[flyleaf - a blank page, not numbered]

## AN ABSTRACT OF THE THESIS OF

Ayush Raj Aryal for the degree of Master of Science in Industrial Engineering presented on February 18, 2019.

Title: Bi-criteria Scheduling in an Assembly Flow Shop with Limited Buffer Storage and Shift Production

## Abstract approved: <br> Rasaratnam Logendran

In this research, the comparative performance of permutation and non-permutation schedules is investigated in an assembly flow shop (AFS) with shift production, where a limited buffer storage is available between two machines. Most of the traditional scheduling problems consider continuous production, i.e., production occurs for 24 hours ( 3 * 8-hour shifts) each day, seven days a week. However, some companies operate only one or two shifts each day, which creates a limited availability constraint on the machines. This causes a discontinuity in production between end and start of two successive production days. To mimic real-life industry practice, dynamic job release and dynamic machine availability times have been considered. Each job considered in a problem can have different weight assigned based on customers' preferences. The setup times between jobs are assumed to be machine- and sequence-dependent. However, at the start of each production day, setup times are not sequence-dependent but depend on machine startup times such as preheating time, pressure build up, etc. The objective of the problem is to minimize the linear combination of total setup time and weighted tardiness. The minimization of total setup time represents producer's interest whereas the minimization of weighted tardiness represents customers' interest. Since these two objectives are not evaluated on a commensurate basis, a normalization factor is used.

The problem is formulated as a mixed-integer linear programming (MILP) model, MILP-1 for permutation schedules and MILP-2 for non-permutation schedules. The MILP models for small-size problem instances are solved to optimality using CPLEX. However,
the problem is shown to be NP-hard. As a result, it is not possible to find an optimal solution within a reasonable time, as the problem size increases. Hence, a meta-heuristic search algorithm based on short-term Tabu Search (TS) and Tabu Search/Path-Relinking (TS/PR) are developed. TS represents a local search algorithm, whereas TS/PR represents a hybridization of local search enhanced with population-based search algorithm. Two algorithms each, are developed for both, permutation (PN) and non-permutation (NPN) sequences. One of the algorithms is based on short term TS and the other is based on TS/PR. The developed heuristics are tested on sixteen small-size problems and their solution quality are compared with the optimal solution obtained from CPLEX. The evaluations show that the developed heuristics obtain good quality solutions within much less computational time. For PN sequence, the best algorithm obtained an average deviation of $0.49 \%$ compared with the optimal solution and for NPN sequence, the deviation is $0.13 \%$. In addition, a slight improvement of $2.68 \%$ was obtained by adopting an NPN sequence over PN sequence for these problem instances.

A statistical designed experiment is conducted to evaluate the difference in performance of the developed heuristics, and permutation and non-permutation schedules. The results show that the TS/PR algorithms outperform short-term TS, in the case of both PN and NPN sequences. The comparison between the solutions from the best PN algorithm and the best NPN algorithm shows that an average improvement of $1.64 \%$ is obtained by implementing an NPN sequence over PN sequence. The statistical analysis shows that the improvement offered by NPN sequence is statistically significant for problems with large number of product types and small number of jobs in each product. In addition, it is also shown that the NPN sequence performs better for non-continuous production as compared to continuous production. The efficiency of the algorithms was analyzed using the computational time required by the algorithms. The results show that PN algorithms require a significantly less computational time as compared to NPN algorithms. Hence, it is recommended that NPN sequences be considered only for the problems with large number of product types and small number of jobs in each product. For other problems, only PN sequence should be considered. TS/PR algorithm is recommended for both, PN and NPN sequences.
©Copyright by Ayush Raj Aryal
February 18, 2019
All Rights Reserved

Bi-criteria Scheduling in an Assembly Flow Shop with Limited Buffer Storage and Shift Production.
by
Ayush Raj Aryal

## A THESIS

submitted to

Oregon State University

in partial fulfillment of the requirements for the degree of

Master of Science

Presented February 18, 2019
Commencement June 2019

## APPROVED:

Major Professor, representing Industrial Engineering

Head of the School of Mechanical, Industrial, and Manufacturing Engineering

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

## ACKNOWLEDGEMENTS

It would have been impossible to write my Master's thesis without my major professor, Dr. Rasaratnam Logendran, who has been a tremendous mentor for me over the last two years. I would like to thank you for patiently reading, correcting and suggesting improvements on various papers and this thesis. Thank you for allowing me to work on an industry-funded project. This project not only helped me in the pursuit of my thesis, but also prepared me for the challenges that I would face in the future. I will always be grateful for the time and effort you put into helping me complete this Master's degree. Your hard work and commitment have always inspired me to strive for excellence.

I would like to extend my gratitude to my committee members: Dr Sarah Emerson, my minor professor, Dr. Hector Vergara, my committee member, and Dr. Brett Tyler, my graduate council representative for their guidance and useful feedbacks. Special thanks to Dr. Emerson for serving as my minor professor.

I would like thank ATI, Inc. for partially supporting my studies through a funded project. They have been extremely cooperative and supportive throughout the duration of the project. I would also like to thank my previous employer at Surya Nepal Pvt. Ltd for helping me grow as a professional engineer. I would like to extend my appreciation to Omid Shahvari for his everlasting friendship and help on this thesis. I wish to thank IME staff members Jean Robinson and Stephanie Grigar for their help. Special thanks to Lori Burgeson for installing and maintaining any hardware/software needed for my research.

My time at Oregon State University has been delightful due to the love and support of my friends. I would like to thank Kshitiz Gyawali, Saroj Karki and Nisha Puri for entertaining me and making me feel at home. I would also like to thank my dear friend Atul Acharya for his everlasting friendship.

Finally, I would like to thank my family, without whom I would not be here.

## TABLE OF CONTENTS

Page

1. INTRODUCTION ..... 1
2. LITERATURE REVIEW ..... 5
2.1. Review of Literature on an Assembly Flow Shop ..... 6
2.2. Review of Literature on Bi-criteria Scheduling ..... 8
2.3. Review of Literature on Non-continuous Production. ..... 10
3. PROBLEM STATEMENT ..... 12
4. MATHEMATICAL MODELS ..... 14
4.1. Normalization of the Objective Function ..... 14
4.2. MILP1 ..... 15
4.3. MILP2 ..... 20
4.4. Choice of the Objective Function ..... 22
4.5. Complexity of the Problem ..... 24
5. HEURISTIC ALGORITHM ..... 26
5.1. Tabu Search ..... 27
5.1.1. Initial Solution Finding Mechanism ..... 30
5.1.2. Neighborhood Function ..... 33
5.1.3. Evaluation of the Objective Function ..... 35
5.1.4. Tabu list ..... 39
5.1.5. Aspiration criterion ..... 41
5.1.6. Steps of the Proposed TS Algorithm ..... 42
5.1.7. Application of the TS Algorithm to an Example Problem ..... 45
5.2. Tabu Search/Path Relinking ..... 57
5.2.1. Initial Population. ..... 58
5.2.2. Path Construction ..... 59
5.2.3. Path Solution Selection ..... 63
5.2.4. Reference Solution Determination. ..... 64
5.3. Calibration of the metaheuristic algorithms ..... 65
6. DATA GENERATION ..... 68

## TABLE OF CONTENTS (Continued)

Page
7. THE QUALITY OF SOLUTIONS OBTAINED FROM THE PROPOSED HEURISTIC ..... 73
8. RESULTS ..... 80
8.1. Experimental Design. ..... 80
9. CONCLUSIONS AND FUTURE RESEARCH ..... 93
BIBLIOGRAPHY ..... 97
APPENDIX ..... 102
Appendix A. Result of statistical analysis for parameter tuning ..... 103

## LIST OF FIGURES

Figure Page
Figure 1. General layout of an assembly flow shop ..... 2
Figure 2. Layout of an uneven assembly flow shop ..... 12
Figure 3. Gantt chart for a PN and NPN Schedule in a two-machine flow shop ..... 24
Figure 4. IS Flowchart ..... 32
Figure 5. Swap move ..... 33
Figure 6. Insert move ..... 33
Figure 7. Neighborhood structure for permutation ..... 34
Figure 8. Neighborhood structure for non-permutation ..... 34
Figure 9. TS flowchart ..... 44
Figure 10. Evaluation of job completion times ..... 51
Figure 11. LCS construction ..... 60
Figure 12. The LCS between two solutions in non-permutation sequence ..... 63
Figure 13. Global and Local Optima in InitialPathSet ..... 64
Figure 14. Flowchart for TS/PR ..... 66
Figure 15. Relationship between $\delta$ and CV ..... 72
Figure 16. Normality of objective function value ..... 82
Figure 17. Deviation of ALG1 from ALG3 ..... 87
Figure 18. Deviation of ALG2 from ALG4 ..... 87
Figure 19. Deviation of ALG3 from ALG4 ..... 87
Figure 20. Deviation of ALG3 from ALG4 ..... 88
Figure 21. Normal Probability Plot for CT ..... 89
Figure 22. Normal Probability Plot for inversed CT ..... 89

## LIST OF TABLES

Table Page
Table 1. Runtimes of the product ..... 23
Table 2. Machine availability times ..... 23
Table 3. Setup times ..... 23
Table 4. Due date and release time of a job ..... 23
Table 5. Nomenclature of algorithms used in this research ..... 26
Table 6. Moves for PTB and PTB2 ..... 35
Table 7. Extreme values of the criteria ..... 35
Table 8. OFV of solutions on the CL for algorithms and without MNSS restriction. ..... 41
Table 9. Example problem ..... 45
Table 10a. Setup time for $\mathrm{M}_{1}$ ..... 45
Table 11. Setup time for sequence generated using SST ..... 46
Table 12. Setup time for sequence generated using LST ..... 47
Table 13. Due date to weight ratio ..... 47
Table 14. Rank of jobs in PS and CS ..... 49
Table 15. Job scheduled at each iteration of IS generation mechanism on $\mathrm{M}_{1}$ ..... 49
Table 16. Job completion times on machine ..... 52
Table 17. NS generation in the first iteration. ..... 54
Table 18. Entries into the CL ..... 56
Table 19. Pseudocode for IP generation of permutation TS/PR ..... 58
Table 20. Possible candidate moves starting from $S^{I}$ ..... 62
Table 21. Pseudo-code for TS/PR ..... 65
Table 22. Due date classification ..... 71

## LIST OF TABLES (Continued)

TableTable 23. CPLEX runs of MILP1 and MILP2.................................................................. 75
Table 24. Solutions from metaheuristic algorithms ..... 76
Table 25. Average deviation for PN algorithms from CPLEX optimal solution. ..... 77
Table 26. Average deviation for PN algorithms from CPLEX bounds ..... 77
Table 27. Average deviation for NPN algorithms from CPLEX optimal solution ..... 78
Table 28. Average deviation for NPN algorithms from CPLEX bounds ..... 79
Table 29. Factors and their levels in the experiment ..... 81
Table 30. ANOVA of the objective function value in split-plot design ..... 82
Table 31. Result of ANOVA and Tukey test on algorithm's performance ..... 86
Table 32. ANOVA of the computational time in split-plot design ..... 89
Table 33. CT of algorithms for different problem structure ..... 91
Table 34. CT of algorithms for continuous and non-continuous production. ..... 92

## Bi-criteria Scheduling in an Assembly Flow Shop with Limited Buffer Storage and Shift Production

## 1. INTRODUCTION

Scheduling problems were first considered in the 1950s with the introduction to simple problems such as minimizing sum of flowtimes of jobs on a single machine, or minimizing makespan of jobs on two machines. The problem got more complex over time as more realistic shop constraints were incorporated into the problem. With increasing competition in the market, manufacturing firms often use scheduling techniques to improve their operational efficiency. One of the most important characteristics of a scheduling problem corresponds to the shop structure or machine configuration in which a job is processed. Typically, the machine configuration of the shop can be classified into a flow shop or a job shop. In a flow shop setting, jobs are processed on a flow line and have at most one operation on each machine, whereas in a job shop, jobs do not have to adhere to a flow line and can have multiple operations on the same machine.

Flow shops can further be classified into a typical flow shop, flexible flow shop and assembly flow shop. A typical flow shop has a single machine at each stage. If there is more than one machine of the same capability in at least one stage, then it is termed as a flexible flow shop. In an assembly flow shop, each job is comprised of multiple components, which are processed separately by independent parallel machines with different capabilities in the first stage. These components are then assembled in the second stage by one assembly machine. A general layout of an assembly flow shop is shown in Figure 1. In a typical assembly flow shop, each component requires only one operation in the first stage, i.e., before it is transferred to the assembly stage. However, that might not always be the case. Some components might require a higher number of operations before it is ready for assembly, i.e., the number of individual operations for each component might not be equal. This type of problem represents an uneven assembly flow shop structure.

Most of the past research on scheduling have considered a continuous 24 hours and seven days a week production. However, many companies do not adhere to continuous production. In some companies, production does not occur on weekends. Other companies


Figure 1. General layout of an assembly flow shop
might operate only one or two shifts a day, corresponding to 8 hours and 16 hours of production each day, respectively. Similarly, some machines might not be available during certain time period due to breakdown or preventive maintenance. This causes a discontinuity in production between two successive days and hence adds machine availability constraints to the problem, i.e., limited machine availability constraints. In some environments, there are limited buffers or no buffers between stages (Hall and Sriskandarajah, 1996, Qian et al., 2009, Liu et al., 2008, Maleki-Darounkolaei et al., 2012). This introduces blocking constraints on job processing. After a job is processed on a machine, it is transferred to the intermediate storage where it waits to be processed by a downstream machine. If the downstream machine is busy and all the intermediate storages are full, then the processing of jobs is blocked on the upstream machine.

A setup time is incurred each time a job is changed on a machine. A setup operation can include tasks such as preparing the machine, cleaning, inspection, machine setting etc. Setup times can be classified into two groups: sequence-independent and sequencedependent. If the duration of a setup time is dependent on current and preceding jobs, the setup is sequence-dependent; otherwise it is considered sequence-independent. Several studies have highlighted the importance of using sequence-dependent setup times (Allahverdi et al., 1999, Maleki-Darounkolaei et al., 2012). In a plant, all jobs might not be released at the start of the planning horizon, i.e., jobs might have different release times. This characteristic is called dynamic job release time. Similarly, machines might also have different availability times. This is called dynamic machine availability.

In most of the scheduling problems, the objective is to minimize a time-based objective, such as the makespan (the completion time of the last job on the last machine), sum of completion times, maximum lateness, total tardiness (positive lateness), and number of tardy jobs. While the minimization of tardiness caters to customers' interests, the minimization of makespan and minimization of sum of completion times cater to supplier's interests-These objectives largely focus on minimizing completion times of jobs on the last machine, which might not yield the best performance values for preceding machines, especially in non-continuous production. From a customers' point of view, using these objectives make sense because customers are only concerned with the due dates and whether or not the jobs are tardy. However, for the producer, the performance of all machines is of interest. Hence, minimization of total setup time is proposed to be used as an objective to accurately represent the producer's interest in non-continuous shift production.

A flow shop scheduling problem can be solved by considering two types of schedules, permutation (PN) and non-permutation (NPN) schedules. In a PN schedule, the processing sequence of jobs is the same for all machines whereas in an NPN schedules, processing sequence of jobs might be different across different machines. Most of the research on flow shop scheduling considers only PN schedules. However, when there is buffer storage in between, NPN schedules might outperform PN schedules (Strusevich and Zwaneveld, 1994, Liao and Huang, 2010).

This research is directly motivated by a real problem from a leading manufacturing company. The shop structure considered in this paper is that of an uneven assembly flow shop with limited intermediate storage and non-continuous shift production. Nonpermutation schedules are allowed. Setup times are sequence- and machine-dependent. Job release times and machine availability times are considered to be dynamic. The main purpose of this research is to find an optimal or near optimal schedule that will minimize the linear combination of total setup time and weighted tardiness. Such an objective is very relevant to current industry practice as there is a need to balance customers' and producer's objectives. Another purpose of this research is to evaluate the performance of permutation
vs. non-permutation schedules in a non-continuous production environment, i.e., limited machine availability. Few researchers in the past have considered limited machine availability in their study. To the best of our knowledge, a bi-criteria scheduling problem in an uneven assembly flow shop with blocking and limited machine availability has not been addressed so far.

## 2. LITERATURE REVIEW

Production scheduling deals with allocating limited number of resources to jobs over time. Researchers have taken a keen interest in this field since the 1960s. Johnson (1954) was the first to develop a systematic approach to obtain an optimal solution for the two-machine makespan minimization problem and also a special case for the threemachine problem. Johnson's algorithm was further extended by Campbell, Dudek and Smith (CDS) (1970) for the $m$-machine problem. CDS algorithm transforms $m$ machines into (m-1) two-virtual machine problems, for which various schedules are developed and the best sequence among them is selected to represent the best sequence for the original m machine problem. Nawaz, Enscore and Ham (NEH) (1983) developed a heuristic which gives priority to jobs with largest processing time. In contrast to CDS, NEH doesn't transform the original problem into a two-machine problem. Instead, it generates partial schedules and adds a job at each iteration, to finally obtain a complete best solution.

There have been several developments in the field of scheduling over the last couple of decades. Allahverdi et al. (2008) presented a comprehensive review on the advancements made in scheduling from mid-1998 to mid-2006. A more recent paper by Allahverdi (2015) provides an extensive review of papers published from mid-2006 to the end of 2014, including static, dynamic, deterministic, and stochastic environments. It includes classification of problems based on shop environments and setup considerations. Shop environments can be categorized as a single-machine, parallel machines, flow shop (regular flow shop, flow shop with blocking, no-wait flow shop, flexible flow shop, and assembly flow shop), job shop, or open shop problem. Setup times can be sequenceindependent or sequence-dependent. The scope of this research is to find an optimal schedule for an assembly flow shop with shift production with the objective of simultaneously minimizing two objectives. Hence, the literature review focuses on scheduling problems in an assembly flow shop with bi-criteria objective function and noncontinuous production due to limited machine availability.

### 2.1. Review of Literature on an Assembly Flow Shop

In a two-stage assembly flow shop (AFS) problem, there are $m$ parallel machines in the first stage while there is only one assembly machine in the second stage. There are $n$ jobs to be scheduled and each job is made up of $m$ individual components. Each component is processed separately and independently by parallel machines at the first stage and the final assembly is performed in the second stage. Thus, each job has a total of $m+$ 1 operations. This problem has many applications in industry such as fire engine assembly plant (Lee et al., 1993), personal computer manufacturing (Potts et al., 1995), distributed database systems (Al-Anzi and Allahverdi, 2006), etc. In particular, many real life scheduling problems can be modelled as a two-stage assembly flow shop.

Lee et al. (1993) were the first to introduce a two-stage AFS problem with $m=2$ (two machines in the first stage and one assembly machine in the second stage) for makespan minimization. This paper showed that the problem is NP-hard in a strong sense with this objective function. Lee et al. (1993) also discussed a few polynomially solvable cases and presented a mathematical model for the problem. Potts et al., (1995) considered the problem with an arbitrary $m$. They showed that the permutation schedules are dominant for makespan minimization. Hariri and Potts (1997) addressed the same problem and derived a lower bound and established several dominance theorems. They also presented a branch and bound algorithm incorporating the lower bound and dominance theorems. Koulamas and Kyparisis (2001) generalized the problem into a three-stage AFS problem with the objective of minimizing makespan. In this problem, a transfer stage is added in between the first stage and the assembly stage. They proposed several heuristics and analyzed the worst case bound for those heuristics.

Tozkapan et al. (2003) investigated a two-stage AFS ( $m$ machines in the first stage) with the objective of minimizing weighted flow time. They presented a branch and bound algorithm utilizing the derived lower bound and dominance relations. This paper showed that permutation schedules are dominant for minimizing weighted flowtime. Al-Anzi and Allahverdi (2006) considered the same problem and proposed three heuristics based on simulated annealing, tabu search and hybrid tabu search. They showed that the hybrid tabu
search is efficient as compared to other heuristics. Allahverdi and Al-Anzi (2009) proposed three heuristics addressing the same problem but with setup times considered separate from the processing times. Framinan and Gonzelez (2017) investigated a two stage assembly flow shop with the objective of minimizing total completion time and proposed a variable local search algorithm, which outperforms existing metaheuristics.

The above-mentioned literatures consider only one machine at the final assembly stage. Sung and Kim (2008) considered a two-stage AFS problem with $m=2$ machines in the first stage and two identical parallel assembly machines in the second stage. Al-Anzi and Allahverdi (2012) addressed the generalized version of this problem with $m$ machines in the first stage and two assembly machines. They proposed three heuristics which outperform the heuristics by Sung and Kim (2008).

Some research has also considered bi-criteria objective function for the AFS problem. Torabzadeh and Zandieh (2010) proposed a cloud-based simulated annealing approach for an AFS problem with $m$ machines in the first stage and one machine in the assembly stage, with the objective of minimizing a weighted sum of makespan and mean completion time. Maleki-Darounkolaei et al. (2012) addressed a three-stage AFS problem with blocking and sequence-dependent setup time and proposed a meta-heuristic based on simulated annealing to minimize the weighted sum of mean completion time and makespan.

While several researchers have addressed a variety of assembly flow shop problems, some gaps can still be identified in the literature. All of the above research assumes that each component requires only one operation in the first stage. However, that might not be true as some components could require more than one operation before it is ready for assembly, i.e. a regular flow shop environment (with two or more machines) in the first stage. Furthermore, none of the above research consider non-continuous production. To the best of our knowledge, AFS problem has been studied so far considering continuous production and with the assumption of single operation for each component in the first stage.

### 2.2. Review of Literature on Bi-criteria Scheduling

Selecting an appropriate objective is a challenge in solving scheduling problems. The objective in the optimization function can be classified into two groups: SupplierOriented and Customer-Oriented. Supplier-oriented objectives include functions such as minimizing makespan, sum of completion time, idle time, and work-in-progress inventory, whereas customer-oriented objectives include minimizing tardiness, minimizing number of tardy jobs, maximum lateness, etc. Most of the earlier research focused on only one of the groups. However, in today's environment, most companies try to reduce their cost while maintaining customers' service level, i.e. minimizing tardiness. Thus, a lot of recent research has been considering multi-objective scheduling problems. Allahverdi and Aldowaisan (2004) addressed the $m$-machine no-wait flow shop scheduling problem with a bi-criteria objective function of minimizing the weighted sum of makespan and maximum lateness. Eren and Güner (2006) considered a bi-criteria scheduling problem with sequence-dependent setup times on a single machine. An integer programing model is presented to minimize the weighted sum of total completion time and tardiness. A heuristic algorithm based on tabu search is also presented to solve large-size problems. Mehravaran and Logendran (2012) considered an unrelated-parallel machine problem with dual resource. The objective function is to minimize a linear combination of weighted flowtime and weighted tardiness. The weighted objective function used in this study simultaneously minimizes both objectives. In this study, the problem is solved in two parts, the first part considering only machine constraint, and the second part considering only labor constraint for the schedule developed in the first part. Another approach to tackle a multi-objective scheduling problem is to obtain pareto-optimal solutions which helps to obtain many nondominated solutions. Moslehi and Mahnam (2011) proposed a pareto-approach to multiobjective flexible flow shop problem using particle swarm optimization and local search. Most research in the past have used weighted objective function method because it provided a flexibility in assigning different weights to each criteria in the objective function based on producer's need at the time. Bozorgirad and Logendran (2013) used a weighted objective function method to address a sequence-dependent hybrid flow shop problem. The objective was to minimize the linear combination of weighted flowtime and weighted
tardiness. Shahvari and Logendran (2016) also used the same objective in a hybrid flow shop batching and scheduling problem. They proposed heuristics based on tabu search/path relinking and applied stage-based procedures to obtain better solutions. Their result showed the benefit of integrating batching decisions into group scheduling approach. Other research from Maleki-Darounkolaei et al. (2012) and Allahverdi and Aldowaisan (2004), which were reviewed in the literature on AFS scheduling, also used weighted objective function method.

The weighted objective method tries to simultaneously minimize the criteria in the objective function. However, issues might arise due to skewness (when the value of one criterion is much larger than the other), and dimensional conflict (when the two criteria do not have the same unit of measurement). When the value of one criterion is much larger than the other, the objective function favors the criterion with larger value as the larger criterion will tend to make a higher contribution to the objective function value. In this case, the objective function might not be meaningful or effective. This problem can be tackled using a normalization approach where each criterion is normalized into a dimensionless quantity between 0 and 1 . Several research considering weighted multicriteria objective function have used normalization to balance the criteria in the objective function. Gagné et al., (2005) presented a hybrid tabu search/variable neighborhood search algorithm for the solution of a bi-objective scheduling problem. The two objectives, setup times and tardiness, are normalized using nadir points (maximum value of the objectives) and ideal point (minimum value of the objectives). Oyetunji and Oluleye (2009) proposed a normalization procedure for a bi-criteria objective function of total completion time and number of tardy jobs. The methodology for determining the nadir and ideal points for these objectives is also demonstrated. Chyu and Chang (2010) proposed a competitive evolution strategy memetic algorithm to solve unrelated-parallel machine scheduling problem with two minimization objectives. These objectives are also normalized using ideal and nadir points.

### 2.3. Review of Literature on Non-continuous Production

Most of the scheduling problems in the past assumes that all machines are continuously available for processing throughout the planning horizon. This assumption might not be justified in all cases because some plants operate only on a single or a double shift. In addition, a pre-planned maintenance schedule might result in machine unavailability during certain times. The period of machine unavailability is called holes. Two cases of limited machine availability are defined by Lee (1997), resumable and nonresumable. In a resumable case, if an operation cannot be before the unavailability period of a machine, then it can continue after the machine becomes available without any cost. In non-resumable case, the disrupted operation has to be totally restarted. Lee (1997) studied a two-machine flow shop problem in which one machine is always available and the other machine has one period of unavailability in the planning horizon, i.e., machine is unavailable from $s$ to $t$, where $0 \leq s \leq t$. The unavailability period is known in advance, i.e., deterministic and the operation is resumable. It was shown that this problem is NP-hard even for makespan minimization. A pseudo-polynomial dynamic programming algorithm is also presented to solve the problem. Błażewicz et al. (2001) proposed a constructive and local search heuristic for a two-machine flow shop problem with resumable operations and up to ten number of holes on either machine. The objective is to find a feasible schedule with minimum makespan. Kubiak et al. (2002) proposed a branch and bound algorithm for the same problem addressed by Błażewicz et al. (2001).

Liao and Chen (2003) studied a single-machine scheduling problem with nonresumable operations and periodic maintenance, i.e., maintenance is required after fixed interval. A branch and bound algorithm is developed to solve the problem to optimality and a heuristic is also developed for large problem instances. Aggoune (2004) considers an $m$-machine flow shop problem with non-resumable operations and several unavailability periods on each machine. They proposed a heuristic algorithm based on tabu search and genetic algorithm to solve the problem. Allaoui et al., (2006) investigated a two-stage hybrid flow shop problem with one machine in the first stage and $m$ machine in the second stage. Each machine is subject to only one deterministic unavailability period and the operations are non-resumable. A branch and bound model is presented for the problem
along with three heuristics. Ma et al. (2010) presents a comprehensive review of scheduling problems with limited machine availability from 1996 to 2009. Huo and Huang (2016) proposed two algorithms based on ant colony to solve an $m$-machine flow shop scheduling problem with limited machine availability. The objective is to minimize total flow time.

The problem addressed in this paper also includes limited machine availability constraint since the production can occur only in a single or double shift, resulting in machine unavailability during remaining hours of the day. However, to the best of our knowledge, an assembly flow shop problem with limited machine availability and limited buffer storage. has not been addressed so far.

## 3. PROBLEM STATEMENT

The problem consists of scheduling $N$ jobs belonging to $p$ different products, where each product contains $n_{i}$ jobs, i.e. $\sum_{\mathrm{i}=1}^{\mathrm{p}} \mathrm{n}_{\mathrm{i}}=\mathrm{N}$. Based on the three field notation $\alpha|\beta| \gamma$ developed by Graham et al. (1979), the problem addressed in this research can be characterized by $A F_{m}, h_{j k} \mid n r-a, S T_{s d}, r_{j}, a_{j}$, block, $\mid F_{l}\left(\alpha \sum S_{i}, \beta \sum w_{j} T_{j}\right)$. The first field ( $\alpha$ ) describes the machine setting, the second field ( $\beta$ ) describes the job characteristics and process constraints and the third field $(\gamma)$ defines the objective function of the problem. The problem includes the following features

- The machine setting resembles that of an assembly flow shop, where each component of a job is first processed independently on different machines and then assembled together at the final machine. The number of operations required by each component before it is ready for assembly might be different (as shown in Figure 2). Two components are required to form a job. Component 1 has two operations (Machines $M_{1}$ and $M_{2}$ ) before assembly and component 2 (Machine $M_{3}$ ) has one operation before assembly (Machine $M_{4}$ ). The machines are not available continuously throughout the planning horizon. The production occurs in one, two or three shifts. In case of one or two shifts (8 and 16 hours each day, respectively), the machines are not available for the remaining period of the day. In addition, if an operation cannot be completed before the end of production hours, it has to be restarted when the production begins the next day, i.e., non-resumable operations.
- The setup time is sequence- and machine-dependent, i.e., the setup time required on a machine depends on the previous job processed on that machine. Setup time


Figure 2. Layout of an uneven assembly flow shop
between jobs belonging to the same product is less than the setup time between jobs from different products. Since the production is not continuous, all machines have to be restarted at the start of each production day. Hence, at the start of each production day, the setup time for each machine does not depend on the previous job on that machine but is dependent on the machine startup time (pre-heat, pressure buildup, etc.).

