DEVELOPMENT OF HEURISTICS FOR SINGLE PROCESSOR SCHEDULING PROBLEMS WITH TARDINESS - RELATED PERFORMANCE MEASURES

BY

SAHEED AKANDE

MATRIC. NO.: 153264

B.Sc.(Hons) Metallurgical and Materials Engineering (Ife)

M.Sc. Industrial and Production Engineering (Ibadan)

MNSE, MNIIE, Regd. Engr.

A Thesis written in the Department of

INDUSTRIAL AND PRODUCTION ENGINEERING

Submitted to the Faculty of Technology

in partial fulfilment of the requirements for the Degree of

DOCTOR OF PHILOSOPHY

of the

UNIVERSITY OF IBADAN.

JUNE, 2017

ABSTRACT

Scheduling problems involve the processing of jobs on processor(s) subject to constraints to optimise performance measures. Determining schedules that minimise tardiness-related performance measures for single processor scheduling problems is complex. Optimal-solutions require prohibitive-time, therefore heuristics are explored in real-life but effectiveness of existing heuristics is inadequate. The objective of this study was to develop effective heuristics for minimizing tardiness-related performance measures for single processor scheduling problems.

Six heuristics were developed and compared to the existing heuristics for minimizing the Total-Tardiness of jobs with Release-Dates (2TRD), Total-Tardiness and Total-Flowtime of jobs with Zero-Release-Dates (3TFZRD) and Total-Tardiness and Total-Flowtime of jobs with Release-Dates (3TFRD). HeuristicI (HeuI) and HeuristicII (HeuII) developed for 2TRD problem were compared to Dynamic Modified Due Date (DMDD). For 3TFZRD problem, heuristicIII (HeuIII) and heuristicIV (HeuIV) developed were compared to heuristicA (HeuA). Generalised Algorithm (GAlg) was compared to heuristicV (HeuV) and heuristicVI (HeuVI) developed for 3TFRD. HeuI and HeuII schedule jobs to reduce lateness. HeuIII and HeuIV reduce waiting-time. HeuV and HeuIV reduce idle-time and waiting-time. Fifty instances each, for problem-sizes (n) from small-sized ($5 \leq n \leq 10$), medium-sized ($10 < n < 30$), and large-sized ($30 \leq n \leq 1000$) randomly generated and industry-based problems involving 5, 10 and 15 jobs were solved in editor module in MATLAB environment. The Branch and Bound (BB) optimal method was used to solve small-and medium-sized problems. For 3TFZRD and 3TFRD problems, a composite-function was formulated and normalised. The Approximation Ratio (AR) test evaluated the heuristics' effectiveness, significant differences were analysed using t-test at $\alpha_{0.05}$.

The small-sized problems objective-function were: BB (9.26±9.12), HeuI (16.32±17.38), HeuII (12.03±10.02), DMDD (20.02±15.21) for 2TRD; BB (0.3059±0.0190), HeuIII (0.3059±0.0190), HeuIV (0.3192±0.0010), HeuA (0.3191±0.0200) for 3TFZRD; BB (0.1664±0.0043), HeuV (0.2350±0.0090), HeuVI (0.2096±0.0094), GAlg (0.3875±0.0380) for 3TFRD. HeuII, HeuIII and HeuVI yielded the AR of 1.85±1.22, 1.00±0.00 and 1.26±0.09 compared to DMDD (4.42±4.33), HeuA

(1.05±0.013) and GAlg (2.33±0.176), respectively. The medium-sized problems objective-function were: BB (287.11±138.54), HeuI (442.15±197.36), HeuII (432.48±137.86), DMDD (449.03±135.54) for 2TRD; BB (0.3397±0.0047), HeuIII (0.3398±0.0201), HeuIV (0.3403±0.0204), HeuA (0.3629±0.0239) for 3TFZRD; BB (0.2410±0.0550), HeuV (0.3395±0.0520), HeuVI (0.3275±0.0530), GAlg (0.6191±0.0440) for 3TFRD. Considering the small-and-medium-sized problems, HeuII and HeuIII were not significantly different from the optimal. The large-sized problems objective-function were: HeuI (254,046±8215), HeuII (311,184±11,643), DMDD (303,044±8642) for 2TRD; HeuIII (0.3492±0.0024), HeuIV (0.3493±0.0024), HeuA (0.3736 ±0.0029) for 3TFZRD; HeuV (0.5511±0.0670), HeuVI (0.5473±0.0690), GAlg (0.7589±0.0960) for 3TFRD. Therefore, regarding 2TRD problem, HeuII was recommended when solving small-and-medium-sized problems and HeuI for large-sized. Considering 3TFZRD and 3TFRD problems, HeuIII and HeuVI were respectively recommended for all the problem sizes. With respect to the industry-based problems, HeuII, HeuIII and HeuVI were (44.3±36.9%), (182.7±113.8%) and (39.0±25.4%), respectively better than firm policies; First-Come-First-Served for 2TRD and 3TFZRD, Shortest-Processing-Time for 3TFRD.

The developed heuristics minimised tardiness-related performance measures for single processor scheduling problems and were more effective compared to the existing ones.

Keywords: Single Processor Scheduling, Heuristic, Effectiveness, Total-Tardiness, Total-Flowtime

Word count: 458

ACKNOWLEDGEMENT

All thanks and adoration is due to Almighty GOD for making this possible because I could not have done it with my own efforts. He perfected the route to the success of this work by connecting me to good people for assistance. In this regards, I am indebted to many people in connection with this research. First of all, I wish to express my gratitude to my supervisor, Professor Ayodeji Emmanuel Oluleye. His direction and suggestions have been insightful and I could not have completed this research and thesis without his support, guidance, humour and general encouragement. In fact, no word can match his contribution to the success of this work. Furthermore, the support, encouragement and the prayer of Professor A.A Aderoba towards this work serve as the catalyst for the completion of the work and my thanks will always be unlimited. Moreover, this research work would not have been accomplished without an enabling environment. This environment in the path of career fulfilment was provided by the founder of Afe Babalola University Ado Ekiti, Aare Afe Babalola. I will always remain grateful to you sir for the “fruit of life you sow upon which unlimited souls are being feed”.

The implementation of the devised models required the knowledge of programing language. Therefore, the contribution of Engr Musa Muhamma and Dr. Abdulwahab Giwa for not just “given me a fish to eat but showed me the way of fishing” will always remain in my heart. The use of MATrix LABoratory (MATLAB) in this research work remains one of my pride as it makes the work to be more unique. Furthermore, the constructive criticism and support of Professor O.E Charles-Owaba, Dr. V.O Oladokun, Dr. C.O Anyaeche, Dr. Adeyeye, Dr. Akanbi, Engr. Tunde Odedairo are greatly appreciated. Also, I deeply appreciate the support of the managing director of VAL-VAL auto clinic Engineering limited located at Ibadan, Oyo state for his support particularly in providing the real life data to demonstrate the utility of the proposed models. The support of my students especially Adewuyi Crystal Adeola, Bashir Olawoyin, Obisesan Ismat as well as Kelvin Ahwin is also appreciated.

Finally, I must thank my mother, my wife and my siblings for their unlimited supports, encouragement and prayer towards the success of this work.

CERTIFICATION

This is to certify that Saheed Akande carried out this project under my supervision in partial fulfilment of the requirements of a Doctor of Philosophy degree in the Department of Industrial and Production Engineering, University of Ibadan

…………………………………………………………………………

Supervisor's Signature

Prof. A. E. Oluleye

B.Sc. (Ibadan), M.Sc. (Cranfield), Ph.D. (Ibadan), MNSE, Reg. Engr.

Dept. of Industrial and Production Engineering

University of Ibadan, Ibadan.

Nigeria

DEDICATION

This work is dedicated to the memory of my late father, Mr Tajudeen Akande; who provided a beacon for my life by the qualities he exhibited and the principles he embraced: Prayer, fear of GOD, love of family and dedication towards any project he embarked on.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS AND ACRONYMS

Abbreviation	Meaning
BB	Branch and Bound

EDD	Earliest Due Date
SPT	Shortest Processing Time
SPTTP	Single Processor Total Tardiness Problems
MDD	Modified Due Date
DMDD	Dynamic Modified Due Date
MST	Minimum Slack Time
ODD	Operation Due Date
SCR	Shortest Critical Ratio
LCOF	Linear Composite Objective Function
GUI	Graphical User Interface
NP	Non-Polynomial
AR	Approximation Ratio
OC	Optimal Count

Acronyms

GAlg	Generalized Algorithm
HeuA	Heuristic A
HeuB	Heuristic B
P	Polynomial
MATLAB	MATrix LABoratory
2TRD	Total Tardiness of jobs with Release Dates
3TFZRD	Total Tardiness and Total Flowtime of jobs with Zero Release Dates
3TFRD	Total tardiness and Total Flowtime of jobs with Release Dates
TFZRD	Total Flowtime of jobs with Zero Release Dates

TFRD	Total Flowtime of jobs with Release Dates
2TZRD	Total Tardiness of jobs with Zero Release Dates
HeuI	Heuristic I
HeuII	Heuristic II
HeuIII	Heuristic III
HeuIV	Heuristic IV
HeuV	Heuristic V
HeuVI	Heuristic VI

LIST OF NOTATIONS AND THEIR DEFINITIONS

Notations	Definition

$\in$	Element of (set membership e.g A={3,9,14}, 3 $\in$ A)
D_j	Due time of job, J
r_j	Release date of job, j
P_J	Processing time of job, J
$\sum_{j=1}^{n} C_j$	Total Completion time
$\sum_{j=1}^{n} E_J$	Total earliness
$\sum_{j=1}^{n} L_j$	Total lateness
$\sum_{j=1}^{n} W_j$	Total waiting time
$\sum_{j=1}^{n} F_J$	Total flow time
N_E	Number of early jobs
N_t	Number of tardy jobs
C_{max}	Makespan
F_{max}	Maximum flowtime
$\alpha_{0.05}$	Probability value at 95% significant level
$\propto, \beta$	these are used as the relative weights associated with the stated performance objective

CHAPTER ONE

INTRODUCTION

1.1 The Scheduling Problem

A scheduling problem involves the processing of job sets on processor(s) subject to constraints to optimise performance measure(s). Examples of performance measures are total tardiness, total flowtime, makespan, completion time, number of tardy jobs, among others. The processing of a job on a processor is called an operation (French, 1982). The goal is to determine a schedule that specifies the time a given job is to be executed (Karger *et al.*, 1999). Generally, the vital elements of scheduling are jobs, processors, performance measures and constraints (Pinedo, 2008). Scheduling deals with the sequencing and time tabling of jobs. Sequencing is the order in which a set of jobs are to be processed on a number of processors. The determination of the start time and the finish time of jobs is called the time tabling (French, 1982; Oyetunji and Oluleye, 2008). Sequencing and time tabling are, therefore, regarded as the subsets of scheduling.

Scheduling is a short-term execution plan of a production planning model. Production scheduling consists of the activities performed in manufacturing and servicing firms in order to manage and control the execution of production processes. A schedule is an assignment problem that describes in details (in terms of time unit) which activities to be performed and how the resources should be utilized to satisfy the plan (Fera *et al.*, 2015).

1.2 Representation of Scheduling Problem

Scheduling problems can be represented by three parameters notation given by : $\alpha|\beta|\gamma$ (Brucker, 2006).

where:

α is the processor environments like single, multiple, parallel processors etc,

β is the job characteristics or constraints like pre-emption, release dates, due dates etc,

γ is the performance measures like flowtime, makespan, tardiness, among other.

1.2.1 Processor environment

The processor environment is characterized by a string of two parameters; $\alpha = \alpha_1\alpha_2$ (Brucker, 2006). Possible values of α_1 are ◦, P, Q, R, PMPM, QMPM, G, X, O, J, F.

If $\alpha_1 \in \{◦, P,Q, R, PMPM,QMPM\}$, where ◦ denotes the empty symbol (thus, $\alpha = \alpha_2$ if $\alpha_1 = ◦$), then each job, *Ji* consists of a single operation.

If $\alpha_1 = ◦$, each job must be processed on a specified dedicated machine.

If $\alpha_1 \in \{P, Q, R\}$, it is called parallel machine. In this environment, there are n jobs J_i $(i = 1,2,3, \dots, n)$ that have to be processed on m machines M_i, $(i = 1,2,3, \dots, m)$ such that each machines can process at most one job at a time and that each job can be processed on one machine at a time. Parallel machine environment can be classifield into identical parallel machine, uniform parallel machine and unrelated parallel machine.

a. Identical parallel machines (α_1= *P):* These are multiprocessors system in which all the processors are identical, in the sense that they have the same computing power. In other word, for the processing time P_{ij} of job J_i on machine M_j we have $p_{ij} = p_i$ for all machines M_J. That is $P_{i1} = P_{i2} = P_{i3} = \cdots = P_{nm}$.
b. Uniform parallel machines (α_1= *Q) :* A uniform parallel machine is a natural generalization of identical machines in which the machines run at different speeds but do so uniformly, that is they differ by some constant speed factors. Thus, for each machine i there is a speed factor S_i, and $p_{ij} = p_i/S_i$, where p_i is the inherent processing requirement of job j.
c. Unrelated Parallel machines(α_1= *R*): In unrelated parallel machines, the processing time of jobs j_1 varies between the machines in a completely arbitrary fashion. The processing time of each job on any of the machines is different and bears no relation with each other. Though there is no relationship between machines speed in an unrelated parallel machine, there may exist hierarchy between the machines. Thus, the machines can be ranked from the highest to the lowest speed.

If α_1 = *PMPM* and α_1 = *QMPM,* it is called multi-purpose machines with identical and uniform speeds, respectively. In a multi-purpose parallel machine scheduling problem, a job can be processed by any machine of an associated, pre specified subset of the machine set. Thus, these problems generalize classical parallel machine problems in which a job can be processed by each machine of the machine set.

