	9	2,48E-06	130	9	2,48E-06	230	18	4,96E-06	330	18	4,96E-06
31	18	4,96E-06	131	18	4,96E-06	231	18	4,96E-06	331	18	4,96E-06
32	9	2,48E-06	132	9	2,48E-06	232	9	2,48E-06	332	18	4,96E-06
33	18	4,96E-06	133	9	2,48E-06	233	18	4,96E-06	333	9	2,48E-06
34	9	2,48E-06	134	9	2,48E-06	234	9	2,48E-06	334	18	4,96E-06
35	9	2,48E-06	135	9	2,48E-06	235	18	4,96E-06	335	9	2,48E-06
36	18	4,96E-06	136	18	4,96E-06	236	9	2,48E-06	336	9	2,48E-06
37	9	2,48E-06	137	18	4,96E-06	237	18	4,96E-06	337	18	4,96E-06
38	18	4,96E-06	138	9	2,48E-06	238	18	4,96E-06	338	9	2,48E-06
39	9	2,48E-06	139	9	2,48E-06	239	9	2,48E-06	339	18	4,96E-06
40	9	2,48E-06	140	9	2,48E-06	240	9	2,48E-06	340	18	4,96E-06
41	18	4,96E-06	141	9	2,48E-06	241	18	4,96E-06	341	18	4,96E-06
42	9	2,48E-06	142	9	2,48E-06	242	18	4,96E-06	342	18	4,96E-06
43	18	4,96E-06	143	9	2,48E-06	243	9	2,48E-06	343	9	2,48E-06
44	18	4,96E-06	144	18	4,96E-06	244	18	4,96E-06	344	9	2,48E-06
45	9	2,48E-06	145	9	2,48E-06	245	18	4,96E-06	345	9	2,48E-06
46	9	2,48E-06	146	9	2,48E-06	246	18	4,96E-06	346	18	4,96E-06
47	18	4,96E-06	147	18	4,96E-06	247	9	2,48E-06	347	9	2,48E-06
48	18	4,96E-06	148	18	4,96E-06	248	18	4,96E-06	348	9	2,48E-06
49	18	4,96E-06	149	9	2,48E-06	249	9	2,48E-06	349	18	4,96E-06
50	18	4,96E-06	150	9	2,48E-06	250	18	4,96E-06	350	9	2,48E-06
AVG	14,04	3,87E-06	AVG	11,88	3,27E-06	AVG	14,76	4,07E-06	AVG	13,68	3,77E-06

Table D.3 The number of job sequences generated by the heuristic when *N*=15

K = 5			*K = 10*			*K = 15*			*K = 20*		
Problem Instance	Number of Job Sequences	%	Problem Instance	Number of Job Sequences	%	Problem Instance	Number of Job Sequences	%	Problem Instance	Number of Job Sequences	%
51	28	2,14E-11	151	14	1,07E-11	251	14	1,07E-11	351	28	2,14E-11
52	14	1,07E-11	152	28	2,14E-11	252	14	1,07E-11	352	14	1,07E-11
53	28	2,14E-11	153	28	2,14E-11	253	42	3,21E-11	353	28	2,14E-11
54	28	2,14E-11	154	28	2,14E-11	254	28	2,14E-11	354	28	2,14E-11
55	14	1,07E-11	155	28	2,14E-11	255	28	2,14E-11	355	28	2,14E-11
56	14	1,07E-11	156	14	1,07E-11	256	14	1,07E-11	356	28	2,14E-11
57	28	2,14E-11	157	14	1,07E-11	257	28	2,14E-11	357	28	2,14E-11
58	28	2,14E-11	158	28	2,14E-11	258	28	2,14E-11	358	14	1,07E-11
59	28	2,14E-11	159	14	1,07E-11	259	14	1,07E-11	359	28	2,14E-11
60	28	2,14E-11	160	28	2,14E-11	260	28	2,14E-11	360	14	1,07E-11
61	28	2,14E-11	161	28	2,14E-11	261	14	1,07E-11	361	28	2,14E-11
62	14	1,07E-11	162	14	1,07E-11	262	28	2,14E-11	362	28	2,14E-11
63	14	1,07E-11	163	28	2,14E-11	263	14	1,07E-11	363	28	2,14E-11
64	28	2,14E-11	164	28	2,14E-11	264	14	1,07E-11	364	14	1,07E-11
65	14	1,07E-11	165	14	1,07E-11	265	28	2,14E-11	365	14	1,07E-11
66	14	1,07E-11	166	14	1,07E-11	266	14	1,07E-11	366	14	1,07E-11
67	14	1,07E-11	167	14	1,07E-11	267	14	1,07E-11	367	28	2,14E-11
68	14	1,07E-11	168	14	1,07E-11	268	28	2,14E-11	368	28	2,14E-11
69	28	2,14E-11	169	14	1,07E-11	269	14	1,07E-11	369	28	2,14E-11
70	14	1,07E-11	170	14	1,07E-11	270	28	2,14E-11	370	28	2,14E-11
71	28	2,14E-11	171	14	1,07E-11	271	14	1,07E-11	371	14	1,07E-11
72	14	1,07E-11	172	14	1,07E-11	272	28	2,14E-11	372	14	1,07E-11
73	28	2,14E-11	173	14	1,07E-11	273	14	1,07E-11	373	28	2,14E-11
74	14	1,07E-11	174	28	2,14E-11	274	14	1,07E-11	374	28	2,14E-11
75	14	1,07E-11	175	14	1,07E-11	275	14	1,07E-11	375	14	1,07E-11
AVG	20,72	1,58E-11	AVG	19,6	1,50E-11	AVG	20,72	1,58E-11	AVG	22,96	1,76E-11