- Jobs have dynamic release times. In other words, all jobs may not be available the start of the planning horizon.
- Each machine has dynamic availability time, which means that not all machines are available at the start of the planning horizon because they might be processing some jobs from the previous planning horizon.
- A limited buffer storage is available between two machines. As shown in Figure $2, S_{i}$ denotes the storage space available after machine $M_{i}$. A blocking constraint is introduced because of the limited buffer storage, i.e., operation on the upstream machine is blocked if there is no storage space available. In addition, there is a minimum wait time for each job at certain storage locations. In Figure 2, $S_{1}$ and $S_{3}$ have a minimum storage time of 2 hours.
- The objective function focuses on minimizing the linear combination of total setup time and weighted tardiness. Each job in the problem is assigned a weight, representing priority level of the job. The job with higher weight receives greater priority. In real industry practice, jobs might have different weights assigned to them depending on associated customer's status, profit margin, etc. Since the two criteria in the objective function are not measured on a commensurate basis, these criteria are first normalized into a dimensionless quantity in the range of 0 and 1. This is done so that the algorithm does not favor one criteria over the other based on the values that represent these criteria. This is discussed in detail in Chapter 4.


## 4. MATHEMATICAL MODELS

A mixed-integer linear programming (MILP) model that represents the constraints of the industrial setting is developed to evaluate the performance of proposed algorithm. Two MILP models are developed, one considering permutation sequence named MILP1, and the other considering non-permutation sequence named MILP2. The two criteria, setup time and tardiness, used in the objective function are normalized to avoid skewness that may arise due to difference in value of these criteria.

### 4.1. Normalization of the Objective Function

A bicriteria objective function is used for the problem which aims to simultaneously minimize the linear combination of total setup time and weighted tardiness. Since, the value of these two criteria might not be in the same range, i.e., one criteria might have a much higher value than the other, skewness might arise as the objective function would favor the criteria with larger value. For example, consider a sequence of a problem with setup time $(S T)$ of 200 min . and weighted tardiness (WT) of 2000 mins . Both the producer's weight $(\alpha)$ and the customers' weight $(\beta)$ are 0.5 . Then the weighted objective function (without normalization) is given by:
$\alpha S T+\beta W T$

The value of the weighted objective function from equation 4.1 is 1100 , where the contribution of weighted tardiness to the objective function is ten times the contribution of setup times. Hence, the algorithm using this objective function will favor the minimization of weighted tardiness more than the minimization of setup time in spite of the fact that the producer's and customers' weights are equal. Therefore, in this research, the criteria are normalized using equation 4.2.

$$
\begin{equation*}
X_{N}=\frac{X-X_{\min }}{X_{\max }-X_{\min }} \tag{4.2}
\end{equation*}
$$

where,
$X_{N}=$ Normalized value of the criteria
$X=$ Value of the criteria for a given schedule
$X_{\max }=$ Maximum value of the criteria (nadir point)
$X_{\text {min }}=$ Minimum value of the criteria (ideal point)

The minimum and maximum values of a criterion are called extreme values. The method of obtaining these extreme values is explained in section 5. After the two criteria in the objective function are normalized, the normalized composite objective function $(N C O F)$ is obtained as shown in equation 4.3.

NCOF $=\alpha S T_{N}+\beta W T_{N}$
where,
$S T_{N}=$ Normalized value of total setup time
$W T_{N}=$ Normalized value of weighted tardiness

### 4.2. MILPP1

MILP1 is the mathematical model formulated for permutation sequence. The indices, sets, parameters, decision variables, and the mathematical model are shown below.

## Indices

| $i, i^{\prime}$ | machines |
| :--- | :--- |
| $j, j^{\prime}$ | products |
| $k, k^{\prime}$ | jobs |
| $g, g^{\prime}$ | components |
| $l$ | time slot |

## Sets

$M \quad$ Set of all machines, $M=A M \cup\left(C M_{1} \cup C M_{2} \cup \ldots . \cup C M_{m}\right)$
$P \quad$ Set of products, $P=\{1,2, \ldots, p\}$
$J_{j} \quad$ Set of jobs belonging to product $j, J_{j}=\left\{1,2, \ldots, n_{j}\right\}$
$Q \quad$ Set of time slots for each machine, $Q=\{1,2, \ldots, q\}$

## Subsets

$\mathrm{CM}_{g}$

AM
$F M_{i}$
SM

EM

## Parameters

$m \quad$ Number of components
$u_{g}$
$p \quad$ Number of products
$n_{j} \quad$ Number of jobs belonging to product $j$
$q \quad$ Number of time slots, $q=\sum_{i=1}^{m} n_{j}$
$S_{i j j} \quad$ Setup time while changing from product $j$ to $j$ ' on machine (if $j=j$ ', then setup time is for the same product change)
$R_{i j} \quad$ Run time of product $j$ on machine $i$
$d_{j k} \quad$ Due date of job $k$ of product $j$
$r_{j k} \quad$ Release time of job $k$ of product $j$
$a_{i} \quad$ Machine availability of machine $i$
$z_{i} \quad$ Number of buffer storage after machine $i$
$t_{i} \quad$ Minimum wait time after being processed on machine $i$
$\alpha \quad$ Producer's weight
$\beta \quad$ Customers' weight
$w_{j k} \quad$ Weights assigned to job $k$ of product $j$;
$E_{i} \quad$ Restart time of machine $i$ at the start of each day
$S T_{\max } \quad$ Maximum value of setup time
$S T_{\text {min }} \quad$ Minimum value of setup time
$W T_{\text {max }} \quad$ Maximum value of weighted tardiness
$W T_{\text {min }} \quad$ Minimum value of weighted tardiness
$B M \quad$ Big-M, a large number
Sh Number of shift each day

Note: A shop environment with $m=2, u_{1}=2$ and $u_{2}=1$ would correspond to the layout in Figure 2.

## Decision Variables

$T s_{i l} \quad$ Start time of slot $l$ on machine $i$
$T f_{i l} \quad$ Finish time of slot $l$ on machine $i$
$T s s_{i j k} \quad$ Start time of batch $k$ of product $j$ on machine $i$
$T f_{i j k} \quad$ Finish time of batch $k$ of product $j$ on machine $i$
$W_{j k l} \quad 1$, if job $k$ of product $j$ is assigned to slot $l ; 0$ else
$Y_{j l j^{\prime}(l+1)} \quad 1$, if product $j$ is processed in slot $l$ and $j^{\prime}$ is processed in slot $(l+1) ; 0$ else
$S T_{i l} \quad$ Setup time at time slot $l$ on machine $i$
$H s_{i l} \quad$ Integer variable representing the starting day of slot $l$ on machine $i$
$T_{j k} \quad$ Tardiness of a job $k$ of product $j$

Model

$$
\begin{equation*}
\operatorname{Min} Z=\alpha \frac{\left(\sum_{i \in M} \sum_{l=1}^{q} S T_{i l}-S T_{\min }\right)}{\left(S T_{\max }-S T_{\min }\right)}-\beta \frac{\left(\sum_{j=1}^{p} \sum_{k=1}^{n_{j}} w_{j k} * T_{j k}-W T_{\min }\right)}{\left(W T_{\max }-W T_{\min }\right)} \tag{4.4}
\end{equation*}
$$

Subject to:

$$
\begin{align*}
& \sum_{l=1}^{q} W_{j k l}=1  \tag{4.5}\\
& j \in P ; k \in J_{j} \\
& \sum_{j=1}^{p} \sum_{k=1}^{n_{j}} W_{j k l}=1  \tag{4.6}\\
& l \in Q \\
& -B M\left(1-W_{j k l}\right) \leq T s s_{i j k}-T s_{i l} \quad i \in M ; j \in P ; k \in J_{j} ; l \in Q  \tag{4.7}\\
& B M\left(1-W_{j k l}\right) \geq T s s_{i j k}-T s_{i l} \quad i \in M ; j \in P ; k \in J_{j} ; l \in Q  \tag{4.8}\\
& T f_{i l}=T s_{i l}+\sum_{j=1}^{p} \sum_{k=1}^{n_{j}}\left(W_{j k l} * R_{i j}\right) \quad i \in M ; l \in Q  \tag{4.9}\\
& T f f_{i j k}=T s s_{i j k}+R_{i j} \quad i \in M ; j \in P ; k \in J_{j}  \tag{4.10}\\
& T s_{i(l+1)} \geq T f_{i l}+S T_{i(l+1)} \quad i \in M ; l \in Q \mid l \neq q  \tag{4.11}\\
& T s_{i 1} \geq a_{i}+S T_{i 1}+E_{i}  \tag{4.12}\\
& i \in M \\
& T s s_{i j k} \geq r_{j k}  \tag{4.13}\\
& i \in S M ; j \in P ; k \in J_{j} \\
& T s s_{i, j k} \geq T f f_{i j k}+t_{i} \quad i \in M \mid i \notin A M ; i^{\prime} \in F M_{i} ; j \in P ; k \in J_{j}  \tag{4.14}\\
& \sum_{j=1}^{p} \sum_{j^{\prime}=1}^{p} Y_{j l j^{\prime}(l+1)}=1 \quad l \in Q \mid l \neq q  \tag{4.15}\\
& Y_{j l j^{\prime}(l+1)} \leq \sum_{k=1}^{n_{j}} W_{j k l} \quad j, j^{\prime} \in P ; l \in Q \mid l \neq q  \tag{4.16}\\
& Y_{j l j(l+1)} \leq \sum_{k=1}^{n_{j^{\prime}}} W_{j k l} \quad \quad j, j^{\prime} \in P ; l \in Q \mid l \neq q  \tag{4.17}\\
& S T_{i l}=\sum_{j=1}^{p} \sum_{j^{\prime}=1}^{p}\left(Y_{j l j^{\prime}(l+1)} * S_{i j j^{\prime}}\right) \quad i \in M ; l \in Q \mid l \neq q  \tag{4.18}\\
& S T_{i 1}=\sum_{j=1}^{p} \sum_{k=1}^{n_{j}}\left(W_{j k 1} * S_{i 0 j}\right) \quad i \in M  \tag{4.19}\\
& T s_{i l} \geq T f_{i^{\prime}\left(l-z_{i}\right)} \quad i \in M \mid i \notin A M ; i^{\prime} \in F M_{i} ; l \in Q  \tag{4.20}\\
& H s_{i l} \geq T s_{i l} /(24 * 60) \quad i \in M ; l \in Q  \tag{4.21}\\
& H s_{i l} \leq T s_{i l} /(24 * 60)+1 \quad i \in M ; l \in Q  \tag{4.22}\\
& T s_{i l} \geq(24 * 60) *\left(H s_{i l}-1\right)+E_{i} \quad i \in M ; l \in Q  \tag{4.23}\\
& (24 * 60) * H s_{i l}-T f_{i l} \geq(3-S h) * 480 \quad i \in M ; l \in Q  \tag{4.24}\\
& T_{j k} \geq T f f_{i j k}-d_{j k} \quad i \in A M ; j \in P ; k \in J_{j} \tag{4.25}
\end{align*}
$$

$$
\begin{align*}
& T_{j k} \geq 0 \quad j \in P ; k \in J_{j}  \tag{4.26}\\
& W_{j k l} \in\{0,1\} ; Y_{j l j^{\prime}(l+1)} \in\{0,1\} ; H s_{i l} \in \text { int } \\
& \quad i \in M ; j \in P ; k \in J_{j} ; l \in Q \tag{4.27}
\end{align*}
$$

## Model Description

The proposed mathematical model is a mixed-integer linear programming model with both binary and general-integer variables. The objective function of the model is to minimize the linear combination of normalized setup time and normalized weighted tardiness as presented in (4.4). A slot based mathematical model is formulated. It is assumed that each machine consists of a set of slots which get occupied by the jobs being processed on that machine. Using these slots, the position of jobs in a given sequence can be identified. In this model, the number of slots is equal to the total number of jobs required to be scheduled. On every machine, each job can be assigned to only one slot. In addition, each slot on a machine should contain only one job. Constraint (4.5) states the requirement that each job must be processed only once on each machine, i.e., each job can be assigned to only one slot on every machine. Constraint (4.6) states that each machine can only process one job at a time, i.e., each slot can contain only one job. Constraint (4.7) and (4.8) are the big-M constraints, which state that if a job is assigned to a particular slot of a machine, the start time of that slot should be the same as the start time of the job. Constraint (4.9) defines that the ending time of a slot must be greater than the starting time of the slot plus the run time of the job assigned to that slot. Constraint (4.10) expresses the requirement that the ending time of the job must be greater than the starting time of the job plus the run time of the job on that machine. Constraint (4.11) states that, for every machine, the starting time of a slot must be greater than the ending time of the previous slot plus the setup time required for the changeover between jobs assigned to these slots. Constraint (4.12) states that the starting time of the first slot of a machine must be greater than the machine availability time plus the setup time. Constraint (4.13) states that, for every machine belonging to set $S M$ (Set of machines, which first processes each individual components), the start time of a job is greater that the release time of the job. Constraint (4.14) states that the start time of a job on a machine must be greater than the ending time of that job on predecessor machine plus the minimum storage time between these machines. Constraint (4.15), (4.16)
and (4.17) work together to quantify a binary variable representing change in batch or product between successive slots. Constraints (4.18) and (4.19) obtain the setup time of every slot of each machine. Constraint (4.20) states that the number of jobs stored between machines cannot be more than the storage capacity, i.e. if the storage capacity between machine 1 and 2 is five, then machine 2 must have completed processing of $3^{\text {rd }}$ job in its sequence (or $3^{\text {rd }}$ slot) if machine 1 needs to start processing $8^{\text {th }}$ batch in its sequence. The storage space is emptied only if the job is completed on the downstream machine. If a job is in-progress, then the storage space is partially occupied, which prevents the upstream machine from using this space for storage. Constraints (4.21), (4.22), (4.23), and (4.24) work together to ensure that a job started on a machine gets completed before the machine becomes unavailable at the end of the shift. Constraints (4.25) and (4.26) calculate the tardiness for the batch. (4.27) defines the variable used.

Note: If the production occurs in three shifts, i.e., continuous machine availability, then constraint (4.21) through (4.24) would not be required.

### 4.3. MILP2

MILP2 is the mathematical model formulated for non-permutation sequence. The indices, sets and parameters are identical to the MILP1 and hence, is not repeated below. The number of binary decision variables in this formulation is higher as compared to MILP1, to allow for different job sequences on different machines.

## Decision Variables

$T s_{i l} \quad$ Start time of slot $l$ on machine $i$
$T f_{i l} \quad$ Finish time of slot $l$ on machine $i$
$T s s_{i j k} \quad$ Start time of batch $k$ of product $j$ on machine $i$
$T_{f f_{i j k}} \quad$ Finish time of batch $k$ of product $j$ on machine $i$
$W_{i j k l} \quad 1$, if job $k$ of product $j$ is assigned to machine $i$ in slot $l ; 0$ else
$Y_{i j j^{\prime}(+1)} \quad 1$, if product $j$ is processed in slot $l$ and $j^{\prime}$ is processed in slot $(l+1)$ on machine $i ; 0$ else
$S T_{i l} \quad$ Setup time at time slot $l$ on machine $i$
$H s_{i l} \quad$ Integer variable representing the starting day of slot $l$ on machine $i$
$T_{j k} \quad$ Tardiness of job $k$ of product $j$

## Model

$\operatorname{Min} Z=\alpha \frac{\left(\sum_{i \in M} \sum_{l=1}^{q} S T_{i l}-S T_{\min }\right)}{\left(S T_{\max }-S T_{\min }\right)}-\beta \frac{\left(\sum_{j=1}^{p} \sum_{k=1}^{n_{j}} w_{j k} * T_{j k}-W T_{\min }\right)}{\left(W T_{\max }-W T_{\min }\right)}$
Subject to:

$$
\begin{array}{ll}
\sum_{l=1}^{q} W_{i j k l}=1 & i \in M ; j \in P ; k \in J_{j} \\
\sum_{j=1}^{p} \sum_{k=1}^{n_{j}} W_{i j k l}=1 & i \in M ; l \in Q \\
-B M\left(1-W_{i j k l}\right) \leq T s s_{i j k}-T s_{i l} & i \in M ; j \in P ; k \in J_{j} ; l \in Q \\
B M\left(1-W_{i j k l}\right) \geq T_{s} s_{i j k}-T s_{i l} & i \in M ; j \in P ; k \in J_{j} ; l \in Q \\
T f_{i l}=T s_{i l}+\sum_{j=1}^{p} \sum_{k=1}^{n_{j}}\left(W_{i j k l} * R_{i j}\right) & i \in M ; l \in Q \\
T f f_{i j k}=T s s_{i j k}+R_{i j} & i \in M ; j \in P ; k \in J_{j} \\
T s_{i(l+1)} \geq T f_{i l}+S T_{i(l+1)} & i \in M ; l \in Q \mid l \neq q \\
T s_{i 1} \geq a_{i}+S T_{i 1}+E_{i} & i \in M ; j \in P ; k \in J_{j} \\
T s s_{i j k} \geq r_{j k} & i \in M \mid i \notin A M ; i^{\prime} \in F M_{i} ; j \in P ; k \in J_{j} \\
T s s_{i \prime j k} \geq T f f_{i j k}+t_{i} & i \in M ; l \in Q \\
\sum_{j=1}^{p} \sum_{j=1}^{p} Y_{i j l j^{\prime}(l+1)}=1 & i \in M ; j \in P ; l \in Q \mid l \neq q \\
Y_{i j l j^{\prime}(l+1)} \leq \sum_{k=1}^{n_{j}} W_{i j k l} & \\
Y_{i j l j^{\prime}(l+1)} \leq \sum_{k=1}^{n_{j^{\prime}}} W_{i j k l} & i \in M ; j \in P ; l \in Q \mid l \neq q
\end{array}
$$

$$
\begin{array}{lc}
S T_{i l}=\sum_{j=1}^{p} \sum_{j^{\prime}=1}^{p}\left(Y_{i j l j^{\prime}(l+1)} * S_{i j j^{\prime}}\right) & i \in M ; l \in Q \mid l \neq q \\
S T_{i 1}=\sum_{j=1}^{p} \sum_{k=1}^{n_{j}}\left(W_{i j k 1} * S_{i 0 j}\right) & i \in M \\
T s_{i l} \geq T f_{i^{\prime}\left(l-z_{i}\right)} & i \in M \mid i \notin A M ; i^{\prime} \in F M_{i} ; l \in Q \\
H s_{i l} \geq T s_{i l} /(24 * 60) & i \in M ; l \in Q \\
H s_{i l} \leq T s_{i l} /(24 * 60)+1 & i \in M ; l \in Q \\
T s_{i l} \geq(24 * 60) *\left(H s_{i l}-1\right)+E_{i} & i \in M ; l \in Q \\
(24 * 60) * H s_{i l}-T f_{i l} \geq(3-S h) * 480 & i \in M ; l \in Q \\
T_{j k} \geq T f f_{i j k}-D D_{j k} & i \in A M ; j \in P ; k \in J_{j} \\
T_{j k} \geq 0 & j \in P ; k \in J_{j} \\
W_{i j k l} \in\{0,1\} ; Y_{i j l j^{\prime}(l+1)} \in\{0,1\} ; H s_{i l} \in i n t & i \in M ; j \in P ; k \in J_{j} ; l \in Q
\end{array}
$$

## Model Description

The functionality of the model is identical to MILP1. The only difference is the increase in the number of binary variables denoted by $W$ and $Y$. In MILP2, there is an additional index $i$, representing machines, included in these binary variables. In MILP2, the sequence of jobs might be different on different machines. Hence, the additional index is required to be able to identify the position of each job on each machine. The number of these binary variables in MILP2 is given by the number of binary variables in MILP1 times the number of machines in the system. This increase in binary variables causes the MILP2 to be much more complex as compared to MILP1.

### 4.4. Choice of the Objective Function

This research aims to simultaneously minimize two criteria, one representing producer's interest and the other representing customers' interest. In most of the scheduling problems, producer's interest is represented by minimizing makespan or minimizing sum of completion times. These criteria depend on the completion time of jobs on the last machine in the flow line. Hence, the schedule obtained from these criteria might not yield
the best performance values for upstream machines. Consider a two-machine flow shop with limited buffer storage in between and a minimum wait time of two hours. Production occurs for a single shift each day, i.e., production time available each day is 480 minutes. Table 1-4 shows the related data for this problem. Figure 3 shows a Gantt chart of two sequences: schedule 1 considering permutation sequence and schedule 2 considering nonpermutation sequence.

Table 1. Runtimes of the product

| Machine | $\boldsymbol{P}_{\mathbf{1}}$ | $\boldsymbol{P}_{\mathbf{2}}$ | $\boldsymbol{P}_{\mathbf{3}}$ |
| :--- | :--- | ---: | ---: |
| $\boldsymbol{M}_{\mathbf{1}}$ | 65 | 95 | 110 |
| $\boldsymbol{M}_{\mathbf{2}}$ | 85 | 120 | 95 |

$P_{i}$ refers to the $i^{\text {th }}$ product and $M_{k}$ refers to the $k^{\text {th }}$ machine

Table 2. Machine availability times

| Machine | Availability <br> Time | Startup <br> time |
| :--- | :---: | :---: |
| $\mathbf{M}_{1}$ | 65 | 30 |
| $\mathbf{M}_{2}$ | 265 | 20 |

Table 3. Setup times

|  | Subsequent Jobs |  |  |  |
| :---: | :---: | ---: | ---: | ---: |
| Preceding | $\boldsymbol{M}_{\mathbf{1}}$ | $\boldsymbol{P}_{\mathbf{1}}$ | $\boldsymbol{P}_{\mathbf{2}}$ | $\boldsymbol{P}_{\mathbf{3}}$ |
|  | Ref, | 10 | 15 | 5 |
|  | $\boldsymbol{P}_{\mathbf{1}}$ | 5 | 9 | 10 |
|  | $\boldsymbol{P}_{\mathbf{2}}$ | 15 | 4 | 15 |
|  | $\boldsymbol{P}_{\mathbf{3}}$ | 9 | 10 | 5 |
| Preceding <br> Jobs | $\boldsymbol{M}_{\mathbf{2}}$ | $\boldsymbol{P}_{\mathbf{1}}$ | $\boldsymbol{P}_{\mathbf{2}}$ | $\boldsymbol{P}_{\mathbf{3}}$ |
|  | Ref. | 10 | 15 | 5 |
|  | $\boldsymbol{P}_{\mathbf{1}}$ | 8 | 12 | 15 |
|  | $\boldsymbol{P}_{\mathbf{2}}$ | 15 | 5 | 10 |
|  | $\boldsymbol{P}_{\mathbf{3}}$ | 10 | 14 | 4 |

Table 4. Due date and release time of a job

| Job | $\boldsymbol{J}_{\mathbf{1 1}}$ | $\boldsymbol{J}_{\mathbf{1 2}}$ | $\boldsymbol{J}_{\mathbf{2 1}}$ | $\boldsymbol{J}_{\mathbf{3 1}}$ | $\boldsymbol{J}_{\mathbf{3 2}}$ |
| :--- | :---: | :---: | :---: | :---: | :---: |
| Release time | 0 | 35 | $\mathbf{1 3 0}$ | 0 | 200 |
| Due date | 1700 | 1920 | 1800 | 480 | 1790 |

$J_{i j}$ refers to the $j^{\text {th }}$ job belonging to the $i^{\text {th }}$ product

In these schedules, jobs $J_{11}, J_{32}$ and $J_{21}$ are processed in day 1 on $M_{1}$, stored overnight and processed the next day on $M_{2}$. Changing the order of these jobs on $M_{1}$, does not affect $M_{2}$ since these jobs are processed the next day on $M_{2}$. The order of these jobs on $M_{1}$ is different for sequences 1 and 2 . In sequence 1 , the order for these jobs on $M_{1}$ is $J_{11^{-}}$ $J_{32}-J_{21}$, whereas in sequence 2 , the job order on $M_{1}$ is $J_{32}-J_{11}-J_{21}$. This change has decreased the completion time of the last job on $M_{1}$ of day 1 by 6 minutes in the case of sequence 2 (schedule 1: 479 min ., schedule 2:473 min.). However, the timings on $M_{2}$ is not affected by this change. Because of this, the values of the criteria, makespan and sum of completion time, is the same for both sequences. Thus, these criteria won't be able to differentiate
between sequence 1 and sequence 2 . In contrast, minimizing the sum of setup time can capture this difference. In this case, minimizing setup time is equivalent to minimizing makespan, contributed by each machine on every production day. Hence, the sum of setup time is chosen to represent the producer's interest in this research.


Figure 3. Gantt chart for a PN and NPN Schedule in a two-machine flow shop

### 4.5. Complexity of the Problem

Scheduling problems can be characterized comprehensively using the mathematical model. In this model, if the value of producer's weight $(\alpha)$ is equal to zero, then the problem turns into a single criteria objective function of minimizing weighted tardiness. Furthermore, if the number of components $(m)$ is considered to be one and there is only one machine in the first stage, then the problem converts to a two-machine regular flow shop problem. Thus, the two-machine flow shop problem is a special case of the assembly flow shop problem addressed in this research. Koulamas (1994) proved that the two-
machine flow shop problem with tardiness minimization is NP-hard in a strong sense. Thus, the research problem can easily be shown to be NP-hard in a strong sense since the special case of the research problem is also NP-hard in a strong sense.

Despite the recent technological advances on processing speeds, it is unlikely that an optimal solution for an NP-hard problem can be obtained within a reasonable time. Only small-size problems can be solved to optimality within a reasonable computation time using an optimization software. For medium- and large-size problems, the software often fails to identify the optimal solution. Therefore, an algorithm needs to be developed to obtain optimal or near-optimal solution for medium- and large-size problems.

## 5. HEURISTIC ALGORITHM

Heuristics and meta-heuristics are the main approaches used to solve medium- and large-size problems for an NP-hard scheduling problem. Metaheuristic algorithms usually show a better performance as compared to heuristic algorithms due to their ability to avoid being trapped in local optima. Metaheuristic algorithms can generally be classified into local search-based and population-based algorithm. In this research, a local search-based metaheuristic (short-term tabu search) and a hybrid algorithm, combining local search and population-based meta-heuristic (tabu search/path relinking), is proposed. A local searchbased algorithm searches for a solution in the local area in the solution space. It starts with an initial solution which is used as the initial seed. At each iteration, the algorithm searches the neighborhood of the seed solution to find a possible solution for the next seed. The next seed is chosen based on its quality as compared to other neighborhood solutions. A population-based algorithm starts with a series of solutions called initial population. At each iteration, the algorithm replaces the older population with another population with superior characteristics.

Al-Anzi and Allahverdi (2006) showed that tabu search methods are good metaheuristics for assembly flow shop problems. They also stated that the hybrid tabu search algorithm performs better than the regular tabu search method. Shahvari and Logendran (2017) proposed two heuristics, based on tabu search (TS) and hybrid tabu search/path relinking (TS/PR), for a hybrid flow shop batching problem. They showed that the hybrid TS/PR outperforms the TS algorithm, especially for complex problems. In this research, four algorithms are proposed where two algorithms are based on short-term TS and the other two algorithms are based on tabu search/path relinking. The nomenclature of the algorithms is shown in Table 5.

Table 5. Nomenclature of algorithms used in this research

| Sequence | Short-term tabu search | Tabu search/path relinking |
| :--- | :---: | :---: |
| Permutation | ALG1 | ALG3 |
| Non-permutation | ALG2 | ALG4 |

### 5.1. Tabu Search

Tabu search was first proposed by Glover (1986) to solve complex optimization problems. The concept of TS was finalized later by Glover $(1989,1990)$ and Laguna et al. (1991). Tabu search has been used extensively to solve complex scheduling problems in the past and all these studies have shown that this method is capable of producing good quality solutions (Nowicki and Smutnicki 1996, Eren and Güner 2006, Mehravaran and Logendran 2012, Shahvari et al. 2012). Aggoune (2004) proposed a tabu search-based algorithm for a flow shop problem with availability constraint. Liao and Huang (2010) developed two heuristics, based on tabu search, to solve a non-permutation flow shop problem with the objective of minimizing tardiness.

Tabu search is a refined form of a popular local search heuristic, namely hill climbing heuristic. The hill climbing heuristic starts with an initial solution and moves progressively towards a better solution at each iteration. The heuristic terminates when a local optimum is found, i.e. if a better solution cannot be found. Tabu search solve this issue by settling for a solution that is inferior to the previous solution. Similar to the hill climbing heuristic, TS starts with an initial solution, called seed, and employs a set of moves to generate the neighborhood solutions from the seed. The neighborhood solutions are evaluated based on the objective function value and the best solution found is chosen as a seed for the next iteration. If a solution better than the previous iteration is not found, the algorithm settles for an inferior solution. TS employs a flexible memory structure to store the information during the search process. This enables the algorithm to guide the search in a more effective and economical manner. Tabu list is a short-term memory structure which stores solutions or configurations which have been explored in the recent past. If a solution is marked as tabu, then it is disregarded which prevents the algorithm from selecting solutions which have already been explored. However, this restriction can be overridden, if the disregarded solution results in a solution quality which is better than the aspiration level. The aspiration level is the best quality solution found so far in the search algorithm.