If $\alpha_1 \in$ {G,X,O, J, F}, it is called a multi-operation model, i.e. associated with each job Ji, there is a set of operations $Oi_1, \dots, Oi,ni$. The machines are dedicated, i.e. all μij are one element sets. Furthermore, there are precedence relations between arbitrary operations. This model is called a general shop. The general shop is indicated by setting α_1 = G. A scheduling problem is said to be a general job shop, if there are no restrictions upon the form of technological constraint and each job has its own processing order and bears no restriction on the processing order of any other jobs (French, 1982, Al-harkan 2008). For a general job shop problem, the number of possible sequences are $(n!)^m$, where n is the number of jobs and m is the number of machines.

Job shops (α_1 = J), flow shops (α_1 = F), open shops (α_1 = O), and mixed shops (α_1 = X) are special cases of the general shop. The job shop scheduling problem consists of *n* jobs, which require processing on at least m different machines. Each job has its own process sequence. Furthermore, the process sequences of the jobs are different from one another. The open shop scheduling problem consists of n jobs which are to be scheduled on m different machines. There is no process sequence for each job, which means that the operations of that job can be performed in any order.

The flowshop scheduling problem is one with several machines available in the shop. The characteristic of this type of shop is that the jobs processed in it use machines in the same order: they all have the same processing routing. There are *n* machines and *m* jobs. Each job contains exactly *n* operations. The *i*th operation of a job must be executed on the *i*th machine. No machine can perform more than one operation simultaneously. For each operation of each job, execution time is specified. Operations within one job must be performed in the specified order. The first operation gets executed on the first machine, then (as the first operation is finished) the second operation on the second machine, and so until the *n*-th operation.

Furthermore, α_2 can either be a positive integer or 0. If α_2 is a positive integer (1, 2,….m), it denotes the number of machines. If α_2 = *k*, then *k* is an arbitrary, but fixed number of machines. If the number of machines is arbitrary, set $\alpha_2 = \circ$.

1.2.2 Job environment

The job characteristics are specified by a set of parameters (β), which contain at most six elements; $\beta_1, \beta_2, \beta_3, \beta_4, \beta_5, and\ \beta_6$ (Brucker, 2006).

β_1 describes the job pre-emption. Pre-emption of a job or an operation means that the processing of a specific task of job may be interrupted and resumed or repeated at a later time, even on another processor. A job or an operation may be interrupted several times. If pre-emption is allowed, set β_1 = preemption, otherwise β_1 does not appear. β_2 describes the precedence relations between jobs. Precedence constraints between jobs implies that a given job must be processed before another specified job. If there is no precedence constraint, β_2 does not appear. β_3 describes the release date (r_i) for each job. The release date of a job is the time the job is available in the shop. It is a variable and varies from zero to any operation time. The dynamic shop is one where jobs arrive intermittently ($r_1 \neq 0$), while the static shop is one where all the jobs are available at the same time ($r_1=0$) for processing. For a static shop, β_3 does not appear. β_4 specifies the restrictions on the processing time or on the number of operations. The processing time of a job is the time the job spends on the processor(s). β_5 specifies the due date (d_i) for each job. The due date of a job is the date the job is expected to be completed. β_6 specifies a batch or partition problem. There are two types of partition problems; the parallel batching (P-batch) problems and the serial batching (S-batch) problems (Ghosh and Nagi, 2007).

For the P-batch problems, all the jobs of a batch are processed in parallel and the length of the batch is equal to the maximum of the processing times of all the jobs in the batch plus the set up time. For the S-batch problems, all the jobs of a batch are processed sequentially and the length of a batch is equal to the sum of the processing times of all jobs in the batch plus the set up time. β_6 indicates either the P-batch or the S-batch problem. If there are no batches, β_6 does not appear.

1.2.3 Performance measures

Performance measures are criteria by which the performance of any solution method can be assessed. Performance measures are also called performance objectives or scheduling criteria. It is not easy to state the performance measure of a scheduling problem (French, 1982). This is because they are numerous, complex and often conflicting. However, solving a scheduling problem involves finding a schedule that optimises a given performance measure. French (1982), Al-harkan (2010), classified performance objectives based on:

i. Completion time: Examples of criteria based on completion time are the total completion time, average completion time, makespan.

ii. Effective resources utilization: These include the total flowtime, maximum flowtime, average flow time, among others

iii. Due date: The main criteria in this category are the total earliness, total tardiness, maximum lateness, total lateness, average lateness, among others.

Performance objectives are formulated by mathematical expressions. French (1982), Oyetunji (2009) and Al-harkan (2010) stated some mathematical expressions used to compute scheduling objectives. Some of the criteria are described as follows:

i. The Makespan (C_{max}): This is the completion time of the last job in a schedule or the maximum completion time. A common scheduling problem is to minimise the makespan. The criterion is used to measure the level of utilization of the processor. The maximum completion time is given by:

$C_{max} = max\ (C_1, C_2\ ,\ C_3 \dots, C_n)$

ii. The total flowtime (F_{tot}): The flowtime is the difference between the completion time and the release date. Total flowtime is the sum of the flowtime of all the jobs. Scheduling criteria based on the total flowtime is given by :

$(F_{tot}) = \sum_{i=1}^{n} \mathrm{F_i} = \sum_{i=1}^{n}(\mathrm{C_i} - r_i)\ = F_1\ +\ F_2\ +\ F_3 + \cdots + F_n$

A common problem is to minimise the total flowtime. This is because flowtime is proportional to the resources utilization.

iii. The Total Tardiness: A job is said to be late or tardy, if it is completed after its due date. Tardiness is similar to lateness except that it carries only positive values. Whenever a job completes before its due dates, its lateness is negative and its tardiness is zero.

Total tardiness (T_{tot}) is given by: T_{tot}: $\sum_{i=1}^{n} T_i = \sum_{i=1}^{n} max\ \{0\ ,(C_i - d_i)\}$

A common problem is to minimise the total tardiness.

iv. The Total Earliness: The earliness is also a due date related performance measure but reflects early delivery of jobs. Total earliness is defined as the summation of earliness of individual jobs.

Total earliness (E_{tot}) = $\sum_{i=1}^{n} E_i = \sum_{i=1}^{n}(d_i - C_i)$

A common scheduling problem is to find a schedule that maximises the total earliness. Earliness is the opposite of lateness; hence, whenever lateness is negative, earliness is positive and vice versa.

1.3 Classification of Scheduling Problems

Scheduling problems are usually represented by three parameters; the processors, the performance objectives and the constraints. Thus, scheduling problems are classified based on the:

i. Number of processors,
ii. Number of performance objectives involved, and
iii. Constraints involved.

1.3.1 Scheduling problems based on the number of processors

Based on the number of processors, scheduling problems can be classified into single processor and multi-processors problems. A single processor scheduling problem is one in which there is only one processor for all the jobs. Some common examples are:

i. Aircraft queuing up to land in airport with only one runway: This is an n job, one processor problem. Assuming the number of aircraft arriving in a day is known, the aircraft are the jobs and the runway is the processor.
ii. Vehicles queuing for spraying and drying in a booth or baker.
iii. An auto repair shop with one working pit lift.
iv. The processing of jobs through a small non-time-sharing computer.
v. A paint manufacturing plant in which the whole plant is devoted to make a colour of paint at a time.
vi. Patients waiting to see a single specialist doctor.
vii. A unit means of transportation of a logistic or a shipping firm

A multi-processors scheduling problem is one in which there are several processors available upon which jobs may be executed. Some common examples are

i. An hospital with many doctors in the same area of specialization that can attend to patients
ii. A bank with many front desk officers or cashiers performing the same duties.
iii. A petrol station with many attendants selling to customers on queue.

A single processor scheduling problems can also exist in a multi-processors system, where bottleneck occurs. A bottleneck is a point of congestion in a system that occurs when the jobs inflow at a point exceed the jobs outflow at that point. A bottleneck creates queue and elongates the overall cycle time. Bottleneck exists in various systems from computer networking to factory assembly lines. For an example, the shipping department of a company may constitute a bottleneck, if the purchasing orders exceeded the distribution rate to the customer. The painting section of an auto repair firm may also constitute a bottleneck, if the total work inflow (output from other subsections like mechanical, electrical and body section) transferred to the painting section is high, compared to the work outflow at the section.

1.3.2 Scheduling problems based on the number of performance objectives

Based on the number of performance objectives involved, scheduling problems can be classified into:

i. Single criterion scheduling problems: These are problems in which only one performance objective is to be optimised.
ii. Multi-criteria scheduling problems: These are problems in which two or more performance objectives are to be optimised. The simplest form of multi-criteria scheduling problems is called the bi-criteria problem (M'Hallah, 2007). Bi-criteria problems are those in which only two performance objectives are to be optimised.

1.3.3 Scheduling problems based on the applied constraint

Two types of constraints are commonly found in scheduling problems: the processor capacity constraints and the technological constraints (Baker and Trietsch, 2013).

Processor capacity constraints deal with the number and the arrangement of processors. Single processor, multiple processors, uniform parallel processors, identical parallel processors, unrelated parallel processors, among others, are common examples of processor capacity constraints. Technological constraints deal with the restriction on the sequencing of the jobs (Al-harkan, 2010). Examples include the static system, dynamic system, deterministic system, stochastic system, precedence constraints, among others. The static system is one in which the set of jobs available for scheduling does not change over time. This implies that all the jobs have equal or effectively zero release dates. In contrast to static systems, scheduling problems in which new jobs appear over time is called the dynamic system. This implies that the jobs have non-zero release dates. Scheduling of jobs on single processor in manufacturing systems is complex, especially when the order of jobs arrival is dynamic (Kaplanoglu, 2014).

Static models have been studied more extensively than dynamic models because they require less computational time, though dynamic models offer a wider application. However, static models often capture the essence of dynamic systems, and the analysis of a static problem uncovers valuable insights and sound heuristic principles that are useful in solving the corresponding dynamic situation (Baker and Trietsch, 2013). Furthermore, when the conditions of a scheduling problem are known with certainty, the model is called deterministic while a problem with an explicit probability distribution is called stochastic (Pinedo, 2008).

1.4 Effectiveness and Efficiency

The effectiveness and efficiency are the two parameters considered in selecting solution methods for a scheduling problem. The effectiveness of an algorithm is a measure of closeness of the value of the objective function of the algorithm to the optimal or existing standard. The efficiency is a measure of computation or execution time required by the algorithm to solve an instance of problem. Based on the effectiveness and efficiency of an algorithms; solution methods can be classified into:

i. Exact Algorithm: These are solution methods that yield optimal result in terms of effectiveness and require a polynomial-time bound computation time (efficient). The limitation of these methods is that they are applied to a specific problem. Example include the Shortest Processing Time (SPT) rule for solving the scheduling problem of minimizing total flowtime with zero release dates.

ii. Enumeration solution methods: These are solution methods that yield optimal solution in terms of effectiveness but require prohibitive execution time, especially for large-sized problems. These include the complete enumeration and implicit enumeration methods (the Dynamic programing method, the branch and bound). The methods can be applied to different classes of optimization problem but with very limited problem sizes due to time complexity.

iii. Approximation solution methods: These are solution methods that yield approximate solutions in terms of effectiveness within a small computation time. These methods can be explored to solve all classes of optimization problems. Examples are heuristics and metaheuristics. In selecting an approximation solution methods to be explored for a given NP-hard problem among a set of polynomial time bound algorithms, effectiveness is given higher priority than efficiency. For an instance, in a minimization problem, if the value of the mean objective function and execution time of the two polynomial bound algorithms AA and DD were found to be AA(14, 0.004minutes) and DD(12, 0.04minutes), then the DD method would be selected in preference to the AA method. Though, the AA is 10 times faster than the DD to achieve the result while the DD is 1.2 times more effective than the AA.

1.5 Importance of Tardiness and Tardiness Related Objectives

Effective scheduling of jobs to optimise the required performance measure(s) is very important in manufacturing, production and servicing systems. This work presents solution methods for minimizing a due date based performance measure (the total tardiness) and an efficient resources utilization measure (the total flowtime). The static and the dynamic release dates constraints are considered. The importance of minimizing the tardiness related objectives to production and servicing firms include:

i. The present industrial environment is characterized by competition. Therefore, customer satisfaction in terms of quality, cost and delivery time is a pre-requisite for survival. A good quality control system will ensure that the product quality conform to the established standard. The products cost and prompt service delivery are partly influenced by the scheduling methodology that optimises both the total tardiness and the total flowtime.

ii. In some manufacturing and production firms, penalty cost incurred on tardy jobs increases with increase in total tardiness. Thus, to minimise the penalty cost, it is necessary to minimise the total tardiness.

iii. Furthermore, minimizing the composite function of the total tardiness and total flowtime implies that the organization attained performance development in production cycle time, respect of expected delivery dates, inventory and work in process management, adaptation and reactivity to variations in customer orders.

iv. In manufacturing, a schedule that minimises the flowtime either singly or as a part of a composite function also minimises the production time and costs. This is because such a schedule achieved optimum utilization of the production facility, staff members, and equipment.

v. The total cost of a schedule is a complex combination of processing costs, inventory costs, processor idle-time costs and tardiness penalty costs, amongst others. Therefore, considering scheduling approaches with sum of total tardiness and total flowtime is very relevant within the context of real life scheduling problems.

vi. A schedule that minimises only the total flowtime will ensure proper inventory management, reduction in production cost and profit maximization but offers no solution to late delivery of goods and services. Thus the firm may incur some tardiness penalty or loses its good will. On the other hand, scheduling approach for minimizing only the total tardiness will ensure prompt delivery of goods and services but with no consideration to breakeven and profit variables. Therefore, combining the two criteria to form a bi-criteria problem will ensure the aggregation of the benefits of each component.

