SCHEDULING OPTIMIZATION OF MANUFACTURING SYSTEMS WITH NO-WAIT CONSTRAINTS

BY

HAMED SAMARGHANDI

A thesis submitted to the Faculty of Graduate Studies of

The University of Manitoba

in partial fulfillment of the requirements of the degree of

DOCTOR OF PHILOSOPHY

Department of Mechanical and Manufacturing Engineering

University of Manitoba

Winnipeg

Abstract

No-wait scheduling problem refers to the set of problems in which a number of jobs are available for processing on a number of machines with the added constraint that there should be no waiting time between consecutive operations of the jobs. It is well-known that most of the no-wait scheduling problems are strongly NP-hard. Moreover, no-wait scheduling problems have numerous real-life applications. This thesis studies a wide range of no-wait scheduling problems, along with side constraints that make such problems more applicable. First, 2-machine no-wait flow shop problem is studied. Afterwards, setup times and single server constraints are added to this problem in order to make it more applicable. Then, job shop version of this problem is further researched. Analytical results for both of these problems are presented; moreover, efficient algorithms are developed and applied to large instances of these problems.

Afterward, general no-wait flow shop problem (NWFS) is the focus of the thesis. First, the NWFS is studied; mathematical models as well as metaheuristics are developed for NWFS. Then, setup times are added to NWFS in order to make the problem more applicable. Finally, the case of sequence dependent setup times is further researched. Efficient algorithms are developed for both problems.

Finally, no-wait job shop (NWJS) problem is studied. Literature has proposed different methods to solve NWJS; the most successful approaches decompose the problem into a timetabling sub-problem and a sequencing sub-problem. Different sequencing and timetabling algorithms are developed to solve NWJS.

This thesis provides insight to several no-wait scheduling problems. A number of theorems are discussed and proved in order to find the optimum solution of no-wait problems with special characteristics. For the problems without such characteristics, mathematical models are developed. Metaheuristics are utilized to deal with large-instances of NP-hard problems. Computational results show that the developed methods in this thesis are very effective and efficient compared to the competitive methods available in the literature.

Acknowledgement

Foremost, I would like to express my sincere gratitude to my advisor Dr. Tarek ElMekkawy for the continuous support of my PhD study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my PhD study.

Besides my advisor, I would like to thank the rest of my thesis committee: Dr. Yunhua Luo, and Dr. Shaahin Filizadeh for their encouragement, insightful comments, and hard questions.

My sincere thanks also goes to University of Manitoba, Natural Sciences and Engineering Research Council of Canada (NSERC), and CancerCare Manitoba (CCMB), for providing me with financial support opportunities and leading me to working on diverse exciting projects.

I would like to take this opportunity to thank Ms. Sue Bates of CancerCare Manitoba for enlightening me with various great ideas. She has been the utmost director and leader.

Last but not the least, I would like to thank my family: my parents, Mohammad Samarghandi and Masoumeh Mahmoudi, for giving birth to me at the first place and supporting me spiritually throughout my life; and my beloved wife, Azin Rahimi, for her continuous sympathy and zealous support.

Dedication

This thesis is dedicated to my father, who taught me that brilliant result is earned by meticulous work.

It is also dedicated to my mother, who offered me unconditional love and support from the day I was born.

I dedicate this thesis to my wife, who made significant sacrifices throughout the course of this thesis.

I, furthermore, dedicate this thesis to my brother and my in-laws, who have presented encouragement that was necessary to have this research done.

Live long, happy, healthy and prosperous. May God be with you.

Table of Contents

List of Tables

List of Figures

Nomenclature

n	Number of jobs
m	Number of machines
J_i	Job i
s_i, t_i	Setup time of J_i on the first and second machine respectively
a_i, b_i	Processing time of J_i on the first and second machine respectively
ST_{s_i} , ST_{t_i}	Starting time of s_i and t_i
ST_{ij}	Starting time of j th operation of the i th job
Π	Set of all permutations of n jobs
π	A permutation in Π
$C_{\max}$	Makespan
O_{ij}	j th operation of the i th job
p_{ij}	Processing time of the j th operation of the job $C_i - S_i = \sum_{j=1}^{m} p_{ij}$ on its respective machine
S_i	Starting time of the J_i
$S_{o_{ij}}$	Starting time of o_{ij}
C_i	Completion time of J_i
ST_{ijk}	Setup time of o_{ij} if scheduled after o_{kj}

ST_{ij0}	Setup time of o_{ij} if J_i is the first job to be scheduled
S	Source of the graph
T	Final node in the graph
u	An arc
$b(u)$	Beginning node of u
$e(u)$	End node of u
$w(u)$	Label of u
$s(i)$	Successor of node i according to the sequence of operations
$r(i)$	Predecessor of node i according to the sequence of operations
$d(i)$	Successor of node i according to disjunction
$v(i)$	Predecessor of node i according to disjunction

Table of Abbreviations

ACO	Ant Colony Optimization
AD (SS, 2-Opt)	Average Deviation between the objective function of the solutions proposed by the 2-opt algorithm and the proposed algorithm when Single Server constraints are considered
AD (SS, LB)	Average Deviation between the problem with Single Server constraints and LB
AICA	Adapted Imperialist Competitive Algorithm
AM	Adaptive Memory
ANOVA	Analysis Of Variance
APAM+SS	Average Percentage of the time Added to the Makespan of the problem with setup times when Single Server constraints are considered
ARE	Average Relative Error
ATSP	Asymmetrical Traveling Salesman Problem
Av	Average
BRE	Best Relative Error
CPM	Critical Path Method
DHS	Discrete Harmony Search
DOE	Design Of Experiments
GA	Genetic Algorithm
GASA	Genetic Algorithm and Simulated Annealing
GHz	Giga Hertz
LB	Lower Bound
MC	Matrix Coding
MCA	Makespan Calculation Algorithm
ND	Non-Delay algorithm
NLA	Node Labeling Algorithm
NP	Non-deterministic Polynomial time
NWFS	No-Wait Flow Shop
NWJS	No-Wait Job Shop
OFV	Objective Function Value
Oper. Time	Operating Time
OR	Operations Research
P	Polynomial time
PBSA	Population Based Simulated Annealing
PRD	Percentage Relative Deviation
PSO	Particle Swarm Optimization
RAM	Random Access Memory
RS	Right Shift algorithm
RSR	Right Shift + Reverse algorithm
SA	Simulated Annealing
SD	Sequence Dependent
Seq	Sequencing
STD	Standard Deviation
STM	Short Term Memory
TS	Tabu Search

TSP	Travelling Salesperson Problem
TT	Time Tabling
VNS	Variable Neighborhood Search
WRE	Worst Relative Error

Copyright Notices

1. With kind permission from Inderscience Publishers:
    - Samarghandi, H. and ElMekkawy, T.Y. (2011) An efficient hybrid algorithm for the two-machine no-wait flow shop problem with separable setup times and single server. European Journal of Industrial Engineering, 5(2): p. 111-131.

2. With kind permission from Taylor & Francis Group:
    - Samarghandi, H. and ElMekkawy, T.Y. A meta-heuristic approach for solving the no-wait flow shop problem. International Journal of Production Research. DOI: 10.1080/00207543.2011.648277

3. With kind permission from Springer Science + Business Media:
    - Samarghandi, H. and ElMekkawy, T.Y. (2012) A genetic algorithm and particle swarm optimization for no-wait flow shop problem with separable setup times and makespan criterion The International Journal of Advanced Manufacturing Technology, 61: p. 1101 - 1114

    - Samarghandi, H. and ElMekkawy, T.Y. On the two-machine no-wait job shop problem with separable setup times and single server constraints. International Journal of Advanced Manufacturing Technology. DOI: 10.1007/s00170-012-4169-1

Chapter 1

Introduction

1.1 Background

In today's competitive business environment, sequencing and scheduling play an imperative role in both manufacturing and service industries; organizations must meet certain due date commitments to their customers since failure to meet such deadlines will cause them lose revenue or heavy penalties; in addition, companies need to schedule their resources efficiently to remain viable relative to their competitors [1]. Different organizations consider different articles as resources. For instance, in a manufacturing environment, machines or personnel are considered as resources; for airlines, runways are considered as resources; physicians or beds are important resources in healthcare context.

Although many different objectives can be defined for scheduling, in general, the most desired optimization criterion is to minimize the cost of production or to maximize the efficiency and revenue [1]. Scheduling can have a major impact on the productivity. As a matter of fact, scheduling process defines who should perform what activities at what time. In other words, scheduling is the process of performing the right tasks at the right time by the right people. An effective scheduling scheme brings competitive advantages to organizations, including:

- Efficient inventory control;
- Increased productivity;
- Accurate delivery dates;
- Minimization of the change in processes;

- Balance in necessary labor loads.

Although scheduling techniques can be applied to continuous processes such as those in refineries, in this research, the main focus is on discrete scheduling in which the units of production are distinguishable and distinct from each other. In such systems, the following machine environments are amongst the most famous:

- Single machine: which is the simplest known environment; and a special case of all the other environments.
- Identical parallel machines: in this environment, several identical machines exist. A job needs one of these several machines for processing. Different assumptions can cause more complicated parallel machine environments such as different speed parallel machines or unrelated parallel machines. In these environments, some of the parallel machines have different specifications.
- Flow shop: there are m machines in the system. All jobs need to be processed by all machines and they follow the same order. Several flow shop environments exist based on the assumption. For instance, in a flexible flow shop, identical (or non-identical) parallel machines exist in at least one of the processing stages.
- Job shop: is a generalization of flow shop. In fact, job shop environment is a flow shop in which the assumption that all the jobs have to follow the same route is relaxed. Similar to flow shop, different job shop environments such as flexible job shops can be defined.
- Open shop: an open shop has m machines to process the jobs. However, there is no restriction about the order of sending the jobs to different machines.

This research is mainly focused on flow shop and job shop machine environments. Moreover, at this point it is beneficial to review different classes of schedules. The following definitions are from [1].

- Non-delay schedule: is a feasible schedule in which no machine is kept idle while an operation is waiting for processing.
- Non-preemptive schedule: are schedules in which all the operations should finish completely once their processing is started. In other words, preemption of the operations is not allowed.
- Active schedule: "a feasible non-preemptive schedule is called active if it is not possible to construct another schedule, by changing the order of processing on the machines, with at least one operation finishing earlier and no operation finishing later."
- Semi-active schedule: "a feasible non-preemptive schedule is called semi-active if no operation can be completed earlier without changing the order of processing on any one of the machines."

Figure 1-1 demonstrates the relation between the above classes of the schedules and the optimum solution.

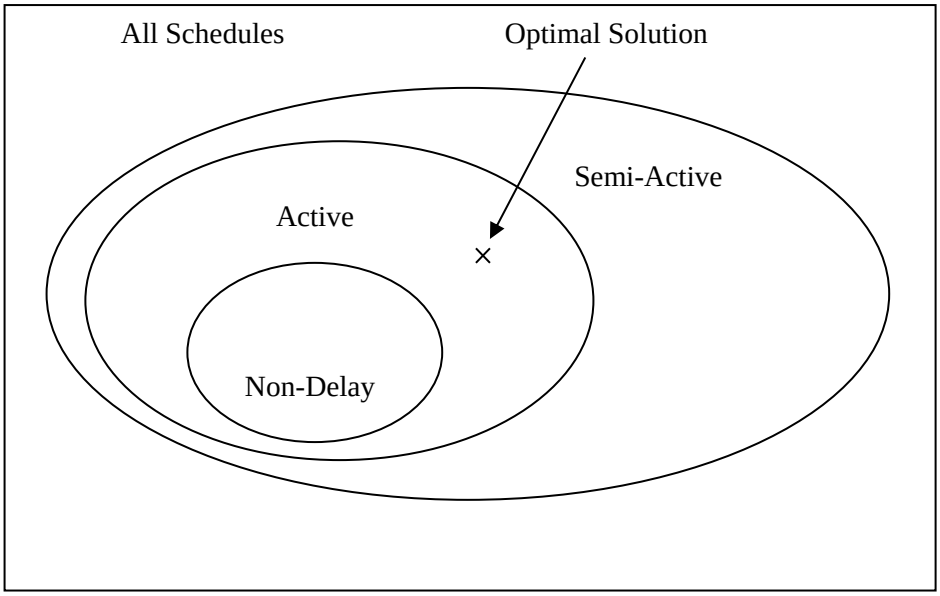

Figure 1-1 Classes of non-preemptive schedules for job shop problems

Different performance criteria can be defined. The most famous and applicable performance measures are makespan and total flow time. Makespan is defined as the maximum completion time of the scheduled jobs. Total flow time is the summation of the time that each job spends in the system.

1.2 Problem Statement

No-wait scheduling problems consist of the set of the problems, in which a number of jobs are given for scheduling on a number of machines with the added constraint that there should be no waiting time between successive operations of the same job. Accordingly, once processing of a job is started, no interruption is permitted between the operations of that job. No-wait scheduling problems include problems in different context such as open shop, flow shop, job shop, flexible shops, etc. In this thesis, the main focus is on the no-wait flow shop and job shop problems.

No-wait flow shop problem is a special case of the classic flow shop problem, in which there should be no waiting time between successive operations of the same job. In flow shop context, all the jobs follow the same order in the system. Figure 1-2 demonstrates a sample no-wait flow shop. In the job shop context, each job is allowed to have its own specific route in the system. Figure 1-3 demonstrates a no-wait job shop system. The considered performance measure throughout this thesis is makespan which is denoted as $C_{\max}$.

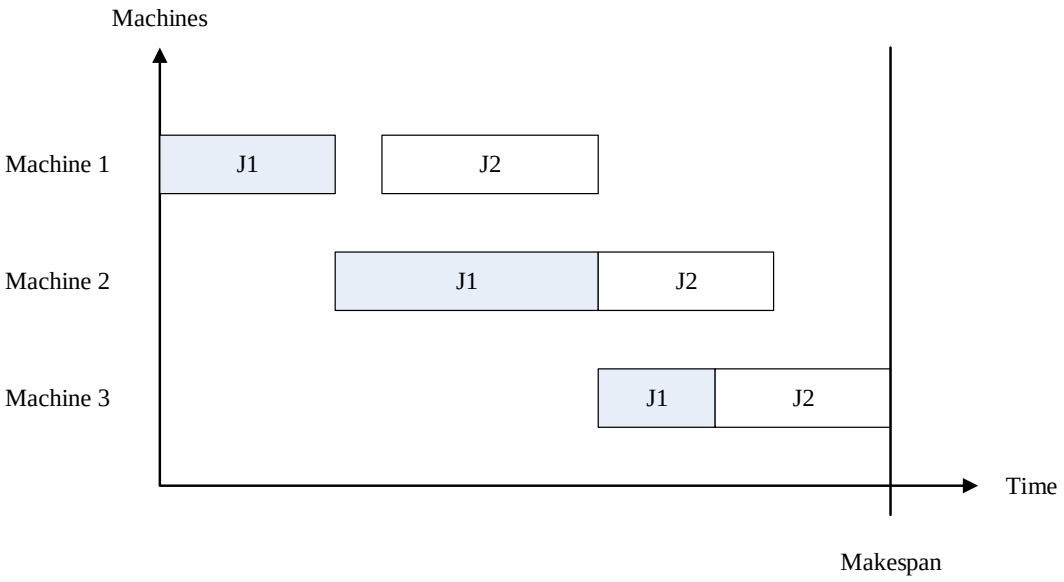

Figure 1-2 Example of a No-Wait Flow Shop

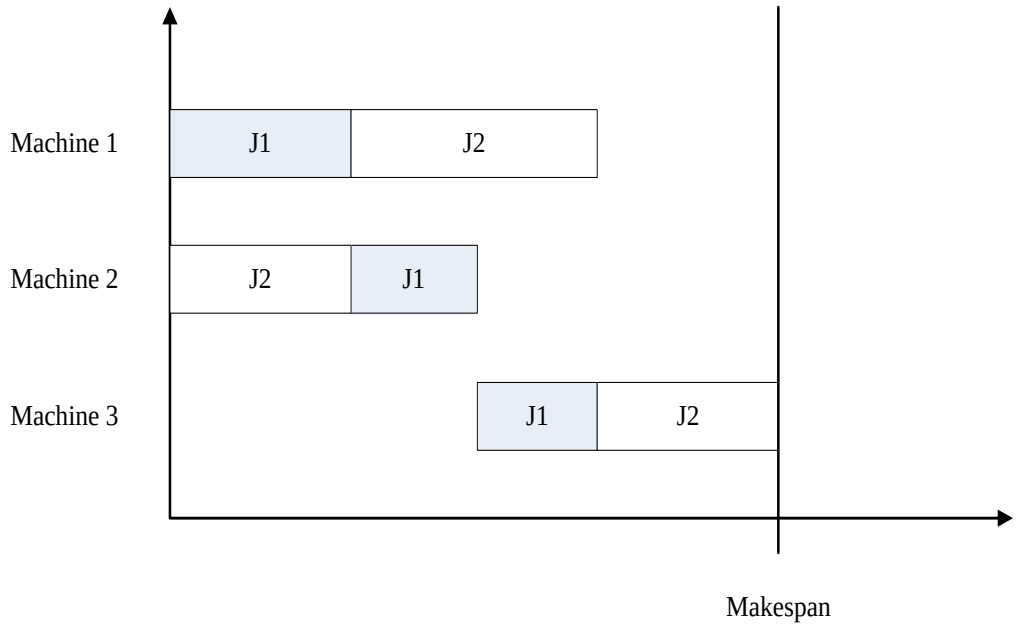

Figure 1-3 Example of a No-Wait Job Shop

This research considers a variety of applicable side constraints and assumptions in addition to the generic no-wait flow shop and job shop problems. For instance, chapters 3 and 5 consider the situation in which separable setup times are combined with no-wait flow shop and job shop problems. In these chapters, a separable setup time on the machine is considered necessary in order to prepare the machine to process an operation. Separable setup time is a special case of setup time that allows the operators to perform the setup on the machine before the operation is ready to be processed.

Chapter 3 and 5 also consider the case of sequence dependent setup times. In such systems, setup times not only depend on the operations to which the setup belongs, but also on the preceding operation scheduled on the machine. All of the mentioned problems are NP-hard to the strong sense. This concept is discussed further in the following section.

1.3 Research Motivation

No-wait scheduling problems are categorized as combinatorial optimization problems. Combinatorial optimization problems are the problems whose feasible region is countable [2]. As it will be discussed in the following sections, all the studied problems in this research are NP-Hard. Moreover, no-wait flow shop problems are transformable to Travelling Salesperson Problem (TSP) [3], which is one of the most famous NP-Hard problems. No-wait job shop problems are a generalization of their flow shop versions and therefore, are NP-Hard as well. As a result of NP-Hardness, finding optimal solutions of large instances of these problems is not possible in a reasonable time. Figure 1-4 depicts the complexity of different classes of problems in combinatorial optimization [2]. As a result, the problem is interesting for further research in order to develop efficient algorithms that generate good-quality solutions (compared to the best solution) in a reasonable time.

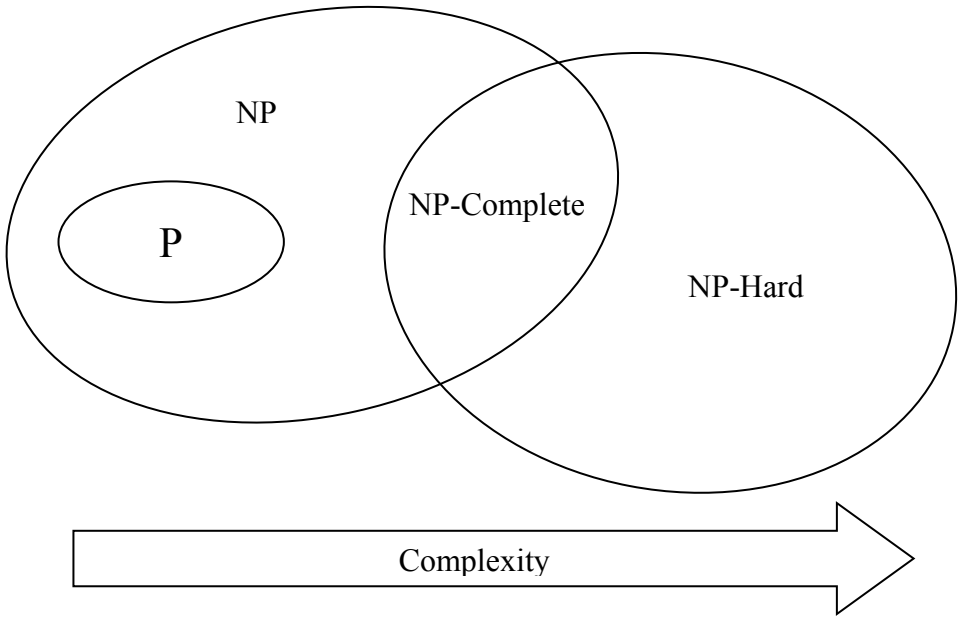

Figure 1-4 Decision Making Problems and Complexity

In addition, no-wait scheduling problems have a large number of real-world applications. As a general rule, whenever the next operation of a job is required to be scheduled within a determined time frame after the previous operations, a no-wait scheduling problem occurs. For example, one can name chemical industries [4], food industries [5], steel production [6], pharmaceutical industries [7], and production of concrete products [8]. For a more comprehensive review of the applications of the problem, the reader is referred to [5]. In other words, no-wait scheduling problems are also interesting for researchers because of their numerous industrial applications.

On the other hand, literature review shows that the number of researches on no-wait scheduling problems is not comparable to the general scheduling problems. There are not enough algorithms developed to deal with no-wait scheduling problems. Also, available algorithms do not deal with many of the applicable assumptions and constraints; examples of such applicable problems include no-wait scheduling problems with sequence independent or dependent setup times. Current thesis studies a number of these situations. Many more will be introduced in chapter 7 as promising research directions for future works.

1.4 Research objectives

Section 1.3 discussed the importance of conducting further research on no-wait scheduling problems from both practical and theoretical aspects. No-wait scheduling problems have numerous practical applications with less literature compared to other scheduling problems. Moreover, most of the no-wait scheduling problems are NP-Hard. Therefore, solving practical large instances of these problems to optimality is not reasonable by means of the current computational technology. Sections 3.5 and 5.4 discuss that solving 2-machine no-wait flow shop and job shop problems with more than 14 and 12 jobs would take more than 3 hours respectively on a modern computer.

Consequently, developing algorithms that generate good-quality (as opposed to the best) solutions to large-scale no-wait scheduling problems in reasonable time is necessary. This research first discusses various no-wait scheduling problems from an analytical point of view. After developing a deep analytical knowledge on each problem, further applicable constraints and assumptions are discussed. Subsequently, efficient and effective metaheuristic algorithms are proposed to deal with the discussed practical problems. Efficiency and effectiveness of the proposed algorithms are verified by applying them to a large number of test problems from the literature. The proposed algorithms prove to be competitive compared to the available methods in the literature. The developed algorithms find several new best-known solutions for the problems listed in the literature.

1.5 Thesis Outline

The rest of the thesis is organized as shown in Figure 1-5. Chapter 2 reviews the available literature on the relevant no-wait scheduling problems. This chapter also lists the points of strength and weaknesses of each method available in the literature. Chapter 3 is devoted to 2-machine no-wait flow shop problem. Efficient algorithms for this problem are proposed after discussing some analytical findings on these problems.

Chapter 4 discusses the findings of solving the general no-wait flow shop problem. This chapter first formulates this problem as a permutation. Afterwards, metaheuristics are developed to deal with this problem. Chapter 4 also studies the general no-wait flow shop problem with separable setup times. After developing a good knowledge on the setup times, this chapter introduces the case of sequence dependent setup times.

Chapter 5 performs a study on 2-machine job shop problem with setup times and single server constraints. Chapter 6 present the findings of this research on the general

no-wait job shop problem. This chapter decomposes the no-wait job shop problem into sequencing and timetabling problems. Then several combinations of different timetabling and sequencing algorithms are tested on a large number of test problems from the literature. Finally, the best combination of the algorithms to solve the no-wait job shop problem is found by following a design of experiments approach. Chapter 7 presents the conclusions and directions for future research efforts.

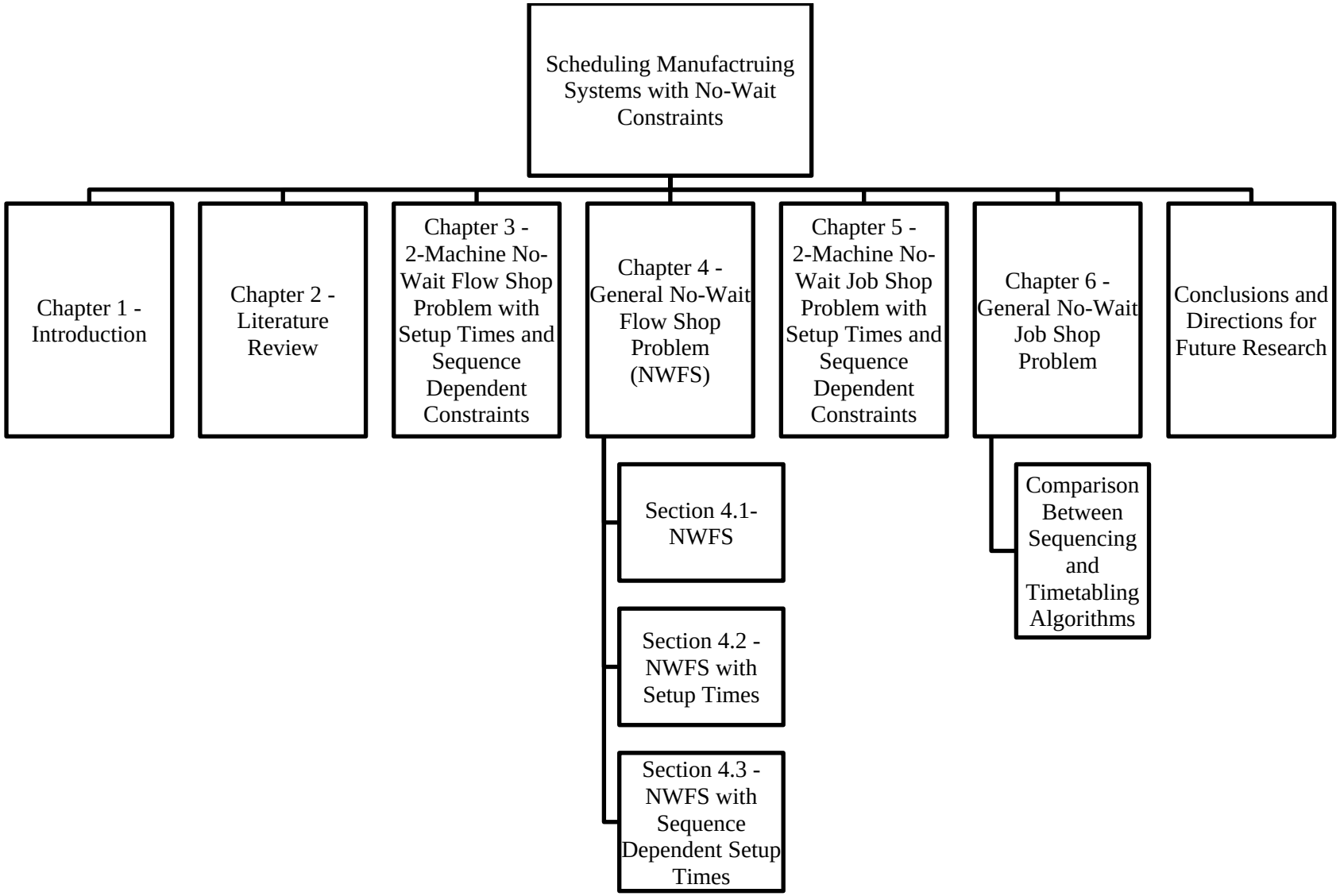

Figure 1-5 Thesis Outline

Chapter 2

Literature Review

2.1 Background

While there are several studies available in the literature that consider the general no-wait flow shop problem, there are few researches that considered the no-wait problems with setup times. To the best of the author's knowledge, the first researches about no-wait scheduling should be credited to Reddi and Ramamoorthy [9], Wismer [6], Grabowski and Syslo [10], Bonney and Gundry [11], King and Spachis [12], Gangadharan and Rajendran [13], Rajendran [4], Glass et al. [14], Sidney et al. [15], and Sviridenko [16]. More recent researches in this field will be reviewed in the following sections.

2.2 No-Wait Flow Shop Problem

Gupta et al. [3] studied the two-machine flow shop problem with setup and removal times and reduced the problem to the famous Travelling Salesperson Problem (TSP). Aldowaisan and Allahverdi [17] considered $F2|no-wait,setup|\sum C_i$. They developed an elimination criterion to obtain optimal solutions for two special cases and a heuristic algorithm for the generic case. Aldowaisan [18] studied this problem further and proposed a local and a global dominance relationship and thereby developed a new heuristic and a branch and bound algorithm.

Macchiaroli et al. [19] discussed the industrial applications of the scheduling problems with no-wait constraints. Their work, however, does not address any method for solving such problems. Cheng et al. [20] studied the two-machine flow shop problem

with separable setup times and single server constraints ($F2,S1|setup|C_{\max}$) and proposed some heuristics for the problem.

Bianco et al. [21] proposed two heuristics for the no-wait flow shop problem with release dates and sequence dependent setup times, and makespan criterion. Sidney et al. [15] considered the two-machine no-wait flow shop problem with anticipatory setup times and makespan; they proposed a heuristic to deal with this problem.

Bertolissi [22] developed a heuristic algorithm for $F|no-wait|\sum C_i$. The results of this algorithm was compared to the results of [23], [11], and [24]. Cheng et al. [25] provided a review of flow shop scheduling research involving setup times.

Shyu et al. [26] developed an Ant Colony Optimization (ACO) to deal with no-wait flow shop scheduling, when minimizing the total completion time is the objective function. Computational results of this algorithm is compared to the results of [17] and [18].

Pranzo [27] considered the two-machine batch scheduling flow shop problem with sequence independent setup times and removal times. This research studied a number of special cases of the problem and reduced these special cases to TSP.

Aldowaisan and Allahverdi [28] proposed six heuristics for $F|no-wait|C_{\max}$ and compared them with the heuristics of [24, 29]. Furthermore, they considered the separable setup time in the problem of $F|no-wait|\sum C_i$; an elimination criterion was also developed to obtain optimal solutions for two special cases and a heuristic algorithm for the generic case.

Grabowski and Pempera [30] proposed 6 new meta-heuristics for $F|no-wait|C_{\max}$ and compared them with the algorithm of [4]. Results of the proposed algorithms of this research are compared to the results of this research. Guirchoun et al. [31] studied a two-stage hybrid flow shop with no-wait constraint between the two stages and proposed a heuristic to solve this problem optimally. A hybrid genetic algorithm developed in Franc et al. [32] to solve the no-wait flow shop problem with sequence dependent setup times and ready times.

Liu et al. [33] proposed a particle swarm optimization with several local searches for $F|no-wait|C_{\max}$ and compared their results with [4]. Su and Lee [34] considered the problem of two-machine no-wait flow shop with separable setup times and single server ($F2|no-wait,setup|\sum C_i$). They developed a heuristic and a branch-and-bound algorithm to solve the problem. They tested their algorithm on a number of randomly generated test problems. Li et al. [35] considered the no-wait flow shop problem with makespan criterion and developed a genetic algorithm to find good-quality solutions for the problems they had generated.

Pan et al. [36] developed a discrete Particle Swarm Optimization (PSO) for the no-wait flow shop problem with both makespan and total flow time criteria. This study further hybridizes the PSO with variable neighborhood descent algorithm. The neighborhood structures in the algorithms of this study are exchange and insert. Pan et al. [37] develops a hybrid discrete PSO to solve no-wait flow shop problem with makespan criterion. Pan et al. [38] develops an iterated greedy heuristic for no-wait flow shop problem with makespan criterion. Pan et al. [39] developed a discrete differential evolution algorithm for the problem of no-wait flow shop with makespan and maximum tardiness criteria. This algorithm follows the same computational approach of [36].

Tavakkoli-Moghaddam et al. [40] developed an immune algorithm for the no-wait flow shop problem with both weighted mean completion time and weighted mean tardiness objectives. Laha and Chakraborty [41] proposed a constructive algorithm for $F|no-wait|C_{\max}$. Huang et al. [42] studied a no-wait two-stage multiprocessor flow shop with setup times and total completion time as the performance measure; they proposed an Ant Colony Optimization for the problem.

Ruiz and Allahverdi [43] developed a number of heuristics to deal with the no-wait flow shop problem when a linear combination of makespan and maximum lateness is considered as the objective function. Huang et al. [44] develop an orthogonal ACO for no-wait flow shop problem. Framinan et al. [45] considers the no-wait flow shop problem with the objective of minimizing the total completion time. This study proposes several heuristics, aimed to generate solutions in short time.

Jarboui et al. [46] develop a hybrid genetic algorithm to minimize the makespan and the total flow time in the no-wait flow shop scheduling problem. The proposed GA is further hybridized with VNS algorithm to improve the solution quality.

Gao et al. [47] develop a discrete harmony search algorithm (DHS) for solving the no-wait flow shop scheduling problem with the objective to minimize total flow time. Gao et al. [48] presents a discrete harmony search (DHS) algorithm for solving no-wait flow shop scheduling problems with makespan criterion.

Engin and Gunaydin [49] developed an adaptive learning approach for $F|no-wait|C_{\max}$. Qian et al. [50] proposes a hybrid algorithm based on differential evolution to minimize the total completion time of the no-wait flow shop scheduling problem with sequence-dependent setup times and release dates.

Ying et al. [51] examines the no-wait flow shop manufacturing cell scheduling problem with sequence dependent family setup times. The study proposes three metaheuristics and applies them to a set of randomly generated problems.

Hohn et al. [52] consider a variant of no-wait flow shop scheduling that is motivated by continuous casting in the multistage production process in steel manufacturing. This research focuses on complexity results and proposes an intuitive optimal algorithm for scheduling on two processing stages with one machine in the first stage.

Jolai et al. [53] consider the no-wait flexible flow shop problem with sequence dependent setup times with maximum completion time minimization objective. This study proposes a population based simulated annealing (PBSA), adapted imperialist competitive algorithm (AICA) and hybridization of adapted imperialist competitive algorithm and population based simulated annealing (AICA+PBSA) to deal with the problem. These algorithms are tested on a set of randomly generated problems.

Naderi et al. [54] consider the no-wait flow shop problem with total tardiness and makespan criteria. This study develops mathematical models for this problem. Also, heuristics are developed to find good-quality solutions for the problem. [55] performs a literature review on the no-wait scheduling problems.

A review of the literature for no-wait flow shop scheduling problems reveals that:

1. Although the no-wait flow shop problems have a number of practical applications, the literature related to them is limited [5, 56].

2. Because of the computational constraints, almost all the heuristics and metaheuristics before 2004 are considered obsolete.
3. Previous algorithms are tested on a limited number of test problems [15, 28, 30, 32, 34, 41, 57].

Chapters 3 and 4 aim to study practical situations. Solutions generated by the proposed algorithms of this research are compared with the solutions of the best available algorithms in the literature. The proposed algorithms prove to be competitive in terms of both quality of the developed solutions and CPU time. Section 2.3 performs a literature review on the no-wait job shop scheduling problems.

2.3 No-Wait Job Shop Problem

As mentioned earlier, the literature on No-Wait Job Shop (NWJS) is very limited compared to the flow shop version or the classical job shop problem. Moreover, most of the early efforts are devoted to complexity results [5, 56, 58-60]. Hall and Sriskandarajah [5] review the efforts that had been performed to solve NWJS problem by then.

Sriskandarajah and Ladet [60] as well as Kubiak [61] proposed pseudo-polynomial time algorithms for two-machine no-wait job shop problems with the added constraint of the unit processing time. Reddi and Ramamoorthy [9] derived a lower bound for the special case of NWJS in which there exists no subsuming jobs in the job set.

Goyal [62] proved that solving NWJS is equal to solving a set of asymmetrical traveling salesman problem (ATSP). Woeginger [63] proved that NWJS problem with makespan criteria is APN-hard for the case of two machines with at most five operations per job and for the case of three machines with at most three operations per job. These

results prove that these problems do not possess a polynomial time approximation scheme, unless P=NP.

Krachenko [64] developed a polynomial algorithm for NWJS problem with 2 machines. Krachenko developed a $O(n^6)$ algorithm to minimize the mean flow time when zero processing time operations are not allowed.

Mascis and Pacciarelli [65, 66] developed a branch and bound algorithm for NWJS. In this research, the idea of disjunctive graphs is extended to alternative graphs; this transforms the NWJS problem with makespan minimization criteria into minimizing the longest path of the alternative graph with no cycles occurred. This algorithm is able to solve instances as large as 15 jobs and 5 machines or 10 jobs and 10 machines in a reasonable time. However, this method does not solve larger instances in a reasonable time. For the job shop scheduling problem with blocking or no-wait constraints, Meloni et al. [67] proposed a rollout method in terms of a general alternative graph model based on a look-ahead strategy.

Moreover, a tabu search method [19], a simulated annealing [7], and a rolling horizon method [68] have been developed for specific production systems, such as galvanic industry, production of pharmaceutics and semiconductor testing facilities. As a result, these papers do not provide computational results on known benchmark instances. Simulated annealing of Raaymakers and Hoogeveen [7] was able to improve the best initial solutions by 10% in average. The two phase tabu search of Macchiaroli et al. [69] was able to improve the solutions of the dispatching rules by an average of 9%. Afterward, Brizuela et al. [70] proposed a genetic algorithm that used to solve classical job shop problems with no-wait constraints imposed to them. Gui-Juan Chang [71] studied a real-time algorithm that implemented NWJS concept in a metal supply chain.

Schuster and Framinan [72] developed a timetabling algorithm that could construct a non-delay timetable for a given sequence. This research combined this approach to a variable neighborhood search and a hybrid of genetic algorithm and simulated annealing (GASA). Additionally, the algorithms were tested on a set of well-known benchmark instances, and provide solutions of a big improvement relative to the reference solutions. Framinan and Schuster [73] enhanced the idea of non-delay timetabling algorithm of [72], developing a new timetabling algorithm that generated non-delay as well as delay timetables out of a given sequence. Schuster [74] developed a tabu search to further improve the results of [72]; Schuster [74] also studied a few complexity problems.

Pan and Huang [75] proposed a hybrid genetic algorithm for NWJS problem. Bozejko and Makuchowski [76] proposed a tabu search method and compared their results with that of [72] and [73].

Zhu et al. [77] proposed a new timetabling method by introducing the concept of right shifting. Right shifting approach makes possible to check more timetables than the reverse timetabling of [73] for a given sequence. Grimes and Hebrard [78] proposed a constraint programming approach to deal with NWJS. It should be noted that although most of best-known solutions to the problems are from these two studies, computational efforts to obtain such solutions in both of the studies is not reasonable. For example, [77] is not very successful when its computational time is limited; good solutions appear only when the algorithm is allowed to iterate without limitation.

Liu and Kozan [79] studied the case of no-wait parallel machine job shop problem. They further used their model to schedule trains with different priorities in a single-line rail network.

Mokhtari et al. [80] studied a NWJS environment in which processing time is variable based on the amount of resources allocated to it. As a result, [80] added the crashing sub-problem into the traditional timetabling and sequencing sub-problems to which NWJS is decomposed. In order to deal with the crashing sub-problem, timetabling algorithm of [72] is enhanced.

Zhu et al. [81] developed a greedy divide and conquer search for a NWJS problem with two machines. According to this method, the 2-machine NWJS is decomposed to several 2-machine no-wait flow shop problems. Bozejko and Makuchowski [82] developed a genetic algorithm for NWJS. This genetic algorithm was able to automatically tune its parameters based on the test case. Although this algorithm does not need tuning and thus is easier to work with, it is not very competitive in terms of solution quality.

Santosa et al. [83] develops a hybrid of cross entropy metaheuristic and genetic algorithm to deal with NWJS. Results of the developed algorithm is compared with [72] and [73]. Finally, Burgy and Groflin [84] studied a NWJS environment, in which sequence dependent setup times are allowed. This study develops a polynomial algorithm for optimal job insertion problem in such environment.

A review of the literature on $J \mid no-wait \mid C_{\max}$ shows that the problem has a number of practical applications; however, the number of researches that have been conducted to deal with this problem are very limited and the need to develop novel and efficient methods to cope with NWJS is strongly felt [5, 56, 77]. As section 6.5 reveals, any study that develops heuristics or metaheuristics before 2003 is considered obsolete due to computational restrictions. Moreover, research on more realistic environments in which NWJS occurs is being paid attention to just recently. The current research studies

the $J \mid no-wait \mid C_{\max}$ and the timetabling algorithms that have been developed for this problem. A design of experiments method is developed to study the relation between the timetabling algorithm and sequencing algorithms. This sheds light on more effective ways to conduct researches in the future.

Chapter 3

2-Machine No-Wait Flow Shop Problem

3.1 Background

No-wait flow shop scheduling problem refers to the set of problems in which a number of jobs are available for processing on a number of machines in a flow shop context with the added constraint that there should be no waiting time between consecutive operations of the jobs. A special case of the no-wait flow shop problem is considered in which all the jobs have two operations to be processed and there are two machines to process all the jobs. When the problem is in a flow shop context, the order of operations for all the jobs is the same. Moreover, each operation demands a setup on its respective machine. Setup times are assumed to be separable. Separable setup means that the setup times are separable from the jobs in the schedule. In other words, machines can be set up for a specific operation, and then remain idle until the operation is ready to be loaded on the machine. The importance of considering setup times in flow shop contexts is highlighted in [18]. Afterwards, it is assumed that there is only one server to perform the setup operations. In other words, setup operations cannot be performed at the same time. No-wait constraints denote that operations of a job have to be processed without waiting at consecutive machines. The considered performance measure of the obtained solutions is the makespan. Following the three-field notation of scheduling problems, the problems can be designated as $F2|no-wait,setup|C_{\max}$ and $F2,S1|no-wait,setup|C_{\max}$ [85].

Classical flow shop problem $F||C_{\max}$ is proven to be NP-hard [86]. Classical flow shop problem has been widely studied in the literature. Reader is refered to [87], [88], and [89] for the proposed optimal methods of solving this problem. [90] proved that the problem of $F|no-wait|C_{\max}$ is NP-Hard. Moreover, [91] proved that the problem of $F2|no-wait|C_{\max}$ is strongly NP-Hard. [92] and [93] analyzed the complexity of the problems with single server. [92] showed that $F2,S1|no-wait,setup|C_{\max}$ is strongly NP-hard or polynomially solvable depending on the interpretation of zero processing or setup times. [93] strengthened some results of [92] and present some new complexity results. Since job shop problem is a generalization of flow shop problem, it is inferred that the job shop version is also strongly NP-hard. Therefore, all of the problems considered in this chapter are strongly NP-hard.

[5] reviewed the literature and the numerous applications of this problem in their survey paper. Scheduling problems with no-wait constraints occur in many industries. For example, in hot metal rolling industries, the heated metal has to undergo a series of operations continuously at high temperatures before it is cooled to prevent defects [57]. Similarly, in the plastic molding and silverware production industries, a series of operations must be performed, each immediately after the others to prevent degradation [57]. Other examples include chemical and pharmaceutical industries [24], food processing industries [5], and semiconductor testing facilities [68]. A thorough literature survey about scheduling problems with setup times is available at [94] and [95]. To the best of the author's knowledge, there is very limited literature available about $F2|no-wait,setup|C_{\max}$ and $F2,S1|no-wait,setup|C_{\max}$ or their job shop versions.

3.2 Notations

The notations used in this chapter are listed below:

n	Number of jobs
J_i	Job i
s_i, t_i	Setup time of J_i on the first and second machine respectively
a_i, b_i	Processing time of J_i on the first and second machine respectively
ST_{s_i} , ST_{t_i}	Starting time of s_i and t_i
ST_{ij}	Starting time of the operation j of the job i on its respective machine
Π	Set of all permutations of n jobs
π	A permutation in Π
$C_{\max}$	Makespan

Brackets are used to indicate consecutive jobs. For example, $a_{[i]}$ refers to the processing time of the job planned to operate after ith job in a given sequence on the first machine.

This chapter considers the two-machine no-wait flow shop problem with separable setup times and single server side constraints, and makespan as the performance measure. A mathematical model of the problem is developed and a number of propositions are proven for the special cases. Furthermore, a hybrid algorithm of Variable Neighborhood Search (VNS) and Tabu Search (TS) is proposed for the generic case. For evaluation, a number of test problems with small instances are generated and solved to optimality. Computational results show that the proposed algorithm is able to reproduce the optimal solutions of all of the small-instance test problems. For larger instances, proposed solutions are compared with the results of the famous 2-Opt algorithm as well as a lower bound. This comparison demonstrates the efficiency of the algorithm to find good-quality solutions.

3.3 Problem Description

Mathematical model of the problem is as follows:

$$\min C_{\max} \tag{3-1}$$

$$C_{\max} \geq ST_{i1} + a_i + b_i;\ i = 1,2,...,n \tag{3-2}$$

$$ST_{[i]1} \geq ST_{i1} + a_i + ST_{s_{[i]}} + s_{[i]};\ i = 1,2,...,n-1 \tag{3-3}$$

$$ST_{[i]2} \geq ST_{i2} + b_i + ST_{t_{[i]}} + t_{[i]};\ i = 1,2,...,n-1 \tag{3-4}$$

$$ST_{i2} = ST_{i1} + a_i;\ i = 1,2,...,n \tag{3-5}$$

$$ST_{s_i} - ST_{t_i} \geq t_i - Mz_i;\ i = 1,2,...,n \tag{3-6}$$

$$ST_{t_i} - ST_{s_i} \geq s_i - M(1 - z_i);\ i = 1,2,...,n \tag{3-7}$$

$$ST_{ij} \geq 0; z_i \in \{0,1\};\ i = 1,2,...,n;\ j = 1,2 \tag{3-8}$$

In this model, (3-3) and (3-4) initiate the setup times, and (3-5) denotes the no-wait constraints. (3-6) and (3-7) designate the single server constraints. Therefore, removing (3-6) and (3-7) will remove the single server constraints from the model. In the above model, M is a sufficiently large number. From a different point of view, if Π is considered as the set of all permutations π of $N = \{1,2,...,n\}$, the two-machine no-wait flow shop problem can be formulated as follows:

$$\min\left\{C_{\max} \mid \pi \in \Pi, \text{Problem constraints applies}\right\} \tag{3-9}$$

New modeling of the problem in (3-9) implies that it is intended to find a permutation π of $1,2,...,n$ that minimizes the makespan based on the problem instance. Consider the example of Table 3-1 in which $n = 3$:

Table 3-1 Problem Data

i	1	2	3
a_i, b_i	3,5	2,4	1,2
s_i, t_i	2,3	1,1	1,1

Obviously, a possible permutation for this problem is $\pi = (1,2,3)$. Figure 3-1 illustrates the Gantt chart of this solution with no-wait, and with single server constraints. Gantt chart of Figure 3-1 depicts the effect of the no-wait and single server constraints on the scheduling of the consecutive operations of a job.