The following are the major components of the tabu search:

- Initial Solution: An initial solution (IS) is required to trigger the search algorithm. It can be chosen randomly or generated by a systematic IS generating mechanism.
- Neighborhood Functions: A neighborhood function, also called perturbations, generates a set of alternate solutions from the seed. It employs a set of moves to change the seed solution and thus, generates its neighborhood solutions.
- Objective function evaluation: The purpose of the algorithm is to identify an optimal or near-optimal solution with minimum objective function value. Hence, a mechanism is needed to evaluate the objective function value of each solution generated during the search procedure.
- Tabu List: At each iteration, the move that resulted in the best solution is stored in the tabu list (TL). As long as the move is in the TL, the move is considered restricted, i.e., the solution obtained using tabu move cannot be selected unless the solution satisfies the condition for an override, meaning that the solution quality must be better than the aspiration criteria. This tabu restriction prevents the search algorithm from revisiting the solution previously explored and thus enables it to avoid being trapped in the same search space. The number of iterations for which a move remains tabu is determined by the tabu list size (TLS). In its simplest form, the TLS is set as 1 , which means there is only one move stored in the TL and this move corresponds to the move that was used to generate the new seed in the previous iteration. The neighborhood of the current seed always contains the parent seed. If the current seed is inferior to the parent seed, then at current iteration, the parent seed might be selected again. This causes the algorithm to cycle back and forth between the same set of solutions. However, this issue is prevented due to the tabu status placed on the move. TL is updated by removing the earliest entry to the list before adding the new move as tabu. TL is a short-term memory structure.
- Aspiration Level: The aspiration level (AL) records the objective function value of the best solution found so far by the search algorithm. Because the TL only stores a part of the information (moves) about the solution, some good solutions might be discarded. To prevent this, a tabu restriction override is included, which allows a solution with tabu move to be selected if the solution is of better quality than the AL.
- Temporary Candidate List: A temporary candidate list (TCL) contains the objective function value of all the neighborhood solutions from the current seed. The TCL is updated at each iteration when a new seed is selected.
- Candidate List: The best solution found at each iteration, including the initial solution, is included in the candidate list (CL). Each solution in the CL has a number of stars associated with it, which denotes the status of the solution: local optimum (2 stars), improving solution or potential local optimum (1 star), nonimproving solution ( 0 star). The CL is also an explicit memory structure which prevents the duplication of the solution into the CL.
- Index List: If a solution turns out to be a local optimum, which is at least as good as both its parent and child, then it is included in the index list (IL). At the end of the search, the best solution in the IL is selected as the final solution.
- Stopping Criteria: The search algorithm terminates if certain stopping criteria are met. Several stopping criteria can be used, such as maximum number of iterations without improvement (MIWOI), maximum number of local optima in the IL (MIL), or maximum computational time (MCT). If a new solution added into the CL is not better than the previous entry into the CL, the iteration without improvement (IWOI) is increased by one; else, it is reset to zero. If the value of IWOI is greater than MIWOI, or the number of entries into the IL is greater than the MIL, or the maximum computation time is reached, the search algorithm is terminated.

The following subsection describes the details of the components used in the proposed algorithm.

### 5.1.1. Initial Solution Finding Mechanism

An initial solution finding mechanism is utilized to generate the IS for the search algorithm. A randomly generated IS is usually of poor quality and hence is time consuming or impossible to improve the solution to an optima. Logendran and Subur (2004) have shown that the final solution obtained from TS is sensitive to the quality of the IS used. Therefore, an IS generating mechanism is used to find a good quality IS. Simple dispatching rules have been used in the past to generate initial solutions for metaheuristic algorithms. For tardiness related problems, dispatching rules such as earliest due date (EDD), hybrid critical ratio (HCR), minimum slack (MSLACK) can be used. For completion time related problems, shortest processing time (SPT) can be used. For a bicriteria problem with weighted objective function, a combination of these rules can be used. Bozorgirad and Logendran (2013), and Shahvari and Logendran (2017) proposed an IS finding mechanism, inspired from weighted shortest processing time (WSPT) and weighted earliest due date (WEDD), for a bi-criteria scheduling problem with the objective of minimizing the linear combination of weighted flowtime and weighted tardiness. A producer's sequence (PS) and customers' sequence (CS) is generated using WSPT and WEDD, respectively. Then the PS and CS are combined, considering the normalization of their positional values $(\alpha . P S+\beta . C S)$. In this research, The $P S$ and $C S$ and generated as follows.

Producer's Sequence: The goal of the PS is to minimize the total setup time. The smallest setup time (SST) rule is used to determine the PS. In this rule, priority is given to the job with the least changeover time from the previous job assigned to that machine. To assign the first job, reference setup time is used. If two or more jobs have the same setup time, tie is broken in favor of the job that has the smallest product identification number. If two or more jobs have the same setup time and product ID, ties are broken in favor of the job with the smallest job identification number.

Customers' Sequence: The goal of the customer is to minimize the weighted tardiness. To determine the CS, WEDD rules is used. This rule assigns priority to jobs with the least due date to weight ratio $d_{j k} / w_{j k}$. With this rule, the job with the smallest due date and largest weight is scheduled first. Ties are broken in the same manner as in PS.

After finding the order of jobs in both PS and CS, the normalized positional value of each job is obtained by the formula $\alpha$. PS $+\beta$. CS), where $P S$ denotes the order of the job in producer's sequence and $C S$ denotes the order of the job in customer's sequence. The job with the least normalized positional value is sequenced first in the final sequence.

Only permutation sequence is considered during IS generation, i.e., for both permutation and non-permutation algorithms, the IS is a permutation sequence. The flowchart for IS generating mechanism is shown in Figure 4. The IS generation mechanism consists of the following steps:

1. Initially, set $g=1$.
2. Select the $g^{\text {th }}$ machine in the subset $S M$ (i.e., the first machine required by the $g^{\text {th }}$ component).
3. Include all jobs into a set NSJ.
4. Select the jobs from NSJ that are released before the machine becomes available. If no such job exists, select the job with the earliest release time.
5. If more than one job remains from the first two steps, the job with the smallest normalized positional value is chosen. Ties are broken in favor of the job with the smaller index (product ID used in conjunction with job ID), as explained in Producer's Sequence).
6. Remove the selected job from NSJ.
7. Repeat steps 4-6 until NSJ is empty.
8. Label the sequence obtained as $I S_{g}$
9. Set $g=g+1$;
10. Repeat steps 1 to 6 until $g>m$
11. Evaluate each $I S_{g}(g=\{1,2, \ldots, m\})$ in terms of the objective function value and select the best one as the IS.


Figure 4. IS Flowchart

### 5.1.2. Neighborhood Function

After the initial solution is generated, it is set as seed. A set of moves are performed on this seed to find neighborhood solutions (NS). These moves cause changes in the structure of the seed. There are two types of moves used in this research.

Swap move: In this move, the position of two jobs in a sequence is exchanged. Consider that the sequence of jobs on machine $1\left(M_{1}\right)$ is $J_{11}, J_{12}, J_{21}, J_{31}$, then the exchange moves between jobs in positions 1 and 3 would result in $J_{21}, J_{12}, J_{11}, J_{31}$ as a new sequence. This move is illustrated in Figure 5.


Figure 5. Swap move

Insert move: In this move, a job from a particular position of a sequence is inserted into another position. The old position and new position of the job cannot be adjacent, i.e., $\mid$ old position --new position $\mid \geq 2$. Insert move from a position to an adjacent position would result in the same sequence as the exchange move. Consider that the sequence of jobs on machine $1\left(M_{1}\right)$ is $J_{11}, J_{12}, J_{21}, J_{31}$, then the insert moves of a job from position 1 into position 3 would result in $J_{12}, J_{21}, J_{11}, J_{31}$ as a new sequence. This move is illustrated in Figure 6.


Figure 6. Insert move

A combination of swap and insert moves are used to generate NS. There is a difference in the neighborhood structure for permutation and non-permutation sequence,
which is shown in Fig. 7 and Fig. 8. The sequence shown in this figure corresponds to a 4machine assembly flow shop shown in Figure 2. Consider a job-pair ( $J_{11}$ and $J_{12}$ ) which is selected for swapping. In the case of permutation, only one neighborhood solution can be generated from this job pair. However, in the case of non-permutation, 15 NS (i.e., ( $2^{m}-1$ )) can be generated with a single job-pair by selecting various combinations of machines in which the swap move is applied. These combinations of machines are $1 ; 2 ; 3 ; 4 ; 1$ and $2 ; 1$ and $3 ; 1$ and $4 ; 2$ and $3 ; 2$ and $4 ; 3$ and $4 ; 1,2$ and $3 ; 1,2$, and $4 ; 1,3$ and $4 ; 2,3$ and $4 ; 1,2,3$ and 4 . In the first combination, the sequence is swapped only on machine 1 ; in the fifth combination, the sequence is swapped on machines 1 and 2; and so on. Liao and Huang (2010) used this structure to generate NS for non-permutation sequence. While the example above is shown for swap moves, the same concept applies for insert moves as well. Hence, the size of neighborhood solution set is much larger for non-permutation sequence. With this increase in the size of neighborhood solution set, the computational time required by the algorithm also increases, as each solution in the set must be evaluated.


Figure 7. Neighborhood structure for permutation


Figure 8. Neighborhood structure for non-permutation

Two types of perturbation mechanisms are used in this research. Perturbation 1 (PTB1) only considers an adjacent pair for swapping, whereas perturbation 2 (PTB2) considers all possible pairs for swap and insert moves. In this research, the TS which uses PTB1 is called slight TS and the TS which uses PTB2 is called strong TS. The possible moves for PTB1 and PTB2 are shown in Table 6. As mentioned before, each possible move shown in Table 6 results in one neighborhood solution for PN sequence and 15 NS for NPN sequence. Since PTB1 has a small neighborhood structure as compared to PTB2, the computational time of slight TS is lower than that of strong TS. However, the quality of the solution is superior in the case of strong TS (Aryal and Logendran, 2018). The shortterm TS, i.e., ALG1 and ALG2, uses PTB2 in this research.

Table 6. Moves for PTB and PTB2

| Perturbation | Move type | Possible moves |
| :--- | :--- | :--- |
| PTB1 | Swap | $1-2,2-3,3-4,4-5$ |
| PTB2 | Swap | $1-2,1-3,1-4,1-5,2-3,2-4,2-5,3-4,3-5,4-5$ |
|  | Insert | $1-3,1-4,1-5,2-4,2-5,3-1,3-5,4-1,4-2,5-1,5-2,5-3$ |

Swap move $(a-b)=$ jobs in position $a$ and $b$ are swapped
Insert move $(a-b)=$ job in position $a$ is inserted into position $b$

### 5.1.3. Evaluation of the Objective Function

Each neighborhood solution generated during the perturbation, must be evaluated in terms of the objective function. A weighted bi-criteria objective function with setup time and weighted tardiness is used in this research. As discussed in section 4.1, these two criteria are normalized using extreme values, i.e., minimum and maximum, of these criteria (refer to equation 4.2). These extreme values are obtained as shown in Table 7.

Table 7. Extreme values of the criteria

| Criteria | Minimum Value | Maximum Value |
| :--- | :--- | :--- |
| Total setup <br> time | Value of the criteria from a <br> solution obtained using SST | Value of the criteria from a <br> solution obtained using LST |
| Total weighted <br> tardiness | 0 | Value of the criteria from a <br> solution obtained using WLDD |

SST = smallest setup time, LST = largest setup time, WLDD = weighted latest due date

The minimum value for the total setup time $\left(S T_{m i n}\right)$ is obtained using smallest setup time (SST) rule. In this rule, a sequence is obtained for each machine independently by giving priority to the job with the smallest setup time with the previous job scheduled on that machine. Similar to initial solution generation, ties are broken in favor of the job with smaller index. The sum of the setup time for the sequence obtained on each machine gives the minimum total setup time. The maximum value for the total setup time $\left(S T_{\max }\right)$ is obtained using largest setup time rule (LST) rule which gives priority to jobs with largest setup time with the previous job scheduled on a machine. The minimum weighted tardiness ( $W T_{\text {min }}$ ) is considered as zero because in an ideal situation, no jobs would be tardy. The maximum value of the weighted tardiness $\left(W T_{\max }\right)$ is obtained with the methodology similar to that of the initial solution generation mechanism. However, instead of prioritizing jobs using normalized positional values, priority is given to the job with the largest due date to weight ratio. The tardiness value of this sequence is evaluated and set as $W T_{\max }$. The extreme values used to calculate the normalized composite objective function (refer to equation 4.3) are considered to be the same for PN and NPN sequence. This is done so that the performance of PN vs NPN sequence can be compared over the same range of maximum and minimum values. After the extreme values are obtained, the $N C O F$ is computed by calculating the completion times and setup times of the sequence.

The total setup time of a sequence can be obtained easily from the setup information. To calculate the weighted tardiness, the completion time of the jobs at the final assembly stage needs to be obtained. In a typical flow shop, this is done by calculating the completion time of all jobs sequentially at each stage, i.e. first stage, followed by the second stage and so on. However, this approach is not applicable for this research problem due to limited storage space constraint. Because of this constraint, the predecessor machine must determine whether the successor machine has processed enough jobs for the storage space to be available or else, the operation is blocked on the predecessor machine. Hence, in order to calculate the completion time, the algorithm goes back and forth between different machines to determine whether the storage space is empty (required for predecessor machine) and whether the job is ready for processing (required for successor machine). A counter $c_{i}$ is used for each machine $i$, which represents the order of a job in
the sequence which is to be processed on a machine, i.e., if $c_{1}=1$, then the first job in the sequence needs to be processed. Initially, the counters are set to one. The algorithm then moves sequentially between machines starting with machine belonging to $S M$, followed by $E M$ and then AM. At each iteration, the completion time of the job at the counter value position of the sequence is calculated. For example, if the counter $c_{1}$ is equal to 5 at a particular iteration, then the completion time of the job occupying the fifth position of the $M_{1}$ sequence is calculated. Completion time is calculated using the formula in 5.1. If the calculated completion time of the job falls during non-production hours (when the machines are not available), then the processing of the job is postponed to the next available day. This is done by increasing the machine availability time to the earliest time it is available on the next production day. For example, if the calculated completion time of a job equals 500 min ., then this completion time is not valid (since a single shift runs only 480 min . each day). Hence, the machine availability time is increased to the start time of next production day plus the equipment restart time, i.e. 1440 (start of next shift) +25 min. (equipment re-start time).

## Completion time of a job

$$
\begin{align*}
& =\text { Max (job release time, machine availabilty time }+ \text { setup time }) \\
& + \text { run time } \tag{5.1}
\end{align*}
$$

The job release times for different machines are calculated as follows:

- For machine $i \in S M$

$$
\begin{equation*}
\text { Job release time }=r_{j k} \tag{5.2}
\end{equation*}
$$

Here, the job release time is equal to the initial release time of the job.

- For machine $i \in M \mid i \notin S M, A M$

$$
\begin{equation*}
\text { Job release time }=T f f_{i, j k}+t_{i,}: i \in F M_{i^{\prime}} \tag{5.3}
\end{equation*}
$$

Here, the job release time is equal to the completion time on predecessor machine plus the minimum storage time.

- For machine $i \in A M$

$$
\begin{equation*}
\text { Job release time }=\operatorname{Max}\left(T f f_{i^{\prime} j k}+t_{i^{\prime}}\right): i^{\prime} \in E M \tag{5.4}
\end{equation*}
$$

Here, the job release time is equal to the maximum completion time on predecessor machine plus the minimum storage time.

For the 4-machine assembly flow shop shown in Figure 2, the job release time would be evaluated as follows:

$$
\text { Job release time }=\left\{\begin{array}{c}
r_{j k}, \text { for } i=1,3 \\
T f f_{1 j k}+t_{1}, \text { for } i=2 \\
\operatorname{Max}\left(T f f_{2 j k}+t_{2}, T f f_{3 j k}+t_{3}\right), \text { for } i=4
\end{array}\right.
$$

The iteration continues until all of the counter values are equal to $N$. If at any iteration, the machine is blocked due to storage space unavailability or job not completed on previous machine, then scheduling for that machine is skipped without updating the counter. Hence, at the next iteration, the counter value for that machine will be the same and the algorithm will try to schedule the same job again.

Infeasible solutions: In a flow shop, there is interdependency between positions of a job in different stages/machines. In a permutation sequence, the job sequence across all machines is the same. Hence, the interdependency is not a concern. In non-permutation sequence, however, performing a perturbation may result in increase of idle times and even infeasibility. Consider the job sequence on machines 1 and 2 to be seq-1a $\left(J_{11}, J_{22}, J_{13}, J_{21}\right.$, $J_{12}$ ) and seq-2a ( $J_{22}, J_{11}, J_{12}, J_{13}, J_{21}$ ). If a perturbation is performed on seq-1a by swapping the positions of $J_{22}$ and $J_{12}$, the new sequence would be seq-1b $\left(J_{11}, J_{12}, J_{13}, J_{21}, J_{22}\right)$ and seq-2b ( $\left.J_{22}, J_{11}, J_{12}, J_{13}, J_{21}\right)$ for machines 1 and 2 , respectively. Here, Job $J_{22}$ is in the fifth position in seq- 1 b and in the first position in seq- 2 b . So, machine 2 must wait a long time before $J_{22}$ is available for processing. If the number of silos between these machines is 4 , then the solution is infeasible. Jobs $J_{11}, J_{12}, J_{13}$ and $J_{21}$ are first processed by machine 1 , which fills up the storage silos. The processing of $J_{22}$ is then halted because storage space is not available. Machine 2 also cannot process the first job $\left(\mathrm{J}_{22}\right)$ in its sequence because it is not yet processed by machine 1 . In order to avoid selection of infeasible solutions, a
penalty is imposed on the OFV of the infeasible solutions, so that the algorithm will move away from infeasible solution space. For each storage constraint violation, a penalty of 0.25 PD is imposed. $P D$ refers to the difference between the positions of the same job on two machines that are connected by a buffer storage. For example, if a job is at the sixth position on a sequence for machine 1 , and the same job is at the first position on sequence for machine 2, then the $P D$ of that job would be 5 . The total penalty is calculated for each storage violation and this penalty is added to a pre-set value of 1 . Since, the NCOF for a feasible sequence ranges from 0 to 1 , the proposed method ensures that the OFV of infeasible solutions is higher than that of feasible solutions.

### 5.1.4. Tabu list

TL is a short-term memory structure which prevents the search algorithm from selecting solutions which have already been explored to avoid being trapped in local optima. TL stores the most recent moves applied during the search. The entries in the TL follows first-in-first-out (FIFO) rule. This means that once the TL reaches its maximum size, the oldest entry is removed, and a new move is inserted into the TL. The number of iterations for which the move remains in the TL is determined by TLS.

In this research, two types of moves, swap and insert, are used to generate NS. Hence, two types of tabu structure are implemented into the algorithm. As discussed in section 5.2.2, a move can be applied to a partial set of machines in the case of nonpermutation sequence. Hence, the set of machines where the moves are applied also needs to be included in the structure. If a job $J_{j k}$ is swapped with another job $J_{j^{\prime} k^{\prime}}$ on machine $i$ and $i^{\prime}\left(j k \neq j^{\prime} k\right.$ ' $)$, the TL stores the index of the two jobs being swapped and the index of the machines where the move was applied, i.e., $P_{s}\left(M_{i}: M_{i,} J_{j k}: J_{j^{\prime} k}\right)$ is stored in the TL. $P_{S}\left(M_{i}: M_{i,} J_{j k}: J_{j \prime k \prime}\right)$ indicates that jobs $J_{j k}$ and $J_{j^{\prime} k^{\prime}}$ cannot be swapped on machine $i$ and $i$, until this move is removed from the TL. The above example shows that the move was applied to two machines out of the entire machine set. However, this may vary as the move can be applied to any subset of machines. In the case of permutation sequence, the moves are always made on the entire set of machines. Hence, the above move would be represented as $P_{s}\left(M_{i}(\forall i \in M) \mid J_{j k}: J_{j k^{\prime} \prime}\right)$.

If a job $J_{j k}$ is inserted from its current position $p$ into another position $p^{\prime}\left(p \neq p^{\prime}\right)$ on machines $i$ and $i$ ', the TL stores the index of the job being inserted along with its position before the move was applied and the index of the machines where the move was applied, i.e., move $P_{I}\left(M_{i}: M_{i^{\prime}} \mid J_{j k}, p\right)$ is stored in the TL. $P_{I}\left(M_{i}: M_{i^{\prime}} \mid J_{j k}, p\right)$ indicates that the job $J_{j k}$ cannot be inserted into position $p$ on machines $i$ and $i$ ' until this move is removed from the TL. As in the case of swap move, the insert move can also be applied to any subset of machines.

One of the characteristics of the problem in this research is the possibility of multiple solutions with the same objective function value. The reasons for this issue are non-continuous production and multiple jobs (belonging to the same product) having the same run times and setup times. As shown in section 4.4, due to non-continuous production, the job sequence on predecessor machines can be changed without affecting the operation times of the successor machines. This results in multiple sequences having the same job completion times in the final machines. Thus, the tardiness of these sequences remains unchanged. In addition, the problem consists of multiple jobs belonging to the same product. Since, the run time and setup time are associated with the product, these jobs share the same run times and setup time as their parent product. Changing the position of these jobs does not result in any change in the total setup time of the sequences. Due to the combination of these two characteristics of the problem, multiple solutions with the same objective function value exist. This creates an issue with the application of tabu list in the algorithm.

TL prevents the algorithm from selecting a solution which was previously explored. However, it does not prevent the algorithm from selecting a different solution with the same objective function value (OFV). Hence, once a local optimum is reached, i.e., a superior solution cannot be found in the neighborhood, the algorithm keeps selecting other solutions with the same OFV until it finally terminates. Figure 10 shows the OFV of the solutions in the CL illustrating this situation. The performance of the algorithm in this instance is similar to that of a hill climbing heuristic. Therefore, to improve the performance of the algorithm, a restriction is added to limit the number of solutions with
the same OFV into the CL. For example, if the maximum number of similar solutions (MNSS) is equal to 2 , then no more than 2 solutions with the same OFV can be entered into the CL. This additional restriction forces the algorithm to settle for an inferior solution, thus preventing the search from being trapped in the local optima. Table 4 shows the OFV of the solutions in the CL using a TS algorithm incorporating this restriction. The algorithm without MNSS restriction finds the best solution at entry \#7 and keeps selecting other solutions with the same OFV until it is terminated. The algorithm with the MNSS restriction, however, settles for an inferior solution at entry \#8 and finds a better solution at entry \#9. It can be observed that the algorithm performance improves with the added restriction.

Table 8. OFV of solutions on the CL for algorithms and without MNSS restriction

| CL Entry \# | Algorithm without MNSS restriction | Algorithm with MNSS =2 |
| :---: | :---: | :---: |
| 1 | 0.39399 | 0.39399 |
| 2 | 0.29283 | 0.29283 |
| 3 | 0.23627 | 0.23627 |
| 4 | 0.22065 | 0.22065 |
| 5 | 0.20761 | 0.20761 |
| 6 | 0.20727 | 0.20727 |
| 7 | 0.20694 | 0.20694 |
| 8 | 0.20694 | 0.20694 |
| 9 | 0.20694 | 0.20700 |
| 10 | 0.20694 | 0.18455 |
| 11 | 0.20694 | 0.18455 |
| 12 | 0.20694 | 0.18477 |
| 13 | 0.20694 | 0.18477 |
| 14 | 0.20694 | 0.18488 |
| 15 | 0.20694 | 0.18488 |
| 16 | 0.20694 | 0.18509 |
| 17 | 0.20694 | 0.18487 |
| 18 |  | 0.18487 |
| 19 |  | 0.18494 |
| 20 | $\mathbf{}$ | 0.18494 |
| Best | $\mathbf{0 . 2 0 6 9 4}$ | $\mathbf{0 . 1 8 4 5 5}$ |

### 5.1.5. Aspiration criterion

Aspiration level (AL) is the OFV of the best solution found so far by the algorithm. Aspiration criterion is a condition that a solution must satisfy in order for it to be released from tabu restriction. If the solution has a better OFV than the AL, then this solution can be selected for next iteration even if the move is tabu, i.e. tabu restriction can be overridden by aspiration criterion.
5.1.6. Steps of the Proposed TS Algorithm

The flowchart for the proposed TS algorithm is shown in Figure 9. The pseudo code for the proposed TS algorithm developed in this research is as follows:

Step 1: Set the value of iteration without improvement (IWOI) to zero
Step 2: Identify an IS, and insert it into the candidate list (CL) and index list (IL)
Step 3: Set aspiration level (AL) = objective function value (OFV) of the IS
Step 4: WHILE (IWOI < Max_IWOI and IL_Size < Max_IL_Size)
Step 4.1: Consider the latest entry into the CL as the seed
Step 4.2: Reset temporary candidate list (TCL). Generate neighborhood solutions (NS) from the seed and find their OFV. For short-term TS, PTB2 is used (strong TS)

Step 4.3: Record the OFV in the TCL
Step 4.4: Select the solution with the best OFV in the TCL
Step 4.5: IF (the solution has already been admitted into the CL), THEN disregard the solution and find the next best solution from the TCL and repeat step 4.5

ELSE, go to step 4.8
Step 4.6: IF (the number of similar solution (NSS) admitted into the CL has exceeded the maximum number of similar solution (MNSS)), THEN disregard the solution and find the next best solution from the TCL and go to step 4.5 ELSE, go to step 4.7
Step 4.7: IF (the current move associated with the selected solution is tabu), THEN go to step 4.8,

ELSE update the tabu list (TL) with the current move and go to step 4.9
Step 4.8: IF (the OFV of the selected solution is worse than AL), THEN disregard this solution and select the next best solution from TCL and go to step 4.5 ELSE, go to step 4.9
Step 4.9: Insert the solution into the CL
Step 4.10: IF (the OFV of the latest entry into the CL is better than its parent), THEN assign a star to the solution, update the value of AL and reset the value of

IWOI to zero
ELSE, increase the value of IWOI by one
IF (the parent solution already has a star), THEN assign another star to the parent solution and insert it into the IL
Step 4.10: IF (the number of entries into the TL is more than the tabu list size (TLS)), THEN remove the earliest entry from the TL

Step 5: Return the best solution in the IL


Figure 9. TS flowchart

### 5.1.7. Application of the TS Algorithm to an Example Problem

The application of TS algorithm is illustrated by means of a randomly generated example problem, as shown in Tables 9 and 10.

Table 9. Example problem

| Machine availability time ( $a_{i}$ ) |  |  | 5 | 168 | 6 | 247 | $w_{j k}$ | $d_{j k}$ |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Equipment restart time ( $E_{i}$ ) |  |  | 20 | 36 | 46 | 35 |  |  |
| Product(j) | Job (k) | $r_{j k}$ | $M_{1}$ | $M_{2}$ | $M_{3}$ | $M_{4}$ |  |  |
|  |  |  | Run time ( $R_{i j}$ ) |  |  |  |  |  |
| 1 | 1 | 0 | 32 | 29 | 60 | 84 | 1 | 1649 |
|  | 2 | 10 |  |  |  |  | 3 | 255 |
|  | 3 | 76 |  |  |  |  | 1 | 1857 |
| 2 | 1 | 1 | 40 | 46 | 59 | 82 | 1 | 1456 |
|  | 2 | 95 |  |  |  |  | 2 | 1840 |
|  | 3 | 5 |  |  |  |  | 2 | 1441 |
|  | 4 | 12 |  |  |  |  | 2 | 478 |
|  | 5 | 1 |  |  |  |  | 2 | 110 |
| 3 | 1 | 7 | 52 | 55 | 52 | 72 | 2 | 1918 |
|  | 2 | 8 |  |  |  |  | 3 | 173 |

Table 10a. Setup time for $\mathrm{M}_{1}$

|  |  | Subsequent <br> product |  |  |
| :---: | :---: | :---: | :---: | :---: |
|  | Product | $P_{1}$ | $P_{2}$ | $P_{3}$ |
| Preceding <br> product | Ref | 11 | 8 | 13 |
|  | $P_{1}$ | 5 | 8 | 13 |
|  | $P_{2}$ | 9 | 5 | 12 |
|  | $P_{3}$ | 14 | 8 | 4 |

Table 10b. Setup time for $M_{2}$

|  |  | Subsequent <br> product |  |  |
| :---: | :---: | :---: | :---: | :---: |
|  | Product | $P_{1}$ | $P_{2}$ | $P_{3}$ |
| Preceding <br> product | Ref | 14 | 9 | 11 |
|  | $P_{1}$ | 5 | 15 | 11 |
|  | $P_{2}$ | 12 | 4 | 12 |
|  | $P_{3}$ | 8 | 12 | 4 |

Table 10c. Setup time for $M_{3}$

|  |  | Subsequent <br> product |  |  |
| :---: | :---: | :---: | :---: | :---: |
|  | Product | $P_{1}$ | $P_{2}$ | $P_{3}$ |
| Preceding <br> product | Ref | 13 | 8 | 9 |
|  | $P_{1}$ | 4 | 14 | 13 |
|  | $P_{2}$ | 12 | 3 | 11 |
|  | $P_{3}$ | 10 | 12 | 3 |

Table 10d. Setup time for $M_{4}$

|  |  | Subsequent <br> product |  |  |
| :---: | :---: | :---: | :---: | :---: |
|  | Product | $P_{1}$ | $P_{2}$ | $P_{3}$ |
| Preceding <br> product | Ref | 9 | 9 | 11 |
|  | $P_{1}$ | 4 | 12 | 10 |
|  | $P_{2}$ | 13 | 3 | 13 |
|  | $P_{3}$ | 15 | 12 | 4 |

The problem consists of 3 products and the total number of jobs is 10 . The problem is for a 4-machine assembly flow shop (as shown in Figure 2) which operates on a single shift per day, i.e., available production time each day is 480 min . All storages ( $S_{1}, S_{2}$, and $S_{3}$ ) are capable of storing a maximum of 5 jobs each. A minimum storage time of 120 min . is required in both $S_{1}$ and $S_{3}$, whereas no storage time restriction is applicable to $S_{2}$. The producer's and customers' weights are 0.4 and 0.6 , respectively. The first step is to calculate the extreme values of each criteria.