1.6 Problem Statement

1.6.1 Broad consideration

Consider the operation of a firm (production or servicing firm) with only one processor or with multi-processors but one processor constituting a bottleneck. Examples of the formal include an auto repair firm with only one working lift, a working condition that requires the use of programmed robot with only one robot, a vehicle spraying shop with only one oven, among others. The shipping department of a firm which product is in high demand will constitute a bottleneck, if it receives larger purchasing orders beyond outflow capacity. A coupling machine of a ball point pen firm also constitutes a bottleneck, if several batches of components of the pen from other departments are ready simultaneously. The spraying section of an auto repair firm usually constitute a bottleneck, if the work output from other sections (like mechanical, electrical, body section) transferred to the spraying section is higher than the outflow of the section. A bottleneck processor has the longest queue in a multi-processors system which ultimately, results in failure to meet the delivery date and, therefore, has to be given special attention.

A situation of job accumulation is inevitable in a bottleneck processor and in a single processor system with inflow consistently exceeding the outflow. This could result in tardiness penalty either in monetary terms or loss of value. Although, proper scheduling has been identified as a solution to the problem, the combinatorial nature of job sequencing makes the solution a complex one. For an instance, ten accumulated jobs can be scheduled in 3, 628,800 different ways.

The accumulated jobs can be classified into two groups: the zero and the non-zero release dates jobs. For both groups, the firm will be interested in completing all the jobs with minimum deviation in the delivery dates. Also, for proper inventory management, the firm will also plan that the jobs spend the minimum possible times in the shop. To achieve this, a schedule that minimises the total tardiness and the total flowtime, either singly or as a composite function, will be required.

1.6.2 Problem classes

A close examination of the problem statement reveals that the scheduling problem considered here can be decomposed into six variants. These are the scheduling problems of minimizing the:

i. Total Flowtime of jobs with Zero Release Dates (TFZRD),
ii. Total Flowtime of jobs with Release Dates (TFRD),
iii. Total Tardiness of Jobs with Zero Release Dates (2TZRD),
iv. Total Tardiness of jobs with Release Dates (2TRD),
v. Composite function of Total Tardiness and Total Flowtime with Release Dates (3TFRD), and
vi. Composite function of Total Tardiness and Total Flowtime with Zero Release Dates (3TFZRD).

Extensive literature review has shown that only the scheduling problem of minimizing the TFZRD is a Class P scheduling problem, while others are NP-hard. The Shortest Processing Time (SPT) algorithm (Smith, 1956; Bansal and Kulkarni, 2015) produced optimal solution for the scheduling problem of minimizing the TFZRD. Also, the Modified Due Date (MDD) (Baker and Bertrand, 1982; Naidu, 2002) and KSA I algorithms (Oyetunji *et al.*, 2012) yielded effective solutions for the scheduling problems of minimizing the 2TZRD and TFRD respectively. However, problems (iv, v and vi), which are tardiness related are open problems for which an improvement can be made on the state of knowledge.

Given the problem description, the following single processor scheduling problems parameters are required:

i. A set of n jobs : $J_1, J_2, J_3, \dots, J_n$
ii. The processing time of each job; p_i
iii. The release or ready time of each job; r_i, and
iv. The due dates of each job; d_i.

1.7 Study Objectives

The primary objective of this study is to develop effective heuristics for solving the single processor scheduling problem of minimizing the**:** total tardiness of jobs with release dates (2TRD), total tardiness and total flowtime of jobs with zero release dates (3TFZRD) and total tardiness and total flowtime of jobs with release dates (3TFRD).

The secondary objectives are:

i. To propose solution methods for 2TRD, 3TFZRD and 3TFRD and to evaluate their effectiveness and efficiency relative to the selected existing solution methods.
ii. To demonstrate the utility of the proposed solution methods.

1.8 Assumptions Made in the Study

In solving the problems outlined in sub-section 1.6.2, the following assumptions are considered

i. Processors never break down and are available throughout the scheduling period.

ii. The problem is deterministic
iii. Splitting of jobs, job cancellation and job pre-emption are not allowed; and
iv. Only one job can be processed at a time. (This is the processor capacity).

1.9 Justification for the Study

The importance of the two performance measures were discussed in section (1.4). Furthermore, the selection of the single processor environment is not only because of its wide industrial application, but also, due to the fact that the multiple processors environment can be partitioned into single processor units. Baker and Trietsch (2009) discussed the concept of fundamental partition of multiple processors into single processor units.

Furthermore, a single processor can also constitute a bottleneck in a chain of processes or processors, such that the output of the production line is limited by the throughput of the bottleneck processor (Michael *et al.*, 2015). Almost every system has a bottleneck. If a system was running at full capacity, at least one processor would be accumulating jobs. A bottleneck processor restricts the entire production processes (Yang *et al.*, 2006). Therefore, identifying the bottleneck processor and scheduling the jobs effectively on it will enhance the output for the whole system (Bao *et al.*, 2012). Identifying bottleneck is critical for improving efficiency in a production line because it identifies the area where accumulation occurs. The processor that accumulates the longest queue is usually the bottleneck. Therefore, identifying the bottleneck and applying a proper scheduling policy on the processor will minimise stalls in production, supply overstock, pressure from customers and low employee morale.

1.10 Outline of the Succeeding Chapters

This thesis is set out in six chapters. Chapter two examines the literature related to this research, including an assessment of the existing models or solution techniques for the subject matter. The basic methodology employed for this research work, the simulated problems generated, the real life problems explored as well as the various comparative analyses carried out on the heuristics proposed are discussed in chapter three. Models' development as well as the implementation of the models for all the problem classes are discussed in chapter four.

The performance of the models in terms of the effectiveness and efficiency, the results of the comparative analyses as well as the utility of the models to real life problems are enumerated and discussed in chapter five. Conclusions based on the results obtained for each problem class, contribution of the work to knowledge, industrial practice and policy as well as recommendations for future research are also discussed in chapter six

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

It is necessary to provide background information and previous research that is relevant to the subject area in this study. This would assist in identifying the knowledge gaps that demand further investigation by critiquing the existing findings. It will also aid in comparing previous findings. In view of this, this chapter presents a comprehensive review of literature on the different classes of scheduling problems. Section 2.1 highlights the historical background of production scheduling in manufacturing. The state of knowledge on the subject matter is discussed in sections 2.2 - 2.6.

2.2 Historical Background of Production Scheduling

Production and servicing managers have changed over the years from capitalists who developed innovative technologies to custodians who struggle to control complex systems to optimise multiple objectives (Skinner, 1985). Production scheduling started in a simple way by determining when work on an order should commence and when the order should be due (Jeffrey, 2008). They did not provide any information about the time required for individual operations or jobs. This type of schedule was widely used before formal methods became available. The history of formal production scheduling started from the first chart developed by Henry Gantt (1916) to advance scheduling systems that now rely on algorithms.

Gantt explicitly discussed scheduling, especially in the job shop environment. Moreover, he discussed the need to coordinate and schedule activities to avoid “interferences”. In 1916, Gantt designed chart used by foremen and supervisors to know whether production was on schedule, ahead of schedule, or behind the schedule.

Gantt chart is used to measure activities by the amount of time needed to complete them. According to Cox *et al.* (1992), ‘Gantt chart is the earliest and best known type of control chart designed to show graphically the relationship between the planned performance and the actual performance’. Though, Gantt chart remains one of the most common tools for planning, controlling and monitoring production, other tools including planning board, loading and line of balance have been developed over the years. However, with the advancement in technology, computer based production scheduling emerged. Among the many benefits of information technology is the ability to execute complex algorithms automatically. The development of better algorithms for creating schedules is thus an important part of the history of production scheduling.

Linear programming was developed in the 1940s and applied to production planning and scheduling problems. George Dantzig (1947) invented the simplex method, a general but powerful technique for solving linear programming problems. In the 1950s, research into sequencing problems motivated by production scheduling problems led to the development of useful algorithms like the Johnson's rule for the two-processor flow shops, the Earliest Due Date (EDD) rule for minimizing maximum lateness, the Shortest Processing Time (SPT) rule for minimizing average flowtime, Shortest Remaining Processing Time (SRPT) rule to minimise the average completion time with release dates (Phillips *et al.*, 1998), Vladimír and Sudhakara (2010) solution method for flow shop scheduling algorithm to minimise completion time for n-jobs m-machines problem, among others.

Solving more difficult problems require a different approach (Roscoe, 1971). The BB technique appeared around 1960. The algorithm implicitly enumerates all the possible solutions and finds an optimal one. The advent of complexity theory in the 1970s explained why some scheduling problems are ‘hard’. Algorithms that can find optimal solutions to hard scheduling problems in reasonable time are unlikely to exist. However, since decision makers need solutions in minimum possible time, the use of search algorithms that can find good solutions have become inevitable. These algorithms are called approximation algorithms or heuristics. Other innovations include genetic algorithms, simulation annealing, ant colony optimization, and other evolutionary computation techniques. Developments in artificial intelligence have led to agent based techniques and rules based procedure that mimic the behavior of human organizations. Scheduling research is continuous and some problems are generated based on the existing situations.

2.3 Solution Methods to Scheduling Problems

There are various solution methods for solving scheduling problems. The choice of a given method depends on the application of the problem, the decision maker or the end user as well as the number of jobs involved (Karger *et al.*, 1997). Akinyemi *et al.* (2002) opined that no specific rules are better than others for all performance objectives. Scheduling solution methods can be classified into the following:

2.3.1 Complete enumeration

In this method, all the possible sequences are analyzed to select the optimal. However, scheduling problems are combinatorial in nature. For an instance, in a $n \times m$ problem size, the number of possible schedules is given by : $(n!)^m$.

where:

n is the number of jobs and

m is the number of processors.

Therefore, the number of possible sequences grow exponentially as the problem size increases. Thus, the method is usually applying in solving scheduling problems with limited problem sizes. This implies that the method is very effective but requires prohibitively high execution time (Panneerselvam, 2007).

2.3.2 Implicit enumeration techniques

These methods list all the possible schedules and eliminate the non-optimal ones from the list leaving those that are optimal (Pinedo, 2008). Common examples of implicit enumeration techniques are the dynamic programming method and the Branch and Bound (BB) method. Like the complete enumeration, the method is also very effective but inefficient.

2.3.2.1 The Branch and Bound (BB) technique

The Branch and Bound (BB) is a general purpose strategy for solving many combinatorial problems. The method consists of two fundamental procedures; the branching and the bounding procedure. Branching is the process of partitioning a large problem into two or more subproblems. Bounding is the process of calculating a lower bound of the optimal solution of a given subproblem (Kager *et al.*, 1999). The branching procedure replaces an original problem by a set of new problems that are:

i. Mutually exclusive and collectively exhaustive subproblems of the original, and
ii. Partially solved versions of the original.

The bounding process provides a means for curtailing and reducing the number of nodes to be exhausted so as to maximises the efficiency of the procedure. It involves calculating a lower bound of the solution to each subproblem generated in the branching process (Neda and Hamid, 2015). This can be done by:

i. Applying a suboptimal method with limited computational effort at the outset of the procedure. For an instance, in a minimization problem, supposing that a suboptimal method has an associated performance measure D. If one of the subproblems in the branching process has an associated lower bound A > D. Then there is no need to further analyze the subproblem in the search for an optimum. This is because, no resulting solution can yield a better value than D. When such a subproblem is found, its branch is said to be fathomed. By not branching any further from the fathomed branches, the enumeration process is reduced.
ii. Computing the contribution of the assigned subproblems to the total objective and consider the node(s) with the lowest value. The nodes with the lowest lower bound will be explored further. Ties are broken arbitrarily, in the first instance.

Having branched the problem, the two common search procedures are the depth first search and the frontier search (French, 1982). In the depth first search, the tree is searched by exploring a branch and systematically working it down to take either of the following decisions:

i. Eliminating the branch either before the final node or at the final node: This decision is taken if the value of the objective function from the branch is worse than the value obtained from the suboptimal method or the previous explored branch.
ii. Exploring the objective function value at the final node of the branch as the trial or the optimal solution. The former decision is valid, if the objective

function value obtained from the branch is better than the value from the suboptimal method and there are more branches to be explored. The latter decision hold, if the branch is the last one and yielded the best value of objective function compared to other explored branches or the suboptimal solution.

Furthermore, in the frontier search, we always branch from the node with the lowest lower bound. In other words, we initially branch to all the number of possible branches or nodes in the first level. The contribution of each of these nodes to the total objective function is calculated. The node with the lowest value is explored further to the next level. However, if more than one node has the same value, all the nodes are advanced to the next level. Furthermore, at the next level, if the value of the contribution of the objective function calculated from the explored node is greater than the value obtain from any of the unexplored node in the previous level, then the unexplored node in the initial level is also further analysed. The process continues until the final schedule is obtained.

The depth-first search method requires a lesser storage but a greater deal of computation compared to the frontier method. Thus, the latter method obtains the optimal faster than the former method (French, 1982). The BB method guarantees optimality in terms of effectiveness but easier to implement in problems with limited number of jobs due to the computational requirements.

2.3.3 Heuristics method

The implicit and complete enumeration methods have one inevitable limitation; the methods require prohibitively high computation time for large problem sizes. heuristic methods avoid this drawback. They can obtain solutions to large problem sizes with limited computational efforts. The limitation of heuristic methods is that, they do not guarantee optimality (Brucker, 2007). However, the effectiveness of a given heuristic can be measured through comparative analyses against the optimal solution for a feasible limited number of jobs (Brucker, 2007). This is the basis on which the present work is anchored

2.3.4 Metaheuristics method

Metaheuristic methods are efficient approaches for solving many optimization problems (Ilhem *et al.*, 2013). A metaheuristic is a procedure designed to provide a good solution to an optimization problem with limited computation capacity (Bianchi, *et al.*, 2009). The methods sample a set of solutions which is too large to be completely enumerated and may make few assumptions about the optimization problem being solved (Blum and Roli, 2003). Like heuristics, metaheuristics do not guarantee optimality. Examples of metaheuristics includes tabu search, simulated annealing, genetic algorithm among others.