Table D.4 The number of job sequences generated by the heuristic when *N*=20

K = 5			*K = 10*			*K = 15*			*K = 20*		
Problem Instance	Number of Job Sequences	%	Problem Instance	Number of Job Sequences	%	Problem Instance	Number of Job Sequences	%	Problem Instance	Number of Job Sequences	%
76	38	1,56E-17	176	19	7,81E-18	276	19	7,81E-18	376	38	1,56E-17
77	38	1,56E-17	177	19	7,81E-18	277	19	7,81E-18	377	38	1,56E-17
78	38	1,56E-17	178	38	1,56E-17	278	19	7,81E-18	378	19	7,81E-18
79	38	1,56E-17	179	38	1,56E-17	279	38	1,56E-17	379	19	7,81E-18
80	38	1,56E-17	180	19	7,81E-18	280	19	7,81E-18	380	19	7,81E-18
81	38	1,56E-17	181	38	1,56E-17	281	19	7,81E-18	381	38	1,56E-17
82	38	1,56E-17	182	38	1,56E-17	282	38	1,56E-17	382	19	7,81E-18
83	19	7,81E-18	183	38	1,56E-17	283	38	1,56E-17	383	38	1,56E-17
84	38	1,56E-17	184	19	7,81E-18	284	38	1,56E-17	384	38	1,56E-17
85	38	1,56E-17	185	38	1,56E-17	285	38	1,56E-17	385	38	1,56E-17
86	38	1,56E-17	186	38	1,56E-17	286	19	7,81E-18	386	19	7,81E-18
87	19	7,81E-18	187	38	1,56E-17	287	38	1,56E-17	387	38	1,56E-17
88	38	1,56E-17	188	19	7,81E-18	288	38	1,56E-17	388	19	7,81E-18
89	19	7,81E-18	189	19	7,81E-18	289	19	7,81E-18	389	38	1,56E-17
90	38	1,56E-17	190	38	1,56E-17	290	19	7,81E-18	390	38	1,56E-17
91	19	7,81E-18	191	38	1,56E-17	291	19	7,81E-18	391	19	7,81E-18
92	38	1,56E-17	192	38	1,56E-17	292	19	7,81E-18	392	38	1,56E-17
93	19	7,81E-18	193	19	7,81E-18	293	19	7,81E-18	393	19	7,81E-18
94	19	7,81E-18	194	19	7,81E-18	294	38	1,56E-17	394	38	1,56E-17
95	38	1,56E-17	195	38	1,56E-17	295	19	7,81E-18	395	19	7,81E-18
96	19	7,81E-18	196	38	1,56E-17	296	19	7,81E-18	396	38	1,56E-17
97	19	7,81E-18	197	19	7,81E-18	297	19	7,81E-18	397	38	1,56E-17
98	38	1,56E-17	198	38	1,56E-17	298	19	7,81E-18	398	19	7,81E-18
99	19	7,81E-18	199	38	1,56E-17	299	38	1,56E-17	399	19	7,81E-18
100	38	1,56E-17	200	19	7,81E-18	300	38	1,56E-17	400	19	7,81E-18
AVG	31,16	1,28E-17	AVG	30,4	1,25E-17	AVG	26,6	1,09E-17	AVG	28,88	1,19E-17

APPENDIX E – COMPARATIVE ANALYSIS OF THE PROPOSED HEURISTIC ALGORITHM FOR THE NO-SETUP CASE

Table E.1 The number of job sequences generated by the heuristic when *N*=5

K = 5			*K = 10*			*K = 15*			*K = 20*		
Problem Instance	Number of Job Sequences	%	Problem Instance	Number of Job	%	Problem Instance	Number of Job Sequences	%	Problem Instance	Number of Job Sequences	%
1_N	8	0,07	101_N	4	0,03	201_N	4	0,03	301_N	4	0,03
2_N	4	0,03	102_N	8	0,07	202_N	8	0,07	302_N	8	0,07
3_N	8	0,07	103_N	4	0,03	203_N	4	0,03	303_N	8	0,07
4_N	4	0,03	104_N	4	0,03	204_N	4	0,03	304_N	8	0,07
5_N	4	0,03	105_N	8	0,07	205_N	8	0,07	305_N	8	0,07
6_N	8	0,07	106_N	8	0,07	206_N	8	0,07	306_N	8	0,07
7_N	4	0,03	107_N	4	0,03	207_N	8	0,07	307_N	8	0,07
8_N	8	0,07	108_N	4	0,03	208_N	4	0,03	308_N	8	0,07
9_N	4	0,03	109_N	8	0,07	209_N	8	0,07	309_N	4	0,03
10_N	4	0,03	110_N	8	0,07	210_N	8	0,07	310_N	8	0,07
11_N	8	0,07	111_N	8	0,07	211_N	4	0,03	311_N	4	0,03
12_N	4	0,03	112_N	4	0,03	212_N	8	0,07	312_N	8	0,07
13_N	8	0,07	113_N	4	0,03	213_N	8	0,07	313_N	8	0,07
14_N	4	0,03	114_N	8	0,07	214_N	4	0,03	314_N	8	0,07
15_N	8	0,07	115_N	8	0,07	215_N	8	0,07	315_N	8	0,07
16_N	8	0,07	116_N	8	0,07	216_N	4	0,03	316_N	4	0,03
17_N	4	0,03	117_N	4	0,03	217_N	8	0,07	317_N	8	0,07
18_N	8	0,07	118_N	8	0,07	218_N	4	0,03	318_N	8	0,07
19_N	8	0,07	119_N	4	0,03	219_N	8	0,07	319_N	4	0,03
20_N	4	0,03	120_N	4	0,03	220_N	8	0,07	320_N	8	0,07
21_N	8	0,07	121_N	12	0,10	221_N	8	0,07	321_N	8	0,07
22_N	4	0,03	122_N	8	0,07	222_N	8	0,07	322_N	8	0,07
23_N	8	0,07	123_N	4	0,03	223_N	4	0,03	323_N	4	0,03
24_N	4	0,03	124_N	8	0,07	224_N	4	0,03	324_N	4	0,03
25_N	8	0,07	125_N	8	0,07	225_N	4	0,03	325_N	8	0,07
AVG	6,08	0,05	AVG	6,4	0,05	AVG	6,24	0,05	AVG	6,88	0,06