- $S T_{\text {min }}$ is obtained by using SST rule. This rule is applied to each machine independently and the setup time thus obtained is calculated. For machine $1, P_{2}$ has the smallest setup time of 8 min . Out of all jobs belonging to $P_{2}, J_{21}$ has the lowest index. Hence, it is positioned first. For the next position, the job which has the least setup time with $J_{21}$ (belonging to $P_{2}$ ) as the preceding job is searched. In this case, the setup time is minimum with the jobs belonging to the same product $P_{2}$ ( 5 min .). Hence, $J_{22}$ is positioned second, followed by $J_{23}, J_{24}$ and $J_{25}$, respectively. Now, all jobs belonging to $P_{2}$ are sequenced. The next job with the least setup time must belong to $P_{1}$ since the setup time between $P_{2}$ and $P_{1}$ is 9 min . and the setup time between $P_{2}$ and $P_{3}$ is 12 min . This process is repeated until all jobs are sequenced on $M_{1}$. The sequence is obtained similarly for $M_{2}, M_{3}$ and $M_{4}$. The setup time thus obtained for each machine is added to get $S T_{\text {min }}$. Table 11 shows the sequence and setup time for each machine. The value of $S T_{\text {min }}$ is 233 min .

Table 111. Setup time for sequence generated using SST

| Machine | Sequence | Setup time |  |  |
| :---: | :---: | :---: | :---: | :---: |
| $M_{1}$ | $J_{21}-J_{22}-J_{23}-J_{24}-J_{25}-J_{11}-J_{12}-J_{13}-J_{31}-J_{32}$ | 64 |  |  |
| $M_{2}$ | $J_{21}-J_{22}-J_{23}-J_{24}-J_{25}-J_{11}-J_{12}-J_{13}-J_{31}-J_{32}$ | 62 |  |  |
| $M_{3}$ | $J_{21}-J_{22}-J_{23}-J_{24}-J_{25}-J_{31}-J_{32}-J_{21}-J_{22}-J_{23}$ | 52 |  |  |
| $M_{4}$ | $J_{11}-J_{12}-J_{13}-J_{31}-J_{32}-J_{21}-J_{22}-J_{23}-J_{24}-J_{25}$ | 55 |  |  |
| Total |  |  |  | $\mathbf{2 3 3}$ |

- The methodology to obtain $S T_{\max }$ is similar to $S T_{\min }$. However, in this case, priority is given to the job with maximum setup time. Table 12 shows the
sequence and the setup time obtained using LST rule. The value of $S T_{\max }$ is 466 min.

Table 12. Setup time for sequence generated using LST

| Machine | Sequence | Setup time |  |  |
| :---: | :---: | :---: | :---: | :---: |
| $M_{1}$ | $J_{31}-J_{11}-J_{32}-J_{12}-J_{21}-J_{13}-J_{22}-J_{23}-J_{24}-J_{25}$ | 94 |  |  |
| $M_{2}$ | $J_{11}-J_{21}-J_{12}-J_{22}-J_{13}-J_{23}-J_{31}-J_{24}-J_{32}-J_{25}$ | 131 |  |  |
| $M_{3}$ | $J_{11}-J_{21}-J_{12}-J_{22}-J_{13}-J_{23}-J_{31}-J_{24}-J_{32}-J_{25}$ | 125 |  |  |
| $M_{4}$ | $J_{31}-J_{11}-J_{21}-J_{12}-J_{22}-J_{13}-J_{23}-J_{32}-J_{24}-J_{25}$ | 116 |  |  |
| Total |  |  |  | $\mathbf{4 6 6}$ |

- The ideal value of weighted tardiness $\left(W T_{\text {min }}\right)$ is assumed to be zero.
- $W T_{\max }$ is obtained by applying WLDD rule. The due date to weight ratio for each job is shown in Table 13. Ordering the job in the decreasing order of due date to weight ratio gives a sequence of $J_{13^{-}} J_{11^{-}} J_{21^{-}} J_{31^{-}} J_{22^{-}} J_{23-} J_{24^{-}} J_{12^{-}} J_{32^{-}}$ $J_{25}$. The weighted tardiness of this sequence is 27047 min. The method used to obtain weighted tardiness of a sequence is similar to the method used to evaluate the objective function of a sequence, as explained in section 5.3.

When generating a sequence to obtain extreme values, constraints such as machine availability time, job release time, precedence constraints, and storage constraints are not considered. The reason for doing so is to identify a sequence representing the best and worst values of the criteria without the limitations placed by these constraints. However,

Table 133. Due date to weight ratio

| Job <br> $\left(J_{j k}\right)$ | $w_{j k}$ | $d_{j k}$ | $d_{j k} / w_{j k}$ |
| :---: | :---: | :---: | :---: |
| $J_{11}$ | 1 | 1649 | 1649.0 |
| $J_{12}$ | 3 | 255 | 85.0 |
| $J_{13}$ | 1 | 1857 | 1857.0 |
| $J_{21}$ | 1 | 1456 | 1456.0 |
| $J_{22}$ | 2 | 1840 | 920.0 |
| $J_{23}$ | 2 | 1441 | 720.5 |
| $J_{24}$ | 2 | 478 | 239.0 |
| $J_{25}$ | 2 | 110 | 55.0 |
| $J_{31}$ | 2 | 1918 | 959.0 |
| $J_{32}$ | 3 | 173 | 57.7 |

during evaluation of the identified sequences, the constraints are considered. For example, the sequence for $W T_{\max }$ is obtained by sequencing the jobs in the decreasing order of the due date to weight ratio. To generate this sequence, only job weight and due dates are considered. However, the above constraints are considered during the evaluation of this sequence to obtain the extreme values.

After the extreme values are identified, the initial solution is generated using the normalized positional values (NPV) of PS and CS. Two sequences are generated by applying this method to $M_{1}$ and $M_{3}$, since these machines are the first machine required by components 1 and 2, respectively. First $M_{1}$ is selected. All jobs are entered into the set of non-scheduled jobs (NSJ). The availability time of $M_{1}$ is 5 min . and the equipment restart time is 20 min . Hence, the actual machine availability time $\left(a_{1}\right)$ is 25 min . At $t=25$, all jobs except $J_{22}$ and $J_{33}$ are released. Hence, the set SJ contains eight jobs. Now, the PS and CS are generated for the set of jobs in SJ. The method for generating PS is similar to the method used to generate the sequence for $S T_{\text {min }}$. CS is obtained by applying WEDD rule, i.e., priority is given to jobs with the least due date to weight ratio. The rank of each job (in SJ) in PS and CS is shown in Table 14. The NPV of each job is calculated. For $J_{11}$, it is given by $0.4 * 5+0.6 * 8=6.8$. The NPV for other jobs in SJ is also shown in Table 13. Since $J_{25}$ has the smallest NPV, it is scheduled first. The setup time required for $J_{25}$ is 8 (reference setup). The completion time of $J_{25}$ on $M_{1}$ is given by $25+8+40=73 \mathrm{~min}$. The machine availability time $a_{1}$ is updated to 73 min . and $J_{25}$ is removed from the set NSJ. Now, nine jobs remain in NSJ. At the next iteration, all jobs except $J_{22}$ and $J_{33}$ are released before $t=73 \mathrm{~min}$. These released jobs are entered into the set SJ. The NPV of each job is calculated by the same method as described above and the job with the least NPV (in this case, $J_{24}$ ) is scheduled after $J_{25}$. The completion time of $J_{24}$ is $73+5+40=118 \mathrm{~min}$. Again, $a_{1}$ is set to 118 min . and $J_{24}$ is removed from the set NSJ. Now all jobs in NSJ are released before $t=118 \mathrm{~min}$. Hence, the set SJ includes 8 jobs (10 total jobs -2 scheduled jobs). This process is repeated until all the jobs are scheduled. The sequence thus obtained is $J_{25}{ }^{-}$ $J_{24}-J_{23}-J_{22}-J_{12}-J_{21}-J_{11}-J_{32}-J_{31}-J_{13}$. The jobs scheduled at each iteration along with the machine availability time and completion time is shown in Table 15. At itrn\#9, the completion time of the job $J_{31}$ is 459 min . Only 21 min . is available for production before
the shift ends at 480 min . Since the last job $\left(J_{13}\right)$ cannot be completed within the remaining time, its production is shifted for the next day. Thus, the available time for next iteration is 1460 min., i.e., start of the next shift (1440) + equipment restart time (20). Next, another sequence is obtained by applying the IS generation mechanism on $M_{3}$. The sequence obtained is $J_{25}-J_{32}-J_{24}-J_{23}-J_{22}-J_{31}-J_{21}-J_{12}-J_{11}-J_{13}$.

The two sequences are now evaluated in terms of their OFV. To obtain the OFV, the total setup time and weighted tardiness values of each sequence needs to be calculated. The total setup time can be calculated easily from the setup time information in Table 10a,

Table 14. Rank of jobs in PS and CS

| Job <br> $\left(J_{j k}\right)$ | Rank |  | NPV |
| :---: | :---: | :---: | :---: |
|  | PS | CS |  |
| $J_{11}$ | 5 | 8 | 6.8 |
| $J_{12}$ | 6 | 3 | 4.2 |
| $J_{21}$ | 1 | n | 4.6 |
| $J_{23}$ | 2 | 5 | 3.8 |
| $J_{24}$ | 3 | 4 | 3.6 |
| $J_{25}$ | 4 | 1 | 2.2 |
| $J_{31}$ | 7 | 6 | 6.4 |
| $J_{32}$ | 8 | 2 | 4.4 |

Table 15. Job scheduled at each iteration of IS generation mechanism on $M_{1}$

| Itrn\# | Job <br> $\left(\boldsymbol{J}_{\boldsymbol{j} \boldsymbol{k}}\right)$ | Availabilty <br> time | Completion <br> time |
| :---: | :---: | :---: | :---: |
| 1 | $\boldsymbol{J}_{25}$ | 5 | 73 |
| 2 | $J_{24}$ | 73 | 118 |
| 3 | $J_{23}$ | 118 | 163 |
| 4 | $J_{22}$ | 163 | 208 |
| 5 | $J_{12}$ | 208 | 249 |
| 6 | $J_{21}$ | 249 | 297 |
| 7 | $J_{11}$ | 297 | 338 |
| 8 | $J_{32}$ | 338 | 403 |
| 9 | $J_{31}$ | 403 | 459 |
| 10 | $J_{13}$ | 1460 | 1492 |

$10 \mathrm{~b}, 10 \mathrm{c}$ and 10 d . Consider the sequence obtained above from $M_{1}\left(I S_{1}\right): J_{25}-J_{24}-J_{23}-J_{22}-J_{12}-$ $J_{21}-J_{11}-J_{32}-J_{31}-J_{13}$. The setup time for this sequence in $M_{1}$ is $8+5+5+5+9+8+9+13$
$+4+14=80 \mathrm{~min}$. Similarly, for $M_{2}, M_{3}$ and $M_{4}$, the setup times are 83,81 and 85 , respectively. The total setup time is equal to $80+83+81+85=329$. The weighted tardiness is obtained by calculating the completion time of the job at the final assembly machine. The steps are shown in Figure 10.

Initially, set counter $c_{1}, c_{2}, c_{3}$ and $c_{4}$ equal to 1 for $M_{1,} M_{2}, M_{3}$ and $M_{4}$, respectively. The first job to be processed is $J_{25}$, which is released at $t=1$. Since, this is the first job, the storage space after $M_{1}\left(S_{1}\right)$ is totally empty. The availability time of $M_{1}$ is 5 min . and the equipment restart time is 20 min . Thus, the completion time of $J_{25}$ is given by $\operatorname{Max}(1,25+8)+40=73 \mathrm{~min}$. The new availability time of $M_{1}\left(a_{1}\right)=73$ and $c_{1}=c_{1}+1$. Next, $M_{3}$ is selected. The availability time of $M_{3}$ is 6 min . and the restart time is 46 min . The setup time and run time associated with $J_{25}$ on $M_{2}$ is 8 (reference setup) and 59 , respectively. The completion time of $J_{25}$ is given by $\operatorname{Max}(1,(6+46)+8)+59$, which equals to 119 min . The counter and availability time of $M_{3}$ is updated. For $M_{2}, J_{25}$ is released at $t=191$, i.e., 73 (completion time on $M_{1}$ ) +120 (minimum storage time at $S_{1}$ ). The availability time of $M_{2}$ is 168 min . and the equipment restart time is 36 min . The setup time and run time associated with $J_{25}$ on $M_{2}$ is 9 (reference setup) and 46, respectively. Thus, the completion time is given by $\operatorname{Max}(191,(168+36)+9)+46$, which is equal to 259 min . $a_{2}$ is updated to 259 and $c_{2}=c_{2}+1$. The release time of a job $\left(J_{j k}\right)$ on $M_{4}$ is given by $\operatorname{Max}(259+0,119+120)$, which is equal to 259 . The completion time is equal to $\operatorname{Max}(259,(247+35)+9)+82)=373 \mathrm{~min}$. The counter and availability time of $M_{4}$ are updated. This process is continued until all of the counter values equal $N$. If the machine is blocked due to storage space unavailability or a job not completed on previous machine, then the iteration on that machine is skipped and the next machine is selected. The completion time calculated for $I S_{1}$ is shown in Table 16.


Figure 10. Evaluation of job completion times

Table 16. Job completion times on machine

| $\begin{aligned} & \hline \mathbf{J o b} \\ & \left(\boldsymbol{J}_{j k}\right) \end{aligned}$ | Completion time on Machine |  |  |  | $w_{j k}$ | $d_{j k}$ | Weighted Tardiness |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | $M_{1}$ | $M_{2}$ | $M_{3}$ | $M_{4}$ |  |  |  |
| $J_{25}$ | 73 | 259 | 119 | 373 | 2 | 110 | 526 |
| $J_{24}$ | 118 | 309 | 181 | 458 | 2 | 478 | 0 |
| $J_{23}$ | 163 | 359 | 243 | 1557 | 2 | 1441 | 232 |
| $J_{22}$ | 208 | 409 | 305 | 1642 | 2 | 1840 | 0 |
| $J_{12}$ | 249 | 450 | 377 | 1739 | 3 | 255 | 4452 |
| $J_{21}$ | 299 | 1522 | 450 | 1833 | 1 | 1456 | 377 |
| $J_{11}$ | 341 | 1563 | 1546 | 2999 | 1 | 1649 | 1350 |
| $J_{32}$ | 411 | 1629 | 1611 | 3081 | 3 | 173 | 8724 |
| $J_{31}$ | 467 | 1697 | 1694 | 3157 | 2 | 1918 | 2478 |
| $J_{13}$ | 1492 | 1768 | 1799 | 3256 | 1 | 1857 | 1399 |

It can be seen that the job completion times on $M_{1}$ in Table 15 and Table 16 are different (starting at the $6^{\text {th }}$ job position), despite having the same sequence. This is because of the storage space limitations considered in the later evaluation. The data in Table 14 refers to the completion times calculated during initial sequence generation on $M_{1}$. During the application of this method, $M_{1}$ is evaluated independently and hence, storage space limitations are not considered. Table 15 consists of the completion times calculated during the evaluation of objective function. In this method, the storage space limitations must be considered as all machines are evaluated. The fifth job on $M_{1}\left(J_{12}\right)$ is completed at 249 min . Without the storage space constraint, the next job $\left(J_{21}\right)$ can be started at $t=257$, i.e., $249+$ 8 (setup time between $P_{1}$ and $P_{2}$ ). However, job $\left(J_{21}\right)$ cannot start at $t=257 . J_{21}$ is the sixth job in the sequence, which means that the previously completed five jobs still occupy the storage space, unless some of these jobs are completed on $M_{2}$. The storage space constraint specifies that a job can only be started after the storage space becomes available, i.e. the job is completed on successor machine. The first job $J_{25}$ is completed at $t=259$ on $M_{2}$ which empties one storage space and thus job $\left(J_{21}\right)$ cannot be started before $t=259$. However, anticipatory setup can be performed during this wait time so that the job can be processed as soon as the storage is available. Thus, the completion time of the job $\left(J_{21}\right)$ is delayed by 2 min . in Table 16. After the completion times are calculated, the weighted tardiness of each job is calculated using equation 5.5. The total weighted tardiness for $I S_{1}$ is shown in Table 16.

$$
\begin{equation*}
T_{j k}=\operatorname{Max}\left(0, w_{j k}\left(T f f_{4 j k}-d_{j k}\right)\right) \tag{5.5}
\end{equation*}
$$

The OFV is then computed using equation 4.3. For $I S_{1}$, the OFV is 0.59823 . The OFV for $I S_{3}$ is also computed similarly, which is given by 0.50769 . Since, the OFV for $I S_{3}$ is lower, it is selected as the IS to trigger the tabu search algorithm.

The IS is entered into the CL and IL. The AL is set to the OFV of the IS, which is equal to 0.50769 . The latest entry into the CL, which in this case is the initial solution, is selected as the seed. A set of perturbations, comprising of swap and insert moves, are performed on the seed to generate NS. In this example problem, an NPN sequence is considered and PTB2 is used to generate NS, i.e., strong TS. For a problem with 10 jobs, a total of 1755 solutions can be generated, 675 from swap moves and 1080 from insert moves. Table 17 shows a portion of the perturbations performed on the IS and the associated objective function values. Note that the size of the neighborhood solution set, for this problem, is fifteen times lower for PN sequence as compared to NPN sequence, provided that the same perturbation type is used for both sequences. In the table below, the solutions corresponding to perturbations 66-71 are infeasible because the OFV of these solutions are greater than one, i.e. a penalty has been incurred due to infeasibility. In this case, the solution with the minimum OFV is obtained from perturbation 1246, i.e. insert the job $J_{31}$ into the $3^{\text {rd }}$ position of the sequence for all machines. The configuration of jobs for this solution is shown below:

Solution selected at first iteration $=\left\{\begin{array}{l}M_{1}: J_{25}, J_{32}, J_{31}, J_{24}, J_{23}, J_{22}, J_{21}, J_{12}, J_{11}, J_{13} \\ M_{2}: J_{25}, J_{32}, J_{31}, J_{24}, J_{23}, J_{22}, J_{21}, J_{12}, J_{11}, J_{13} \\ M_{3}: J_{25}, J_{32}, J_{31}, J_{24}, J_{23}, J_{22}, J_{21}, J_{12}, J_{11}, J_{13} \\ M_{4}: J_{25}, J_{32}, J_{31}, J_{24}, J_{23}, J_{22}, J_{21}, J_{12}, J_{11}, J_{13}\end{array}\right\}$
This solution is selected as the seed for the next iteration and is inserted into the CL. Since the OFV of the new seed is lower than the parent (IS), it has the potential of being a local optimum. Hence, a star is assigned to this CL entry. The move corresponding to this solution is inserted into the tabu list. The tabu list size in this case is five. Therefore, this move remains tabu for the next five iterations. In addition, the OFV of the new seed is lower than the AL ( $0.408479<0.50769)$. Thus, the AL is updated to 0.408479 .

Table 17. NS generation in the first iteration

| Perturbation \# | Move Type | Move | OFV |
| :---: | :---: | :---: | :---: |
| 1 | Swap | M1, M2, M3, M4: J25 \& J32 | 0.465827 |
| 2 |  | M1, M2, M3: J25 \& J32 | 0.598243 |
| 3 |  | M1, M2, M4: J25 \& J32 | 0.478842 |
| 4 |  | M1, M3, M4: J25 \& J32 | 0.572315 |
| 5 |  | M2, M3, M4: J25 \& J32 | 0.471368 |
| 6 |  | M1, M2: J25 \& J32 | 0.61026 |
| 7 |  | M1, M3: J25 \& J32 | 0.493677 |
| 8 |  | M1, M4: J25 \& J32 | 0.584332 |
| 9 |  | M2, M3: J25 \& J32 | 0.60354 |
| 10 |  | M2, M4: J25 \& J32 | 0.483385 |
| 11 |  | M3, M4: J25 \& J32 | 0.572754 |
| 12 |  | M1: J25 \& J32 | 0.505694 |
| 13 |  | M2: J25 \& J32 | 0.615557 |
| 14 |  | M3: J25 \& J32 | 0.49711 |
| 15 |  | M4: J25 \& J32 | 0.584771 |
| 16 |  | M1, M2, M3, M4: J25 \& J24 | 0.512344 |
| 17 |  | M1, M2, M3: J25 \& J24 | 0.626478 |
| 18 |  | M1, M2, M4: J25 \& J24 | 0.627011 |
| 19 |  | M1, M3, M4: J25 \& J24 | 0.751173 |
| 20 |  | M2, M3, M4: J25 \& J24 | 0.627011 |
| $\vdots$ |  | : | $\vdots$ |
| 66 |  | M1, M2: J25 \& J31 | 2.25 |
| 67 |  | M1, M3: J25 \& J31 | 3.5 |
| 68 |  | M1, M4: J25 \& J31 | 4.75 |
| 69 |  | M2, M3: J25 \& J31 | 4.75 |
| 70 |  | M2, M4: J25 \& J31 | 3.5 |
| 71 |  | M3, M4: J25 \& J31 | 2.25 |
| ! |  | $\vdots$ | $\vdots$ |
| 672 |  | M1: J11 \& J13 | 0.507685 |
| 673 |  | M2: J11 \& J13 | 0.507685 |
| 674 |  | M3: J11 \& J13 | 0.507685 |
| 675 |  | M4: J11 \& J13 | 0.507685 |
| 676 | Insert | M1, M2, M3, M4: J25 into the 3rd position | 0.46667 |
| 677 |  | M1, M2, M3: J25 into the 3rd position | 0.600461 |
| 678 |  | M1, M2, M4: J25 into the 3rd position | 0.479286 |
| 679 |  | M1, M3, M4: J25 into the 3rd position | 0.718062 |
| 680 |  | M2, M3, M4: J25 into the 3rd position | 0.471368 |
| 681 |  | M1, M2: J25 into the 3rd position | 0.612478 |
| 682 |  | M1, M3: J25 into the 3rd position | 0.610229 |



In the next iteration, another set of NS is generated using the new seed and the best solution is selected. If the selected solution violates the tabu restriction, then this solution is discarded, and the next best solution is selected. In addition to the tabu restriction, the selected solution can also be discarded if the solution is already in the CL or the number of similar solutions (solution with the same OFV) exceeds the maximum limit. Table 18 shows all the entries into the CL with their status and OFV. The solution selected at this iteration has an OFV of 0.365964 , which is lower than both the parent seed and the AL. Hence, a star is assigned to this CL entry and the AL is updated. The solution selected at the third iteration is also assigned a star and the AL is updated to 0.340253 . At the fourth iteration, the selected solution is not better than its parent. Hence, another star is assigned
to the parent seed, i.e. this solution is a local optimum and is thus inserted into the IL. The IWOI is increased by one at this iteration. The process continues until a stopping criteria is met. In this case, the algorithm stops after the IL size reaches 5 (including the IS). It can be observed from Table 18 that several solutions have the same OFV (solution 3 and 4, solution 5 and 6 , solution 7 and 8 and others). The limit set on the maximum number of similar solutions ( 2 , in this example) allowed in the CL prevents the algorithm from selecting more than two solutions with the same OFV. This prevents the algorithm from repeatedly selecting similar solutions at each iteration until the maximum iteration limit is met.

Table 18. Entries into the CL

| Entry \# | OFV | Entry \# | OFV |
| :--- | :--- | :--- | :--- |
| 0 (IS) | 0.507685 | 15 | 0.342804 |
| 1 | $0.408479^{*}$ | 16 | 0.342937 |
| 2 | $0.365964^{*}$ | 17 | 0.342937 |
| 3 | $0.340253^{* *}$ | 18 | 0.343958 |
| 4 | 0.340253 | 19 | 0.343958 |
| 5 | 0.341517 | 20 | $0.342560^{*}$ |
| 6 | 0.341517 | 21 | $0.323676^{*}$ |
| 7 | 0.341651 | 22 | $0.321369^{* *}$ |
| 8 | 0.341651 | 23 | 0.321369 |
| 9 | 0.341717 | 24 | 0.322655 |
| 10 | 0.341717 | 25 | 0.322655 |
| 11 | 0.343004 | 26 | 0.323055 |
| 12 | $0.341540^{* *}$ | 27 | $0.321768^{* *}$ |
| 13 | 0.341540 | 28 | 0.321768 |
| 14 | 0.342804 |  |  |

### 5.2. Tabu Search/Path Relinking

A hybridization of TS algorithm with another algorithm generally leads to an improved performance of the algorithm. Al-Anzi and Allahverdi (2006) proposed three heuristics, based on basic TS, simulated annealing and hybrid TS, for a two-stage assembly flow shop and showed that the hybrid TS algorithm outperforms the other two algorithms. Gagné et al., 2005 presented a hybrid tabu search/variable neighborhood search algorithm for a multi-objective scheduling problem and showed that the hybrid metaheuristic is both effective and efficient in finding good solution. A short-term TS might not yield good quality solutions because of its inability to utilize information on good quality solutions. Hence, TS has often been used with long-term memory function to increase the efficacy of the algorithm. TS with path relinking (PR) serves the same purpose, but adds a stochastic component to the search algorithm, in contrast to the deterministic approach used by longterm memory function.

PR was originally proposed by Glover (1986) as an intensification and diversification strategy of exploring the path connecting elite solutions obtained from TS. PR is intimately related to TS and derives additional advantages by utilizing the memory structure that can adapt to various combinatorial optimization problems. PR is generally embedded with a local search algorithm (TS, in this case), which is used to explore the search space created by generating a path between a given set of elite solutions. Zeng et al. (2013) investigated different ways to integrate PR techniques into a hypervolume-based multi-objective local search algorithm to solve a bi-criteria flowshop problem. Shahvari and Logendran (2016) proposed an algorithm based on tabu search/path relinking (TS/PR) to solve a bi-criteria batching and scheduling problem in a hybrid flowshop. Peng et al. (2015) demonstrated the efficacy of TS/PR algorithm, both in terms of solution quality and computational time, for a job shop scheduling problem.

In this research, a path relinking heuristic is incorporated into the TS-based heuristic to enhance the efficacy of the algorithm. In the algorithm, TS and PR work in tandem with each other, where PR generates a trajectory or path between two elite solutions and TS explores the search space from the path solutions. PR mainly integrates two key
components to ensure search efficiency: 1) the construction approach used to generate path between solutions, and 2) the method used to choose the reference solution $\left(S^{R}\right)$. TS/PR starts with the initial population (IP) which consists of a set of high quality solutions. At the start of the algorithm, the IP forms the population set $(P)$. At each iteration, two solutions are randomly selected from the set $P$ as initial solution ( $S^{I}$ ) and guiding solution $\left(S^{G}\right)$. The solution that begins the path is called $S^{I}$ and the solution that the path leads to is called $S^{G}$. All intermediate solutions generated during path formation is stored in InitialPathSet. A set of high quality solutions is selected from the initial path set to form PromisingPathSet. The solutions in PromisingPathSet is then evaluated to select $S^{R}$, which is used to update the set $P$. The iteration continues until some stopping criteria is met.

### 5.2.1. Initial Population

The IP is generated using a method similar to the one proposed by Shahvari and Logendran (2017) in which the elite solutions obtained from the TS algorithm are used to populate the IP. First, the optimized solution from the TS is added into the IP and the rest of the solutions is added from the CL and the IL of the TS. The pseudocode for the IP generation of permutation TS/PR (ALG3) used in this research is shown in Table 19. $P_{\text {size }}$ indicates the size of the IP. If the size of the IL ( $I L_{\text {size }}$ ) is less than $P_{\text {size }}$, then all solutions in the IL, except the optimized solution which is already entered, are added to the IP and the rest of the solutions $\left(P_{\text {size }}-I L_{\text {size }}\right)$ are randomly selected from the CL.

Table 19. Pseudocode for IP generation of permutation TS/PR

```
\(S^{T S} \leftarrow\) Permutation strong TS (ALG1) //Section 5.1.6
\(I P \leftarrow S^{T S}\)
if \(\left(P_{\text {size }}\right) \leq\) ILsize
    \(I P \leftarrow \operatorname{Select}\left(P_{\text {size }}-1\right)\) random solutions from IL (except \(\left.S^{T S}\right)\)
else
    \(I P \leftarrow\) Select all solutions from IL (except \(S^{T S}\) )
    \(I P \leftarrow\) Select \(\left(P_{\text {size }}-I L_{\text {size }}\right)\) non-repeated random solutions from \(C L\)
endif
```

The IP of non-permutation TS/PR (ALG4) is populated with both PN and NPN solutions. The reason for using both PN and NPN solutions is to diversify the path generation mechanism. Half of the solutions in the IP come from IL and CL of nonpermutation strong TS (ALG2) and the rest come from the final population set of permutation TS/PR (ALG3). For example, if the $P_{\text {size }}$ is 10 , then 5 solutions come from ALG2 and the rest come from ALG3. The method of adding NPN solutions to the IP from ALG2 is similar to the method in Table 18, i.e., first the optimized solution is added, and the rest comes from IL and then CL. To add the PN solutions, the output solution from ALG3 is added first, and the rest will be added from the final population set of ALG3.