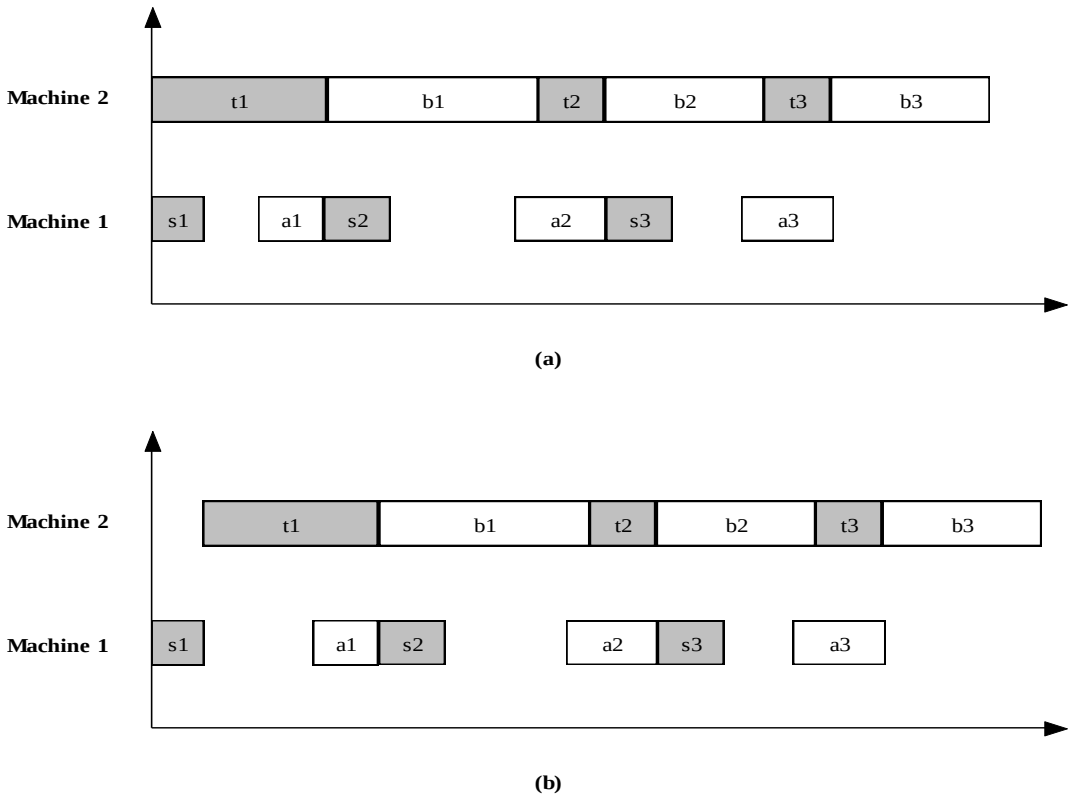

Figure 3-1 Gantt chart: (a) without single server, (b) with single server

The following algorithm calculates the makespan of a given permutation with single server constraints:

1. Set $i=1$, $ST_{s_1}=0$, and $ST_{t_1}=s_1$.

2. Set $ST_{11}=\begin{cases} s_1 & \text{if } t_1 \le a_1 \\ ST_{t_1}+t_1-a_1 & \text{if } t_1 > a_1 \end{cases}$

3. Set $ST_{12}=ST_{11}+a_1$. And set $i \leftarrow i+1$.

4. Suppose that $i=u;1<u\le n$, then $ST_{s_u}=ST_{(u-1)1}+a_{u-1}$, and $ST_{t_u}=ST_{s_u}+s_u$.

5. $ST_{u1}=\begin{cases} ST_{s_u}+s_u & \text{if } t_u \le a_u \\ ST_{t_u}+t_u-a_u & \text{if } t_u > a_u \end{cases}$

6. $ST_{u2}=ST_{u1}+a_u$

7. If $i<n$, set $i \leftarrow i+1$ and go back to step 4. Otherwise, $C_{\max}=ST_{n2}+b_2$.

This algorithm is proved to be very efficient [34, 92]. The above algorithm can easily be modified for the problem without single server constraints. The only change would be modifying step 1 and 4 to $1'$ and $4'$ as follows:

$1'$. Set $i=1$, $ST_{s_1}=0$, and $ST_{t_1}=0$.

$4'$. Suppose that $i=u;1<u\le n$, then $ST_{s_u}=ST_{(u-1)1}+a_{u-1}$, and $ST_{t_u}=ST_{(u-1)2}+b_{u-1}$.

Computational complexity of this simple yet effective algorithm is $O(n)$ which makes it possible to search vast areas of the solution space very fast. Moreover, this algorithm shows that $F2|no-wait|c_{\max}$ can be formulated as follows:

$$\underset{s\in S_n}{Min}\left\{a_1+\sum_{i=1}^{n-1}Max\left(0,a_{[i]}-b_i\right)+\sum_{i=1}^{n}b_i\right\}=\underset{s\in S_n}{Min}\left\{a_1+\sum_{i=1}^{n-1}Max\left(0,a_{[i]}-b_i\right)\right\}+\sum_{i=1}^{n}b_i \quad (3\text{-}10)$$

In addition, $F2|no-wait,setup|c_{\max}$ can be formulated as follows:

$$\begin{aligned} &\underset{s\in S_n}{Min}\left\{Max\left(s_1+a_1,t_1\right)+\sum_{i=1}^{n-1}Max\left(0,s_{[i]}+a_{[i]}-b_i-t_{[i]}\right)+b_1+\sum_{i=2}^{n}\left(t_i+b_i\right)\right\}= \\ &\underset{s\in S_n}{Min}\left\{Max\left(s_1+a_1,t_1\right)+\sum_{i=1}^{n-1}Max\left(0,s_{[i]}+a_{[i]}-b_i-t_{[i]}\right)\right\}+\sum_{i=1}^{n}b_i+\sum_{i=2}^{n}t_i \end{aligned} \tag{3-11}$$

Therefore, the following propositions hold:

Proposition 1, $F2|no-wait|c_{\max}$**:** if $s_l \in S$ is such that $a_{[i]} \le b_i; i=1,2,...,n-1$ and $a_1 = \min_i\{a_i\}$, then s_l is an optimal solution of its instance.

Proof: assume that s_l is not an optimal solution of its respective problem instance. Exchanging the jobs i and $[j]$ $(i,[j]\neq 1)$ results in another permutation s_l'. The difference between the makespan of s_l and s_l' belongs to the mentioned exchange. Based on the proposition's assumptions, $a_{[i]}-b_i \le 0$ in s_l. In s_l', if $a_{[i]}-b_j \le 0$ and $a_j-b_{[i]} \le 0$, then the objective function value is as good as s_l. Otherwise, the makespan of s_l' is greater than the makespan of s_l. On the other hand, assume that s_l' is generated from s_l by exchanging the jobs $i=1$ and j. Since $a_1 = \min_i\{a_i\}$, using (3-10) shows that the makespan will increase; and the proposition follows.

Proposition 2, $F2|no-wait,setup|c_{\max}$**:** if $s_l \in S$ is such that $a_{[i]}+s_{[i]} \le b_i+t_{[i]}; i=1,2,...,n-1$ and $a_1+s_1 = \min_i\{a_i+s_i\}$, then s_l is an optimal solution of its instance.

Proof: follows the same steps of the proof of the proposition 1 and using (3-11) instead of (3-10).

Proposition 3, $F2|no-wait|c_{\max}$ **:** if $s_l \in S$ is such that $a_i \geq b_j; \forall i, j \in s_l$, then s_l is an optimal solution of its instance if $b_n = \min_i \{b_i\}$.

Proof: according to (3-10):

$$c_{\max}^{s_l} = \left\{ a_1 + \sum_{i=1}^{n-1} Max\left(0, a_{[i]} - b_i\right) \right\} + \sum_{i=1}^{n} b_i \tag{3-12}$$

Based on the proposition's assumptions:

$$c_{\max}^{s_l} = \left\{ a_1 + \sum_{i=1}^{n-1} \left(a_{[i]} - b_i\right) \right\} + \sum_{i=1}^{n} b_i = \sum_{i=1}^{n} a_i - \sum_{i=1}^{n-1} b_i + \sum_{i=1}^{n} b_i = \sum_{i=1}^{n} a_i + b_n \tag{3-13}$$

And the proof follows.

Proposition 4, $F2|no-wait, setup|c_{\max}$ **:** if $s_l \in S$ is such that $t_1 \leq s_1 + a_1$, and $b_i - a_{[i]} \leq s_{[i]} - t_{[i]}; i = 1,2,...,n-1$, and $b_n = \min_i \{b_i\}$, then s_l is an optimal solution of its instance.

Proof: follows the same steps of the proof of the proposition 3 and using (3-11) instead of (3-10).

Proposition 5, $F2,S1|no-wait, setup|c_{\max}$ **:** if $s_l \in S$ is such that $a_i \geq t_i; \forall i$, and $b_i - s_{[i]} \leq a_{[i]} - t_{[i]}; \forall i \neq n$, and $s_1 + a_1 = \min_i \{s_i + a_i\}$, and $b_n = \min_i \{b_i\}$, then s_l is an optimal solution of its instance.

Proof: follows the same steps of the proof of the proposition 4 and using (3-11) instead of (3-10).

3.4 The Proposed Algorithm

The proposed algorithm is a hybrid of Variable Neighborhood Search (VNS) and TS. VNS, a meta-heuristic proposed in [96, 97], is based upon a simple principle:

systematic change of the neighborhood within the search. VNS searches in the neighborhood of the current solution in order to move to a better feasible solution. After a number of futile searches, the algorithm chooses another neighborhood structure and the search continues.

The proposed algorithm uses two widely used neighborhood structures: exchange and insert. Consider a current permutation $\pi = (1, 2, ..., i-1, i, i+1, ..., j-1, j, j+1, ..., n)$. The exchange neighborhood randomly chooses two jobs in permutation π (i and j for example) and exchanges the place of these two jobs in the sequence with each other, generating the new permutation $\pi' = (1, 2, ..., i-1, j, i+1, ... j-1, i, j+1, ..., n)$. On the other hand, the insert neighborhood randomly chooses an operation (i for example) and a place (j for example) in the sequence and inserts the operation in the chosen place, shifting all the operations between i and j (including j) one level back in the sequence. Thus, the permutation $\pi' = (1, 2, ..., i-1, i+1, ... j-1, j, i, j+1, ..., n)$ will be generated.

TS is a meta-heuristic approach developed by Glover [98]. At each iteration, TS moves from a current solution in the feasible space to the best solution in its neighborhood. TS uses a tabu mechanism to prevent repetition of the same solutions. For example, a tabu list or a Short Term Memory (STM) can be used to prevent the existence of some specifications of the old solutions in the newly produced solutions for the next θ iterations. Other familiar routines used by TS are aspiration criterion, diversification, and intensification. For more details, the reader is referred to [98]. In the following, the elements of the proposed algorithm are described.

In order to diversify the search, L initial solutions (or permutations) will be generated randomly to initiate the search. L is a control parameter set by the user. The

algorithm calculates the objective function value of each of the generated solutions and sorts them in an ascending order based on their objective function values. Therefore, the best solution with the least makespan stays at the top of the list while the worst solution remains at the bottom of the list.

The process of selecting a solution to be stored in the list, and searching for a better solution in its neighborhood fulfills diversification as well as intensification. The selection process is done through a probabilistic rule in which better solutions stored in the list are more probable to be selected. This means that less competitive solutions stored in the list can also be selected for further search in their neighborhood. In other words, the diversification procedure is performed by the selection of less competitive solutions, while selecting good solutions carries out intensification. As iterations continue, the algorithm fills the list with better solutions and removes uncompetitive schedules from the list. This gradually reduces the significance of diversification while increases the importance of intensification. It can be inferred from the above explanations that the procedure is sensitive to the probability function applied to the selection procedure. The proposed algorithm employs the probability function introduced in [99]. This probability assigns the selection probability $\frac{2i}{|L|\times|L+1|}$ to the ith worst solution stored in the list.

Afterwards, the VNS algorithm uses the exchange sub-algorithm in order to find a better solution in the neighborhood of the selected solution. Exchange sub-algorithm stops after R futile searches and the algorithm activates the insert sub-algorithm for the same solution. Insert sub-algorithm is performed for a maximum of R' times. Then, if the exchange or insert cannot find a better solution, the algorithm selects another

solution from the list and starts searching in the neighborhood of the newly selected solution.

The exchange or insert sub-algorithms stop at any point a better solution is found in the neighborhood of the current solution. Consequently, the improved solution will be added to the list, and the list will be updated. Then, the algorithm removes the worst solution from the list in order to preserve the length of the list. Furthermore, if the exchange or insert improves the selected schedule by exchanging the jobs i and j, (i, j) will be added to the tabu list for the next θ iterations of the algorithm. Therefore, this movement will be forbidden during the next θ iterations of the algorithm unless this is the only exchange that improves the objective function of the current solution or if this exchange leads to a solution that improves the best objective function value found so far. The algorithm stops after T iterations and passes the best found solution to the final intensification sub-algorithm.

3.4.1 Final Intensification

The proposed algorithm uses a straightforward procedure as its intensification sub-algorithm. The final intensification sub-algorithm is performed on the best solution found. This sub-algorithm exchanges the location of the first two adjacent jobs in the sequence and evaluates the objective function value of the new permutation. If the objective function of the new permutation is improved by the exchange, the algorithm accepts this exchange and restarts the whole procedure from the beginning. Nevertheless, if this exchange does not improve the objective function value of the solution, the exchanged jobs will move back to their original locations and the algorithm will be applied to the next two adjacent jobs.

3.4.2 Stepwise Procedure

Step 0: Set the values of the control parameters: L (number of initial solutions), R (number of times that exchange procedure is performed), R' (number of times that insert procedure is performed), θ (length of the tabu list), and T (total number of the iterations of the algorithm).

Step 1: Randomly generate L initial solutions, calculate the objective function of each solution, store them in a list, and sort them in an ascending order. Set $j \leftarrow 1$.

Step 2: If $j <= T$, choose a solution from the list based on the probability function introduced. Set: $i \leftarrow 1$. If $j > T$, go to step 8.

Step 3: If $i > R$, set $i \leftarrow 1$ and go to step 4. If $i <= R$, perform the exchange sub-algorithm. If the objective function of the new solution is improved by the exchange, go to step 6. Otherwise, set $i \leftarrow i+1$ and repeat this step.

Step 4: If $i > R'$, go to step 5. If $i <= R'$, perform the insert sub-algorithm. If the objective function of the new solution is improved by the insert, go to step 6. Otherwise, set $i \leftarrow i+1$ and repeat this step.

Step 5: Set $j \leftarrow j+1$. Go to step 2.

Step 6: Check if the exchange or insert characteristics are not included in the tabu list. If the characteristics are included, go to step 7. Otherwise, add the exchange or insert characteristics to the tabu list. Add the new solution to the list and remove the worst solution from the list. Set: $j \leftarrow j+1$. Go to step 2.

Step 7: Check the described aspiration criterion. If the aspiration criterion is satisfied, add the exchange or insert characteristics to the tabu list. Add the new solution

to the list and remove the worst solution from the list. Set: $j \leftarrow j+1$. Go to step 2. Otherwise, Set: $j \leftarrow j+1$. Go to step 2.

Step 8: Perform the final intensification procedure on the best solution found. Return the final solution of this step as the final solution of the algorithm.

3.5 Computational Results

As seen in step 1 of the stepwise algorithm, five control parameters should be set for the proposed algorithm to start the search. Values of these parameters affect the algorithm's performance. Extensive sensitivity analysis is performed on these parameters to determine the effect of different values of the parameters on the performance of the algorithm. For the sensitivity analysis, the approach introduced in [100] is used. Based on this approach, one problem from each considered set of large instances is chosen. Then, the problems were solved with different combinations of parameter values until the best combination is observed. Based on these observations, the following values are proposed:

$$\begin{aligned} &L = \max\{5, n/100\} \\ &R = \min\{200, 2\times n\} \\ &R' = \min\{200, 2\times n\} \\ &\theta = n \\ &T = 3\times n \end{aligned} \tag{3-14}$$

Furthermore, in order to examine the efficiency of the algorithm, a lower bound is considered for each test problem. The lower bound can be obtained by (3-15) for a given problem instance:

$$LB = \max\{\sum_{i=1}^{n}(s_i + a_i), \sum_{i=1}^{n}(t_i + b_i)\} \tag{3-15}$$

Clearly, smaller gaps between LB and the objective function of the final solution demonstrates the efficiency of the proposed algorithm. For a specific problem, a

small gap means that the algorithm is able to find solutions in which the operating time of the jobs on one of the machines is scheduled parallel to the operating time of the jobs on the other machine. Moreover, the same test problems but without single server constraints are considered and the proposed algorithm is applied to them. Smaller gaps between the objective function values of the same problem without and with single server constraints demonstrate that the proposed algorithm is able to generate schedules that are close to the problem without single server constraints.

The proposed algorithm is coded in C++ and run on a PC equipped with a 3GHz Intel Pentium IV CPU and 2 GB of RAM. To verify the performance of the algorithm, five problems with ten jobs, five problems with twelve jobs, and five problems with fourteen jobs were randomly generated. These test problems have been generated randomly, and based on a uniform probability distribution function with a_i and b_i integer numbers between 0 and 100, s_i an integer number between 0 and a_i, and t_i an integer number between 0 and b_i. The uniform distribution is chosen to generate test problems since it is evident that it results in difficult problems to solve [101, 102].

This set of the test problems with smaller instances is solved to optimality using the mathematical model presented. Moreover, the proposed algorithm is applied to the same set of problems. Computational results of these problems are presented in Table **3-2**. Table **3-2** indicates that the proposed algorithm was able to find the optimal solution of all of instances. It should be noted that solving test problems with more than 14 jobs to optimality takes more than 3 hours. In Table **3-2**, the first column indicates the problem number. Second column represents the number of jobs in the instance. Rest of the columns represent the objective function value and CPU time of the problems without setup, with setup, and with setup and single server, when the respective test

problem is solved by the mathematical model and by the proposed algorithm. Table **3-2** demonstrates that the gap between the optimal solution and the solution found by the proposed algorithm is 0 in all cases.

In addition, the proposed algorithm was applied to a large number of test problems. In order to examine the performance of the algorithm thoroughly, data for the test problems are generated from different intervals. The operation times were generated from [1,10] and [1,100] intervals. Setup times were generated from the intervals $[1,10\alpha]$ and $[1,100\alpha]$ where α can be 0.25, 0.5, 0.75, and 1. Problems with 20, 50, 100, 200, 500, and 1000 jobs were considered in order to test the algorithm on a variety of problems with different sizes. For each problem size, 12 different combinations of operation time, setup time, and α were considered. Additionally, for each specific combination of operation time, setup time, and α, 20 random test problems were generated; this sums up to 1440 test problems of different sizes, and 14400 independent runs of the proposed algorithm as each problem was solved 10 times. Table 3-3 shows the used categories to generate problems' data. In Table 3-4 to Table 3-7, first column gives the problem number. Second column represents the number of jobs of the problem. Columns 3, 4 and 5 are assigned to the interval in which the operation times and setup times are generated as well as the value of α. Next 6 columns summarize the computational results for the problems without setup time, with setup time, and with setup time and single server constraints. As mentioned before, 20 problems were generated for each combination of operation time, setup time, and alpha, given in the tables. Moreover, each generated problem is independently solved 10 times. First, the standard deviation (STD) of these 10 objective functions was calculated. Afterwards, the average of these standard deviations is presented in the Average STD columns.

Table 3-2 Optimal Solutions of Problems with Small Instances

Prob. No.	No. of Jobs	Without Setup Time				Without Single Server Constraints				With Single Server Constraints			
		Optimal Solution		Proposed Algorithm		Optimal Solution		Proposed Algorithm		Optimal Solution		Proposed Algorithm	
		OFV	Time	OFV	Time	OFV	Time	OFV	Time	OFV	Time	OFV	Time
1	**10**	586	0.92	586	0.95	924	1.11	924	2.84	927	1.33	927	2.98
2	**10**	650	1	650	0.88	1,037	1.20	1,037	2.65	1,041	1.44	1,041	2.78
3	**10**	644	1.18	644	0.90	1,035	1.42	1,035	2.70	1,037	1.70	1,037	2.83
4	**10**	412	0.93	412	0.92	747	1.11	747	2.77	747	1.33	747	2.91
5	**10**	526	1	526	1	791	1.20	791	3.01	791	1.44	791	3.16
6	**12**	715	120.32	715	0.94	1,110	144.39	1,110	2.83	1,113	173.27	1,113	2.97
7	**12**	779	119.98	779	0.93	1,233	143.97	1,233	2.78	1,237	172.77	1,237	2.92
8	**12**	710	121.89	710	1.04	1,169	146.26	1,169	3.13	1,173	175.52	1,173	3.29
9	**12**	528	120.34	528	1.11	1,019	144.41	1,019	3.34	1,019	173.29	1,019	3.50
10	**12**	637	125.52	637	0.98	966	150.62	966	2.93	966	180.74	966	3.08
11	**14**	816	3,800.12	816	1.21	1,261	4,560.15	1,261	3.63	1,264	5,472.18	1,264	3.81
12	**14**	878	3,912.54	878	1.30	1,345	4,695.04	1,345	3.90	1,349	5,634.05	1,349	4.10
13	**14**	808	4,017.65	808	1.19	1,299	4,821.18	1,299	3.58	1,300	5,785.42	1,300	3.75
14	**14**	639	3,909	639	1.30	1,251	4,690.80	1,251	3.90	1,251	5,628.96	1,251	4.09
15	**14**	738	4,143.30	738	1.24	1,104	4,971.96	1,104	3.71	1,104	5,966.35	1,104	3.90

*Time is in seconds

Table 3-3 Problem Data Categories

Category Number	Operation Time	Setup Time	α	Category Number	Operation Time	Setup Time	α
1	[1,10]	[1,10]	0.25	**2**	[1,10]	[1,10]	0.5
3	[1,10]	[1,10]	0.75	**4**	[1,10]	[1,10]	1
5	[1,100]	[1,10]	0.25	**6**	[1,100]	[1,10]	0.5
7	[1,100]	[1,10]	0.75	**8**	[1,100]	[1,10]	1
9	[1,100]	[1,100]	0.25	**10**	[1,100]	[1,100]	0.5
11	[1,100]	[1,100]	0.75	**12**	[1,100]	[1,100]	1

The lower the values of the average STD, the more consistently the algorithm is able to solve the problems. In addition, in the following table, Average Time is the average CPU time in milliseconds. Column 12, APAM+SS, calculates the percentage of the time added to the makespan of the problem with setup times when single server constraints are added to the problem. AD (SS, LB) column evaluates the average deviation between the problem with single server constraints and LB. Moreover, the

same problems with single server constraints were solved by the famous 2-opt algorithm for comparison purposes. For a permutation problem, 2-opt algorithm arbitrarily chooses two elements of the permutation and exchanges the two elements; objective function will be calculated and the exchange will be accepted if an improvement is noticed. The algorithm continues until no such improvement can be made [103]. In this chapter, each problem is solved by the 2-opt algorithm. Then, the average deviation between the objective function of the solutions proposed by the 2-opt algorithm and the proposed algorithm is presented in the AD (SS, 2-Opt) column. This column shows that the solutions obtained by the proposed algorithm have always been superior to the 2-Opt algorithm. Table 3-4 to Table 3-7 are abridged for presentation purposes.

Table 3-7 shows that the average CPU time for finding a near-optimal solution for test problems with 1000 jobs is less than 60 seconds which indicates the efficiency of the algorithm. Moreover, the small difference between LB and the objective function of the final solutions with setup time and single server constraints is an index of the quality of the proposed solutions. The lowest average deviation between the objective function of the problem with single server and LB is 1.34%. This means that the optimal solution of this problem is at most 1.34% less than the objective function proposed by the introduced framework. It should be noted that the highest average deviation between the objective function of the problem with single server and LB is 16.80 and belongs to the problems with 1000 jobs. Furthermore, Table 3-7 demonstrates that the average deviation between the problem with single server constraints and LB is 6.45%. According to the problem size, this is a good deviation. It should be noted that the standard deviations in all cases are low. This indicates the consistency of the proposed algorithm in obtaining good-quality solutions. The small deviation between the makespan when single server constraints are added to the problem with setup times

confirms that the proposed algorithm is able to find solutions that single server constraints have slight adversity on their quality. In addition, the deviation between the objective function of the 2-opt algorithm and the proposed algorithm for the problem with single server is another indication of the efficiency of the algorithm.

Table 3-4 Computational Results for the Problems with 20 and 50 Jobs

Prob. Num.	No. of Jobs	Oper. Time	Setup Time	α	Without Setup		With Setup		With Single Server Setup		APAM +SS	AD(SS, *LB*)	AD(SS, 2-Opt)
					Av. STD	Av. Time	Av. STD	Av. Time	Av. STD	Av. Time			
1	20	[1,10]	[1,10]	0.25	0.40	87.84	0.55	87.17	0.43	102.47	0.55	4.06	10.24
2	20	[1,10]	[1,10]	0.5	0.56	86.14	0.51	86.73	0.44	102.90	2.10	5.56	8.89
3	20	[1,10]	[1,10]	0.75	0.57	86.34	0.52	87.33	0.50	103.94	4.04	7.94	8.00
4	20	[1,10]	[1,10]	1	0.53	86.85	0.38	87.38	0.43	106.14	5.44	10.63	6.41
5	20	[1,100]	[1,10]	0.25	2.73	90.10	3.21	89.57	3.06	98.05	0.03	3.13	12.92
6	20	[1,100]	[1,10]	0.5	3.39	89.94	3.67	89.26	3.46	99.22	0.01	2.83	13.87
7	20	[1,100]	[1,10]	0.75	3.32	89.94	3.22	89.38	3.49	98.28	0.06	2.93	14.47
8	20	[1,100]	[1,10]	1	4.06	90.55	4.50	89.73	4.16	99.76	0.10	3.36	13.65
9	20	[1,100]	[1,100]	0.25	3.81	90.17	3.97	89.94	3.79	100.87	0.49	3.33	11.24
10	20	[1,100]	[1,100]	0.5	3.14	90.16	3.06	90.79	3.29	103.32	1.56	4.98	10.66
11	20	[1,100]	[1,100]	0.75	3.51	89.68	2.82	90.30	2.90	106.10	4.02	8.15	8.58
12	20	[1,100]	[1,100]	1	3.33	90.94	1.68	90.54	2.11	110.48	7.02	15.41	8.46
13	50	[1,10]	[1,10]	0.25	0.75	242.88	0.64	242.17	0.73	294.75	0.65	2.13	11.63
14	50	[1,10]	[1,10]	0.5	0.83	243.56	0.62	243.33	0.74	299.75	1.62	4.62	10.36
15	50	[1,10]	[1,10]	0.75	0.66	243.51	0.60	244.95	0.53	304.70	3.20	6.67	8.52
16	50	[1,10]	[1,10]	1	0.57	243.24	0.50	247.30	0.34	311.86	6.17	11.30	6.97
17	50	[1,100]	[1,10]	0.25	3.71	253.57	4.02	253.21	3.62	295.15	0.02	2.35	15.41
18	50	[1,100]	[1,10]	0.5	3.63	252.65	3.65	252.28	3.97	294.20	0.05	1.62	14.59
19	50	[1,100]	[1,10]	0.75	4.11	254.88	3.97	253.96	4.31	296.36	0.07	2.36	15.20
20	50	[1,100]	[1,10]	1	4.14	254.72	5.14	253.74	4.78	297.46	0.15	2.53	15.45
21	50	[1,100]	[1,100]	0.25	4.36	255.39	5.29	254.38	4.53	302.27	0.63	3.10	13.17
22	50	[1,100]	[1,100]	0.5	4.45	253.90	3.70	254.28	4.20	307.91	1.92	4.62	10.95
23	50	[1,100]	[1,100]	0.75	3.83	252.53	3.09	255.15	2.96	316.24	4.21	8.36	8.75
24	50	[1,100]	[1,100]	1	4.86	254.39	2.18	255.74	1.93	323.93	6.62	13.25	8.64

* Time in milliseconds

Table 3-5 Computational Results for the Problems with 100 and 200 Jobs

Prob. Num.	No. of Jobs	Oper. Time	Setup Time	α	Without Setup		With Setup		With Single Server Setup		APAM +SS	AD(SS, LB)	AD(SS, 2-Opt)
					Av. STD	Av. Time	Av. STD	Av. Time	Av. STD	Av. Time			
1	100	[1,10]	[1,10]	0.25	0.94	585.92	0.98	586.92	0.84	716.57	0.74	2.43	10.45
2	100	[1,10]	[1,10]	0.5	0.94	585.41	0.74	595.61	0.70	733.41	2.07	4.73	8.45
3	100	[1,10]	[1,10]	0.75	0.92	588.47	0.69	600.82	0.62	743.34	3.45	7.37	7.33
4	100	[1,10]	[1,10]	1	0.96	586.64	0.46	606.85	0.24	754.88	6.34	12.86	7.45
5	100	[1,100]	[1,10]	0.25	5.75	645.77	4.70	639.56	5.18	735.70	0.00	1.42	15.35
6	100	[1,100]	[1,10]	0.5	5.15	643.56	4.92	637.10	5.35	734.63	0.05	1.34	14.00
7	100	[1,100]	[1,10]	0.75	5.20	648.23	5.82	642.20	5.12	740.25	0.07	1.87	14.86
8	100	[1,100]	[1,10]	1	5.11	650.98	5.39	645.56	5.47	744.24	0.09	1.83	14.27
9	100	[1,100]	[1,100]	0.25	4.56	641.49	4.12	637.65	4.49	740.84	0.55	2.15	11.30
10	100	[1,100]	[1,100]	0.5	5.19	646.21	4.04	639.10	4.32	761.82	1.86	4.56	9.27
11	100	[1,100]	[1,100]	0.75	5.20	638.49	3.44	641.30	2.56	786.52	4.51	8.75	7.79
12	100	[1,100]	[1,100]	1	5.17	645.80	2.34	645.64	1.39	800.84	7.81	14.86	8.88
13	200	[1,10]	[1,10]	0.25	1.53	1,573.16	1.67	1,584.81	1.62	1,861.96	0.53	2.33	11.00
14	200	[1,10]	[1,10]	0.5	1.51	1,574.39	1.41	1,600.13	1.34	1,909.22	1.99	4.76	8.70
15	200	[1,10]	[1,10]	0.75	1.70	1,569.90	1.24	1,610.38	0.77	1,929.37	3.78	7.89	6.89
16	200	[1,10]	[1,10]	1	1.63	1,575.31	0.89	1,622.54	0.24	1,942.35	7.02	13.63	5.03
17	200	[1,100]	[1,10]	0.25	9.78	1,844.84	9.49	1,821.93	10.29	2,054.75	0.01	1.94	14.99
18	200	[1,100]	[1,10]	0.5	10.27	1,845.92	10.77	1,819.65	10.26	2,049.73	0.02	1.96	15.40
19	200	[1,100]	[1,10]	0.75	10.62	1,849.57	10.78	1,824.93	11.31	2,068.13	0.05	2.10	14.72
20	200	[1,100]	[1,10]	1	11.16	1,852.31	10.58	1,827.97	9.97	2,062.67	0.12	2.17	14.66
21	200	[1,100]	[1,100]	0.25	9.43	1,836.96	9.12	1,817.40	9.91	2,073.81	0.53	2.31	12.65
22	200	[1,100]	[1,100]	0.5	10.77	1,849.06	10.58	1,831.90	9.28	2,124.00	2.08	5.57	9.91
23	200	[1,100]	[1,100]	0.75	9.53	1,835.59	8.45	1,819.91	6.59	2,163.88	4.25	8.58	7.83
24	200	[1,100]	[1,100]	1	10.50	1,852.00	5.11	1,803.29	2.21	2,137.60	7.18	14.57	5.12

* Time in milliseconds

Table 3-6 Computational Results for the Problems with 500 and 1000 Jobs

Prob. Num.	No. of Jobs	Oper. Time	Setup Time	α	Without Setup		With Setup		With Single Server Setup		APAM +SS	AD(SS, LB)	AD(SS, 2-Opt)
					Av. STD	Av. Time	Av. STD	Av. Time	Av. STD	Av. Time			
1	500	[1,10]	[1,10]	0.25	4.47	10,426.91	4.12	10,607.52	4.21	12,142.86	0.47	3.24	10.52
2	500	[1,10]	[1,10]	0.5	4.67	10,452.75	4.34	10,672.99	3.73	12,279.72	2.19	5.80	8.35
3	500	[1,10]	[1,10]	0.75	4.74	10,452.71	3.78	10,709.85	2.72	12,325.32	3.78	8.78	6.86
4	500	[1,10]	[1,10]	1	4.00	10,398.84	3.34	10,846.57	0.84	12,063.47	7.05	13.98	4.67
5	500	[1,100]	[1,10]	0.25	34.35	14,094.78	31.89	14,317.53	34.99	15,689.96	0.00	3.29	14.54
6	500	[1,100]	[1,10]	0.5	34.14	14,022.12	33.45	14,274.57	34.92	15,636.23	0.04	3.31	14.23
7	500	[1,100]	[1,10]	0.75	34.63	14,127.98	36.14	14,178.18	31.98	15,631.12	0.06	3.05	14.11
8	500	[1,100]	[1,10]	1	35.39	14,156.32	32.15	14,290.73	34.70	15,705.52	0.09	3.17	13.50
9	500	[1,100]	[1,100]	0.25	32.61	14,130.70	33.72	14,356.85	30.18	15,925.97	0.57	3.86	12.03
10	500	[1,100]	[1,100]	0.5	32.55	14,047.28	32.18	14,186.13	27.80	16,022.19	2.15	6.12	9.62
11	500	[1,100]	[1,100]	0.75	35.07	14,041.43	27.60	14,090.34	19.20	15,921.43	4.31	10.22	7.17
12	500	[1,100]	[1,100]	1	30.77	14,098.69	21.00	13,616.13	6.40	15,093.33	7.60	15.68	6.76
13	1000	[1,10]	[1,10]	0.25	14.27	26,172.30	14.39	25,569.55	14.77	29,273.00	0.36	6.20	7.73
14	1000	[1,10]	[1,10]	0.5	15.63	26,125.25	15.65	25,678.25	13.74	30,190.80	1.85	8.33	6.21
15	1000	[1,10]	[1,10]	0.75	13.93	26,120.70	15.00	25,750.60	12.36	30,819.80	3.11	10.57	6.48
16	1000	[1,10]	[1,10]	1	14.55	26,119.90	13.41	25,856.45	7.55	32,123.70	6.15	15.11	8.82
17	1000	[1,100]	[1,10]	0.25	125.81	54,612.00	116.18	52,476.40	126.92	57,481.70	0.02	7.73	10.24
18	1000	[1,100]	[1,10]	0.5	111.36	54,596.80	134.49	52,457.60	132.84	57,641.30	0.12	7.65	9.78
19	1000	[1,100]	[1,10]	0.75	125.97	54,529.30	136.37	52,396.70	129.70	57,665.70	0.04	7.72	9.71
20	1000	[1,100]	[1,10]	1	125.44	54,512.60	127.29	52,376.20	118.68	57,866.70	0.09	7.42	9.56
21	1000	[1,100]	[1,100]	0.25	141.26	54,524.70	133.40	52,370.10	123.72	58,920.80	0.44	7.68	8.37
22	1000	[1,100]	[1,100]	0.5	123.27	54,515.00	123.14	52,382.80	115.03	60,897.30	1.75	9.16	6.68
23	1000	[1,100]	[1,100]	0.75	127.56	54,527.00	125.35	52,400.20	116.32	63,107.20	3.65	12.15	5.56
24	1000	[1,100]	[1,100]	1	124.88	54,523.80	127.44	52,378.40	72.64	65,462.30	6.71	16.80	7.04

* Time in milliseconds

Table 3-7 Summary of the Computational Results for Large Instance Problems

Num.	No. of Jobs	Without Setup		With Setup		With Single Server Setup		APAM +SS	AD(SS, *LB*)	AD(SS, 2-Opt)
		Av. STD	Av. Time	Av. STD	Av. Time	Av. STD	Av. Time			
1	20	2.45	89.05	2.34	89.01	2.34	102.63	2.11	6.02	10.62
2	50	2.99	250.43	2.78	250.87	2.72	303.71	2.11	5.24	11.64
3	100	3.76	625.58	3.14	626.52	3.02	749.42	2.30	5.35	10.78
4	200	7.37	1,754.91	6.67	1,748.73	6.15	2,031.45	2.30	5.65	10.58
5	500	23.95	12,870.87	21.98	13,012.28	19.30	14,536.43	2.36	6.71	10.20
6	1000	88.66	54,542.65	90.18	52,404.80	82.02	59,880.38	2.01	9.71	8.02
Average		**21.53**	**11,688.92**	**21.18**	**11,355.37**	**19.26**	**12,934.00**	**2.20**	**6.45**	**10.31**

* Time in milliseconds

Figure 3-2 depicts the average percentage added to the makespan when single server is considered (black bars) and average deviation between the problem with single server constraints and LB (grey bars) for each of the mentioned categories of Table **3-3**. This figure disregards the problem size and uses the averages obtained from different problem sizes from each category. Figure 3-2 illustrates that both of the above measures increase as the operation time increases and setup time becomes closer to the operation time.

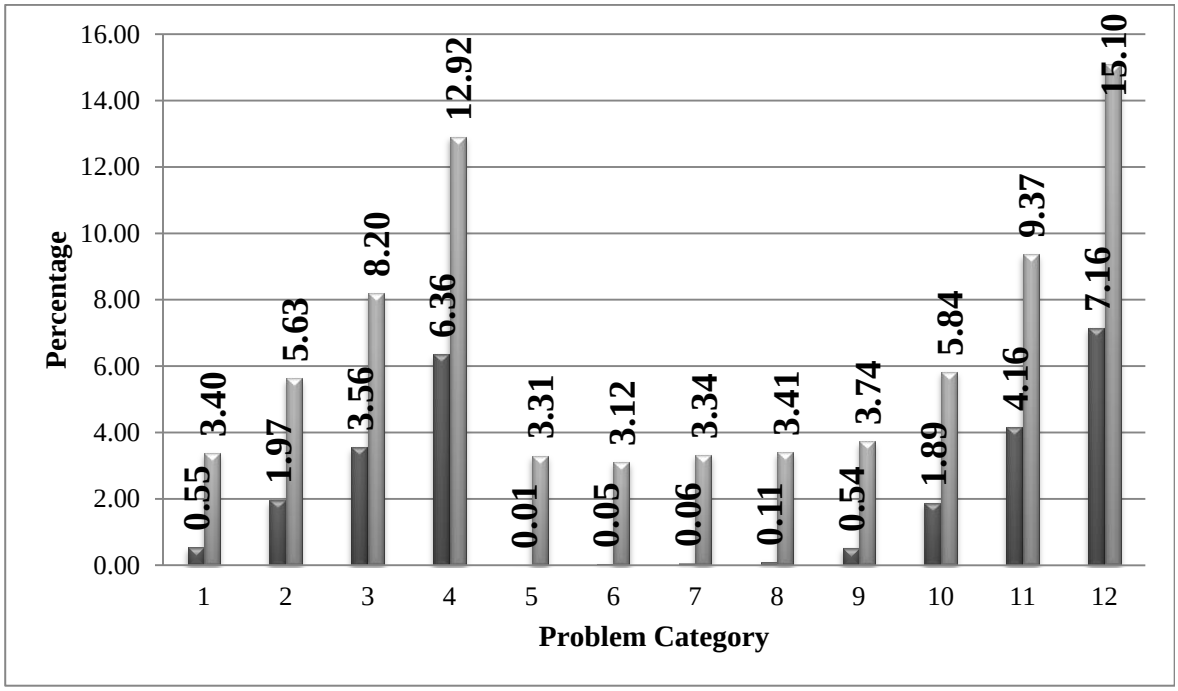

Figure 3-2 Average Percentage Added to the Makespan When Single Server is Considered (black bars) and Average Deviation between the Problem with Single Server Constraints and LB (grey bars)

3.6 Conclusion

In this chapter, two-machine, no-wait flow shop problem with separable setup times is consiered. Moreover, single server side constraints were added to the problem. It is proven that the problems are strongly NP-Hard. As a result, it is not possible to find optimal solutions of large and practical instances of these problems in a reasonable time. A mathematical model of the problem as well as an effective method for calculating the objective function of a given permutation were developed.

A TSVNS was developed and applied to numerous instances of the problems. This algorithm is able to search vast areas of the feasible space employing the reasonable time complexity of the method for calculating the objective function. Computational results of the TSVNS show the efficiency of this algorithm in finding optimal and near optimal solutions for the problems in reasonable time.

Chapter 4

The General No-Wait Flow Shop Problem (NWFS)

4.1 Background

No-wait flow shop scheduling problems refer to the set of problems, in which a number of jobs are available for processing on a number of machines in a flow shop context with the added constraint that there should be no waiting time between consecutive operations of the jobs. Since the problem is in a flow shop context, all the jobs follow the same order in the shop. No-wait constraints denote that operations of a job have to be processed without interruption on consecutive machines. The considered performance measure of the obtained solutions is makespan. Following the three-field notation of the scheduling problems, the mentioned problem can be designated as $F \mid no-wait \mid C_{\max}$ [85]. In other words, the problem that is studied in this chapter is a generalization of the problem that was previously studied in chapter 3.

Afterward, NWFS with separable setup times is considered. Separable setup time means that one can set up a machine for a specific job, and then the machine can be left idle until the job becomes ready for processing. Following the three-field notation of the scheduling problems, this problem can be designated as $F \mid no-wait, setup \mid C_{\max}$.

Finally, NWFS problem is considered with separable sequence dependent setup times. Sequence dependent setup times are considered for each operation. This means that the setup time of an operation on a machine is dependent on the previous operation on the same machine. Following the three-field notation of the scheduling problems, the mentioned problem can be designated as $F \mid no-wait, S_{sd} \mid C_{\max}$.

$F||C_{\max}$ or the classical flow shop problem is NP-Hard [86] and it has been widely studied in the literature. For a review on the exact methods to solve $F||C_{\max}$ refer to [87], [88], and [89]. [12] proved that the no-wait flow shop problem with makespan performance measure ($F|no-wait|C_{\max}$) can be transformed to Asymmetric Travelling Salesperson Problem (ATSP). [90] proved that ($F|no-wait|C_{\max}$) is NP-Hard. [18] transformed the no-wait flow shop problem with separable setup times and makespan criterion ($F|no-wait,setup|C_{\max}$) to ATSP. Since $F|no-wait,S_{sd}|C_{\max}$ is a generalization of $F|no-wait|C_{\max}$ and $F|no-wait,setup|C_{\max}$, it can be inferred that $F|no-wait,S_{sd}|C_{\max}$ is also strongly NP-Hard.

NWFS problem has a number of industrial applications; chemical industries [4], food industries [5], still production [6], pharmaceutical industries [7], and production of concrete products [8], to name a few. For a more comprehensive review of the applications of the problem, the reader is referred to [5].

Although setup times make the problem more applicable and realistic, there are only few researches available in the literature that considers the setup times in the NWFS. Sequence independent and dependent setup times occur in numerous practical practices of no-wait scheduling. Examples of such circumstances include:

- Adjusting jigs and fixtures for processing different products.
- Re-tooling of multi-tool machines.
- Cleaning the machines in order to make them ready for the next operations. Cleaning is an essential part of the manufacturing processes in industries such as textile, plastic, chemical, semi-conductor, pharmaceutical, and food industries.

Industrial applications mentioned in the literature for $F|no-wait,S_{sd}|C_{\max}$ include chemical industries [4], food industries [5], steel production [6], pharmaceutical industries [7], and production of concrete products [8]. [5] provide a comprehensive review of the applications of the problem. [94] and [18] explain why considering setup times in no-wait scheduling problems is essential.

In order to explore the feasible region of the NWFS problem, tabu search of the chapter 3 is hybridized with a Particle Swarm Optimization (PSO). In the proposed approach, PSO algorithm is used in order to move from one solution to a neighborhood solution. A new coding and decoding technique is employed to efficiently map the discrete feasible space to the set of integer numbers. The proposed PSO will further use this coding technique to explore the solution space and move from one solution to a neighborhood solution. Afterwards, the algorithm decodes the solutions to its respective feasible solution in the discrete feasible space and returns the new solutions to the TS. The algorithm is tested by solving a large number of problems available in the literature. Computational results show that the proposed algorithm is able to outperform competitive methods; and improve many of the best-known solutions of the test problems.

To deal with $F|no-wait,setup|C_{\max}$, same PSO algorithm is combined with the GA of section 5.3. In this chapter, a Genetic Algorithm (GA) combined with a Particle Swarm Optimization (PSO) is proposed to solve the problem efficiently. Computational results show that the proposed framework outperforms the previous methods developed for $F|no-wait|C_{\max}$ and improves many of the best-known solutions of the test problems available in the literature. Furthermore, a number of problems with setup times are generated and the proposed algorithms are applied to them. To verify the efficiency

of the proposed algorithms, the results of the proposed algorithms with the results of the famous 2-Opt algorithm are compared [103]; although the 2-Opt algorithm is used to solve the TSP problem in [103], it can easily be modified to be applied to the under-study problems. In the modified version of 2-Opt, the search method in the feasible space remains the same. However, 2-Opt is modified so that it calculates the makespan of the no-wait flow shop with setup. This is possible because the problem is modeled as a permutation problem. The comparison between the results of the 2-Opt algorithm and the developed algorithms for the problems with setup time shows that the proposed algorithms are able to find good-quality solutions for the test problems; and outperform the competitive methods.

Finally, in order to find good-quality solutions for $F|no-wait,S_{sd}|C_{\max}$, a PSO algorithm is proposed. The proposed PSO uses a specific coding/encoding framework in order to map the feasible region of $F|no-wait,S_{sd}|C_{\max}$ to a form, through which the developed PSO is able to explore the feasible region of the problem. The coding framework is called Matrix Coding (MC).

The proposed PSO, when applied to $F|no-wait|C_{\max}$ and $F|no-wait,setup|C_{\max}$, outperforms the competitive algorithms from the literature by improving many of the best-known solutions of the available test problems; for instance, refer to [33]; [104]; [30]; and [72]. Although $F|no-wait,S_{sd}|C_{\max}$ has numerous practical applications, it has received little attention in the literature. Consequently, in order to study the performance of the proposed PSO when applied to $F|no-wait,S_{sd}|C_{\max}$, sequence dependent setup times are incorporated in the algorithm aimed to deal with $F|no-wait,setup|C_{\max}$. Afterwards, the developed PSO,

the modified algorithm of $F \mid no-wait, setup \mid C_{\max}$, and the modified 2-Opt algorithm of [103] are applied to a number of randomly generated test problems. Computational results show that the proposed PSO is able to find good-quality solutions for the test problems and outperform the competitive methods. In addition, the proposed PSO is significantly faster than the competitive methods. It is hoped that the presented results will be used as benchmark by other researchers interested in solving this scheduling problem in the future.

4.2 Notations

The notation that is used throughout this chapter is listed as follows:

n	Number of jobs
m	Number of machines
J_i	Job i
o_{ij}	j th operation of J_i
p_{ij}	Processing time of the j th operation of J_i on its respective machine
S_i	Starting time of the J_i
$S_{o_{ij}}$	Starting time of o_{ij}
ST_{ij}	Setup time of the j th operation of the job J_i on its respective machine when setup times are not sequence dependent
ST_{ijk}	Setup time of o_{ij} if scheduled after o_{kj} when setup times are sequence dependent
ST_{ij0}	Setup time of o_{ij} if J_i is the first job to be scheduled when setup times are sequence dependent
π_l	Sequence l
$C_{\max}$	Makespan

Brackets are used to indicate consecutive jobs, i.e., $S_{[i]}$ refers to the starting time of the job planned to operate after the i th job in a given sequence.