2.4 Complexity of Scheduling Problems

Complexity theory provides a mathematical framework by which computational problems are studied (Sipser, 1997). The theory provides a means of measuring the performance of algorithms with respect to computational time, T. The time complexity of a solution method expresses the total number of operations such as additions, multiplications and comparisons for each problem instance as a function of the size of the instance (Weisstein, 2015). The input size of a typical scheduling problem is bounded by the number of jobs *n*, the number of machines, *m* and the number of bits to represent the largest problem parameter (the processing time, the due date, the release date, etc.).

An algorithm is a step-by-step procedure for solving computational problems (Brian and Daniel 2007). For a given input, it generates the correct output after a finite number of steps. An algorithm is said to be polynomial, if its running time is bounded by a polynomial input size. Polynomial algorithms are sometimes called efficient algorithms. For scheduling problems, typical values of polynomial execution time includes $O(n^2)$ and O(nm), where the big O-symbol stress that the numbers of elementary computations of the algorithm grow at the same rate as the function Cn^2, where C is a constant.

The class of all polynomially solvable scheduling problems is called the Class P. Examples of Class P scheduling problems are the scheduling problem of minimizing the total flow time of jobs with zero release dates, the scheduling problem of minimizing the number of tardy jobs with zero release dates, the scheduling problem of minimizing the maximum tardiness of jobs with zero release dates, the scheduling problem of minimizing the total flowtime of jobs with zero release dates, among others.

Another class of optimisation problems is known as the NP-hard problems. For such problems, no polynomial time algorithms are known that yield optimal solutions. The methods capable of producing optimal results for NP-hard scheduling problems are the complete enumeration and the implicit enumeration. However, these methods require prohibitive computation time especially for large problem sizes. Therefore, a NP-hard scheduling problem with large number of jobs cannot be practically solved optimally. Brucker (2007) stated that small sized NP-hard problems can be solved by BB method, while larger sized problems required approximation algorithms or heuristics.

2.5 Approaches to NP-hard Scheduling Problems

The first step in solving a scheduling problem is to determine the problem complexity status. This can be done by extensive literature review. Many scheduling problems are known to be NP-hard. Therefore, it is unlikely to obtain optimal solution to those problems efficiently, i.e. by polynomial time algorithms. An optimal solution can be found for a NP-hard problem by various methods of reduced enumeration, either by a BB algorithm or by formulating the problem as a mathematical programming problem. In both cases, for problems of practical interest only small-sized problems can be handled due to the time complexity involved. An exact solution to a job shop problem with 10 jobs and 10 machines remained unknown for decades (Papadimitriou, 1994).

In order to find a “good” solution within an acceptable time for a NP-hard scheduling problem with large problem sizes, approximation algorithms or the heuristics solution methods can be developed. An algorithm is called an approximation algorithm, if it is possible to establish analytically the closeness of the generated solution to the optimum (either in the worst-case or on the average). The approximation ratio gives the relative closeness of an approximation algorithm to the optimal.

The performance of a heuristic is usually analysed experimentally, through a number of runs using either generated instances or known benchmark instances. Heuristics can be very simple but effective. Therefore, a well-tailored heuristic, which performance has been guaranteed experimentally for a given objective, is usually implemented in real life problems.

Furthermore, when a solution method besides the complete enumeration and an implicit enumeration technique yields an optimal solution for a Class P scheduling problem, the term exact algorithm or simply say algorithm is used. Example is the Shortest Processing Time (SPT) algorithm for solving the scheduling problem of minimizing the total flow time of jobs with zero release date on a single machine. Early Due Date (EDD) algorithm for minimizing the total lateness or maximum tardiness of jobs with zero release date on a single machine.

2.6 Single Processor Total Tardiness Problems (SPTTP)

Scheduling a set of jobs which are to be processed on a single processor to minimise the total tardiness is known as the Single Processor Total Tardiness Problems (SPTTP). Due to the complexity of the SPTTP, Du and Leung (1990) proved that both the static and the dynamic variants of the problems are NP-hard with a given set of independent jobs. This implies that no heuristic has been found that yields optimal solutions. Thus, complete and implicit enumeration techniques are usually employed for solving the problems. However, computational time increases exponentially as the problem size grows (Potts and Van Wassenhove, 1982). This limits the use of enumeration techniques.

Nevertheless, Baker and Bertrand (1982) produced a near-optimal solution for the static variant of the problem. The solution is called the Modified Due Date (MDD) rule. Bean and Hall (1985) discusses extensively the accuracy of MDD rule. Extensive literature review shows that different researchers have explored approximation algorithm or an enumerative technique like the BB for solving the dynamic version of the problem. For instance, Chu (1992) proposed a BB algorithm that can solve up to 25 jobs for the static variant of the problem and up to 20 jobs for the dynamic variant. Baptiste *et al.*, (2004) presented a BB procedure with a time window constraint which represents the time interval within which each job can be scheduled. Furthermore, Chu and Portmann (1992) explored the dynamic version of the MDD rule to solve the problem. Naidu (2003) also stated that the dynamic version of the MDD rule solves the problem satisfactorily. Moreover, Baker and Triesch (2013) also listed authors who have explored heuristics to solve the problem. This is shown in Table 2.1.

Table 2.1: Some existing heuristics that solve the SPTTP with release dates

Performance Measure	**Best Rule (s)**	**Rules Compared**	**Author**
Total tardiness	SPT, ODD	SCR	Kanet and Hayya (1982)
	SPT, EDD	MST, SCR	Baker and Bertrand (1981)
	MST, SPT, EDD	OST, S/OPN, ODD, SCR	Muhlemann *et al.*,(1982)
	SPT/CR	COVERT	Anderson, and Nyirenda (1990)

Source: Baker and Trietsch (2013): Principles of Sequencing and Scheduling

Baker and Trietsch (2013) stated that none of the heuristics produces the optimal or near optimal solution. Thus, the problem remains open for further research. Therefore, proposing new heuristics that can produce better results or solutions not significantly different from the optimal even for small sized problems will fill this gap. In addition, selecting among the existing heuristics based on performance for comparative analyses against the intended proposed heuristics would also contribute to the state of knowledge. This work will achieve this feat.

2.7 Multi-criteria Scheduling Problems

These are scheduling problems in which two or more performance objectives are to be optimised. Multi-criteria scheduling problems are usually NP-hard (Rahimi, 2007). Parviz (2009) stated that the larger the number of performance objectives in scheduling problems, the higher the degree of computational complexity involved. When a scheduling problem contains only two criteria, it is called a bi-criteria scheduling problem (Ehrgott and Grandibleux, 2000). The total cost of a schedule is a complex combination of processing cost, inventory cost, processor idle-time cost and lateness penalty cost, amongst others (Oyetunji and Oluleye, 2009). A performance measure usually represents only one component of the total costs of a schedule (French, 1982). Considering scheduling problems with more than one criterion is very relevant within the context of real life scheduling problems (Nagar, *et al.*, 1995).

A necessary condition for multicriteria scheduling problems is the presence of more than one criterion. The sufficient condition is that the criteria must be conflicting. Two or more criteria are said to be conflicting, if the solution methods that produced optimal or near optimal for one impair or does not necessarily yield optimal or near optimal for the others. Two criteria are said to be “strictly”, if increase in satisfaction of one results in decrease in satisfaction of the other. However, the sufficient condition does not necessarily stipulate “strictly” conflicting criteria (Tapan, 2012).

2.7.1 Formulation of multi-criteria scheduling problems

Generally, scheduling problems can be formulated, using three parameters notation given by: $\alpha|\beta|\gamma$ (Brucker, 2006).

where :

α is the number of processor(s). α = 1 for single processor, α = 2 for two processors,

β is the job characteristics like pre-emption, release date, due date, among others, and

γ is the objective function.

Furthermore, Branke *et al.*, (2008); Deb 2011 defined multi-objectives scheduling problems as:

$$Min/Max\ \{f_1\ (x), f_2\ (x), f_3\ (x), \dots, f_m(x)\} \tag{2.1}$$

subject to $x \in S$

where:

S is the set of feasible solutions,

x is the decision vector, $\mathbf{x} = (x_1, x_2, x_3, \dots, x_n)$,

f_i is the performance objectives or criteria (i = 1,2, . . . , m), and

m is the number of performance criteria to be optimised.

2.8 Solution Methods to Multi-criteria Scheduling Problems

There are basically three approaches by which multi-criteria scheduling problems may be solved. These are: the hierarchical, simultaneous, and pareto-optimal approaches.

2.8.1 Hierarchical or Lexicographic optimization

This is an approach in which one of the objectives is considered as the objective to be optimised while others are considered as the constraints. This implies that the method optimises one goal at a time starting with the highest priority goal and terminating with the lowest, without degrading the quality of the higher priority goal (Taha, 2007; Ali, 2016).

T'kindt *et al.* (2001) proposed a BB algorithm and a heuristic for solving a-two processor flow shop scheduling problem with the objective of minimizing the total flowtime, subject to the constraint that the makespan is minimum. The problem was represented as F||Lex($C_{max}F_{tot}$) where the notation Lex($C_{max}F_{tot}$) indicates that the makespan (denoted by C_{max}) is minimised first and among all the schedules that minimise the makespan, a schedule was found that minimises the total flowtime (denoted by F_{tot}). Results of computational tests showed that the proposed BB procedure was effective in solving the problem for up to 25 jobs. For problems containing larger number of jobs, the proposed heuristic, which was also used as the upper bound in the proposed BB, was quite effective in finding a good schedule. Rajendran (1995) proposed a multi-criteria scheduling problem in which the objective function is minimization of the weighted sum of machine idle time, total flow time and makespan. T'kindt *et al.* (2002) proposed the SACO algorithm for a-two processor flow shop scheduling problem with the objective of minimizing the total completion time and the makespan. The latter was minimised prior to the former. The problem was represented as: $F_2||\text{Lex}(C_{max},\sum C_{ij})$. Computational experiments revealed that the SACO yielded better results compared to existing heuristics. It was conjectured that for larger problems, SACO was the most efficient.

Also, Hoogeveen (2005) solved a single processor hierarchical minimization problem in which the dominant criterion was the total completion time and the recessive criterion was the maximum lateness. The problem was represented as $1||\text{Lex}\left(\sum_j C_j, L_{max}\right)$. The SPT algorithm was used to minimise the total completion time ($\sum_j C_j$). Maximum lateness (L_{max}) was minimised, using the EDD schedules that conformed to the SPT sequence. Abdullah (2010) studied multicriteria scheduling problems to minimise total tardiness subject to maximum earliness or tardiness. Furthermore, Haidar and Tariq (2011) explored the EDD, SPT and MST heuristics to solve a single processor scheduling problem of minimizing three hierarchical criteria; the maximum tardiness, the maximum earliness and the sum of square of completion time. Ulungu *et al*, (1999) proposed a tool called MOSA, for solving multicriteria problem.

However, the hierarchical approach has two major limitations; the approach solves problems only in part and if no criterion is dominant, the method leads to a schedule that is unbalanced.

2.8.2 Simultaneous optimization method

In this method, the criteria are aggregated into single objective function called the Linear Composite Objective Function (LCOF). The expression for the LCOF, with respect to criteria X, Y and Z, is given by:

$$F(X,Y) = \alpha X + \beta Y + \gamma Z \quad (2.2)$$

Thus, the LCOF consists of the sum of the relative weights of each objective multiplied by the value of each of the objectives.

Mathematically, LCOF can be expressed as:

$$\text{Optimise } F(X) = a_1X_1 + a_2X_2 + a_3X_3 +, \dots, +a_mX_m = \sum_{k=1}^{m} a_kX_k \quad (2.3)$$

$$\text{such that } \sum_{k=1}^{m}(a_k) = 1 \quad 0 < a_k < 1 \text{, for all } k = 1, 2, \dots, m \quad (2.4)$$

where:

$a_1, a_2, \dots, a_m$ are the relative weights of criteria $X_1, X_2, \dots, X_n$ to be optimised, respectively.

F(X) is the Linear Composite Objective Function (LCOF), and

m is number of the criteria to be optimised.

Simultaneous optimization method has been explored for solving multi-criteria scheduling problems. For instance, Tabucanon and Cenna (1991) suggested a method for minimizing the mean flowtime and the maximum tardiness by assigning weights to both criteria. They generated efficient schedules based on the Wassenhove and Gelders (1980) algorithm and explored simulation approach to solve the problem. Furthermore, Farhad and Vahid (2009) explored simultaneous optimization for multi-criteria problem for minimizing the composite function of total machining costs (total completion time), the earliness, the tardiness penalties and the total makespan. Oyetunji and Oluleye (2010) proposed two heuristics (HR9 and HR10) for solving the bi-criteria scheduling problem for minimizing the total completion time and the number of tardy jobs with release dates on single processor. HR9 and HR10 were compared with the HR7 heuristic proposed by Oyetunji (2010). Based on the experimental results, the HR7 heuristic was recommended for the bi-criteria problem with number of jobs (n) less than 30 jobs while the HR10 heuristic was recommended for the bi-criteria problems with a minimum number of thirty jobs ($n \geq 30$).

Erenay *et al.* (2010) considered the bi-criteria scheduling problems for minimizing the number of tardy jobs and the average flowtime on single processor. They proposed four heuristics, two of which were constructive algorithms and the others were metaheuristics approaches. From the analysis, it was concluded that the proposed constructive algorithms produced efficient schedules and performed better than the existing heuristics. Oladokun *et al.* (2011) developed a bi-criteria algorithm for the simultaneous optimization of the makespan (C_{max}) and the number of tardy jobs (N_T) on a single processor with sequence dependent set-up time. An existing single criterion algorithm called the Set Sequencing Algorithm (SSA), was adopted for solving the bi-criteria problem. Graphical User Interface (GUI) based software of the algorithm was developed and its performance was evaluated with real life problems. 150 problems, with sizes ranging from 20 to 150 were randomly generated. The values of the C_{max} and the N_T of the solution sequences were analysed. The output sequences gave an average reduction of 32.1% in both measures compared with the input sequences. It was concluded that the SSA was suitable for the bi-criteria problem.