At each iteration of TS/PR, two solutions are randomly selected from the population set, which is called a PairSet. TS/PR is implemented on the selected pair in both directions ( $S^{I} \leftrightharpoons S^{G}$ ). Two new solutions are thus obtained, one from each direction, which is used to replace the two worst solutions in the set $P$. Hence, $P$ is updated at each iteration and the process continues until a maximum number of consecutive iterations without improvement (MIWI) is reached.

### 5.2.2. Path Construction

After a PairSet is randomly selected, it is checked against a TabuSet, which records all pairs of solutions previously selected in the search procedure. If the PairSet is in the TabuSet, then a new PairSet is selected. One of the solutions from the PairSet is selected as $S^{I}$ and the other one as $S^{G}$. In order to generate a path from $S^{I}$ to $S^{G}$, the distance between these solutions needs to be computed. Sevaux and Sörensen (2005) showed that there are several measures that can be used to calculate the distance between two sequences. Swapbased operator and insertion-based operator appear prominently in neighborhood search of flowshop problems (Nowicki and Smutnicki, 1996).Taillard (1990) showed that the insertion-based operator is more effective in a neighborhood search. Hence, we propose longest common subsequence-based construction (LCS-based construction), in terms of insertion operator, because it enables knowing the minimum number of moves to get to the guiding solution (Zeng et al., 2013).

In the LCS-based construction method, the jobs belonging to LCS are identified. The length of the LCS (number of jobs belonging to LCS) gives a measure of similarity shared between two sequences and its interval varies between $[1, N]$, where $N$ is the number of total jobs. The distance between two sequences ( $\underline{d}$ ) indicates the minimum number of moves required to move from $S^{I}$ to $S^{G}$ and is given by $N$ minus the length of LCS. The interval of $d$ varies between $[0, N-1]$. The LCS can be calculated by a dynamic programming algorithm in $\mathrm{O}\left(N^{2}\right)$, which is similar to a well-known Needleman-Wunsch algorithm (Schiavinotto and Stützle, 2007). The jobs which do not belong to LCS, are called candidate jobs. The LCS is computed using the following iterative procedure:

- Step 1: Obtain the smallest value of $p+q$ such that $J_{p}^{S^{I}}=J_{q}^{S^{G}}$. A tie is broken in favor of $S^{I}$.
- Step 2: Determine the minimum forward distance between $J_{p}^{S^{I}}=J_{p+1}^{S^{I}}$ in $S^{G}$ and $J_{q}^{S^{G}}=J_{q+1}^{S^{G}}$ in $S^{I}$.
- Step 3: Select the jobs corresponding to the initial and final positions on the minimum forward distance, in both $S^{I}$ and $S^{G}$, as jobs belonging to the LCS.
- Step 4: Update $p$ and $q$ by the last selected positions of LCS in $S^{I}$ and $S^{G}$.
- Step 5: Repeat steps 2-4 until $p=N$ or $q=N$.

Initial Solution

| 12 | 13 | 21 | 32 | 11 | 41 | 31 | 22 |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |

Guiding Solution

| 13 | 32 | 11 | 12 | 31 | 41 | 21 | 22 |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |

Note: Jobs are represented by index, i.e., 12 indicates the $2^{\text {nd }}$ job belonging to the $1^{\text {st }}$ product. Jobs in LCS are highlighted in green

Figure 11. LCS construction

Consider the initial and guiding solution as shown in Figure 11. In this example, the minimum value of $p+q$ is $3(p=2$ and $q=1)$, since $J_{2}^{S^{I}}=J_{1}^{S^{G}}=13$. The forward distance between $J_{2}^{S^{I}}=13$ and $J_{3}^{S^{I}}=21$ in $S^{G}$ is 5 (jobs $32,11,12,31$ and 41) and the forward distance between $J_{1}^{S^{G}}=13$ and $J_{2}^{S^{G}}=32$ in $S^{I}$ is 1 (job 21 only). Hence, the forward minimum distance is 1 and therefore, jobs 13 and 32 belongs to the LCS, both in
$S^{I}$ and $S^{G}$. The last job selected as LCS is 32 , for both $S^{I}$ and $S^{G}$. The position of 32 in $S^{I}$ and $S^{G}$ is 4 and 2 , respectively. Therefore, the value of $p$ and $q$ are updated to 4 and 2, respectively. In the next iteration, the forward distance between $J_{4}^{S^{I}}=32$ and $J_{5}^{S^{I}}=11$ in $S^{G}$ is 0 and the forward distance between $J_{2}^{S^{G}}=32$ and $J_{3}^{S^{G}}=11$ in $S^{I}$ is also 0 . Since ties are broken in favor of the forward distance in $S^{I}, J_{3}^{S^{G}}=11$ is added to the LCS. Note that in this instance, breaking the tie in favor of the forward distance in $S^{G}$ would result in the same job $J_{5}^{S^{I}}=11$ being added to the LCS. However, this might not always be the case. The values of $p$ and $q$ are now updated to 5 and 3 , respectively. The iteration is repeated until either $p$ or $q$ equals $N$. For this example, the jobs belonging to the LCS are 13, 32, 11, 41 and 22, as shown in Figure 11. The length of the LCS is 5 and the distance $d$ is equal to $3(8-5)$. The rest of the jobs, i.e., 12, 21, and 32 , belong to candidate jobs. The candidate jobs will be moved from their initial position in $S^{I}$, one move at a time, in order to reach $S^{G}$. Thus, 3 moves must be applied to move from $S^{I}$ to $S^{G}$. PR generates a new solution by applying one move at each step and thus, decreases the distance between $S^{I}$ and $S^{G}$ by 1 . Hence, a total of $d-1$ intermediate path solutions are generated during path relinking. Path solutions from $S^{I}$ to $S^{G}$ can be generated using the following steps:

- Step 1: Determine the jobs belonging to LCS and the candidate jobs using the method described above.
- Step 2: Determine all possible insertion points of all candidate jobs. In the above example, the first candidate job 12 is located between LCS jobs 11 and 41 in $S^{G}$. Hence, insertion point of job 12 must be between LCS jobs 11 and 41 in $S^{I}$, i.e., job 12 can be removed from its current position in $S^{I}$ and inserted between 11 and 41. The new sequence would then be 13-21-32-11-12-41-3122. Note that a candidate job might have more than one insertion point. For example, the second candidate job 21 is located between LCS jobs 41 and 22 in $S^{G}$. Hence, insertion point of job 21 must also be between LCS jobs 41 and 22 in $S^{I}$. Since job 41 and 22 are not adjacent in $S^{I}$, the candidate job 21 can be inserted between job 41 and 31, or job 31 and 22. This is shown in Table 20.
- Step 3: From each insertion point identified in step 2, a new solution can be generated. Evaluate each solution that is generated, in terms of their OFV, from all possible candidate moves.
- Step 4: Randomly select a solution belonging to the top $20 \%$ of the generated solutions as the current solution $\left(S^{C}\right)$, i.e., if the total number of possible candidate moves is 20 , then the current solution would be chosen randomly from the four best solutions.
- Step 5: Enter $S^{C}$ into the InitialPathSet, $S^{I} \leftarrow S^{C}$ and $d \leftarrow d-1$.
- Step 6. Go to step 1 until $d=0$.

Table 20. Possible candidate moves starting from $S^{I}$

|  | Candidate jobs in the initial solution |  |  |  |
| :--- | :---: | :---: | :---: | :---: |
|  | 12 | 21 |  | 31 |
| Insertion point | $(11,41)$ | $(41,31)$ | $(31,22)$ | $(11,41)$ |
| OFV (move) | 0.4675 | 0.4236 | 0.4528 | 0.4827 |

At each step of path generation, there are several possible candidate moves that can be selected. However, only one is chosen at each step. Therefore, based on the criteria used to select a move, we can generate the path between $S^{I}$ and $S^{G}$ in several ways (Zeng et al., 2013). Shahvari and Logendran (2017) have proposed to evaluate all possible candidate moves, in terms of the objective function value, at each step and then randomly select a solution from a set of best solutions (belonging to the global or local optima). In this research, we use the same methodology. At each step, various solutions are generated by applying all possible candidate moves. Each solution generated is evaluated, in terms of their OFV, and $20 \%$ of the best solutions are selected. Out of these solutions, a new intermediate solution is randomly selected and entered into the InitialPathSet.

Table 19 shows the possible insertion points of all candidate jobs along with the OFV of the solutions resulting from the move (refer to the example shown in Figure 2). Starting from $S^{I}$, there are only four possible moves at the first step of path generation. Hence, four possible intermediate solutions can be generated. Since, there are only four solutions, there is only one solution belonging to the top $20 \%$ (roundup $(20 \%$ of 4$)=1$ ),
i.e., the best solution. Hence, the solution with an OFV of 0.4236 is chosen as $S^{C}$ and inserted into the InitialPathSet. $S^{I}$ is replaced by $S^{C}$ and the distance is decreased by 1 . The process is repeated until $d=0$.

The example presented above explains the path construction technique for a single machine sequence, i.e., a permutation sequence. The method for a non-permutation sequence also follows a similar technique. In this case, the LCS of each machine sequence is computed separately. For example, LCS between $S^{I}$ and $S^{G}$ of M1 is computed separately from $S^{I}$ and $S^{G}$ of M2. Each machine sequence has a separate set of candidate jobs and a distance associated with its sequence, as shown in Figure 12. The sum of LCS distance of all machines equals the total distance ( $d$ ). In this example, the total distance is 12 , which means that a minimum of 12 moves must be applied to $S^{I}$ in order to reach $S^{G}$. Thus 11 intermediate solutions would be generated during path construction. Each move corresponds to the movement of one candidate job, from its current position $(p)$ to the next position ( $p^{\prime}$ ) on a single machine sequence. For example, insertion of candidate job 12 between 11 and 41 in $S^{I}$ of M1 indicates one move.



Candidate Jobs || Distance
M1: 12,21,31 || 3

M2: 11,12,32,22,31 || 5

M3: 12,32 || 2

M4: 11,12 || 2

Total Distance $=12$

Figure 12. The LCS between two solutions in non-permutation sequence

### 5.2.3. Path Solution Selection

Each of the two consecutive solutions in the InitialPathSet differ only by one insert move. Hence, it is not productive to apply improvement procedure to all solutions in the InitialPathSet, because many of these solutions would lead to the same local optimum. Several methods have been proposed to select solutions from InitialPathSet that are entered into PromisingPathSet. Peng et al. (2015) proposed a strategy based on adaptive distance-
control mechanism to obtain promising solutions. Zeng et al. (2013) selected a set of nondominating solutions from the InitialPathSet, which are entered into PromisingPathSet. In Shahvari and Logendran (2016), a set of global and local optima are selected from the InitialPathSet, which is called PromisingPathSet. This research adopts the methodology proposed by Shahvari and Logendran (2016). Figure 13 shows the set of global and local optima solutions in the InitialPathSet. In this case, solutions A, B, C, D, E, F and G would be entered into the PromisingPathSet.


Figure 13. Global and Local Optima in InitialPathSet

### 5.2.4. Reference Solution Determination

A reference solution $\left(\mathrm{S}^{\mathrm{R}}\right)$ is determined from the solutions in the PromisingPathSet, which is used to update the population set $P$. First, a slight TS is applied to optimize each solution in the PromisingPathSet to a local optimum. The solution with the best OFV is selected and further optimized using a strong TS. This optimized solution is selected as $S^{R}$. The reason that a slight TS is used initially is that it is not too time consuming but can optimize a solution to some extent that a solution more promising than others can be selected. The reference solution needs to be optimized as much as possible. Hence, a strong TS is used to optimize the selected solution.

At each iteration of the algorithm, two reference solutions are obtained from the PairSet by applying TS/PR in both directions ( $S^{I} \leftrightharpoons S^{G}$ ). These two reference solutions replace the two worst solutions in the population set $P$. Hence, at each iteration, the set $P$ gets updated. The iteration continues until the MIWI is reached. The pseudo-code for the TS/PR procedure is shown in Table 21 and the flowchart is shown in Figure 14.

Table 21. Pseudo-code for TS/PR

```
Input: Problem Data //Section 7
Output: The best schedule \(S^{\text {best }}\) found so far
\(S^{T S} \leftarrow\) Short-Term Tabu Search //Section 5.1
\(P=\left\{S^{I}, \ldots, S^{p-1}, S^{T S}\right\} \leftarrow\) Population_Initialization \(\left(S^{T S}\right) / /\) Section 5.2.1
\(S^{\text {best }}=\arg \min \left\{f\left(S^{i}\right) \mid i=1, \ldots, p\right\}\)
repeat
    Randomly select one solution pair \(\left\{S^{i}, S^{j}\right\}\) from \(P\),
    where \(S^{i} \in P, S^{j} \in P, S^{i} \neq S^{j}\) and \(\left\{S^{i}, S^{j}\right\} \notin\) TabuSet
8: \(\quad S^{p^{+1}} \leftarrow\) Path_Relinking \(\left(S^{i}, S^{j}\right)\),
    \(S^{p^{+2}} \leftarrow\) Path Relinking \(\left(S^{j}, S^{i}\right) / /\) Section 5.2.2
    if \(S^{p+1}\left(\right.\) or \(\left.S^{p+2}\right)\) is better than \(S^{\text {best }}\) then
            \(S^{\text {best }} \leftarrow S^{p+1}\left(\right.\) or \(\left.S^{p^{+2}}\right)\)
    end if
    Add \(S^{p^{+1}}\) and \(S^{p^{+2}}\) to population set \(P\)
    Identify the two worst solutions, \(S^{k}\) and \(S^{l}\) in \(P\)
    Remove \(S^{k}\) and \(S^{l}\) from \(P\)
    TabuSet \(\leftarrow\left(S^{i}, S^{j}\right)\)
        until a stopping criterion is satisfied
        return \(S^{\text {best }}\)
```


### 5.3. Calibration of the metaheuristic algorithms

Several parameters affect the performance of the algorithms. These parameters were tuned separately by performing an experimental analysis for each problem structure (small-small, small-large, large-small and large-large). TS algorithm has four parameters that need to be tuned, i.e., TLS, MNSS, MIWOI and MIL. TS/PR algorithm has two parameters that need tuning. i.e., P_size and MIWI. Different levels of these parameters were used to perform the experimental design. The levels are as follows:

## TS algorithm

$$
\begin{array}{ll}
\text { TLS }-\{5,10,15,20,25,30\} & \text { MIWOI }-\{5,10,15,20,25,30\} \\
\text { MNSS }-\{2,4,6,8,10\} & \text { MIL }=\{5,10,15,20,25,30\}
\end{array}
$$



Figure 14. Flowchart for TS/PR

## TS/PR algorithms

$$
\text { P_size }-\{5,10,15,20\}
$$

$$
\text { MIWI }-\{3,5,8,10\}
$$

These levels were determined based on preliminary runs made by varying each parameter separately while keeping others fixed at a high value. Performing a full factorial experiment on all four parameters of the TS algorithm would require a lot of experimental runs and thus, would be time consuming. Hence, the analysis was performed in a two 2-
factor ANOVA instead of one 4 -factor ANOVA. Since, TLS and MNSS are type of memory structures that affect the direction of the search procedure significantly, these are the most important parameters. Therefore, the first 2-factor ANOVA include TLS and MNSS as the factors of interest while the other two are kept at a high value of 30 . The best factor levels are chosen from this analysis and then the second 2-factor ANOVA is performed, which includes MIWOI and MIL as the factors of interest. In the case of TS/PR, a 2-factor ANOVA is performed with P_size and MIWI as the factors of interest. The parameter values of TS to be used in TS/PR is determined from the previous ANOVA analysis. It is worth noting that the smaller level of a factor is preferred if no significant difference is observed between two factor levels. For example, if there is no significant difference between the levels of MIWOI set at 15 and 20, then 15 is chosen as the best level because it requires less computational time. The factor levels chosen for different algorithms are shown in Table A. 1 of Appendix A. The result of ANOVA is shown in Table A. 2 - A. 41 in Appendix A.

## 6. DATA GENERATION

The data generation method used in this research is based on the previous study by Logendran and Subur (2004) and Shahvari and Logendran (2017). The problem instances are classified into four structures, (small, small), (small, large), (large, small) and (large, large). The first term in the parenthesis denotes the number of products and the second term denotes the number of jobs belonging to each product. The "small" and "large" refer to a number generated from a uniform distribution unif $[2,5]$ and unif $[6,10]$, respectively. These ranges are determined by reviewing the previous literature by Schaller et al. (2000), Lu and Logendran (2011), and Shahvari and Logendran (2017), and considering the computational time. In these problem structures, (small, small) and (large, large) are smalland large-size problems, respectively, whereas the other two are medium-size problems. The problem has two components, with the first component requiring two machines before assembly and the second component requiring only one machine before it is ready for assembly. In addition, storages 1 and 2 have a minimum storage time of 120 min . in each, whereas storage 3 does not have any storage time restrictions. The shop layout is shown in Figure 3.

The run time of a component on any machine is given by the size of the component divided by the machine's throughput. Usually, in a plant with higher throughput, the throughput of the associated machines would also be higher, i.e., the run times would be lower. Hence, in this research, the individual machine throughput is first generated based on the plant's throughput. Then, the run times are determined by dividing the size of the component by the machine's capacity. Consider a product with a batch size of 100 kg that requires 20 kg of component 1 and 80 kg of component 2. The machine throughput of machines 1 (processes component 1) and 3 (processes component 2) are $40 \mathrm{~kg} / \mathrm{hr}$ and 80 $\mathrm{kg} / \mathrm{hr}$, respectively. The run time of this product on machine 1 and machine 3 is thus 30 min . and 60 min ., respectively. The capacity of a plant is classified into three levels, low, medium and high. The plant capacity at each level is determined as follows:

- Low plant throughput - Unif $[3,5] \mathrm{kg} / \mathrm{hr}$
- Medium plant throughput - Unif $(5,8] \mathrm{kg} / \mathrm{hr}$
- High plant throughput - Unif $[8,10] \mathrm{kg} / \mathrm{hr}$

After the plant capacity is known, the machine capacity is generated as follows:

- Machine 1 throughput - Unif $[24,44] \% \times$ plant capacity
- Machine 2 throughput - Unif $[18,38] \% \times$ plant capacity
- Machine 3 throughput - Unif $[87,107] \% \times$ plant capacity
- Machine 3 throughput - Unif $[71,91] \% \times$ plant capacity

These ranges are determined based on the data of five different plants. The average throughput of each machine with respect to the plant throughput was calculated. This average was varied by 10 percentage points on either side to obtain the range for the uniform distribution. The batch size of each product is obtained from a uniform distribution unif $[400,600] \times 10 \mathrm{~kg}$. Similarly, percentage size of component 1 in each product is obtained from a uniform distribution unif $[10,30] \% \times$ product size. For example, if the batch size of a product is 5100 kg and the percentage size of component 1 is $20 \%$, then component 1 contains 1020 kg and component 2 contains the rest, i.e., 4080 kg . With the machine capacity and the component size known, the run times can be calculated easily as discussed above. These run times are rounded to the nearest integer in minutes.

In the problem considered in this research, there is some setup time incurred even when changing between jobs belonging to the same product. This setup time is typically lower than the setup time between jobs belong to different products. Thus, the setup times between jobs belonging to same and different products are generated from a uniform distribution unif $[3,5]$ min. and unif $[5,8]$ min., respectively. The equipment startup time is generated from a uniform distribution unif $[10,60] \mathrm{min}$. As noted before, there is a weight assigned to each job, and it is generated from a uniform distribution unif $[1,3]$, where 1 indicates the least important job. Three combinations of producer's and customers' weights are used to represent different scenarios. The value of $\alpha=0.6$ and $\beta=0.4$ indicates that the producer's objective should be prioritized. Similarly, the value of $\alpha=0.4$ and $\beta=0.6$ indicates that the customers' objective should be prioritized. An equal weights of 0.5 for $\alpha$ and $\beta$ indicate equal importance to both objectives.

The release time of a job and the availability time of a machine is generated from a Poisson distribution with a mean arrival rate of 3 per hour. The random number must take integer values. Shahvari and Logendran (2017) have used Poisson distribution to simulate the model for job arrival and machine availability. In a flow shop problem, the availability time of machines belonging to second or later stages must be delayed considering the machine availability time of earlier stages. For example, a machine availability time of 20 min. for machine 1 indicates that this machine is still processing a job from previous planning horizon. In a flow shop, this job must also be processed by machine 2 and other successor machines. In this problem, machines 1 and 3 are the first machines to process components 1 and 2, respectively. The availability time of these machine can be generated from an exponential distribution with $\lambda=1 / 20$. For machines 2 and 4 , the availability time is calculated as follows:

$$
\begin{align*}
& a_{2}=\operatorname{Max}\left(a_{1}+t_{1}, \operatorname{Exp}[20]+\overline{S_{2}}\right)+\overline{R_{2}}  \tag{7.1}\\
& a_{4}=\operatorname{Max}\left(a_{3}+t_{3}, \operatorname{Exp}[20]+\overline{S_{4}}, a_{2}\right)+\overline{R_{4}} \tag{7.2}
\end{align*}
$$

$\bar{S}_{l}$ indicates the average setup time on machine $i$ and $\bar{R}_{l}$ indicates the average run time on machine $i$. These are calculated as follows:

$$
\begin{align*}
& \bar{S}_{l}=\left(\sum_{j=1}^{p} \sum_{j^{\prime}=1}^{p} S_{i j j^{\prime}}\right) / p^{2}  \tag{7.3}\\
& \bar{R}_{l}=\sum_{j=1}^{p} R_{i j} / p \tag{7.4}
\end{align*}
$$

Previous works by Kim et al. (2002) and Pandya and Logendran (2011) have shown that the generation of meaningful due dates play a vital role in evaluating the effectiveness of the proposed heuristic algorithm. Two factors, namely the tardiness factor $(\tau)$ and the range factor $(R)$ are used to generate different due dates. The tardiness factor $\tau$ is defined as $\tau=1-\bar{d} / C_{\max }$ where $\bar{d}$ is the average due date and $C_{\max }$ is the estimated maximum completion time of all jobs. The due date range factor R indicates the due date variability and is defined as $R=\left(d_{\max }-d_{\min }\right) / C_{\max }$, where $d_{\max }$ and $d_{\min }$ indicate the maximum and minimum due date, respectively. Different combinations of $\tau$ and $R$ can be used to generate due dates representing different characteristics as shown in Table 22.

Table 22. Due date classification

| $\boldsymbol{T}$ | $\boldsymbol{R}$ | Degree of Tightness | Width of Range |
| :---: | :---: | :---: | :---: |
| 0.2 | 0.2 | Loose | Narrow |
| 0.2 | 0.5 | Loose | Medium |
| 0.2 | 0.8 | Loose | Wide |
| 0.5 | 0.2 | 0.5 | Narrow |
| 0.5 | 0.5 | 0.5 | Medium |
| 0.5 | 0.8 | 0.5 | Wide |
| 0.8 | 0.2 | 0.8 | Narrow |
| 0.8 | 0.5 | 0.8 | Medium |
| 0.8 | 0.8 | 0.8 | Wide |

In this research, the range factor is set at 0.5 , which provides a medium range of due dates. The due date is generated from a composite uniform distribution. With a probability $\tau$, the due date is from a distribution unif $[\bar{d}-R \bar{d}, \bar{d}]$ and with a probability (1 - $\tau)$, it is from a distribution unif $\left[\bar{d}, \bar{d}+\left(C_{\max }-\bar{d}\right) R\right]$. The estimated maximum completion time can be obtained by a similar iterative procedure shown before to calculate the completion time of a job (Section 5.1.3 and Section 5.1.7). However, in this case, an adjusted average setup time is used instead of using actual setup time. The adjusted average setup time is given by $\delta \times \bar{S}_{l}$, where $\delta$ is the average setup time adjuster. In reality, the best schedules tend to use smaller setup time. Hence, using just the average setup time ( $\bar{S}_{l}$ ) would provide an inaccurate representation of the makespan. Thus, $\delta$ is introduced to represent makespan for the best schedules. To obtain a value of $\delta$, the coefficient of variation $(\mathrm{CV})$ is calculated for the setup times on a machine, $\mathrm{CV}=s / \bar{x}$, where s is the sample standard deviation and $\bar{x}$ is the mean. A linear relationship between $\delta$ and CV is assumed, as shown in Figure 15: $\delta=0.9$ when $\mathrm{CV}=0.01$ and $\delta=0.1$ when $\mathrm{CV}=1$.


Figure 15. Relationship between $\delta$ and CV

## 7. THE QUALITY OF SOLUTIONS OBTAINED FROM THE PROPOSED HEURISTIC

The major advantage of a heuristic algorithm is its ability to find optimal or near optimal solution in a very short time as compared to using exact methods to find an optimal solution. However, the efficacy of the proposed algorithm must be tested before it is applied to solve real problems. This is done by evaluating the quality of the solution obtained by the algorithm and the computational time it takes. Typically, an optimal solution obtained using an exact method is compared with the solution obtained from the algorithm. However, for NP-hard problems, an exact method such as brand-and-bound might not identify an optimal solution in a reasonable time. If an optimal solution is unknown, then the solution obtained from the algorithm can be compared with a suitable lower bound for the problem. The mathematical model discussed in Chapter 4 is used to obtain an optimal solution or lower bound for small-size problems.

Two mathematical models are used, MILP1 for PN sequence and MILP2 for nonNPN sequence. There are two sets of binary variables and one set of integer variables used in the models. For MILP1, the binary variables are $W_{j k l}$ and $Y_{j l j^{\prime}(l+1)}$, and the integer variable is $H s_{i l}$. The total number of $W_{j k l,}, Y_{j j^{\prime}(l+1)}$ and $H s_{i l}$ are $N^{2}, m^{2}(N-1)$ and $\left(\sum_{g=1}^{m} u_{g}+1\right) \times N$. Therefore, for the example problem discussed in Chapter 5, i.e., a 4-machine problem with 10 jobs belonging to 3 groups, will have $10^{2} W_{j k l}, 3^{2} \times(10-1) Y_{j l j^{\prime}(l+1)}$ and $4 \times 10 H s_{i l}$. The total number of binary and integer variables are 127 and 40, respectively. As discussed in Chapter 4, the number of binary variables for the same problem considering NPN sequence (i.e., MILP2) increases by $\left(\sum_{g=1}^{m} u_{g}+1\right)$ times, i.e. times the total number of machines. Thus, the number of binary and integer variables for MILP2 are 508 and 40, respectively.

An optimal solution for a problem was identified by solving the corresponding mathematical model using branch-and-bound technique incorporated in the IBM CPLEX 12.7 software. The software was installed and run on an Intel Core i7-2600, 3.4 GHz processor with 8 GB RAM. The amount of time required by CPLEX to identify an optimal solution is large, partly due to large number of binary variables in the model. The presence
of big-M constraints in the model results in weak LP relaxation, further increasing complexity of the model. In addition, the problems have multiple solutions with the same OFV, which results in model symmetry. The combination of these factors presents a major obstacle in solving the MILP model to optimality using CPLEX. Based on the test, CPLEX is not able to identify an optimal solution even for small-size problems within an allotted time of 8 hours ( 28800 seconds). CPLEX offers a lower bound $(L B)$ in case it doesn't identify an optimal solution. However, the lower bounds offered is trivial because of weak LP relaxation and model symmetry.

Sixteen small-size problem instances were generated and solved using CPLEX for both permutation and non-permutation sequences. The data generated for these problem instances used the same procedure as described in Chapter 6. Table 22 shows the results of CPLEX runs. From the table, it can clearly be seen that the model for NPN sequence is more complex than the model for PN sequence. The CPLEX computation time ( $C T$ ) for MILP2 is significantly higher than MILP1 for each problem instance. In addition, CPLEX was able to solve MILP2 to optimality in only 9 instances out of 16 problem instances, within an allocated time of 28800 seconds. However, in the case of MILP1, optimal solution was obtained for 13 problems. For problems 1-9 (where optimal solution was found for both PN and NPN sequences), an average improvement of $2.68 \%$ was obtained by adopting the NPN sequence. The percentage improvement is calculated using the formula, $\left(\left(U B_{P N}-U B_{N P N}\right) / U B_{P N}\right) * 100$, where $U B$ stands for upper bound. This seems to suggest that it might be beneficial to drop the PN sequence restriction. However, in the case of 4 problems, no improvement was observed by adopting the NPN sequence in spite of the longer computational time required by CPLEX to obtain the optimal solution. Consider problem 5 as an example. The computation time taken to find an optimal solution for MILP1 is 1.538 seconds, whereas it took 899.99 seconds to arrive at the same solution for MILP2. The result obtained from a paired t-test (significance level of 0.05) on the OFV of both sequences showed that the improvement observed on the OFV by allowing for NPN sequence is not statistically significant ( $p$ value: 0.12 ). This result pertains to only small problems (total jobs < 12) and cannot be extended to larger problems. For larger problems, statistical analysis is performed in Chapter 8 using the solution obtained from using the
meta-heuristics. From Table 23, it can be seen that the gap between the upper bound and the lower bound is very large, i.e. the lower bound is not meaningful. CPLEX includes a symmetry detection parameter to automatically detect certain type of symmetry in the model and allows the user to choose the degree of symmetry breaking reduction to be executed during the preprocessing phase. The default setting allows CPLEX to choose the degree of symmetry breaking to apply. A trial run of the problem was performed with the default value as well as the most aggressive setting for the symmetry breaking. Since the most aggressive setting did not yield the desired performance improvements, default setting was used and reported in the table below. The percentage gap is calculated using the formula, $((U B-L B) / U B) * 100$.