4.3 Mathematical Model

Based on the defined notations, different mathematical models of $F \mid no-wait \mid C_{\max}$ are presented in [30, 33]. A mathematical model for $F \mid no-wait, setup \mid C_{\max}$ is as follows:

$$Min\ C_{\max} \tag{4-1}$$

$$C_{\max} \geq S_{o_{im}} + p_{im};\ i = 1,2,...,n \tag{4-2}$$

$$S_{o_{[i]j}} \geq S_{o_{ij}} + p_{ij} + ST_{[i]j};\quad i = 1,2,...,n-1 \quad j = 1,2,...,n \tag{4-3}$$

$$S_{o_{i(j+1)}} = S_{o_{ij}} + p_{ij};\quad i = 1,2,...,n \quad j = 1,2,...,m-1 \tag{4-4}$$

$$S_{o_{ij}} \geq 0;\ i = 1,2,...,n \quad j = 1,2,...,m \tag{4-5}$$

In this model, the objective function is to minimize the makespan. (4-2) relates the makespan of the objective function to the decision variables. (4-3) calculates the starting time of the operations as well as the starting time of the setup times. (4-4) imposes the no-wait constraints.

Moreover, a mathematical model for $F \mid no-wait, S_{sd} \mid C_{\max}$ is as follows:

$$Min\ C_{\max} \tag{4-6}$$

$$C_{\max} \geq S_{o_{im}} + p_{im};\qquad i = 1,2,...,n \tag{4-7}$$

$$S_{o_{[i]j}} \geq S_{o_{ij}} + p_{ij} + ST_{[i]ji};\quad i = 1,2,...,n-1 \quad j = 1,2,...,m \tag{4-8}$$

$$S_{o_{i[j]1}} = S_{o_{ij}} + p_{ij};\qquad i = 1,2,...,n \quad j = 1,2,...,m-1 \tag{4-9}$$

$$S_{o_{ij}} \geq 0;\qquad i = 1,2,...,n \quad j = 1,2,...,m \tag{4-10}$$

In this model, the objective function is to minimize the makespan. (4-2) initiates the makespan in the model as the objective function by relating it to the decision variables. (4-3) indicates that the starting time of the j th operation of $[i]$ (or the job scheduled after i) should not be sooner than the starting time of the j th operation of i plus its processing time plus the setup time of $o_{[i]j}$ when its previous operation (i in this

case) is taken into consideration. (4-4) imposes the no-wait constraints; finally, (4-10) sets the non-negativity constraints. In other words, mathematical models of $F \mid no-wait, setup \mid C_{\max}$ and $F \mid no-wait, S_{sd} \mid C_{\max}$ are almost alike except for replacing the separable setups with sequence dependent setup times.

4.4 No-Wait Flow Shop Problem

4.4.1 Problem Description

Based on the defined notations, different mathematical models of the problem are presented in [30, 33]. However, from a different point of view, if Π_n is considered as the set of all permutations π of $N = \{1, 2, ..., n\}$ jobs, the no-wait flow shop problem can be formulated as follows:

$$\begin{aligned} &\min C_{\max} \\ &s.t: \\ &\pi \in \Pi_n \\ &\text{No-wait constraints apply} \end{aligned} \tag{4-11}$$

Permutation π represent the order of the priority for allocating the jobs to the Gantt chart. Because of the no-wait constraints, once processing a specific job is started, the job should be processed by the successive machines with no interruption between the consecutive operations. Therefore, the total flow time of J_i can be calculated as follows:

$$C_i - S_i = \sum_{j=1}^{m} p_{ij} \tag{4-12}$$

Consequently, the modeling of the problem in (4-11) holds; and the problem can be reduced to a permutation problem (which is still strongly NP-hard) and a timetabling problem (that in no-wait flow shop problem belongs to P) provided that the no-wait constraints are applied to a specific permutation when the Gantt chart is in preparation, or when the objective function is calculated. In other words, (4-11) implies that a successful search method should explore the feasible region of the problem in order to

find the **permutation** that minimizes the makespan. Moreover, (4-11) also implies that increasing the number of machines in an instance of the problem can barely make the problem more difficult, while increasing the number of jobs makes a specific instance of the problem more complicated.

The idea behind the proposed TS and PSO is to initiate the search by a number of random solutions. The search will be initiated by the TS of section 3.4. In order to move from one solution to a neighborhood solution, TS will pass a sub-section of the current permutation to the PSO. Then PSO, using a one-to-one mapping, will map this smaller permutation into a set of factoradic base numbers and efficiently explore this newly generated set to find a better sub-permutation. The factoradic base is explained in details in section 4.4.2.3. Then the algorithm uses the inverse mapping to map the factoradic number to a unique member of the set of sub-permutations. Afterwards, this sub-permutation will be returned to the TS, proposing a completely new permutation; makespan of the new permutation will be evaluated by the following timetabling algorithm:

1. Set $S_{o_{11}} = 0$, $i \leftarrow 1$.
2. Set $S_{o_{1j}} = S_{o_{1(j-1)}} + p_{1(j-1)}; j = 2,3,...,m$.
3. Set $i \leftarrow i+1$, and $j \leftarrow 2$. Then:
   a. $S_{O_{i1}} = S_{o_{(i-1)1}} + p_{(i-1)1}$
   b. If $S_{o_{i(j-1)}} + p_{i(j-1)} \geq S_{O_{(i-1)j}} + p_{(i-1)j}$, then $S_{o_{ij}} = S_{o_{i(j-1)}} + p_{i(j-1)}$.
   c. If $S_{o_{i(j-1)}} + p_{i(j-1)} < S_{o_{(i-1)j}} + p_{(i-1)j}$, then $S_{o_{ij}} = S_{o_{(i-1)j}} + p_{(i-1)j}$; set $k = j-1$, and:
      i. Set $S_{o_{ik}} \leftarrow S_{o_{ik}} + \left\{\left(S_{o_{(i-1)j}} + p_{(i-1)j}\right) - \left(S_{o_{i(j-1)}} + p_{i(j-1)}\right)\right\}$
      ii. If $k > 1$, set $k \leftarrow k-1$, and go back to 3.3.1. Otherwise, proceed to 3.4.

d. If $j = m$, proceed to 4. Otherwise, set $j \leftarrow j+1$, and go back to 3.2.

4. If $i = n$, stop. $C_{\max} = S_{o_{nm}} + p_{nm}$. Otherwise, go back to 3.

Computational complexity of this simple yet effective algorithm is $O(mn)$. Once the objective function of a proposed permutation is calculated, in case a change is imposed to the permutation, the above algorithm can be applied only to the modified section of the permutation to calculate the new objective function. Developed algorithm of section 4.4.2 exploits this fact in order to reduce the computational time and calculate the objective function of the neighborhood solutions efficiently.

4.4.2 The Proposed Algorithm

The proposed algorithm is a hybrid of the TS of the section 3.4 and PSO. Literature is rich with successful implementations of TS on different problems [105], [106], [107], [108], [109]. In the following subsections, the elements of the proposed algorithm are described.

PSO algorithm has been widely used by researchers to solve combinatorial optimization problems since its introduction in [110, 111]. In PSO a number of particles are moved in search space through a systematic approach. In PSO, at time t, each particle i has a position, $x_i(t)$, and a velocity, $v_i(t)$. Current positions of the particles as well as their best-ever positions are stored in a memory. Velocity of the particles will be changed based on the historical information stored in the memory and also random information. The new velocities will be used to update the current position of the particles, where their new objective function will be evaluated. Since historical data is used in updating the particle velocity, particles tend to return to their historical best position which results in early convergence. To overcome this unwanted phenomena, different velocity update techniques have been developed. The proposed PSO uses one

of the most successful functions available to update the particle velocity. First TS is described, and then the coding approach, which ultimately leads to the proposed PSO, is explained.

4.4.2.1 Adaptive Memory, Diversification and Intensification

As mentioned before, the proposed algorithm uses all the building blocks of the developed TS of the section 3.4. Adaptive memory, diversification, and intensification procedures are described in section 3.4. In short, TS generates a number of initial solutions and stores them in the Adaptive Memory (AM). According to a probabilistic approach, a solution will be selected from the AM for further exploration. This further exploration is performed using the developed PSO.

4.4.2.2 Tabu List

After selecting a solution from the AM, TS will randomly select two jobs (i and j for example) from the permutation. This permutation is called π_1. Suppose that i and j are chosen such that there are at most $x-2$ jobs between them (x is a parameter, set by the user). Therefore, a new permutation is generated with x jobs (π_2). Although the indices of the jobs in π_2 is not from 1 to x, one can easily map these indices into $\{1,2,...,x\}$. Then TS passes π_2 to the PSO. PSO will code this permutation into the set of factoradic numbers. Then PSO will search the factoradic set for a better permutation in the neighborhood of π_2. The result of the PSO is another number in the same factoradic base. Then another algorithm will use an inversion technique in order to map this factoradic number to a new permutation, which is called π_3. π_3 will substitute π_2 in π_1 which results in a new permutation that includes all the jobs (π_4). This procedure will be followed for at most R times or until a better sequence is found. If using the procedure is not successful, the algorithm selects another solution from the AM.

However, if a better sequence is found during the search, the improved solution will be added to the AM, and the list will be updated. At this point, the worst solution stored in the AM is removed to preserve the initial length of the list. This also establishes a gradual shift from diversification to intensification. This procedure exploits the effective characteristics of the TS algorithm by employing a tabu list.

If the exchange or insert improves the selected schedule, (i, j), the indices of the selected jobs, is added to the tabu list for the next θ iterations of the algorithm, forbidding selection of the same jobs at the same time during the next θ iterations of the algorithm. However, this selection is allowed if this is the only choice that improves the makespan of the current solution, or this selection results in a solution with the best found makespan so far. The algorithm stops after T iterations and sends the best-found solution to the final intensification sub-algorithm of the section 3.4.1. At this point, all the building blocks of the TS are introduces. In the sequel, the coding technique and the PSO algorithm are explained.

4.4.2.3 Factoradic base

Factoradic is a specially constructed number system. Factoradics provide a lexicographical index for permutations [112]. The idea of the factoradic is closely linked to that of the Lehmer code [112]. Factoradic is a factorial-based mixed radix numeral system: the ith digit from right side is to be multiplied by $i!$. In this numbering system, the rightmost digit is always 0, the second 0, or 1, the third 0, 1 or 2 and so on [112]. For instance, 38 in decimal base can be shown as $(1_4 2_3 1_2 0_1 0_0)$ in factoradic base.

$$(1_4 2_3 1_2 0_1 0_0) = 1\times 4! + 2\times 3! + 1\times 2! = 38_{10} \quad (4\text{-}13)$$

The factoradic numbering system is unambiguous. No number can be represented in more than one way because the sum of consecutive factorials multiplied by their index is always the next factorial minus one [112]:

$$\sum_{i=0}^{n} i \times i! = (n+1)! - 1 \tag{4-14}$$

More detailed information about factoradic base and factoradic numbering system can be found in [100, 112-114].

4.4.2.4 Relation between factoradic base and permutations

There is a natural mapping between the integers $0, 1, ..., n! - 1$ (or equivalently the factoradic numbers with n digits) and the permutations of n elements in lexicographical order, when the integers are expressed in factoradic form. This mapping has been termed the Lehmer code. For example, with $n = 3$, this mapping is shown in Table **4-1**.

Table 4-1 Natural mapping between factoradic numbers and permutations when n=3

Decimal	Factoradic	Permutation
0_{10}	$0_2 0_1 0_0$	$1, 2, 3$
1_{10}	$0_2 1_1 0_0$	$1, 3, 2$
2_{10}	$1_2 0_1 0_0$	$2, 1, 3$
3_{10}	$1_2 1_1 0_0$	$2, 3, 1$
4_{10}	$2_2 0_1 0_0$	$3, 1, 2$
5_{10}	$2_2 1_1 0_0$	$3, 2, 1$

For mapping factoradic numbers into permutations and vice versa, two straightforward algorithms are presented in [114]. Computational complexity of these algorithms are $O(n)$ which makes the algorithms able to efficiently map the permutations to factoradic numbers, factoradic numbers to decimal numbers and vice

versa. These two algorithms are presented as algorithm 1 and algorithm 2. Table **4-2** demonstrates the mapping of $(1_2 1_1 0_0) \rightarrow (2-3-1)$ based on algorithm 1.

Algorithm 1- mapping factoradic to permutation

Step 1- consider a list of possible digits of factoradic base in ascending order $f = \{0,1,...,n-1\}$, as well as a list of possible numbers in permutation $p = \{1,2,...,n\}$.

Step 2- for $k = 1,2,...,n$: choose the k^{th} digit in factoradic representation, and find this digit in f. Suppose that this digit is the i^{th} digit in f. Choose the i^{th} element of p, set this element as the k^{th} element of permutation, and remove this element from p.

Table 4-2 Mapping $(1_2 1_1 0_0) \rightarrow (2-3-1)$ based on algorithm 1

Iteration	$k=1$	$k=2$	$k=3$
Factoradic	1	1	0
F	{0,1,2}	{0,1,2}	{0,1,2}
P	{1,2,3}	{1,3}	{1}
Permutation	2	3	1

Note that not only the described procedure is useful in the proposed PSO, but also defines a useful framework for mapping a discrete feasible space to an integer number set by a one-to-one mapping.

Algorithm 2- mapping permutation to factoradic

Step 1- consider a list of possible digits of factoradic base in ascending order $f = \{0,1,...,n-1\}$, as well as a list of possible numbers in permutation $p = \{1,2,...,n\}$.

Step 2- for $k = 1,2,...,n$, choose the k^{th} digit in permutation, and find this digit in p. Suppose that this digit is the i^{th} digit in p. Choose the i^{th} element of f, set this element as the k^{th} element of factoradic, and remove this element from p.

Table 4-3 demonstrates the mapping of $(2-3-1) \rightarrow (1_2 1_1 0_0)$ based on algorithm 2.

Table 4-3 Mapping $(2-3-1)\rightarrow(1_2 1_1 0_0)$ based on algorithm 1

Iteration	$k=1$	$k=2$	$k=3$
Permutation	2	3	1
P	{1,2,3}	{1,3}	{1}
F	{0,1,2}	{0,1,2}	{0,1,2}
Factoradic	1	1	0

Now that the coding approach is introduced, the PSO algorithm will be explained in the following section.

4.4.2.5 The Proposed PSO

4.4.2.5.1 Initial Solution

The proposed PSO needs an initial solution to initiate solution space exploration. This initial solution is basically the particle used during the search. Since no particle is born or destroyed during the search, the PSO's input is one permutation, and its output is also one permutation. When this permutation is provided by the TS, algorithm 2 will be used in order to code it to an integer number. Note that based on the one-to-one mapping described in section 4.4.2.4, a random integer number in $[0, x!-1]$ is identical to a random permutation of x jobs. Therefore, these two words can be used interchangeably during the rest of the thesis.

4.4.2.5.2 Particle velocity, neighborhood structure, and stopping criterion

Since the proposed PSO algorithm uses 1 particle to explore the feasible space, 1 particle velocity is also needed to update the particle's position during iterations of the algorithm. The PSO algorithm initially generates a random integer numbers as particle velocity in order to update the particle's position. Note that velocities must be in an appropriate interval so that the particle remains in feasible space after being updated. Since the feasible solution interval is $[0, x!-1]$, the appropriate velocity interval which guarantees feasibility of the particle after update in k th iteration is $[-v^k, (x!-1)-v^k]$.

In which v is the velocity vector. Algorithm must also modify particle velocity during the search to guide the particle through the more desirable areas of the feasible region. Originally, PSO algorithm uses equation (4-15) to update velocities [110, 111]:

$$\overrightarrow{v^{k+1}} \leftarrow \overrightarrow{v^{k}} + cr(\vec{p} - \overrightarrow{h^{k}}) \tag{4-15}$$

Where c is the velocity constant, r is a random number in the interval $[0,1]$, h^k is the current position of the particle in iteration k, and p is the particle's best position so far. However, the proposed PSO employs a development of the original velocity update formula:

$$v^{k+1} \leftarrow \chi\left(wv^{k} + cr(p - h^{k})\right) \tag{4-16}$$

In which w and χ, inertia weight and constriction coefficient, are calculated as follows:

$$\begin{aligned} w &= w_{\max} - \frac{w_{\max} - w_{\min}}{b} \times k \\ \chi &= \frac{2}{c - 2 + \sqrt{c^2 - 4c}}; c > 4 \end{aligned} \tag{4-17}$$

$w_{\max}$ and $w_{\min}$ are two parameters set by the user, b is the total number of iterations, and k is the number of current iteration. Unfortunately, equation (4-16) does not guarantee that the particle remains in the feasible space after update. Therefore, the following conditions are considered:

$$v^{k+1} \leftarrow \begin{cases} r' \times (-h^{k}) & if\ v^{k} < -h^{k} \\ r' \times (x! - 1 - h^{k}) & if\ v^{k} > x! - 1 - h^{k} \end{cases} \tag{4-18}$$

where r' is a random number in the interval $[0,1]$. Particle position is updated by (4-19):

$$h^{k+1} \leftarrow \left\lfloor h^{k} + v^{k} \right\rfloor \tag{4-19}$$

Equation (4-19) will transfer the particle to its neighborhood. After updating the particle position, the algorithm first maps the number to its unique respective permutation and then evaluates the makespan of this new permutation. At this point, the variable p will be updated if necessary. The algorithm will stop and return p to the TS as the final solution after b iterations. b is a parameter set by the user.

4.4.3 Computational Results

As seen in section 4.4.2, seven control parameters should be tuned for the proposed algorithm to initiate the search. Values of these parameters affect the algorithm's performance. The approach of Appendix 1 is employed for tuning the parameters. Based on these observations, the following experimentally derived values are proposed for the parameters:

$$
\begin{aligned}
&L = n \\
&\theta = \left\lfloor \frac{n}{3} \right\rfloor \\
&T = 500 \times n \\
&x = \begin{cases} 4 & \text{for problems with optimal solution} \\ 12 & \text{for problems without optimal solution} \end{cases} \\
&w_{\max} = 1 \\
&w_{\min} = 0.5 \\
&c = 4.1 \\
&b = 10x
\end{aligned}
\tag{4-20}
$$

The proposed algorithm was coded with Microsoft Visual C++ 2008 and run on a PC with 3GHz Intel Pentium IV CPU and 2 GB of RAM. To test the efficiency of the proposed algorithm, a set of 29 well studied problems were chosen from the literature. All the test problems are available at OR-Library [115]. Test problems can be divided into two sets. Problems in the first set are called car1 through car8, introduced by [116]. The optimal solution for the no-wait version of this set is known. The second set consists of 21 problems that are called rec01, rec03, through rec41, introduced by [117]. [117] found this set of problems difficult to solve. Moreover, the optimal solution for the no-

wait version of this set is unknown. These two sets of problems have been widely used in the literature especially by [4], [30], [33], and many others. This is important because this provides an opportunity to compare the proposed algorithm with many other methods.

In the literature considered in the following tables for comparison, each problem is solved 20 times, and the best obtained objective function value along with the average and worst objective function values are reported. The same approach is used in this research in order to maintain integrity with literature. In addition, the average CPU times as well as the standard deviation of the obtained makespans are reported. In the following tables, the best, average, and worst relative errors from the solutions found by [4] are reported. These values are presented as Best Relative Error (BRE), Average Relative Error (ARE), and Worst Relative Error (WRE) in each table respectively. These values are calculated as follows:

$$BRE = \min_{i=1,\ldots,20} \left\{ 100 \times \left| \frac{c^i_{\max} - c^*_{\max}}{c^*_{\max}} \right| \right\} \tag{4-21}$$

$$ARE = \frac{\sum_{i=1}^{20} 100 \times \left| \frac{c^i_{\max} - c^*_{\max}}{c^*_{\max}} \right|}{20} \tag{4-22}$$

$$WRE = \min_{i=1,\ldots,20} \left\{ 100 \times \left| \frac{c^i_{\max} - c^*_{\max}}{c^*_{\max}} \right| \right\} \tag{4-23}$$

Table 4-4 compares the computational results of the proposed algorithm with the particle swarm optimizations proposed by [33] and [104]. In this table, HPSO, HPSO_NONEH, and HPSO_NOSA are proposed by [33], and PSOVNS is proposed by [104]. The problems in this table are from the set of problems with known optimum solutions. In Table 4, the first two columns present the problem name and size. The third column is the optimal solution. The next 4 columns give the BRE, ARE, WRE, and

average CPU time of the proposed algorithm. The same information about HPSO, HPSO_NONEH, HPSO_NOSA, and PSOVNS are presented afterwards. Table 4-5 compares the computational results of the proposed algorithm with the heuristics proposed by [30] and [72]. Since only the BRE is reported in [30], the BRE of the proposed algorithm is shown in this table. It should be noted that in this table, VNS and GASA are proposed by [72] and the rest of the heuristics are presented by [30]. When these results are compared, it can be deduced that the proposed algorithm shows a very good consistency in finding good-quality solutions for small-size instances.

Table 4-6 contrasts the computational results of the proposed algorithm with those of [33] and [4]. It should be noted that Table 4-6 only compares the results of the proposed algorithm with HPSO since the superiority of the HPSO over HPSO_NONEH, HPSO_NOSA, and PSOVNS is evident in [33]. Test problems of Table 4-6 are amongst the second set of the problems. Therefore, the optimal solutions of these problems are unknown. Table 4-6 demonstrates that the proposed algorithm is able to improve the solutions of [33] in 14 out of 21 test problems (66.67%); and for the rest of the test problems, the solutions of the proposed algorithm is as good as those of [33].

Moreover, in terms of the ARE and WRE, the average performance of the proposed algorithm is better than the HPSO of [33] in all of the test problems. Table 4-7 is the comparison of the results of the proposed algorithm with [30] and [72] for the problems without optimal solution. Again, the superiority of the proposed framework over the competitors is clear since the proposed method is able to improve the best-known solutions of 6 out of 21 test cases (28.57%); and for the rest of the test problems, the generated solution of the proposed algorithm is as good as the solutions of the several competitors in the literature. A detailed computational result of the proposed

TS/PSO comes in Table 4-8. Table 4-8 demonstrates that the proposed algorithm improves all the solutions proposed by [4], up to 13 percent in some cases.

Table 4-4 Comparison of the Results of the Proposed algorithm with [33] and [104] for the Problems with Optimal Solution

	n, m	Optimal Solution	Proposed Algorithm				HPSO			HPSO NONEH		
			BRE	ARE	WRE	Time	BRE	ARE	WRE	BRE	ARE	WRE
car1	11,5	8,142	0.00	0.06	0.32	0.00	0.00	0.00	0.00	0.00	0.00	0.00
car2	13,4	8,242	0.00	0.05	0.17	0.00	0.00	0.18	0.61	0.00	0.37	0.62
car3	12,5	8,866	0.00	0.00	0.00	0.00	0.00	0.06	0.24	0.00	0.11	0.27
car4	14,4	9,195	0.00	0.24	1.21	0.00	0.00	1.85	4.29	0.70	1.96	3.46
car5	10,6	9,159	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.41	3.68
car6	8,9	9,690	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
car7	7,7	7,705	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.42
car8	8,8	9,372	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Average		**N/A**	**0.00**	**0.04**	**0.21**	**0.00**	**0.00**	**0.26**	**0.64**	**0.09**	**0.36**	**1.06**

Table 4-4 (Continued)

	n, m	HPSO NOSA			PSOVNS		
		BRE	ARE	WRE	BRE	ARE	WRE
car1	**11,5**	**0.00**	**0.00**	**0.00**	**0.00**	**1.22**	**3.87**
car2	**13,4**	**0.06**	**0.59**	**0.62**	**0.00**	**0.67**	**3.00**
car3	**12,5**	**0.25**	**0.58**	**1.05**	**0.00**	**0.33**	**1.17**
car4	**14,4**	**3.58**	**7.42**	**9.07**	**0.07**	**1.74**	**4.18**
car5	**10,6**	**0.00**	**2.27**	**4.89**	**0.00**	**0.04**	**0.55**
car6	**8,9**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
car7	**7,7**	**0.00**	**0.17**	**1.28**	**0.00**	**0.00**	**0.00**
car8	**8,8**	**0.00**	**0.26**	**0.36**	**0.00**	**0.00**	**0.00**
Average		**0.49**	**1.41**	**2.16**	**0.00**	**0.5**	**1.6**

Table 4-5 Comparison of the Results of the Proposed algorithm with [30] and [72] for the Problems with Optimal Solution

	n, m	Optimal Solution	Proposed Algorithm (BRE)	VNS	GASA	DS	DS+M	TS	TS+M	TS+MP
car1	11,5	8,142	0.00	0.70	0.00	0.00	0.00	0.00	0.00	0.00
car2	13,4	8,242	0.00	0.20	0.00	0.62	0.62	0.00	0.00	0.00
car3	12,5	8,866	0.00	0.00	0.00	0.08	0.08	0.00	0.00	0.00
car4	14,4	9,195	0.00	1.60	0.00	2.77	2.77	0.00	0.00	0.00
car5	10,6	9,159	0.00	3.50	0.00	0.00	0.00	0.00	0.00	0.00
car6	8,9	9,690	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
car7	7,7	7,705	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
car8	8,8	9,372	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Average	**N/A**	**N/A**	**0.00**	**0.75**	**0.00**	**0.43**	**0.43**	**0.00**	**0.00**	**0.00**

Table 4-6 Comparison of the Results of the Proposed algorithm with [33] for the Problems without Optimal Solution

Prob. Number	Prob. Name	n, m	[4] Makespan	Proposed Algorithm								[33]		
				Objective Function Value and The percentage of Improvement Over the Makespan of [4]						Standard Deviation	Average CPU Time	Objective Function Value		
				Best		Average		Worst				Best	Average	Worst
1	**rec01**	20,5	1590	1,395.00	**12.26**	1,511.00	**4.97**	1,520.00	**4.40**	69.72	0.34	3.77	3.39	2.96
2	**rec03**	20,5	1457	1,361.00	**6.59**	1,361.00	**6.59**	1,361.00	**6.59**	0.00	0.31	**6.59**	6.15	3.36
3	**rec05**	20,5	1637	1,511.00	**7.70**	1,517.30	**7.31**	1,520.00	**7.15**	4.62	0.28	7.39	7.15	6.66
4	**rec07**	20,10	2119	2,042.00	**3.63**	2,046.00	**3.45**	2,048.00	**3.35**	3.06	0.38	**3.63**	3.11	2.31
5	**rec09**	20,10	2141	2,027.00	**5.32**	2,033.21	**5.03**	2,042.00	**4.62**	7.54	0.40	4.58	4.26	3.60
6	**rec11**	20,10	1946	1,881.00	**3.34**	1,885.00	**3.13**	1,892.00	**2.77**	5.57	0.39	**3.34**	2.30	1.28
7	**rec13**	20,15	2709	2,545.00	**6.05**	2,547.70	**5.95**	2,552.00	**5.80**	3.53	0.42	**6.05**	5.47	4.80
8	**rec15**	20,15	2691	2,529.00	**6.02**	2,533.40	**5.86**	2,538.00	**5.69**	4.50	0.40	**6.02**	5.69	4.91
9	**rec17**	20,15	2740	2,587.00	**5.58**	2,588.00	**5.55**	2,589.00	**5.51**	1.00	0.44	**5.58**	5.42	5.07
10	**rec19**	30,10	3157	2,861.00	**9.38**	2,870.42	**9.08**	2,876.00	**8.90**	7.58	0.51	9.15	8.50	6.46
11	**rec21**	30,10	3015	2,822.00	**6.40**	2,825.77	**6.28**	2,830.00	**6.14**	4.00	0.49	5.70	5.33	4.74
12	**rec23**	30,10	3030	2,700.00	**10.89**	2,701.85	**10.83**	2,702.00	**10.83**	1.11	0.53	10.80	9.72	8.65
13	**rec25**	30,15	3835	3,593.00	**6.31**	3,611.65	**5.82**	3,658.00	**4.62**	33.47	0.55	5.71	5.17	4.25
14	**rec27**	30,15	3655	3,431.00	**6.13**	3,437.32	**5.96**	3,441.00	**5.85**	5.06	0.58	**6.13**	5.04	4.13
15	**rec29**	30,15	3583	3,291.00	**8.15**	3,300.50	**7.88**	3,303.00	**7.81**	6.33	0.61	7.81	6.93	5.69
16	**rec31**	50,10	4631	4,336.00	**6.37**	4,354.40	**5.97**	4,398.00	**5.03**	31.84	1.29	5.92	5.20	4.51
17	**rec33**	50,10	4770	4,466.00	**6.37**	4,471.50	**6.26**	4,482.00	**6.04**	8.13	1.91	5.51	4.08	3.17
18	**rec35**	50,10	4718	4,417.00	**6.38**	4,424.88	**6.21**	4,428.00	**6.15**	5.67	1.85	6.02	5.13	3.98
19	**rec37**	75,20	8979	8,081.00	**10.00**	8,111.21	**9.66**	8,145.00	**9.29**	32.02	3.42	8.89	8.20	7.40
20	**rec39**	75,20	9158	8,517.00	**7.00**	8,539.01	**6.76**	8,558.00	**6.55**	20.52	3.51	6.79	5.67	4.26
21	**rec41**	75,20	9344	8,520.00	**8.82**	8,570.50	**8.28**	8,583.00	**8.14**	33.36	3.50	7.94	6.77	5.91
Average	**N/A**	**N/A**	**N/A**	**N/A**	**7.08**	**N/A**	**6.52**	**N/A**	**6.25**	**13.74**	**1.05**	**6.35**	**5.65**	**4.67**

- Average CPU time of HPSO is 5 seconds; run on a 2.2 GHz Pentium processor.

Table 4-7 Comparison of the Results of the Proposed algorithm with [30] and [72] for the Problems without Optimal Solution

Prob. Number	Prob. Name	n, m	[4] Makespan	Proposed Algorithm (BRE)	VNS	GASA	DS	DS+M	TS	TS+M	TS+MP
1	**rec01**	20,5	1,590.00	**12.26**	2.77	3.96	3.71	3.58	4.03	3.96	3.96
2	**rec03**	20,5	1,457.00	**6.59**	4.32	4.46	3.43	4.43	**6.59**	**6.59**	**6.59**
3	**rec05**	20,5	1,637.00	**7.70**	7.03	6.90	5.62	5.62	7.39	7.64	**7.70**
4	**rec07**	20,10	2,119.00	**3.63**	2.31	3.45	1.09	1.08	**3.63**	**3.63**	**3.63**
5	**rec09**	20,10	2,141.00	**5.32**	2.38	4.48	3.60	3.60	4.62	4.58	4.58
6	**rec11**	20,10	1,946.00	**3.34**	1.54	3.34	1.44	1.44	**3.34**	**3.34**	**3.34**
7	**rec13**	20,15	2,709.00	**6.05**	5.76	5.65	3.43	4.43	**6.05**	**6.05**	**6.05**
8	**rec15**	20,15	2,691.00	**6.02**	5.91	6.02	4.83	4.83	5.91	**6.02**	5.91
9	**rec17**	20,15	2,740.00	**5.58**	5.15	5.47	5.51	5.51	**5.58**	**5.58**	**5.58**
10	**rec19**	30,10	3,157.00	**9.38**	7.57	5.45	7.70	7.44	9.72	9.25	**9.38**
11	**rec21**	30,10	3,015.00	**6.40**	4.21	2.22	3.68	4.68	6.37	6.30	6.17
12	**rec23**	30,10	3,030.00	**10.89**	10.70	6.70	7.29	7.29	10.76	10.73	**10.89**
13	**rec25**	30,15	3,835.00	**6.31**	5.45	2.69	3.08	3.08	5.97	**6.31**	6.21
14	**rec27**	30,15	3,655.00	**6.13**	5.83	2.60	3.64	3.64	5.64	6.10	5.83
15	**rec29**	30,15	3,583.00	**8.15**	7.23	3.99	7.23	7.36	7.94	**8.15**	7.94
16	**rec31**	50,10	4,631.00	**6.37**	4.71	-2.72	3.76	3.78	5.90	6.13	6.22
17	**rec33**	50,10	4,770.00	**6.37**	5.35	-4.78	1.97	2.01	5.51	6.31	**6.37**
18	**rec35**	50,10	4,718.00	**6.38**	5.51	-3.67	4.94	4.94	6.08	6.17	5.91
19	**rec37**	75,20	8,979.00	**10.00**	**10.00**	-5.89	7.80	7.92	9.41	9.49	9.36
20	**rec39**	75,20	9,158.00	**7.00**	5.32	-8.80	4.97	5.12	**7.00**	6.99	6.91
21	**rec41**	75,20	9,344.00	**8.82**	7.41	-6.79	6.08	6.08	8.78	8.57	**8.82**
Average	**N/A**	**N/A**	**N/A**	**7.08**	**5.55**	**1.65**	**4.51**	**4.66**	**6.49**	**6.57**	**6.54**

- Average CPU time of DS and DS+M is 0 second; run on a 1000 MHz Pentium processor.
- Average CPU time of TS, TS+M, and TS+MP is 0.5 seconds; run on a 1000 MHz Pentium processor.
- Average CPU time of GASA and VNS are 50 and 210 seconds; run on a 1400 MHz Athlon processor.

Table 4-8 Detailed Results of the Proposed Algorithm

Problem Number	Problem Name	n, m	[4] Makespan	Minimum Deviation		Average Deviation		Maximum Deviation		Standard Deviation	Average CPU Time
				Best Found OFV	BRE	Average OFV	ARE	Worst OFV	WRE		
1	**rec01**	20,5	1,590	1,395.00	12.26	1,511.00	**4.97**	1,520.00	**4.40**	69.72	0.34
2	**rec03**	20,5	1,457	1,361.00	6.59	1,361.00	**6.59**	1,361.00	**6.59**	0.00	0.31
3	**rec05**	20,5	1,637	1,511.00	7.70	1,517.30	**7.31**	1,520.00	**7.15**	4.62	0.28
4	**rec07**	20,10	2,119	2,042.00	3.63	2,046.00	**3.45**	2,048.00	**3.35**	3.06	0.38
5	**rec09**	20,10	2,141	2,027.00	5.32	2,033.21	**5.03**	2,042.00	**4.62**	7.54	0.40
6	**rec11**	20,10	1,946	1,881.00	3.34	1,885.00	**3.13**	1,892.00	**2.77**	5.57	0.39
7	**rec13**	20,15	2,709	2,545.00	6.05	2,547.70	**5.95**	2,552.00	**5.80**	3.53	0.42
8	**rec15**	20,15	2,691	2,529.00	6.02	2,533.40	**5.86**	2,538.00	**5.69**	4.50	0.40
9	**rec17**	20,15	2,740	2,587.00	5.58	2,588.00	**5.55**	2,589.00	**5.51**	1.00	0.44
10	**rec19**	30,10	3,157	2,861.00	9.38	2,870.42	**9.08**	2,876.00	**8.90**	7.58	0.51
11	**rec21**	30,10	3,015	2,822.00	6.40	2,825.77	**6.28**	2,830.00	**6.14**	4.00	0.49
12	**rec23**	30,10	3,030	2,700.00	10.89	2,701.85	**10.83**	2,702.00	**10.83**	1.11	0.53
13	**rec25**	30,15	3,835	3,593.00	6.31	3,611.65	**5.82**	3,658.00	**4.62**	33.47	0.55
14	**rec27**	30,15	3,655	3,431.00	6.13	3,437.32	**5.96**	3,441.00	**5.85**	5.06	0.58
15	**rec29**	30,15	3,583	3,291.00	8.15	3,300.50	**7.88**	3,303.00	**7.81**	6.33	0.61
16	**rec31**	50,10	4,631	4,336.00	6.37	4,354.40	**5.97**	4,398.00	**5.03**	31.84	1.29
17	**rec33**	50,10	4,770	4,466.00	6.37	4,471.50	**6.26**	4,482.00	**6.04**	8.13	1.91
18	**rec35**	50,10	4,718	4,417.00	6.38	4,424.88	**6.21**	4,428.00	**6.15**	5.67	1.85
19	**rec37**	75,20	8,979	8,081.00	10.00	8,111.21	**9.66**	8,145.00	**9.29**	32.02	3.42
20	**rec39**	75,20	9,158	8,517.00	7.00	8,539.01	**6.76**	8,558.00	**6.55**	20.52	3.51
21	**rec41**	75,20	9,344	8,520.00	8.82	8,570.50	**8.28**	8,583.00	**8.14**	33.36	3.50
Average	**N/A**	**N/A**	**N/A**	**N/A**	**7.08**	**N/A**	**6.52**	**N/A**	**6.25**	**13.74**	**1.05**

4.5 No-Wait Flow Shop Problem with Setup Time

4.5.1 Problem Description

Because of the no-wait constraints, when the starting time of the first operation of a specific job is determined, starting time of the rest of the operations of that job can be calculated using (4-4). Therefore, in order to calculate the makespan of a solution of $F \mid no-wait, setup \mid C_{\max}$, one should determine the starting time of the first operation of each job, and then calculate the starting time of the rest of the operations based on (4-4). This means that the solutions of the problem can be considered as permutations of the jobs. A permutation list will indicate the priority of scheduling the jobs. Once a permutation list is available, first operations of each job should be scheduled as soon as possible; starting time of the rest of the operations can be calculated by (4-4). In other words, if Π_n is considered as the set of all permutations π of $N = \{1, 2, ..., n\}$ jobs, the no-wait flow shop problem with setup can be formulated as follows:

$$\begin{aligned} &\min C_{\max} \\ &s.t: \\ &\pi \in \Pi_n \\ &\text{No-wait constraints apply} \end{aligned} \tag{4-24}$$

Because of the no-wait constraints, once processing a specific job is started, the job should be processed by the successive machines with no interruption between the consecutive operations. Therefore, the total flow time of J_i can be calculated as follows:

$$C_i - S_i = \sum_{j=1}^{m} p_{ij} \tag{4-25}$$

Consequently, the modeling of the problem in (4-24) holds; and the problem can be reduced to a permutation problem. These results are in accordance with the findings of [3], who proposed that the problem can be transformed to TSP. Herein, the algorithm that calculates the makespan of a given permutation is presented as follows:

1. $S_{o_{11}} = ST_{11}$.
2. For $k = 2$ to m, $S_{O_{1k}} = \max\{ST_{1k}, S_{O_{1(k-1)}} + p_{1(k-1)}\}$. If $ST_{1k} > S_{O_{1(k-1)}} + p_{1(k-1)}$, set $d = ST_{1k} - (S_{O_{1(k-1)}} + p_{1(k-1)})$, and for $h = 1,2,...,k-1$, set $S_{O_{1h}} \leftarrow S_{O_{1h}} + d$.
3. Set $i \leftarrow 2; j \leftarrow 1$.
4. $S_{o_{ij}} = S_{o_{(i-1)j}} + p_{(i-1)j} + ST_{ij}$.
5. $j \leftarrow j + 1$.
6. $S_{O_{ij}} = \max\{S_{o_{(i-1)j}} + p_{(i-1)j} + ST_{ij}, S_{O_{i(j-1)}} + p_{i(j-1)}\}$. If $S_{o_{(i-1)j}} + p_{(i-1)j} + ST_{ij} > S_{O_{i(j-1)}} + p_{i(j-1)}$, set $d = S_{o_{(i-1)j}} + p_{(i-1)j} + ST_{ij} - (S_{O_{i(j-1)}} + p_{i(j-1)})$, and for $h = 1,2,..., j-1$, set $S_{O_{ih}} \leftarrow S_{O_{ih}} + d$.
7. If $i = n$, stop. $C_{\max} = S_{o_{nm}} + p_{nm}$. Otherwise, set $i \leftarrow i + 1$ and $j = 1$. Go back to step 4.

This algorithm starts with a priority list or equivalently, a permutation. It determines the starting time of the first operation of the first priority in the permutation. Then the algorithm calculates the starting time of the rest of the operations of that job, while imposing the no-wait constraints. When scheduling the first job in the priority list is finished, using the same method, the algorithm schedules the next priority in the list, and goes to the next job until the scheduling is finished. Computational complexity of this algorithm is $O(mn)$.

This problem is solved using the GA that will be explained in details in section 5.3 as well as the hybrid of the GA of section 5.3 and PSO of section 4.4.2.5. The hybrid algorithm is called GA+PSO. As it will be mentioned in section 5.3, the local search in the proposed GA is a simple procedure that includes exchanging the priorities in the permutation list. GA+PSO modifies the local search of this GA algorithm. In

GA+PSO, the described PSO of section 4.2.3.5 is used in order to perform the local search.

4.5.2 Computational Results

As seen in section 4.5.1, the developed GA has four control parameters and GA+PSO has 9 parameters. The parameters must be tuned to obtain the best performance of the developed algorithms. The following experimentally derived values are proposed for the GA parameters:

$$
\begin{aligned}
&Pop = \begin{cases} \left\lfloor \frac{n}{2} \right\rfloor & n \text{ is even} \\ \left\lfloor \frac{n}{2} \right\rfloor + 1 & n \text{ is odd} \end{cases} \\
&P_c = 0.5 \\
&P_m = 0.5 \\
&P_l = 0.2
\end{aligned}
\tag{4-26}
$$

The proposed values of parameters for GA+PSO are as follows:

$$
\begin{aligned}
&Pop = \begin{cases} \left\lfloor \frac{n}{2} \right\rfloor & n \text{ is even} \\ \left\lfloor \frac{n}{2} \right\rfloor + 1 & n \text{ is odd} \end{cases} \\
&P_c = 0.5 \\
&P_m = 0.5 \\
&P_l = 0.2 \\
&x = \begin{cases} 4 & \text{for problems with optimal solution} \\ 10 & \text{for problems without optimal solution} \end{cases} \\
&w_{\max} = 1 \\
&w_{\min} = 0.5 \\
&c = 4.1 \\
&b = 10x
\end{aligned}
\tag{4-27}
$$

The programming language chosen to code the algorithms is Microsoft Visual C++ 2008; all the test problem instances are solved on a PC equipped with a 3GHz Intel Pentium IV CPU and 2 GB of RAM. To test the efficiency of the proposed GA and

GA+PSO, the same set of 29 problems of section 4.4.3 were chosen. First these problems (with setup times equal to zero) were solved and the results were compared with the most recent (and the best-known) algorithms in the literature. Afterwards, random setup times for each operation were generated based on the following rule:

$$0 \le ST_{ij} < p_{ij}; \quad i = 1,2,...,n \quad j = 1,2,...,m \quad ST_{ij} \text{ is integer} \tag{4-28}$$

Then the algorithm results were compared with the results of the 2-Opt algorithm.

4.5.2.1 Computational Results for the Problems without Setup Times

For this set of problems, the result reporting scheme is the same as the approach of section 4.4.3. That is, solving each problem 20 times and reporting the best obtained objective function value along with the average, and worst objective function values as well as the average CPU time and the standard deviation of the obtained makespans. Table 4-11 demonstrates that the proposed GA is able to improve the best-known solutions of 16 out of 21 test problems (76%); GA+PSO improves the best-known solution of 20 out of 21 test problems (95.2%).

Moreover, in terms of the averages, the average performance of the proposed GA is better than the HPSO of [33] in 12 out of 21 problems (57%); in terms of the ARE and WRE, the average performance of the proposed GA+PSO is better than the HPSO of [33] in all of the test problems. Table 4-12 is the comparison of the results of the proposed algorithm with [30] and [72] for the problems with unknown optimal solution. It should be noted that the proposed GA improves all the solutions proposed by [4], up to 11.93 percent in some cases. Again, the superiority of the GA+PSO over the competitors is clear since the proposed method is able to outperform other algorithms in 19 out of 21 test cases (90.5%). GA+PSO algorithm improves all the solutions proposed by [4], up to 13 percent in some cases. Moreover, the proposed GA takes very short time

to solve large test cases. For example, for problems with 75 jobs and 20 machines, the algorithm uses slightly more than 1 minute of the CPU time to improve the best solutions found by [4], [30], [72], and [33]. A comparison between Table 4-8 and Table 4-12 reveals that GA+PSO is more successful than TS+PSO for the studied NWFS test problems.