Table E.2 The number of job sequences generated by the heuristic when *N*=10

K = 5			*K = 10*			*K = 15*			*K = 20*		
Problem Instance	Number of Job Sequences	%	Problem Instance	Number of Job Sequences	%	Problem Instance	Number of Job Sequences	%	Problem Instance	Number of Job Sequences	%
26_N	18	4,96E-06	126_N	18	4,96E-06	226_N	18	4,96E-06	326_N	9	2,48E-06
27_N	9	2,48E-06	127_N	9	2,48E-06	227_N	9	2,48E-06	327_N	18	4,96E-06
28_N	9	2,48E-06	128_N	9	2,48E-06	228_N	18	4,96E-06	328_N	9	2,48E-06
29_N	18	4,96E-06	129_N	18	4,96E-06	229_N	9	2,48E-06	329_N	9	2,48E-06
30_N	9	2,48E-06	130_N	9	2,48E-06	230_N	18	4,96E-06	330_N	18	4,96E-06
31_N	9	2,48E-06	131_N	18	4,96E-06	231_N	18	4,96E-06	331_N	18	4,96E-06
32_N	18	4,96E-06	132_N	9	2,48E-06	232_N	9	2,48E-06	332_N	18	4,96E-06
33_N	18	4,96E-06	133_N	9	2,48E-06	233_N	9	2,48E-06	333_N	9	2,48E-06
34_N	18	4,96E-06	134_N	9	2,48E-06	234_N	9	2,48E-06	334_N	18	4,96E-06
35_N	9	2,48E-06	135_N	18	4,96E-06	235_N	18	4,96E-06	335_N	9	2,48E-06
36_N	18	4,96E-06	136_N	18	4,96E-06	236_N	9	2,48E-06	336_N	9	2,48E-06
37_N	9	2,48E-06	137_N	9	2,48E-06	237_N	9	2,48E-06	337_N	18	4,96E-06
38_N	18	4,96E-06	138_N	9	2,48E-06	238_N	9	2,48E-06	338_N	9	2,48E-06
39_N	9	2,48E-06	139_N	9	2,48E-06	239_N	9	2,48E-06	339_N	18	4,96E-06
40_N	9	2,48E-06	140_N	9	2,48E-06	240_N	9	2,48E-06	340_N	18	4,96E-06
41_N	9	2,48E-06	141_N	9	2,48E-06	241_N	9	2,48E-06	341_N	18	4,96E-06
42_N	9	2,48E-06	142_N	9	2,48E-06	242_N	18	4,96E-06	342_N	18	4,96E-06
43_N	9	2,48E-06	143_N	18	4,96E-06	243_N	18	4,96E-06	343_N	9	2,48E-06
44_N	18	4,96E-06	144_N	18	4,96E-06	244_N	18	4,96E-06	344_N	9	2,48E-06
45_N	9	2,48E-06	145_N	9	2,48E-06	245_N	9	2,48E-06	345_N	9	2,48E-06
46_N	9	2,48E-06	146_N	9	2,48E-06	246_N	18	4,96E-06	346_N	18	4,96E-06
47_N	18	4,96E-06	147_N	18	4,96E-06	247_N	9	2,48E-06	347_N	9	2,48E-06
48_N	9	2,48E-06	148_N	9	2,48E-06	248_N	9	2,48E-06	348_N	9	2,48E-06
49_N	18	4,96E-06	149_N	9	2,48E-06	249_N	18	4,96E-06	349_N	18	4,96E-06
50_N	18	4,96E-06	150_N	9	2,48E-06	250_N	18	4,96E-06	350_N	9	2,48E-06
AVG	12,96	3,57E-06	AVG	11,88	3,27E-06	AVG	12,96	3,57E-06	AVG	13,32	3,67E-06

Table E.3 The number of job sequences generated by the heuristic when *N*=15

K = 5			*K = 10*			*K = 15*			*K = 20*		
Problem Instance	Number of Job Sequences	%	Problem Instance	Number of Job Sequences	%	Problem Instance	Number of Job Sequences	%	Problem Instance	Number of Job Sequences	%
51_N	14	1,07E-11	151_N	14	1,07E-11	251_N	14	1,07E-11	351_N	28	2,14E-11
52_N	28	2,14E-11	152_N	28	2,14E-11	252_N	14	1,07E-11	352_N	14	1,07E-11
53_N	28	2,14E-11	153_N	28	2,14E-11	253_N	14	1,07E-11	353_N	28	2,14E-11
54_N	14	1,07E-11	154_N	28	2,14E-11	254_N	14	1,07E-11	354_N	14	1,07E-11
55_N	28	2,14E-11	155_N	28	2,14E-11	255_N	28	2,14E-11	355_N	28	2,14E-11
56_N	14	1,07E-11	156_N	28	2,14E-11	256_N	14	1,07E-11	356_N	14	1,07E-11
57_N	28	2,14E-11	157_N	14	1,07E-11	257_N	28	2,14E-11	357_N	14	1,07E-11
58_N	28	2,14E-11	158_N	28	2,14E-11	258_N	14	1,07E-11	358_N	14	1,07E-11
59_N	28	2,14E-11	159_N	14	1,07E-11	259_N	28	2,14E-11	359_N	28	2,14E-11
60_N	14	1,07E-11	160_N	14	1,07E-11	260_N	28	2,14E-11	360_N	14	1,07E-11
61_N	28	2,14E-11	161_N	28	2,14E-11	261_N	28	2,14E-11	361_N	28	2,14E-11
62_N	14	1,07E-11	162_N	14	1,07E-11	262_N	28	2,14E-11	362_N	28	2,14E-11
63_N	14	1,07E-11	163_N	28	2,14E-11	263_N	14	1,07E-11	363_N	28	2,14E-11
64_N	14	1,07E-11	164_N	28	2,14E-11	264_N	14	1,07E-11	364_N	14	1,07E-11
65_N	28	2,14E-11	165_N	14	1,07E-11	265_N	14	1,07E-11	365_N	14	1,07E-11
66_N	14	1,07E-11	166_N	14	1,07E-11	266_N	14	1,07E-11	366_N	14	1,07E-11
67_N	14	1,07E-11	167_N	14	1,07E-11	267_N	14	1,07E-11	367_N	28	2,14E-11
68_N	14	1,07E-11	168_N	14	1,07E-11	268_N	28	2,14E-11	368_N	28	2,14E-11
69_N	28	2,14E-11	169_N	28	2,14E-11	269_N	14	1,07E-11	369_N	14	1,07E-11
70_N	14	1,07E-11	170_N	14	1,07E-11	270_N	28	2,14E-11	370_N	28	2,14E-11
71_N	14	1,07E-11	171_N	14	1,07E-11	271_N	14	1,07E-11	371_N	14	1,07E-11
72_N	28	2,14E-11	172_N	14	1,07E-11	272_N	28	2,14E-11	372_N	28	2,14E-11
73_N	28	2,14E-11	173_N	14	1,07E-11	273_N	14	1,07E-11	373_N	14	1,07E-11
74_N	28	2,14E-11	174_N	28	2,14E-11	274_N	14	1,07E-11	374_N	14	1,07E-11
75_N	14	1,07E-11	175_N	14	1,07E-11	275_N	14	1,07E-11	375_N	14	1,07E-11
AVG	20,72	1,58E-11	AVG	20,16	1,54E-11	AVG	19,04	1,46E-11	AVG	20,16	1,54E-11