However, there are two major setbacks associated with the simultaneous optimization method. These are the problems of skewness (arising when the value of one criterion is a multiple of the other) and dimensional conflict (arising when the two criteria have different units). Both challenges were tackled through normalization proposition for scheduling problems with release dates (Oyetunji and Oluleye, 2009; Akande *et al* 2015). However, to the best of this researcher's knowledge, the normalization procedure for the static environment in which the release dates is zero remained an open problem. This work fills the gap.

2.8.3 Pareto-optimal solution approach

Pareto-optimal approach deals with the situation in which there is no information about the associated weight or the attached importance of either of the criteria. In this technique, a set of compromise solutions on both criteria are sought. These are called the pareto-optimal solutions (Oyetunji, 2011). This approach involves finding a set of all pareto-optimal schedules.

Hoogeveen (2005) stated that finding a set of pareto-optimal schedules is required in order to obtain a solution to a bi-criteria scheduling problem in polynomial time. Once the set of all pareto-optimal schedules has been obtained, the schedule that yields the minimum value of the composite objective function gives the solution to the bi-criteria problem. Fatih *et al.* (2009) proposed a pareto optimization approach for minimizing the number of tardy jobs and the average flowtime on single processor.

One limitation of pareto-optimal approach is that it does not provides a final solution to multi-criteria problems. It only provides a set of schedules (Oyetunji and Oluleye, 2009). This requires the decision maker to select among the set of solutions, the best schedule that optimises the required objective. Therefore, meaningful research is necessary to support the decision maker for the post-pareto analysis phase.

Heidi and David (2006) proposed two methods to prune the size of pareto-optimal sets. The first method is called the non-numerical ranking. The method provides the decision maker a set of solutions that match his preferences and compares solutions with different objective function combinations. The second approach is called the clustering method. In this approach, the *k*-means algorithm was proposed to group the pareto-optimal set into *k* different clusters with members of the same cluster similar to each other. The solution that was closest to the centroid of each cluster was chosen to be the representative solution in its respective cluster. Thus, the size of the pareto-optimal set was reduced to *k* general solutions to be analyzed by the decision maker.

2.9 Bi-criteria Scheduling Problems

The advantages of implementing a bi-criteria scheduling approach over its single criteria constituents are enormous. For an instance, exploring a schedule that minimises only the total flowtime will ensure reduction in production cost and profit maximization but offers no solution to late delivery of goods and services. Thus the firm's may incur some tardiness penalty or lose its good will. On the other hand, minimizing only the total tardiness will ensure prompt delivery of goods and services but with no consideration to breakeven and profit variables. Also, minimising total flowtime does not involve due date, thus firms problem is addressed. In the case involving total tardiness, the client's problem has priority and is therefore the problem addressed. Therefore, combining the two criteria to form a bi-criteria problem will better ensure the benefits accruing to both parties.

Two variants of the bi-criteria scheduling problems for minimizing the total tardiness and the total flowtime are consider in this study; the static (zero release dates) and the dynamic (non-zero release dates) variants. For the static variant, the SPT algorithm yields optimal solutions for the total flowtime while the MDD algorithm yields a near-optimal solution for the total tardiness. However, not much work has been found for the bi-criteria combination of the two objectives. This may be due to the fact that multi-criteria scheduling problems are NP-hard, independent of the complexity status of the corresponding single criterion problems (Della Croce *et al.*, 2003). Nevertheless, extensive literature review shows that only few solution methodologies (two heuristics) were proposed for the problem. The heuristics were named the heuristicA and the heuristicB. The authors of the work claimed the heuristics were the first proposed solution method and thus are not compared with any other model (Sen and Dillepan, 1999). Furthermore, the authors did not normalize the two objectives. There is therefore, no guarantee on the performance as well as the balance of the schedule obtained. In addition, the authors applied the heuristics only to four job sizes; 6, 7, 8, and 9 jobs. In this regard, there is a need to propose new heuristics for this problem and normalize the heuristics to avoid skewness towards either of the criteria. In addition, normalizing the heuristicA and heuristicB and comparing their performance to the intended proposed heuristics will also contribute to the state of knowledge. Also, implementing BB for small-sized problems and utilizing its results for benchmarking to measure the performance of the aforementioned heuristics will be a great feat. This work attempted to fill these gaps.

Furthermore, the introduction of non-zero release dates to a scheduling approach offers wider application. Extensive literature review shows that no direct heuristic has been found that solves the non-zero release dates variant of the problem. This is probably due to the fact that each of the constituent objective function has been found to be NP hard. Thus, the bi-criteria combination of the two criteria is strongly NP hard. However, Oyetunji and Oluleye (2012) proposed a Generalized Algorithm (GAlg) for solving multi-criteria scheduling problems using the individual objective. The algorithm is valid for both the static and dynamic environments. Therefore, it was presumed that implementing a GAlg for the problems and comparing the performance of the new proposed heuristics as well as the other heuristics from the literature to the GAlg (for both the static and dynamic variants) would also contribute to the state of knowledge on the subject matter.

CHAPTER THREE

METHODOLOGY

3.1 Introduction

A research project contains a variety of dovetailing steps and procedure that are utilized in achieving the underlined objectives. The steps are interconnected and the validity of research findings depend on the acceptability of the methodology. In this regard, this chapter presents the step-wise procedure as well as the methods employed in this research work. In addition, the data analyses and the computer language to be utilized for coding the various solution methods to be developed are also enumerated.

3.2 The Procedure

The first step of a research is the identification of open problems. Extensive literature review has shown that significant contribution can be made to the state of knowledge for single processor scheduling problems with tardiness related criteria, especially, when considering its bicriteria variants with total flow time. Therefore, this work classified the identified problems into three classes which are the scheduling problem for minimizing the **:** Total Tardiness of jobs with Release Dates (2TRD), composite function of Total Tardiness and Total Flowtime with Zero Release Dates (3TFZRD) and composite function of the Total Tardiness and Total Flowtime of jobs with Release Dates (3TFRD). The problems are named the Class I, Class II and Class III, respectively.

Furthermore, in solving a scheduling problem, it is necessary to determine the complexity status of the problem in order to select a suitable solution method. It has also been established through literature review that the identified problems are NP-hard. Thus, due to the complexity of the NP-hard problems, heuristic solution method was devised. However, in order to develop good heuristics for minimizing the stated criteria, the functional relationships between job parameters and the performance measures was also established. To achieve this, the problems were simplified and analysed with equations. This also aided in further understanding of the problems which is the key to models' development to produce good results. Therefore, two heuristics based on the iterative search method were developed for each problem class.

The heuristics proposed for the Class I problem were applied directly. The two criteria that are involved in the Class II and Class III problems are expressed as a Linear Composite Objective Function (LCOF). This is called the simultaneous optimization approach. The approach was explored because it yields a final schedule without further analysis unlike the pareto approach and also gives a balanced solution independent of the relative weight of each of the constituent criteria compared to the hierarchical method. In addition, a case of the LCOF in which the two criteria are equally important (carry the same weight) was considered. This was to ensure a balance aggregation of the benefits of the two measures. However, the heuristics proposed remained valid, if the weight of either of the criteria changes. Although, the LCOF equation would need to be altered to accommodate the change.

However, the problems of the skewness (arising when the values of one criterion is a multiple of the other) and the dimensional conflict (arising when the two criteria have different units) limit the use of the simultaneous optimization method. To avoid these problems, normalization was required. For the Class III problem in which the dynamic environment was considered, the existing normalization procedure found in the literature was first explored. For the Class II, in which the static environment was considered, a new normalization corollary was proposed. The heuristics proposed were then applied to minimise the normalized LCOF.

The heuristics proposed were tested on randomly generated problems for performance measure. To achieve this, some single processor scheduling problems were simulated. MATLAB R2010 programing language was explored. The desktop tool module (editor) was utilized. The choice of MATLAB over some other computer programming languages, like C++, Fortran, VISUAL basic, etc. is based on the fact that it can be used interactively. This means, if some commands are typed for executions at the MATLAB command window, the results are given immediately. The parameters that were generated are:

i. The number of jobs (n),

ii. The processing time (p_i),

iii. The release date (r_i), and

iv. The due date, (d_i).

Gursel *et al.* (2012) concept was adopted to generate these variables. The relations are given as follows:

v. The processing time P follows the uniform distribution U(1,10). This implies that Pi varies randomly from 1-10 inclusive.

vi. The release dates R follows the uniform distribution U(0, 40). This implies that Ri varies randomly from 0-40 inclusive.

vii. The due dates D_i is given by the equation; $Di = R_i + kP_i$. (3.1)
where k is uniformly distributed between U(1,4).

The equations would be modified to suit the problem Class II, in which zero release dates constraints is being considered.

Furthermore, the performance of a given solution method can be influenced by the problem size. In other words, some solution methods perform better under smaller number of jobs, while some perform under large number of jobs. Therefore, in evaluating the performance of a solution method to scheduling problems, the method must be tested with a wide range of problem sizes. To achieve this, eighteen different problem sizes ranging from 5 to 1000 jobs and with fifty problem instances under each problem size were generated and explored for each problem class. The adopted range of problem sizes (5-1000 jobs) was to show that the heuristics proposed would be suitable for the small scale, medium scale, large scale and global firms. For instance, a small scale firm could be like small auto mechanic shops that can have up to 10 jobs waiting for processing. Also, a medium scale firm can have up 25 jobs on their production line. Furthermore, a very large and global firm (like Amazon shipping company, Maersk shipping Group) can have up to 1000 jobs on the line. Also, it is an established fact that a sample size (n) that is not less than 30 ($n \geq 30$) is referred to as large sample size (Oyawale, 2006). Using this fact, the considered problem sizes were classified into the small-sized problems ($5 \leq n \leq 10$), medium-sized problems ($15 \leq n \leq 25$), and large-sized problems ($30 \leq n \leq 1000$). The adopted fifty problem instances was to ensure that the results obtained were not simply due to chance but a true reflection of the performance of the solution methods.

Furthermore, the implementation of the proposed heuristics over a wide range of simulated problem sizes succeeded the development of the heuristics and the simulation of single processor scheduling problems. This requires a great deals of computation and thus the use of computer programing language for coding is inevitable. For consistency and to achieve integration of the procedure, the same programing language (MATLAB) explored for the problem generation was also utilized. The coding of the heuristics and the implementation of the generated problems were in four phases or files. These are;

i. The random problem generation file,
ii. The single instance file to solve a single instance of the problem,
iii. The execution file, where the fifty instances required for any size problem were loaded and solved to obtained mean value of the objective function, and
iv. The run problem file where the saved random problems generated as well as the execution file were loaded and run to obtain the required output.

The output of the coding includes the following:

i. The single processor scheduling problems parameters: These are the processing time, the release dates as well as the due dates. They are the output of the random problem generation file.
ii. The value of the objective function of a single instance problem and the mean value of the objective function for the considered fifty instance: These are the single instant effectiveness and the mean effectiveness of the solution methods respectively. For the single criteria problem (2TRD), the effectiveness is the total tardiness while for the two bicriteria problems, the effectiveness is measure by the normalized Linear Composite Objective Function (LCOF).
iii. The value of the execution time for each instant of the problem as well as the mean value of the execution time for the fifty instances of the problem: These are the single instant efficiency and the mean value of the efficiency respectively.
iv. The final sequence of each of the problem instance.

Furthermore, some heuristics would be selected based on their performance from the literature to solve the same problems implemented for the proposed heuristics. The Branch and Bound (BB) procedure which gives an optimal result but requires a prohibitively high execution time would also be implemented for benchmarking. However, due to the prohibitive execution time, the BB method would only be applied to problem sizes not exceeding twenty-five jobs. The output (the effectiveness and efficiency) for all the implemented solution methods in each problem class would be grouped into the small-sized ($5 \leq n \leq 10$), medium-sized (($15 \leq n \leq 25$) and large-sized ($30\leq n \leq 1000$) problems for various comparative analyses. The standard deviation of the effectiveness and the efficiency of each of the problems-sized (small, medium and large sized) would also be computed.

3.3 Data Analysis

The relative effectiveness and efficiency of different solution methods were considered in selecting the method to generate schedules in production or servicing systems. Therefore, to demonstrate the utility of the heuristics proposed, the following analyses were carried out.

3.3.1 The mean and standard deviation

The mean of the effectiveness (values of the objective function) as well as the efficiency (the execution time) for the small, medium and large problem-sizes for the three problem classes were computed. Furthermore, in order to measure the spread or the closeness of the mean value to the value of the objective function in each of the individual number of jobs, the standard deviations was also computed. It is expected that the standard deviations would have large values (may even be greater than mean in some cases) for Class 1 problem in which the total tardiness in the objective function. This is because, the values of the objective function would be greatly spread out due to the combinatorial nature of scheduling problems. In other words, the data are unlikely to cluster around the mean. However, for Class II and III, where normalization were carried out, the value of objective function is expected to lie between 0-1, the standard deviation is expected to be small.

3.3.2 The approximation ratio (A.R)

The approximation ratio of an algorithm is the ratio of the value of the objective function obtained from the algorithm to the benchmark value ($\boldsymbol{BM_{value}}$). The benchmark value can either be the optimal value from the BB method or the standard value obtained from the most performed heuristic. For an example, the A.R of a given AA_6 heuristic is determined by:

$$\text{A.R of } AA_6 = \frac{\boldsymbol{AA_{6\ Value}}}{\boldsymbol{BM_{value}}} \tag{3.1}$$

A lower approximation ratio (A.R) value indicates better performance for a minimization problem. This is because, the ratio measures the closeness of the heuristics to the benchmark value. Thus, a lower A.R value indicates the heuristic is closer to the $\boldsymbol{BM_{value}}$ while a higher A.R value indicates the heuristic is far away from the $\boldsymbol{BM_{value}}$.