Table 4-9 Comparison of the Results of the Proposed algorithm with [33] and [104] for the Problems with Optimal Solution

	n, m	Optimal Solution	Proposed GA				Proposed GA+PSO				HPSO			HPSO NONEH			HPSO NOSA			PSOVNS		
			BRE	ARE	WRE	T	BRE	ARE	WRE	T	BRE	ARE	WRE	BRE	ARE	WRE	BRE	ARE	WRE	BRE	ARE	WRE
car1	11,5	8,142	0.00	0.06	0.3	0.45	0.00	0.03	0.22	0.48	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.22	3.87
car2	13,4	8,242	0.00	0.06	0.16	0.4	0.00	0.02	0.1	0.54	0.00	0.18	0.61	0.00	0.37	0.62	0.06	0.59	0.62	0.00	0.67	3.00
car3	12,5	8,866	0.00	0.00	0.00	0.42	0.00	0.00	0.00	0.53	0.00	0.06	0.24	0.00	0.11	0.27	0.25	0.58	1.05	0.00	0.33	1.17
car4	14,4	9,195	0.00	0.22	1.23	0.41	0.00	0.2	1.21	0.56	0.00	1.85	4.29	0.70	1.96	3.46	3.58	7.42	9.07	0.07	1.74	4.18
car5	10,6	9,159	0.00	0.00	0.00	0.41	0.00	0.00	0.00	0.45	0.00	0.00	0.00	0.00	0.41	3.68	0.00	2.27	4.89	0.00	0.04	0.55
car6	8,9	9,690	0.00	0.00	0.00	0.45	0.00	0.00	0.00	0.39	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
car7	7,7	7,705	0.00	0.00	0.00	0.33	0.00	0.00	0.00	0.36	0.00	0.00	0.00	0.00	0.02	0.42	0.00	0.17	1.28	0.00	0.00	0.00
car8	8,8	9,372	0.00	0.00	0.00	0.46	0.00	0.00	0.00	0.39	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.26	0.36	0.00	0.00	0.00
Average		**N/A**	**0.00**	**0.04**	**0.21**	**0.42**	**0.00**	**0.03**	**0.19**	**0.46**	**0.00**	**0.26**	**0.64**	**0.09**	**0.36**	**1.06**	**0.49**	**1.41**	**2.16**	**0.00**	**0.5**	**1.6**

Table 4-10 Comparison of the Results of the Proposed algorithm with [30] and [72] for the Problems with Optimal Solution

	n, m	Optimal Solution	Proposed GA (BRE)	Proposed GA+PSO (BRE)	VNS	GASA	DS	DS+M	TS	TS+M	TS+MP
car1	11,5	8,142	0.00	0.00	0.70	0.00	0.00	0.00	0.00	0.00	0.00
car2	13,4	8,242	0.00	0.00	0.20	0.00	0.62	0.62	0.00	0.00	0.00
car3	12,5	8,866	0.00	0.00	0.00	0.00	0.08	0.08	0.00	0.00	0.00
car4	14,4	9,195	0.00	0.00	1.60	0.00	2.77	2.77	0.00	0.00	0.00
car5	10,6	9,159	0.00	0.00	3.50	0.00	0.00	0.00	0.00	0.00	0.00
car6	8,9	9,690	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
car7	7,7	7,705	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
car8	8,8	9,372	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Average	**N/A**	**N/A**	**0.00**	**0.00**	**0.75**	**0.00**	**0.43**	**0.43**	**0.00**	**0.00**	**0.00**

Table 4-11 Comparison of the Results of the Proposed algorithm with [33] for the Problems with unknown Optimal Solution

Problem Number	Problem Name	n, m	[4] Makespan	Proposed GA							
				Objective Function Value and The percentage of Improvement Over the Makespan of [4]							Average CPU Time
				BRE		ARE		WRE		Standard Deviation	
1	**rec01**	20,5	1590	1,526.00	4.19	1,530.60	3.88	1,550.00	2.58	41.57	2.91
2	**rec03**	20,5	1457	1,385.00	5.20	1,395.10	4.44	1,423.00	2.39	63.65	2.94
3	**rec05**	20,5	1637	1,519.00	7.77	1,527.55	7.17	1,546.00	5.89	4.87	2.74
4	**rec07**	20,10	2119	2,042.00	3.77	2,058.90	2.92	2,078.00	1.97	3.00	4.96
5	**rec09**	20,10	2141	2,043.00	4.80	2,055.65	4.15	2,088.00	2.54	14.89	4.64
6	**rec11**	20,10	1946	1,890.00	2.96	1,916.30	1.55	1,956.00	-0.51	12.00	4.40
7	**rec13**	20,15	2709	2,545.00	6.44	2,575.85	5.17	2,640.00	2.61	9.23	6.19
8	**rec15**	20,15	2691	2,529.00	6.41	2,542.50	5.84	2,583.00	4.18	26.57	6.09
9	**rec17**	20,15	2740	2,587.00	5.91	2,606.10	5.14	2,635.00	3.98	10.58	6.37
10	**rec19**	30,10	3157	2,891.00	9.20	2,926.80	7.87	2,964.00	6.51	24.39	5.73
11	**rec21**	30,10	3015	2,843.00	6.05	2,849.70	5.80	2,874.00	4.91	12.93	6.71
12	**rec23**	30,10	3030	2,707.00	11.93	2,738.50	10.64	2,762.00	9.70	7.85	6.85
13	**rec25**	30,15	3835	3,625.00	5.79	3,642.90	5.27	3,677.00	4.30	20.11	9.47
14	**rec27**	30,15	3655	3,457.00	5.73	3,474.85	5.18	3,514.00	4.01	35.82	9.31
15	**rec29**	30,15	3583	3,301.00	8.54	3,366.25	6.44	3,408.00	5.13	42.68	9.11
16	**rec31**	50,10	4631	4,336.00	6.80	4,354.40	6.35	4,398.00	5.30	23.01	10.79
17	**rec33**	50,10	4770	4,509.00	5.79	4,593.00	3.85	4,678.00	1.97	33.90	10.84
18	**rec35**	50,10	4718	4,469.00	5.57	4,486.55	5.16	4,512.00	4.57	8.72	10.67
19	**rec37**	75,20	8979	8,170.00	9.90	8,199.95	9.50	8,304.00	8.13	17.18	61.70
20	**rec39**	75,20	9158	8,593.00	6.58	8,686.35	5.43	8,786.00	4.23	56.39	61.16
21	**rec41**	75,20	9344	8,627.00	8.31	8,695.05	7.46	8,849.00	5.59	25.69	63.60
Average	**N/A**	**N/A**	**N/A**	**N/A**	**6.55**	**N/A**	**5.68**	**N/A**	**4.29**	**23.57**	**14.63**

Table 4-11 (Continued)

Problem Number	Problem Name	n, m	Proposed GA+PSO								[33]		
			Objective Function Value and The percentage of Improvement Over the Makespan of [4]						Standard Deviation	Average CPU Time	Objective Function Value		
			BRE		ARE		WRE				BRE	ARE	WRE
1	**rec01**	20,5	1,395.00	**13.98**	1,512.00	**5.16**	1,527.00	**4.13**	40.74	3.69	3.77	3.39	2.96
2	**rec03**	20,5	1,361.00	**7.05**	1,361.00	**7.05**	1,361.00	**7.05**	0.00	3.72	6.59	6.15	3.36
3	**rec05**	20,5	1,514.00	**8.12**	1,517.00	**7.91**	1,519.00	**7.77**	4.77	3.47	7.39	7.15	6.66
4	**rec07**	20,10	2,042.00	**3.77**	2,047.00	**3.52**	2,052.00	**3.27**	1.11	6.27	3.63	3.11	2.31
5	**rec09**	20,10	2,042.00	**4.85**	2,045.00	**4.69**	2,048.00	**4.54**	2.01	5.87	4.58	4.26	3.60
6	**rec11**	20,10	1,881.00	**3.46**	1,882.50	**3.37**	1,894.00	**2.75**	0.53	5.57	3.34	2.30	1.28
7	**rec13**	20,15	2,545.00	**6.44**	2,548.50	**6.30**	2,552.00	**6.15**	3.05	7.83	6.05	5.47	4.80
8	**rec15**	20,15	2,529.00	**6.41**	2,529.06	**6.40**	2,530.00	**6.36**	0.04	7.70	6.02	5.69	4.91
9	**rec17**	20,15	2,587.00	**5.91**	2,588.00	**5.87**	2,588.00	**5.87**	0.08	8.06	5.58	5.42	5.07
10	**rec19**	30,10	2,874.00	**9.85**	2,884.50	**9.45**	2,895.00	**9.05**	3.50	7.25	9.15	8.50	6.46
11	**rec21**	30,10	2,827.00	**6.65**	2,827.72	**6.62**	2,828.00	**6.61**	0.06	8.49	5.70	5.33	4.74
12	**rec23**	30,10	2,707.00	**11.93**	2,723.50	**11.25**	2,740.00	**10.58**	7.64	8.67	10.80	9.72	8.65
13	**rec25**	30,15	3,593.00	**6.74**	3,595.00	**6.68**	3,597.00	**6.62**	1.26	11.98	5.71	5.17	4.25
14	**rec27**	30,15	3,434.00	**6.44**	3,447.00	**6.03**	3,460.00	**5.64**	12.35	11.78	6.13	5.04	4.13
15	**rec29**	30,15	3,291.00	**8.87**	3,296.00	**8.71**	3,301.00	**8.54**	7.55	11.53	7.81	6.93	5.69
16	**rec31**	50,10	4,336.00	**6.80**	4,342.50	**6.64**	4,349.00	**6.48**	6.59	13.65	5.92	5.20	4.51
17	**rec33**	50,10	4,484.00	**6.38**	4,491.00	**6.21**	4,498.00	**6.05**	23.45	13.72	5.51	4.08	3.17
18	**rec35**	50,10	4,441.00	**6.24**	4,454.50	**5.92**	4,468.00	**5.60**	8.45	13.50	6.02	5.13	3.98
19	**rec37**	75,20	8,163.00	**10.00**	8,194.50	**9.57**	8,226.00	**9.15**	14.48	78.06	8.89	8.20	7.40
20	**rec39**	75,20	8,593.00	6.58	8,618.00	**6.27**	8,643.00	**5.96**	25.86	77.36	**6.79**	5.67	4.26
21	**rec41**	75,20	8,627.00	**8.31**	8,642.50	**8.12**	8,658.00	**7.92**	16.31	80.45	7.94	6.77	5.91
Average	**N/A**	**N/A**	**N/A**	**7.37**	**N/A**	**6.75**	**N/A**	**6.48**	**8.56**	**18.50**	**6.35**	**5.65**	**4.67**

Table 4-12 Comparison of the Results of the Proposed algorithm with [30] and [72] for the Problems with unknown Optimal Solution

Problem Number	Problem Name	n, m	[4] Makespan	Proposed GA (BRE)	Proposed GA+PSO (BRE)	VNS	GASA	DS	DS+M	TS	TS+M	TS+MP
1	**rec01**	20,5	1,590.00	3.88	**13.98**	2.77	3.96	3.71	3.58	4.03	3.96	3.96
2	**rec03**	20,5	1,457.00	4.44	**7.05**	4.32	4.46	3.43	4.43	6.59	6.59	6.59
3	**rec05**	20,5	1,637.00	7.17	**8.12**	7.03	6.90	5.62	5.62	7.39	7.64	7.70
4	**rec07**	20,10	2,119.00	2.92	**3.77**	2.31	3.45	1.09	1.08	3.63	3.63	3.63
5	**rec09**	20,10	2,141.00	4.15	**4.85**	2.38	4.48	3.60	3.60	4.62	4.58	4.58
6	**rec11**	20,10	1,946.00	1.55	**3.46**	1.54	3.34	1.44	1.44	3.34	3.34	3.34
7	**rec13**	20,15	2,709.00	5.17	**6.44**	5.76	5.65	3.43	4.43	6.05	6.05	6.05
8	**rec15**	20,15	2,691.00	5.84	**6.41**	5.91	6.02	4.83	4.83	5.91	6.02	5.91
9	**rec17**	20,15	2,740.00	5.14	**5.91**	5.15	5.47	5.51	5.51	5.58	5.58	5.58
10	**rec19**	30,10	3,157.00	7.87	**9.85**	7.57	5.45	7.70	7.44	9.72	9.25	9.38
11	**rec21**	30,10	3,015.00	5.80	**6.65**	4.21	2.22	3.68	4.68	6.37	6.30	6.17
12	**rec23**	30,10	3,030.00	10.64	**11.93**	10.70	6.70	7.29	7.29	10.76	10.73	10.89
13	**rec25**	30,15	3,835.00	5.27	**6.74**	5.45	2.69	3.08	3.08	5.97	6.31	6.21
14	**rec27**	30,15	3,655.00	5.18	**6.44**	5.83	2.60	3.64	3.64	5.64	6.10	5.83
15	**rec29**	30,15	3,583.00	6.44	**8.87**	7.23	3.99	7.23	7.36	7.94	8.28	7.94
16	**rec31**	50,10	4,631.00	6.35	**6.80**	4.71	-2.72	3.76	3.78	5.90	6.13	6.22
17	**rec33**	50,10	4,770.00	3.85	**6.38**	5.35	-4.78	1.97	2.01	5.51	6.31	6.37
18	**rec35**	50,10	4,718.00	5.16	**6.24**	5.51	-3.67	4.94	4.94	6.08	6.17	5.91
19	**rec37**	75,20	8,979.00	9.50	**10.00**	10.00	-5.89	7.80	7.92	9.41	9.49	9.36
20	**rec39**	75,20	9,158.00	5.43	6.58	5.32	-8.80	4.97	5.12	**7.00**	6.99	6.91
21	**rec41**	75,20	9,344.00	7.46	8.31	7.41	-6.79	6.08	6.08	8.78	8.57	**8.82**
Average	**N/A**	**N/A**	**N/A**	**5.68**	**7.37**	**5.55**	**1.65**	**4.51**	**4.66**	**6.49**	**6.57**	**6.54**

4.5.2.2 Computational Results for the Problems with Setup Times

The sets of problems with and without known optimal solutions are considered since the problem sizes in the two sets differ significantly. Setup time for each operation is generated based on (4-28). For clarification purposes, the new problems are called Car1+S through Car8+S, and Rec01+S through Rec41+S. Since these problems are generated in this thesis and being solved for the first time, the results of the proposed GA and GA+PSO are compared with the results of the 2-Opt algorithm. For a permutation problem, 2-opt algorithm arbitrarily chooses two elements of the permutation and exchanges these two elements; objective function will be calculated and the exchange will be accepted if an improvement is noticed. The algorithm continues until no such improvement can be made.

Table 4-13 gives the computational results of the Car1+s through Car8+s problems; this table highlights the performance of the proposed algorithms compared to the 2-Opt algorithm.

Table 4-14 presents the computational results of the proposed GA for the Rec01+S through Rec41+S. The average row reveals that the difference between BRE, ARE, and WRE gaps is small. This shows that the proposed GA and GA+PSO are able to suggest consistent results. In addition, the average standard deviation, reflected in the STD column is another indicator of the consistency of the proposed methods. Moreover, comparing the 2-Opt algorithm results and the proposed algorithms' solutions concludes the effectiveness of the algorithm. Once again, the results of the GA+PSO outperform the results of the proposed GA.

4.6 No-Wait Flow Shop Problem with Separable Sequence Dependent Setup Time

4.6.1 Problem Description

(4-4) implies that once the starting time of o_{i1} is known, it is possible to calculate the starting time of $o_{ij}; j = 2,3,...,m$. Accordingly, it is possible to reduce the problem to finding the best time to start $o_{i1}; i = 1,2,...,n$. In other words, $F \mid no-wait, S_{sd} \mid C_{\max}$ can be reduced to a permutation problem. In other words, it is possible to reduce $F \mid no-wait, S_{sd} \mid C_{\max}$ to ATSP.

The proposed PSO exploits the above fact and searches the set of permutations of a problem instance in order to find the permutation that minimizes the makespan of that instance. Herein, the Makespan Calculation Algorithm (MCA) that calculates the makespan of a given permutation is presented as follows:

1. $S_{o_{11}} = ST_{110}$.
2. For $k = 2$ to m, $S_{O_{1[k]}} = \max\{ST_{1[k]0}, S_{O_{1k}} + p_{1k}\}$. If $ST_{1[k]0} > S_{O_{1k}} + p_{1k}$, set $d = ST_{1[k]0} - (S_{O_{1k}} + p_{1k})$, and for $h = 1,2,...,k-1$, set $S_{O_{1h}} \leftarrow S_{O_{1h}} + d$.
3. Set $i \leftarrow 1; j \leftarrow 1$.
4. $S_{o_{[i]j}} = S_{o_{ij}} + p_{ij} + ST_{[i]ji}$.
5. $j \leftarrow j + 1$.
6. $S_{O_{[i]j}} = \max\{S_{o_{ij}} + p_{ij} + ST_{[i]ji}, S_{O_{i],(j-1)}} + p_{[i],(j-1)}\}$. If $S_{o_{ij}} + p_{ij} + ST_{[i]ji} > S_{O_{i],(j-1)}} + p_{[i],(j-1)}$, set $d = S_{o_{ij}} + p_{ij} + ST_{[i]ji} - (S_{O_{i],(j-1)}} + p_{[i],(j-1)})$, and for $h = 1,2,..., j-1$, set $S_{O_{[i]h}} \leftarrow S_{O_{[i]h}} + d$.
7. If $i = n$, stop. $C_{\max} = S_{o_{nm}} + p_{nm}$. Otherwise, set $i \leftarrow i + 1$ and $j = 1$. Go back to step 4.

The algorithm starts with a sequence of jobs or equivalently, a permutation. It determines the starting time of the first operation of the first job in the given permutation. Then the algorithm considers the sequence dependent setup times and imposes the no-wait constraints, while calculating the starting time of the rest of the operations of that job. When scheduling the first job in the sequence is finished, using the same method, the algorithm schedules the next job in the sequence, and continues until the scheduling is finished. Computational complexity of this algorithm is $O(mn)$.

Table 4-13 Computational Results of the Problems Car1+S through Car8+S

Prob. No.	n, m	2-Opt OFV*	Proposed GA						
			Best Makespan		Average Makespan		Worst Makespan		Average CPU Time
			OFV*	Gap** (%)	OFV*	Gap** (%)	OFV*	Gap** (%)	
car1+S	**11,5**	14,423.00	12,313.00	17.14	12,320.50	17.07	12,343.00	16.85	1.76
car2+S	**13,4**	16,917.00	12,786.00	32.31	12,831.80	31.84	13,073.00	29.40	1.82
car3+S	**12,5**	16,453.00	12,443.00	32.23	12,468.95	31.95	12,616.00	30.41	1.79
car4+S	**14,4**	17,461.00	13,637.00	28.04	13,637.75	28.03	13,647.00	27.95	1.89
car5+S	**10,6**	18,014.00	13,128.00	37.22	13,219.40	36.27	13,650.00	31.97	1.68
car6+S	**8,9**	15,531.00	13,233.00	17.37	13,315.50	16.64	13,896.00	11.77	1.32
car7+S	**7,7**	11,765.00	11,249.00	4.59	11,275.60	4.34	11,330.00	3.84	1.30
car8+S	**8,8**	13,776.00	11,902.00	15.75	11,912.80	15.64	11,938.00	15.40	1.45
Average		**N/A**	**N/A**	**23.08**	**N/A**	**22.72**	**N/A**	**20.95**	**1.63**

Table 4-13 (Continued)

Prob. No.	n, m	2-Opt OFV*	Proposed GA+PSO						
			Best Makespan		Average Makespan		Worst Makespan		Average CPU Time
			OFV*	Gap** (%)	OFV*	Gap** (%)	OFV*	Gap** (%)	
car1+S	**11,5**	14,423.00	11,784.00	22.39	11,808.65	22.14	11,947.00	20.72	2.46
car2+S	**13,4**	16,917.00	11,786.00	43.53	11,795.90	43.41	11,826.00	43.05	2.55
car3+S	**12,5**	16,453.00	12,443.00	32.23	12,494.95	31.68	12,617.00	30.40	2.51
car4+S	**14,4**	17,461.00	13,535.00	29.01	13,648.30	27.94	13,838.00	26.18	2.65
car5+S	**10,6**	18,014.00	13,128.00	37.22	13,128.00	37.22	13,128.00	37.22	2.35
car6+S	**8,9**	15,531.00	13,233.00	17.37	13,233.00	17.37	13,233.00	17.37	1.85
car7+S	**7,7**	11,765.00	10,290.00	14.33	10,290.00	14.33	10,290.00	14.33	1.82
car8+S	**8,8**	13,776.00	11,902.00	15.75	11,902.00	15.75	11,902.00	15.75	2.03
Average		**N/A**	**N/A**	**26.48**	**N/A**	**26.23**	**N/A**	**25.63**	**2.28**

* Objective Function Value

** Gap is between the algorithm's OFV and 2-Opt algorithm

Table 4-14 Computational Results of the Problems Rec01+S through Rec41+S

Prob. No.	Prob. Name	n, m	2-Opt OFV*	Proposed GA							
				OFV*							Average CPU Time
				Best Makespan	Gap** (%)	Average Makespan	Gap** (%)	Worst Makespan	Gap** (%)	STD	
1	**rec01+S**	20,5	2,721.00	2,166.00	25.62	2,175.95	25.05	2,222.00	22.46	3.99	20.57
2	**rec03+S**	20,5	2,625.00	2,033.00	29.12	2,041.40	28.59	2,070.00	26.81	5.45	20.59
3	**rec05+S**	20,5	2,706.00	2,136.00	26.69	2,146.60	26.06	2,159.00	25.34	8.53	20.51
4	**rec07+S**	20,10	3,729.00	2,795.00	33.42	2,801.10	33.13	2,836.00	31.49	12.29	21.34
5	**rec09+S**	20,10	3,423.00	2,941.00	16.39	2,946.95	16.15	2,985.00	14.67	7.61	21.35
6	**rec11+S**	20,10	3,599.00	2,553.00	40.97	2,574.40	39.80	2,605.00	38.16	5.59	21.21
7	**rec13+S**	20,15	4,431.00	3,418.00	29.64	3,435.05	28.99	3,476.00	27.47	20.54	22.26
8	**rec15+S**	20,15	4,193.00	3,358.00	24.87	3,371.05	24.38	3,438.00	21.96	24.77	22.04
9	**rec17+S**	20,15	4,435.00	3,372.00	31.52	3,389.95	30.83	3,424.00	29.53	7.60	22.26
10	**rec19+S**	30,10	5,665.00	4,247.00	33.39	4,276.40	32.47	4,352.00	30.17	12.36	25.43
11	**rec21+S**	30,10	5,171.00	3,964.00	30.45	3,986.75	29.70	4,026.00	28.44	14.82	25.58
12	**rec23+S**	30,10	5,224.00	3,886.00	34.43	3,907.40	33.70	3,931.00	32.89	10.01	25.43
13	**rec25+S**	30,15	6,387.00	4,799.00	33.09	4,825.10	32.37	4,863.00	31.34	6.80	26.81
14	**rec27+S**	30,15	6,401.00	4,611.00	38.82	4,634.25	38.12	4,704.00	36.08	32.62	26.95
15	**rec29+S**	30,15	6,681.00	4,624.00	44.49	4,647.30	43.76	4,697.00	42.24	39.86	26.78
16	**rec31+S**	50,10	8,494.00	6,176.00	37.53	6,204.30	36.91	6,210.00	36.78	29.30	37.57
17	**rec33+S**	50,10	9,278.00	6,467.00	43.47	6,503.80	42.66	6,662.00	39.27	62.86	37.61
18	**rec35+S**	50,10	8,969.00	6,617.00	35.54	6,638.70	35.10	6,602.00	35.85	26.94	37.62
19	**rec37+S**	75,20	16,227.00	11,420.00	42.09	11,493.80	41.18	11,516.00	40.91	33.80	87.64
20	**rec39+S**	75,20	16,896.00	11,741.00	43.91	11,792.90	43.27	11,909.00	41.88	98.37	88.10
21	**rec41+S**	75,20	16,749.00	11,636.00	43.94	11,668.55	43.54	11,768.00	42.33	63.71	88.42
Average		**N/A**	**N/A**	**N/A**	**34.26**	**N/A**	**33.61**	**N/A**	**32.19**	**25.14**	**34.57**

Table 4-14 (Continued)

Prob. No.	Prob. Name	n, m	2-Opt OFV*	Proposed GA+PSO							
				OFV*							Average CPU Time
				Best Makespan	Gap** (%)	Average Makespan	Gap** (%)	Worst Makespan	Gap** (%)	STD	
1	**rec01+S**	20,5	2,721.00	2,161.00	25.91	2,167.85	25.52	2,177.00	24.99	13.81	28.80
2	**rec03+S**	20,5	2,625.00	2,020.00	29.95	2,034.00	29.06	2,056.00	27.68	13.59	28.83
3	**rec05+S**	20,5	2,706.00	2,133.00	26.86	2,143.25	26.26	2,151.00	25.80	3.35	28.71
4	**rec07+S**	20,10	3,729.00	2,789.00	33.70	2,796.85	33.33	2,832.00	31.67	8.27	29.87
5	**rec09+S**	20,10	3,423.00	2,922.00	17.15	2,931.15	16.78	2,961.00	15.60	14.02	29.88
6	**rec11+S**	20,10	3,599.00	2,552.00	41.03	2,567.50	40.18	2,585.00	39.23	17.78	29.69
7	**rec13+S**	20,15	4,431.00	3,417.00	29.68	3,432.20	29.10	3,452.00	28.36	11.87	31.16
8	**rec15+S**	20,15	4,193.00	3,337.00	25.65	3,356.10	24.94	3,377.00	24.16	13.20	30.86
9	**rec17+S**	20,15	4,435.00	3,372.00	31.52	3,385.65	30.99	3,402.00	30.36	13.25	31.16
10	**rec19+S**	30,10	5,665.00	4,222.00	34.18	4,261.00	32.95	4,295.00	31.90	31.99	35.60
11	**rec21+S**	30,10	5,171.00	3,939.00	31.28	3,986.60	29.71	4,014.00	28.82	34.00	35.81
12	**rec23+S**	30,10	5,224.00	3,838.00	36.11	3,850.00	35.69	3,869.00	35.02	12.79	35.60
13	**rec25+S**	30,15	6,387.00	4,774.00	33.79	4,781.45	33.58	4,792.00	33.28	16.63	37.53
14	**rec27+S**	30,15	6,401.00	4,580.00	39.76	4,622.30	38.48	4,681.00	36.74	20.03	37.73
15	**rec29+S**	30,15	6,681.00	4,562.00	46.45	4,592.35	45.48	4,680.00	42.76	20.21	37.49
16	**rec31+S**	50,10	8,494.00	6,115.00	38.90	6,156.15	37.98	6,273.00	35.41	28.92	52.60
17	**rec33+S**	50,10	9,278.00	6,418.00	44.56	6,460.80	43.60	6,581.00	40.98	32.60	52.65
18	**rec35+S**	50,10	8,969.00	6,507.00	37.84	6,526.85	37.42	6,652.00	34.83	12.53	52.67
19	**rec37+S**	75,20	16,227.00	11,391.00	42.45	11,440.70	41.84	11,598.00	39.91	48.55	122.70
20	**rec39+S**	75,20	16,896.00	11,581.00	45.89	11,674.60	44.72	11,859.00	42.47	29.71	123.34
21	**rec41+S**	75,20	16,749.00	11,537.00	45.18	11,620.10	44.14	11,789.00	42.07	33.98	123.79
Average		**N/A**	**N/A**	**N/A**	**35.14**	**N/A**	**34.37**	**N/A**	**32.96**	**20.53**	**48.40**

*Objective Function Value

**Gap is between the algorithm's OFV and 2-Opt algorithm

4.6.2 The Proposed PSO

PSO algorithm has been a widely used metaheuristic since its introduction in [110, 111]. In PSO, at time t, each particle i has a position, $x_i(t)$, and a velocity, $v_i(t)$. PSO stores the current position of the particles as well as the best ever position; historical and random information are used in order to update velocities during the algorithm's iterations; using the modified velocities, particles' positions are updated. The proposed PSO uses a specific coding system to map permutations of jobs into representations that PSO is able to work with. In this research, this coding scheme is referred to as Matrix Coding (MC). MC is previously introduced and applied to a layout problem in [118].

After generating a number of initial solutions, the proposed PSO stores the local and global best positions; afterwards, PSO generates the velocity vectors. At this point, job permutations are transformed into MCs. Each MC represents a particle. PSO moves the particles based on the generated velocity vectors; such transfers are carried out according to a probabilistic approach. Once all the transfers are completed, PSO transforms the new MCs into their respective permutations and the objective function of the new permutations is calculated. The algorithm then updates the values of the local and global bests and proceeds with next iterations. The proposed PSO uses the MCA algorithm of section 4.6.1 to calculate the objective function values of the generated permutations.

4.6.2.1 Initial Solutions and Matrix Coding

Generating P initial solutions or equivalently particles initiates the algorithm. P is a parameter of the algorithm, which is set by the user. Once the particles are generated, PSO calculates their objective function values (makespans) using the MCA algorithm of section 4.6.1. The best objective function value along with the

corresponding permutation will be assigned to the global best variable or g. Since the particles have not moved in the feasible region yet, every particle is considered to be at its local best position or b.

Then, each permutation is transformed into its corresponding MC. MCs are the appropriate representation of permutations, with which PSO is able to function. Suppose that at iteration k, permutation $(1,2,3,...,N)$ represents particle l. MC representation of this permutation is the orthogonal $N \times N$ matrix (4-29).

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \tag{4-29}$$

Every element of this matrix is represented by x_{klij}, in which k is the iteration number, l is the particle number, i is the column number, and j is the row number. MC is such that:

$$\begin{aligned} &\sum_{i=1}^{n} x_{klij} = 1; \forall j \\ &\sum_{j=1}^{n} x_{klij} = 1; \forall i \end{aligned} \tag{4-30}$$

In addition, $x_{klij} = 1$ means that in iteration k, job i is placed in the j th position of the permutation l. For instance, the MC of the permutation $(2,1,3)$ is as follows:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4-31}$$

Once the algorithm codes the initial solutions using the MCs, velocity vectors are generated in order to update the position of the particles and transfer them to the neighborhood solutions.

4.6.2.2 Velocity Vectors and Neighborhood Structure

Suppose that the algorithm is in its k th iteration. For each particle $l = 1,2,...,P$, velocity vectors, represented by $\vec{v}_{kl}$, are updated using (4-32).

$$\vec{v}_{kl} \leftarrow \vec{v}_{(k-1)l} + U(0,\phi_1)\left(\vec{b}_l - \vec{x}_{kl}\right) + U(0,\phi_2)\left(\vec{g} - \vec{x}_{kl}\right) \tag{4-32}$$

In which, $U(0,\phi_i); i = 1,2$ is uniform probability distribution with parameters 0 and ϕ_i. $\phi_i; i = 1,2$ is a parameter, set by the user. b_l is the MC of the local best position of the particle l. g represents the MC of the global best position. x_{kl} is the MC of the particle l in iteration k. In order to limit the values of velocity vectors, and encourage the PSO to search the promising areas more thoroughly, the following checks will be done:

$$\begin{aligned} &\vec{v}_{kl} \leftarrow \min\{\vec{v}_{kl}, 15\} \\ &\vec{v}_{kl} \leftarrow \max\{\vec{v}_{kl}, -15\} \end{aligned} \tag{4-33}$$

It should be noted that $\vec{v}_{1l} = 1; \forall l$. Once the velocity vectors are updated, the algorithm is ready to move the particles to their new positions. First the algorithm considers the set of all available places in the iteration as $A = \{1,2,...,N\}$ and sets $x_{klij} = 0; \forall i, j$. PSO uses (4-34) in order to calculate the probability of setting $x_{klij} = 1; j = 1$.

$$E_{klij} = \frac{(1+e^{-v_{klij}})^{-1}}{\sum_{i \in A}(1+e^{-v_{klij}})^{-1}}; \forall i \tag{4-34}$$

If the algorithm chooses that $x_{klij}=1; j=1$, the set of available places will be updated as $A \leftarrow \{1,2,...,N\}-\{i\}$; the same procedure will be repeated for $j=2,3,...,N$, until the position of particle l is updated. At this point, PSO calculates the objective function values of the new particles and updates the values of g and b_l if necessary; and the search will continue.

4.6.2.3 Illustrative Example

As an illustrative example, suppose that the matrix of v_{klij} is given as follows:

$$\begin{bmatrix} 1 & 2 & 1 \\ 1 & 3 & -3 \\ 1 & 5 & -12 \end{bmatrix} \tag{4-35}$$

Using (4-34) to calculate the values of $E_{klij}; j=1$ will result in:

$$(0.31, 0.37, 0.31) \tag{4-36}$$

Suppose that the algorithm chooses to set $x_{kl21}=1$. The partial MC would be:

$$\begin{bmatrix} 0 & 1 & 0 \\ & 0 & \\ & 0 & \end{bmatrix} \tag{4-37}$$

And the list of available places should be updated to $A=\{1,3\}$. The algorithm sets $j \leftarrow 2$, and the values of $E_{klij}; j=2$ will be $(0.94, 0.06)$. Suppose that the algorithm sets $x_{kl12}=1$. The partial MC is as follows:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & \end{bmatrix} \tag{4-38}$$

Since there is only one place left in MC, the complete MC would be:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4\text{-}39)$$

Which is equal to permutation $(2,1,3)$. The proposed PSO stops after I iterations. I is a parameter, set by the user. Next section presents the computational results.

4.6.3 Computational Results

As seen in section 4.6.2, the developed PSO has 4 parameters that must be tuned to obtain the best performance from the algorithm. The following experimentally derived values are proposed for the PSO parameters:

$$\begin{aligned} &P = n \\ &\phi_i = 1; i = 1,2 \\ &I = 100.n \end{aligned} \quad (4\text{-}40)$$

The developed PSO is coded using Microsoft Visual C++ 2008; all the test problem instances are solved on a PC equipped with a 3 GHz Intel Pentium IV CPU and 2 GB of RAM. As mentioned before, the proposed PSO is applied to test problems of $F \mid no-wait \mid C_{\max}$, $F \mid no-wait, setup \mid C_{\max}$, and $F \mid no-wait, S_{sd} \mid C_{\max}$.

4.6.3.1 Computational Results for $F \mid no-wait \mid C_{\max}$ Test Problems

The same set of test problems of sections 4.4.3 and 4.5.2.1 are chosen to test the efficiency of the proposed PSO compared to the previous algorithms developed for $F \mid no-wait \mid C_{\max}$. Table 4-15 compares the computational results of the proposed algorithms with those of [33] and [4].

Table 4-15 demonstrates that the proposed PSO improves the best-known solution of 14 out of 21 test problems (66.67%), and for the rest of the problems, the proposed solutions of the developed PSO are never inferior to the solutions of [33]. Moreover, in terms of ARE and WRE, the average performance of the proposed PSO is

better than the HPSO of [33] in 20 and 19 cases respectively. It should be noted that the proposed PSO improves all the solutions proposed by [4], up to 12.26 percent in some cases. The proposed PSO takes very short time to solve large test cases. For example, for problems with 75 jobs and 20 machines, the algorithm uses slightly more than 1 second from the CPU time, which is a considerable improvement from the algorithms of the section 4.4.2 and 4.5.2. Small standard deviation (STD) values are a sign of the consistency of the proposed algorithm. One can verify that the proposed algorithm is able to re-generate all the ARE values of section 4.5.2.1.

4.6.3.2 Computational Results for $F \mid no-wait, setup \mid C_{\max}$ Test Problems

29 test cases of $F \mid no-wait, setup \mid C_{\max}$ were generated in section 4.5.2.2: car1+S through car8+S and rec01+S through rec41+S. Since $F \mid no-wait, setup \mid C_{\max}$ is a special case of $F \mid no-wait, S_{sd} \mid C_{\max}$, the proposed PSO can be applied to these test cases. In section 4.5.2.2, these problems were solved using 2-Opt algorithm as well as GA and GA+PSO. Table 4-16 and Table 4-17 compare the results of section 4.5.2.2 with the solutions proposed by the developed PSO for car1+S through car8+S and rec01+S through rec41+S respectively. For the case of rec+S problems, the developed PSO improves all the solutions proposed by GA and 14 out of 21 solutions proposed by GA+PSO. Since both algorithms use the same platform to conduct the computational results, it is possible to compare the CPU times of the proposed PSO with GA and GA+PSO. While average CPU time of the developed algorithm for rec+S problems is 0.58 seconds, the average CPU times of GA and GA+PSO are 34.57 and 48.40 seconds respectively. In addition, the average standard deviation, reflected in the STD column is another indicator of the consistency of the proposed PSO. In other words, the proposed PSO outperforms the competitive methods.

Table 4-15 Comparison of the Results of the Proposed PSO with [33] for the Test Problems without Setup Times

Prob. Number	Prob. Name	n, m	[4] Makespan	Proposed PSO							[33]		
				Objective Function Value and The percentage of Improvement Over the Makespan of [4]						Average CPU Time (Second)	Objective Function Value		
				BRE		ARE		WRE			BRE	ARE	WRE
1	**rec01**	20,5	1590	**1,395.00**	**12.26**	**1,512.00**	**4.91**	**1,534.00**	**3.52**	105.64	0.19	3.77	3.39	2.96
2	**rec03**	20,5	1457	**1,361.00**	**6.59**	**1,361.00**	**6.59**	**1,361.00**	**6.59**	0.00	0.18	**6.59**	6.15	3.36
3	**rec05**	20,5	1637	**1,511.00**	**7.70**	**1,519.30**	**7.19**	1,533.00	6.35	9.12	0.23	7.39	7.15	**6.66**
4	**rec07**	20,10	2119	**2,042.00**	**3.63**	**2,054.00**	**3.07**	**2,066.00**	**2.50**	9.46	0.27	**3.63**	3.11	2.31
5	**rec09**	20,10	2141	**2,027.00**	**5.32**	**2,034.50**	**4.97**	**2,042.00**	**4.62**	11.09	0.09	4.58	4.26	3.60
6	**rec11**	20,10	1946	**1,881.00**	**3.34**	**1,884.54**	**3.16**	**1,894.00**	**2.67**	5.50	0.26	**3.34**	2.30	1.28
7	**rec13**	20,15	2709	**2,545.00**	**6.05**	**2,558.70**	**5.55**	**2,578.00**	**4.84**	28.18	0.36	**6.05**	5.47	4.80
8	**rec15**	20,15	2691	**2,529.00**	**6.02**	**2,529.00**	**6.02**	**2,529.00**	**6.02**	0.00	0.34	**6.02**	5.69	4.91
9	**rec17**	20,15	2740	**2,587.00**	**5.58**	**2,587.95**	**5.55**	**2,588.00**	**5.55**	0.22	0.37	**5.58**	5.42	5.07
10	**rec19**	30,10	3157	**2,861.00**	**9.38**	**2,867.50**	**9.17**	**2,874.00**	**8.96**	6.67	0.40	9.15	8.50	6.46
11	**rec21**	30,10	3015	**2,822.00**	**6.40**	**2,825.00**	**6.30**	**2,828.00**	**6.20**	3.08	0.40	5.70	5.33	4.74
12	**rec23**	30,10	3030	**2,700.00**	**10.89**	2,735.75	9.71	**2,751.00**	**9.21**	28.74	0.26	10.80	**9.72**	8.65
13	**rec25**	30,15	3835	**3,593.00**	**6.31**	**3,593.00**	**6.31**	**3,593.00**	**6.31**	0.00	0.35	5.71	5.17	4.25
14	**rec27**	30,15	3655	**3,431.00**	**6.13**	**3,464.65**	**5.21**	**3,504.00**	**4.13**	34.16	0.35	**6.13**	5.04	**4.13**
15	**rec29**	30,15	3583	**3,291.00**	**8.15**	**3,306.50**	**7.72**	**3,312.00**	**7.56**	8.73	0.52	7.81	6.93	5.69
16	**rec31**	50,10	4631	**4,336.00**	**6.37**	**4,343.80**	**6.20**	**4,349.00**	**6.09**	6.53	0.60	5.92	5.20	4.51
17	**rec33**	50,10	4770	**4,466.00**	**6.37**	**4,510.00**	**5.45**	**4,522.00**	**5.20**	45.04	0.62	5.51	4.08	3.17
18	**rec35**	50,10	4718	**4,417.00**	**6.38**	**4,455.00**	**5.57**	**4,468.00**	**5.30**	18.45	0.61	6.02	5.13	3.98
19	**rec37**	75,20	8979	**8,081.00**	**10.00**	**8,180.50**	**8.89**	**8,280.00**	**7.78**	61.70	2.08	8.89	8.20	7.40
20	**rec39**	75,20	9158	**8,517.00**	**7.00**	**8,555.00**	**6.58**	**8,593.00**	**6.17**	80.37	1.49	**6.79**	5.67	4.26
21	**rec41**	75,20	9344	**8,520.00**	**8.82**	**8,573.50**	**8.25**	**8,627.00**	**7.67**	100.32	1.54	7.94	6.77	5.91
Average	**N/A**	**N/A**	**N/A**	**N/A**	**7.08**	**N/A**	**6.30**	**N/A**	**5.87**	**26.81**	**0.55**	**6.35**	**5.65**	**4.67**

Table 4-16 Computational Results of the Problems Car1+S through Car8+S

			GA						
	n, m	2-Opt OFV*	OFV*						CPU Time (second)
			Best Makespan	Gap** (%)	Average Makespan	Gap** (%)	Worst Makespan	Gap** (%)	
car1+S	**11,5**	14,423.00	12,313.00	14.63	12,320.50	14.58	12,343.00	14.42	1.76
car2+S	**13,4**	16,917.00	12,786.00	24.42	12,831.80	24.15	13,073.00	22.72	1.82
car3+S	**12,5**	16,453.00	12,443.00	24.37	12,468.95	24.21	12,616.00	23.32	1.79
car4+S	**14,4**	17,461.00	13,637.00	21.90	13,637.75	21.90	13,647.00	21.84	1.89
car5+S	**10,6**	18,014.00	**13,128.00**	**27.12**	13,219.40	26.62	13,650.00	24.23	1.68
car6+S	**8,9**	15,531.00	**13,233.00**	**14.80**	13,315.50	14.27	13,896.00	10.53	1.32
car7+S	**7,7**	11,765.00	11,249.00	4.39	11,275.60	4.16	11,330.00	3.70	1.30
car8+S	**8,8**	13,776.00	**11,902.00**	**13.60**	11,912.80	13.52	11,938.00	13.34	1.45
Average		---	---		---	**17.93**	---	**16.76**	**1.63**

Table 4-16 (Continued)

			GA+PSO						
	n, m	2-Opt OFV*	OFV*						CPU Time (second)
			Best Makespan	Gap** (%)	Average Makespan	Gap** (%)	Worst Makespan	Gap** (%)	
car1+S	**11,5**	14,423.00	**11,784.00**	**18.30**	**11,808.65**	**18.13**	**11,947.00**	**17.17**	2.46
car2+S	**13,4**	16,917.00	**11,786.00**	**30.33**	11,795.90	30.27	11,826.00	30.09	2.55
car3+S	**12,5**	16,453.00	**12,443.00**	**24.37**	12,494.95	24.06	12,617.00	23.31	2.51
car4+S	**14,4**	17,461.00	**13,535.00**	**22.48**	13,648.30	21.84	13,838.00	20.75	2.65
car5+S	**10,6**	18,014.00	**13,128.00**	**27.12**	**13,128.00**	**27.12**	**13,128.00**	**27.12**	2.35
car6+S	**8,9**	15,531.00	**13,233.00**	**14.80**	**13,233.00**	**14.80**	**13,233.00**	**14.80**	1.85
car7+S	**7,7**	11,765.00	10,290.00	12.54	10,290.00	12.54	**10,290.00**	**12.54**	1.82
car8+S	**8,8**	13,776.00	**11,902.00**	**13.60**	**11,902.00**	**13.60**	**11,902.00**	**13.60**	2.03
Average		---	---	**20.44**	---	**20.30**	---	**19.92**	**2.27**

Table 4-16 (Continued)

	n, m	2-Opt OFV*	Proposed PSO						
			OFV*						CPU Time (second)
			Best Makespan	Gap** (%)	Average Makespan	Gap** (%)	Worst Makespan	Gap** (%)	
car1+S	**11,5**	14,423.00	**11,784.00**	**18.30**	11,920.85	17.35	12,434.00	13.79	0.33
car2+S	**13,4**	16,917.00	**11,786.00**	**30.33**	**11,793.20**	**30.29**	**11,814.00**	**30.16**	0.16
car3+S	**12,5**	16,453.00	**12,443.00**	**24.37**	**12,443.60**	**24.37**	**12,455.00**	**24.30**	0.13
car4+S	**14,4**	17,461.00	**13,535.00**	**22.48**	**13,600.50**	**22.11**	**13,692.00**	**21.59**	0.14
car5+S	**10,6**	18,014.00	**13,128.00**	**27.12**	13,142.20	27.04	13,199.00	26.73	0.15
car6+S	**8,9**	15,531.00	**13,233.00**	**14.80**	13,463.75	13.31	13,941.00	10.24	0.21
car7+S	**7,7**	11,765.00	**10,249.00**	**12.89**	**10,279.75**	**12.62**	**10,290.00**	**12.54**	0.15
car8+S	**8,8**	13,776.00	**11,902.00**	**13.60**	**11,902.00**	**13.60**	**11,902.00**	**13.60**	0.19
Average		**---**	**---**	**20.49**	**---**	**20.09**	**---**	**19.12**	**0.18**

* Objective Function Value

** Gap is between the algorithms' OFV and 2-Opt algorithm's OFV

Table 4-17 Computational Results of the Problems Rec01+S through Rec41+S

			GA							
			OFV*							CPU Time (second)
	n, m	2-Opt OFV*	Best Makespan	Gap** (%)	Average Makespan	Gap** (%)	Worst Makespan	Gap** (%)	STD	
rec01+S	20,5	2,721.00	2,166.00	20.40	2,175.95	20.03	2,222.00	18.34	3.99	20.57
rec03+S	20,5	2,625.00	2,033.00	22.55	2,041.40	22.23	2,070.00	21.14	5.45	20.59
rec05+S	20,5	2,706.00	2,136.00	21.06	2,146.60	20.67	2,159.00	20.21	8.53	20.51
rec07+S	20,10	3,729.00	2,795.00	25.05	2,801.10	24.88	2,836.00	23.95	12.29	21.34
rec09+S	20,10	3,423.00	2,941.00	14.08	2,946.95	13.91	2,985.00	12.80	7.61	21.35
rec11+S	20,10	3,599.00	2,553.00	29.06	2,574.40	28.47	2,605.00	27.62	5.59	21.21
rec13+S	20,15	4,431.00	3,418.00	22.86	3,435.05	22.48	3,476.00	21.55	20.54	22.26
rec15+S	20,15	4,193.00	3,358.00	19.91	3,371.05	19.60	3,438.00	18.01	24.77	22.04
rec17+S	20,15	4,435.00	3,372.00	23.97	3,389.95	23.56	3,424.00	22.80	7.60	22.26
rec19+S	30,10	5,665.00	4,247.00	25.03	4,276.40	24.51	4,352.00	23.18	12.36	25.43
rec21+S	30,10	5,171.00	3,964.00	23.34	3,986.75	22.90	4,026.00	22.14	14.82	25.58
rec23+S	30,10	5,224.00	3,886.00	25.61	3,907.40	25.20	3,931.00	24.75	10.01	25.43
rec25+S	30,15	6,387.00	4,799.00	24.86	4,825.10	24.45	4,863.00	23.86	6.80	26.81
rec27+S	30,15	6,401.00	4,611.00	27.96	4,634.25	27.60	4,704.00	26.51	32.62	26.95
rec29+S	30,15	6,681.00	4,624.00	30.79	4,647.30	30.44	4,697.00	29.70	39.86	26.78
rec31+S	50,10	8,494.00	6,176.00	27.29	6,204.30	26.96	6,210.00	26.89	29.30	37.57
rec33+S	50,10	9,278.00	6,467.00	30.30	6,503.80	29.90	6,662.00	28.20	62.86	37.61
rec35+S	50,10	8,969.00	6,617.00	26.22	6,638.70	25.98	6,602.00	26.39	26.94	37.62
rec37+S	75,20	16,227.00	11,420.00	29.62	11,493.80	29.17	11,516.00	29.03	33.80	87.64
rec39+S	75,20	16,896.00	11,741.00	30.51	11,792.90	30.20	11,909.00	29.52	98.37	88.10
rec41+S	75,20	16,749.00	11,636.00	30.53	11,668.55	30.33	11,768.00	29.74	63.71	88.42
Average	**---**	**---**	**---**	**25.29**	**---**	**24.93**	**---**	**24.11**	**25.14**	**34.57**

Table 4-17 (Continued)

	n, m	2-Opt OFV*	GA+PSO							
			OFV*							CPU Time (second)
			Best Makespan	Gap** (%)	Average Makespan	Gap** (%)	Worst Makespan	Gap** (%)	STD	
rec01+S	20,5	2,721.00	2,161.00	20.58	2,167.85	20.33	2,177.00	19.99	10.77	28.80
rec03+S	20,5	2,625.00	2,020.00	23.05	2,034.00	22.51	2,056.00	21.68	14.72	28.83
rec05+S	20,5	2,706.00	2,133.00	21.18	**2,143.25**	**20.80**	**2,151.00**	**20.51**	23.04	28.71
rec07+S	20,10	3,729.00	**2,789.00**	**25.21**	2,796.85	25.00	**2,832.00**	**24.05**	33.18	29.87
rec09+S	20,10	3,423.00	**2,922.00**	**14.64**	2,931.15	14.37	**2,961.00**	**13.50**	20.53	29.88
rec11+S	20,10	3,599.00	**2,552.00**	**29.09**	2,567.50	28.66	2,585.00	28.17	15.08	29.69
rec13+S	20,15	4,431.00	3,417.00	22.88	3,432.20	22.54	**3,452.00**	**22.09**	55.47	31.16
rec15+S	20,15	4,193.00	3,337.00	20.41	3,356.10	19.96	3,377.00	19.46	66.88	30.86
rec17+S	20,15	4,435.00	3,372.00	23.97	3,385.65	23.66	3,402.00	23.29	20.53	31.16
rec19+S	30,10	5,665.00	4,222.00	25.47	4,261.00	24.78	**4,295.00**	**24.18**	33.37	35.60
rec21+S	30,10	5,171.00	3,939.00	23.83	3,986.60	22.90	4,014.00	22.37	40.02	35.81
rec23+S	30,10	5,224.00	**3,838.00**	**26.53**	3,850.00	26.30	**3,869.00**	**25.94**	27.03	35.60
rec25+S	30,15	6,387.00	**4,774.00**	**25.25**	**4,781.45**	**25.14**	4,792.00	24.97	18.36	37.53
rec27+S	30,15	6,401.00	4,580.00	28.45	4,622.30	27.79	**4,681.00**	**26.87**	88.08	37.73
rec29+S	30,15	6,681.00	**4,562.00**	**31.72**	**4,592.35**	**31.26**	4,680.00	29.95	107.62	37.49
rec31+S	50,10	8,494.00	6,115.00	28.01	6,156.15	27.52	**6,273.00**	**26.15**	79.10	52.60
rec33+S	50,10	9,278.00	6,418.00	30.83	6,460.80	30.36	6,581.00	29.07	169.73	52.65
rec35+S	50,10	8,969.00	6,507.00	27.45	**6,526.85**	**27.23**	6,652.00	25.83	72.74	52.67
rec37+S	75,20	16,227.00	11,391.00	29.80	**11,440.70**	**29.50**	11,598.00	28.53	91.26	122.70
rec39+S	75,20	16,896.00	**11,581.00**	**31.46**	11,674.60	30.90	11,859.00	29.81	265.61	123.34
rec41+S	75,20	16,749.00	11,537.00	31.12	11,620.10	30.62	11,789.00	29.61	172.01	123.79
Average	**---**	**---**	**---**	**25.76**	**---**	**25.34**	**---**	**24.57**	**67.86**	**48.40**

Table 4-17 Continued

	n, m	2-Opt OFV*	Proposed PSO							
			OFV*							CPU Time (second)
			Best Makespan	Gap** (%)	Average Makespan	Gap** (%)	Worst Makespan	Gap** (%)	STD	
rec01+S	20,5	2,721.00	**2,147.00**	**21.10**	**2,149.35**	**21.01**	**2,165.00**	**20.43**	17.25	0.20
rec03+S	20,5	2,625.00	**2,017.00**	**23.16**	**2,029.75**	**22.68**	**2,037.00**	**22.40**	15.98	0.19
rec05+S	20,5	2,706.00	**2,129.00**	**21.32**	2,150.27	20.54	2,157.00	20.29	13.70	0.24
rec07+S	20,10	3,729.00	**2,789.00**	**25.21**	**2,795.65**	**25.03**	2,838.00	23.89	11.75	0.29
rec09+S	20,10	3,423.00	**2,922.00**	**14.64**	**2,930.30**	**14.39**	2,968.00	13.29	10.71	0.10
rec11+S	20,10	3,599.00	**2,552.00**	**29.09**	**2,552.35**	**29.08**	**2,553.00**	**29.06**	0.49	0.28
rec13+S	20,15	4,431.00	**3,390.00**	**23.49**	**3,429.55**	**22.60**	3,480.00	21.46	24.46	0.38
rec15+S	20,15	4,193.00	**3,331.00**	**20.56**	**3,336.00**	**20.44**	**3,345.00**	**20.22**	4.52	0.36
rec17+S	20,15	4,435.00	**3,369.00**	**24.04**	**3,384.67**	**23.68**	**3,393.00**	**23.49**	10.14	0.39
rec19+S	30,10	5,665.00	**4,216.00**	**25.58**	**4,255.80**	**24.88**	4,314.00	23.85	28.74	0.42
rec21+S	30,10	5,171.00	**3,930.00**	**24.00**	**3,970.80**	**23.21**	**4,002.00**	**22.61**	20.86	0.42
rec23+S	30,10	5,224.00	**3,825.00**	**26.78**	**3,842.58**	**26.44**	3,870.00	25.92	41.64	0.27
rec25+S	30,15	6,387.00	**4,774.00**	**25.25**	4,784.00	25.10	**4,788.00**	**25.04**	32.09	0.37
rec27+S	30,15	6,401.00	**4,573.00**	**28.56**	**4,597.85**	**28.17**	4,682.00	26.86	23.94	0.37
rec29+S	30,15	6,681.00	**4,562.00**	**31.72**	**4,588.82**	**31.32**	**4,603.00**	**31.10**	25.91	0.55
rec31+S	50,10	8,494.00	**6,106.00**	**28.11**	**6,152.90**	**27.56**	6,276.00	26.11	51.03	0.63
rec33+S	50,10	9,278.00	**6,411.00**	**30.90**	**6,429.70**	**30.70**	**6,468.00**	**30.29**	14.07	0.65
rec35+S	50,10	8,969.00	**6,504.00**	**27.48**	6,531.85	27.17	**6,584.00**	**26.59**	19.02	0.64
rec37+S	75,20	16,227.00	**11,388.00**	**29.82**	11,442.90	29.48	**11,585.00**	**28.61**	66.78	2.19
rec39+S	75,20	16,896.00	**11,581.00**	**31.46**	**11,613.38**	**31.27**	**11,630.00**	**31.17**	12.97	1.57
rec41+S	75,20	16,749.00	**11,497.00**	**31.36**	**11,530.25**	**31.16**	**11,574.00**	**30.90**	104.55	1.62
Average	---	---	---	**25.89**	---	**25.52**	---	**24.93**	**26.22**	**0.58**

*Objective Function Value

**Gap is between the algorithms' OFV and 2-Opt algorithm's OFV

4.6.3.3 Computational Results for $F \mid no-wait, S_{sd} \mid C_{\max}$ Test Problems

In order to validate the performance of the proposed PSO when dealing with $F \mid no-wait, S_{sd} \mid C_{\max}$ problems, 29 random test problems based on the test problems of [116] (car1+SD through car8+SD) and [117] (rec01+SD through rec41+SD) are generated. Random sequence dependent setup times are generated based on the following rule:

$$0 \le ST_{[i]ji} < p_{[i]j}; \quad i = 1,2,...,n-1 \quad j = 1,2,...,m \quad ST_{ij} \text{ is integer} \tag{4-41}$$

These test problems are solved by the proposed PSO as well as 2-Opt algorithm and GA+PSO of section 4.5 after incorporating sequence dependent setup times in GA+PSO. For such comparison, only GA+PSO of section 4.5 is considered since superiority of GA+PSO over PSO of section 4.5 is evident from. Table 4-18 and Table 4-19 present the computational results of $F \mid no-wait, S_{sd} \mid C_{\max}$ problems. For the case of car+SD problems, the proposed PSO is never inferior to GA+PSO. And generates better solutions for 3 test cases; this number elevates to 15 for rec+SD test problems. In addition, average CPU time of the proposed PSO is 1.12 seconds, while the average CPU time of GA+PSO is 58.08 seconds on the same machine. Obviously, the proposed PSO outperforms the GA+PSO of section 4.3.