Table E.4 The number of job sequences generated by the heuristic when *N*=20

K = 5			*K = 10*			*K = 15*			*K = 20*		
Problem Instance	Number of Job Sequences	%	Problem Instance	Number of Job Sequences	%	Problem Instance	Number of Job Sequences	%	Problem Instance	Number of Job Sequences	%
76_N	19	7,81E-18	176_N	38	1,56E-17	276_N	19	7,81E-18	376_N	38	1,56E-17
77_N	38	1,56E-17	177_N	38	1,56E-17	277_N	19	7,81E-18	377_N	38	1,56E-17
78_N	19	7,81E-18	178_N	38	1,56E-17	278_N	19	7,81E-18	378_N	19	7,81E-18
79_N	38	1,56E-17	179_N	38	1,56E-17	279_N	19	7,81E-18	379_N	19	7,81E-18
80_N	38	1,56E-17	180_N	19	7,81E-18	280_N	19	7,81E-18	380_N	19	7,81E-18
81_N	38	1,56E-17	181_N	38	1,56E-17	281_N	19	7,81E-18	381_N	38	1,56E-17
82_N	38	1,56E-17	182_N	38	1,56E-17	282_N	19	7,81E-18	382_N	19	7,81E-18
83_N	19	7,81E-18	183_N	38	1,56E-17	283_N	38	1,56E-17	383_N	38	1,56E-17
84_N	38	1,56E-17	184_N	19	7,81E-18	284_N	38	1,56E-17	384_N	19	7,81E-18
85_N	19	7,81E-18	185_N	19	7,81E-18	285_N	38	1,56E-17	385_N	38	1,56E-17
86_N	38	1,56E-17	186_N	38	1,56E-17	286_N	19	7,81E-18	386_N	19	7,81E-18
87_N	19	7,81E-18	187_N	19	7,81E-18	287_N	19	7,81E-18	387_N	38	1,56E-17
88_N	38	1,56E-17	188_N	19	7,81E-18	288_N	19	7,81E-18	388_N	19	7,81E-18
89_N	19	7,81E-18	189_N	19	7,81E-18	289_N	19	7,81E-18	389_N	38	1,56E-17
90_N	19	7,81E-18	190_N	19	7,81E-18	290_N	19	7,81E-18	390_N	19	7,81E-18
91_N	19	7,81E-18	191_N	38	1,56E-17	291_N	19	7,81E-18	391_N	19	7,81E-18
92_N	38	1,56E-17	192_N	38	1,56E-17	292_N	19	7,81E-18	392_N	38	1,56E-17
93_N	19	7,81E-18	193_N	19	7,81E-18	293_N	19	7,81E-18	393_N	38	1,56E-17
94_N	19	7,81E-18	194_N	19	7,81E-18	294_N	38	1,56E-17	394_N	38	1,56E-17
95_N	38	1,56E-17	195_N	38	1,56E-17	295_N	19	7,81E-18	395_N	19	7,81E-18
96_N	19	7,81E-18	196_N	38	1,56E-17	296_N	19	7,81E-18	396_N	19	7,81E-18
97_N	38	1,56E-17	197_N	19	7,81E-18	297_N	19	7,81E-18	397_N	38	1,56E-17
98_N	38	1,56E-17	198_N	38	1,56E-17	298_N	19	7,81E-18	398_N	19	7,81E-18
99_N	19	7,81E-18	199_N	38	1,56E-17	299_N	38	1,56E-17	399_N	38	1,56E-17
100_N	38	1,56E-17	200_N	19	7,81E-18	300_N	38	1,56E-17	400_N	19	7,81E-18
AVG	28,88	1,19E-17	AVG	29,64	1,22E-17	AVG	23,56	9,68E-18	AVG	28,12	1,16E-17

APPENDIX F – ANALYSES OF SOLUTION IMPROVEMENT IN PHASE-2 FOR THE SETUP CASE

Table F.1 Objective Function Improvement in Phase-2 (Tabu Search) when *K*=5

K=5, *N*=5			
Problem Instance	Phase-1	Tabu	Objective Improvement (%)
1	653	653	0,00
2	1780	1780	0,00
3	1806	1760	2,55
4	1907	1907	0,00
5	2812	2812	0,00
6	1565	1434	8,37
7	2892	2892	0,00
8	1368	1338	2,19
9	3946	3946	0,00
10	1573	1573	0,00
11	3352	3198	4,59
12	1809	1809	0,00
13	1764	1596	9,52
14	2579	2579	0,00
15	4539	4497	0,93
16	2846	2833	0,46
17	1174	1174	0,00
18	3055	2866	6,19
19	3619	3531	2,43
20	1031	986	4,36
21	3435	3354	2,36
22	2027	2005	1,09
23	3163	3163	0,00
24	3739	3700	1,04
25	2011	1958	2,64
AVG			1,95

K=5, *N*=10			
Problem Instance	Phase-1	Tabu	Objective Improvement (%)
26	4705	4683	0,47
27	5603	5507	1,71
28	3554	3554	0,00
29	4795	4615	3,75
30	4179	4179	0,00
31	5861	5712	2,54
32	5031	5031	0,00
33	3007	2988	0,63
34	6701	6701	0,00
35	6179	6179	0,00
36	5336	5300	0,67
37	3631	3631	0,00
38	6870	6690	2,62
39	4008	4008	0,00
40	3300	3300	0,00
41	3615	3526	2,46
42	4919	4919	0,00
43	6008	5881	2,11
44	3728	3684	1,18
45	4287	4287	0,00
46	2653	2653	0,00
47	6106	6097	0,15
48	5308	5049	4,88
49	6933	6853	1,15
50	5422	5414	0,15
AVG			0,98

K=5, *N*=15			
Problem Instance	Phase-1	Tabu	Objective Improvement (%)
51	5713	5645	1,19
52	7474	7474	0,00
53	4669	4642	0,58
54	9021	8906	1,27
55	6987	6987	0,00
56	8495	8495	0,00
57	6800	6794	0,09
58	8716	8587	1,48
59	7850	7706	1,83
60	13730	13501	1,67
61	9817	9661	1,59
62	8354	8354	0,00
63	6239	6239	0,00
64	9130	9028	1,12
65	6871	6871	0,00
66	14240	14240	0,00
67	6226	6226	0,00
68	8099	8099	0,00
69	8391	8337	0,64
70	11536	11536	0,00
71	4810	4758	1,08
72	6241	6241	0,00
73	10022	9858	1,64
74	12660	12660	0,00
75	5454	5454	0,00
AVG			0,57

K=5, N=20			
Problem Instance	Phase-1	Tabu	Objective Improvement (%)
76	7464	7449	0,20
77	11014	10941	0,66
78	14195	14177	0,13
79	13199	13068	0,99
80	15493	15257	1,52
81	11029	10878	1,37
82	14022	13998	0,17
83	7514	7514	0,00
84	11520	11456	0,56
85	10198	10128	0,69
86	11889	11765	1,04
87	8366	8366	0,00
88	7723	7651	0,93
89	8805	8805	0,00
90	10860	10850	0,09
91	7997	7997	0,00
92	10079	10016	0,63
93	8146	8146	0,00
94	9607	9607	0,00
95	10837	10822	0,14
96	7794	7794	0,00
97	13550	13550	0,00
98	18127	18115	0,07
99	10855	10855	0,00
100	10865	10484	3,51
AVG			0,51