Table 23. CPLEX runs of MILP1 and MILP2

| $\begin{aligned} & \text { \# } \\ & \text { E } \\ & \text { I } \\ & \text { od } \end{aligned}$ |  | $\begin{aligned} & 0 \\ & \stackrel{0}{0} \\ & 0 \\ & \# \\ & \# \end{aligned}$ | PN CPLEX (MILP1) |  |  | NPN CPLEX (MILP2) |  |  | \%Imp | \% Gap |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  | $\boldsymbol{U B} \boldsymbol{B}_{P N}$ | $C T_{P N}$ | $L \boldsymbol{B}_{\text {PN }}$ | $\boldsymbol{U B} \boldsymbol{B}_{\text {NPN }}$ | $\boldsymbol{C T}_{\text {NPN }}$ | $L B_{\text {NPN }}$ |  | $\begin{gathered} z_{2}^{2} \\ 0 \\ 0 \\ n \\ 2 \\ 2 \\ 0 \end{gathered}$ | $\begin{aligned} & 2 \\ & 0 \\ & 0 \\ & 0 \\ & 0 \\ & 0 \\ & 2 \\ & 2 \\ & 0 \end{aligned}$ |
| 1 | 2 | 7 | 0.407042 | 0.292 | 0.407042 | 0.407042 | 8.15 | 0.407042 | 0.00\% | 0.0\% |  |
| 2 | 2 | 8 | 0.434941 | 1.47 | 0.434941 | 0.434941 | 8.60 | 0.434941 | 0.00\% | 0.0\% |  |
| 3 | 2 | 8 | 0.315878 | 2.401 | 0.315878 | 0.294545 | 32.02 | 0.294545 | 6.75\% | 0.0\% |  |
| 4 | 3 | 9 | 0.235904 | 1.329 | 0.235904 | 0.202296 | 101.41 | 0.202296 | 14.25\% | 0.0\% |  |
| 5 | 3 | 9 | 0.517717 | 1.538 | 0.517717 | 0.517717 | 899.99 | 0.517717 | 0.00\% | 0.0\% |  |
| 6 | 4 | 10 | 0.199779 | 3.786 | 0.199779 | 0.199779 | 422.23 | 0.199779 | 0.00\% | 0.0\% |  |
| 7 | 2 | 10 | 0.235967 | 4.87 | 0.235967 | 0.231198 | 69.27 | 0.231198 | 2.02\% | 0.0\% |  |
| 8 | 3 | 11 | 0.265741 | 10.69 | 0.265741 | 0.262808 | 11111.63 | 0.262808 | 1.10\% | 0.0\% |  |
| 9 | 3 | 11 | 0.206612 | 18.7 | 0.206612 | 0.206597 | 668.91 | 0.206597 | 0.01\% | 0.0\% |  |
| 10 | 5 | 12 | 0.139089 | 51.54 | 0.139089 | 0.129819 | 28800 | 0.06204 | - | 0.0\% | 52.2\% |
| 11 | 4 | 12 | 0.369047 | 55.81 | 0.369047 | 0.367567 | 28800 | 0.221364 | - | 0.0\% | 39.8\% |
| 12 | 3 | 12 | 0.212582 | 44.8 | 0.212582 | 0.212582 | 28800 | 0.145533 | - | 0.0\% | 31.5\% |
| 13 | 4 | 15 | 0.215353 | 3158.21 | 0.215353 | 0.215353 | 28800 | 0.057301 | - | 0.0\% | 73.4\% |
| 14 | 4 | 16 | 0.287526 | 28800 | 0.231569 | 0.398584 | 28800 | 0.047468 | - | 19.5\% | 88.1\% |
| 15 | 5 | 18 | 0.214537 | 28800 | 0.050518 | 0.252673 | 28800 | -0.01374 | - | 76.5\% | 105.4\% |
| 16 | 5 | 20 | 0.244529 | 28800 | 0.040691 | 0.331723 | 28800 | -0.01486 | - | 83.4\% | 104.5\% |
|  |  |  | Average | 4063.70 |  |  | 12408.15 |  | 2.68\% |  |  |

Note: Computation time (CT) is measured in seconds.

The solution obtained using meta-heuristic algorithms for the same problem is shown in Table 24. The computational time of these algorithms is very short as compared to CPLEX, which shows the time-wise advantage of using meta-heuristic algorithms. The description of the algorithms used is as follows:

- ALG1: Short-term TS using PTB2 for PN sequence
- ALG2: Short-term TS using PTB2 for NPN sequence
- ALG3: TS/PR for PN sequence
- ALG4: TS/PR for NPN sequence

Table 24. Solutions from metaheuristic algorithms

| Problem <br> $\#$ | ALG1 |  | ALG2 |  | ALG3 |  | ALG4 |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | $U B_{A L G 1}$ | $C T_{A L G 1}$ | $U B_{A L G 2}$ | $C T_{A L G 2}$ | $U B_{A L G 3}$ | $C T_{A L G 3}$ | $U B_{A L G 4}$ | $C T_{A L G 4}$ |
| 1 | 0.407042 | 0.037 | 0.407042 | 0.203 | 0.407042 | 0.736 | 0.407042 | 8.379 |
| 2 | 0.434941 | 0.028 | 0.434941 | 0.384 | 0.434941 | 1.219 | 0.434941 | 15.715 |
| 3 | 0.315878 | 0.027 | 0.294545 | 0.319 | 0.315878 | 1.211 | 0.294545 | 13.028 |
| 4 | 0.235904 | 0.057 | 0.241453 | 0.387 | 0.235904 | 1.257 | 0.202296 | 20.948 |
| 5 | 0.517717 | 0.04 | 0.517717 | 0.472 | 0.517717 | 1.685 | 0.517717 | 19.063 |
| 6 | 0.199779 | 0.078 | 0.242513 | 0.617 | 0.199779 | 2.056 | 0.199779 | 24.531 |
| 7 | 0.235967 | 0.061 | 0.231198 | 0.596 | 0.235967 | 2.158 | 0.231198 | 25.016 |
| 8 | 0.320777 | 0.108 | 0.337624 | 0.724 | 0.265741 | 3.369 | 0.262808 | 39.123 |
| 9 | 0.209133 | 0.081 | 0.209133 | 0.963 | 0.209133 | 2.869 | 0.209133 | 35.201 |
| 10 | 0.141958 | 0.217 | 0.164762 | 2.081 | 0.139089 | 4.674 | 0.116362 | 55.509 |
| 11 | 0.38593 | 0.083 | 0.38593 | 0.911 | 0.38593 | 3.138 | 0.38593 | 34.586 |
| 12 | 0.212582 | 0.091 | 0.212582 | 1.016 | 0.212582 | 3.258 | 0.212582 | 35.999 |
| 13 | 0.273962 | 0.216 | 0.288937 | 4.554 | 0.217137 | 8.991 | 0.215353 | 151.27 |
| 14 | 0.307629 | 0.254 | 0.307629 | 3.115 | 0.287526 | 8.164 | 0.287526 | 90.58 |
| 15 | 0.251587 | 0.426 | 0.244759 | 4.392 | 0.211265 | 35.121 | 0.194653 | 332.458 |
| 16 | 0.236762 | 0.654 | 0.268932 | 6.615 | 0.220767 | 24.03 | 0.216128 | 328.309 |

The output from each algorithm is compared to the upper and lower bounds obtained from CPLEX. The PN algorithms (ALG1 and ALG3) are compared to the MILP1 bounds and the NPN algorithms (ALG2 and ALG4) are compared to the MILP2 bounds. For MILP1, an optimal solution was obtained from CPLEX for 13 problems and the percentage deviation of the algorithms from the optimal solution is shown in Table 25. The percentage deviation is calculated using the formula, $\left(\left(U B_{A L G}-U B_{P N}\right) / U B_{P N}\right) * 100$. The
heuristics show a good overall performance with an average percentage deviation of 3.55 $\%$ and $0.49 \%$ for ALG1 and ALG3, respectively. However, for some problems (problems 1, 8 and 13), the solution obtained from ALG1 has a large deviation from the optimal solution. In contrast, ALG3 performs better for all problem instances with the maximum deviation of $4.37 \%$ for problem 11 . Hence, TS/PR algorithm shows superior performance even for small problem instances. Note that this result is based on few small problems, which cannot be relied upon to make an objective conclusion. A detailed statistical analysis is performed in Chapter 8 to uncover the statistical significance of the algorithms.

Table 25. Average deviation for PN algorithms from CPLEX optimal solution

| Problem | Average Deviation \% |  |
| :---: | :---: | :---: |
|  | $\boldsymbol{U B}_{A L G 1}$ vs $\boldsymbol{U} \boldsymbol{B}_{P N}$ | $\boldsymbol{U B}_{A L G 3}$ vs $\boldsymbol{U} \boldsymbol{B}_{P N}$ |
| 1 | $0.00 \%$ | $0.00 \%$ |
| 2 | $0.00 \%$ | $0.00 \%$ |
| 3 | $0.00 \%$ | $0.00 \%$ |
| 4 | $0.00 \%$ | $0.00 \%$ |
| 5 | $0.00 \%$ | $0.00 \%$ |
| 6 | $0.00 \%$ | $0.00 \%$ |
| 7 | $0.00 \%$ | $0.00 \%$ |
| 8 | $17.16 \%$ | $0.00 \%$ |
| 9 | $1.21 \%$ | $1.21 \%$ |
| 10 | $2.02 \%$ | $0.00 \%$ |
| 11 | $4.37 \%$ | $4.37 \%$ |
| 12 | $0.00 \%$ | $0.00 \%$ |
| 13 | $21.39 \%$ | $0.82 \%$ |

Table 26. Average deviation for PN algorithms from CPLEX bounds

|  | Average Deviation \% |  |  |  |
| :---: | :---: | :---: | :---: | :---: |
| Problem \# | $\boldsymbol{U} \boldsymbol{B}_{A L G I}$ vs $\boldsymbol{U} \boldsymbol{B}_{P N}$ | $\boldsymbol{U} \boldsymbol{B}_{A L G I}$ vs $\boldsymbol{L} \boldsymbol{B}_{P N}$ | $\boldsymbol{U B}_{A L G 3}$ vs $\boldsymbol{U} \boldsymbol{B}_{P N}$ | $\boldsymbol{U} \boldsymbol{B}_{A L G 3}$ vs $\boldsymbol{L} \boldsymbol{B}_{P N}$ |
| 14 | $6.53 \%$ | $24.72 \%$ | $0.00 \%$ | $19.46 \%$ |
| 15 | $14.73 \%$ | $79.92 \%$ | $-1.55 \%$ | $76.09 \%$ |
| 16 | $-3.28 \%$ | $82.81 \%$ | $-10.76 \%$ | $81.57 \%$ |

For the remaining three problems (problem 14, 15 and 16), where an optimal solution was not identified, the percentage deviations are measured with both upper and lower bounds of CPLEX as shown in Table 26. The percentage deviation from upper and
lower bounds are obtained from the formula, $\left(\left(U B_{A L G}-U B_{P N}\right) / U B_{P N}\right) * 100$ and $\left(\left(U B_{A L G}\right.\right.$ $\left.\left.-L B_{P N}\right) / U B_{P N}\right) * 100$, respectively. It should be noted that the lower bounds are infeasible solutions, which might not be close to the true optimal solution. Since, the lower bounds obtained from CPLEX are weak, large percentage deviations are observed when comparing the lower bounds to the solutions obtained from the algorithms. For problems 14-16, the average deviations of ALG1 and ALG3 with CPLEX upper bounds are $5.99 \%$ and $-4.10 \%$, respectively. The negative sign indicates that the solution from the algorithm is better that the best feasible solution obtained by CPLEX within the allotted time, i.e., ALG3 was able to obtain a better quality solution in less computational time as compared to CPLEX. This highlights the advantages of using meta-heuristic algorithms as the problem complexity increases.

Table 27. Average deviation for NPN algorithms from CPLEX optimal solution

| Problem \# | Average Deviation \% |  |
| :---: | :---: | :---: |
|  | UB $_{\text {ALG2 vs UB }}$ NPN | UB $_{\text {ALG4 vs UB }}$ NPN |
|  | $0.00 \%$ | $0.00 \%$ |
| 2 | $0.00 \%$ | $0.00 \%$ |
| 3 | $0.00 \%$ | $0.00 \%$ |
| 4 | $16.22 \%$ | $0.00 \%$ |
| 5 | $0.00 \%$ | $0.00 \%$ |
| 6 | $17.62 \%$ | $0.00 \%$ |
| 7 | $0.00 \%$ | $0.00 \%$ |
| 8 | $22.16 \%$ | $0.00 \%$ |
| 9 | $1.21 \%$ | $1.21 \%$ |

In the case of NPN model (MILP2), an optimal solution was obtained by CPLEX for 9 problems (problems 1-9). The average deviation of NPN algorithms, ALG2 and ALG4, from the optimal solution is shown in Table 27. ALG4 seems to perform far better with an average deviation of $0.13 \%$ as compared to that of ALG2 (6.36\%). For problems 10-16, the deviation of the algorithms from CPLEX bounds is shown in Table 28. Similar to the PN algorithms, the deviation of the algorithm with the lower bound is high. The deviation of both the algorithms, ALG2 and ALG3, from CPLEX upper bound is negative ( -0.67 for ALG2 and $-18.39 \%$ for ALG4). This indicates that, overall, the algorithms obtained better quality solution than CPLEX in less computational time. It can be observed
from Table 28 that, as the problem size increases ( $\#$ of jobs $\geq 15$ ), the algorithms begin to outperform the branch-and-bound technique utilized by CPLEX.

Table 28. Average deviation for NPN algorithms from CPLEX bounds

| Problem \# | Average Deviation \% |  |  |  |
| :---: | :---: | :---: | :---: | :---: |
|  | $\mathbf{U B}_{\text {ALG } 2}$ vs UB ${ }_{\text {NPN }}$ | $\mathbf{U B}_{\text {ALG2 }}$ vs LB ${ }_{\text {NPN }}$ | $\mathrm{UB}_{\text {ALG4 }}$ vs UB ${ }_{\text {NPN }}$ | $\mathrm{UB}_{\text {ALG4 }} \mathrm{vs}$ LB ${ }_{\text {NPN }}$ |
| 10 | 21.21\% | 62.35\% | -11.56\% | 46.68\% |
| 11 | 4.76\% | 42.64\% | 4.76\% | 42.64\% |
| 12 | 0.00\% | 31.54\% | 0.00\% | 31.54\% |
| 13 | 25.47\% | 80.17\% | 0.00\% | 73.39\% |
| 14 | -29.57\% | 84.57\% | -38.63\% | 83.49\% |
| 15 | -3.23\% | 105.61\% | -29.81\% | 107.06\% |
| 16 | -23.35\% | 105.53\% | -53.48\% | 106.88\% |
| Average | -0.67\% | 73.20\% | -18.39\% | 70.24\% |

## 8. RESULTS

Chapter 7 showed that the proposed search algorithms are efficient in solving small problem instances as compared to a branch-and-bound technique. While the branch-andbound technique takes hours to solve a small problem instance, the heuristic algorithm can do so in a matter of minutes or even seconds. The main purpose of this research is to develop efficient algorithms to solve an assembly flow shop problem in an industrial setting and evaluate the performance of these algorithms under different constraints and complexity, i.e., PN vs NPN sequence, continuous vs non-continuous production and small to large problem instances. Thus, an experimental setup is designed to address the following issues:

1. To determine if the solutions from the proposed algorithms are statistically different in terms of solution quality and computational time for a given test problem.
2. To evaluate the performance of a PN vs NPN sequence and determine if the NPN sequence offers a significant advantage over PN sequence.
3. To evaluate if the performance of the PN vs NPN sequence differs in noncontinuous production as compared to continuous production.

### 8.1. Experimental Design

A multi-factor split-plot experimental design is used to address the research questions above. The solution quality, measured in terms of its objective function value and the computation time, are used as the response variables to analyze the algorithms' performance. The factors that are used to generate a particular problem such as problem structure (Str), plant capacity (PC), due date tightness (DDT), number of shifts (NoS) and scenario ( $S c$ ) are placed in the main-plot. The four algorithms used in this research belong to the sub-plot factor, as it is a factor of primary importance. The split-plot design with six factors is shown in Table 29. All the problems are randomly generated using the method described in Chapter 6. Hence, no two problems are exactly the same, which causes large
variability in the response variables. This variation can be reduced by treating each problem instance as a block. Blocking is necessary to eliminate the impact caused by the different problem instances. Hence, if a difference in algorithm's performance is identified, it can be wholly attributed to the effect of the algorithm.

Table 29. Factors and their levels in the experiment

| Factor Name | Levels |
| :--- | :--- |
| Whole Plot |  |
| $\quad$ Structure (Str) | (Small, Small), (Small, Large), (Large, Small), (Large, Large) |
| Plant capacity (PC) | Low, Medium, High |
| Due date tightness (DDT) | Tight (0.2), Medium (0.5), Loose (0.8) |
| Number of shift (NoS) | $1,2,3$ |
| Scenario (Sc) | $(\alpha=0.4, \beta=0.6),(\alpha=0.5, \beta=0.5),(\alpha=0.6, \beta=0.4)$ |
| Sub-Plot |  |
| $\quad$ Algorithm (Alg) | ALG1, ALG2, ALG3, ALG4 |

Ten replications have been randomly generated for every combination of Str, PC, $D D T, N o S$ and $S c$ factors. Each of these replications have been solved by all four algorithms, which resulted in a total number of 12960 runs ( 324 combinations of Str, PC, $D D T, N o S$ and $S c \times 10$ replications $\times 4$ algorithms $=12960$ ). The statistical model for this design is:

$$
\begin{aligned}
& \quad y_{i j k l m n o}=\mu+\gamma_{i}+\rho_{j}+\tau_{k}+\varphi_{l}+\omega_{m}+\delta_{n}+(\rho \tau)_{j k}+(\rho \varphi)_{j l}+(\rho \omega)_{j m}+ \\
& (\rho \delta)_{j n}+(\tau \varphi)_{k l}+(\tau \omega)_{k m}+(\tau \delta)_{k n}+(\varphi \omega)_{l m}+(\varphi \delta)_{l n}+(\omega \delta)_{m n}+(\rho \tau \varphi)_{j k l}+ \\
& (\rho \tau \omega)_{j k m}+(\rho \tau \delta)_{j k n}+(\tau \varphi \omega)_{k l m}+(\tau \varphi \delta)_{k l n}+(\varphi \omega \delta)_{l m n}+(\rho \tau \varphi \omega)_{j k l m}+ \\
& (\rho \tau \varphi \delta)_{j k l n}+(\tau \varphi \omega \delta)_{k l m n}+(\rho \tau \varphi \omega \delta)_{j k l m n}+\theta_{j k l m n}+\vartheta_{o}+(\gamma \vartheta)_{i o}+(\rho \vartheta)_{j o}+ \\
& (\tau \vartheta)_{k o}+(\varphi \vartheta)_{l o}+(\omega \vartheta)_{m o}+(\delta \vartheta)_{n o}+(\rho \tau \vartheta)_{j k o}+(\rho \varphi \vartheta)_{j l o}+(\rho \omega \vartheta)_{j m o}+ \\
& (\rho \delta \vartheta)_{j n o}+(\tau \varphi \vartheta)_{k l o}+(\tau \omega \vartheta)_{k m o}+(\tau \delta \vartheta)_{k n o}+(\varphi \omega \vartheta)_{l m o}+(\varphi \delta \vartheta)_{l n o}+ \\
& (\omega \delta \vartheta)_{m n o}+(\rho \tau \varphi \vartheta)_{j k l o}+(\rho \tau \omega \vartheta)_{j k m o}+(\rho \tau \delta \vartheta)_{j k n o}+(\tau \varphi \omega \vartheta)_{k l m o}+(\tau \varphi \delta \vartheta)_{k l n o}+ \\
& (\varphi \omega \delta \vartheta)_{l m n o}+(\rho \tau \varphi \omega \vartheta)_{j k l m o}+(\rho \tau \varphi \delta \vartheta)_{j k l n o}+(\tau \varphi \omega \delta \vartheta)_{k l m n o}+(\rho \tau \varphi \omega \delta \vartheta)_{j k l m n o}+ \\
& \epsilon_{i j k l m n o}
\end{aligned}
$$

$i=1,2, \ldots, 10 ; j=1,2,3,4 ; k, l, m, n=1,2,3$; and $o=1,2,3,4$ where $\mu$ is the overall mean effect, $\gamma_{i}$ is the replicate (Rep) effect, $\rho_{j}$ is the $\operatorname{Str}$ effect, $\tau_{k}$ is the $P C$ effect,
$\varphi_{l}$ is the $D D T$ effect, $\omega_{m}$ is the $N o S$ effect, $\delta_{n}$ is the $S c$ effect, $\theta_{j k l m n}$ is the main-plot error , $\vartheta_{o}$ is the $A l g$ effect, and $\epsilon_{i j k l m n o}$ is the sub-plot error.

The normal probability plot of the objective function value is shown in Figure 16, which shows that the distribution of the objective function value is not exactly normal. However, in this research, this distribution is considered normal for the purpose of statistical analysis because 1) ANOVA is robust to normality assumption for large sample sizes, 2) data transformation does not convert the distribution into normal, and 3) lack of widely accepted non-parametric test for multi-factor analysis. The resulting ANOVA table is shown in Table 30.


Figure 16. Normality of objective function value

Table 30. ANOVA of the objective function value in split-plot design

| Source | SS | MS Num | DF Num | F Ratio | Prob > F |
| :--- | ---: | ---: | ---: | ---: | ---: |
| Main-plot | 0.07031 | 0.00781222 |  | 1.10939143 | 0.3522 |
| Rep | 12.6486 | 4.2162 | 3 | 598.730553 | 0.0000 |
| Str | 0.60542 | 0.30271 | 2 | 42.9869849 | 0.0000 |
| PC | 114.59 | 57.295 | 2 | 8136.29975 | 0.0000 |
| DDT | 0.34202 | 0.17101 | 2 | 24.284643 | 0.0000 |
| Nos | 5.06754 | 2.53377 | 2 | 359.813461 | 0.0000 |
| Sc | 0.16757 | 0.02792833 | 6 | 3.96602307 | 0.0006 |
| Str*PC | 0.52939 | 0.08823167 | 6 | 12.5295277 | 0.0000 |



| PC*DDT*Sc*Alg | 0.00654 | 0.0002725 | 24 | 1.42920609 | 0.0799 |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Str*Nos*Sc*Alg | 0.00596 | 0.00016556 | 36 | 0.86830462 | 0.6933 |
| PC*Nos*Sc*Alg | 0.00498 | 0.0002075 | 24 | 1.08829454 | 0.3475 |
| DDT*Nos*Sc*Alg | 0.0069 | 0.0002875 | 24 | 1.50787798 | 0.0530 |
| Str*PC*DDT*Nos*Alg | 0.01511 | 0.00020986 | 72 | 1.10067808 | 0.2621 |
| Str*PC*DDT*Sc*Alg | 0.01626 | 0.00022583 | 72 | 1.18444908 | 0.1369 |
| Str*PC*Nos*Sc*Alg | 0.01273 | 0.00017681 | 72 | 0.92730854 | 0.6518 |
| Str*DDT*Nos*Sc*Alg | 0.01228 | 0.00017056 | 72 | 0.89452858 | 0.7253 |
| PC*DDT*Nos*Sc*Alg | 0.01131 | 0.00023563 | 48 | 1.23580435 | 0.1276 |
| Str*PC*DDT*Nos*Sc*Alg | 0.02522 | 0.00017514 | 144 | 0.91856721 | 0.7475 |
| Subplot error | 1.66794 | 0.00019067 | 8748 |  |  |
| Total | 166.3177 |  | 12959 |  |  |

Based on the ANOVA table, all factors in the main-plot (Str, PC, DDT, NoS and $S c$ ) have statistically significant effect on the objective function value of the tested problems (p-value $<0.05$ ). After accounting for the effect of these factors, the ANOVA table in the sub-plot shows statistically significant difference in the performance of the algorithms. Hence, further analysis is warranted to evaluate the difference in the algorithm's performance. Tukey test is a single step multiple pair-wise comparison procedure used to determine which means are significantly different from one another, which is used in this research to detect which pair of algorithms have significant difference in performance. The pairwise comparison as described in the hypotheses below is of particular interest.

Hypothesis 1: This hypothesis is to evaluate the performance of TS algorithm vs TS/PR algorithm in a PN sequence. The null hypothesis states that performance of the TS algorithm does not have any significant difference with that of TS/PR algorithm, in a PN sequence.

$$
\begin{aligned}
& H_{0}: \mu_{A L G 1}-\mu_{A L G 3}=0 \\
& H_{1}: \mu_{A L G 1}-\mu_{A L G 3} \neq 0
\end{aligned}
$$

Hypothesis 2: This hypothesis is to evaluate the performance of TS algorithm vs TS/PR algorithm in an NPN sequence. The null hypothesis states that performance of the TS algorithm does not have any significant difference with that of TS/PR algorithm, in an NPN sequence.

$$
\begin{aligned}
& H_{0}: \mu_{A L G 2}-\mu_{A L G 4}=0 \\
& H_{1}: \mu_{A L G 2}-\mu_{A L G 4} \neq 0
\end{aligned}
$$

Hypothesis 3: This hypothesis is to determine if the NPN sequence offers a significant advantage, in terms of solution quality, over that of the PN sequence. The null hypothesis states that NPN sequence does not offer any significant advantage over PN sequence.

$$
\begin{aligned}
& H_{0}: \mu_{A L G 3}-\mu_{A L G 4}=0 \\
& H_{1}: \mu_{A L G 3}-\mu_{A L G 4} \neq 0
\end{aligned}
$$

Based on the size of the problem, i.e. problem structure (small-small, small-large, large-small, large-large), the performance of the algorithms might vary. Hence the pairwise comparison of means is done is performed separately at each level of problem structure. The results of the analysis are summarized in Table 31 below. For every level of problem structure, a significant difference in the algorithm's performance was detected using ANOVA. From the table, it can be seen that $H_{0}$ is rejected in favor of $H_{1}$ in both, Hypothesis 1 and Hypothesis 2, for all problem structures. Hence, it can be concluded that TS/PR algorithm yields significantly better solutions as compared to TS algorithm in the case of both PN and NPN sequences for all problem structures. A pair-wise comparison between ALG3 and ALG4, i.e., Hypothesis 3 shows that there is no significant difference between these two algorithms for small-small, small-large and large-large problems. This means that, for these problem structures, the NPN sequence does not offer significant advantage over the PN sequence. However, for large-small problems, the null hypothesis $\left(H_{0}\right)$ is rejected in favor of $H_{1}$, which means that the NPN sequence yields significantly better solutions than the PN sequence for this problem structure. The large-small problems have large number of product types and small number of jobs in each product. In this scenario, the number of similar solutions (solutions having the same OFV) is less as compared to other problem structures. The NPN sequence is advantageous in these instances as it can improve the solution's OFV by performing NPN perturbations. A box-
plot is also presented to further highlight the performance of NPN sequence for different problem structures.

Table 31. Result of ANOVA and Tukey test on algorithm's performance

| Test | Small-small | Small-large | Large-small | Large-large |
| :--- | :--- | :--- | :--- | :--- |
| ANOVA (Alg) | Significant (p- <br> value: 0.000$)$ | Significant <br> $(\mathrm{p}$-value: 0.000$)$ | Significant <br> $(\mathrm{p}$-value: 0.000$)$ | Significant <br> $(\mathrm{p}$-value: 0.000$)$ |
| Hypothesis 1 | Reject null | Reject null | Reject null | Reject null |
| Hypothesis 2 | Reject null | Reject null | Reject null | Reject null |
| Hypothesis 3 | Fail to reject null | Fail to reject null | Reject null | Fail to reject null |

Figures 17-19 shows the box-plot of the comparison between algorithms for different problem structures. The deviation, in terms of solution quality, of ALG1 from ALG3 is calculated as $d e v_{1}=\left(\left(O F V_{A L G 1}-O F V_{A L G 3}\right) / O F V_{A L G 3}\right) \times 100$ and presented in Figure 17. It can be seen from the figure that, for a PN sequence, the average deviation increases as the problem complexity increases, i.e., ALG3 is more advantageous for larger problems. Moreover, the length of the box also increases with the problem complexity which means that ALG3 was able to identify a better solution than ALG1 in more instances for larger problems. The deviation of ALG2 from ALG4 is calculated as $\operatorname{dev}_{2}=$ $\left(\left(O F V_{A L G 2}-O F V_{A L G 4}\right) / O F V_{A L G 4}\right) \times 100$ and presented in Figure 18. This figure shows that, for an NPN sequence, ALG4 yields a better solution in more instances than ALG2, as the problem size increases. Figure 19 shows the average deviation of ALG3 from ALG4, which is calculated as $\operatorname{dev}_{2}=\left(\left(O F V_{A L G 3}-O F V_{A L G 4}\right) / O F V_{A L G 4}\right) \times 100$. The statistical analysis shows that there is no significant difference between PN sequence (ALG3) and NPN sequence (ALG4) in three out of four problem structures. Figure 19 illustrates that the deviation is not uniform across problem structures, i.e., deviation is higher for largesmall problems as compared to other problem structures. The problems belonging to smalllarge and large-large have large number of jobs belonging to the same product. As previously discussed in Section 5.1.4, this results in multiple solutions having the same objective function value. In this scenario, it is likely that a PN sequence has the same objective function value as an NPN sequence. Hence, the advantage offered by NPN sequence is not significant in these problem structures. However, NPN sequences are
advantageous in instances where there is large number of product types and small number of jobs in each product.