Table 4-18 Computational Results of the Problems car1+SD through car8+SD

			GA+PSO						
	n, m	2-Opt OFV*	OFV*						CPU Time (second)
			Best Makespan	Gap** (%)	Average Makespan	Gap** (%)	Worst Makespan	Gap** (%)	
car1+SD	**11,5**	12,813.00	**10,379.00**	**19.00**	10,710.35	16.41	11,100.00	13.37	2.95
car2+SD	**13,4**	13,835.00	11,308.00	18.27	11,422.25	17.44	11,615.00	16.05	3.06
car3+SD	**12,5**	16,940.00	12,024.00	29.02	12,202.90	27.96	12,389.00	26.87	3.01
car4+SD	**14,4**	16,559.00	12,443.00	24.86	12,668.70	23.49	13,065.00	21.10	3.18
car5+SD	**10,6**	14,993.00	**11,945.00**	**20.33**	12,198.65	18.64	12,394.00	17.33	2.82
car6+SD	**8,9**	15,254.00	**12,015.00**	**21.23**	12,326.40	19.19	12,396.00	18.74	2.22
car7+SD	**7,7**	11,741.00	**9,795.00**	**16.57**	9,796.70	16.56	**9,797.00**	**16.56**	2.18
car8+SD	**8,8**	13,291.00	**11,525.00**	**13.29**	11,611.35	12.64	**11,698.00**	**11.99**	2.44
Average		**N/A**	**---**	**20.32**	**---**	**19.04**	**---**	**17.75**	**2.73**

Table 4-18 (Continued)

			Proposed PSO						
	n, m	2-Opt OFV*	OFV*						CPU Time (second)
			Best Makespan	Gap** (%)	Average Makespan	Gap** (%)	Worst Makespan	Gap** (%)	
car1+SD	**11,5**	12,813.00	**10,379.00**	**19.00**	**10,389.30**	**18.92**	**10,402.00**	**18.82**	0.11
car2+SD	**13,4**	13,835.00	**11,231.00**	**18.82**	**11,275.00**	**18.50**	**11,415.00**	**17.49**	0.11
car3+SD	**12,5**	16,940.00	**11,877.00**	**29.89**	**11,983.60**	**29.26**	**12,109.00**	**28.52**	0.11
car4+SD	**14,4**	16,559.00	**12,420.00**	**25.00**	**12,481.60**	**24.62**	**12,599.00**	**23.91**	0.10
car5+SD	**10,6**	14,993.00	**11,945.00**	**20.33**	**12,013.80**	**19.87**	**12,289.00**	**18.04**	0.10
car6+SD	**8,9**	15,254.00	**12,015.00**	**21.23**	**12,015.00**	**21.23**	**12,015.00**	**21.23**	0.11
car7+SD	**7,7**	11,741.00	**9,795.00**	**16.57**	**9,796.00**	**16.57**	**9,797.00**	**16.56**	0.10
car8+SD	**8,8**	13,291.00	**11,525.00**	**13.29**	**11,564.25**	**12.99**	**11,698.00**	**11.99**	0.10
Average		**N/A**	**N/A**	**20.52**	**---**	**20.25**	**---**	**19.57**	**0.11**

*Objective Function Value

**Gap is between the algorithms' OFV and 2-Opt algorithm's OFV

Table 4-19 Computational Results of the Problems Rec01+SD through Rec41+SD

			GA							
			OFV*							CPU Time (second)
	n, m	2-Opt OFV*	Best Makespan	Gap** (%)	Average Makespan	Gap** (%)	Worst Makespan	Gap** (%)	STD	
rec01+SD	20,5	2,610.00	2,145.00	17.82	2,175.40	16.65	2,194.00	15.94	8.33	34.56
rec03+SD	20,5	2,436.00	1,925.00	20.98	**1,950.20**	**19.94**	**1,995.00**	**18.10**	19.08	34.60
rec05+SD	20,5	2,677.00	2,046.00	23.57	2,087.20	22.03	2,121.00	20.77	17.82	34.45
rec07+SD	20,10	3,539.00	2,661.00	24.81	2,706.10	23.53	2,758.00	22.07	37.93	35.84
rec09+SD	20,10	3,527.00	2,679.00	24.04	2,702.20	23.39	2,751.00	22.00	22.85	35.86
rec11+SD	20,10	3,354.00	2,586.00	22.90	2,603.50	22.38	2,633.00	21.50	19.72	35.63
rec13+SD	20,15	4,203.00	3,354.00	20.20	3,385.35	19.45	3,414.00	18.77	15.53	37.39
rec15+SD	20,15	4,312.00	3,287.00	23.77	3,298.25	23.51	**3,312.00**	**23.19**	8.87	37.03
rec17+SD	20,15	4,203.00	3,298.00	21.53	**3,315.20**	**21.12**	3,368.00	19.87	35.52	37.39
rec19+SD	30,10	5,300.00	3,858.00	27.21	**3,910.30**	**26.22**	4,056.00	23.47	40.81	42.72
rec21+SD	30,10	5,076.00	3,779.00	25.55	**3,820.30**	**24.74**	3,889.00	23.38	35.83	42.97
rec23+SD	30,10	5,048.00	3,650.00	27.69	3,686.25	26.98	**3,739.00**	**25.93**	28.24	42.72
rec25+SD	30,15	6,352.00	**4,654.00**	**26.73**	4,748.60	25.24	4,902.00	22.83	73.70	45.04
rec27+SD	30,15	6,101.00	4,604.00	24.54	**4,638.30**	**23.97**	**4,696.00**	**23.03**	29.57	45.28
rec29+SD	30,15	6,344.00	4,428.00	30.20	4,484.35	29.31	4,557.00	28.17	27.86	44.99
rec31+SD	50,10	8,068.00	6,020.00	25.38	6,082.00	24.62	6,154.00	23.72	44.76	63.12
rec33+SD	50,10	8,645.00	6,204.00	28.24	6,269.10	27.48	6,403.00	25.93	45.19	63.18
rec35+SD	50,10	8,741.00	6,197.00	29.10	**6,260.60**	**28.38**	**6,369.00**	**27.14**	50.70	63.20
rec37+SD	75,20	15,379.00	10,931.00	28.92	11,038.45	28.22	**11,147.00**	**27.52**	61.00	147.24
rec39+SD	75,20	16,575.00	11,317.00	31.72	11,411.85	31.15	**11,529.00**	**30.44**	56.07	148.01
rec41+SD	75,20	16,470.00	11,364.00	31.00	11,528.50	30.00	11,716.00	28.86	95.70	148.55
Average	**---**	**---**	**---**	**25.52**	**---**	**24.68**	**---**	**23.46**	**36.91**	**58.08**

Table 4-19 (Continued)

	n, m	2-Opt OFV*	GA+PSO							
			OFV*							CPU Time (second)
			Best Makespan	Gap** (%)	Average Makespan	Gap** (%)	Worst Makespan	Gap** (%)	STD	
rec01+SD	20,5	2,610.00	**2,139.00**	**18.05**	**2,163.85**	**17.09**	**2,174.00**	**16.70**	14.25	0.16
rec03+SD	20,5	2,436.00	**1,902.00**	**21.92**	1,956.80	19.67	2,007.00	17.61	44.51	0.16
rec05+SD	20,5	2,677.00	**2,028.00**	**24.24**	**2,055.90**	**23.20**	**2,090.00**	**21.93**	25.70	0.16
rec07+SD	20,10	3,539.00	**2,652.00**	**25.06**	**2,696.65**	**23.80**	**2,734.00**	**22.75**	21.56	0.24
rec09+SD	20,10	3,527.00	**2,657.00**	**24.67**	**2,696.95**	**23.53**	**2,710.00**	**23.16**	9.70	0.23
rec11+SD	20,10	3,354.00	**2,558.00**	**23.73**	**2,587.60**	**22.85**	**2,616.00**	**22.00**	20.59	0.23
rec13+SD	20,15	4,203.00	**3,309.00**	**21.27**	**3,349.00**	**20.32**	**3,398.00**	**19.15**	31.31	0.31
rec15+SD	20,15	4,312.00	**3,222.00**	**25.28**	**3,292.40**	**23.65**	3,341.00	22.52	37.58	0.32
rec17+SD	20,15	4,203.00	**3,271.00**	**22.17**	3,315.30	21.12	**3,359.00**	**20.08**	18.13	0.33
rec19+SD	30,10	5,300.00	**3,848.00**	**27.40**	3,932.40	25.80	**4,012.00**	**24.30**	32.31	0.39
rec21+SD	30,10	5,076.00	**3,756.00**	**26.00**	3,820.95	24.73	**3,855.00**	**24.05**	27.36	0.39
rec23+SD	30,10	5,048.00	**3,628.00**	**28.13**	**3,680.60**	**27.09**	3,746.00	25.79	31.94	0.38
rec25+SD	30,15	6,352.00	4,655.00	26.72	**4,712.95**	**25.80**	**4,819.00**	**24.13**	43.13	0.53
rec27+SD	30,15	6,101.00	**4,565.00**	**25.18**	4,639.65	23.95	4,701.00	22.95	37.95	0.51
rec29+SD	30,15	6,344.00	**4,422.00**	**30.30**	**4,481.45**	**29.36**	**4,528.00**	**28.63**	33.62	0.51
rec31+SD	50,10	8,068.00	**5,966.00**	**26.05**	**6,042.60**	**25.10**	**6,125.00**	**24.08**	57.79	0.96
rec33+SD	50,10	8,645.00	**6,186.00**	**28.44**	**6,266.00**	**27.52**	**6,354.00**	**26.50**	50.04	1.03
rec35+SD	50,10	8,741.00	**6,169.00**	**29.42**	6,279.60	28.16	6,374.00	27.08	54.74	0.98
rec37+SD	75,20	15,379.00	**10,782.00**	**29.89**	**10,988.15**	**28.55**	11,188.00	27.25	108.80	5.24
rec39+SD	75,20	16,575.00	**11,189.00**	**32.49**	**11,334.85**	**31.61**	11,594.00	30.05	89.31	5.16
rec41+SD	75,20	16,470.00	**11,324.00**	**31.24**	**11,479.30**	**30.30**	**11,667.00**	**29.16**	111.46	5.23
Average	**---**	**---**	**---**	**26.08**	**---**	**24.92**	**---**	**23.80**	**42.94**	**1.12**

*Objective Function Value

**Gap is between the algorithms' OFV and 2-Opt algorithm's OFV

4.1 Conclusion

This chapter considered the scheduling problems of $F|no-wait|C_{\max}$, $F|no-wait,setup|C_{\max}$, and $F|no-wait,S_{sd}|C_{\max}$. All of the mentioned problems are strongly NP-Hard. Mathematical models were developed for the problems; and all the problems were reduced to permutation problems. An algorithm for calculating the makespan of a given permutation of jobs was developed for each problem. Three algorithms were proposed to deal with the sequencing part: a hybrid of TS and a PSO based on factoradic coding, a GA and its hybrid with factoradic coding PSO, and a PSO based on MC coding. Both factoradic coding and MC are methods that transform the permutations of jobs into a form that PSO is able to efficiently operate with.

Experimentally inferred rules were used for tuning the parameters of the developed algorithms. Computational results on the available test problems in the literature with small and large instances revealed the efficiency of the proposed methods in finding good-quality solutions for the test problems in a very short time, compared to the other methods. The proposed algorithms improved many of the best-known solutions in the literature.

Computational results from GA and TS, when compared to the hybridized versions of these algorithms, prove the effectiveness of hybridization for the problems studied in this chapter. As it was mentioned in this chapter, when hybridized, the exchange local search procedure of GA and TS were replaced with a PSO algorithm. Potentially, PSO is able to explore through a wider selection of solutions compared to exchange. In fact, once the input string of length n is fed to exchange, this algorithm's

feasible region includes $\binom{n}{2} = \frac{n(n-1)}{2}$ solutions while PSO's feasible region includes $n!$ solutions.

Chapter 5

2-Machine No-Wait Job Shop Problem

5.1 Background

This chapter mainly generalizes the results of chapter 3 for the case of 2-machine flow shop problem with setup times and single server constraints. Accordingly, the problems studied in this chapter include $J2|no-wait|C_{\max}$, $J2|no-wait,setup|C_{\max}$ and $J2,S1|no-wait,setup|C_{\max}$. All of the problems that are studied in this chapter are strongly NP-Hard as they are a generalization of their flow shop versions, and since the flow shop versions are NP-Hard, NP-Hardness of job shop versions are intuitive.

In this chapter, the same notation of chapter 3 is used. Moreover, some of the theorems of chapter 3 are generalized for the job shop problems studied in this chapter. A method to calculate the makespan of a given permutation is developed. An efficient GA is proposed to deal with the problems. Computational results show that the proposed GA is very competitive when applied to small and large instance problems.

5.2 Problem Description

Figure 5-1 depicts the four possible conditions of the consecutive jobs based on the different processing routes they might have on the two machines when a sequence π of $J2|no-wait,setup|C_{\max}$ is considered.

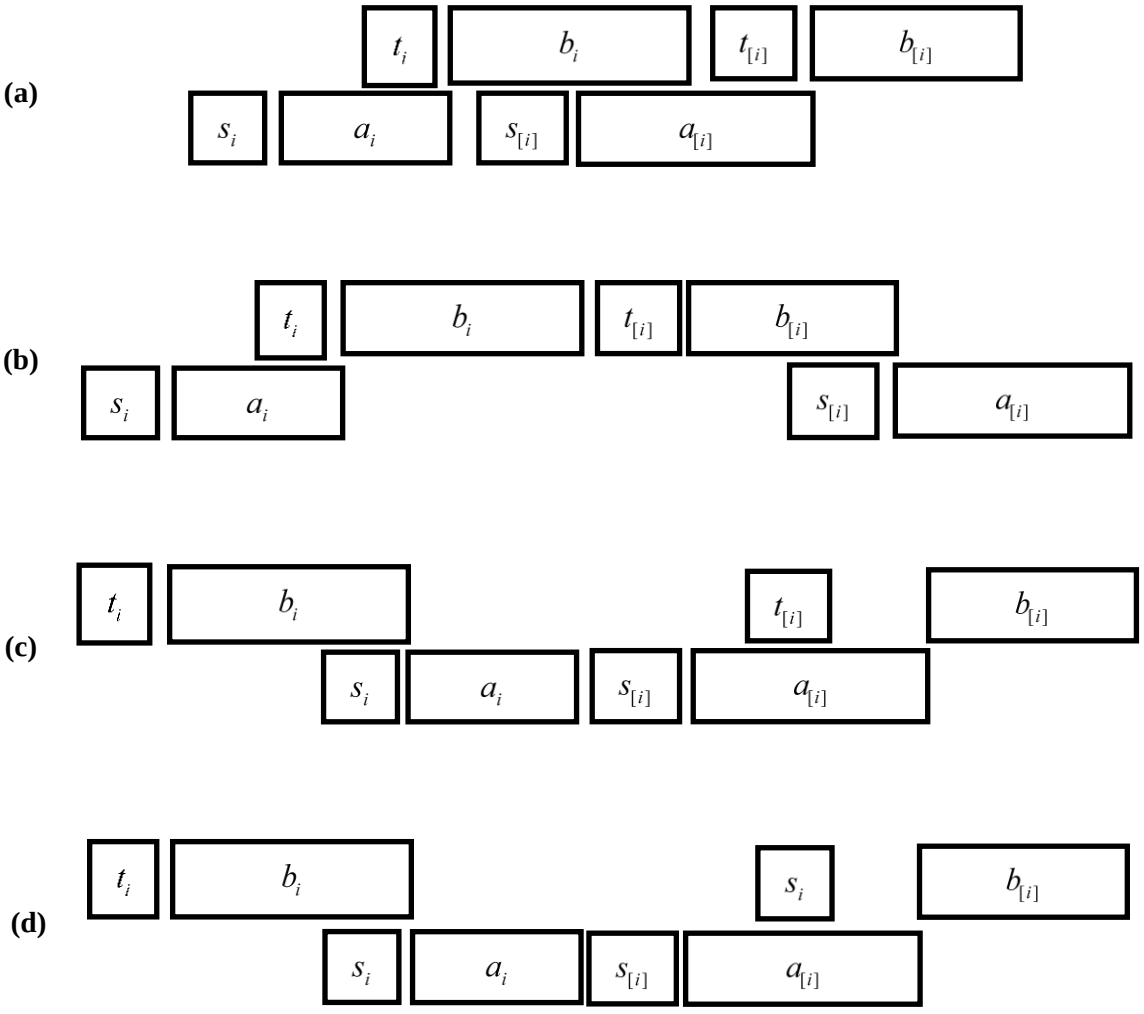

Figure 5-1 Four possible conditions of the consecutive jobs

This figure shows that $J2|no-wait,setup|C_{\max}$ can be formulated as a permutation problem if all the setup times are non-zero; non-zero setup times is one of the assumptions in this research. In other words, in none of the 4 cases only one of the operations $a_{[i]}$ or $b_{[i]}$ can be exchanged with a_i or b_i with the aim of reducing the makespan without violating the no-wait constraints. The same applies to $J2,S1|no-wait,setup|C_{\max}$. It should be noted that in either case, whether zero setup times permitted or not, both of the problems are NP-hard and algorithms with polynomially bounded computational complexity cannot be developed to solve the

problems unless $P = NP$. If zero setup times are permitted, then the problems are categorized under no-wait job shop problem, which is NP-hard [72]. When zero setup times are not permitted, the problem can be reduced to TSP [34]. Based on the above notations and explanations, a mathematical model of $J2, S1 \mid no-wait, setup \mid C_{\max}$ can be developed as follows:

$$\min C_{\max} \tag{5-1}$$

$$C_{\max} \geq ST_{i1} + a_i;\ i = 1,2,...,n \tag{5-2}$$

$$C_{\max} \geq ST_{i2} + b_i;\ i = 1,2,...,n \tag{5-3}$$

$$ST_{i1} \geq ST_{s_i} + s_i;\ i = 1,2,...,n-1 \tag{5-4}$$

$$ST_{i2} \geq ST_{t_i} + t_i;\ i = 1,2,...,n-1 \tag{5-5}$$

$$ST_{s_{[i]}} \geq ST_{i1} + a_i;\ i = 1,2,...,n-1 \tag{5-6}$$

$$ST_{t_{[i]}} \geq ST_{i2} + b_i;\ i = 1,2,...,n-1 \tag{5-7}$$

$$ST_{i2} \geq ST_{i1} + a_i - Mz_i;\ i = 1,2,...,n \tag{5-8}$$

$$ST_{i2} \leq ST_{i1} + a_i + Mz_i;\ i = 1,2,...,n \tag{5-9}$$

$$ST_{i1} \leq ST_{i2} + b_i + M(1 - z_i);\ i = 1,2,...,n \tag{5-10}$$

$$ST_{i1} \geq ST_{i2} + b_i - M(1 - z_i);\ i = 1,2,...,n \tag{5-11}$$

$$ST_{s_i} - ST_{t_i} \geq t_i - My_i;\ i = 1,2,...,n \tag{5-12}$$

$$ST_{t_i} - ST_{s_i} \geq s_i - M(1 - y_i);\ i = 1,2,...,n \tag{5-13}$$

$$\begin{aligned} &ST_{ij} \geq 0;\ ST_{s_i} \geq 0;\ ST_{t_i} \geq 0;\ y_i \in \{0,1\} \\ &i = 1,2,...,n \\ &j = 1,2 \end{aligned} \tag{5-14}$$

In this model, (5-1) is the objective function that minimizes the makespan. (5-2) and (5-3) relate the makespan to the completion time of the last operation of the last job.

(5-4) and (5-5) initiate the setup times; in other words, a certain operation could be started only after its setup time is completed. M is a sufficiently large positive number. $z_i = 0$ means that job J_i is first processed by machine 1 and then processed by machine 2; $z_i = 1$ otherwise. (5-6) and (5-7) guarantee that the setup times and operating times do not overlap.

(5-8), (5-9), (5-10), and (5-11) are the no-wait constraints; if $z_i = 0$, (5-8) and (5-9) are active while (5-10) and (5-11) remain inactive. In this case, second operation of J_i will start on machine 2 when the first operation of the same job on machine 1 is completed. If $z_i = 1$, the second operation of J_i on the first machine cannot be processed unless the first operation on machine 2 is processed; this means that (5-8) and (5-9) are inactive when (5-10) and (5-11) are active. Finally, (5-12) and (5-13) are single server constraints. y_i is one of the variables of the model. This variable along with (5-12) and (5-13) guarantees that setup operations of J_i should not overlap each other.

As mentioned earlier, $J2 \mid no-wait, setup \mid C_{\max}$ and $J2, S1 \mid no-wait, setup \mid C_{\max}$ are permutation problems when all the setup times are absolutely positive. Therefore, it can be inferred that in order to solve these problems, one can search the set of all permutations of numbers $1, 2, ..., n$ to find the permutation that minimizes the makespan of the problem. Consider $\pi_n = (J_1, J_2, ..., J_j, ..., J_n)$ is a permutation of n jobs. Accordingly, disjoint sets $L_k, k \geq 1$ that partition π_n can be defined as follows:

$$L_1 = \left\{ J_j \in \pi_n \mid j \text{ is the smallest index where } z_j \neq z_{j+1} \right\} \quad (5\text{-}15)$$

$$L_2 = \left\{ J_j \in \pi_n \mid j \in \{\pi_n - L_1\} \wedge j \text{ is the smallest index where } z_j \neq z_{j+1} \right\} \quad (5\text{-}16)$$

$$L_k = \left\{ J_j \in \pi_n \mid j \in \left\{ \pi_n - L_1 - L_2 - \ldots - L_{k-1} \right\} \wedge j \text{ is the smallest index where } z_j \neq z_{j+1} \right\} \qquad (5\text{-}17)$$

Once the above sets are formed, each set represents a $F2 \mid no-wait, setup \mid C_{\max}$ sub-problem (for $F2 \mid no-wait, setup \mid C_{\max}$, $k=1$); together these sub-problems establish the $J2 \mid no-wait, setup \mid C_{\max}$ with non-zero setup times. Thus, the problem of $J2 \mid no-wait \mid C_{\max}$ can be formulated as (5-18). it should be noted that (5-18) is a generalization of the formula proved for $F2 \mid no-wait \mid C_{\max}$ in [119].

$$\underset{\pi \in \Pi}{Min} \left\{ \begin{array}{l} \sum\limits_{\substack{L_i \\ i=2p+1; p=0,1,\ldots}} \left(a_{\min j \in L_i} + \sum\limits_{w=\min j \in L_i}^{(\max j \in L_i)-1} \max\left(0, a_{[w]} - b_w\right) \right) + \sum\limits_{w=\min j \in L_i}^{\max j \in L_i} b_w \\ + \sum\limits_{\substack{L_i \\ i=2p; p=0,1,\ldots}} \left(b_{\min j \in L_i} + \sum\limits_{w=\min j \in L_i}^{(\max j \in L_i)-1} \max\left(0, b_{[w]} - a_w\right) \right) + \sum\limits_{w=\min j \in L_i}^{\max j \in L_i} a_w \end{array} \right\} \qquad (5\text{-}18)$$

Moreover, a generalization to $J2 \mid no-wait, setup \mid c_{\max}$ from the formula given in [119] for $F2 \mid no-wait, setup \mid c_{\max}$ is as follows.

$$\underset{\pi \in \Pi}{Min} \left\{ \begin{array}{l} \sum\limits_{\substack{L_i \\ i=2p+1; p=0,1,\ldots}} \left(\max\left(s_{\min j \in L_i} + a_{\min j \in L_i}, t_{\min j \in L_i} \right) \right) \\ + \sum\limits_{w=\min j \in L_i}^{(\max j \in L_i)-1} \max\left(0, s_{[w]} + a_w - b_w - t_{[w]}\right) + \sum\limits_{w=\min j \in L_i}^{\max j \in L_i} b_w + \sum\limits_{w=(\min j \in L_i)+1}^{\max j \in L_i} t_w \\ + \sum\limits_{\substack{L_i \\ i=2p; p=0,1,\ldots}} \left(\max\left(t_{\min j \in L_i} + b_{\min j \in L_i}, s_{\min j \in L_i} \right) \right) \\ + \sum\limits_{w=\min j \in L_i}^{(\max j \in L_i)-1} \max\left(0, t_{[w]} + b_w - a_w - s_{[w]}\right) + \sum\limits_{w=\min j \in L_i}^{\max j \in L_i} a_w + \sum\limits_{w=(\min j \in L_i)+1}^{\max j \in L_i} s_w \end{array} \right\} \qquad (5\text{-}19)$$

According to (5-19), the objective function of a feasible solution π of the problem $J2,S1 \mid no-wait, setup \mid C_{\max}$ can be calculated by the following algorithm.

1. Set $i=1$

2. If $z_i = 0$:

2.1. $ST_{s_1} = 0; ST_{t_1} = s_1$

2.2. $ST_{11} = \begin{cases} s_1 & \text{if } t_1 \le a_1 \\ ST_{t_1} + t_1 - a_1 & \text{Otherwise} \end{cases}$

2.3. $ST_{12} = ST_{11} + a_1$

2.4. Set $i \leftarrow i + 1$

3. If $z_i = 1$

3.1. $ST_{t_1} = 0; ST_{s_1} = t_1$

3.2. $ST_{12} = \begin{cases} t_1 & \text{if } s_1 \le b_1 \\ ST_{s_1} + s_1 - b_1 & \text{Otherwise} \end{cases}$

3.3. $ST_{11} = ST_{12} + b_1$

3.4. Set $i \leftarrow i + 1$

4. Suppose that $i = u; 1 < u \le n,\ z_i = 0$. Then:

4.1. If $z_{i-1} = 0$, then:

4.1.1. $ST_{s_i} = ST_{(i-1)1} + a_{(i-1)}$

4.1.2. $ST_{t_i} = \begin{cases} ST_{s_i} + s_i & \text{if } s_i \ge b_{(i-1)} \\ ST_{(i-1)2} + b_{(i-1)} & \text{Otherwise} \end{cases}$

4.1.3. $ST_{i1} = \begin{cases} ST_{s_i} + s_i & \text{if } t_i + b_{i-1} - s_i \le a_i \\ ST_{t_i} + t_i - a_i & \text{Otherwise} \end{cases}$

4.1.4. $ST_{i2} = ST_{i1} + a_i$

4.2. If $z_{i-1} = 1$, then:

4.2.1. $ST_{t_i} = ST_{(i-1)2} + b_{(i-1)}$

4.2.2. $ST_{s_i} = \begin{cases} ST_{t_i} + t_i & \text{if } t_i \ge a_{(i-1)} \\ ST_{(i-1)1} + a_{(i-1)} & \text{Otherwise} \end{cases}$

4.2.3. $ST_{i1} = ST_{s_i} + s_i$

4.2.4. $ST_{i2} = ST_{i1} + a_i$

5. Suppose that $i = u; 1 < u \le n,\ z_i = 1$. Then:

5.1. If $z_{i-1} = 0$, then:

5.1.1. $ST_{s_i} = ST_{(i-1)1} + a_{(i-1)}$

5.1.2. $ST_{t_i} = \begin{cases} ST_{s_i} + s_i & \text{if } s_i \ge b_{(i-1)} \\ ST_{(i-1)2} + b_{(i-1)} & \text{Otherwise} \end{cases}$

5.1.3. $ST_{i2} = ST_{t_i} + t_i$

5.1.4. $ST_{i1} = ST_{i2} + b_i$

5.2. If $z_{i-1} = 1$, then:

5.2.1. $ST_{t_i} = ST_{(i-1)2} + b_{(i-1)}$

5.2.2. $ST_{s_i} = \begin{cases} ST_{t_i} + t_i & \text{if } t_i \ge a_{(i-1)} \\ ST_{(i-1)1} + a_{(i-1)} & \text{Otherwise} \end{cases}$

5.2.3. $ST_{i2} = \begin{cases} ST_{t_i} + t_i & \text{if } b_i \ge (ST_{s_i} + s_i) - (ST_{t_i} + t_i) \\ ST_{s_i} + s_i - b_i & \text{Otherwise} \end{cases}$

5.2.4. $ST_{i1} = ST_{i2} + b_i$

6. If $i < n$, set $i \leftarrow i + 1$ and go back to step 4. Otherwise, $C_{\max} = \max\{ST_{n1} + a_n, ST_{n2} + b_n\}$.

A simpler version of this algorithm is proved to be very efficient [34, 92]. The above algorithm can easily be modified for the problem without single server constraints as follows:

1. Set $i = 1$

2. If $z_i = 0$:

2.1. $ST_{s_1} = 0;\ ST_{t_1} = 0$

2.2. $ST_{11} = \begin{cases} s_1 & \text{if } t_1 \le a_1 + s_1 \\ ST_{t_1} + t_1 - a_1 & \text{Otherwise} \end{cases}$

2.3. $ST_{12} = ST_{11} + a_1$

2.4. Set $i \leftarrow i + 1$

3. If $z_i = 1$

3.1. $ST_{t_1} = 0;\ ST_{s_1} = 0$

3.2. $ST_{12} = \begin{cases} t_1 & \text{if } s_1 - t_1 \le b_1 \\ ST_{s_1} + s_1 - b_1 & \text{Otherwise} \end{cases}$

3.3. $ST_{11} = ST_{12} + b_1$

3.4. Set $i \leftarrow i + 1$

4. Suppose that $i = u; 1 < u \le n,\ z_i = 0$. Then:

4.1. If $z_{i-1} = 0$, then:

4.1.1. $ST_{s_i} = ST_{(i-1)1} + a_{(i-1)};\ ST_{t_i} = ST_{(i-1)2} + b_{(i-1)}$

4.1.2. $ST_{i1} = \begin{cases} ST_{s_i} + s_i & \text{if } t_i + b_{i-1} - s_i \le a_i \\ ST_{t_i} + t_i - a_i & \text{Otherwise} \end{cases}$

4.1.3. $ST_{i2} = ST_{i1} + a_i$

4.2. If $z_{i-1} = 1$, then:

4.2.1. $ST_{s_i} = ST_{(i-1)1} + a_{(i-1)};\ ST_{t_i} = ST_{(i-1)2} + b_{(i-1)}$

4.2.2. $ST_{i1} = \begin{cases} ST_{s_i} + s_i & \text{if } a_i \ge (ST_{t_i} + t_i) - (ST_{s_i} + s_i) \\ ST_{t_i} + t_i - a_i & \text{otherwise} \end{cases}$

4.2.3. $ST_{i2} = ST_{i1} + a_i$

5. Suppose that $i = u; 1 < u \le n,\ z_i = 1$. Then:

5.1. If $z_{i-1} = 0$, then:

5.1.1. $ST_{s_i} = ST_{(i-1)1} + a_{(i-1)}$; $ST_{t_i} = ST_{(i-1)2} + b_{(i-1)}$

5.1.2. $ST_{i2} = \begin{cases} ST_{t_i} + t_i & \text{if } b_i \geq (ST_{s_i} + s_i) - (ST_{t_i} + t_i) \\ ST_{s_i} + s_i - b_i & \text{Otherwise} \end{cases}$

5.1.3. $ST_{i1} = ST_{i2} + b_i$

5.2. If $z_{i-1} = 1$, then:

5.2.1. $ST_{s_i} = ST_{(i-1)1} + a_{(i-1)}$; $ST_{t_i} = ST_{(i-1)2} + b_{(i-1)}$

5.2.2. $ST_{i2} = \begin{cases} ST_{t_i} + t_i & \text{if } b_i \geq (ST_{s_i} + s_i) - (ST_{t_i} + t_i) \\ ST_{s_i} + s_i - b_i & \text{Otherwise} \end{cases}$

5.2.3. $ST_{i1} = ST_{i2} + b_i$

6. If $i < n$, set $i \leftarrow i + 1$ and go back to step 4. Otherwise, $C_{\max} = \max\{ST_{n1} + a_n, ST_{n2} + b_n\}$.

The correctness of the above algorithms can be verified by using mathematical induction and considering all of the different possible conditions in the Gantt chart. Computational complexity of this simple yet effective algorithm is $O(n)$.

5.3 The Proposed Algorithm

Genetic algorithm (GA) is a search technique that is used to find near-optimal solutions for optimization problems. Genetic algorithms are a particular class of evolutionary algorithms (EA) that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover. GA also uses chromosomes to code the feasible solutions of the problem.

5.3.1 Chromosome Structure and GA Operations

Chromosome structure (Genotype) is one of the most important aspects of the genetic algorithm. In the proposed GA, each permutation in Π is a chromosome.

Moreover, each permutation π in Π can be a feasible solution of the problem if the described algorithm of chapter 3 is used to schedule the jobs. This also defines the extraction of solutions from chromosomes (Phenotype). It is worthwhile to mention that the proposed GA uses the operations, defined by [120].

The proposed GA generates Pop random permutations for its first generation. Then, the algorithm calculates the makespan of each of the permutations which will be used as the fitness label of that permutation. Size of the population, Pop, which is an even number and a parameter of the algorithm, remains constant for all generations. New generations are made from the existing generation, using four operations: crossover, mutation, immigration, and local search.

In crossover operation the existing generation is randomly partitioned into $Pop/2$ pairs of parents, and crossover operation is performed on each pair with probability P_c. If a pair is not selected for crossover, each individual of the pair is considered for mutation operation with probability P_m and then for local search with probability P_l. Crossover, mutation, and local search will be described in this section.

Crossover operation on a pair of parents, P_1 and P_2, produces two children, C_1 and C_2. Let $f(I)$ be the makespan of schedule I. If $r_i.(f(P_i)+f(C_i)) \le f(P_i)$ then C_i will be selected for local search with probability P_l and then goes to new generation and P_i dies out ($i=1,2$). r_i is a random number generated from the interval $[0,1]$ for each i. Otherwise C_i dies out and P_i is considered for mutation with probability P_m and then for local search with probability P_l. It should be noted that $r_i.(f(P_i)+f(C_i)) \le f(P_i)$ determines how much advancement in the quality of genes should be expected in consecutive generations. However, since r_i is a random number, the amount of gene progress differs in each iteration. Afterwards, immigration operation is also performed

before finalizing the cycle of producing a new generation. Immigration operation feeds the gene pool with randomly generated genes, helps maintaining the gene diversity, and prevents immature convergence.

For immigration operation, a chromosome will be randomly generated and is called NEW. An individual I, is selected randomly from the current population. Let the probability of leaving I be $P_{Leave}(I, NEW) = \frac{f(I)}{f(I) + f(NEW)}$. A random number is generated from the interval $[0,1]$. If this random number is less than $P_{Leave}(I, NEW)$, NEW replaces I, otherwise NEW is discarded. The immigration operation provides the new generation with the chance of bringing new desirable characteristics to it. Chromosome with least makespan value in the final generation is the result given by the algorithm. Pop, P_s, P_m and P_l are adjustable parameters of the algorithm.

5.3.2 Crossover

The proposed GA uses a one-point crossover. Suppose that $P_1 = (J_1^1, J_2^1, ..., J_n^1)$ and $P_2 = (J_1^2, J_2^2, ..., J_n^2)$ are the two individuals that are selected for crossover. The one-point crossover selects an integer number $r \in [1, n]$. Then, crossover operation is performed and the result is C_1 and C_2 whose chromosomes are defined as $(J_1^{c_1}, ..., J_r^{c_1}, J_{r+1}^{c_1}, ..., J_n^{c_1})$ and $(J_1^{c_2}, ..., J_r^{c_2}, J_{r+1}^{c_2}, ..., J_n^{c_2})$. $(J_1^{c_1}, ..., J_r^{c_1}) = (J_1^1, ..., J_r^1)$ and $J_a^{c_1} = J_b^2; a = r+1, ..., n$ where b is the lowest index such that $J_b^2 \notin \left\{ J_1^{c_1}, ..., J_{a-1}^{c_1} \right\}$. And $(J_1^{c_2}, ..., J_r^{c_2}) = (J_1^2, ..., J_r^2)$ and $J_a^{c_2} = J_b^1; a = r+1, ..., n$ where b is the lowest index such that $J_b^2 \notin \left\{ J_1^{c_2}, ..., J_{a-1}^{c_2} \right\}$. Figure 5-2 demonstrates the one-point crossover operation.

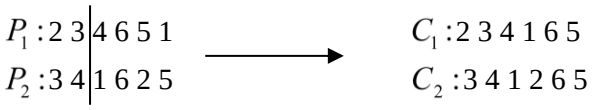

Figure 5-2 One-point crossover operation

5.3.3 Mutation

Let $P=(J_1,J_2,...,J_n)$ be the selected chromosome for mutation. Then, the algorithm generates two integer numbers $r_1,r_2 \in [1,n-1]$ and an integer number $a \in [0,1]$. If $a \geq 0.5$, then the new chromosome will be $P_{new}=(J_1,...,J_{r_1},J_{r_2},...,J_n,J_{r_1+1},...,J_{r_2-1})$, while if $a<0.5$, the new chromosome will be $P_{new}=(J_{r_1+1},...,J_{r_2}J_1,...,J_{r_1},J_{r_2+1},...,J_n)$. Figure 5-3 demonstrates the mutation operation.

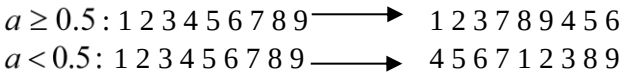

Figure 5-3 Mutation operation when n=9, r_1=3, and r_2=7

5.3.4 Local Search

This sub-algorithm exchanges the location of the first two adjacent genes in the chromosome and evaluates the fitness function of the new chromosome. If the fitness function of the new chromosome has improved by the exchange, algorithm accepts this exchange and restarts the exchange sub-procedure. Nevertheless, if this exchange does not improve the fitness function of the chromosome, the two genes will move back to their original locations and exchange will be applied to the next two adjacent genes.

5.4 Computational Results

As seen in section 5.3, 4 control parameters should be set for the proposed GA to start the search. Values of these parameters affect the algorithm's performance. In order

to tune the algorithm parameters, approach of section 3.5 is utilized. Based on these observations, the following values are proposed:

$$Pop = \frac{n}{5} \qquad P_c = 0.5$$
$$P_m = 0.5 \qquad P_l = 0.2 \tag{5-20}$$

Furthermore, in order to examine the efficiency of the algorithm, a lower bound for each test problem is considered. This lower bound can be obtained by (5-21) for each job i:

$$LB = \max\{\sum_{i=1}^{n}(s_i + a_i), \sum_{i=1}^{n}(t_i + b_i)\} \tag{5-21}$$

Clearly, smaller gaps between LB and the objective function of the final solution demonstrates the efficiency of the proposed algorithm. For a specific problem, a small gap means that the algorithm is able to find solutions in which the operating time of the jobs on one of the machines is scheduled parallel to the operating time of the jobs on the other machine. Moreover, the same test problems without single server constraints were considered and the proposed algorithm was applied to them. Smaller gaps between the objective function values of the same problem without and with single server constraints demonstrate that the proposed algorithm is able to generate schedules that are close to the problem without single server constraints.

The proposed algorithm was coded in C++ and run on a PC equipped with a 3GHz Intel Pentium IV CPU and 2 GB of RAM. To verify the performance of the algorithm, 5 problems with 10, 11, and 12 jobs each were randomly generated and solved to optimality using the mathematical model of chapter 3. These test problems have been generated based on a uniform probability distribution function with a_i and b_i integer numbers between 0 and 100, s_i an integer number between 0 and a_i, and t_i an integer number between 0 and b_i. The uniform distribution is chosen to generate test

problems since it is evident that it results in difficult problems [101, 102]. Afterwards, the proposed algorithm was applied to the same set of problems. Computational results of these problems are presented in Table 5-1. Table 5-1 indicates that the proposed algorithm is able to find the optimal solution of all of the instances in this set. It should be noted that solving test problems with more than 12 jobs to optimality takes more than 3 hours. In Table 5-1, the first column indicates the problem number. Second column represents the number of jobs in the instance. Rest of the columns represent the objective function value and CPU time of the problems without setup, with setup, and with setup and single server, when the respective test problem is solved by the mathematical model and by the proposed algorithm. Last two column of this table represent the lower bounds calculated by (5-21), and the gap between the lower bound and the optimal solution. Table 5-1 demonstrates that the gap between the optimal solution and the solution found by the proposed algorithm is 0 in all cases.

In order to examine the performance of the algorithm on larger test problems, another set of test problems was generated. The operating times were generated from [1,10] and [1,100] intervals. Setup times were generated from the intervals $[1,10\alpha]$ and $[1,100\alpha]$ where α can be 0.25, 0.5, 0.75, and 1. Problems with 20, 50, 100, 200, 500, and 1000 jobs were considered in order to test the algorithm on a variety of problems with different sizes. For each problem size, 12 different combinations of operating time, setup time, and α were considered. Additionally, for each specific combination of operation time, setup time, and α, 20 random test problems were generated, which sums up to 1440 test problems of different sizes, and 14400 independent runs of the proposed algorithm as each problem is solved 10 times. Table 5-2 shows the categories to generate problems' data.