Table F.2 Objective Function Improvement in Phase-2 (Tabu Search) when *K*=10

K=10, *N*=5			
Problem Instance	Phase-1	Tabu	Objective Improvement (%)
101	5002	5002	0,00
102	3201	3201	0,00
103	3069	3069	0,00
104	2929	2929	0,00
105	4351	3157	27,44
106	7378	7031	4,70
107	5350	5350	0,00
108	11854	11701	1,29
109	2209	2163	2,08
110	3259	3063	6,01
111	4274	4060	5,01
112	2476	2476	0,00
113	5756	5756	0,00
114	7266	6728	7,40
115	4798	4666	2,75
116	2487	2194	11,78
117	5076	5063	0,26
118	8826	8405	4,77
119	5189	5050	2,68
120	10304	9494	7,86
121	3757	3563	5,16
122	4437	4419	0,41
123	4728	4728	0,00
124	9242	8269	10,53
125	8701	8543	1,82
AVG			4,08

K=10, *N*=10			
Problem Instance	Phase-1	Tabu	Objective Improvement (%)
126	5881	5857	0,41
127	15574	15574	0,00
128	13850	13850	0,00
129	14457	13749	4,90
130	13353	13353	0,00
131	7409	7317	1,24
132	19765	19765	0,00
133	12468	12468	0,00
134	13643	13643	0,00
135	17220	17220	0,00
136	11474	11310	1,43
137	17901	17797	0,58
138	21184	21184	0,00
139	22871	22871	0,00
140	12380	12380	0,00
141	13571	13571	0,00
142	8840	8840	0,00
143	25375	25375	0,00
144	17186	16480	4,11
145	17372	17372	0,00
146	12590	12590	0,00
147	19055	18381	3,54
148	20736	20725	0,05
149	10669	10669	0,00
150	15182	15182	0,00
AVG			0,65

K=10, *N*=15			
Problem Instance	Phase-1	Tabu	Objective Improvement (%)
151	21102	21102	0,00
152	25164	24131	4,11
153	19544	19291	1,29
154	22382	21935	2,00
155	18432	18206	1,23
156	22111	22111	0,00
157	29343	29343	0,00
158	36597	36339	0,70
159	19355	19355	0,00
160	29339	29330	0,03
161	30826	30704	0,40
162	24555	24555	0,00
163	29088	28514	1,97
164	22277	22045	1,04
165	40240	40240	0,00
166	35315	35315	0,00
167	28213	28213	0,00
168	17524	17524	0,00
169	20241	20241	0,00
170	26240	26240	0,00
171	21904	21904	0,00
172	20817	20817	0,00
173	32861	32861	0,00
174	18961	18913	0,25
175	37731	37731	0,00
AVG			0,52

K=10, *N*=20			
Problem Instance	Phase-1	Tabu	Objective Improvement
176	32683	32683	0,00
177	45725	45725	0,00
178	38837	38426	1,06
179	33648	32709	2,79
180	38009	38009	0,00
181	32887	32848	0,12
182	34962	34268	1,99
183	37398	36233	3,12
184	37863	37863	0,00
185	22175	22141	0,15
186	40204	40072	0,33
187	35773	35726	0,13
188	40543	40543	0,00
189	40329	40329	0,00
190	49259	49245	0,03
191	30491	30112	1,24
192	32738	32399	1,04
193	30748	30748	0,00
194	51566	51566	0,00
195	26110	26032	0,30
196	40668	40118	1,35
197	23791	23791	0,00
198	44350	44191	0,36
199	21198	21019	0,84
200	33682	33682	0,00
AVG			0,59

Table F.3 Objective Function Improvement in Phase-2 (Tabu Search) when *K*=15

K=15, *N*=5			
Problem Instance	Phase-1	Tabu	Objective Improvement (%)
201	9032	9032	0,00
202	10642	10092	5,17
203	7897	7897	0,00
204	8100	8100	0,00
205	9727	9016	7,31
206	8871	8631	2,71
207	17128	16792	1,96
208	14935	14728	1,39
209	10795	10795	0,00
210	8951	8319	7,06
211	16513	16513	0,00
212	8949	8188	8,50
213	11507	11507	0,00
214	15094	15094	0,00
215	20760	20760	0,00
216	9101	8992	1,20
217	5385	5385	0,00
218	19169	19169	0,00
219	15992	15193	5,00
220	11136	9168	17,67
221	11480	10425	9,19
222	11541	10011	13,26
223	14301	14301	0,00
224	22268	22268	0,00
225	13637	13637	0,00
AVG			3,22

K=15, *N*=10			
Problem Instance	Phase-1	Tabu	Objective Improvement (%)
226	20702	20312	1,88
227	31861	31861	0,00
228	25557	25249	1,21
229	21665	21442	1,03
230	24723	24298	1,72
231	17702	17517	1,05
232	47009	47009	0,00
233	32958	32246	2,16
234	25015	25015	0,00
235	29587	29301	0,97
236	36780	36780	0,00
237	25834	25703	0,51
238	39279	38854	1,08
239	41008	41008	0,00
240	35465	35465	0,00
241	26794	26463	1,24
242	20665	20226	2,12
243	50105	50105	0,00
244	40666	39918	1,84
245	30876	30734	0,46
246	21961	21591	1,68
247	35315	35315	0,00
248	34641	34529	0,32
249	37871	37871	0,00
250	30819	30672	0,48
AVG			0,79

K=15, *N*=15			
Problem Instance	Phase-1	Tabu	Objective Improvement
251	36063	36063	0,00
252	46021	46021	0,00
253	37562	37403	0,42
254	35402	35149	0,71
255	50491	49996	0,98
256	35872	35872	0,00
257	40290	39895	0,98
258	69107	68526	0,84
259	49618	49618	0,00
260	38765	38687	0,20
261	53844	53844	0,00
262	50894	50265	1,24
263	44343	44343	0,00
264	61127	61127	0,00
265	66587	66086	0,75
266	52527	52527	0,00
267	37220	37220	0,00
268	44668	43991	1,52
269	74889	74889	0,00
270	56309	54415	3,36
271	49413	49413	0,00
272	39009	38276	1,88
273	55332	55332	0,00
274	61458	61458	0,00
275	48045	48045	0,00
AVG			0,52

K=15, N=20			
Problem Instance	Phase-1	Tabu	Objective Improvement (%)
276	48353	48353	0,00
277	65876	65876	0,00
278	51720	51720	0,00
279	46854	46698	0,33
280	47972	47972	0,00
281	51716	51716	0,00
282	89606	88109	1,67
283	69634	69086	0,79
284	53555	53477	0,15
285	64336	64112	0,35
286	74841	74841	0,00
287	59375	59174	0,34
288	76014	75966	0,06
289	85177	85177	0,00
290	71141	71141	0,00
291	48890	48890	0,00
292	54544	54544	0,00
293	103059	103059	0,00
294	76264	75601	0,87
295	62208	62208	0,00
296	56002	56002	0,00
297	71476	71476	0,00
298	78655	78655	0,00
299	82163	81601	0,68
300	69927	69810	0,17
AVG			0,22