3.3.3 Test of mean (t-test)

The A.R test reveals that there is difference between two solution methods by determining the ratio of performance of the two solution methods to each other. However, the test fails to indicate the significance of the observed variation. T-test is used to determine if the values of the objective function obtained for different solution methods are statistically different. The common types of t-tests are; the one sample t-test, an independent samples t-test and the dependent samples t-test.

The one sample t-test compares the mean of one group against a known, predetermined or benchmark value. For an example, a cut point for a test score. An independent t-test compares the means of two independent groups. For an example, comparing the performance of two different algorithms on the same problem. A dependent t-test compares two groups that are related or from the same source. For an instance, comparing the results obtained from implementing the same algorithm on two different problems.

In this work, an independent t-test was carried out because the performance of different heuristics were too compared. However, four common types of independent t-tests include;

i. Paired two samples for mean: The paired two samples for mean t-test is normally used when you are testing twice on the same subject**.** For an example, testing two scheduling solution methods on the same problem. The test shows whether the results from the two solution methods are significant or not.

ii. Two samples assuming equal variance: This is used to test two groups to see if there is statistical significance or if the results may have occurred by chance. However, the test is applied, if there is an explicit information that the population variance of the two groups are equal.

iii. Two samples assuming unequal variance. This test is used to determine whether there is a difference between two groups within the population. For an example, to test whether there is difference in two different solutions on a given problem.

iv. Another test for means is called the Z-test. This is used where normal distribution is applied and is basically used when dealing with problems relating to large samples not less than thirty ($n \geq 30$).

However, the paired two samples for mean t-test was be adopted in this study. This is because there was no information about the variance of the objective function for the solution methods.

Furthermore, to explore t-test, the value of the probability (α) is required. This depends on the confidence level and whether a one-tailed or a two-tailed test is used. The confidence level describes the uncertainty of a sampling method. Often, researchers choose 90%, 95%, or 99% confidence levels, but 95% level is commonly used and it was adopted in this work. The 95% confidence level means that there is a 95% chance that the calculated means contains the true population mean.

A one-tailed test and a two-tailed test are alternative ways of computing the statistical significance between two set of parameters. A two-tailed test is used to test the significance of a claim of no differences between two set of parameters. In other words, a two-tailed test will test if the mean of the objective value obtained from a heuristic AA is significantly not different from that obtained from a heuristic BB (one direction). In contrast, a one-tailed test is used if deviation in only one direction is considered possible. This implies that a one- tail test will only test if the mean of the objective value obtained from a heuristic AA is significantly greater than that obtained from a heuristic BB. At the 95% level, for a one-tailed test, the α value is 5% = 0.05. Thus, it is written as $\alpha_{0.05}$. For a two-tailed test, the α-value is 0.1 (0.05×2) (Oyawale, 2006). Thus, it is written as $\alpha_{0.01}$

The interpretation using the $\alpha_{0.05}$ for a one-tailed test in a minimization problem is stated as follows:

i. If the $\alpha_{0.05}$ value is greater than 0.05 ($p > 0.05$), then the two solution methods are not significantly different from each other.
ii. If the $\alpha_{0.05}$ value is less than 0.05 ($p < 0.05$), then the two solution methods are significantly different from each other. The method with the lower objective function is the superior method.

Similarly, for a two-tailed test in a minimization problem, the $\alpha_{0.05}$ -value can be interpreted as follows:

iii. If the $\alpha_{0.05}$ value is greater than 0.1 ($p > 0.1$), then the two solution methods are not significantly different from each other.
iv. If the $\alpha_{0.05}$ value is less 0.1 ($p < 0.1$), then the two solution methods are significantly different from each other. The method with the lower objective function is the superior method.

A confidence level of 95% and a probability value of 0.05 (one-tailed test) would be adopted in this work.

3.3.4 The optimal count (OC) test

This test counts the number of times the heuristics achieve optimality. In other words, the number of times the heuristics yield the same value of the objective function as the BB method in the fifty instances for small problem sizes. While the A.R test measures the differences between the optimal and the heuristics results, the t-test shows the significance of the observed differences. However, the trends of performance of the heuristics as the problem sizes increases as well as the number of times the heuristics yield optimal in the fifty problem instances for small problem size are reveal by the optimal count test.

3.3.5 Consistency test and the ranking of the solution methods

In order to assess the consistency of the proposed heuristics against other implemented solution methods, the number of times each of the methods yields the best results in the fifty problem instances were computed and expressed in percentages. All the solution methods for each problem class was also ranked in the order of their effectiveness and efficiency.

3.4 The Model Implementation for Real-Life Problems

To further demonstrate the utility of the proposed heuristics, the solution methods were applied on real life data. The real life problems were obtained from the painting section of the VAL-VAL Auto Clinic, Ibadan, Oyo State, Nigeria.

The painting section of the VAL-VAL Auto-clinic Engineering limited located at Ibadan was visited. The unit carries out the following activities;

i. Spraying of vehicles,
ii. Drying and baking of vehicles, and
iii. Coordinating the rentage of the baker (oven).

The unit is equipped with a baking oven, where spraying is carried out. The oven serves as the processor, while the vehicles are the jobs. The sprayed vehicle (job) is kept inside the oven for certain period of time for proper drying. This is the processing time. The oven can only house a car at a time. This is the processor capacity. Some auto repair shops and automobile dealers also rent the baker for specified period of time. Therefore, the job schedule on the baker includes the vehicle brought by customers to the VAL-VAL Auto-Clinic Engineering limited and the vehicle brought by other firms intended to rent the oven.

3.4.1 Data collection

Data were collected from the firm using the customer detail forms and by interviewing the workshop manager. A customer details form of an automobile firm usually contain information about the vehicle release date. The following pieces of information would be gathered from the firm.

i. The release date**:** The date the job (vehicle) was brought to the firm or the date any firm intended to rent the oven registered their intention was used as the release date of the job. It were assumed that scheduling starts from the 1st day of the Month. Thus, if a vehicle was received on the 6th of May 2015. The release date is 6. Therefore, the release dates values would ranges from 1-31 inclusive depending on the month of the year.

ii. The processing time**:** The processing time of job is the time or day the vehicle is expected to stay in the baker or the number of day(s) the firm that rents the baker intends to spend.

iii. The due date: The date the firm gives to the customer that the job would be completed is the due date. For instance, if a vehicle is brought to the firm on 30th of May, 2015, and the firm promises the customer to pick the vehicle on 3rd of June, 2014. Then, the release date is 30, and the due date is 34.

3.5 The MATrix LABoratory (MATLAB)

The MATLAB is a powerful computing system for handling calculations involving scientific and engineering problems. The name MATLAB stands for MATrix LABoratory, because the system was designed to make matrix computations easy. However, the flexibility of MATLAB environment makes it easy to execute algorithms development, system analyses, differential equations, mechanical system analyses and electric circuits modelling.

One of the advantages of MATLAB over some other computer programming languages, like C++, Fortran, VISUAL basic, etc. is that it can be used interactively. This means if some commands are typed for executions at the MATLAB command window, the results are given immediately. Therefore, this work will provide a standard code and layout to generate any number of problem instances with unlimited number of jobs. The general codes to execute the unlimited problem instances from any algorithm was also provided.

CHAPTER FOUR

MODEL DEVELOPMENT

4.1 Introduction

The effectiveness and the efficiency must be taken into consideration while designing a scheduling heuristic. Therefore, a scheduling heuristic may not necessarily involve numerous computational steps once it achieves a level of acceptable accuracy. This is measured through comparative analyses against the currently explored models. Therefore, the key to developing a good heuristic for a given problem is the clear understanding of the problem and not the numbers of the computational steps. Having this in mind, heuristics are then developed for solving the problems considered in this work. Therefore, this chapter discusses the development of heuristics for the Classes 1, II and III problems. The need for normalization as well as the corollary utilized for the normalization of the Classes II and III problems are also enumerated. Furthermore, for all the problem classes, the solution methods selected from the literature for comparative analyses are discussed.

4.2 The Scheduling Problem of Minimizing the 2TRD

The scheduling problem of minimizing the total tardiness on single processor with non-zero release dates was explored as the Class 1 of this study. The analysis of the problem as well as the proposed heuristics are now discussed:

4.2.1 Problem definition

Given a single processor scheduling problem, where a set of *n* jobs have to be sequenced on a processor to minimise the total tardiness. Assuming that only one job can be processed at a time and that the problem is deterministic. Thus, the release dates (r_i), the processing time (p_i) and the due dates (d_i) of every job (J_i) are known with certainty. The time the processing of a job, i starts on the processor, designated as S_i, possesses a property defined by:

$$S_i \geq r_i \tag{4.1}$$

The completion time (C_i) of a job i is defined as:

$$\mathrm{C_i} = \mathrm{S_i} + p_\mathrm{i} \tag{4.2}$$

where:

$S_i = C_{i-1}$ or r_i

For i = 1

$\mathrm{S_i} = r_i$

$$\mathrm{C_i} = \mathrm{p_i} + \mathrm{r_i} \tag{4.3}$$

For job in position i +1 (job that succeeded the job in position i),

If $\mathrm{C_i} > r_{\mathrm{i+1}}$

$$\mathrm{C_{i+1}} = \mathrm{C_i} + p_{\mathrm{i+1}} \tag{4.4}$$

$$\mathrm{S_{i+1}} = C_i \tag{4.5}$$

If $\mathrm{C_i} \leq r_{\mathrm{i+1}}$

$$\mathrm{C_{i+1}} = \mathrm{r_{i+1}} + \mathrm{P_{i+1}} \tag{4.6}$$

$\mathrm{S_{i+1}} = r_i$

A job is said to be late or tardy if it is completed after its due date.

The tardiness is given by: $T_i = max\ \{0\ ,(C_i - d_i)\}$ (4.7)

The total tardiness is (T_{tot}): $\sum_{i=1}^{n} T_i = \sum_{i=1}^{n} max\ \{0\ ,(C_i - d_i)\}$ (4.8)

4.2.2 Materials and methods

The methods adopted in this study involves proposing and implementing two new heuristics to solve the problem. Two heuristics were also selected (based on their performance) from the literature to solve the problem. In addition, the BB procedure was also implemented for small and medium-sized problems. Comparative analyses was then carried out on all the implemented solution methods.

4.2.3 Selected solution methods from the literature

Among the solution methods found in the literature, the following heuristics were selected for implementation. The solution methods are now outlined;

i. The Minimum Slack Time (MST) Rule: The MST rule schedules jobs in the order of increasing slack time. The slack time (S_T) of a job j is given by:

$$S_T = a_j(t) - P_j \tag{4.9}$$

where:

S_T is the slack time,

p_j is the processing time of job j,

$a_j(t)$ is the allowance of the job j.

The allowance ($a_j(t)$) of a job j is given by:

$$a_j(t) = d_j - t \tag{4.10}$$

where:

d_j is the due date of job j, and

t is the sequencing time which can either be the completion time of the job at position (j-1) or the release date of job j.

ii. The Modified Due Date (MDD) Rule: The MDD rule at any time schedules the next job from the set of all unscheduled jobs 'U' with the smallest priority index (Π_i). The priority index is given by:

$$\Pi_i = \{max\{t + p_i, d_i\}\} \tag{4.11}$$

where:

t is the starting time of the next unscheduled job i ($i \in U$) which can either be the completion time of job in position i-1 or the release date of job i,

p_i is the processing time, and

d_i is the due date.

If there are only two jobs j and k to be scheduled at a time t, job j will precedes job k if $\{max\{t + p_j, d_j\}\} \leq \{max\{t + p_k, d_k\}\}$.

However, the MDD rule does not consider two jobs at a time when there are more than two unscheduled jobs. It considers all the available jobs, computes their priority indices (Π_i) and chooses the job with the least priority index.

iii. The Shortest Processing Time (SPT) Rule: The SPT rule schedules jobs in the order of non-decreasing processing time.

iv. The Early Due Date (EDD) Rule: The EDD rule schedules jobs in the order of non-decreasing due date.

4.2.4 Proposed solution methods

Two heuristics; named heuristicI (HeuI) and heuristicII (HeuII) are proposed for this problem. The solution methods are now described.

i. HeuI

In order to minimise the total tardiness of jobs with non-zero release dates on a single processor, three parameters are involved; the processing time, the release date and the due date. In any schedule, the effects of the parameters are significant at the beginning, while towards the tail end the effect of both the processing times and the due dates are dominant. However, as the length of the schedule increases, the processing time is the most dominant parameter. In this regard, the HeuI algorithm combines the EDD with the SPT to obtain a schedule. The algorithm is as follows:

Initialization

JobSet A = [J_1, J_2, J_3, . . . , J_n], set of given jobs

JobSet B = [0], set of scheduled jobs

JobSet C = [J_1', J_2', J_3', . . . , J_n'], set of unscheduled jobs, J_j' = J_j

JobSet D = [J_1', J_2', J_3', . . . , J_n'], set of unscheduled jobs, J_j' = J_j

JobSet E = [J_1', J_2', J_3', . . . ,J_n'], set of unscheduled jobs, J_j' = J_j

JobSet U = [J_1', J_2', J_3', . . . ,J_n'], set of unscheduled jobs, J_j' = J_j

The steps now follow;

STEP 1: Arrange JobSet A in the order of non-decreasing processing times and put same in JobSet C. If there is a tie, break arbitrarily

STEP 2: Arrange JobSet A in the order of non-decreasing due date and put same in JobSet D. If there is a tie, break arbitrarily

STEP 3: Compute the tardiness of each of the jobs in the JobSet C and the JobSet D

STEP 4: Combine the two schedules by scheduling the job in the same level and with the lower tardiness. If there is a tie, break the tie with due date. The resultant schedule is called JobSet E

STEP 5: Check if any job exists more than once in the JobSet E. If yes, remove the repeated job from the back position. Compute the length of the resultant schedule. The resultant schedule is called the JobSet U. Otherwise, JobSet E is renamed JobSet U

STEP 6: If the length of the JobSet U is equal to length of the JobSet A. Go to step 9. Else, go to step 7

STEP 7: Subtract JobSet U from JobSet A to obtain the jobs that have not been scheduled. The jobs constitute JobSet H

STEP 8: Arrange JobSet H at the back of JobSet U in the order of the due date

STEP 9: Compute the total tardiness of the JobSet U. If the total tardiness of JobSet U is less than that of JobSet C, JobSet C is the JobSet B, otherwise JobSet U is the JobSet B.