Figure 17. Deviation of ALG1 from ALG3


Figure 18. Deviation of ALG2 from ALG4


Figure 19. Deviation of ALG3 from ALG4

To answer the research question 3, solution quality of PN sequence (ALG3) and NPN sequence (ALG4) is compared separately for continuous and non-continuous production instances. The Tukey's test shows that there is no significant difference between PN and NPN sequences for continuous production. However, in the case of non-continuous production, the improvement offered by NPN sequence is significant. Figure 20 also shows that the percentage improvement is higher in the case of non-continuous production as compared to continuous production. As discussed in Section 3, the discontinuity in production allows NPN perturbations in predecessor machines without affecting the final completion times of jobs, which contributes to the evaluation of weighted tardiness, i.e., the second part of the objective function. This provides an advantage to the NPN sequence as it can improve setup times (the first part of the objective function) in predecessor machines without affecting the tardiness. Because of this, the advantage of NPN sequence over PN sequence is less pronounced in continuous production as compared to noncontinuous production.


Figure 20. Deviation of ALG3 from ALG4

Although the computational time (CT) taken by the meta-heuristic algorithm is short as compared to the branch-and-bound technique implemented in CPLEX, a split-plot ANOVA is performed using CT as the response variable to analyze the difference in the algorithm's efficiency, i.e., time taken to find the best solution. The result is shown in Table 32. In this case, the response variable has a huge deviation from a normal distribution, as seen from Figure 21. Hence, data transformation is performed using a log function to make
its distribution close to normal. The normal probability plot of the transformed data is shown in Figure 22.


Figure 21. Normal Probability Plot for CT


Figure 22. Normal Probability Plot for inversed CT

Table 32. ANOVA of the computational time in split-plot design

| Source <br> Whole Plot | SS | MS Num | DF Num | F Ratio | Prob > F |
| :--- | :---: | :---: | :---: | :---: | :---: | :---: |
| Rep |  |  |  |  |  |
| Str | $2.08 E+01$ | $2.31 E+00$ | 9 | 0.587863 | 0.8081 |
| PC | $6.55 E+04$ | $2.18 E+04$ | 3 | 5547.782 | 0.0000 |
| DDT | $1.22 E+01$ | $6.12 E+00$ | 2 | 1.554388 | 0.2115 |
| NoS | $1.22 E+01$ | $6.11 E+00$ | 2 | 1.551605 | 0.2121 |
| Sc | $5.29 E+01$ | $2.64 E+01$ | 2 | 6.719426 | 0.0012 |
|  | $2.00 E+01$ | $1.00 E+01$ | 2 | 2.546222 | 0.0786 |



| Str*PC*Sc*Alg | 1.48039 | 0.041121944 | 36 | 0.94621 | 0.5610 |
| :---: | :---: | :---: | :---: | :---: | :---: |
| Str*DDT*Sc*Alg | 1.89588 | 0.052663333 | 36 | 1.211775 | 0.1796 |
| $P C * D D T * S c * A l g$ | 1.31673 | 0.05486375 | 24 | 1.262407 | 0.1755 |
| Str*NoS*Sc*Alg | 1.8329 | 0.050913889 | 36 | 1.171521 | 0.2220 |
| $P C * N o S * S c * A l g$ | 1.16302 | 0.048459167 | 24 | 1.115038 | 0.3161 |
| $D D T * N o S * S c * A l g$ | 0.69388 | 0.028911667 | 24 | 0.665253 | 0.8891 |
| Str*PC*DDT*NoS*Alg | 3.31827 | 0.046087083 | 72 | 1.060457 | 0.3413 |
| Str ${ }^{*} P C * D D T * S c * A l g$ | 4.08111 | 0.056682083 | 72 | 1.304246 | 0.0434 |
| Str*PC*NoS*Sc*Alg | 3.15964 | 0.043883889 | 72 | 1.009762 | 0.4550 |
| Str*DDT*NoS*Sc*Alg | 3.90257 | 0.054202361 | 72 | 1.247188 | 0.0773 |
| $P C * D D T * N o S * S c * A l g$ | 1.98233 | 0.041298542 | 48 | 0.950273 | 0.5711 |
| Str*PC*DDT*NoS*Sc*Alg | 7.02248 | 0.048767222 | 144 | 1.122126 | 0.1524 |
| Subplot error | 380.18502 | 0.04345965 | 8748 |  |  |
| Total | 101289.9699 |  | 12959 |  |  |

The main plot in the above table shows that only the problem structure and number of shifts have significant effect on the computational time of the algorithm. The sub-plot shows that there is significant difference in the CT of the algorithms. The average time taken by the algorithm for different problem structures is shown in Table 33. The table shows that the time taken by NPN algorithms, ALG2 and ALG4, is significantly greater than the PN algorithms, ALG1 and ALG3, respectively. Since, the NPN sequence does not significantly improve the solution quality as compared to the PN sequence, it is advantageous to consider only PN sequence, given the increased CT required by NPN algorithms. Table 34 shows the average CT of algorithms for continuous and noncontinuous production. The result shows that the time taken by all algorithms is higher in the case of non-continuous production. This suggests that the limited machine availability constraint, i.e., non-continuous production, increases the complexity of the problem.

Table 33. CT of algorithms for different problem structure

|  | Computational time (sec) |  |  |  |
| :---: | :---: | :---: | :---: | :---: |
| Problem |  |  |  |  |
| Structure | ALG1 | ALG2 | ALG3 | ALG4 |
| LL | 57.29 | 173.68 | 810.27 | 2027.86 |
| LS | 1.94 | 11.50 | 19.68 | 87.58 |
| SL | 3.00 | 13.33 | 31.07 | 92.60 |
| SS | 0.13 | 0.78 | 1.11 | 4.43 |
| Overall | $\mathbf{1 5 . 5 9}$ | $\mathbf{4 9 . 8 2}$ | $\mathbf{2 1 5 . 5 3}$ | $\mathbf{5 5 3 . 1 2}$ |

Table 34. CT of algorithms for continuous and non-continuous production

|  | Computational time (sec) |  |  |  |
| :--- | :---: | :---: | :---: | :---: |
| Type | ALG1 | ALG2 | ALG3 | ALG4 |
| Non-continuous | 16.04 | 50.05 | 223.28 | 574.79 |
| Continuous | 14.69 | 49.37 | 200.05 | 509.77 |
| $\quad$ Overall | $\mathbf{1 5 . 5 9}$ | $\mathbf{4 9 . 8 2}$ | $\mathbf{2 1 5 . 5 3}$ | $\mathbf{5 5 3 . 1 2}$ |

## 9. CONCLUSIONS AND FUTURE RESEARCH

An uneven assembly flow shop scheduling problem with limited machine availability has been addressed in this research. Both permutation and non-permutation sequence has been considered to solve this problem. The setup time is considered to be machine and sequence-dependent, which implies that the setup time required for a job depends on the machine on which it is scheduled, and the previous job scheduled on that machine. The setup time between jobs belonging to the same product is less than the setup time between jobs belonging to different products. The machines have a dynamic availability time, which means that each machine may become available at a different time than the start of the planning horizon. Each job has a release time, due date and weight associated with it. The job release time is considered to be dynamic, i.e., jobs can be released at any time during the planning horizon. The due date of the job can be viewed as the shipment date and the weight of the job indicates the priority assigned to the job. Jobs with higher weights are prioritized over the lesser weighted jobs. The machines also have limited machine availability, i.e., the machines are not available continuously for the entire planning horizon. The production occurs in 8 -hour shifts and the number of shifts can be one, two or three. In the case of one and two shifts, the production occurs for 8 and 16 hours each day, respectively. This means that the machines are not available for production for the rest of the day. Furthermore, there is limited storage space between two machines. Thus, a blocking constraint is introduced, which means that a job is blocked on a predecessor machine if the storage space following that machine is not empty.

The goal is to simultaneously minimize two objectives, total setup time (representing producer's interest) and total weighted tardiness (representing customers' interest). Since the values of these two criteria might not be in the same range, a normalization technique is implemented so that the value of each criteria falls between 0 and 1 . The normalized criteria are then combined into a single objective function using the weights assigned to each criterion.

The problem is formulated as a mixed-integer linear programming model with the objective function focused on minimizing the linear combination of two normalized
objectives, setup time and weighted tardiness. Two models are developed, MILP1 for PN sequence and MILP2 for NPN sequence. Since the reduced version of the research problem is shown to be strongly NP-hard by previous works, the computational complexity of this problem is also strongly NP-hard. Thus, an exact method such as branch-and-bound technique can only be used to solve small problem instances. For medium and large problems, the branch-and-bound technique may never find an optimal solution even after spending an extremely large computational time.

Knowing the inefficiency of the branch-and-bound technique, a meta-heuristic algorithm is developed and applied to solve the problems. Two algorithms each, were developed for PN and NPN sequences, where one of the algorithms was based on shortterm tabu search and the other was based on tabu search/path-relinking. An important part of this research is to compare the ability of TS and TS/PR algorithms to find the best solution for different problems. The initial solution is generated by combining two sequences, each focusing on minimizing a single objective. The first sequence PS is focused on minimizing the setup time and is obtained using SST rule whereas the second sequence CS is focused on minimizing weighted tardiness and is obtained using WEDD rule. Normalized weights are used to combine both sequences together.

In order to assess the effectiveness of the developed algorithms, sixteen small-size problems were generated and solved using CPLEX for both, PN and NPN sequences. These problems were also solved using the meta-heuristic algorithms. For the 16 problems solved, it was observed that CPLEX could find an optimal solution for only 13 problems in the case of PN sequence and 9 problems in the case of NPN sequence. CPLEX was having difficulty in solving some of these problems to optimality, more so for NPN sequence. This was due to weak LP relaxation of the model and the existence of symmetry in the model. For the problems where an optimal solution was found by CPLEX, an average improvement of $2.68 \%$ was observed by adopting the NPN sequence over PN sequence. The optimal solutions were also compared with the solutions obtained from metaheuristic algorithms. The best solutions were obtained from TS/PR algorithms, ALG3 (for PN sequence) and ALG4 (for NPN sequence), which had an average deviation of $0.49 \%$ and
$0.13 \%$, respectively, from the optimal solution. This demonstrates the capability of the developed algorithms to identify high quality solutions. Moreover, the average CT of TS/PR algorithm ALG3 was 5.33 second, which is significantly lower than 4063.7 second, as required by CPLEX to solve for PN sequence. For NPN sequences, the CT of ALG4 was 60.09 seconds, as compared to 12408.35 seconds required by CPLEX. This supports the fact that the implementation of meta-heuristic algorithms is very time-efficient.

A multi-factor split-plot design is developed to analyze the significance in the algorithm's performance, both in terms of solution quality and computational time. Factors that define a problem, such as problem structure, plant capacity, due date tightness, number of shifts, scenario and replicates are placed in the main plot. The algorithm, which is the primary factor of interest, is placed in the sub-plot. The reason for this design is to analyze the effect of algorithms' performance without the influence of problem parameters. The result shows that, TS/PR outperforms short term TS for all problem structures, in the case of both, PN and NPN sequences. The best PN algorithm, i.e., ALG3, was compared with the best NPN algorithm, i.e. ALG4, to determine if the NPN sequence offers a significant advantage over PN sequence. An average improvement of $1.68 \%$ was observed by adopting the NPN sequence. The statistical analysis showed that the improvement offered by NPN sequence is not statistically significant for small-small, small-large and large-large problems. However, the improvement seemed to be significant in problems with high product variety, i.e., large-small problems. In addition, it was also observed that the performance of NPN sequence is better in the case of non-continuous production. For continuous production, NPN sequence did not yield any significant advantage. The results also show that the CT for PN algorithms is significantly lower for PN algorithms. Hence, it would be advantageous to consider only PN sequence for problems with small number of product types or with large number of jobs belonging to same products, given the higher efficiency and equivalent effectiveness of PN algorithms as compared to NPN algorithms. For large-small problems, NPN sequence is recommended. TS/PR algorithms (ALG3 and ALG4) are recommended for PN and NPN sequences, respectively, because of their ability to obtain superior solutions.

Future research could focus on adding complexity to the flow shop. The problem addressed in this research can be generalized into $m$-component assembly flowshop with each component requiring one or more operations before assembly stage. Assembly flow shops with $m$ components have been studied in the past by different researchers (Sung and Kim 2008, Torabzadeh and Zandieh, 2010, Al-Anzi and Allahverdi, 2012). However, the condition of multiple operations required by a component before assembly has not been studied so far. Machine skipping is a characteristic that is implemented in numerous manufacturing plants. Shahvari and Logendran (2017) have addressed a hybrid flow shop batch scheduling problem considering machine skipping. Thus, future research may consider generalizing the research problem into an $m$-machine flow shop and introducing machine skipping to the problem.

Further research could also focus on comparing the performance of the tabu searchbased algorithms with other heuristics such as genetic algorithm (GA) and particle swarm optimization (PSO) in solving the problem addressed in this research. Various researchers have compared the performance of TS-based heuristics with other heuristics in solving different types of scheduling problems (Al-Anzi and Allahverdi, 2012, Bozorgirad 2013, Shahvari 2016). The performance of these heuristics is shown to be different for different type of problems. Hence, more research insights can be obtained by implementing GA and PSO to solve the research problem and comparing their performance to the TS algorithms.

## BIBLIOGRAPHY

Aggoune, R., 2004. Minimizing the makespan for the flow shop scheduling problem with availability constraints. Eur. J. Oper. Res., EURO Young Scientists 153, 534-543.

Al-Anzi, F.S., Allahverdi, A., 2012. Better heuristics for a two-stage multi-machine assembly scheduling problem to minimize total completion time. Int. J. Oper. Res. 9, 66-75.

Al-Anzi, F.S., Allahverdi, A., 2006. A hybrid tabu search heuristic for the two-stage assembly scheduling problem. Int. J. Oper. Res. 3, 109-119.

Allahverdi, A., 2015. The third comprehensive survey on scheduling problems with setup times/costs. Eur. J. Oper. Res. 246, 345-378.

Allahverdi, A., Al-Anzi, F.S., 2009. The two-stage assembly scheduling problem to minimize total completion time with setup times. Comput. Oper. Res. 36, 27402747.

Allahverdi, A., Aldowaisan, T., 2004. No-wait flowshops with bicriteria of makespan and maximum lateness. Eur. J. Oper. Res. 152, 132-147.

Allahverdi, A., Gupta, J.N.D., Aldowaisan, T., 1999. A review of scheduling research involving setup considerations. Omega 27, 219-239.

Allahverdi, A., Ng, C.T., Cheng, T.C.E., Kovalyov, M.Y., 2008. A survey of scheduling problems with setup times or costs. Eur. J. Oper. Res. 187, 985-1032.

Allaoui, H., Artiba, A., Elmaghraby, S.E., Riane, F., 2006. Scheduling of a two-machine flowshop with availability constraints on the first machine. Int. J. Prod. Econ., Control and Management of Productive Systems 99, 16-27.

Aryal, A. and Logendran, R., 2017. Assembly Flowshop Scheduling with Shift Production. IIE Annual Conference Proceedings, Institute of Industrial and Systems Engineers (IISE).

Błażewicz, J., Breit, J., Formanowicz, P., Kubiak, W., Schmidt, G., 2001. Heuristic algorithms for the two-machine flowshop with limited machine availability. Omega 29, 599-608.

Bozorgirad, M.A., Logendran, R., 2013. Bi-criteria group scheduling in hybrid flowshops. Int. J. Prod. Econ. 145, 599-612.

Bozorgirad, M.A., 2013. Bi-criteria group scheduling with learning in hybrid flow shops (Unpublished doctoral dissertation). Oregon State University, Corvallis, Oregon.

Campbell, H.G., Dudek, R.A., Smith, M.L., 1970. A Heuristic Algorithm for the n Job, m Machine Sequencing Problem. Manag. Sci. 16, B630-B637.

Chyu, C.C., Chang, W.S., 2010. A competitive evolution strategy memetic algorithm for unrelated parallel machine scheduling to minimize total weighted tardiness and flow time, in: The 40th International Conference on Computers Indutrial Engineering. Presented at the The 40th International Conference on Computers Indutrial Engineering, pp. 1-6.

Eren, T., Güner, E., 2006. A bicriteria scheduling with sequence-dependent setup times. Appl. Math. Comput. 179, 378-385.

Framinan, J.M., Perez-Gonzalez, P., 2017. The 2-stage assembly flowshop scheduling problem with total completion time: Efficient constructive heuristic and metaheuristic. Comput. Oper. Res. 88, 237-246.

Gagné, C., Gravel, M., Price, W.L., 2005. Using metaheuristic compromise programming for the solution of multiple-objective scheduling problems. J. Oper. Res. Soc. 56, 687-698.

Glover, F., 1990. Tabu search-part II. ORSA J. Comput. 2, 4-32.
Glover, F., 1989. Tabu search-part I. ORSA J. Comput. 1, 190-206.
Glover, F., 1986. Future paths for integer programming and links to artificial intelligence. Comput. Oper. Res. 13, 533-549.

Graham, R.L., Lawler, E.L., Lenstra, J.K., Kan, A.H.G.R., 1979. Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey. Ann. Discrete Math., Discrete Optimization II 5, 287-326.

Hall, N.G., Sriskandarajah, C., 1996. A Survey of Machine Scheduling Problems with Blocking and No-Wait in Process. Oper. Res. 44, 510-525.

Hariri, A.M.A., Potts, C.N., 1997. A branch and bound algorithm for the two-stage assembly scheduling problem. Eur. J. Oper. Res. 103, 547-556.

Huo, Y., Huang, J.X., 2016. Parallel Ant Colony Optimization for Flow Shop Scheduling Subject to Limited Machine Availability, in: Parallel and Distributed Processing Symposium Workshops, 2016 IEEE International. IEEE, pp. 756-765.

Johnson, S.M., 1954. Optimal two- and three-stage production schedules with setup times included. Nav. Res. Logist. Q. 1, 61-68.

Kim, D.-W., Kim, K.-H., Jang, W., Chen, F.F., 2002. Unrelated parallel machine scheduling with setup times using simulated annealing. Robot. Comput.-Integr. Manuf. 18, 223-231.

Koulamas, C., 1994. The total tardiness problem: review and extensions. Oper. Res. 42, 1025-1041.

Koulamas, C., J. Kyparisis, G., 2001. The three-stage assembly flowshop scheduling problem. Comput. Oper. Res. 28, 689-704.

Kubiak, W., Błażewicz, J., Formanowicz, P., Breit, J., Schmidt, G., 2002. Two-machine flow shops with limited machine availability. Eur. J. Oper. Res. 136, 528-540.

Laguna, M., Barnes, J.W., Glover, F.W., 1991. Tabu search methods for a single machine scheduling problem. J. Intell. Manuf. 2, 63-73.

Lee, C.-Y., 1997. Minimizing the makespan in the two-machine flowshop scheduling problem with an availability constraint. Oper. Res. Lett. 20, 129-139.

Lee, C.-Y., Cheng, T.C.E., Lin, B.M.T., 1993. Minimizing the Makespan in the 3-Machine Assembly-Type Flowshop Scheduling Problem. Manag. Sci. 39, 616-625.

Liao, C.J., Chen, W.J., 2003. Single-machine scheduling with periodic maintenance and nonresumable jobs. Comput. Oper. Res. 30, 1335-1347.

Liao, L.-M., Huang, C.-J., 2010. Tabu search for non-permutation flowshop scheduling problem with minimizing total tardiness. Appl. Math. Comput. 217, 557-567.

Liu, B., Wang, L., Jin, Y.-H., 2008. An effective hybrid PSO-based algorithm for flow shop scheduling with limited buffers. Comput. Oper. Res., Part Special Issue: Bioinspired Methods in Combinatorial Optimization 35, 2791-2806.

Logendran, R., Subur, F., 2004. Unrelated parallel machine scheduling with job splitting. IIE Trans. 36, 359-372.

Lu, D., 2011. Bi-criteria group scheduling with sequence-dependent setup time in a flow shop (Unpublished master's thesis). Oregon State University, Corvallis, Oregon.

Ma, Y., Chu, C., Zuo, C., 2010. A survey of scheduling with deterministic machine availability constraints. Comput. Ind. Eng., Scheduling in Healthcare and Industrial Systems 58, 199-211.

Maleki-Darounkolaei, A., Modiri, M., Tavakkoli-Moghaddam, R., Seyyedi, I., 2012. A three-stage assembly flow shop scheduling problem with blocking and sequencedependent set up times. J. Ind. Eng. Int. 8, 26.

Mehravaran, Y., Logendran, R., 2012. Non-permutation flowshop scheduling in a supply chain with sequence-dependent setup times. Int. J. Prod. Econ., Green Manufacturing and Distribution in the Fashion and Apparel Industries 135, 953963.

Moslehi, G., Mahnam, M., 2011. A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search. Int. J. Prod. Econ. 129, 14-22.

Nawaz, M., Enscore Jr, E.E., Ham, I., 1983. A heuristic algorithm for the m-machine, njob flow-shop sequencing problem. Omega 11, 91-95.

Nowicki, E., Smutnicki, C., 1996. A fast tabu search algorithm for the permutation flowshop problem. Eur. J. Oper. Res. 91, 160-175.

Pandya, V. and R. Logendran (2010). Weighted tardiness minimization in flexible flowshops. IIE Annual Conference Proceedings, Institute of Industrial and Systems Engineers (IISE).

Panwalkar, S.S., Dudek, R.A., Smith, M.L., 1973. Sequencing Research and the Industrial Scheduling Problem, in: Symposium on the Theory of Scheduling and Its Applications. Springer, Berlin, Heidelberg, pp. 29-38.

Peng, B., Lü, Z., Cheng, T.C.E., 2015. A tabu search/path relinking algorithm to solve the job shop scheduling problem. Comput. Oper. Res. 53, 154-164.

Potts, C.N., Sevast'janov, S.V., Strusevich, V.A., Van Wassenhove, L.N., Zwaneveld, C.M., 1995. The Two-Stage Assembly Scheduling Problem: Complexity and Approximation. Oper. Res. 43, 346-355.

Qian, B., Wang, L., Huang, D., Wang, W., Wang, X., 2009. An effective hybrid DE-based algorithm for multi-objective flow shop scheduling with limited buffers. Comput. Oper. Res., Part Special Issue: Operations Research Approaches for Disaster Recovery Planning 36, 209-233.

Schaller, J.E., Gupta, J.N., Vakharia, A.J., 2000. Scheduling a flowline manufacturing cell with sequence dependent family setup times. Eur. J. Oper. Res. 125, 324-339.

Schiavinotto, T., Stützle, T., 2007. A review of metrics on permutations for search landscape analysis. Comput. Oper. Res. 34, 3143-3153.

Sevaux, M., Sörensen, K., 2005. Permutation distance measures for memetic algorithms with population management, in: Proceedings of 6th Metaheuristics International Conference (MIC'05). Citeseer.

Shahvari, O., Logendran, R., 2017. An enhanced tabu search algorithm to minimize a bicriteria objective in batching and scheduling problems on unrelated-parallel machines with desired lower bounds on batch sizes. Comput. Oper. Res. 77, 154176.

Shahvari, O., 2016. Bi-criteria batching and scheduling in hybrid flow shops (Unpublished doctoral dissertation). Oregon State University, Corvallis, Oregon.

Shahvari, O., Logendran, R., 2016. Hybrid flow shop batching and scheduling with a bicriteria objective. Int. J. Prod. Econ. 179, 239-258.

Shahvari, O., Salmasi, N., Logendran, R., Abbasi, B., 2012. An efficient tabu search algorithm for flexible flow shop sequence-dependent group scheduling problems. Int. J. Prod. Res. 50, 4237-4254.

Strusevich, V.A., Zwaneveld, C.M., 1994. On non-permutation solutions to some two machine flow shop scheduling problems. Math. Methods Oper. Res. 39, 305-319.

Taillard, E., 1990. Some efficient heuristic methods for the flow shop sequencing problem. Eur. J. Oper. Res. 47, 65-74.

Torabzadeh, E., Zandieh, M., 2010. Cloud theory-based simulated annealing approach for scheduling in the two-stage assembly flowshop. Adv. Eng. Softw. 41, 1238-1243.

Tozkapan, A., Kırca, Ö., Chung, C.-S., 2003. A branch and bound algorithm to minimize the total weighted flowtime for the two-stage assembly scheduling problem. Comput. Oper. Res. 30, 309-320.

Zeng, R.-Q., Basseur, M., Hao, J.-K., 2013. Solving bi-objective flow shop problem with hybrid path relinking algorithm. Appl. Soft Comput. 13, 4118-4132.

## APPENDIX

Appendix A. Result of statistical analysis for parameter tuning

Table A.1. Parameter value for algorithms

| Algorithm | Perturbation Type | Parameter | Small, Small | Small, <br> Large | Large, Small | Large, Large |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| ALG1 | P1 (Slight TS) | TLS | 10 | 20 | 25 | 5 |
|  |  | MNSS | 2 | 2 | 2 | 2 |
|  |  | MIWOI | 15 | 20 | 25 | 15 |
|  |  | MIL | 15 | 5 | 10 | 5 |
| ALG1 | P2 (Strong TS) | TLS | 20 | 5 | 20 | 5 |
|  |  | MNSS | 2 | 2 | 2 | 2 |
|  |  | MIWOI | 15 | 25 | 15 | 30 |
|  |  | MIL | 15 | 5 | 5 | 5 |
| ALG2 | P1 (Slight TS) | TLS | 20 | 20 | 25 | 5 |
|  |  | MNSS | 2 | 2 | 2 | 2 |
|  |  | MIWOI | 25 | 20 | 25 | 25 |
|  |  | MIL | 15 | 5 | 10 | 5 |
| ALG2 | P2 (Strong TS) | TLS | 5 | 5 | 5 | 5 |
|  |  | MNSS | 2 | 2 | 2 | 2 |
|  |  | MIWOI | 10 | 20 | 25 | 25 |
|  |  | MIL | 15 | 5 | 5 | 5 |
| ALG3 | - | P_size | 5 | 5 | 5 | 10 |
|  |  | MIWI | 3 | 3 | 3 | 8 |
| ALG4 | - | P_size | 5 | 5 | 10 | 10 |
|  |  | MIWI | 3 | 5 | 5 | 5 |

Table A.2. ANOVA for TLS, MNSS for ALG1 with P1 (Small-Small)

| Source <br> MAIN EFFECTS | Sum of Squares | Df | Mean Square | F-Ratio | P-Value |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| A:TLS |  |  |  |  |  |
| B:MNSS | 0.0312939 | 5 | 0.0063 | 10.18 | 0 |
| C:Block | 0.000347446 | 4 | $9 \mathrm{E}-05$ | 0.14 | 0.9668 |
| RESIDUAL | 8.27463 | 24 | 0.3448 | 560.75 | 0 |
| TOTAL (CORRECTED) | 0.440231 | 716 | 0.0006 |  |  |
| Al465 | 749 |  |  |  |  |

All F-ratios are based on the residual mean square error.

Table A.3. ANOVA for TLS, MNSS for ALG1 with P2 (Small-Small)

| Source <br> MAIN EFFECTS | Sum of Squares | Df | Mean Square | F-Ratio | P-Value |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| A:TLS |  |  |  |  |  |
| B:MNSS | 0.00980421 | 4 | 0.00245105 | 10.71 | 0 |
| C:Block | 0.00275777 | 5 | 0.000551554 | 2.41 | 0.0352 |
| RESIDUAL | 10.5788 | 24 | 0.440785 | 1925.32 | 0 |
| TOTAL (CORRECTED) | 0.163922 | 716 | 0.000228942 |  |  |
| Ald | 10.7553 | 749 |  |  |  |

All F-ratios are based on the residual mean square error.

Table A.4. ANOVA for TLS, MNSS for ALG2 with P1 (Small-Small)

| Source <br> MAIN EFFECTS | Sum of Squares | Df | Mean Square | F-Ratio | P-Value |  |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| A:TLS |  |  |  |  |  |  |
| B:MNSS | $0.74708 \mathrm{E}-05$ | 4 | $6.86771 \mathrm{E}-06$ | 0.04 | 0.9965 |  |
| C:Block | 9.18453 | 5 | 0.0029748 | 18.63 | 0 |  |
| RESIDUAL | 0.114299 | 24 | 0.382689 | 2397.26 | 0 |  |
| TOTAL (CORRECTED) | 9.31373 | 716 | 0.000159636 |  |  |  |
| All F-ratios are based on the residual mean square error. |  |  |  |  |  |  |

Table A.5. ANOVA for TLS, MNSS for ALG2 with P2 (Small-Small)

| Source <br> MAIN EFFECTS | Sum of Squares | Df | Mean Square | F-Ratio | P-Value |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| A:TLS |  |  |  |  |  |
| B:MNSS | $9.28307 \mathrm{E}-05$ | 5 | $1.85661 \mathrm{E}-05$ | 0.17 | 0.9741 |
| C:Block | 0.00126913 | 4 | 0.000317282 | 2.88 | 0.0219 |
| RESIDUAL | 9.41044 | 24 | 0.392102 | 3562.39 | 0 |
| TOTAL (CORRECTED) | 0.078808 | 716 | 0.000110067 |  |  |
| Ald | 749 |  |  |  |  |

All F-ratios are based on the residual mean square error.