In Table 5-3 to Table 5-5, first column gives the problem number. Second column represents the number of jobs. Columns 3, 4 and 5 are assigned to the interval in which the operating times and setup times are generated as well as the value of α. Next 6 columns summarize the computational results for the problems without setup time, with setup time, and with setup time and single server constraints. First, the standard deviation (STD) of the 10 objective functions generated from applying the GA to each test problem was calculated. The average of these standard deviations is presented in the Average STD columns. The lower the values of the Average STD, the more consistent the algorithm is when applied to different problems. Note that in the following tables, Average Time is the average CPU time in milliseconds. Column 12, APAM+SS, calculates the percentage added to the makespan of the problem with setup times when single server constraints are also considered.

$AD(SS, LB)$ column evaluates the average deviation between the problem with single server constraints and LB. The same problems with single server constraints were also solved by the famous 2-opt algorithm for comparison purposes. For a permutation problem, 2-opt algorithm arbitrarily chooses two jobs in the permutation and exchanges the places of these jobs; objective function will be calculated and the exchange will be accepted if an improvement is noticed. The algorithm continues until no such improvement can be made [103]. After applying the 2-opt algorithm to the problems, the average deviation between the objective function of the solutions proposed by the 2-opt algorithm and the proposed GA is presented in the $AD(SS, 2-opt)$ column. This column shows that the solutions obtained by the GA are always superior to the 2-opt algorithm.

Table 5-1 Solutions of Problems with Small Instances

Prob. No.	No. of Jobs	Without Setup Time				Without Single Server Constraints				With Single Server Constraints			
		Optimal Solution		Proposed Algorithm		Optimal Solution		Proposed Algorithm		Optimal Solution		Proposed Algorithm	
		OFV*	Time	OFV*	Time	OFV*	Time	OFV*	Time	OFV*	Time	OFV*	Time
1	**10**	**685.00**	1.02	**685.00**	2.065	**1,024.00**	1.624	**1,024.00**	2.189	**1,086.00**	1.949	**1,086.00**	2.298
2	**10**	**672.00**	1.10	**672.00**	1.964	**1,011.00**	1.764	**1,011.00**	2.082	**1,051.00**	2.116	**1,051.00**	2.186
3	**10**	**424.00**	1.30	**424.00**	2.746	**749.00**	2.080	**749.00**	2.911	**770.00**	2.496	**770.00**	3.056
4	**10**	**553.00**	1.02	**553.00**	2.642	**832.00**	1.630	**832.00**	2.801	**860.00**	1.956	**860.00**	2.941
5	**10**	**637.00**	1.10	**637.00**	2.862	**887.00**	1.755	**887.00**	3.034	**905.00**	2.106	**905.00**	3.185
6	**11**	**744.00**	132.36	**744.00**	2.938	**1,123.00**	17.762	**1,123.00**	3.114	**1,185.00**	19.999	**1,185.00**	3.270
7	**11**	**685.00**	131.98	**685.00**	3.003	**1,031.00**	17.547	**1,031.00**	3.183	**1,071.00**	19.283	**1,071.00**	3.342
8	**11**	**487.00**	134.07	**487.00**	3.023	**815.00**	17.534	**815.00**	3.204	**887.00**	20.129	**887.00**	3.365
9	**11**	**604.00**	132.38	**604.00**	2.996	**913.00**	17.892	**913.00**	3.176	**941.00**	20.827	**941.00**	3.335
10	**11**	**675.00**	138.07	**675.00**	3.089	**973.00**	16.972	**973.00**	3.274	**991.00**	21.003	**991.00**	3.438
11	**12**	**825.00**	4,180.14	**825.00**	3.231	**1,224.00**	6,688.218	**1,224.00**	3.425	**1,286.00**	8,025.862	**1,286.00**	3.596
12	**12**	**749.00**	4,303.79	**749.00**	3.193	**1,165.00**	6,886.063	**1,165.00**	3.385	**1,232.00**	8,263.276	**1,232.00**	3.554
13	**12**	**537.00**	4,419.42	**537.00**	3.173	**912.00**	7,071.068	**912.00**	3.363	**1,007.00**	8,485.281	**1,007.00**	3.532
14	**12**	**652.00**	4,299.90	**652.00**	3.217	**998.00**	6,879.845	**998.00**	3.410	**1,026.00**	8,255.814	**1,026.00**	3.581
15	**12**	**746.00**	4,557.63	**746.00**	3.275	**1,065.00**	221.626	**1,065.00**	3.472	**1,083.00**	8,750.645	**1,083.00**	3.645

*** Objective Function Value**

Table 5-2 Problem Data Categories

Category Number	Operation Time	Setup Time	α	Category Number	Operation Time	Setup Time	α
1	[1,10]	[1,10]	0.25	**2**	[1,10]	[1,10]	0.5
3	[1,10]	[1,10]	0.75	**4**	[1,10]	[1,10]	1
5	[1,100]	[1,10]	0.25	**6**	[1,100]	[1,10]	0.5
7	[1,100]	[1,10]	0.75	**8**	[1,100]	[1,10]	1
9	[1,100]	[1,100]	0.25	**10**	[1,100]	[1,100]	0.5
11	[1,100]	[1,100]	0.75	**12**	[1,100]	[1,100]	1

Table 5-6 shows that the average CPU time of finding a near-optimal solution for test problems with 1000 jobs is less than 9 seconds which indicates the efficiency of the algorithm. Moreover, the small difference between LB and the objective function of the final solutions with setup time and single server constraints demonstrates the ability of the proposed GA in finding good-quality solutions. The lowest average deviation between the objective function of the problem with single server and LB is 5.55%. This means that the optimal solution of this problem is at most 5.55% less than the objective function proposed by the introduced framework. It should be noted that the highest average deviation between the objective functions of the problem with single server and LB is 21.59% and belongs to the problems with 1000 jobs. Furthermore, Table 5-6 demonstrates that the average deviation between the problem with single server constraints and LB is 11.55%. It should be noted that the standard deviations in all cases are low. This indicates the consistency of the proposed GA in obtaining good-quality solutions. The small deviation between the makespan when single server constraints are added to the problem with setup times confirms that the proposed GA is able to find solutions that single server constraints have slight adversity on their quality. In addition, the deviation between the objective function of the 2-opt algorithm and the proposed GA for the problems with single server constraints is another indication of the efficiency of the algorithm.

Table 5-3 Computational Results for the Problems with 20 and 50 Jobs

Prob. Num.	No. of Jobs	Oper. Time	Setup Time	α	Without Setup		With Setup		With Single Server Setup		APAM+SS	AD(SS,LB)	AD(SS, 2-opt)
					Av. STD	Av. Time	Av. STD	Av. Time	Av. STD	Av. Time			
1	20	[1,10]	[1,10]	0.25	0.50	36.41	0.64	40.06	0.58	48.07	0.96	7.42	19.70
2	20	[1,10]	[1,10]	0.5	1.82	42.46	1.71	46.71	2.00	56.05	2.86	11.65	16.15
3	20	[1,10]	[1,10]	0.75	1.81	42.07	1.69	46.28	1.97	55.54	4.45	10.12	15.25
4	20	[1,10]	[1,10]	1	1.91	42.20	1.52	46.42	2.08	55.71	8.63	15.72	11.08
5	20	[1,100]	[1,10]	0.25	14.03	42.64	14.23	46.91	13.04	56.29	0.17	8.75	24.15
6	20	[1,100]	[1,10]	0.5	13.33	42.48	12.67	46.73	12.95	56.07	0.08	8.50	23.60
7	20	[1,100]	[1,10]	0.75	12.36	42.69	12.90	46.97	12.21	56.36	0.21	11.15	21.70
8	20	[1,100]	[1,10]	1	14.93	42.76	14.06	47.04	12.68	56.45	0.10	9.72	22.06
9	20	[1,100]	[1,100]	0.25	12.92	42.62	13.50	46.89	12.13	56.26	0.78	8.70	18.32
10	20	[1,100]	[1,100]	0.5	13.21	43.22	14.76	47.55	15.19	57.06	3.25	10.44	16.48
11	20	[1,100]	[1,100]	0.75	15.00	42.79	11.02	47.07	14.75	56.49	4.42	11.34	12.39
12	20	[1,100]	[1,100]	1	13.31	42.41	10.20	46.66	13.78	55.99	7.30	13.69	11.43
13	50	[1,10]	[1,10]	0.25	1.83	119.29	1.83	131.22	2.05	157.47	0.90	5.55	22.10
14	50	[1,10]	[1,10]	0.5	2.99	120.44	2.65	132.49	3.17	158.99	3.30	7.95	17.34
15	50	[1,10]	[1,10]	0.75	2.79	120.58	2.66	132.64	3.12	159.17	4.74	9.25	15.90
16	50	[1,10]	[1,10]	1	2.82	120.69	1.89	132.77	3.14	159.32	9.05	14.03	12.58
17	50	[1,100]	[1,10]	0.25	18.05	125.51	18.76	138.07	18.00	165.68	0.06	6.28	27.99
18	50	[1,100]	[1,10]	0.5	19.05	125.4	17.22	138.01	17.30	165.61	0.18	6.78	24.11
19	50	[1,100]	[1,10]	0.75	19.13	125.79	18.42	138.37	20.09	166.05	0.14	6.05	26.84
20	50	[1,100]	[1,10]	1	18.10	125.36	19.33	137.90	19.20	165.48	0.24	6.53	24.85
21	50	[1,100]	[1,100]	0.25	19.02	125.37	17.05	137.92	17.76	165.50	1.03	7.33	22.80
22	50	[1,100]	[1,100]	0.5	18.94	125.35	15.83	137.89	21.42	165.46	2.93	7.56	18.80
23	50	[1,100]	[1,100]	0.75	18.25	125.35	13.01	137.89	18.42	165.46	6.25	11.21	15.16
24	50	[1,100]	[1,100]	1	18.35	125.34	12.22	137.88	21.09	165.45	9.14	15.27	13.75

Table 5-4 Computational Results for the Problems with 100 and 200 Jobs

Prob. Num.	No. of Jobs	Oper. Time	Setup Time	α	Without Setup		With Setup		With Single Server Setup		APAM+SS	AD(SS,LB)	AD(SS, 2-opt)
					Av. STD	Av. Time	Av. STD	Av. Time	Av. STD	Av. Time			
1	100	[1,10]	[1,10]	0.25	3.62	230.77	3.20	253.85	3.59	304.62	1.13	6.09	21.66
2	100	[1,10]	[1,10]	0.5	4.31	243.42	3.82	267.76	4.63	321.31	3.57	8.48	18.81
3	100	[1,10]	[1,10]	0.75	4.49	243.34	3.94	267.67	4.75	321.21	5.04	10.14	16.07
4	100	[1,10]	[1,10]	1	4.18	243.39	2.98	267.73	4.48	321.27	9.67	14.70	12.93
5	100	[1,100]	[1,10]	0.25	29.44	253.18	28.56	278.50	30.32	334.20	0.10	6.31	28.48
6	100	[1,100]	[1,10]	0.5	26.54	253.27	28.68	278.60	30.57	334.32	0.12	7.01	27.45
7	100	[1,100]	[1,10]	0.75	30.19	252.83	29.57	278.11	28.87	333.74	0.13	5.97	27.36
8	100	[1,100]	[1,10]	1	29.35	253.46	27.19	278.81	28.72	334.57	0.24	6.16	26.73
9	100	[1,100]	[1,100]	0.25	27.14	253.00	28.97	278.30	27.54	333.96	1.12	6.01	22.91
10	100	[1,100]	[1,100]	0.5	29.31	253.66	24.91	279.03	32.13	334.83	3.10	8.16	19.92
11	100	[1,100]	[1,100]	0.75	28.46	253.30	21.65	278.63	27.13	334.36	6.02	11.02	16.51
12	100	[1,100]	[1,100]	1	29.00	253.51	18.92	278.86	29.65	334.63	10.17	16.09	13.34
13	200	[1,10]	[1,10]	0.25	6.84	430.39	6.41	473.43	7.13	568.11	1.00	8.02	20.94
14	200	[1,10]	[1,10]	0.5	7.93	476.20	6.82	523.82	7.33	628.58	3.16	8.99	18.11
15	200	[1,10]	[1,10]	0.75	7.31	476.00	6.73	523.60	7.41	628.32	5.41	11.36	15.60
16	200	[1,10]	[1,10]	1	7.19	476.10	5.74	523.71	7.39	628.45	9.26	15.75	12.95
17	200	[1,100]	[1,10]	0.25	49.39	493.40	53.93	542.74	52.80	651.29	0.06	8.06	26.63
18	200	[1,100]	[1,10]	0.5	51.19	492.93	47.99	542.22	53.53	650.67	0.09	7.63	26.82
19	200	[1,100]	[1,10]	0.75	54.88	493.61	53.22	542.97	49.85	651.57	0.17	7.80	26.01
20	200	[1,100]	[1,10]	1	51.42	493.73	44.85	543.10	47.85	651.72	0.20	7.13	25.23
21	200	[1,100]	[1,100]	0.25	56.42	493.12	51.07	542.43	50.59	650.92	1.00	7.47	22.41
22	200	[1,100]	[1,100]	0.5	49.33	493.45	47.62	542.80	51.94	651.35	3.24	9.15	19.09
23	200	[1,100]	[1,100]	0.75	50.34	494.27	47.28	543.70	53.29	652.44	5.93	12.18	15.74
24	200	[1,100]	[1,100]	1	52.53	494.22	40.53	543.64	48.61	652.37	10.16	16.63	12.91

Table 5-5 Computational Results for the Problems with 500 and 1000 Jobs

Prob. Num.	No. of Jobs	Oper. Time	Setup Time	α	Without Setup		With Setup		With Single Server Setup		APAM+SS	AD(SS,LB)	AD(SS, 2-opt)
					Av. STD	Av. Time	Av. STD	Av. Time	Av. STD	Av. Time			
1	500	[1,10]	[1,10]	0.25	13.57	3,563.02	15.78	3,919.32	14.07	4,703.19	0.95	11.12	17.83
2	500	[1,10]	[1,10]	0.5	15.37	2,513.92	14.51	2,765.31	14.46	3,318.37	3.11	12.15	15.29
3	500	[1,10]	[1,10]	0.75	15.54	2,512.02	15.51	2,763.22	15.34	3,315.87	5.12	14.04	13.40
4	500	[1,10]	[1,10]	1	18.79	2,511.90	14.82	2,763.09	13.54	3,315.71	9.16	18.56	11.01
5	500	[1,100]	[1,10]	0.25	137.44	2,578.82	133.69	2,836.70	121.86	3,404.04	0.04	12.63	22.54
6	500	[1,100]	[1,10]	0.5	124.41	2,578.84	136.59	2,836.72	121.36	3,404.07	0.06	12.66	22.24
7	500	[1,100]	[1,10]	0.75	118.20	2,579.32	135.06	2,837.25	123.24	3,404.70	0.11	11.68	22.23
8	500	[1,100]	[1,10]	1	136.56	2,577.08	125.22	2,834.79	126.90	3,401.75	0.24	11.83	21.22
9	500	[1,100]	[1,100]	0.25	130.68	2,579.16	130.97	2,837.08	131.41	3,404.49	0.93	11.70	18.94
10	500	[1,100]	[1,100]	0.5	127.30	2,574.92	123.29	2,832.41	127.75	3,398.89	3.00	12.67	15.91
11	500	[1,100]	[1,100]	0.75	134.38	2,571.56	119.26	2,828.72	127.43	3,394.46	5.84	15.25	13.49
12	500	[1,100]	[1,100]	1	117.41	2,570.58	117.71	2,827.64	127.15	3,393.17	9.56	19.14	11.28
13	1000	[1,10]	[1,10]	0.25	30.44	6,744.28	30.90	7,418.70	32.74	8,902.44	0.76	14.96	14.17
14	1000	[1,10]	[1,10]	0.5	31.70	6,729.28	33.27	7,402.20	34.05	8,882.64	2.95	15.73	12.23
15	1000	[1,10]	[1,10]	0.75	33.12	6,724.85	30.22	7,397.34	32.14	8,876.80	4.68	16.93	11.17
16	1000	[1,10]	[1,10]	1	29.40	6,708.53	34.97	7,379.38	28.43	8,855.25	8.39	20.86	19.17
17	1000	[1,100]	[1,10]	0.25	259.04	6,812.58	262.72	7,493.83	253.02	8,992.60	0.04	18.20	17.35
18	1000	[1,100]	[1,10]	0.5	286.12	6,813.28	258.92	7,494.60	276.26	8,993.52	0.05	17.48	17.46
19	1000	[1,100]	[1,10]	0.75	251.65	6,798.68	255.39	7,478.54	278.01	8,974.25	0.08	17.29	17.43
20	1000	[1,100]	[1,10]	1	276.91	6,807.93	286.79	7,488.72	268.47	8,986.46	0.22	16.98	16.92
21	1000	[1,100]	[1,100]	0.25	269.30	6,797.40	291.87	7,477.14	254.76	8,972.57	0.89	15.77	14.90
22	1000	[1,100]	[1,100]	0.5	262.95	6,792.03	259.28	7,471.23	258.42	8,965.47	2.76	16.19	12.69
23	1000	[1,100]	[1,100]	0.75	296.92	6,789.50	263.73	7,468.45	261.11	8,962.14	5.52	18.58	10.76
24	1000	[1,100]	[1,100]	1	293.54	6,792.33	272.57	7,471.56	285.16	8,965.87	8.82	21.59	19.28

Table 5-6 Summary of the Computational Results for Large Instance Problems

Num.	No. of Jobs	Without Setup		With Setup		With Single Server Setup		APAM +SS	AD(SS, LB)	AD(SS, 2-Opt)
		Av. STD	Av. Time	Av. STD	Av. Time	Av. STD	Av. Time			
1	20	9.59	42.07	9.08	46.27	9.45	55.53	2.77	10.60	17.69
2	50	13.28	123.72	11.74	136.09	13.73	163.30	3.16	8.65	20.19
3	100	20.50	248.93	18.53	273.82	21.03	328.58	3.37	8.85	21.01
4	200	37.06	483.95	34.35	532.35	36.48	638.82	3.31	10.01	20.20
5	500	90.80	2,642.60	90.20	2,906.85	88.71	3,488.25	3.18	13.62	17.12
6	1000	193.42	6,775.89	190.05	7,453.47	188.55	8,944.17	2.93	17.55	15.29
Average		**60.78**	**1,719.53**	**58.99**	**1,891.48**	**59.66**	**2,269.78**	**3.12**	**11.55**	**18.58**

5.5 Conclusion

In this chapter, two-machine, no-wait job shop problem with separable setup times was studied. Further, the single server side constraints were considered. The problem is proven to be strongly NP-hard. Therefore, finding optimal solutions of large instances of the problem is not possible in a reasonable time. Mathematical model of the problem was developed and an effective method for calculating the objective function and assigning the jobs to the machines was introduced.

A metaheuristic algorithm was proposed to deal with the problem. This algorithm is able to search vast areas of the feasible space employing the reasonable time complexity of the method for assigning jobs to machines and calculating the objective function. Computational results on the randomly generated test problems with small and large instances show the efficiency of the proposed algorithm in finding optimal and near optimal solutions for the problems in a very short time.

Chapter 6

General No-Wait Job Shop Problem (NWJS)

6.1 Background

No-Wait Job Shop (NWJS) scheduling problems refer to the set of problems in which a number of jobs are available for processing on a number of machines in a job shop context with the added constraint that there should be no waiting time between consecutive operations of the jobs. Since the problem is in a job shop context, the order of operations to be processed is not necessarily identical for different jobs. No-wait constraints denote that operations of a job have to be processed without interruption on consecutive machines. The considered performance measure is makespan. Following the three-field notation of the scheduling problems, the mentioned problem can be designated as $J \mid no-wait \mid C_{\max}$ [85].

[5] showed that NWJS problems, especially large-instance NWJS problems, are difficult to solve. [58] proved that NWJS is NP-hard in the strong sense even for the problems with two machines. On the other hand, NWJS problem has numerous real-world industrial applications. For example one can name chemical industries [4], food industries [5], steel production [6], pharmaceutical industries [7], and production of concrete products [8]. For a more comprehensive review of the applications of the problem, the reader is referred to [5]. This implies that $J \mid no-wait \mid C_{\max}$ is interesting for the researchers because of its significance in theory and applications in real-world environments.

As a result, NWJS has been studied by many researchers in the past few years. In almost all the studies, NWJS is decomposed into two sub-problems: timetabling and sequencing. The sequencing sub-problem is the problem of searching for a sequence of jobs that provide a better opportunity to find a good-quality solution. This sequence represents the priority of the jobs when they are assigned to a schedule.

Timetabling, an NP-hard problem in NWJS context [77], denotes the problem of developing a feasible schedule from the provided priority list or sequence of the jobs. There exist many algorithms in NWJS literature for timetabling. In this chapter, this important question is aimed to be answered: is the timetabling or the sequencing algorithm more important to the effectiveness of the developed algorithm? To answer this question, different algorithms are developed and combined. The result is a strong algorithm that is able to reproduce the best-known solutions in the literature in a very short time when compared to the competitors in the literature.

The approach is to investigate 3 different sequencing methods: Tabu Search (TS) of section 3.4, a hybrid of Tabu Search with Variable Neighborhood Search (TSVNS) of section 3.4, and a hybrid of Tabu Search with Particle Swarm Optimization (TSPSO) of section 4.4.2. These sequencing algorithms are combined with the most successful timetabling algorithms from the literature; namely, non-delay timetabling of [72], reverse timetabling of [73], shift timetabling of [77], and a new reverse right shift timetabling that is developed in this chapter. Afterwards, these different approaches are applied to a vast number of test problems from the literature. The answer to the above question will be justified using the computational results and statistical analysis.

6.2 Notation

The notations used throughout this chapter is as follows:

S	Source of the graph
T	Final node in the graph
u	An arc
$b(u)$	Beginning node of u
$e(u)$	End node of u
$w(u)$	Label of u
$s(i)$	Successor of node i according to the sequence of operations
$r(i)$	Predecessor of node i according to the sequence of operations
$d(i)$	Successor of node i according to disjunction
$v(i)$	Predecessor of node i according to disjunction
$l(i)$	Label of node i

6.3 Problem Description

A typical NWJS instance includes a set of m machines and n jobs. Every job has its own specific route throughout its processing. Additionally, each job is composed of a series of operations to be performed by different machines. It is assumed that each job should go to each machine exactly once. Processing sequence is a permutation π of the set of jobs, meaning that jobs should be arranged orderly by their index in π. A processing sequence is denoted as a vector $\pi = (\pi_{[1]}, \pi_{[2]}, ..., \pi_{[n]})$. $\pi_{[i]}$ denoted the job that is placed in the ith position of the sequence and is not necessarily equal to π_i. A feasible schedule should also satisfy the following constraints: 1) sequence: each job must be processed according to its predefined order of operations; no interruption/preemption is allowed. 2) synchronicity: no job can be processed by more than one machine at the same time; no machine can process more than one operation at the same time. 3) no-wait: there should be no waiting time between consecutive operations of a job.

Effective algorithms usually decompose NWJS into two sub-problems: timetabling and sequencing. Sequencing sub-problem consists of finding a processing sequence of the optimal schedule. Timetabling sub-problem consists of determining a feasible schedule of the sequence, obtained from sequencing level. As mentioned before, both sub-problems are proved to be NP-hard in the strong sense [60]. NWJS is modeled by means of disjunctive graphs, which is a di-graph. A di-graph consists of a set of nodes V and arcs A. Consider the NWJS instance given in Table 6-1.

Table 6-1 One Instance of NWJS

Jobs	Sequence	Processing Time		
		Machine 1	Machine 2	Machine 3
1	1 2 3	10	5	8
2	1 3 2	5	3	15
3	2 1 3	8	10	8

Figure 6-1 illustrates the non-oriented disjunctive graph of the problem instance given in Table 6-1. In this graph, S and T are dummy nodes that correspond to the start and finish of the whole process. Each row belongs to a specific job. For example, row 1 includes operations 1, 2 and 3, which belong to job 1; row 2 includes operations 4, 5 and 6, which belong to the operations of job 2, and so on. Non-oriented disjunctive arcs are in different textures to demonstrate that their corresponding operations should be processed by a specific machine. In the examples, dashed line demonstrates that the process should be performed by machine 1, dotted line corresponds to machine 2, and dash-dot corresponds to machine 3. Labels on the arcs specify the operating times; for instance label 10 of the arc between nodes 1 and 2 demonstrates the operating time of operation 1 of job 1. Suppose the sequence $\pi = (1,2,3)$ of the above instance. One of the possible orientations of the non-oriented graph of Figure 6-1 comes in Figure 6-2. In the oriented disjunctive graph, each disjunctive arc will have a label as well, which

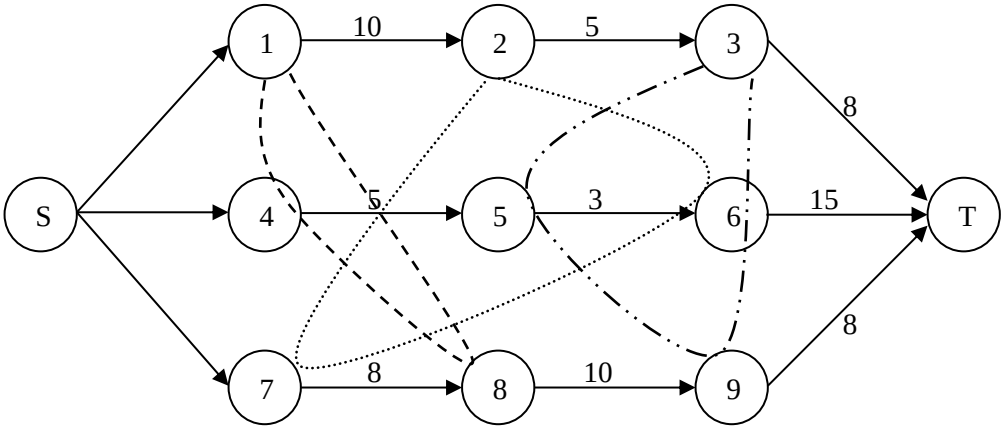

Figure 6-1 Non-Oriented Disjunctive Graph

demonstrates the operating time of the node it is started from. Moreover, according to the orientation given in Figure 6-2, it is possible to label the nodes as well.

The labeled oriented graph is depicted in Figure 6-3. Figure 6-3 represents the starting time of each operation, and since it satisfies all the mentioned constraints, it is thus a feasible schedule with makespan equal to 67.

The Node Labeling Algorithm (NLA) is as follows:

1. Set $l(S)=0$, $l(T)=0$, and $k \leftarrow 1$.
2. Suppose that $\Pi_{[i]}$ is the set of operations of $\pi_{[i]}$; suppose $J=\{i \in V \mid i \in \pi_{[1]}\}$. Set $l(j)=l(s(j))+w(j,s(j)); \forall j \in J$.

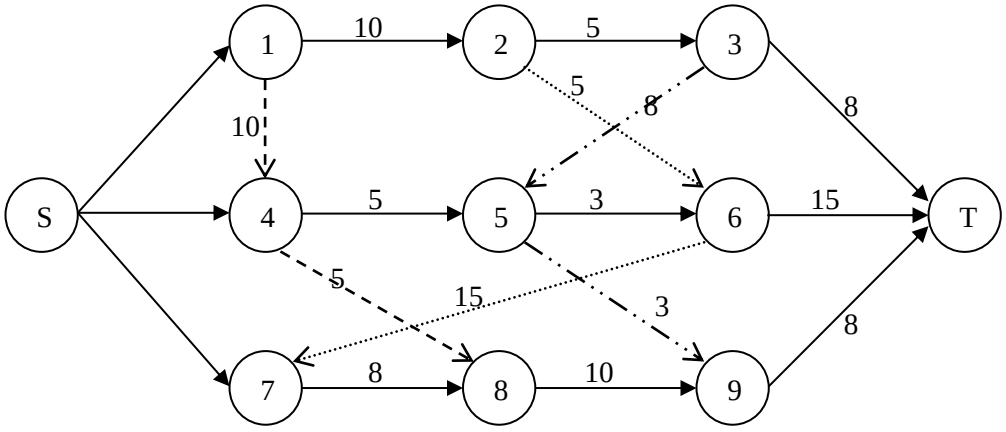

Figure 6-2 Oriented Disjunctive Graph

3. Set $k \leftarrow k+1$.

3.1. Suppose $J=\{i \in V \mid i \in \pi_{[k]}\}$;

3.2. $j=\min\{a \in J\}$

3.3. Calculate:

$$K_1 = l(s(j)) + w(j, s(j))$$

$$K_2 = l(d(j)) + w(j, s(j))$$

$$l(j) = \max\{K_1, K_2\}.$$

If $K_2 > K_1$ and $\exists k \in J \mid k < j$ then:

3.3.1. $p \leftarrow j$

3.3.2. $p_1 = \max\{a \in J \mid a < p\}$

3.3.3. $l(p_1) = l(r(p_1)) - w(p_1, r(p_1))$

3.3.4. $p_1 \leftarrow p_1 - 1$. If $p_1 \in J$, go to step 3.3.3.

3.4. Set $j \leftarrow j+1$.

If $j <= \max\{a \in J\}$, go back to step 3.3; otherwise,

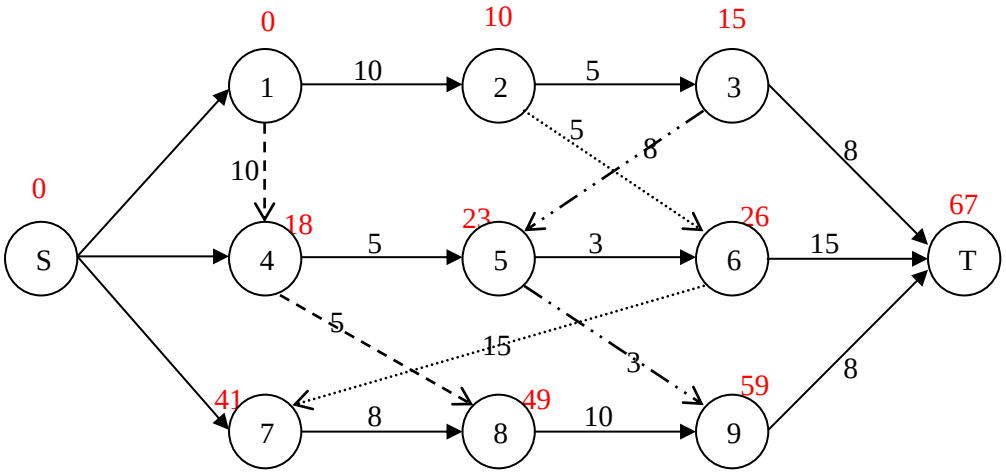

Figure 6-3 Labeled Oriented Disjunctive Graph

$$l(T) = \max\{l(T), l(j) + w(j,T)\}.$$

4. If $k < n$ go back to step 3. Otherwise, stop. $C_{\max} = l(T)$.

The reader can easily verify the correctness of the labeling of Figure 6-3 by calculating the labels according to the above algorithm. One can deduce from the above argument that the most important aspect of a timetabling algorithm is an efficient graph orientation based on a given sequence. [72] defined a non-delay schedule for a NWJS instance. According to this definition, a non-delay schedule can be generated out of a given sequence, if the starting time of each operation is set as early as possible without violating any of the NWJS constraints. [73] enhanced the idea of non-delay timetabling by developing the reverse timetabling algorithm. Reverse timetabling allows the generation of delay timetables; thus explores the solution space of a given schedule more comprehensively. Basic idea behind reverse timetabling is to calculate the non-delay timetabling of the [72] as well as the reverse timetable, and to choose the minimum of the two as the best possible schedule for a given sequence. A reverse timetable can be generated by reversing the sequence as well as reversing the processing route of all the jobs. For example, reverse of the sequence $(1,2,3)$ is $(3,2,1)$; and the reverse of the process route of the instance given in table 1 is $(3,2,1;2,3,1;3,1,2)$. It is proved in [73] that the reverse timetable is a feasible schedule for NWJS; however, generating this schedule from any given sequence might not be possible, when the non-delay timetabling algorithm of [72] is used.

[77] introduced a different timetabling procedure: shift timetabling. Shift timetabling has the freedom of right shifting some of the jobs in the Gantt chart, and then left shifting the rest of the jobs as much as possible in order to generate a new non-delay schedule. Basically, shifting timetabling as introduces by [77] is exchanging the

places of a limited number of jobs in a Gantt chart with the hope that a better non-delay schedule can be obtained. However, this new non-delay schedule might be such that no other sequence is associated with it. Although generating such schedules means missing all delay schedules and limiting the algorithm to the set of non-delay schedules, applying right shift approach might be still beneficial.

In order to exploit the strength of the right shift timetabling, yet benefiting from a search in the set of non-delay as well as delay schedules, Right Shift + Reverse (RSR) timetabling approach is employed. RSR timetabling is a combination of the explained methods. RSR first generates a non-delay schedule based on the definition of [72]. Afterward, a reverse schedule will be generated according to the reverse timetabling of [73]. Then a right shift will be applied to the above two schedules; this will provide four different schedules. Lastly, the algorithm performs a left shift on all four schedules; and the best of the four schedules will be chosen as the schedule that associates the given sequence. In the next section, this approach is described in detail.

6.4 The Proposed Algorithms

Section 6.4.1 describes the proposed timetabling algorithm. The developed algorithms for sequencing come in section 6.4.2.

6.4.1 Right Shift + Reverse (RSR) Algorithm

RSR algorithm, first, generates 4 different timetables: non-delay, reverse, right shift, reverse right shift. Then, a left shifting will be applied to all schedules, and the best would be chosen to associate with the given sequence. The four timetabling algorithms are explained based on the graph modeling. Usage of this modeling significantly simplifies the descriptions.

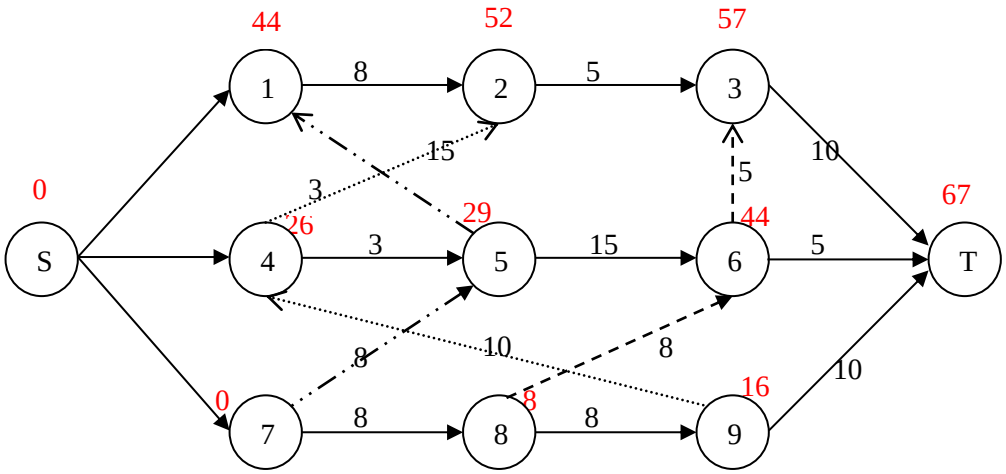

Figure 6-4 Reverse Schedule

6.4.1.1 Non-delay schedule

Given a sequence, non-delay schedule means that all the disjunctive arcs will start from the operations of the first job in the sequence, and end at the operations of the second job in the sequence. Then, they will start from the operations of the second job and end at the operations of the third job, and so on. Figure 6-3 depicts the non-delay schedule for the sequence $(1,2,3)$.

6.4.1.2 Reverse schedule

First the sequence should be reversed. Afterward, the routing of operations of each job in the graph will be reversed; and the labels of the operations will be modified based on the problem information. Then, a non-delay schedule will be performed to this new graph. Figure 6-4 gives the reverse schedule for sequence $(1,2,3)$.

6.4.1.3 Right Shift Schedule

To perform a right shift schedule, scheduling starts with a usual non-delay schedule. However, after scheduling L jobs, job $L+1$ will not be scheduled. Instead, job $L+2$ is scheduled. After scheduling job $L+2$, job $L+1$ will be scheduled and the scheduling continues until the algorithm reaches the job $2L+1$. The rest of the operations do not differ from a non-delay schedule. L is a parameter set by the user.

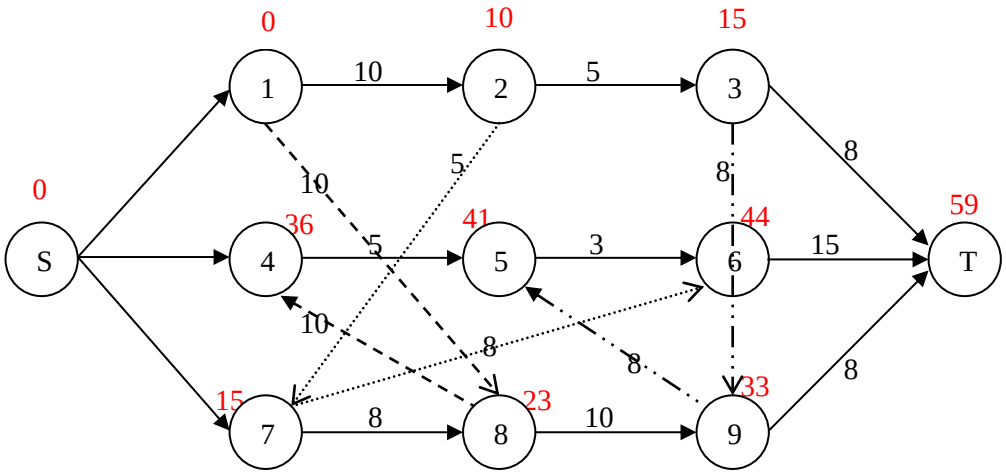

Figure 6-5 Right Shift Schedule

In the graph modeling, the only difference with a non-delay schedule is when the algorithm reaches the job $L+1$. At this point, disjunctive arcs will first enter to the nodes of job $L+2$, and then to the nodes of job $L+1$. Figure 6-5 performs right shift timetabling on sequence $(1,2,3)$, where $L=1$.

6.4.1.4 Reverse Right Shift Schedule

First the reverse timetabling will be applied. Then the right shift timetabling algorithm will be applied to the result of the reverse timetabling. Figure 6-6 performs right shift timetabling on sequence $(1,2,3)$, where $L=1$. As one can verify, these four

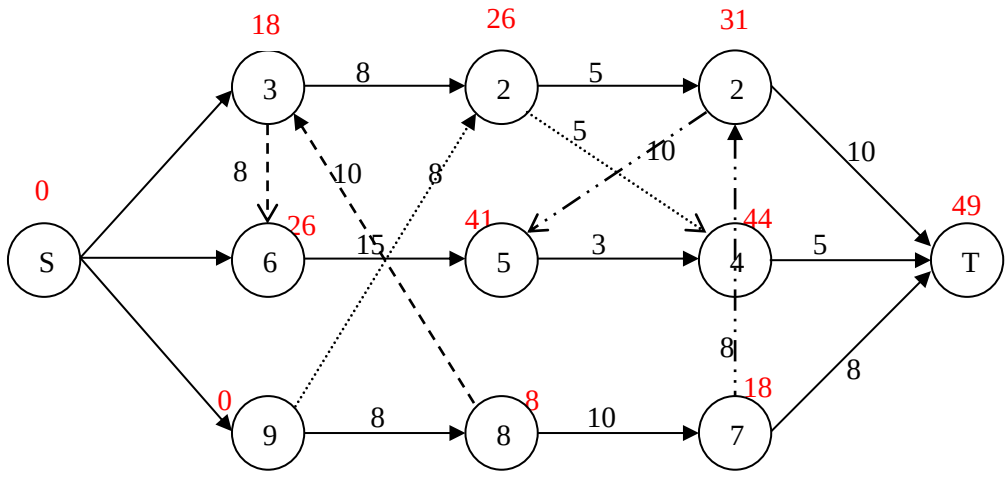

Figure 6-6 Reverse Right Shift Schedule

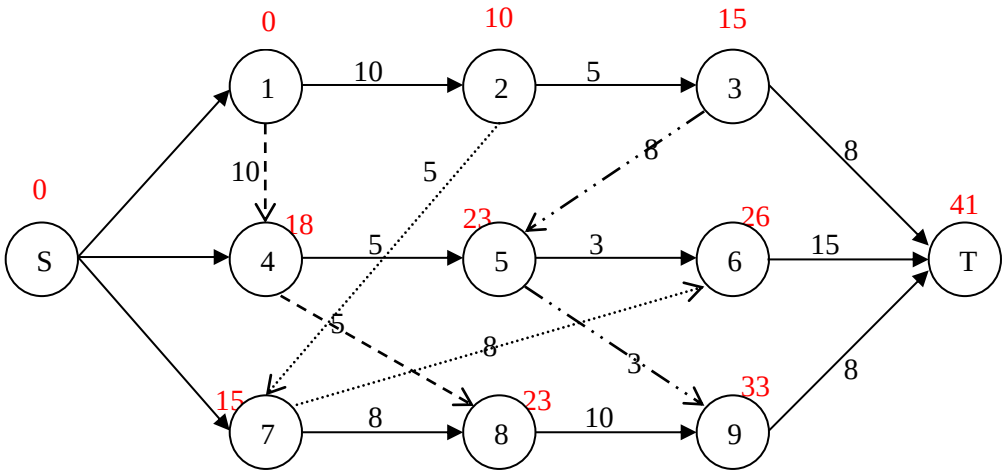

Figure 6-7 Left Shifting Algorithm

different methods have so far achieved four different schedules. And if one wants to choose one of them, the makespan of Figure 6-6 is the best of the four. Section 6.4.1.5 explains the last step of the timetabling approaches, which is left shifting.

6.4.1.5 Left Shifting the Schedules

At this point, left shifting sub-algorithm will be applied to all obtained schedules. Left shifting is a procedure that makes sure that all the operations of all the jobs start as soon as possible without violating the schedule feasibility. According to the graph model, left shifting algorithm can be summarized as follows. First, the longest path in the disjunctive graph is found. Critical Path Method (CPM) or Dijkstra algorithm can be used in order to find this path. In this research Dijkstra algorithm is used [121, 122]. After finding the longest path, a disjunctive arc on this path is chosen randomly and its direction will be changed. The Node Labeling Algorithm (NLA) is performed to the new graph in order to check whether the new graph has a smaller makespan. If the makespan is not shorter, the direction of the arc is changed back to its original direction and another disjunctive arc will be chosen until the makespan improves. This process will

continue until no better disjunctive graph is obtained. Figure 6-7 gives the result of applying left shifting algorithm on the graph of Figure 6-3.

6.4.2 Sequencing Algorithms

So far, four different timetabling algorithms are explained that need to be fed by a sequence. In this research TS of section 3.3.2, hybrid of TS and VNS of section 3.3.2 (or TSVNS), and hybrid of TS and PSO of section 4.2.3 (or TSPSO) are employed to provide such sequences.

6.5 Computational Results and Analysis

Four control parameters should be tuned for the proposed TS to initiate the search; the number of control parameters is 5 for TSVNS. For TSPSO, values of 7 control parameters should be set. The following experimentally derived values are proposed for the TS parameters:

$$\begin{aligned} &Y = n \\ &\theta = \left\lfloor \frac{n}{2} \right\rfloor \\ &L = 4 \\ &R = n \\ &I = 100n \end{aligned} \tag{6-1}$$

For TSVNS, the values are as follows:

$$\begin{aligned} &Y = n \\ &\theta = \left\lfloor \frac{n}{2} \right\rfloor \\ &L = 4 \\ &R = n \\ &r = 3 \\ &I = 100n \end{aligned} \tag{6-2}$$

The values of TSPSO parameters are as follows:

$$
\begin{aligned}
&Y = \left\lfloor \frac{n}{2} \right\rfloor \\
&\theta = \left\lfloor \frac{n}{3} \right\rfloor \\
&I = 100 \times n \\
&L = 4 \\
&x = \begin{cases} 4 & \text{for problems with known optimal solution} \\ 10 & \text{for problems without known optimal solution} \end{cases} \\
&w_{\max} = 1 \\
&w_{\min} = 0.5 \\
&c = 4.1 \\
&b = 10x
\end{aligned}
\tag{6-3}
$$

In (6-1) and (6-2), n is the number of jobs, Y is the length of the adaptive memory or equally the number of initial solutions, θ is the length of the tabu list, L is the right shifting parameter, R is the number of iterations before changing to the next neighborhood structure, r is the parameter of exchange-r neighborhood, and I is the total number of algorithm's iterations. In (6-3), x is the number of jobs that PSO algorithm initiates its search on them, $w_{\min}$ and $w_{\max}$ are used to calculate the inertia weight, c is velocity constant and b is the number of iterations of the PSO algorithm. The coding interface of the proposed algorithm is Microsoft Visual C++ 2008; and the computational experiments were performed on a PC equipped with a 3GHz Intel CPU and 1 GB of RAM.

Test problems can be categorized into two major subsets. First category contains 29 small-instance problems with a known optimal solution. The optimal solution of these problems are known from [65]. Second category contains rather larger-instance problems with unknown optimal solution. All the test problems are accessible from OR-Library [115].

In the problems set, la01 through la40 are from [123], orb01 to orb05 are from [124], and swv01 to swv10 are from [125]. First, the generated solutions from different timetabling and sequencing algorithms are compared with the most competitive approaches in the literature: genetic algorithm of [19] which is referred to as GA, tabu search of [72] which is abbreviated as TS, genetic algorithm and variable neighborhood search of [73] which they tend to call GASA, hybrid tabu search of [76], abbreviated as HTS, and heuristics of [78] which they tend to call tdom.

It should be noted that [76] report two sets of solutions. In the first set, they run their algorithm until they reach to the CPU time of [72]. In the second set, there is no limitation considered on CPU time. Clearly, the results reported to the second set surpass those of the first set. The CPU time is not reported for the second set. The proposed algorithms plus the rest of the algorithms from literature had been limited based on CPU time or the number of iterations. Nevertheless, these solutions, mainly because of their competitive makespans, provide a good base for the comparison purposes of this research. Therefore, these results are used as a reference to compare the generated solutions.

Since the proposed algorithms contain a number of probabilistic elements, each problem is solved 10 times; this will also provide enough replications for statistical analysis that comes later in this section. The minimum makespan obtained for each test problem through these 10 efforts (Min) is reported with its associated CPU time in seconds (Time) as well as Percentage Relative Deviation (PRD). PRD is calculated according to (6-4).

$$PRD^{\text{Algorithm}} = 100 \times \frac{C_{\max}^{\text{Algorithm}} - C_{\max}^{\text{Refernece}}}{\min\left\{C_{\max}^{\text{Algorithm}}, C_{\max}^{\text{Refernece}}\right\}} \tag{6-4}$$

In (6-4), $C_{\max}^{\text{Algorithm}}$ is the minimum proposed makespan of each algorithm, and $C_{\max}^{\text{Refernece}}$ is the same value, extracted from [19]. Table 6-2, Table 6-3 and Table 6-4 respectively compare the computational results of the developed TS, TSVNS, and TSPSO along with the 4 different timetabling problems. The best makespan from the literature along with the paper that report this result also comes in these tables. In terms of CPU time, as one can expect, TS takes the least time and after that it comes TSVNS and then TSPSO. In addition, non-delay timetabling is the least time-consuming algorithm followed by reverse, right shift, and finally reverse right shift. A check on the last rows of these tables suggests that TS algorithm, when combined with reverse timetabling, generates the best makespans. This claim is proved using Design of Experiments (DOE) study which is discussed in details in the next section.