STEP 10: Compute the total tardiness of the required schedule (JobSet B).

STEP 11: Stop.

ii. HeuII

This is a modification of the HeuI. The two heuristics are based on the same principle. However, the step 1 in the HeuI is replaced by arranging the JobSet A in the order of the sum of the processing times and the release dates.

4.2.5 Application of the BB to solve the problem

The BB procedure implemented for this problem is described as follows:

The frontier search method was explored to branch, while the dynamic DMDD heuristic was used to bound the branching tree. The problem p(0) was partitioned into *n* subproblems, p(1), p(2), . . . , p(n), by assigning the first position in the sequence to each of the nodes in the first level of the branching tree. Thus, the p(1) is the same problem, but with job 1 fixed in the first position; the p(2) is similar, but with job 2 fixed in the first position, and so on.

Let *s* denote a partial sequence of jobs from among the *n* jobs originally in the problem. Also, let js denote the partial sequence in which s is immediately preceded by job j. Therefore, the completion time (C_j) of the sequence js is given by;

For j = 1

$$C_j = p_j + r_j \tag{4.12}$$

For job in sequence j+1 i:e job s

$$C_{j+1} = \begin{cases} C_j + p_{j+1}\,, \; C_j > r_{j+1} \\ r_{j+1} + p_{j+1}, \; C_j \leq r_{j+1} \end{cases} \tag{4.13}$$

Let Q(s) represent some subproblems at level k in the branching tree, where $k \leq n$. These subproblems will be the original problems at that level with the different jobs assigned at the first positions in each node at the level. Associated with Q(s) is a value, V_s, which is the contribution of the assigned jobs in each level to the total tardiness. That is,

$$V_s = \sum_{j\in s'} T_j = \sum_{j=1}^{s} (\max\{0,\ C_{t(js)} - d_s\}) \tag{4.14}$$

This value of V_s is calculated for all the nodes in each level and compared to the lower bound obtained from the DMDD heuristics (for the whole problem). As the level progresses, the value of the lower bound obtained from each node increases. However, at any level, k, any node whose value of V_s is greater than the heuristic value is discarded. The process continues until all the jobs are scheduled.

The bounding process provides a means for curtailing and reducing the number of nodes to be exhausted in order to improve the efficiency of the procedure. The bounding procedure calculates a lower bound by applying the DMDD heuristic at the outset and compares the value to the objective function obtained from all the branches in all the nodes at each level.

4.3 Bi-criteria Scheduling Problems

The bi-criteria scheduling problem of minimizing the LCOF of total tardiness and total flowtime on a single processor with zero release dates was explored as the Class II. The non- zero release dates variant of the Class II was named as the Class III. The problems analyse, the proposed normalization procedure as well as the proposed heuristics are now discussed.

4.3.1 Problem definition

Given the following for a single processor scheduling problem**:**

i. A set of n jobs; J_1 , J_2 , …, J_n

ii. The processing time of each job, P_i, and

iii. The due date of each job, d_i .

The job tardiness T_i, is defined by;

$$T_i = \max\{0 ,(C_i - d_i)\} \tag{4.15}$$

The total tardiness (T_{tot}) is thus defined as:

$$T_{tot} = \sum_{i=1}^{n} T_i = \sum_{i=1}^{n} max \{0, (C_i - d_i)\} \tag{4.16}$$

Similarly, the flowtime, F_i of job is defined by:

$$F_i = C_i - r_i \tag{4.17}$$

The total flowtime (F_{tot}) is given by:

F$_{\text{tot}}$ = $\sum_{i=1}^{n} F_i$ = $\sum_{i=1}^{n} (C_i - r_i)$ (4.18)

For the Class II, the release dates of all the jobs is zero, thus

F$_{\text{tot}}$: $\sum_{i=1}^{n} F_i$ = $\sum_{i=1}^{n} C_i = F_i + F_2 + F_{i3} + \ldots + F_n$ (4.19)

Using the notations of Graham *et al.* (1979), the Class II problem is represented as:

$1 \mid \mid (\sum_{i=1}^{n} T_i, \sum_{i=1}^{n} F_i)$ (4.20)

Similarly, the Class III problem is represented as:

$1 \mid \boldsymbol{r_i} \mid (\sum_{i=1}^{n} T_i, \sum_{i=1}^{n} F_i)$ (4.21)

Using the simultaneous approach, the LCOF is defined as:

LCOF = $(\alpha \sum_{i=1}^{n} T_i \; + \; \beta \sum_{i=1}^{n} F_i)$ (4.22)

where:

α is the relative weight of the total flowtime, and

β is the relative weight of the total tardiness.

Also, the sum of the relative weights of the two criteria is equal to 1.

$\alpha + \beta = 1$ (4.23)

Therefore, the LCOF consists of the sum of the relative weights of each objective multiplied by the value of the objectives (Akande *et al.*, 2014).

A bi-criteria problem can only be solved, if the values of $\propto$ and β are known (Oyetunji and Oluleye, 2009). In this work, a case of the total tardiness criterion being as important as the total flowtime criterion is considered.

Thus, $\alpha = \beta = 0.5$

Therefore; LCOF = $0.5(\sum_{i=1}^{n} T_i \; + \; \sum_{i=1}^{n} F_i)$ (4.24)

However, the problems of skewness and dimensional conflict associated with the LCOF are tackled through normalization proposition.

4.4 Normalization

A composite objective function will be biased or skewed towards one of the criteria if the value of that criterion is a multiple of the other. Similarly, if there is dimensional conflict, then the resultant composite function is said to be unbalanced. To obtain a balanced and an unbiased composite function, normalization is required. Oyetunji and Oluleye (2009) showed that the normalized value of any criterion in a multicriteria problem can be expressed as:

$$X_N = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{4.25}$$

where:

X_{min} is the minimum possible value of the criterion X,

X_{max} is the maximum possible value of the criterion X,

X is the particular value of the criterion X obtained from a solution method, and

X_N is the normalized value of the criterion X.

Therefore, the problem is to determine the minimum and the maximum possible values of each criterion. These values are called the extreme values.

4.4.1 The extreme values

For a minimization problem, the minimum possible value of a scheduling criterion represents the best value the criterion can attain, irrespective of the solution method adopted. On the other hand, the maximum possible value of a scheduling criterion represents the worst value the criterion can attain, irrespective of the solution method adopted. These values are called the extreme values.

The two objectives (the total flowtime and the total tardiness) considered in this work have the same unit, their values, especially at the beginning of a very good schedule (where total tardiness is likely to be zero or very small), may not be in the same range. Thus, the value of each of the objectives were normalised to the range of [0, 1], thereby yielding dimensionless quantities/variables as in Cochran *et al.* (2003), (Oyetunji, 2011).

Oyetunji and Oluleye (2009) proposed some equations to determine the extreme values of some objective functions with non-zero release date constraint. This is shown in Table 4.1.

Table 4.1: The maximum and the minimum possible values of the objectives

Objectives	Maximum Values	Minimum Values
$\mathrm{C_{tot}}$	$\mathrm{n(R_{max})} + nP_1 + (n-1)\,P_{n-2} \pm \ldots \mp 1(P_{n)} = \mathrm{n(R_{max})} + nP_1 + (n-i)P_{(n-i-1)} + 1(P_{n)}$ where i = 1 : n-2	$nP_1 + (n-1)P_{n-2} + \ldots + 1(P_{n)} = nP_1 + (n-i)P_{(n-i+1)} + 1(P_{n)}$ where i = 1 : n-2
$\mathrm{F_{tot}}$	$\mathrm{n(R_{max})} + nP_1 + (n-1)P_{n-2} + +1(P_{n)} - \mathrm{n(R_{min})} = nP_1 + (n-i)P_{(n-i+1)} + 1(P_{n)} - \mathrm{n(R_{min})}$	$nP_1 + (n-1)P_{n-2} + \ldots + 1(P_{n)} - \mathrm{n\,(R_{max})} = nP_1 + (n-i)P_{(n-i+1)} + 1(P_{n)} - \mathrm{n(R_{max})}$
$\mathrm{T_{tot}}$	$\sum_{j=1}^{n} d_j - (\mathrm{C_{tot}})_{max}$	0

Source: Oyetunji and Oluleye (2009): Assessing solution methods to mixed multi objectives scheduling problems.

However, the equations in Table 4.1 do not apply to a static system in which the release dates of all the jobs is zero. To solve this problem, a new corollary was proposed.

4.4.2 The proposed corollary

This corollary is based on the fact that for single processor scheduling problems, optimal or near-optimal solutions exist for some single criterion, scheduling problems with zero release dates. For instance, the SPT algorithm yields an optimal solution for the total flow time, total completion time, average flow time, among others. Also, the EDD and Moore algorithms yield optimal solutions for the maximum tardiness and number of tardy jobs, respectively.

The proposed corollary states that in order to determine the extreme values of a scheduling criterion for which a known optimal schedule or solution exists, the schedules corresponding to the two extreme values are notional.

PROOF: The validity of the corollary falls under two arguments;

i. Notional schedule is impossible and all the jobs corresponding to the extreme points must be represented on the Gantt chart (argument against the corollary)
ii. A notional schedule is possible with only one job corresponding to the extreme points represented on the Gantt chart (argument in favour of the corollary).

In order to validate the corollary, the two arguments were tested for a scheduling problem of minimizing the total flowtime on a single processor with zero release dates. The problem is solved optimally by the Shortest Processing Time (SPT) algorithm.

4.4.3 Argument against the corollary

The extreme values of the objective were determined using the argument against the corollary as follows;

i. The minimum possible total flowtime

The total flowtime is the sum of the flowtime. This is given as

$$\sum_{i=1}^{n} F_t = F_1 + F_2 + F_3 + F_4 + . . . + F_n \tag{4.26}$$

The flowtime is the sum of the waiting time (w_i) and the processing time (p_i). This is given as;

$F_i = w_i + p_i$ (4.27)

The minimum possible value of the flowtime is the flowtime obtained when the waiting time is zero. However, the waiting time can only be zero for job scheduled in the first position. Furthermore, since only one job can be processed at a time, then it follows that for jobs scheduled in position 2, 3, 4,….n. the waiting time will always be greater than zero.

The minimum flowtime can thus be analyzed as follows;

For job scheduled in position 1, the minimum possible flowtime is given by

$F_i = p_1$ (because, w = 0) (4.28)

For job scheduled in position 2, the minimum possible flowtime is given by

$F_i = p_1 + p_2$ (because, w = p_1) (4.29)

For job scheduled in position 3, the minimum possible flowtime is given by;

$F_i = (p_1 + p_2) + p_3$ (4.30)

Therefore, for any job schedule in position, k, the waiting time is given by;

$w_k = \sum_{i=1}^{k-1} p_i$ (4.31)

$k = 2,3, \dots, n$

For k = 1, $w_k = 0$

The minimum possible total flowtime,

$F_k = p_1 + \{\sum_{i=1}^{k-1} p_i + \sum_{i=2}^{n} p_i\}$ (4.32)

For $k = 2,3, \dots, n.$

Where p_1 is the flowtime of the first scheduled job.

The minimum possible value of the total flowtime,

$F_k = p_1 + \{\sum_{i=1}^{k-1} p_k + \sum_{i=2}^{n} p_i\}$ (4.33)

ii. The maximum possible total flowtime

The maximum possible value of the flowtime is achieved by maximizing the waiting time. The maximum waiting time will depend on the position of the job in the schedule.

The waiting time for the job in the position k is given by;

$$w_k = \sum_{i=1}^{k-1} p_i \qquad \text{For} \quad k = 2,3,\dots,n \tag{4.34}$$

The waiting time is maximum, when the job is scheduled in the last position (k = n),

The maximum waiting time for the job in the last position, n, is given by;

$$w_k = \sum_{i=1}^{n-1} p_i \tag{4.35}$$

The maximum waiting time for the job in the position k-1 is given by;

$$w_k = \sum_{i=1}^{k-2} p_i \tag{4.36}$$

The maximum waiting time for the job in the position 1 is given by:

$$w_k = 0 \tag{4.37}$$

Thus, the flowtime for any such job i scheduled to positions 2, 3, . . . , n is given by:

$$F_k = w_k + p_k = \sum_{i=1}^{k-1} p_i + p_k \tag{4.38}$$

For k = 2, 3, 4, . . . , n

Therefore, the maximum possible value of the total flowtime is given as:

$$\boldsymbol{F_{tot} = p_1 + \sum_{k=2}^{n} \left\{ \sum_{i=1}^{k-1} p_i + p_k \right\}} \tag{4.39}$$

4.4.4 Argument in favour of the corollary

To validate the corollary, the use of notional schedules must be justified. From equation 4.22 discussed above, the normalized value of scheduling criteria is defined as;

$$X_N = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{4.22}$$

The equation (4.22) is similar to equation of slope of a straight line AB shown by Figure 4.1.