Table F.4 Objective Function Improvement in Phase-2 (Tabu Search) when *K*=20

K=20, *N*=5				*K*=20, *N*=10				*K*=20, *N*=15				K=20, N=20			
Problem Instance	Phase-1	Tabu	Objective Improvement (%)	Problem Instance	Phase-1	Tabu	Objective Improvement (%)	Problem Instance	Phase-1	Tabu	Objective Improvement (%)	Problem Instance	Phase-1	Tabu	Objective Improvement (%)
301	14592	14592	0,00	326	45655	45505	0,33	351	70568	70292	0,39	376	100055	99349	0,71
302	18166	18124	0,23	327	59933	57103	4,72	352	88190	88190	0,00	377	110013	108955	0,96
303	16699	15123	9,44	328	43362	43362	0,00	353	54635	54369	0,49	378	93625	93625	0,00
304	13782	13071	5,16	329	37616	37616	0,00	354	61527	61200	0,53	379	83128	83128	0,00
305	15608	14384	7,84	330	40863	40356	1,24	355	85029	84809	0,26	380	79499	79499	0,00
306	15914	14670	7,82	331	33663	33645	0,05	356	68525	63364	7,53	381	89618	87626	2,22
307	32068	31381	2,14	332	79885	76916	3,72	357	71008	70573	0,61	382	148070	148070	0,00
308	21630	21539	0,42	333	55961	55961	0,00	358	117458	117458	0,00	383	122335	121428	0,74
309	21346	21333	0,06	334	43905	43813	0,21	359	88439	86818	1,83	384	96904	93766	3,24
310	16325	15337	6,05	335	51366	51366	0,00	360	72483	72483	0,00	385	111662	111563	0,09
311	31723	31723	0,00	336	63910	63910	0,00	361	96693	94092	2,69	386	128868	128868	0,00
312	17618	15626	11,31	337	47191	47004	0,40	362	84244	83853	0,46	387	100661	100658	0,00
313	18359	18117	1,32	338	72653	72653	0,00	363	72065	71893	0,24	388	129241	129241	0,00
314	26212	26046	0,63	339	70555	70104	0,64	364	115843	115843	0,00	389	150745	150364	0,25
315	36987	34394	7,01	340	63270	63192	0,12	365	101816	101816	0,00	390	128438	126898	1,20
316	17464	17385	0,45	341	44750	44541	0,47	366	90747	90747	0,00	391	88219	88219	0,00
317	9510	9472	0,40	342	39032	37742	3,30	367	67651	67434	0,32	392	95945	95808	0,14
318	32570	28941	11,14	343	82326	82326	0,00	368	79479	79050	0,54	393	179506	179506	0,00
319	29743	29743	0,00	344	62040	62040	0,00	369	125000	123868	0,91	394	140677	140425	0,18
320	15500	13195	14,87	345	45041	45041	0,00	370	97375	96003	1,41	395	104990	104990	0,00
321	21961	19988	8,98	346	34697	34337	1,04	371	82706	82706	0,00	396	96783	96531	0,26
322	20866	17656	15,38	347	61241	61241	0,00	372	61539	61539	0,00	397	127873	126770	0,86
323	25814	25814	0,00	348	59050	59050	0,00	373	92094	88416	3,99	398	148261	148261	0,00
324	36001	36001	0,00	349	65718	65534	0,28	374	118598	117279	1,11	399	150070	150070	0,00
325	25145	24824	1,28	350	56334	56334	0,00	375	85356	85356	0,00	400	134845	134845	0,00
AVG			4,48	AVG			0,66	AVG			0,93	AVG			0,43

APPENDIX G – ANALYSES OF SOLUTION IMPROVEMENT IN PHASE-2 FOR THE NO-SETUP CASE

Table G.1 Objective Function Improvement in Phase-2 (Tabu Search) when *K*=5

K=5, *N*=5

Problem Instance	Phase-1	Tabu	Objective Improvement (%)
1_N	165	159	3,64
2_N	1142	1142	0,00
3_N	1250	1232	1,44
4_N	615	615	0,00
5_N	603	603	0,00
6_N	1038	912	12,14
7_N	2361	2361	0,00
8_N	647	617	4,64
9_N	1835	1835	0,00
10_N	1146	1146	0,00
11_N	2837	2683	5,43
12_N	1334	1334	0,00
13_N	944	776	17,80
14_N	1584	1584	0,00
15_N	2637	2595	1,59
16_N	1591	1489	6,41
17_N	649	649	0,00
18_N	1442	1230	14,70
19_N	2629	2608	0,80
20_N	325	325	0,00
21_N	1764	1683	4,59
22_N	1107	1107	0,00
23_N	1777	1740	2,08
24_N	1947	1947	0,00
25_N	1673	1555	7,05
AVG			3,29

K=5, *N*=10

Problem Instance	Phase-1	Tabu	Objective Improvement (%)
26_N	3823	3787	0,94
27_N	4315	4315	0,00
28_N	2118	2118	0,00
29_N	2646	2546	3,78
30_N	1979	1979	0,00
31_N	3483	3483	0,00
32_N	2994	2981	0,43
33_N	1826	1792	1,86
34_N	3362	3321	1,22
35_N	3690	3690	0,00
36_N	3523	3470	1,50
37_N	2602	2602	0,00
38_N	5705	5525	3,16
39_N	2068	2068	0,00
40_N	2191	2191	0,00
41_N	2573	2573	0,00
42_N	3013	3013	0,00
43_N	2645	2645	0,00
44_N	1367	1323	3,22
45_N	1622	1622	0,00
46_N	1878	1878	0,00
47_N	1855	1846	0,49
48_N	2265	2265	0,00
49_N	3316	3236	2,41
50_N	4396	4388	0,18
AVG			0,77

K=5, *N*=15

Problem Instance	Phase-1	Tabu	Objective Improvement (%)
51_N	3971	3971	0,00
52_N	6032	6031	0,02
53_N	2357	2276	3,44
54_N	6255	6255	0,00
55_N	2711	2622	3,28
56_N	6828	6828	0,00
57_N	4533	4489	0,97
58_N	5656	5461	3,45
59_N	4401	4339	1,41
60_N	7906	7906	0,00
61_N	5778	5622	2,70
62_N	3776	3776	0,00
63_N	2456	2456	0,00
64_N	7183	7183	0,00
65_N	2118	2098	0,94
66_N	6744	6744	0,00
67_N	4968	4968	0,00
68_N	5020	5020	0,00
69_N	5431	5400	0,57
70_N	6384	6384	0,00
71_N	3384	3384	0,00
72_N	5139	5073	1,28
73_N	4782	4618	3,43
74_N	7263	7254	0,12
75_N	3959	3959	0,00
AVG			0,86