Table A.6. ANOVA for TLS, MNSS for ALG1 with P1 (Small-Large)

| Source <br> MAIN EFFECTS | Sum of Squares | Df | Mean Square | F-Ratio | P-Value |
| :--- | :--- | :--- | :--- | :--- | :--- |
| A:TLS |  |  |  |  |  |
| B:MNSS | 0.01467 | 5 | 0.0029 | 16.7 | 0 |
| C:Block | $3.09542 \mathrm{E}-05$ | 4 | $8 \mathrm{E}-06$ | 0.04 | 0.9963 |
| RESIDUAL | 6.43633 | 24 | 0.2682 | 1526.9 | 0 |
| TOTAL (CORRECTED) | 0.125755 | 716 | 0.0002 |  |  |
| Ald | 749 |  |  |  |  |

All F-ratios are based on the residual mean square error.

Table A.7. ANOVA for TLS, MNSS for ALG1 with P2 (Small-Large)

| Source <br> MAIN EFFECTS | Sum of Squares | Df | Mean Square | F-Ratio | P-Value |
| :--- | :--- | :--- | :--- | :--- | :--- |
| A:TLS |  |  |  |  |  |
| B:MNSS | $4.43507 \mathrm{E}-05$ | 5 | $8.87014 \mathrm{E}-06$ | 0.29 | 0.9187 |
| C:Block | 0.00290975 | 4 | 0.000727438 | 23.77 | 0 |
| RESIDUAL | 5.71451 | 24 | 0.238105 | 7779.61 | 0 |
| TOTAL (CORRECTED) | 0.0219141 | 716 | $3.06062 \mathrm{E}-05$ |  |  |
| Al3938 | 749 |  |  |  |  |

All F-ratios are based on the residual mean square error.

Table A.8. ANOVA for TLS, MNSS for ALG2 with P1 (Small-Large)
Source MAIN EFFECTS

| A:TLS | 0.00194202 | 5 | 0.000388404 | 6.77 | 0 |
| :--- | :--- | :--- | :--- | :--- | :--- |
| B:MNSS | 0.00148445 | 4 | 0.000371113 | 6.47 | 0 |
| C:Block | 6.33012 | 24 | 0.263755 | 4595.13 | 0 |
| RESIDUAL | 0.0410975 | 716 | $5.73988 \mathrm{E}-05$ |  |  |
| TOTAL (CORRECTED) | 6.37464 | 749 |  |  |  |

All F-ratios are based on the residual mean square error.

Table A.9. ANOVA for TLS, MNSS for ALG2 with P2 (Small-Large)
Source
MAIN EFFECTS
A:TLS
Sum of Squares Df Mean Square F-Ratio P-Value
$6.22361 \mathrm{E}-05$
0.000895289
$5 \quad 1.24472 \mathrm{E}-05$
0.59
0.7062

B:MNSS
C:Block
RESIDUAL
5.5465
$\begin{array}{llll}4 & 0.000223822 & 10.64 & 0 \\ 24 & 0.231104 & 10989.26 & 0\end{array}$
$0.0150575 \quad 7160.00002103$
TOTAL (CORRECTED) 5.56252749
All F-ratios are based on the residual mean square error.

Table A.10. ANOVA for TLS, MNSS for ALG1 with P1 (Large-Small)

Source MAIN EFFECTS

| A:TLS | 0.0847708 | 5 | 0.017 | 65.95 | 0 |
| :--- | :--- | :--- | :--- | :--- | :--- |
| B:MNSS | 0.000109797 | 4 | $3 \mathrm{E}-05$ | 0.11 | 0.9802 |
| C:Block | 7.80212 | 24 | 0.3251 | 1264.6 | 0 |
| RESIDUAL | 0.184062 | 716 | 0.0003 |  |  |
| TOTAL (CORRECTED) | 8.07106 | 749 |  |  |  |

All F-ratios are based on the residual mean square error.

Table A.11. ANOVA for TLS, MNSS for ALG1 with P2 (Large-Small)

| Source | Sum of Squares | Df | Mean Square | F-Ratio | P-Value |
| :--- | :--- | :--- | :--- | :--- | :--- |
| MAIN EFFECTS |  |  |  |  |  |
| A:TLS | 0.000310561 | 5 | $6.21122 \mathrm{E}-05$ | 9 | 0 |
| B:MNSS | 0.00014966 | 4 | 0.000037415 | 5.42 | 0.0003 |
| C:Block | 9.17015 | 24 | 0.382089 | 55393.92 | 0 |
| RESIDUAL | 0.00493874 | 716 | $6.89768 \mathrm{E}-06$ |  |  |
| TOTAL (CORRECTED) | 9.17555 | 749 |  |  |  |

All F-ratios are based on the residual mean square error.

Table A.12. ANOVA for TLS, MNSS for ALG2 with P1 (Large-Small)

Source
MAIN EFFECTS

| A:TLS | 0.00484786 | 5 | 0.000969572 | 54.74 | 0 |
| :--- | :--- | :--- | :--- | :--- | :--- |
| B:MNSS | 0.000659532 | 4 | 0.000164883 | 9.31 | 0 |
| C:Block | 7.56644 | 24 | 0.315268 | 17798.91 | 0 |
| RESIDUAL | 0.0126824 | 716 | $1.77128 \mathrm{E}-05$ |  |  |
| TOTAL (CORRECTED) | 7.58463 | 749 |  |  |  |

All F-ratios are based on the residual mean square error.

Table A.13. ANOVA for TLS, MNSS for ALG2 with P2 (Large-Small)

| Source <br> MAIN EFFECTS | Sum of Squares | Df | Mean Square | F-Ratio | P-Value |
| :--- | :--- | :--- | :--- | :--- | :--- |
| A:TLS | $3.97248 \mathrm{E}-06$ | 5 | $7.94 \mathrm{E}-07$ | 0.25 | 0.941 |
| B:MNSS | 0.000324945 | 4 | $8.12363 \mathrm{E}-05$ | 25.33 | 0 |
| C:Block | 9.20308 | 24 | 0.383462 | 119544.4 | 0 |
| RESIDUAL | 0.00229671 | 716 | $3.20769 \mathrm{E}-06$ |  |  |
| TOTAL (CORRECTED) | 9.2057 | 749 |  |  |  |

All F-ratios are based on the residual mean square error.

Table A.14. ANOVA for TLS, MNSS for ALG1 with P1 (Large-Large)

| Source <br> MAIN EFFECTS | Sum of Squares | Df | Mean Square | F-Ratio | P-Value |
| :--- | :--- | :--- | :--- | :--- | :--- |
| A:TLS |  |  |  |  |  |
| B:MNSS | $4.47414 \mathrm{E}-05$ | 5 | $8.94828 \mathrm{E}-06$ | 1.49 | 0.1901 |
| C:Block | 0.000206549 | 4 | $5.16372 \mathrm{E}-05$ | 8.61 | 0 |
| RESIDUAL | 4.14482 | 24 | 0.172701 | 28799 | 0 |
| TOTAL (CORRECTED) | 0.00429374 | 716 | $5.99685 \mathrm{E}-06$ |  |  |
| Ald | 749 |  |  |  |  |

All F-ratios are based on the residual mean square error.

Table A.15. ANOVA for TLS, MNSS for ALG1 with P2 (Large-Large)

| Source <br> MAIN EFFECTS | Sum of Squares | Df | Mean Square | F-Ratio | P-Value |
| :--- | :--- | :--- | :--- | :--- | :--- |
| A:TLS | $7.62277 \mathrm{E}-05$ | 5 | $1.52455 \mathrm{E}-05$ | 1.07 | 0.3772 |
| B:MNSS | 0.00160272 | 4 | 0.000400679 | 28.05 | 0 |
| C:Block | 5.33893 | 24 | 0.222455 | 15574.24 | 0 |
| RESIDUAL | 0.010227 | 716 | $1.42835 \mathrm{E}-05$ |  |  |
| TOTAL (CORRECTED) | 5.35083 | 749 |  |  |  |

All F-ratios are based on the residual mean square error.

Table A.16. ANOVA for TLS, MNSS for ALG2 with P1 (Large-Large)

| Source <br> MAIN EFFECTS | Sum of Squares | Df | Mean Square | F-Ratio | P-Value |  |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| A:TLS |  | $1.94033 \mathrm{E}-05$ | 5 | $3.88065 \mathrm{E}-06$ | 1.17 | 0.321 |
| B:MNSS | 0.000182414 | 4 | $4.56035 \mathrm{E}-05$ | 13.78 | 0 |  |
| C:Block | 8.73951 | 24 | 0.364146 | 110016 | 0 |  |
| RESIDUAL | 0.00236992 | 716 | $3.30994 \mathrm{E}-06$ |  |  |  |
| TOTAL (CORRECTED) | 8.74208 | 749 |  |  |  |  |
| All F-ratios are based on the residual mean square error. |  |  |  |  |  |  |

Table A.17. ANOVA for TLS, MNSS for ALG2 with P2 (Large-Large)
Source
MAIN EFFECTS

| A:TLS | $6.43 \mathrm{E}-08$ | 5 | $1.29 \mathrm{E}-08$ | 0.03 | 0.9996 |
| :--- | :--- | :--- | :--- | :--- | :--- |
| B:MNSS | $3.70061 \mathrm{E}-05$ | 4 | $9.25152 \mathrm{E}-06$ | 20.72 | 0 |
| C:Block | 11.5258 | 24 | 0.480241 | 1075505 | 0 |
| RESIDUAL | 0.000319712 | 716 | $4.47 \mathrm{E}-07$ |  |  |
| TOTAL (CORRECTED) | 11.5261 | 749 |  |  |  |

All F-ratios are based on the residual mean square error.

Table A.18. ANOVA for MIWOI, ILS for ALG1 with P1 (Small-Small)

| Source <br> MAIN EFFECTS | Sum of Squares | Df | Mean Square | F-Ratio | P-Value |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| A:MIWOI |  |  |  |  |  |
| B:ILS | 0.20179 | 5 | 0.0404 | 60.29 | 0 |
| C:Block | 0.0313526 | 5 | 0.0063 | 9.37 | 0 |
| RESIDUAL | 11.6003 | 24 | 0.4833 | 722.05 | 0 |
| TOTAL (CORRECTED) | 0.579038 | 865 | 0.0007 |  |  |
| All | 12.4125 | 899 |  |  |  |

All F-ratios are based on the residual mean square error.

Table A.19. ANOVA for MIWOI, ILS for ALG1 with P2 (Small-Small)

| Source <br> MAIN EFFECTS | Sum of Squares | Df | Mean Square | F-Ratio | P-Value |
| :--- | :--- | :--- | :--- | :--- | :--- |
| A:MIWOI |  |  |  |  |  |
| B:ILS | 0.0949282 | 5 | 0.0189856 | 56.2 | 0 |
| C:Block | $4.97 \mathrm{E}-03$ | 5 | $9.95 \mathrm{E}-04$ | 2.94 | 0.0121 |
| RESIDUAL | 12.7962 | 24 | 0.533177 | 1578.17 | 0 |
| TOTAL (CORRECTED) | 0.292235 | 865 | 0.000337844 |  |  |
| Ald | 13.1884 | 899 |  |  |  |

All F-ratios are based on the residual mean square error.

Table A.20. ANOVA for MIWOI, ILS for ALG2 with P1 (Small-Small)

| Source <br> MAIN EFFECTS | Sum of Squares | Df | Mean Square | F-Ratio | P-Value |  |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| A:MIWOI |  |  |  |  |  |  |
| B:ILS | 0.0439826 | 5 | 0.00879652 | 34.08 | 0 |  |
| C:Block | 0.0103749 | 5 | 0.00207498 | 8.04 | 0 |  |
| RESIDUAL | 10.3522 | 24 | 0.431343 | 1670.97 | 0 |  |
| TOTAL (CORRECTED) | 0.22329 | 10.6299 | 865 | 0.000258139 |  |  |
| All F-ratios are based on the residual mean square error. |  |  |  |  |  |  |

Table A.21. ANOVA for MIWOI, ILS for ALG2 with P2 (Small-Small)
Source
MAIN EFFECTS

| A:MIWOI | 0.0144544 | 5 | 0.00289088 | 22.86 | 0 |
| :--- | :--- | :--- | :--- | :--- | :--- |
| B:ILS | 0.00362204 | 5 | 0.000724409 | 5.73 | 0 |
| C:Block | 10.7734 | 24 | 0.44889 | 3550.2 | 0 |
| RESIDUAL | 0.109371 | 865 | 0.000126441 |  |  |
| TOTAL (CORRECTED) | 10.9008 | 899 |  |  |  |

All F-ratios are based on the residual mean square error.

Table A.22. ANOVA for MIWOI, ILS for ALG1 with P1 (Small-Large)

| Source <br> MAIN EFFECTS | Sum of Squares | Df | Mean Square | F-Ratio | P-Value |
| :--- | :--- | :--- | :--- | :--- | :--- |
| A:MIWOI |  |  |  |  |  |
| B:ILS | 0.0164934 | 5 | 0.00329868 | 27.44 | 0 |
| C:Block | $4.26501 \mathrm{E}-05$ | 5 | $8.53002 \mathrm{E}-06$ | 0.07 | 0.9965 |
| RESIDUAL | 7.68708 | 24 | 0.320295 | 2664.3 | 0 |
| TOTAL (CORRECTED) | 0.103987 | 865 | 0.000120216 |  |  |
| All F-ratios are based on the residual mean square error. |  |  |  |  |  |

Table A.23. ANOVA for MIWOI, ILS for ALG1 with P2 (Small-Large)

| Source <br> MAIN EFFECTS | Sum of Squares | Df | Mean Square | F-Ratio | P-Value |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| A:MIWOI |  |  |  |  |  |
| B:ILS | 0.00233216 | 5 | 0.000466432 | 15.51 | 0 |
| C:Block | $4.37 \mathrm{E}-05$ | 5 | $8.74 \mathrm{E}-06$ | 0.29 | 0.9182 |
| RESIDUAL | 6.63228 | 24 | 0.276345 | 9190.33 | 0 |
| TOTAL (CORRECTED) | 0.0260098 | 865 | $3.00691 \mathrm{E}-05$ |  |  |
| All | 899 |  |  |  |  |

All F-ratios are based on the residual mean square error.

Table A.24. ANOVA for MIWOI, ILS for ALG2 with P1 (Small-Large)

Source
MAIN EFFECTS

| A:MIWOI | 0.00914565 | 5 | 0.00182913 | 32.02 | 0 |
| :--- | :--- | :--- | :--- | :--- | :--- |
| B:ILS | 0.000053013 | 5 | $1.06026 \mathrm{E}-05$ | 0.19 | 0.9681 |
| C:Block | 7.54207 | 24 | 0.314253 | 5500.76 | 0 |
| RESIDUAL | 0.0494166 | 865 | 0.000057129 |  |  |
| TOTAL (CORRECTED) | 7.60069 | 899 |  |  |  |

Sum of Squares Df Mean Square F-Ratio P-Value
$\begin{array}{lllll}0.00914565 & 5 & 0.00182913 & 32.02 & 0\end{array}$
$\begin{array}{lllll}0.000053013 & 5 & 1.06026 \mathrm{E}-05 & 0.19 & 0.9681\end{array}$
$\begin{array}{llllll}7.54207 & 24 & 0.314253 & 5500.76 & 0\end{array}$
$0.0494166 \quad 865 \quad 0.000057129$
TOTAL (CORRECTED) 7.60069
899

All F-ratios are based on the residual mean square error.

Table A.25. ANOVA for MIWOI, ILS for ALG2 with P2 (Small-Large)

| Source <br> MAIN EFFECTS | Sum of Squares | Df | Mean Square | F-Ratio | P-Value |
| :--- | :--- | :--- | :--- | :--- | :--- |
| A:MIWOI | 0.000570462 | 5 | 0.000114092 | 10.82 | 0 |
| B:ILS | $9.68275 \mathrm{E}-05$ | 5 | $1.93655 \mathrm{E}-05$ | 1.84 | 0.1032 |
| C:Block | 6.76487 | 24 | 0.28187 | 26734.61 | 0 |
| RESIDUAL | 0.00911991 | 865 | $1.05433 \mathrm{E}-05$ |  |  |
| TOTAL (CORRECTED) | 6.77466 | 899 |  |  |  |

All F-ratios are based on the residual mean square error.

Table A.26. ANOVA for MIWOI, ILS for ALG1 with P1 (Large-Small)

| Source <br> MAIN EFFECTS | Sum of Squares | Df | Mean Square | F-Ratio | P-Value |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| A:MIWOI |  |  |  |  |  |
| B:ILS | 0.0957728 | 5 | 0.0192 | 95.57 | 0 |
| C:Block | 0.00452182 | 5 | 0.0009 | 4.51 | 0.0005 |
| RESIDUAL | 9.01436 | 24 | 0.3756 | 1873.9 | 0 |
| TOTAL (CORRECTED) | 0.173374 | 865 | 0.0002 |  |  |
| Ald | 899 |  |  |  |  |

All F-ratios are based on the residual mean square error.

Table A.27. ANOVA for MIWOI, ILS for ALG1 with P2 (Large-Small)

| Source <br> MAIN EFFECTS | Sum of Squares | Df | Mean Square | F-Ratio | P-Value |
| :--- | :--- | :--- | :--- | :--- | :--- |
| A:MIWOI |  |  |  |  |  |
| B:ILS | 0.00110015 | 5 | 0.000220031 | 37.42 | 0 |
| C:Block | $5.56 \mathrm{E}-06$ | 5 | $1.11 \mathrm{E}-06$ | 0.19 | 0.9667 |
| RESIDUAL | 10.6846 | 24 | 0.445193 | 75704.18 | 0 |
| TOTAL (CORRECTED) | 0.0050868 | 865 | $5.88069 \mathrm{E}-06$ |  |  |
| All F-ratios are based on the residual mean square error. |  |  |  |  |  |

Table A.28. ANOVA for MIWOI, ILS for ALG2 with P1 (Large-Small)

| Source <br> MAIN EFFECTS | Sum of Squares | Df | Mean Square | F-Ratio | P-Value |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| A:MIWOI |  |  |  |  |  |
| B:ILS | 0.00581828 | 5 | 0.00116366 | 76.6 | 0 |
| C:Block | 0.00075769 | 5 | 0.000151538 | 9.98 | 0 |
| RESIDUAL | 9.17734 | 24 | 0.382389 | 25171.97 | 0 |
| TOTAL (CORRECTED) | 0.0131403 | 865 | $1.51911 \mathrm{E}-05$ |  |  |
| All F-ratios are based on the residual mean square error. |  |  |  |  |  |

Table A.29. ANOVA for MIWOI, ILS for ALG2 with P2 (Large-Small)

| Source <br> MAIN EFFECTS | Sum of Squares | Df | Mean Square | F-Ratio | P-Value |
| :--- | :--- | :--- | :--- | :--- | :--- |
| A:MIWOI | 0.000142175 | 5 | $2.84351 \mathrm{E}-05$ | 13.87 | 0 |
| B:ILS | $3.27669 \mathrm{E}-06$ | 5 | $6.55 \mathrm{E}-07$ | 0.32 | 0.9013 |
| C:Block | 10.3146 | 24 | 0.429776 | 209593 | 0 |
| RESIDUAL | 0.0017737 | 865 | $2.05052 \mathrm{E}-06$ |  |  |
| TOTAL (CORRECTED) | 10.3165 | 899 |  |  |  |

All F-ratios are based on the residual mean square error.

Table A.30. ANOVA for MIWOI, ILS for ALG1 with P1 (Large-Large)

| Source <br> MAIN EFFECTS | Sum of Squares | Df | Mean Square | F-Ratio | P-Value |
| :--- | :--- | :--- | :--- | :--- | :--- |
| A:MIWOI |  |  |  |  |  |
| B:ILS | 0.000243106 | 5 | $5 \mathrm{E}-05$ | 12.45 | 0 |
| C:Block | $1.24541 \mathrm{E}-05$ | 5 | $2 \mathrm{E}-06$ | 0.64 | 0.6708 |
| RESIDUAL | 4.93013 | 24 | 0.2054 | 52613 | 0 |
| TOTAL (CORRECTED) | 0.00337729 | 4.93377 | 865 | $4 \mathrm{E}-06$ |  |
| All F-ratios are based on the residual mean square error. |  |  |  |  |  |
| R |  |  |  |  |  |

Table A.31. ANOVA for MIWOI, ILS for ALG1 with P2 (Large-Large)

| Source <br> MAIN EFFECTS | Sum of Squares | Df | Mean Square | F-Ratio | P-Value |
| :--- | :--- | :--- | :--- | :--- | :--- |
| A:MIWOI | 0.000769404 | 5 | 0.000153881 | 14.11 | 0 |
| B:ILS | $7.61 \mathrm{E}-06$ | 5 | $1.52 \mathrm{E}-06$ | 0.14 | 0.9831 |
| C:Block | 6.43306 | 24 | 0.268044 | 24574.81 | 0 |
| RESIDUAL | 0.00943479 | 865 | $1.09073 \mathrm{E}-05$ |  |  |
| TOTAL (CORRECTED) | 6.44327 | 899 |  |  |  |

All F-ratios are based on the residual mean square error.

Table A.32. ANOVA for MIWOI, ILS for ALG2 with P1 (Large-Large)

| Source <br> MAIN EFFECTS | Sum of Squares | Df | Mean Square | F-Ratio | P-Value |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| A:MIWOI |  |  |  |  |  |
| B:ILS | $1.65098 \mathrm{E}-05$ | 5 | 0.000013302 | 14.07 | 0 |
| C:Block | $4.89989 \mathrm{E}-06$ | 5 | $3.80 \mathrm{E}-07$ | 0.4 | 0.8476 |
| RESIDUAL | 24 | 0.206568 | 218534.3 | 0 |  |
| TOTAL (CORRECTED) | 4.000817634 | 865 | $9.45 \mathrm{E}-07$ |  |  |
| All F-ratios are based on the residual mean square error. |  |  |  |  |  |

Table A.33. ANOVA for MIWOI, ILS for ALG2 with P2 (Large-Large)

| Source <br> MAIN EFFECTS | Sum of Squares | Df | Mean Square | F-Ratio | P-Value |
| :--- | :--- | :--- | :--- | :--- | :--- |
| A:MIWOI | 0.000084788 | 5 | $1.69576 \mathrm{E}-05$ | 25.65 | 0 |
| B:ILS | $8.68 \mathrm{E}-07$ | 5 | $1.74 \mathrm{E}-07$ | 0.26 | 0.9334 |
| C:Block | 8.32251 | 24 | 0.346771 | 524562.6 | 0 |
| RESIDUAL | 0.000571824 | 865 | $6.61 \mathrm{E}-07$ |  |  |
| TOTAL (CORRECTED) | 8.32317 | 899 |  |  |  |

All F-ratios are based on the residual mean square error.

Table A.34. ANOVA for MIWI, P_size for ALG3 (Small-Small)

| Source <br> MAIN EFFECTS | Sum of Squares | Df | Mean Square | F-Ratio | P-Value |
| :--- | :--- | :--- | :--- | :--- | :--- |
| A:MIWI |  |  |  |  |  |
| B:P_size | 0.000363968 | 3 | 0.000121323 | 1.39 | 0.2461 |
| C:Block | $8.67222 \mathrm{E}-05$ | 3 | $2.89074 \mathrm{E}-05$ | 0.33 | 0.8031 |
| RESIDUAL | 6.27495 | 24 | 0.261456 | 2991.27 | 0 |
| TOTAL (CORRECTED) | 0.0322529 | 369 | $8.74063 \mathrm{E}-05$ |  |  |
| All F-ratios are based on the residual mean square error. |  |  |  |  |  |

Table A.35. ANOVA for MIWI, P_size for ALG4 (Small-Small)

| Source <br> MAIN EFFECTS | Sum of Squares | Df | Mean Square | F-Ratio | P-Value |
| :--- | :--- | :--- | :--- | :--- | :--- |
| A:MIWI | 0.000251505 | 3 | $8.38349 \mathrm{E}-05$ | 1.4 | 0.2429 |
| B:P_size | 0.000328296 | 3 | 0.000109432 | 1.83 | 0.142 |
| C:Block | 6.21947 | 24 | 0.259145 | 4323.56 | 0 |
| RESIDUAL | 0.022117 | 369 | $5.99377 \mathrm{E}-05$ |  |  |
| TOTAL (CORRECTED) | 6.24217 | 399 |  |  |  |

All F-ratios are based on the residual mean square error.

Table A.36. ANOVA for MIWI, P_size for ALG3 (Small-Large)
Source Sum of Squares Df Mean Square F-Ratio P-Value MAIN EFFECTS

| A:MIWI | 0.00018301 | 3 | $6.10032 \mathrm{E}-05$ | 1.38 | 0.2475 |
| :--- | :--- | :--- | :--- | :--- | :--- |
| B:P_size | 0.000066382 | 3 | $2.21273 \mathrm{E}-05$ | 0.5 | 0.6812 |
| C:Block | 2.76549 | 24 | 0.115229 | 2613.29 | 0 |
| RESIDUAL | 0.0162704 | 369 | $4.40933 \mathrm{E}-05$ |  |  |
| TOTAL (CORRECTED) | 2.78201 | 399 |  |  |  |

All F-ratios are based on the residual mean square error.

Table A.37. ANOVA for MIWI, P_size for ALG4 (Small-Large)

| Source <br> MAIN EFFECTS | Sum of Squares | Df | Mean Square | F-Ratio | P-Value |
| :--- | :--- | :--- | :--- | :--- | :--- |
| A:MIWI | 0.000155231 | 3 | $5.17438 \mathrm{E}-05$ | 3.05 | 0.0287 |
| B:P_size | $6.42444 \mathrm{E}-05$ | 3 | $2.14148 \mathrm{E}-05$ | 1.26 | 0.2872 |
| C:Block | 2.64952 | 24 | 0.110396 | 6505.61 | 0 |
| RESIDUAL | 0.00626172 | 369 | $1.69694 \mathrm{E}-05$ |  |  |
| TOTAL (CORRECTED) | 2.656 | 399 |  |  |  |

All F-ratios are based on the residual mean square error.

Table A.38. ANOVA MIWI, P_size for ALG3 (Large-Small)

Source
MAIN EFFECTS

| A:MIWI | 0.000155534 | 3 | $5.18447 \mathrm{E}-05$ | 0.69 | 0.5601 |
| :--- | :--- | :--- | :--- | :--- | :--- |
| B:P_size | $4.29849 \mathrm{E}-05$ | 3 | $1.43283 \mathrm{E}-05$ | 0.19 | 0.9032 |
| C:Block | 4.64208 | 24 | 0.19342 | 2564.99 | 0 |
| RESIDUAL | 0.0278254 | 369 | $7.54076 \mathrm{E}-05$ |  |  |
| TOTAL (CORRECTED) | 4.6701 | 399 |  |  |  |

Table A.39. ANOVA for MIWI, P_size for ALG4 (Large-Small)

| Source <br> MAIN EFFECTS | Sum of Squares | Df | Mean Square | F-Ratio | P-Value |
| :--- | :--- | :--- | :--- | :--- | :--- |
| A:MIWI | 0.000329974 | 3 | 0.000109991 | 4.53 | 0.0039 |
| B:P_size | 0.000284724 | 3 | $9.49079 \mathrm{E}-05$ | 3.91 | 0.009 |
| C:Block | 4.62333 | 24 | 0.192639 | 7938.48 | 0 |
| RESIDUAL | 0.00895432 | 369 | $2.42665 \mathrm{E}-05$ |  |  |
| TOTAL (CORRECTED) | 4.6329 | 399 |  |  |  |

All F-ratios are based on the residual mean square error.

Table A.40. ANOVA for MIWI, P_size for ALG3 (Large-Large)

| Source <br> MAIN EFFECTS | Sum of Squares | Df | Mean Square | F-Ratio | P-Value |
| :--- | :--- | :--- | :--- | :--- | :--- |
| A:MIWI | 0.00225633 | 3 | 0.000752109 | 4.55 | 0.0038 |
| B:P_size | 0.00471393 | 3 | 0.00157131 | 9.51 | 0 |
| C:Block | 3.1204 | 24 | 0.130017 | 787.14 | 0 |
| RESIDUAL | 0.0609498 | 369 | 0.000165176 |  |  |
| TOTAL (CORRECTED) | 3.18832 | 399 |  |  |  |

All F-ratios are based on the residual mean square error.

Table A.41. ANOVA for MIWI, P_size for ALG4 (Large-Large)

| Source <br> MAIN EFFECTS | Sum of Squares Df | Mean Square | F-Ratio | P-Value |  |
| :--- | :--- | :--- | :--- | :--- | :--- |
| A:MIWI | 0.00038224 | 3 | 0.000127413 | 7.245106 | 0 |
| B:P_size | 0.00032894 | 3 | 0.000109647 | 6.23484 | 0.000386812 |
| C:Block | 4.15967 | 24 | 0.173319583 | 9855.473 | 0 |
| RESIDUAL | 0.00648928 | 369 | $1.75861 \mathrm{E}-05$ |  |  |
| TOTAL (CORRECTED) | 4.16687046 | 399 |  |  |  |

All F-ratios are based on the residual mean square error.