Table 6-2 Computational Results of TS with Different Timetabling Algorithms

				The Proposed TS											
				Non-Delay			Reverse			Right Shift			Reverse Right Shift		
Prob.	Size	Source	Ref.	Makespan	Time	PRD	Makespan	Time	PRD	Makespan	Time	PRD	Makespan	Time	PRD
la01	10*5	GA	971.00	**971.00**	0.87	0.00	**971.00**	1.38	0.00	1,028.00	1.66	5.87	975.00	4.30	0.41
la02	10*5	GA	937.00	990.00	0.83	5.66	**937.00**	1.35	0.00	**937.00**	1.53	0.00	961.00	3.84	2.56
la03	10*5	GA	820.00	862.00	0.76	5.12	**820.00**	1.29	0.00	869.00	1.52	5.98	**820.00**	3.48	0.00
la04	10*5	GA	887.00	911.00	0.92	2.71	**887.00**	1.32	0.00	**887.00**	1.59	0.00	**887.00**	3.76	0.00
la05	10*5	GA	777.00	**777.00**	0.78	0.00	**777.00**	1.31	0.00	787.00	1.54	1.29	**777.00**	3.14	0.00
la06	15*5	GA	1,248.00	1,364.00	1.69	9.29	**1,248.00**	2.83	0.00	1,337.00	3.40	7.13	1,355.00	7.25	8.57
la07	15*5	GA	1,172.00	1,358.00	1.55	15.87	**1,172.00**	2.84	0.00	1,246.00	3.15	6.31	1,257.00	7.41	7.25
la08	15*5	GA	1,244.00	1,304.00	1.82	4.82	**1,244.00**	3.34	0.00	1,320.00	3.75	6.11	1,354.00	8.49	8.84
la09	15*5	GA	1,358.00	1,434.00	1.88	5.60	1,449.00	3.28	6.70	1,456.00	3.84	7.22	1,448.00	8.47	6.63
la10	15*5	GA	1,287.00	**1,287.00**	1.79	0.00	**1,287.00**	3.25	0.00	1,381.00	3.76	7.30	1,355.00	8.49	5.28
la16	10*10	GA	1,575.00	**1,575.00**	2.41	0.00	1,621.00	3.62	2.92	**1,575.00**	4.63	0.00	**1,575.00**	13.02	0.00
la17	10*10	GA	1,371.00	**1,371.00**	2.17	0.00	1,384.00	3.50	0.95	**1,371.00**	4.56	0.00	1,389.00	12.55	1.31
la18	10*10	GA	1,417.00	1,511.00	2.10	6.63	**1,417.00**	3.47	0.00	1,507.00	4.40	6.35	1,507.00	13.01	6.35
la19	10*10	GA	1,482.00	1,548.00	2.40	4.45	**1,482.00**	3.97	0.00	1,586.00	4.94	7.02	1,491.00	13.62	0.61
la20	10*10	GA	1,526.00	1,614.00	2.33	5.77	**1,526.00**	3.82	0.00	**1,526.00**	4.63	0.00	1,614.00	14.40	5.77
orb01	10*10	GA	1,615.00	**1,615.00**	2.26	0.00	1,626.00	3.45	0.68	**1,615.00**	4.44	0.00	**1,615.00**	13.67	0.00
orb02	10*10	GA	1,485.00	1,518.00	2.45	2.22	1,518.00	3.78	2.22	1,518.00	5.01	2.22	1,518.00	13.53	2.22
orb03	10*10	GA	1,599.00	1,621.00	2.09	1.38	**1,599.00**	3.68	0.00	1,603.00	4.28	0.25	**1,599.00**	13.50	0.00
orb04	10*10	GA	1,653.00	1,699.00	2.50	2.78	**1,653.00**	4.26	0.00	1,748.00	5.31	5.75	1,684.00	15.38	1.88
orb05	10*10	GA	1,365.00	1,409.00	2.60	3.22	1,367.00	3.82	0.15	1,409.00	5.25	3.22	1,385.00	13.66	1.47
la11	20*5	tdom	1,619.00	1,760.00	2.75	8.71	**1,619.00**	3.29	0.00	1,683.00	5.71	3.95	1,747.00	15.28	7.91
la12	20*5	tdom	1,414.00	1,541.00	2.78	8.98	1,535.00	5.48	8.56	1,558.00	6.00	10.18	1,558.00	17.33	10.18
la13	20*5	HTS	1,580.00	1,691.00	2.94	7.03	**1,580.00**	6.29	0.00	1,683.00	6.00	6.52	1,693.00	16.91	7.15
la14	20*5	tdom	1,578.00	1,761.00	4.11	11.60	1,722.00	6.89	9.13	1,758.00	6.91	11.41	1,766.00	18.53	11.91
la15	20*5	tdom	1,679.00	1,813.00	2.85	7.98	1,747.00	5.20	4.05	1,797.00	5.89	7.03	1,795.00	17.02	6.91
la21	15*10	tdom	2,030.00	2,185.00	5.35	7.64	**2,030.00**	11.60	0.00	2,198.00	11.34	8.28	2,244.00	34.67	10.54
la22	15*10	tdom	1,852.00	2,042.00	5.30	10.26	1,964.00	9.32	6.05	1,942.00	11.07	4.86	1,960.00	27.56	5.83
la23	15*10	tdom	2,021.00	2,275.00	5.35	12.57	2,073.00	8.73	2.57	2,073.00	10.94	2.57	2,144.00	24.81	6.09
la24	15*10	tdom	1,972.00	2,169.00	5.19	9.99	**1,972.00**	8.57	0.00	2,097.00	10.85	6.34	2,171.00	25.54	10.09
la25	15*10	tdom	1,906.00	2,068.00	5.12	8.50	1,954.00	8.46	2.52	2,068.00	10.42	8.50	2,034.00	28.35	6.72

Table 6-2 Continued

				The Proposed TS											
				Non-Delay			Reverse			Right Shift			Reverse Right Shift		
Prob.	**Size**	**Source**	**Ref.**	**Makespan**	**Time**	**PRD**	**Makespan**	**Time**	**PRD**	**Makespan**	**Time**	**PRD**	**Makespan**	**Time**	**PRD**
la26	20*10	HTS	2,506.00	2,784.00	9.41	11.09	**2,506.00**	17.30	0.00	2,775.00	20.26	10.73	2,774.00	64.90	10.69
la27	20*10	tdom	2,671.00	2,958.00	9.81	10.75	2,821.00	16.80	5.62	2,940.00	20.09	10.07	2,951.00	74.24	10.48
la28	20*10	HTS	2,552.00	**2,552.00**	9.39	0.00	**2,552.00**	16.42	0.00	2,930.00	22.51	14.81	2,909.00	43.80	13.99
la29	20*10	HTS	2,300.00	2,587.00	9.79	12.48	**2,300.00**	17.01	0.00	2,626.00	24.34	14.17	2,641.00	65.83	14.83
la30	20*10	HTS	2,452.00	2,781.00	8.56	13.42	**2,452.00**	14.84	0.00	2,691.00	19.34	9.75	2,821.00	58.59	15.05
la31	30*10	HTS	3,498.00	3,953.00	24.29	13.01	3,713.00	38.01	6.15	3,987.00	53.58	13.98	3,869.00	133.41	10.61
la32	30*10	HTS	3,882.00	4,411.00	23.66	13.63	4,193.00	46.98	8.01	4,304.00	55.75	10.87	4,223.00	119.85	8.78
la33	30*10	HTS	3,454.00	3,733.00	28.49	8.08	**3,454.00**	49.08	0.00	3,945.00	55.86	14.22	3,877.00	118.59	12.25
la34	30*10	HTS	3,659.00	3,924.00	25.33	7.24	3,879.00	44.16	6.01	3,879.00	51.35	6.01	**3,659.00**	120.85	0.00
la35	30*10	HTS	3,552.00	4,031.00	22.31	13.49	3,896.00	40.95	9.68	4,059.00	47.34	14.27	4,011.00	106.70	12.92
la36	15*15	tdom	2,685.00	2,990.00	10.11	11.36	2,815.00	19.14	4.84	2,900.00	21.13	8.01	2,990.00	47.26	11.36
la37	15*15	tdom	2,831.00	3,245.00	11.21	14.62	**2,831.00**	21.18	0.00	3,202.00	26.99	13.10	3,212.00	50.72	13.46
la38	15*15	tdom	2,525.00	2,784.00	10.22	10.26	2,706.00	18.21	7.17	2,734.00	27.74	8.28	2,717.00	52.86	7.60
la39	15*15	tdom	2,660.00	2,876.00	10.99	8.12	2,725.00	16.77	2.44	2,943.00	30.83	10.64	2,842.00	52.43	6.84
la40	15*15	tdom	2,564.00	2,903.00	10.93	13.22	2,845.00	17.94	10.96	2,854.00	33.33	11.31	2,943.00	49.94	14.78
swv01	20*10	HTS	2,318.00	2,429.00	8.97	4.79	2,328.00	14.03	0.43	2,465.00	30.42	6.34	2,441.00	48.67	5.31
swv02	20*10	HTS	2,417.00	2,491.00	7.49	3.06	2,484.00	14.16	2.77	2,533.00	22.95	4.80	2,517.00	46.31	4.14
swv03	20*10	HTS	2,381.00	2,515.00	8.67	5.63	**2,381.00**	14.59	0.00	2,490.00	19.30	4.58	**2,381.00**	70.82	0.00
swv04	20*10	HTS	2,462.00	2,607.00	7.99	5.89	2,520.00	14.98	2.36	2,623.00	19.38	6.54	2,581.00	60.00	4.83
swv05	20*10	HTS	2,333.00	2,529.00	7.94	8.40	**2,333.00**	14.90	0.00	**2,333.00**	19.00	0.00	**2,333.00**	59.03	0.00
swv06	20*15	HTS	3,290.00	3,564.00	15.12	8.33	**3,290.00**	29.33	0.00	3,474.00	33.72	5.59	3,600.00	97.35	9.42
swv07	20*15	HTS	3,188.00	3,453.00	16.67	8.31	**3,188.00**	32.68	0.00	3,441.00	34.65	7.94	3,448.00	115.47	8.16
swv08	20*15	HTS	3,423.00	3,712.00	18.71	8.44	**3,423.00**	34.08	0.00	**3,423.00**	35.26	0.00	3,579.00	108.26	4.56
swv09	20*15	HTS	3,270.00	3,515.00	16.56	7.49	**3,270.00**	29.01	0.00	3,423.00	33.03	4.68	3,454.00	88.04	5.63
swv10	20*15	HTS	3,462.00	3,622.00	15.25	4.62	**3,462.00**	25.84	0.00	**3,462.00**	35.02	0.00	3,600.00	85.41	3.99
Average				**N/A**	**7.74**	**7.17**	**N/A**	**13.67**	**2.05**	**N/A**	**17.30**	**6.41**	**N/A**	**42.91**	**6.40**
Percentage of the Times with Best Solution*				**12.73**			**56.36**			**16.36**			**16.36**		

*Since best solution can occur with more than one algorithm, summation of percentages might be more than 100%.

Table 6-3 Computational Results of TSVNS with Different Timetabling Algorithms

				The Proposed TSVNS											
				Non-Delay			Reverse			Right Shift			Reverse Right Shift		
Prob.	Size	Source	Ref.	Makespan	Time	PRD	Makespan	Time	PRD	Makespan	Time	PRD	Makespan	Time	PRD
la01	10*5	GA	971.00	**971.00**	1.33	0.00	975.00	3.77	0.41	**971.00**	2.53	0.00	975.00	6.02	0.41
la02	10*5	GA	937.00	**937.00**	1.34	0.00	**937.00**	2.21	0.00	**937.00**	2.73	0.00	974.00	5.72	3.95
la03	10*5	GA	820.00	854.00	1.69	4.15	**820.00**	2.40	0.00	862.00	2.74	5.12	**820.00**	5.31	0.00
la04	10*5	GA	887.00	**887.00**	1.61	0.00	**887.00**	2.78	0.00	**887.00**	2.72	0.00	888.00	5.59	0.11
la05	10*5	GA	777.00	793.00	1.41	2.06	781.00	2.34	0.51	781.00	2.39	0.51	781.00	5.18	0.51
la06	15*5	GA	1,248.00	1,301.00	3.39	4.25	1,346.00	6.32	7.85	1,346.00	7.43	7.85	1,322.00	14.55	5.93
la07	15*5	GA	1,172.00	**1,172.00**	2.66	0.00	1,246.00	6.13	6.31	1,232.00	8.20	5.12	1,234.00	12.09	5.29
la08	15*5	GA	1,244.00	1,323.00	3.68	6.35	1,283.00	7.42	3.14	1,326.00	10.55	6.59	1,291.00	14.88	3.78
la09	15*5	GA	1,358.00	1,480.00	2.96	8.98	1,365.00	7.36	0.52	1,461.00	7.16	7.58	1,452.00	15.85	6.92
la10	15*5	GA	1,287.00	1,410.00	2.85	9.56	1,386.00	7.97	7.69	1,328.00	8.79	3.19	1,363.00	15.66	5.91
la16	10*10	GA	1,575.00	1,635.00	3.98	3.81	**1,575.00**	8.46	0.00	**1,575.00**	7.56	0.00	**1,575.00**	18.92	0.00
la17	10*10	GA	1,371.00	1,398.00	3.98	1.97	**1,371.00**	7.57	0.00	1,398.00	7.76	1.97	**1,371.00**	23.29	0.00
la18	10*10	GA	1,417.00	**1,417.00**	3.83	0.00	1,507.00	7.54	6.35	1,507.00	7.02	6.35	1,507.00	21.36	6.35
la19	10*10	GA	1,482.00	1,553.00	4.01	4.79	1,491.00	8.48	0.61	1,580.00	10.61	6.61	1,491.00	22.74	0.61
la20	10*10	GA	1,526.00	1,678.00	3.90	9.96	**1,526.00**	7.80	0.00	1,614.00	8.08	5.77	**1,526.00**	24.47	0.00
orb01	10*10	GA	1,615.00	1,637.00	3.29	1.36	1,638.00	5.84	1.42	1,648.00	7.07	2.04	1,626.00	27.42	0.68
orb02	10*10	GA	1,485.00	1,518.00	3.97	2.22	1,518.00	6.13	2.22	1,518.00	8.11	2.22	1,518.00	35.04	2.22
orb03	10*10	GA	1,599.00	1,617.00	3.58	1.13	**1,599.00**	3.58	0.00	1,603.00	7.64	0.25	**1,599.00**	28.57	0.00
orb04	10*10	GA	1,653.00	1,738.00	5.14	5.14	1,712.00	6.16	3.57	**1,653.00**	8.94	0.00	1,699.00	38.72	2.78
orb05	10*10	GA	1,365.00	1,409.00	5.22	3.22	1,367.00	6.20	0.15	1,438.00	8.88	5.35	1,367.00	24.34	0.15
la11	20*5	tdom	1,619.00	**1,619.00**	4.33	0.00	1,768.00	11.66	9.20	**1,619.00**	9.92	0.00	1,732.00	26.11	6.98
la12	20*5	tdom	1,414.00	1,539.00	4.78	8.84	1,578.00	11.18	11.60	1,545.00	10.18	9.26	1,609.00	25.46	13.79
la13	20*5	HTS	1,580.00	1,743.00	5.21	10.32	1,696.00	10.80	7.34	**1,580.00**	10.33	0.00	1,713.00	24.95	8.42
la14	20*5	tdom	1,578.00	1,766.00	5.59	11.91	1,711.00	13.31	8.43	**1,578.00**	11.10	0.00	**1,578.00**	22.29	0.00
la15	20*5	tdom	1,679.00	1,834.00	5.04	9.23	1,816.00	9.27	8.16	1,811.00	10.10	7.86	1,828.00	25.79	8.87
la21	15*10	tdom	2,030.00	2,138.00	9.23	5.32	2,265.00	17.65	11.58	2,242.00	17.57	10.44	2,177.00	48.57	7.24
la22	15*10	tdom	1,852.00	1,943.00	9.25	4.91	1,972.00	14.21	6.48	1,981.00	17.11	6.97	**1,852.00**	76.30	0.00
la23	15*10	tdom	2,021.00	2,231.00	10.50	10.39	2,139.00	17.41	5.84	2,187.00	16.97	8.21	2,202.00	72.00	8.96
la24	15*10	tdom	1,972.00	2,180.00	9.50	10.55	2,214.00	14.20	12.27	2,092.00	16.35	6.09	2,097.00	65.96	6.34
la25	15*10	tdom	1,906.00	2,041.00	8.56	7.08	2,082.00	13.05	9.23	2,070.00	16.04	8.60	**1,906.00**	48.91	0.00

Table 6-3 Continued

				The Proposed TSVNS											
				Non-Delay			Reverse			Right Shift			Reverse Right Shift		
Prob.	**Size**	**Source**	**Ref.**	**Makespan**	**Time**	**PRD**	**Makespan**	**Time**	**PRD**	**Makespan**	**Time**	**PRD**	**Makespan**	**Time**	**PRD**
la26	20*10	HTS	2,506.00	2,706.00	16.90	7.98	2,634.00	25.16	5.11	2,738.00	30.48	9.26	2,757.00	82.60	10.02
la27	20*10	tdom	2,671.00	2,963.00	17.09	10.93	2,967.00	27.27	11.08	2,889.00	29.86	8.16	2,986.00	84.28	11.79
la28	20*10	HTS	2,552.00	2,807.00	16.98	9.99	**2,552.00**	30.17	0.00	2,847.00	35.92	11.56	2,831.00	91.99	10.93
la29	20*10	HTS	2,300.00	2,654.00	18.44	15.39	2,628.00	32.56	14.26	2,632.00	35.59	14.43	2,619.00	87.53	13.87
la30	20*10	HTS	2,452.00	2,768.00	15.42	12.89	**2,452.00**	29.41	0.00	2,862.00	34.15	16.72	2,837.00	90.07	15.70
la31	30*10	HTS	3,498.00	4,022.00	51.05	14.98	3,923.00	66.12	12.15	3,833.00	74.76	9.58	3,855.00	185.78	10.21
la32	30*10	HTS	3,882.00	4,128.00	65.99	6.34	4,202.00	71.33	8.24	4,313.00	105.09	11.10	4,325.00	195.38	11.41
la33	30*10	HTS	3,454.00	3,824.00	70.49	10.71	3,868.00	78.45	11.99	3,842.00	101.18	11.23	3,826.00	200.43	10.77
la34	30*10	HTS	3,659.00	3,877.00	75.55	5.96	3,930.00	73.15	7.41	4,033.00	85.21	10.22	4,063.00	182.01	11.04
la35	30*10	HTS	3,552.00	4,028.00	49.36	13.40	**3,552.00**	69.76	0.00	3,988.00	75.43	12.27	**3,552.00**	273.01	0.00
la36	15*15	tdom	2,685.00	2,963.00	22.70	10.35	**2,685.00**	30.06	0.00	2,967.00	33.87	10.50	2,984.00	109.28	11.14
la37	15*15	tdom	2,831.00	3,244.00	22.88	14.59	**2,831.00**	31.16	0.00	3,096.00	36.46	9.36	3,236.00	91.77	14.31
la38	15*15	tdom	2,525.00	2,842.00	18.57	12.55	2,734.00	31.80	8.28	2,687.00	35.14	6.42	2,851.00	105.42	12.91
la39	15*15	tdom	2,660.00	2,969.00	18.26	11.62	2,866.00	29.60	7.74	2,965.00	38.34	11.47	2,866.00	113.03	7.74
la40	15*15	tdom	2,564.00	2,952.00	19.16	15.13	2,716.00	35.64	5.93	2,594.00	38.68	1.17	2,848.00	136.10	11.08
swv01	20*10	HTS	2,318.00	2,504.00	17.80	8.02	2,443.00	22.35	5.39	2,432.00	28.93	4.92	2,422.00	95.07	4.49
swv02	20*10	HTS	2,417.00	2,540.00	16.04	5.09	2,593.00	22.29	7.28	2,532.00	27.51	4.76	2,524.00	87.38	4.43
swv03	20*10	HTS	2,381.00	2,543.00	12.75	6.80	2,521.00	23.67	5.88	2,556.00	29.67	7.35	2,534.00	96.87	6.43
swv04	20*10	HTS	2,462.00	2,551.00	15.30	3.61	2,638.00	23.27	7.15	2,549.00	29.86	3.53	2,574.00	82.78	4.55
swv05	20*10	HTS	2,333.00	2,487.00	13.62	6.60	2,493.00	25.14	6.86	2,589.00	28.73	10.97	2,530.00	67.24	8.44
swv06	20*15	HTS	3,290.00	3,458.00	25.53	5.11	**3,290.00**	29.33	0.00	3,596.00	60.93	9.30	3,537.00	140.05	7.51
swv07	20*15	HTS	3,188.00	3,439.00	29.37	7.87	3,339.00	32.68	4.74	3,459.00	69.91	8.50	3,338.00	178.54	4.71
swv08	20*15	HTS	3,423.00	3,710.00	30.95	8.38	**3,423.00**	34.08	0.00	3,699.00	81.75	8.06	**3,423.00**	231.22	0.00
swv09	20*15	HTS	3,270.00	3,391.00	30.35	3.70	**3,270.00**	29.01	0.00	3,444.00	74.20	5.32	3,376.00	184.19	3.24
swv10	20*15	HTS	3,462.00	3,604.00	28.06	4.10	**3,462.00**	25.84	0.00	3,761.00	52.56	8.64	3,717.00	180.94	7.37
Average				**N/A**	**15.53**	**6.90**	**N/A**	**21.43**	**4.73**	**N/A**	**28.12**	**6.34**	**N/A**	**75.91**	**5.82**
Percentage of the Times with Best Solution*				**10.91**			**29.09**			**14.55**			**18.18**		

*Since best solution can occur with more than one algorithm, summation of percentages might be more than 100%.

Table 6-4 Computational Results of TSPSO with Different Timetabling Algorithms

				The Proposed TSPSO											
				Non-Delay			Reverse			Right Shift			Reverse Right Shift		
Prob.	**Size**	**Source**	**Ref.**	**Makespan**	**Time**	**PRD**	**Makespan**	**Time**	**PRD**	**Makespan**	**Time**	**PRD**	**Makespan**	**Time**	**PRD**
la01	10*5	GA	971.00	975	0.86	0.41	**971**	1.47	0.00	1,031.00	0.87	6.18	1,016.00	3.89	4.63
la02	10*5	GA	937.00	966	0.82	3.09	**937**	2.06	0.00	961	1.08	2.56	988	4.1	5.44
la03	10*5	GA	820.00	854	0.79	4.15	829	1.3	1.10	862	1	5.12	843	4.58	2.80
la04	10*5	GA	887.00	**887**	0.87	0.00	**887**	1.79	0.00	**887**	1.24	0.00	888	3.93	0.11
la05	10*5	GA	777.00	781	0.86	0.51	**777**	2.01	0.00	784	1.16	0.90	**777**	3.58	0.00
la06	15*5	GA	1,248.00	1,353.00	1.67	8.41	1,348.00	3.66	8.01	**1,248.00**	2.46	0.00	1,362.00	8.48	9.13
la07	15*5	GA	1,172.00	1,252.00	1.65	6.83	1,280.00	3.92	9.22	1,276.00	3	8.87	1,251.00	10.2	6.74
la08	15*5	GA	1,244.00	1,318.00	1.9	5.95	1,334.00	4.01	7.23	1,314.00	2.98	5.63	1,289.00	11.5	3.62
la09	15*5	GA	1,358.00	1,425.00	1.85	4.93	1,442.00	4.21	6.19	1,447.00	2.59	6.55	1,429.00	11.65	5.23
la10	15*5	GA	1,287.00	1,314.00	1.93	2.10	1,363.00	4.85	5.91	1,371.00	2.24	6.53	1,327.00	15.05	3.11
la16	10*10	GA	1,575.00	1,635.00	3.82	3.81	**1,575.00**	3.76	0.00	1,665.00	2.63	5.71	**1,575.00**	11.31	0.00
la17	10*10	GA	1,371.00	1,459.00	2.82	6.42	1,398.00	3.92	1.97	1,430.00	3.14	4.30	1,436.00	11.83	4.74
la18	10*10	GA	1,417.00	**1,417.00**	2.56	0.00	1,507.00	3.62	6.35	**1,417.00**	3.55	0.00	**1,417.00**	11.89	0.00
la19	10*10	GA	1,482.00	1,581.00	2.72	6.68	1,497.00	4.37	1.01	1,569.00	3.01	5.87	1,512.00	11.19	2.02
la20	10*10	GA	1,526.00	**1,526.00**	2.7	0.00	1,608.00	3.91	5.37	**1,526.00**	2.98	0.00	**1,526.00**	11.37	0.00
orb01	10*10	GA	1,615.00	**1,615.00**	2.3	0.00	1,626.00	4.59	0.68	1,637.00	2.66	1.36	**1,615.00**	11.55	0.00
orb02	10*10	GA	1,485.00	**1,485.00**	2.7	0.00	1,518.00	4.74	2.22	1,518.00	3.5	2.22	1,518.00	12.73	2.22
orb03	10*10	GA	1,599.00	**1,599.00**	2.32	0.00	**1,599.00**	4.19	0.00	**1,599.00**	2.41	0.00	**1,599.00**	14.11	0.00
orb04	10*10	GA	1,653.00	1,738.00	2.45	5.14	1,712.00	4.69	3.57	1,684.00	3.53	1.88	**1,653.00**	14.55	0.00
orb05	10*10	GA	1,365.00	1,415.00	2.76	3.66	1,367.00	4.64	0.15	**1,365.00**	3.83	0.00	**1,365.00**	13.89	0.00
la11	20*5	tdom	1,619.00	1,784.00	2.83	10.19	1,760.00	7.36	8.71	1,782.00	3.56	10.07	1,686.00	23.22	4.14
la12	20*5	tdom	1,414.00	1,545.00	3.61	9.26	**1,414.00**	6.26	0.00	1,566.00	3.5	10.75	1,569.00	22.91	10.96
la13	20*5	HTS	1,580.00	1,670.00	3.82	5.70	1,744.00	5.87	10.38	1,725.00	3.81	9.18	1,717.00	23.62	8.67
la14	20*5	tdom	1,578.00	1,815.00	4.64	15.02	**1,578.00**	6.6	0.00	1,776.00	3.85	12.55	1,799.00	25.8	14.01
la15	20*5	tdom	1,679.00	1,772.00	4.12	5.54	1,780.00	6.93	6.02	1,752.00	3.27	4.35	1,781.00	15.46	6.08
la21	15*10	tdom	2,030.00	2,163.00	8.06	6.55	2,101.00	10.87	3.50	2,151.00	6.44	5.96	2,251.00	26.61	10.89
la22	15*10	tdom	1,852.00	2,007.00	8.01	8.37	1,983.00	10.53	7.07	2,019.00	6.9	9.02	**1,852.00**	27.16	0.00
la23	15*10	tdom	2,021.00	2,159.00	8.38	6.83	2,157.00	10.57	6.73	2,154.00	7.7	6.58	2,175.00	27.24	7.62
la24	15*10	tdom	1,972.00	2,074.00	8.89	5.17	2,148.00	9.71	8.92	2,108.00	6.03	6.90	2,056.00	27.11	4.26
la25	15*10	tdom	1,906.00	2,034.00	6.87	6.72	2,123.00	9.26	11.39	2,029.00	5.79	6.45	2,021.00	24.88	6.03

Table 6-4 Continued

				The Proposed TSPSO											
				Non-Delay			Reverse			Right Shift			Reverse Right Shift		
Prob.	Size	Source	Ref.	Makespan	Time	PRD	Makespan	Time	PRD	Makespan	Time	PRD	Makespan	Time	PRD
la26	20*10	HTS	2,506.00	2,828.00	11.75	12.85	2,773.00	17.61	10.65	2,765.00	10.73	10.34	2,709.00	56.76	8.10
la27	20*10	tdom	2,671.00	2,975.00	11.14	11.38	3,074.00	20.59	15.09	2,972.00	12.83	11.27	2,955.00	52.92	10.63
la28	20*10	HTS	2,552.00	2,837.00	11.67	11.17	2,715.00	20.52	6.39	2,926.00	14	14.66	**2,552.00**	46.68	0.00
la29	20*10	HTS	2,300.00	2,645.00	11.13	15.00	**2,300.00**	25.76	0.00	2,640.00	10.52	14.78	2,620.00	52.24	13.91
la30	20*10	HTS	2,452.00	2,808.00	10.22	14.52	**2,452.00**	25.05	0.00	2,812.00	10.8	14.68	2,845.00	46.36	16.03
la31	30*10	HTS	3,498.00	3,900.00	26.85	11.49	**3,498.00**	79.68	0.00	4,017.00	26.11	14.84	3,866.00	123.5	10.52
la32	30*10	HTS	3,882.00	4,334.00	38.91	11.64	**3,882.00**	67.7	0.00	**3,882.00**	28.62	0.00	4,289.00	131.92	10.48
la33	30*10	HTS	3,454.00	3,840.00	32.06	11.18	**3,454.00**	50.94	0.00	3,835.00	26.75	11.03	3,883.00	170.48	12.42
la34	30*10	HTS	3,659.00	3,878.00	31.87	5.99	3,820.00	47.05	4.40	3,928.00	28.75	7.35	3,949.00	115.54	7.93
la35	30*10	HTS	3,552.00	4,120.00	29.17	15.99	**3,552.00**	43.17	0.00	3,951.00	25.28	11.23	3,880.00	112.29	9.23
la36	15*15	tdom	2,685.00	2,902.00	10.73	8.08	2,885.00	17.14	7.45	2,857.00	10.67	6.41	2,884.00	52.81	7.41
la37	15*15	tdom	2,831.00	3,129.00	11.91	10.53	**2,831.00**	18.83	0.00	3,189.00	11.64	12.65	3,190.00	62.56	12.68
la38	15*15	tdom	2,525.00	2,785.00	11.36	10.30	2,757.00	17.85	9.19	**2,525.00**	12.5	0.00	2,748.00	66.04	8.83
la39	15*15	tdom	2,660.00	2,898.00	10.9	8.95	2,874.00	17.46	8.05	2,942.00	12.58	10.60	2,884.00	73.33	8.42
la40	15*15	tdom	2,564.00	2,866.00	13.43	11.78	**2,564.00**	19.26	0.00	2,716.00	13.32	5.93	2,917.00	63.04	13.77
swv01	20*10	HTS	2,318.00	2,437.00	8.2	5.13	2,423.00	15.23	4.53	2,491.00	9.46	7.46	2,493.00	46.58	7.55
swv02	20*10	HTS	2,417.00	2,514.00	9.37	4.01	2,482.00	14.48	2.69	2,519.00	8.66	4.22	2,533.00	46.57	4.80
swv03	20*10	HTS	2,381.00	2,549.00	8.78	7.06	2,472.00	17.25	3.82	2,532.00	9.72	6.34	2,457.00	49.53	3.19
swv04	20*10	HTS	2,462.00	2,637.00	9.55	7.11	2,560.00	16.7	3.98	2,582.00	10.08	4.87	2,600.00	51.45	5.61
swv05	20*10	HTS	2,333.00	2,509.00	9.16	7.54	2,500.00	16.54	7.16	2,535.00	9.96	8.66	2,538.00	45.56	8.79
swv06	20*15	HTS	3,290.00	3,589.00	18.3	9.09	3,448.00	28.28	4.80	3,606.00	20.45	9.60	3,321.00	95.61	0.94
swv07	20*15	HTS	3,188.00	3,407.00	18.38	6.87	3,464.00	31.88	8.66	3,440.00	20.86	7.90	3,389.00	103	6.30
swv08	20*15	HTS	3,423.00	3,701.00	20.26	8.12	3,680.00	40.43	7.51	3,543.00	20.49	3.51	3,830.00	120.38	11.89
swv09	20*15	HTS	3,270.00	3,440.00	20.5	5.20	3,391.00	33.17	3.70	3,510.00	18.84	7.34	3,524.00	91.25	7.77
swv10	20*15	HTS	3,462.00	3,611.00	16.51	4.30	3,667.00	33.1	5.92	3,724.00	18.03	7.57	3,712.00	90.63	7.22
Average				**N/A**	**9.30**	**6.85**	**N/A**	**16.21**	**4.28**	**N/A**	**9.13**	**6.52**	**N/A**	**43.86**	**6.13**
Percentage of the Times with Best Solution*				**10.91**			**29.09**			**14.55**			**18.18**		

*Since best solution can occur with more than one algorithm, summation of percentages might be more than 100%.

6.5.1 Statistical Analysis Using Design of Experiments

Design of Experiments (DOE) is a statistical method that is used to analyze the effect of some variables (or factors) on a process performance (response). The most important feature of using DOE is its ability to study the interaction effects between the considered factors. This study tends to analyze the effect of applying different sequencing and timetabling algorithms on the quality of the obtained solutions of NWJS problems. The quality of the obtained solutions can be measured either using makespan or equivalently, PRD of NWJS problems.

This research also wants to determine the best combination of sequencing and timetabling algorithms when a NWJS problem is solved. In order to perform such analysis, one should identify the potential factors that may impact the algorithms' performance. Therefore, the following factors are considered 1) test problem, 2) sequencing algorithm, and 3) timetabling algorithm. The PRD is used as the system's response. The analysis is done in order to verify the effect of each factor as well as their interactions on the resulted PRD values of the obtained solutions. Since each test

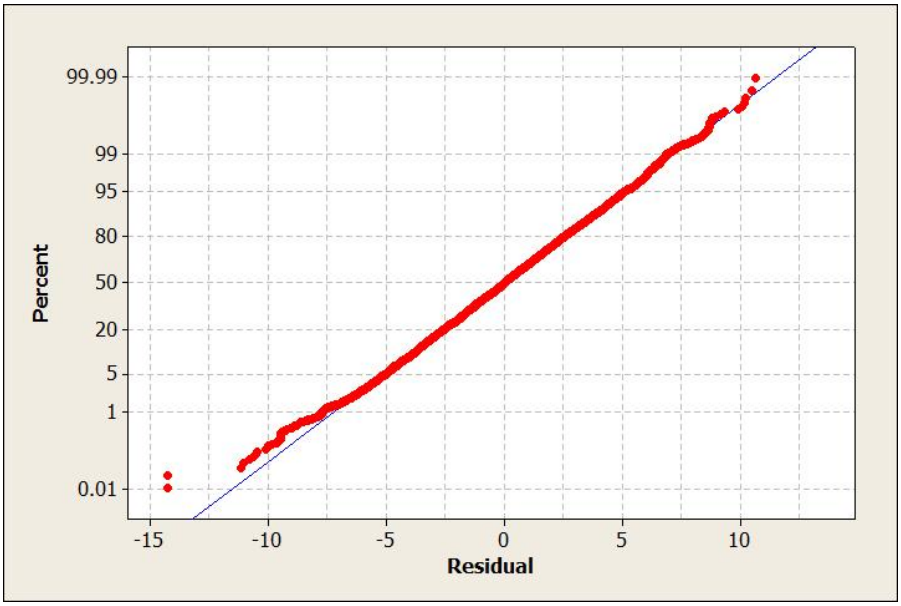

Figure 6-8 Normal Probability Plot of Residuals

problem is solved 10 times independently, 10 replications are considered for each combination of the above factors. The first factor is the test problem. This factor has 55 levels, which is equal to the number of test problems considered. Second factor, sequencing algorithm, has 3 levels (TS, TSVNS, TSPSO). Third factor, timetabling algorithm, has 4 levels (non-delay, reverse, right shift, reverse right shift). Hence, the total number of runs is 6600 (55x3x4x10). The analysis is performed at 5% significance level using the Minitab software.

In order for variance analysis to be valid, residuals should follow a normal distribution. Figure 8 illustrates the normal probability plot of residuals. Residual values should be close to the normal probability line in order to deduce that the residuals show normal distribution characteristics. Figure 8 confirms that the residuals are very close to the normal line.

All in all, Figure 6-8 verifies the normality assumption as well as the validity of the variance analyses tests. Accordingly, null hypothesis of the designed ANOVA is that: 1) there is no meaningful and significant difference between the proposed sequencing and timetabling algorithms or studied problems, and differences in generated makespan values are as a result of chance; 2) there is no two-way interaction between sequencing and timetabling, sequencing and problems, or timetabling and problems; 3) there is no three-way interaction between sequencing, timetabling, and problems.

Alternative hypothesis is: null hypothesis is not true. Table 6-5 summarizes the ANOVA results for the described model.

Table 6-5 Analysis of Variance for PRD, using Adjusted SS for Tests

Source	Degree of Freedom	Sequential Sums of Squares	Adjusted Sums of Squares	Adjusted Mean Square Value	F-Value	P-Value
Problem	54	108746.7	108746.7	2013.8	204.42	0.000
Sequencing	2	94.7	94.7	47.3	4.8	0.008
Timetabling	3	2248.9	2248.9	749.6	76.09	0
Problem * Sequencing	108	1752	1752	16.2	1.65	0
Problem * Timetabling	162	3380.1	3380.1	20.9	2.12	0
Sequencing * Timetabling	6	1556.2	1556.2	259.4	26.33	0
Problem * Sequencing * Timetabling	324	5016.2	5016.2	15.5	1.57	0
Error	5940	58517.7	58517.7	9.9		
Total	6599	181312.4				
	$R^2 = 67.73\%$		$R^2_{adj} = 64.14\%$			

P-values of Table 6-5 indicate that all the sources of variation along with their interactions have a significant effect on the PRDs. This means that not only developing progressive sequencing and timetabling algorithms is important, but also one should pay attention to their interaction with each other and their interaction with the solved problems. In other words, when the right timetabling and sequencing algorithms are combined to solve certain problems, there will be 3 effective factors contributing to the quality of the solutions: the sequencing algorithm, the timetabling algorithm, and the

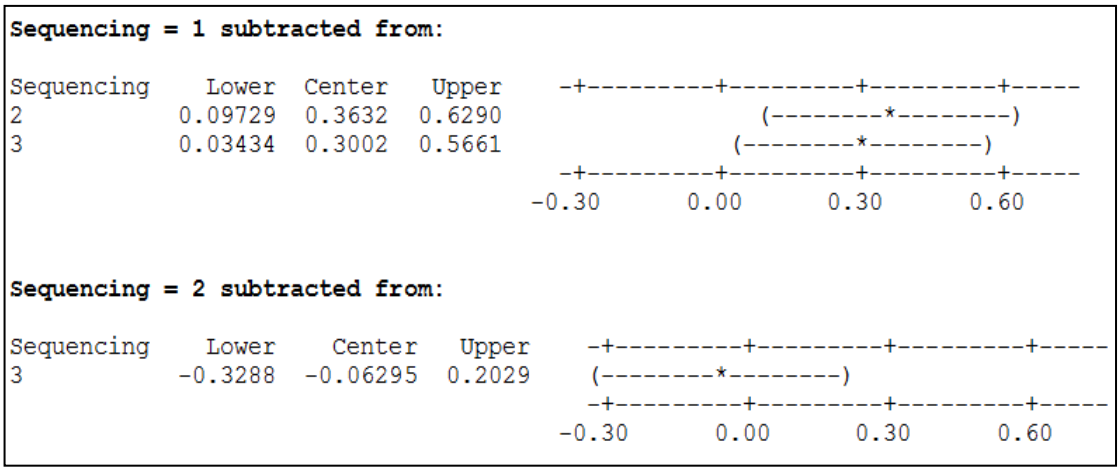

Figure 6-9 Tukey 95.0% Simultaneous Confidence Intervals; Response Variable PRD; All Pairwise Comparisons among Levels of Sequencing

positive interaction that such combination creates. On the other hand, if either the sequencing or the timetabling algorithms are not carefully selected, the quality of the solutions will be affected not only because of the inappropriate choice, but also by the negative interaction that exists between the two. The values of R^2 and R^2_{adj} are both more than 64%. This means that the used DOE model is able to address more than 64% of the variations between the different PRD values, which is a high value for practical purposes; and indicates the adequacy of model in describing the reality.

In order to determine the superior sequencing and timetabling algorithms, Tukey's simultaneous test with 95% confidence level is performed. The simultaneous confidence intervals are depicted in Figure 6-9. It should be noted that in sequencing,

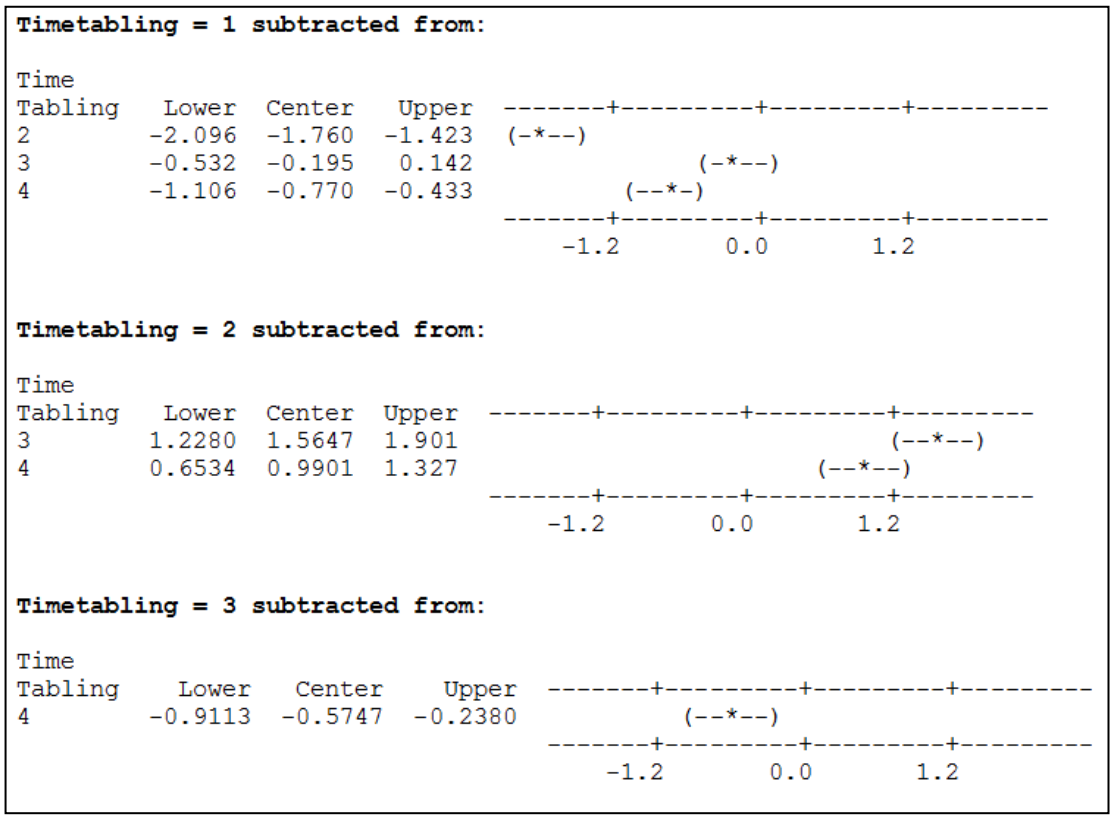

Figure 6-10 Tukey 95.0% Simultaneous Confidence Intervals; Response Variable PRD; All Pairwise Comparisons among Levels of Timetabling

code 1 stands for TS, 2 denotes TSVNS, and 3 represents TSPSO; in timetabling code 1 to 4 indicate non-delay, reverse, right shift, and reverse right shift, respectively. Figure 6-9 depicts that while there is no significant difference between TSVNS and TSPSO, a significant difference between TS algorithm and TSVNS and TSPSO exists; TS is superior to the other sequencing algorithms.

Figure 6-10 illustrates the results of Tukey's simultaneous test with 95% confidence levels on timetabling algorithms. Using this figure, one can deduce the order of superiority of timetabling algorithms as reverse, reverse right shift, and indifferent between non-delay and right shift, which is very different from the intuitive deductions. This means that although right shift algorithm seems to explore areas of the feasible region that the rest of the algorithms do not explore, statistically, there is no significant difference between PRD values of this algorithm and non-delay algorithm, which seems

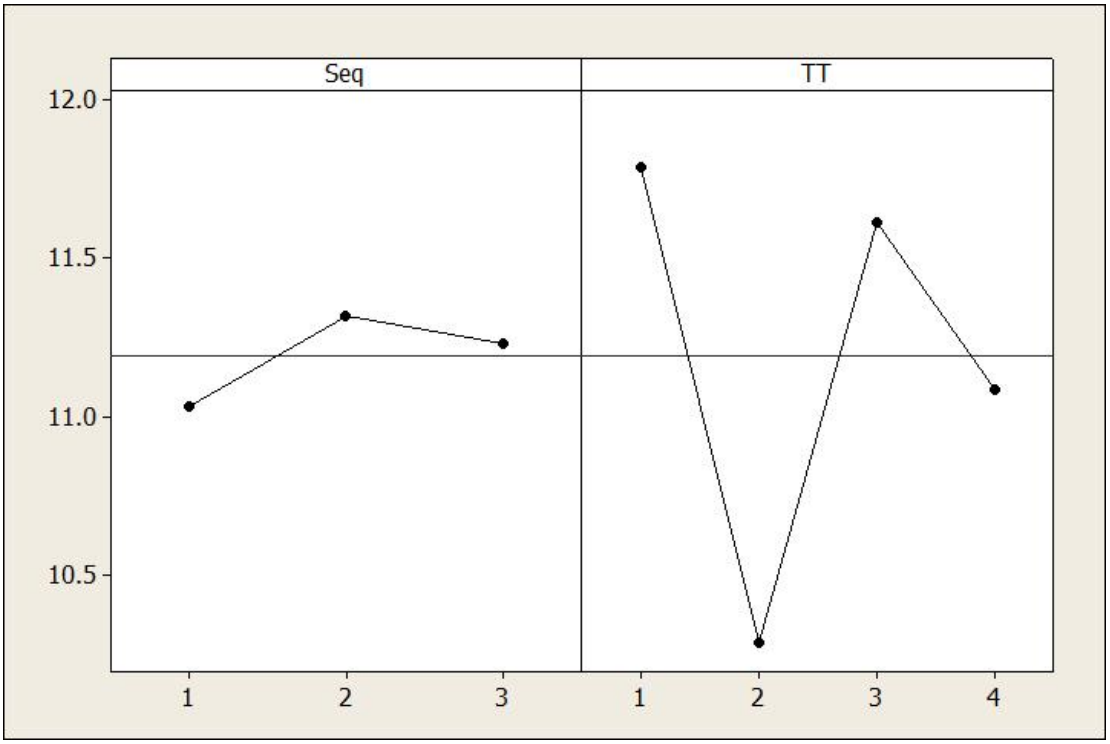

Figure 6-11 Main Effects Plot for PRD

to be the simplest timetabling procedure. Main effects plot of Figure 6-11 further demonstrates the correctness of the above argument.