K=5, N=20

Problem Instance	Phase-1	Tabu	Objective Improvement (%)
76_N	5647	5647	0,00
77_N	6076	6065	0,18
78_N	8654	8654	0,00
79_N	7199	7190	0,13
80_N	6156	6123	0,54
81_N	5801	5652	2,57
82_N	6202	6174	0,45
83_N	5708	5708	0,00
84_N	9233	9169	0,69
85_N	5300	5300	0,00
86_N	6352	6228	1,95
87_N	4744	4744	0,00
88_N	4032	3955	1,91
89_N	6952	6952	0,00
90_N	8726	8726	0,00
91_N	6113	6113	0,00
92_N	6022	5974	0,80
93_N	5862	5862	0,00
94_N	7304	7304	0,00
95_N	8440	8425	0,18
96_N	6154	6154	0,00
97_N	7589	7501	1,16
98_N	9277	9265	0,13
99_N	6253	6253	0,00
100_N	8835	8513	3,64
AVG			0,57

Table G.2 Objective Function Improvement in Phase-2 (Tabu Search) when *K*=10

K=10, *N*=5			
Problem Instance	Phase-1	Tabu	Objective Improvement (%)
101_N	4428	4428	0,00
102_N	1592	1583	0,57
103_N	2315	2315	0,00
104_N	1779	1779	0,00
105_N	3308	2168	34,46
106_N	4708	4317	8,31
107_N	3898	3898	0,00
108_N	5808	5808	0,00
109_N	1518	1452	4,35
110_N	2145	1968	8,25
111_N	1901	1873	1,47
112_N	1516	1516	0,00
113_N	4699	4699	0,00
114_N	6513	5975	8,26
115_N	1863	1751	6,01
116_N	1500	1392	7,20
117_N	2189	2189	0,00
118_N	5838	5760	1,34
119_N	3358	3358	0,00
120_N	8111	8111	0,00
121_N	3020	2808	7,02
122_N	3651	3633	0,49
123_N	3442	3442	0,00
124_N	7671	6754	11,95
125_N	5606	5448	2,82
AVG			4,10

K=10, *N*=10			
Problem Instance	Phase-1	Tabu	Objective Improvement (%)
126_N	4470	4446	0,54
127_N	9813	9813	0,00
128_N	10183	10183	0,00
129_N	11099	10572	4,75
130_N	11670	11670	0,00
131_N	5428	5365	1,16
132_N	17921	17921	0,00
133_N	10162	10162	0,00
134_N	9864	9864	0,00
135_N	10319	10185	1,30
136_N	10108	9944	1,62
137_N	15945	15945	0,00
138_N	18086	18086	0,00
139_N	17314	17314	0,00
140_N	5564	5564	0,00
141_N	10798	10798	0,00
142_N	6741	6741	0,00
143_N	15692	15289	2,57
144_N	13936	13314	4,46
145_N	10988	10988	0,00
146_N	8777	8777	0,00
147_N	10462	9891	5,46
148_N	12929	12929	0,00
149_N	7071	7071	0,00
150_N	13223	13223	0,00
AVG			0,87

K=10, *N*=15			
Problem Instance	Phase-1	Tabu	Objective Improvement (%)
151_N	17271	17271	0,00
152_N	17115	16574	3,16
153_N	17486	17232	1,45
154_N	15529	15232	1,91
155_N	15356	15173	1,19
156_N	14315	14309	0,04
157_N	20751	20751	0,00
158_N	24840	24756	0,34
159_N	17062	17062	0,00
160_N	24694	24694	0,00
161_N	23549	23465	0,36
162_N	18967	18967	0,00
163_N	18392	17847	2,96
164_N	19626	19370	1,30
165_N	31001	31001	0,00
166_N	26723	26723	0,00
167_N	25181	25181	0,00
168_N	14260	14260	0,00
169_N	16696	16474	1,33
170_N	20231	20231	0,00
171_N	11128	11128	0,00
172_N	17432	17432	0,00
173_N	26015	26015	0,00
174_N	16484	16404	0,49
175_N	22176	22176	0,00
AVG			0,58

K=10, *N*=20			
Problem Instance	Phase-1	Tabu	Objective Improvement (%)
176_N	24878	24874	0,02
177_N	28387	28228	0,56
178_N	26474	26079	1,49
179_N	29015	28185	2,86
180_N	21484	21484	0,00
181_N	20610	20497	0,55
182_N	26829	26288	2,02
183_N	23483	23447	0,15
184_N	24713	24713	0,00
185_N	15738	15738	0,00
186_N	29573	29515	0,20
187_N	26634	26634	0,00
188_N	27430	27430	0,00
189_N	31786	31786	0,00
190_N	32221	32221	0,00
191_N	26798	26461	1,26
192_N	24871	24627	0,98
193_N	19355	19355	0,00
194_N	34263	34263	0,00
195_N	22012	21934	0,35
196_N	20943	20329	2,93
197_N	19307	19307	0,00
198_N	33311	33152	0,48
199_N	12837	12763	0,58
200_N	29152	29152	0,00
AVG			0,58

Table G.3 Objective Function Improvement in Phase-2 (Tabu Search) when *K*=15

K=15, *N*=5			
Problem Instance	Phase-1	Tabu	Objective Improvement (%)
201_N	8272	8272	0,00
202_N	8739	8371	4,21
203_N	6437	6437	0,00
204_N	3984	3984	0,00
205_N	3167	2680	15,38
206_N	7250	7178	0,99
207_N	15808	15472	2,13
208_N	9606	9606	0,00
209_N	6299	6084	3,41
210_N	7752	7074	8,75
211_N	15116	15116	0,00
212_N	7685	6928	9,85
213_N	8769	8758	0,13
214_N	12071	12071	0,00
215_N	15598	14264	8,55
216_N	5753	5753	0,00
217_N	3990	3924	1,65
218_N	13543	13543	0,00
219_N	13096	12466	4,81
220_N	7855	5717	27,22
221_N	6922	6188	10,60
222_N	9276	7270	21,63
223_N	8927	8927	0,00
224_N	15006	15006	0,00
225_N	12622	12622	0,00
AVG			4,77