Last but not least is the interaction between different factors. As Table 6-5 indicates, there is a significant interaction between the 3 mentioned factors. Analysis of interaction in Figure 6-12 reveals that PRD values of the problems under-the-study show a considerable improvement when sequencing algorithms are combined with reverse timetabling algorithm. The best PRD values for the problems under-the-study can be obtained by combining TS with reverse algorithm. When non-delay algorithm is used for timetabling, TSPSO is the best sequencing method. TSPSO is the most successful sequencing algorithm when reverse right shift timetabling is used. For the case of right

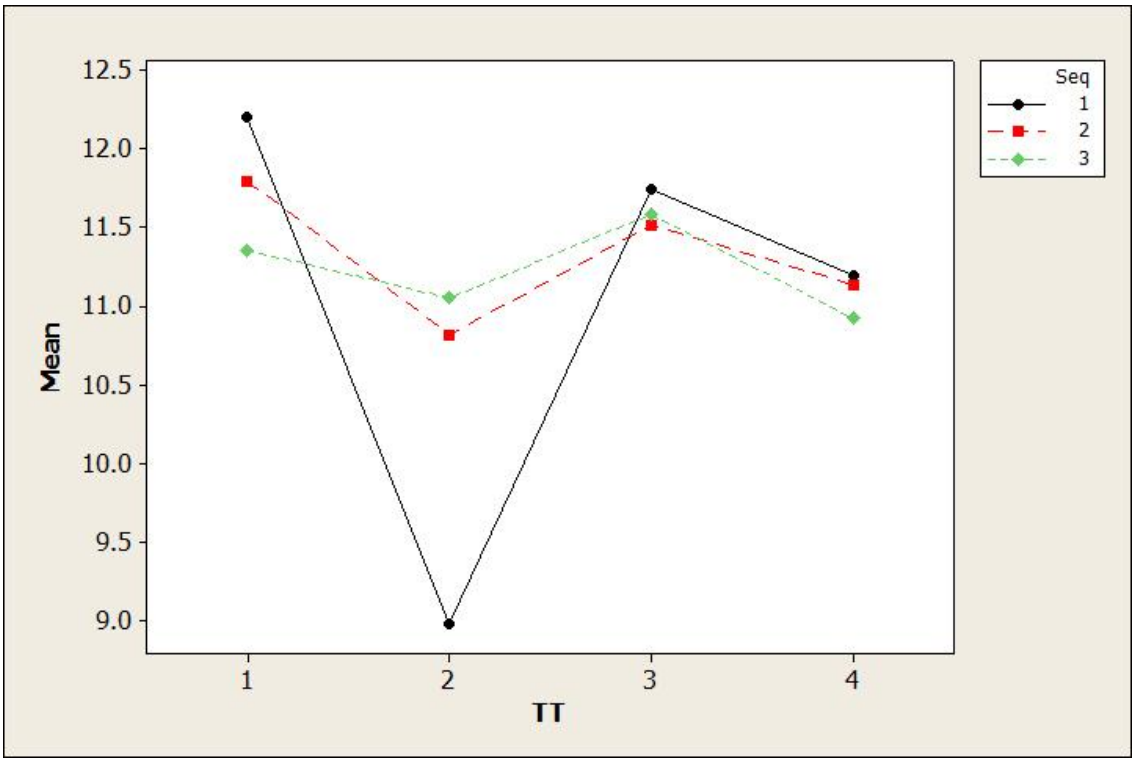

Figure 6-12 Interaction Plot for PRD Values

shift timetabling, TSVNS gives the best results.

Figure 6-13 illustrates more insight about the efficiency of different timetabling and sequencing algorithms when the problems without known optimal solutions are categorized into different groups. In this figure, problems with 20 jobs or less and 10 machines or less are considered as category 1 and the rest of the problems are classified under category 2. Average PRD from different combinations of sequencing and timetabling algorithms are demonstrated in this figure. As expected, while combination of TS with timetabling algorithms seems promising, especially for category 1, combination of TS with reverse timetabling is superior all the times. In this figure, ND denotes non-delay timetabling, R represents reverse, RS stands for right shift and RRS indicates reverse right shift.

6.6 Conclusion

This chapter considered the scheduling problem when a set of jobs are available for processing in a no-wait job shop context. This problem is proven to be strongly NP-hard. In other words, finding optimal solutions of large instances of this problem is not possible in a reasonable time. Therefore, the problem is decomposed into two other NP-hard problems: sequencing and timetabling. In order to deal with the timetabling problem, first the problem is modeled using the disjunctive graph. Then three famous timetabling algorithms were considered from the literature; a new timetabling algorithm was introduced by combining the previous three algorithms from the literature. Moreover, three different sequencing algorithms were developed: a tabu search, a hybrid of tabu search and variable neighborhood search, a hybrid of tabu search and particle swarm optimization.

Different combinations of the mentioned timetabling and sequencing algorithms were applied to a large set of test problems available in the literature. A Design of

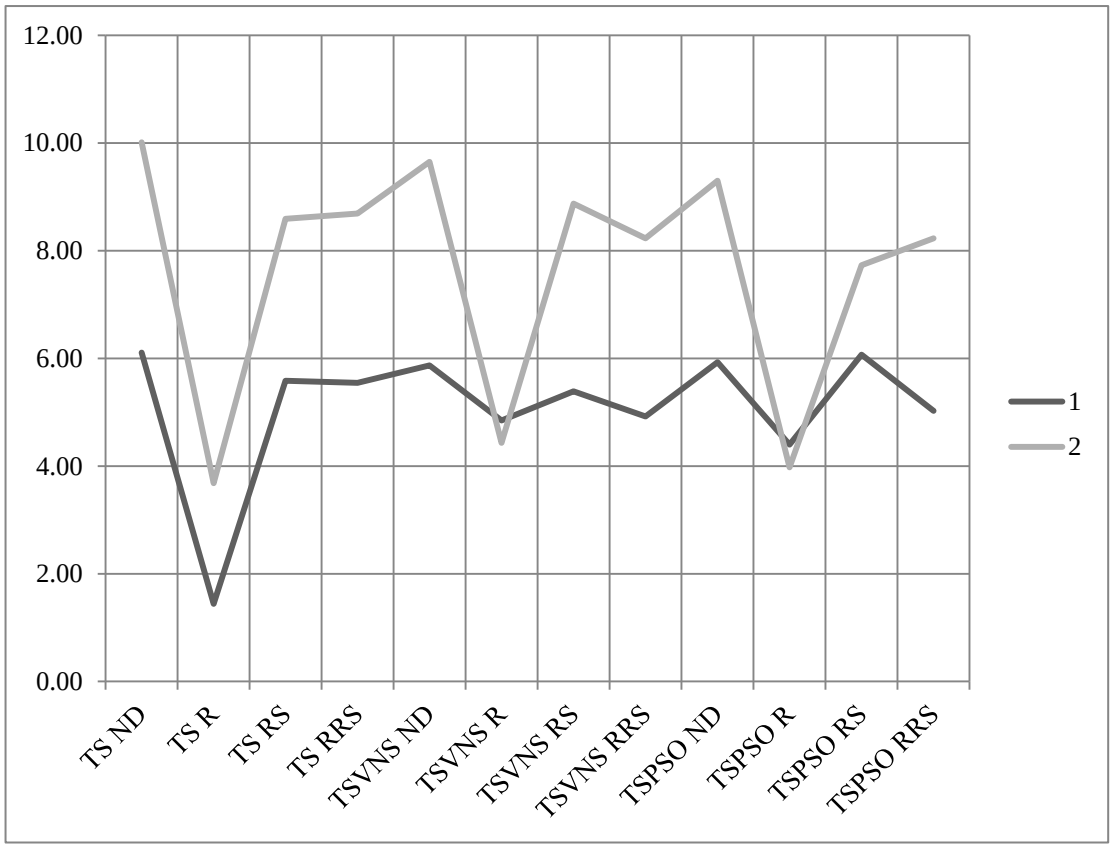

Figure 6-13 Average PRD values for problems without known optimal solution

Experiment (DOE) model was developed in order to determine the best combination of the sequencing and timetabling methods. Analysis of the DOE model provides a deep insight into the effective factors as well as their interactions on the obtained solutions' makespan. Moreover, DOE analysis proves that the most complicated algorithms do not necessarily obtain the best solutions.

Chapter 7

Conclusion and Future Research

7.1 Background

As mentioned in previous chapters, no-wait scheduling problems have numerous industrial and real-world applications. In this research, a number of no-wait scheduling problems were studied: 2-machine no-wait flow shop/job shop problem with setup time and single server constraints; general no-wait flow shop problem; no-wait flow shop problem with separable setup times; no-wait flow shop problem with sequence dependent setup times; and no-wait job shop problem. Section 7.2 summarizes the contributions of this thesis. Section 7.3 lists some recommendations for future researches in this direction.

7.2 Research Contributions

7.2.1 2-machine No-Wait Flow Shop Problem with Setup Times and Single Server Constraints

- Chapter 3 proves a number of theorems for special cases of 2-machine no-wait flow shop problem with and without setup times and single server constraints.
- A mathematical model is developed for the problems. A number of small instance problems are solved to optimality using the mathematical models.
- An efficient algorithm that calculates the makespan of a given permutation is introduced.
- A metaheuristic, namely a TSVNS is developed to solve the problem.

- The algorithm proves to be very effective in dealing with small and large instance test problems. The proposed algorithm is able to produce the exact solutions for all the small instance problems.

This work has been published in a journal paper [119].

7.2.2 General No-Wait Flow Shop Problems

- Chapter 4 first studied the general NWFS. NWFS is reduced to a permutation problem. An algorithm to calculate the makespan of a given permutation is developed.
- A novel PSO is proposed that transforms the feasible region of the NWFS (permutation of jobs) to a subset of integer numbers according to a one-to-one mapping.
- The proposed PSO is applied to a large number of test problems from the literature. Computational results section lists a number of new best-known solutions for the problems available in the literature.
- A GA and GAPSO are proposed to solve NWFS with separable setup times. Both algorithms prove to be very competitive when applied to randomly generated problems.
- Lastly, the problem of NWFS when separable sequence dependent setup times are in effect is introduced.
- A new PSO algorithm is developed. The PSO algorithm employs the concept of Matrix Coding (MC) to map the feasible region of the problem to specifically coded matrices. The developed PSO is able to utilize the MCs in a very efficient way. The new coding system reduces the computational time of the algorithm compared to the PSO proposed for NWFS with separable setup times.

The results of this research are published in two journal papers [126, 127], two conference papers [128, 129], and a third journal paper under review [130].

7.2.3 2-machine No-Wait Job Shop Problem with Setup Times and Single Server Constraints

- Chapter 5 generalizes the theorems of chapter 3 ($F2|no-wait,setup|C_{\max}$) for the case of 2-machine job shop problem.
- A mathematical model for the $J2,S1|no-wait,setup|C_{\max}$ is developed.
- A method to calculate the makespan of a given permutation is introduced.
- An efficient GA is developed to deal with the problem.
- The proposed GA is applied to a number of small and large case instances; the developed GA proves to be very effective both for small and large instances.

This work is published in a journal paper [131], and a conference paper [132].

7.2.4 General No-Wait Job Shop Problem

- NWJS is a generalization of $J2|no-wait|C_{\max}$, studied in chapter 5.
- NWJS is decomposed into two NP-hard sub-problems: sequencing and timetabling.
- The algorithms of chapters 3, 4, and 5 are generalized to deal with the sequencing sub-problem.
- Consequently, different timetabling methods is identified in the literature. A new timetabling algorithm is developed by combining the timetabling algorithms from the literature.

- Important questions arise at this point. Namely: which sequencing and which timetabling algorithm is the most successful? Is there an interaction between the sequencing and timetabling algorithms?
- A method based on design of experiments (DOE) is implemented to determine the importance of timetabling algorithm against sequencing algorithm when combined together to be applied to NWJS.
- Accordingly, 3 different sequencing algorithms are combined with 4 distinct timetabling methods.
- Design of experiments reveals that not only both the timetabling and sequencing algorithms are important, but also there is a significant interaction between the two algorithms. In other words, a good combination of the two algorithms generates solutions that are superior to poor combinations.

The results of the research performed in this chapter are presented as a conference paper [133] and under review as a journal paper [134].

7.3 Recommendations for Future Research

The opportunities for the future research are countless, especially when other applicable situations are taken into consideration. An abridged list of recommendations follows:

- Considering the following conditions:
  - Ready time
  - Due date
  - Weighted importance of the jobs
  - Uncertainty in the operating time
  - Sequence dependent setup times for NWJS

- Considering batch processing of the jobs:
    - Prioritizing batches of jobs once batches are formed
    - Partial batch processing
    - Batch size determination
- Generalizing no-wait constraints into time-lag constraints, and the applications of the problems in industry and service sector.
- Considering uncertainty:
    - Machine breakdown
    - Job cancellation
    - Variation of processing time
    - Other uncertainties.
- Considering other objective functions instead of makespan
- Multi-objective optimization
- Developing exact methods that are able to generate the best solution for small instances of the above problems in a reasonable time.
- Considering different new applications of the no-wait scheduling problems; for instance in service sector.
- Developing algorithms to generate semi-active or active schedules for NWJS

It is beneficial to elaborate on some of the complexities that arise when more applicable conditions are considered. For instance, Figure 7-1 demonstrates the case when sequence dependent setup times are considered for NWJS. This figure shows an exemplary partial timetable from permutation $(1,2,3)$, where jobs 1 and 2 are already scheduled and the algorithm is about to schedule job 3. If the algorithm decides to check

the possibility of placing job 3 in the illustrated time slot, not only the setup time of that operation of job 3 should be considered, but also the change in the setup time of the respective operation of job 1 should be paid attention to as this setup time might also need to be updated due to the change in its previous operation.

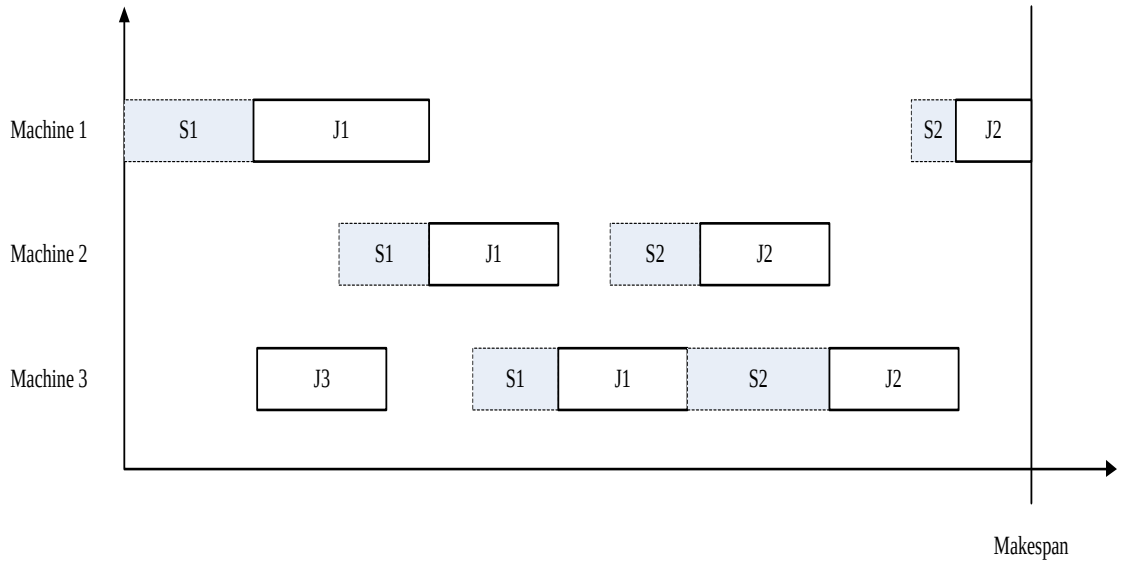

Figure 7-1 Sequence Dependent Setup and NWJS

References

1. Pinedo, M., *Scheduling, Theory, Algorithms, and Systems*. 2 ed. 2002, London, Sydney, Toronto, Mexico, New Delhi, Tokyo, Singapore, Rio de Janeiro: Prentice Hall Inc.

2. Garey, M.R. and D.S. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*. 2002, New York: Freeman.

3. Gupta, J.N.D., V.A. Strusevich, and C.M. Zwaneveld, *Two-stage no-wait scheduling models with setup and removal times separated.* Computers & Operations Research, 1997. **24**(1): p. 1025-1031.

4. Rajendran, C., *A no-wait flowshop scheduling heuristic to minimize makespan.* Journal of the Operational Research Society, 1994. **45**: p. 472-478.

5. Hall, N. and C. Sriskandarajah, *A survey of machine scheduling problems with blocking and no-wait in process.* Operations Research, 1996. **44**: p. 510-525.

6. Wismer, D., *Solution of the flowshop-scheduling with no intermediate queues.* Operations Research, 1972. **20**: p. 689-697.

7. Raaymakers, W. and J. Hoogeveen, *Scheduling multipurpose batch process industries with no-wait restrictions by simulated annealing.* European Journal of Operational Research, 2000. **126**: p. 131-151.

8. Grabowski, J. and J. Pempera, *Sequencing of jobs in some production system.* European Journal of Operational Research, 2000. **125**: p. 535-550.

9. Reddi, S. and C. Ramamoorthy, *On the flowshop sequencing problem with no-wait in process.* Operational Research Quarterly, 1972. **23**: p. 323-331.

10. Grabowski, J. and M. Syslo, *On some machine Sequencing problems (I).* Applications of Mathematicae, 1973. **13**: p. 340-345.

11. Bonney, M. and S. Gundry, *Solutions to the constrained flowshop sequencing problem.* Operational Research Quarterly, 1976. **24**: p. 869-883.

12. King, J. and A. Spachis, *Heuristics for flowshop scheduling.* International Journal of Production Research, 1980. **18**: p. 343-357.

13. Gangadharan, R. and C. Rajendran, *Heuristic algorithms for scheduling in no-wait flowshop.* International Journal of Production Economics, 1993. **32**: p. 285-290.

14. Glass, C.A., J. Gupta, and C. Potts, *Two-machine no-wait flow shop scheduling with missing operations.* Mathematics of Operations Research, 1999. **24**(4): p. 911-924.

15. Sidney, J., C. Potts, and C. Sriskandarayah, *Heuristic for scheduling two-machine no-wait flow shops with anticipatory setups.* Operations Research Letters, 2000. **26**(4): p. 165-173.

16. Sviridenko, M., *Makespan minimization in no-wait flow shops: A polynomial time approximation scheme.* SIAM Journal on Discrete Mathematics, 2003. **16**(2): p. 313-322.

17. Aldowaisan, T. and A. Allahverdi, *Total flowtime in no-wait flowshops with separated setup times.* Computers & Operations Research 1998. **25**: p. 757-765.

18. Aldowaisan, T., *A new heuristic and dominance relations for no-wait flowshops with setups.* Computers & Operations Research, 2001. **28**: p. 563-584.

19. Macchiaroli, R., S. Molè, and S. Riemma, *Modelling and optimization of industrial manufacturing processes subject to no-wait constraints.* International Journal of Production Research, 1999. **37**(11): p. 2585 - 2607.

20. Cheng, T.C.E., G. Wang, and C. Sriskandarajah, *One-operator two-machine flowshop scheduling with setup and dismounting times.* Computers and Operations Research, 1999. **26**: p. 715-730.

21. Bianco, L., P. Dell'Olmo, and S. Giordani, *Flow shop no-wait scheduling with sequence dependent setup times and release dates.* INFOR, 1999. **37**(1): p. 3-20.

22. Bertolissi, E., *Heuristic algorithm for scheduling in the no-wait flow-shop.* Journal of Materials Processing Technology, 2000. **107**(1 - 3): p. 459 - 465.

23. Bertolissi, E. *A simple no-wait flow-shop scheduling heuristic for the no-wait flowshop problem*. in *15th International Conference on Computer-Aided Production Engineering, CAPE'99*. 1999. Durham, UK.

24. Rajendran, C. and D. Chaudhuri, *Heuristic algorithms for continuous flow-shop problem.* Naval Research Logistics, 1990. **37**: p. 695-705.

25. Cheng, T., J. Gupta, and G. Wang, *A review of flowshop scheduling research with setup times.* Production and Operations Management, 2002. **9**: p. 262-282.

26. Shyu, S.J., B.M.T. Lin, and P.Y. Yin, *Application of ant colony optimization for no-wait flowshop scheduling problem to minimize the total completion time.* Computers & Industrial Engineering, 2004. **47**(2 - 3): p. 181 - 193.

27. Pranzo, M., *Batch scheduling in a two-machine flow shop with limited buffer and sequence independent setup times and removal times* European Journal of Operational Research, 2004. **153**(3): p. 581-592

28. Aldowaisan, T. and A. Allahverdi, *New heuristics for m-machine no-wait flowshop to minimize total completion time.* Omega, 2004: p. 345-352.

29. Chen, C., R. Neppalli, and N. Ajaber, *Genetic algorithms applied to the continuous flow shop problem.* Computers & Industrial Engineering, 1996. **30**: p. 919-929.

30. Grabowski, J. and J. Pempera, *Some local search algorithms for no-wait flow-shop problem with makespan criterion.* Computers & Operations Research, 2005. **32**: p. 2197–2212.

31. Guirchoun, S., P. Martineau, and J.C. Billaut, *Total completion time minimization in a computer system with a server and two parallel processors.* Computers & Operations Research 2005. **32**: p. 599-611.

32. Franc, P.M., A.G. Tin, and L.S. Buriol, *Genetic algorithms for the no-wait flowshop sequencing problem with time restrictions.* International Journal of Production Research, 2006. **44**(5): p. 939 - 957.

33. Liu, B., L. Wang, and Y.-H. Jin, *An effective hybrid particle swarm optimization for no-wait flow shop scheduling.* International Journal of Advanced Manufacturing Technology, 2007. **31**: p. 1001-1011.

34. Su, L.H. and Y.Y. Lee, *The two-machine flowshop no-wait scheduling problem with a single server to minimize the total completion time.* Computers & Operations Research, 2008. **35**: p. 2952-2963.

35. Li, X., Q. Wang, and C. Wu, *Heuristic for no-wait flow shops with makespan minimization.* International Journal of Production Research, 2008. **46**(9): p. 2519 - 2530.

36. Pan, Q.-K., M.F. Tasgetiren, and Y.-C. Liang, *A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem.* Computers & Operations Research, 2008. **35**(9): p. 2807 - 2839.

37. Pan, Q.-K., L. Wang, MF Tasgetiren, B.-H Zhao, *A hybrid discrete particle swarm optimization algorithm for the no-wait flow shop scheduling problem with makespan criterion* The International Journal of Advanced Manufacturing Technology, 2008. **38**(3 - 4): p. 337 - 347.

38. Pan, Q.-K., L. Wang, and B.-H. Zhao, *An improved iterated greedy algorithm for the no-wait flow shop scheduling problem with makespan criterion* The International Journal of Advanced Manufacturing Technology, 2008. **38**(7 - 8): p. 778 - 786.

39. Pan, Q.-K., L. Wang, and B. Qian, *A novel differential evolution algorithm for bi-criteria no-wait flowshop scheduling problems.* Computers & Operations Research, 2009. **36**(8): p. 2498 - 2511.

40. Tavakkoli-Moghaddam, R., A.R. Rahimi-Vahed, and A.H. Mirzaei, *Solving a multi-objective no-wait flow shop scheduling problem with an immune algorithm* The International Journal of Advanced Manufacturing Technology, 2008. **36**(9 - 10): p. 969 - 981.

41. Laha, D. and U.K. Chakraborty, *A constructive heuristic for minimizing makespan in no-wait flow shop scheduling.* International Journal of Advanced Manufacturing Technology, 2009. **41**: p. 97-109.

42. Huang, R.H., C.-L. Yang, and Y.-C. Huang, *No-wait two-stage multiprocessor flow shop scheduling with unit setup* The International Journal of Advanced Manufacturing Technology, 2009. **44**: p. 921-927.

43. Ruiz, R. and A. Allahverdi, *New heuristics for no-wait flow shops with a linear combination of makespan and maximum lateness.* International Journal of Production Research, 2009. **47**(20): p. 5717 - 5738.

44. Huang, Q., J. Xiao, and J. Zhang, *Orthogonal ant colony system for no-wait flow-shop scheduling* Computer Engineering and Design, 2010. **31**(6): p. 1274 - 1278.

45. Framinan, J.M., M.S. Nagano, and J.V. Moccellin, *An efficient heuristic for total flowtime minimisation in no-wait flowshops* The International Journal of Advanced Manufacturing Technology, 2010. **46**(9 - 12): p. 1049 - 1057.

46. Jarboui, B., M. Eddaly, and P. Siarry, *A hybrid genetic algorithm for solving no-wait flowshop scheduling problems* The International Journal of Advanced Manufacturing Technology, 2011. **54**(9 - 12): p. 1129 - 1143.

47. Gao, K.-Z., Q.-K. Pan, and J.-Q. Li, *Discrete harmony search algorithm for the no-wait flow shop scheduling problem with total flow time criterion* International Journal of Advanced Manufacturing Technology, 2011. **56**(5 - 8): p. 683 - 692.

48. Gao, K.-Z., S. Xie, H. Jiang, J-K. Li, *Discrete harmony search algorithm for the no wait flow shop scheduling problem with makespan criterion* Advanced Intelligent Computing, 2012. **6838**: p. 592 - 599.

49. Engin, O. and C. Günaydin, *An adaptive learning approach for no-wait flowshop scheduling problems to minimize makespan.* International Journal of Computational Intelligence Systems 2011. **4**(4): p. 521 - 529.

50. Qian, B., H-B. Zhou, R. Hu, F-H. Xiang, H*ybrid differential evolution optimization for no-wait flow-shop scheduling with sequence-dependent setup times and release dates* Advanced Intelligent Computing, 2012. **6838**: p. 600 - 611.

51. Ying, K.-C., Z-J. Lee, C-C. Lu, S-W. Lin, *Metaheuristics for scheduling a no-wait flowshop manufacturing cell with sequence-dependent family setups* The International Journal of Advanced Manufacturing Technology, 2012. **58**(5 - 8): p. 671 - 682.

52. Höhn, W., T. Jacobs, and N. Megow, *On Eulerian extensions and their application to no-wait flowshop scheduling* Journal of Scheduling, Article In Press.

53. Jolai, F., M. Rabiee, and H. Asefi, *A novel hybrid meta-heuristic algorithm for a no-wait flexible flow shop scheduling problem with sequence dependent setup times.* International Journal of Production Research, Article in Press.

54. Naderi, B., M. Aminnayeri, M. Piri, M.H.H. Yazdi, *Multi-objective no-wait flowshop scheduling problems: models and algorithms.* International Journal of Production Research, Article In Press.

55. Sapkal, S.U., D. Laha, and D.K. Behera, *Optimization techniques for no-wait manufacturing scheduling: a review*. Advanced Materials Research, 2012. **488 - 489**: p. 1114 - 1118.

56. Goyal, S.K. and C. Sriskandarajah, *No-wait shop scheduling: computational complexity and approximate algorithms.* Opsearch, 1988. **25**: p. 220-244.

57. Liaw, C.F., *An efficient simple metaheuristic for minimizing the makespan in two-machine no-wait job shops.* Computers and Operations Research, 2008. **35**: p. 3276-3283.

58. Sahni, S. and Y. Cho, *Complexity of scheduling shops with no wait in process.* Mathematics of Operations Research, 1979. **448**: p. 448-457.

59. Kamoun, H. and C. Sriskandarajah, *The complexity of scheduling jobs in repetitive manufacturing systems.* European Journal of Operational Research, 1993. **70**: p. 350 - 364.

60. Sriskandarajah, C. and P. Ladet, *Some no-wait shops scheduling problems: Complexity aspects.* European Journal of Operational Research, 1986. **24**: p. 424 - 438.

61. Kubiak, W., *A pseudo-polynomial algorithm for a two-machine no-wait jobshop scheduling problem.* European Journal of Operational Research, 1989. **43**: p. 267 - 270.

62. Goyal, S.K., *Job-Shop sequencing problem with no wait in process.* International Journal of Production Research, 1975. **13**(2): p. 197 - 206.

63. Woeginger, G.J., *Inapproximability results for no-wait job shop scheduling.* Operations Research Letters, 2004. **32**: p. 320 - 325.

64. Kravchenko, S.A., *A polynomial algorithm for a two-machine no-wait job-shop scheduling problem.* European Journal of Operational Research, 1998. **106**: p. 101 - 107.

65. Mascis, A. and D. Pacciarelli, *Job-shop scheduling with blocking and no-wait constraints.* European Journal of Operational Research, 2002. **143**: p. 498-517.

66. Mascis, A. and D. Pacciarelli, *Machine scheduling via alternative graphs*. 2000, Report DIA 46-2000, Dipartimento di Informatica e Automazione, Università Roma Tre: Rome.

67. Meloni, C., D. Pacciarelli, and M. Pranzo, *A rollout metaheuristic for job shop scheduling problems.* Annals of Operations Research, 2004. **131**: p. 215 - 235.

68. Ovacik, I. and R. Uzsoy, *Decomposition method for complex factory scheduling problems*. 1997: Dordrecht: Kluwer Academic Publishing.

69. Macchiaroli, R., S. Mole, S. Riemma, L. Trifiletti, *Design and implementation of a tabu search algorithm to solve the no-wait job-shop scheduling problem*, in *In Proceeding of the CESA1996, Lille*. 1996. p. 467 - 472.

70. Brizuela, C.A., Y. Zhao, and N. Sannomiya, *No-wait and blocking job-shops: Challenging problems for GA's*, in *In IEEE international conference on systems, man, and cybernetics*. 2001: Tucson, Arizona, USA. p. 2349 - 2354.

71. Chang, G.-J., J.-Q. Xu, and J.-H. Zhang, *On-line no-wait job shop scheduling in supply chain*, in *Chinese Control and Decision Conference*. 2008: Yantai, Shandong. p. 176 - 180.

72. Schuster, C. and J. Framinan, *Approximative procedures for no-wait job shop scheduling.* Operations Research Letters, 2003. **31**: p. 308-318.

73. Framinan, J.M. and C. Schuster, *An enhanced timetabling procedure for the no-wait job shop problem: a complete local search approach.* Computers & Operations Research, 2006. **331**: p. 1200 - 1213.

74. Schuster, C., *No-wait job shop scheduling: tabu search and complexity of subproblems.* Mathematical Methods of Operations Research, 2006. **63**: p. 473 - 491.

75. Pan, J.C.-H. and H.-C. Huang, *A hybrid genetic algorithm for no-wait job shop scheduling problems.* Expert Systems with Applications, 2009. **36**: p. 5800 - 5806.

76. Bozejko, W. and M. Makuchowski, *A fast hybrid tabu search algorithm for the no-wait job shop problem.* Computers & Industrial Engineering, 2009. **56**: p. 1502 - 1509.

77. Zhu, J., X. Li, and Q. Wang, *Complete local search with limited memory algorithm for no-wait job shops to minimize makespan.* European Journal of Operational Research, 2009. **198**: p. 378 - 386.

78. Grimes, D. and E. Hebrard, *Job shop scheduling with setup times and maximal time-lags: a simple constraint programming approach.* Lecture Notes in Computer Science, 2010. **6140**: p. 147 - 161.

79. Liu, S.Q. and E. Kozan, *Scheduling trains with priorities: a no-wait blocking parallel-machine job-shop scheduling model.* Transportation Science, 2011. **45**(2): p. 175 - 198.

80. Mokhtari, H., I.N.K. Abadi, and S.H. Zegordi, *Production capacity planning and scheduling in a no-wait environment with controllable processing times: An integrated modeling approach.* Expert Systems with Applications, 2011. **38**: p. 12630 - 12642.

81. Zhu, J., X. Li, and W. Shen, *A divide and conquer-based greedy search for two-machine no-wait job shop problems with makespan minimisation.* International Journal of Production Research, 2011.

82. Bozejko, W. and M. Makuchowski, *Solving the no-wait job-shop problem by using genetic algorithm with automatic adjustment.* International Journal of Advanced Manufacturing Technology, 2011. **57**: p. 735 - 752.

83. Santosa, B., M.A. Budiman, and S.E. Wiratno, *A cross entropy-genetic algorithm for m-machines no-wait job-shop scheduling problem.* Journal of Intelligent Learning Systems and Applications, 2011. **3**: p. 171 - 180.

84. Burgy, R. and H. Groflin, *Optimal job insertion in the no-wait job shop.* Journal of Combinatorial Optimization.

85. Lawler, E.L., J.K. Lenstra, A.H.G.R. Kan, D.B. Shmoys, *Sequencing and scheduling: algorithms and complexity*, in *Handbooks in operations research and management science*, S.C. Graves, A.H.G.R. Kan, and P. Zipkin, Editors. 1993, North-Holland.

86. Garey, M.R., D.S. Johnson, and R. Sethi, *The complexity of flowshop and jobshop scheduling.* Mathematics of Operations Research, 1976. **1**: p. 117-129.

87. Cadambi, B. and Y. Sathe, *Two-machine flowshop scheduling to minimize mean flow time.* Operational Research Society of India, 1993. **30**: p. 35-41.

88. DellaCroce, F., M. Ghirardi, and R. Tadei, *An improved branch-and-bound algorithm for the two machine total completion time flow shop problem.* European Journal of Operational Research, 2002. **139**: p. 293-301.

89. Akkan, C. and S. Karabati, *The two-machine flowshop total completion time problem: improved lower bounds and a branch-and-bound algorithm.* European Journal of Operational Research, 2004. **159**: p. 420-429.

90. Röck, H., *Some new results in flow shop scheduling.* Zeitschrift für Operations Research, 1984. **28**: p. 1-16.

91. Papadimitriou, C.H. and P.C. Kanellakis, *Flowshop scheduling with limited temporary storage.* Journal of the Associated Computer Machinery, 1980. **20**: p. 533-549.

92. Glass, C.A., Y.M. Shafransky, and V.A. Strusevich, *Scheduling for parallel dedicated machines with a single server.* Naval Research Logistics, 2000. **47**(4): p. 304-328.

93. Brucker, P., S. Knust, and G. Wang, *Complexity results for flow-shop problems with a single server.* European Journal of Operational Research, 2005. **165**: p. 398-407.

94. Allahverdi, A., J. Gupta, and T. Aldowaisan, *A review of scheduling research involving setup considerations.* Omega, 1999. **27**: p. 219-239.

95. Allahverdi, A., C.T. Ng, T.C.E. Cheng, M.Y. Kovalyov, *A survey of scheduling problems with setup times or costs.* European Journal of Operational Research 2008. **187**: p. 985-1032.

96. Mladenovic, N., *A variable neighborhood algorithm - a new metaheuristic for combinatorial optimization*, in *Optimization Days*. 1995: Montreal. p. 112.

97. Mladenovic, N. and P. Hansen, *Variable neighborhood search.* Computers & Operations Research, 1997. **24**: p. 1097-1100.

98. Glover, F., *Future paths for integer programming and links to artificial intelligence.* Computers & Operations Research, 1986. **13**: p. 533-549.

99. Rochat, Y. and E. Taillard, *Probabilistic diversification and intensification in local search for vehicle routing.* Journal of Heuristics, 1995. **1**: p. 147 - 167.

100. Behroozi, M., *A meta-heuristic approach for a special class of job shop scheduling problem*, in *Faculty of Industrial Engineering*. 2009, Sharif University of Technology: Tehran, Iran.

101. Campbell, H., R. Dudek, and M. Smith, *A heuristic algorithm for the n-job, m-machine sequencing problem.* Management Science, 1970. **16**: p. 630-637.

102. Dannenbring, D., *An evaluation of flowshop sequencing heuristics.* Management Science, 1977. **23**: p. 1174-1182.

103. Engels, C. and B. Manthey, *Average-case approximation ratio of the 2-opt algorithm for the TSP.* Operations Research Letters, 2009. **37**: p. 83-84.

104. Tasgetiren, M., M. Sevkli, YC. Liang, G. Gencyilmaz, *Particle swarm optimization algorithm for permutation flowshop sequencing problem.* Lecture Notes on Computer Science, 2004. **3172**: p. 382-389.

105. Zhu, W., J. Curry, and A. Marquez, *SIMD tabu search for the quadratic assignment problem with graphics hardware acceleration.* International Journal of Production Research, 2010. **48**(4): p. 1035 - 1047.

106. Lina, S.W. and K.C. Ying, *Applying a hybrid simulated annealing and tabu search approach to non-permutation flowshop scheduling problems.* International Journal of Production Research, 2009. **47**(5): p. 1411 - 1424.

107. Pitts, R.A. and J.A. Ventura, *Scheduling flexible manufacturing cells using Tabu Search.* International Journal of Production Research, 2009. **47**(24): p. 6907 - 6928.

108. Sahin, R. and O. Turkbey, *A new hybrid tabu-simulated annealing heuristic for the dynamic facility layout problem.* International Journal of Production Research, 2009. **4**(24): p. 6855 - 6873.

109. Finke, D.A., D.J. Medeiros, and M.T. Traband, *Multiple machine JIT scheduling: a tabu search approach.* International Journal of Production Research, 2007. **45**(21): p. 4899 - 4915.

110. Eberhart, R.C. and J. Kennedy. *A new optimizer using particle swarm theory*. in *Sixth international symposium on micro machine and human science*. 1995. Nagoya, Japan.

111. Kennedy, J. and R.C. Eberhart. *A discrete binary version of the particle swarm algorithm*. in *IEEE International Conference on Systems, Man, and Cybernetics*. 1997.

112. Knuth, D., *Seminumerical Algorithms*. Third ed. The Art of Computer Programming. Vol. 2. 1997, Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

113. Doliskani, J.N., E. Malekian, and A. Zakerolhosseini, *A cryptosystem based on the symmetric group S_n*. International Journal of Computer Science and Network Security, 2008. **8**(2): p. 226-234.

114. McCaffrey, J. *Using permutations in .net for improved systems security*. 2003; Available from: http://msdn.microsoft.com/en-us/library/aa302371.aspx.

115. Beasley, J.E. *OR-Library: distributing test problems by electronic mail*. July 2009; Available from: http://people.brunel.ac.uk/~mastjjb/jeb/info.html.

116. Carlier, J., *Ordonnancements a contraintes disjonctives.* RAIRO Recherche Operationnelle, 1978. **12**: p. 333-351.

117. Reeves, C., *A genetic algorithm for flowshop sequencing.* Computers and Operations Research, 1995. **22**: p. 5-13.

118. Kulturel-Konak, S. and A. Konak, *A new relaxed flexible bay structure representation and particle swarm optimization for the unequal area facility layout problem.* Engineering Optimization 2011. **43**(12): p. 1263 - 1287.

119. Samarghandi, H. and T.Y. ElMekkawy, *An efficient hybrid algorithm for the two-machine no-wait flow shop problem with separable setup times and single server*. European Journal of Industrial Engineering, 2011. **5**(2): p. 111-131.

120. Shadrokh, S. and F. Kianfar, *A genetic algorithm for resource investment project scheduling problem, tardiness permitted with penalty*. European Journal of Operational Research, 2007. **181**: p. 86-101.

121. Dijkstra, E.W., *A note on two problems in connexion with graphs.* Numerische Mathematik, 1959. **1**: p. 269 - 271.

122. Cormen, T.H., et al., *Dijkstra's algorithm*, in *Introduction to Algorithms*. 2001, MIT Press and McGraw-Hill. p. 595 - 601.

123. Lawrence, S., *Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (Supplement)*. 1984, Pittsburgh, Pennsylvania: Graduate School of Industrial Administration, Carnegie-Mellon University.

124. Applegate, D. and W. Cook, *A computational study of the job-shop scheduling instance.* ORSA Journal on Computing, 1991. **3**: p. 149 - 156.

125. Storer, R.H., S.D. Wu, and R. Vaccari, *New search spaces for sequencing instances with application to job shop scheduling.* Management Science, 1992. **38**: p. 1495 - 1509.

126. Samarghandi, H. and T.Y. ElMekkawy, *A genetic algorithm and particle swarm optimization for no-wait flow shop problem with separable setup times and makespan criterion* International Journal of Advanced Manufacturing Technology, Article in Press.

127. Samarghandi, H. and T.Y. ElMekkawy, *A meta-heuristic approach for solving the no-wait flow shop problem.* International Journal of Production Research, Article in Press.

128. Samarghandi, H. and T.Y. ElMekkawy, *A genetic algorithm for the no-wait flow shop problem with separable setup times*, in *Canadian Operational Research Society (CORS) Annual Meeting*. 2011: St. John's, NL, Canada.

129. Samarghandi, H. and T.Y. ElMekkawy, *Solving no-wait flow shop problem with sequence dependent setup times*, in *Proceedings of The Canadian Society for Mechanical Engineering International Congress (CSME)*. 2012: Winnipeg, MB, Canada.

130. Samarghandi, H. and T.Y. ElMekkawy, *Using particle swarm optimization to solve the no-wait flow shop problem with sequence dependent setup times.* International Journal of Computer Integrated Manufacturing, Under Review.

131. Samarghandi, H. and T.Y. ElMekkawy, *On the two-machine no-wait job shop problem with separable setup times and single server constraints.* International Journal of Advanced Manufacturing Technology, Article in Press.

132. Samarghandi, H. and T.Y. ElMekkawy, *On the two-machine no-wait job shop problem with separable setup times and single server constraints*, in *Proceedings of The Canadian Society for Mechanical Engineering International Congress (CSME)*. 2012: Winnipeg, MB, Canada.

133. Samarghandi, H. and T.Y. ElMekkawy, *A comparison of different timetabling and sequencing algorithms in the no-wait job shop problem, a design of experiments approach*, in *Canadian Operational Research Society (CORS) Annual Meeting*. 2012: Niagara Falls, ON, Canada.

134. Samarghandi, H., T.Y. ElMekkawy, and A.-M. Ibrahem, *A comparison of different timetabling and sequencing algorithms in the no-wait jobshop problem, a design of experiments approach.* International Journal of Production Research, Article in Press.

Appendix 1

Tuning Algorithm Parameters

This appendix describes the procedure through which parameters of the PSO developed for x are tuned. As seen in section 4.6.3, the developed PSO has 4 parameters that must be tuned to obtain the best performance from the algorithm. Sensitivity analysis has been performed to determine the effect of the different values of the parameters on the performance of the algorithm. Accordingly, three different problems from the set of the test problems were chosen: car7+SD, car19+SD and car31+SD. section 4.6.3.3 gives more information on how these test problems are generated. Then each problem was solved 5 times with different combinations of parameter values as follows:

$$1 \rightarrow \left\{ \begin{array}{l} P = 2n \\ \phi_i = 2; i = 1,2 \\ I = 200n \end{array} \right\} \tag{9-1}$$

$$2 \rightarrow \left\{ \begin{array}{l} P = n \\ \phi_i = 1; i = 1,2 \\ I = 100n \end{array} \right\} \tag{9-2}$$

$$3 \rightarrow \left\{ \begin{array}{l} P = \frac{n}{2} \\ \phi_i = 0.85; i = 1,2 \\ I = 80n \end{array} \right\} \tag{9-3}$$

$$4 \rightarrow \left\{ \begin{array}{l} P = \frac{n}{3} \\ \phi_i = 0,5; i = 1,2 \\ I = 50n \end{array} \right\} \tag{9-4}$$

Table 9-1 gives the results of the analysis of variance (ANOVA) on the obtained results. The considered factors are the combination of algorithm parameters as defined in (9-1) through (9-4), and the test problems: rec7+SD, rec19+SD and rec31+SD.

Table 9-1 Analysis of Variance for Makespan

Source	Degree of Freedom	Sequential Sums of Squares	Adjusted Sums of Squares	Adjusted Mean Square Value	F-Value	P-Value
Problem	2	114502690	114502690	57251345	63795.72	0
Combination	3	12167	12167	4056	4.52	0.007
Problem * Combination	6	36889	36889	6148	6.85	0
Error	48	43076	43076	897		
Total	59	114594822				
	$R^2 = 99.96\%$		$R^2_{adj} = 99.95\%$			

In order to confirm that the above ANOVA table is valid, residuals should follow a normal distribution. Figure 4 illustrates the normal probability plot of residuals. Residual values should be close to the normal probability line in order to deduce that the residuals show normal distribution characteristics.

Figure 9-1 confirms that the residuals are very close to the normal line.

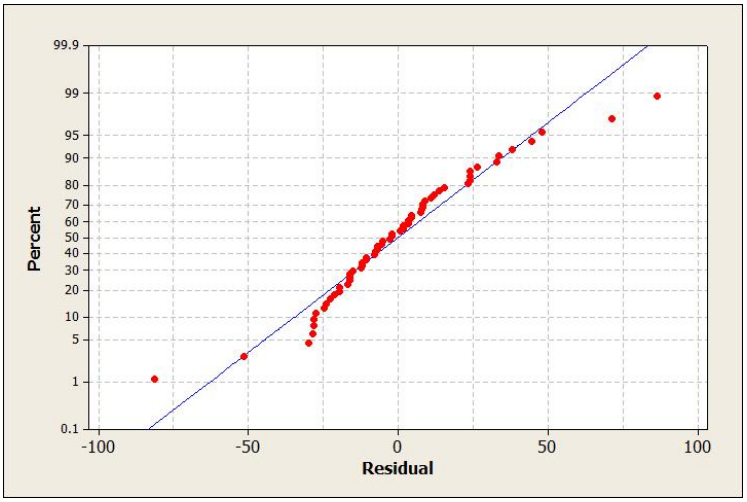

Figure 9-1 Normal Probability Plot of the Residuals

As Table 9-1 indicates, introduced combinations in (9-1) through (9-4) have an actual effect on the makespan of the studied test problems. In order to find the best combination amongst the four combinations, main effects plot proves useful.

Figure 9-2 illustrates the main effects plot.

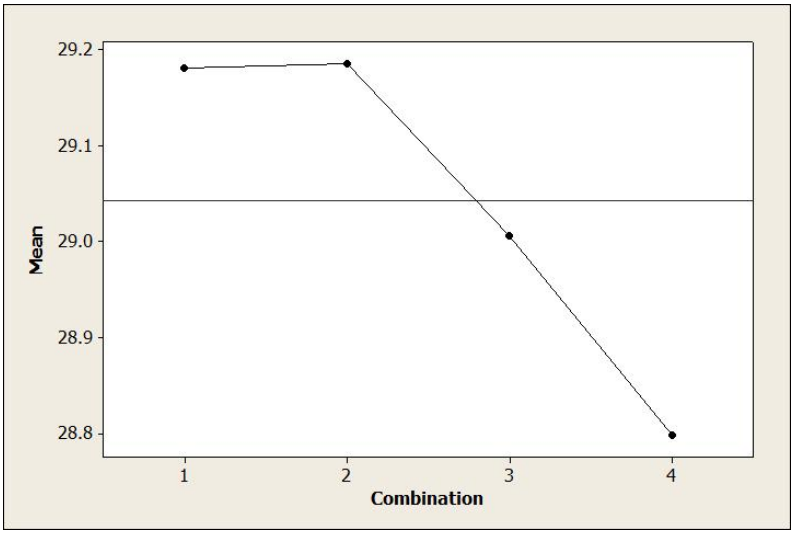

Figure 9-2 Main Effects Plot

Figure 9-2 demonstrates that combinations (9-1) and (9-2) are more desirable than combinations (9-3) and (9-4). Since the difference between combinations (9-1) and (9-2) is negligible, and combination (9-2) needs less iteration and particles to proceed, this combination is chosen to perform computational experiments in section 4.6.3. Similar method has been used throughout this thesis for tuning algorithms' parameters.