K=15, *N*=10			
Problem Instance	Phase-1	Tabu	Objective Improvement (%)
226_N	18769	18415	1,89
227_N	22692	22692	0,00
228_N	20188	19975	1,06
229_N	15247	15247	0,00
230_N	15927	15528	2,51
231_N	14327	14217	0,77
232_N	43254	43254	0,00
233_N	25101	25101	0,00
234_N	16010	16010	0,00
235_N	24138	23889	1,03
236_N	34362	34362	0,00
237_N	22950	22950	0,00
238_N	32411	32411	0,00
239_N	33516	33516	0,00
240_N	25601	25601	0,00
241_N	21743	21743	0,00
242_N	16565	16477	0,53
243_N	36834	36314	1,41
244_N	34635	34459	0,51
245_N	22077	22077	0,00
246_N	13833	13445	2,80
247_N	24537	24537	0,00
248_N	24449	24449	0,00
249_N	27051	27006	0,17
250_N	28177	28030	0,52
AVG			0,53

K=15, *N*=15			
Problem Instance	Phase-1	Tabu	Objective Improvement (%)
251_N	30665	30665	0,00
252_N	35404	35404	0,00
253_N	27851	27851	0,00
254_N	21902	21902	0,00
255_N	33806	33750	0,17
256_N	22325	22325	0,00
257_N	35345	34991	1,00
258_N	57780	57780	0,00
259_N	35512	35396	0,33
260_N	28251	28221	0,11
261_N	45138	44098	2,30
262_N	46819	46315	1,08
263_N	38370	38370	0,00
264_N	51416	51416	0,00
265_N	52953	52953	0,00
266_N	36627	36627	0,00
267_N	30248	30248	0,00
268_N	35001	34399	1,72
269_N	57309	57309	0,00
270_N	44820	43146	3,73
271_N	34887	34887	0,00
272_N	27830	27501	1,18
273_N	38789	38789	0,00
274_N	42966	42966	0,00
275_N	43571	43571	0,00
AVG			0,46

K=15, N=20			
Problem Instance	Phase-1	Tabu	Objective Improvement (%)
276_N	42081	42081	0,00
277_N	51364	51364	0,00
278_N	40366	40366	0,00
279_N	28832	28832	0,00
280_N	28723	28723	0,00
281_N	44955	44955	0,00
282_N	78489	78489	0,00
283_N	50827	50173	1,29
284_N	37482	37452	0,08
285_N	57465	57411	0,09
286_N	69110	69110	0,00
287_N	47850	47850	0,00
288_N	63796	63796	0,00
289_N	68784	68784	0,00
290_N	52354	52354	0,00
291_N	38687	38687	0,00
292_N	42816	42816	0,00
293_N	78023	78023	0,00
294_N	62004	61341	1,07
295_N	43799	43799	0,00
296_N	39386	39386	0,00
297_N	52001	52001	0,00
298_N	54663	54663	0,00
299_N	64478	64391	0,13
300_N	57617	57469	0,26
AVG			0,12

Table G.4 Objective Function Improvement in Phase-2 (Tabu Search) when *K*=20

K=20, *N*=5			
Problem Instance	Phase-1	Tabu	Objective Improvement (%)
301_N	13544	13544	0,00
302_N	15696	15367	2,10
303_N	14517	13087	9,85
304_N	8321	7644	8,14
305_N	7086	5771	18,56
306_N	12720	12476	1,92
307_N	30133	29446	2,28
308_N	15156	14809	2,29
309_N	14190	14190	0,00
310_N	14695	13674	6,95
311_N	29811	29811	0,00
312_N	15879	13891	12,52
313_N	14562	14394	1,15
314_N	22662	22426	1,04
315_N	29772	26502	10,98
316_N	12691	12691	0,00
317_N	7521	7483	0,51
318_N	25740	21975	14,63
319_N	25980	25980	0,00
320_N	10726	8148	24,04
321_N	15701	13752	12,41
322_N	17745	14007	21,07
323_N	18072	18072	0,00
324_N	28033	28033	0,00
325_N	22971	22812	0,69
AVG			6,04

K=20, *N*=10			
Problem Instance	Phase-1	Tabu	Objective Improvement (%)
326_N	42487	42487	0,00
327_N	47746	45076	5,59
328_N	36298	36298	0,00
329_N	28599	28599	0,00
330_N	27678	27435	0,88
331_N	28619	28571	0,17
332_N	75632	72188	4,55
333_N	45877	45877	0,00
334_N	31658	31656	0,01
335_N	43915	43915	0,00
336_N	60222	60222	0,00
337_N	43168	42981	0,43
338_N	63805	63805	0,00
339_N	60083	59827	0,43
340_N	48285	48224	0,13
341_N	37719	37640	0,21
342_N	33958	32567	4,10
343_N	64844	64844	0,00
344_N	53082	53082	0,00
345_N	33241	33241	0,00
346_N	23702	23408	1,24
347_N	45756	45756	0,00
348_N	42178	42178	0,00
349_N	52660	52476	0,35
350_N	52684	52684	0,00
AVG			0,72

K=20, *N*=15			
Problem Instance	Phase-1	Tabu	Objective Improvement (%)
351_N	65001	64746	0,39
352_N	74235	74235	0,00
353_N	43661	43401	0,60
354_N	41625	41625	0,00
355_N	61277	61208	0,11
356_N	43160	43160	0,00
357_N	64461	64461	0,00
358_N	104424	104424	0,00
359_N	71144	69456	2,37
360_N	58618	58618	0,00
361_N	86824	85501	1,52
362_N	78988	78603	0,49
363_N	63662	63588	0,12
364_N	102492	102492	0,00
365_N	83894	83894	0,00
366_N	72874	72874	0,00
367_N	58777	58560	0,37
368_N	66027	65779	0,38
369_N	98759	98759	0,00
370_N	82157	80834	1,61
371_N	62455	62455	0,00
372_N	47861	46968	1,87
373_N	68821	68821	0,00
374_N	90677	90677	0,00
375_N	79250	79250	0,00
AVG			0,39

K=20, N=20			
Problem Instance	Phase-1	Tabu	Objective Improvement (%)
376_N	91251	90545	0,77
377_N	91060	89970	1,20
378_N	78132	78132	0,00
379_N	56947	56947	0,00
380_N	46829	46829	0,00
381_N	79970	78624	1,68
382_N	133260	133260	0,00
383_N	95079	94911	0,18
384_N	74568	74568	0,00
385_N	99699	99605	0,09
386_N	121154	121154	0,00
387_N	90639	90636	0,00
388_N	112493	112493	0,00
389_N	128446	128151	0,23
390_N	99286	99286	0,00
391_N	74726	74726	0,00
392_N	79629	79584	0,06
393_N	146245	145659	0,40
394_N	120970	120718	0,21
395_N	79469	79469	0,00
396_N	74196	74196	0,00
397_N	100833	99961	0,86
398_N	110921	110921	0,00
399_N	130089	129229	0,66
400_N	115074	115074	0,00
AVG			0,25