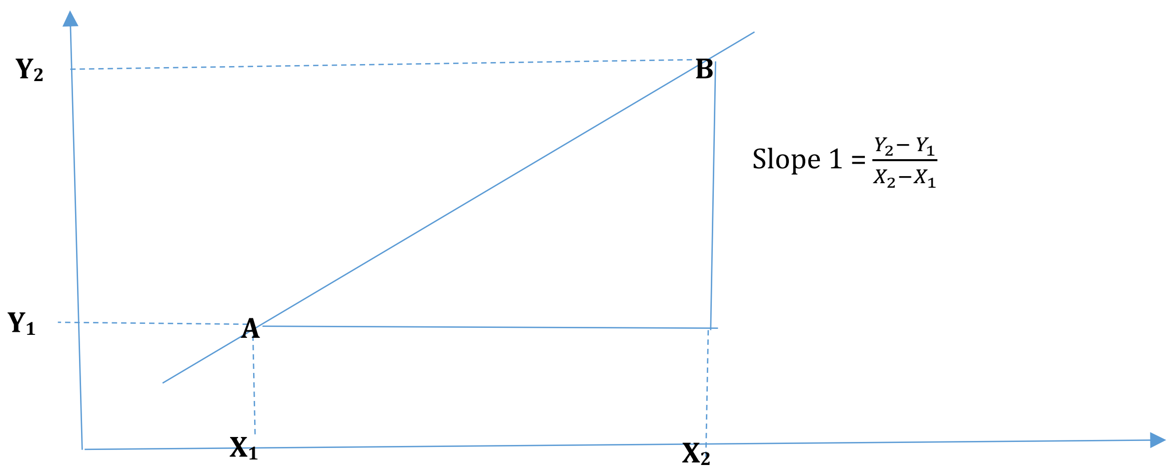


Figure 4.1: The slope of a straight line graph

However, consider some notional values that are not part of the data used to plot the graph in Figure 4.1 and in such a way that can extend the minimum point (A) to point (C) and the maximum point (B) to point (D) on the graph as shown in Figure 4.2. Then by analysis, the slope of the graph remains unchanged, that is slope 1 = slope 2 (This simply means the normalized value remain unchanged), though the minimum and the maximum values on the graph change.

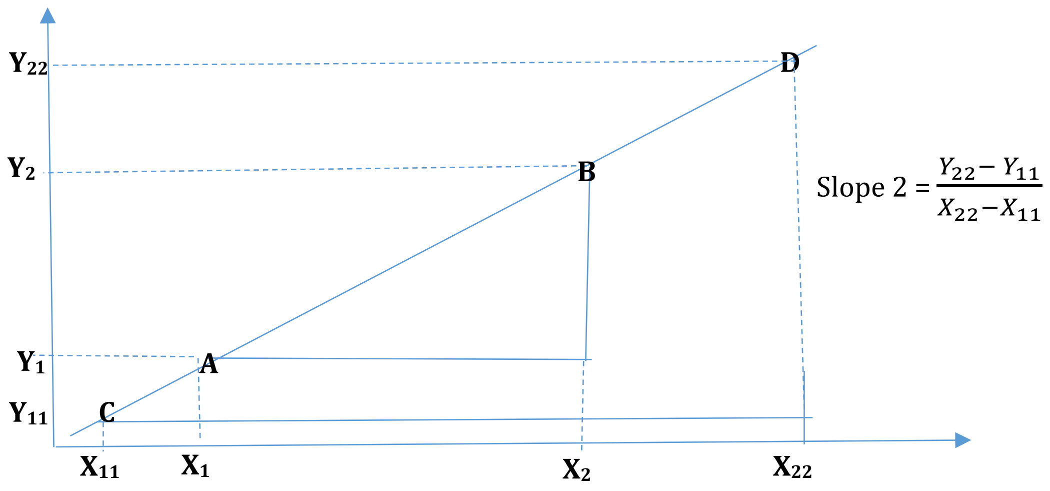


Figure 4.2: The slope of a straight line graph with notional extreme points.

Therefore, the extreme values of flowtime are now determined using the argument in favour of the corollary as follows;

i. The minimum possible total flowtime

The total flowtime

$\sum_{i=1}^{n} F_t = F_1 + F_2 + F_3 + F_4 + . . . + F_n$ (4.40)

Similarly, the flowtime is the total time a job spends in the shop. When the release date of a job is zero, the flowtime is the sum of the job waiting time (w_i) and the processing time (p_i) of the job.

$F_i = w_i + p_i$ (4.41)

The waiting time of a job can only be zero, if the job is scheduled in the position 1. Therefore, the minimum waiting time (besides the real waiting time from the SPT schedule) is zero and it can only be obtained by assuming that all the jobs are scheduled in position 1 (notional indeed since it is not real to schedule all the jobs in position 1 simultaneously). Also, the notional assumption is necessary since it is impossible to find any real schedule better than the optimal (the minimum flowtime from the SPT).

Therefore, the minimum waiting time is zero. $w_{min} = 0$ (4.42)

Therefore, the minimum possible flowtime, $F_{min} = P_i$ (4.43)

The minimum possible total flowtime is given by: $F_{tot}^{min} = \sum_{1}^{n} p_i$ (4.44)

ii. The maximum possible total flowtime

For any job scheduled in position, k, the waiting time is given by;

$w_k = \sum_{i=1}^{k-1} p_i \qquad k = 2,3,\dots,n$ (4.45)

Therefore, the waiting time is maximum when k = n. Where k is the job position and n is the number of job. Thus, the waiting time is maximum when job is scheduled in the position k. The maximum waiting time of each of the job is given by:

$w_{max} = \sum_{i=1}^{n-1} p_i$ (4.46)

Therefore, for consistency purpose, a notional maximum waiting time is also obtained by assuming that all the jobs are scheduled in the last position.

The maximum possible value of the flowtime of a job is given by;

$\sum_{i=1}^{n-1} p_i + p_i$ (4.47)

The maximum possible total flowtime of all the jobs in the schedule is given by;

$n\{\sum_{i=1}^{n-1} p_i\} \ + \ \sum_{i=1}^{n} p_i$ (4.48)

Equation (4.48) contains two components; the waiting time component and the process time component.

The waiting time component = $n\{\sum_{i=1}^{n-1} p_i\}$ (4.49)

In the waiting time component, it was assumed that the maximum waiting time is equal for all the jobs.

The processing time component = $\sum_{i=1}^{n} p_i$ (4.50)

In the processing time component, though the maximum possible waiting time is associated with each of the processing time, the processing time of each of the job remains unchanged.

Furthermore, the following points were found to support the corollary:

i. For a minimization scheduling problem for which a known optimal solution exists, there exists no real schedule that can yield a better result than the optimal solution. For an instance, there is no real schedule (not notional) better than the SPT algorithm schedule (optimal) for the scheduling problem of minimizing the total flow time of jobs with zero release dates.
ii. If the argument (i) is valid, then any other minimum possible flow time can only be a notional schedule. Thus, the corresponding maximum possible flow time must also be a notional schedule for consistency.
iii. Analysis with examples reveal that using the argument against the corollary, the extreme values and the value obtained from a near optimal heuristic upon which the normalization was carried out are all the same. Thus, the normalized value is indeterminate $(0/0)$.

    (See Appendix IV, for illustrative example to demonstrate the need for normalization and the validity of the proposed normalization procedure)

These points also apply to other scheduling problems for which optimal solutions exist. Therefore, the corollary is valid. The extreme values of the following criteria are determined using the corollary:

a. The completion time

The completion time (C_i) of job *i* is defined as:

$$C_i = F_i + r_i \tag{4.51}$$

Therefore, when the release date is zero, the flowtime is equal to the completion time. The SPT algorithm also yields optimal solution for scheduling problem of minimizing the total completion time under static condition. Thus, the extreme values of the completion time are equal to that of the flowtime.

The minimum possible value of total completion time is given by:

$$C_{tot}^{min} = F_{tot}^{min} = \sum_{1}^{n} p_i \tag{4.52}$$

The maximum possible value of total completion time is given by:

$$C_{tot}^{max} = n\{\sum_{i=1}^{n-1} p_i\} + \sum_{i=1}^{n} p_i \tag{4.53}$$

b. The total tardiness

A job is said to be tardy, if it is completed after its due date.

Total tardiness is given by: $T_{tot} = \sum_{i=1}^{n} T_i = \sum_{i=1}^{n} max\,\{0\,,(C_i - d_i)\}$ (4.54)

Since the tardiness can never be negative, it implies that the minimum possible value of total tardiness, $T_{min} = 0$.

The total tardiness is maximum when the completion time is maximum since the due date is an independent variable.

The maximum possible value of total tardiness is given by:

$$T_{tot}^{max} = \left(C_{tot}^{max} - \sum_{i=1}^{n} d_i\right) \tag{4.55}$$

c. The total lateness

Lateness is the difference between the completion time and the due date of the job.

Lateness is given by; $L_i = (C_i - d_i)$ (4.56)

Lateness is minimum when the completion time is minimum and it is maximum when the completion time is maximum because the due date is an independent variable. Thus, the extreme values of lateness are given by;

Minimum possible value of Total Lateness: $L_{tot}^{min} = (C_{tot}^{min} - \sum_{i=1}^{n} d_i)$ (4.57)

Maximum possible value of Total Lateness: $L_{tot}^{max} = (C_{tot}^{max} - \sum_{i=1}^{n} d_i)$ (4.58)

d. The number of tardy jobs

Recall that a job is said to be tardy, if it is completed after its due date. Evidently, the minimum possible number of tardy jobs is zero while the maximum possible number of tardy jobs is the total number of jobs.

e. The number of late jobs

This measures the number of jobs that are completed after the due dates. The sum of the number of late jobs and the number of early jobs is less than or equal to the total number of jobs. Thus when the number of late jobs is maximum, the number of early jobs is minimum (zero) and vice versa. Similarly, it is obvious that the minimum and maximum possible values of average number of tardy jobs ($N_{T\ avg}$) and average number of early jobs (N_{Eavg}) criteria are 0 and 1, respectively.

Similarly, the minimum possible number of on time jobs (jobs completed on the given due date) is zero while the maximum possible number of On time jobs is the total number of jobs. Table 4.2 shows the equations to determine the extreme values of some performance measures.

Table 4.2: The extreme values of some objective functions under static condition.

Objective function	minimum value	maximum value
The total flowtime	$\sum_{1}^{n} P_i$	$n\left\{\sum_{i=1}^{n-1} P_i\right\} + \sum_{i=1}^{n} P_i$
The total completion time	$\sum_{1}^{n} P_i$	$n\left\{\sum_{i=1}^{n-1} P_i\right\} + \sum_{i=1}^{n} P_i$
The total tardiness	0	$C_{tot}^{max} - \sum_{i=1}^{n} (d_i)$
The total lateness	$C_{tot}^{min} - \sum_{i=1}^{n} d_i$	$C_{tot}^{max} - \sum_{i=1}^{n} (d_i)$
The total earliness	$\sum_{i=1}^{n} d_i - C_{tot}^{max}$	$(\sum_{i=1}^{n} d_i - C_{tot}^{min})$
The total number of tardy jobs	0	N
The total number of late jobs	0	N
Average number of tardy jobs	0	1
Number of on time jobs	0	N

Therefore, once the normalized value of each criterion is known, the normalized composite objective function of the multicriteria problem is given by:

$$NTLCOF = 0.5\,(NT_X + NF_Y) \tag{4.59}$$

where:

$NTLCOF$ is the normalized total composite function,

N_X is the normalized value of criterion X, and

N_y is the normalized value of criterion Y.

The value of $NTLCOF$ will be used to assess the performance of the solution methods.

4.5 Solution Methods to the Bi-criteria Problems

The method used in this study involves implementation of the GAlg and the BB for the Class II and III problems. The heuristic A and heuristic B from the literature were also implemented for the Class II problem. For the Class III problem, the dynamic MDD heuristic was also implemented since no direct heuristic was found in the literature. Two heuristics were proposed for both classes.

4.5.1 Solution methods from the literature

The heuristics found in the literature for the Class II problem are outlined as follows;

i. Heuristic A: The jobs are arranged in EDD. Then, if there is a string of consecutive tardy jobs at the end of the sequence, the tardy jobs are rearranged in SPT.

ii. Heuristic B: The jobs are arranged in non-decreasing order of $p_i - d_i$. Then, if there is a string of consecutive tardy jobs at the end of the sequence, the tardy jobs are rearranged in SPT.

iii. The Generalized Algorithm (GAlg) for the Class II: The flowtime is minimised, using the SPT algorithm, while the total tardiness is minimised, using the DMDD rule.

Similarly, the heuristics implemented for the Class III problem are outlined as follows;

iv. The Generalized Algorithm (GAlg): The flowtime is minimised, using the KSAI algorithm (Oyetunji *et al.*, 2012), while the total tardiness is minimised, using the Dynamic Modified Due Date (DMDD) algorithm (Naidu, 2002).

v. The DMDD Algorithm: Though the algorithm was proposed for static and dynamic variants of scheduling problems for minimizing the total tardiness. The performance of the DMDD algorithm as well as the sparsity of heuristics that solve the Class III problem compelled the use of the heuristic for the bicriteria problem.

4.5.2 Application of the BB to solve the problem

The frontier search method was explored to branch, while the GAlg heuristic was used to bound the branching tree. Let s denote a partial sequence of jobs from among the *n* jobs originally in the problem. Also, let j(s) denote the partial sequence in which s is immediately preceded by job j. Associated with j(s) is a value, V_s which is the contribution of the assigned jobs in each level to the total normalized LCOF. The value of V_s is calculated for all the nodes at each level and compared to the lower bound obtained from the GAlg heuristic. Then, the nodes with V_s values higher than the lower bound value obtained from the GAlg are discarded, while other nodes are explored. The process continues until all the jobs are scheduled.

4.5.3 The Proposed solution methods for the Class II problem

Bi-criteria scheduling problems are NP-hard. Therefore, heuristic approach are desired to obtain good schedules. The two proposed heuristics; named HeuristicIII (HeuIII) and HeuristicIV (HeuIV) are now described.

i. HeuIII :

The logic behind this method is as follows;

The total flowtime scheduling problem with zero release dates on single processor can be solved optimally by the SPT algorithm, while the MDD algorithm yields a near-optimal solution for the equivalent problem but with the total tardiness as the performance measure.

Each of the criteria can be expressed mathematically as follows;

Total Tardiness (T_{tot}) : $\sum_{i=1}^{n} T_i = \sum_{i=1}^{n} max\ \{0\ ,(C_i - d_i)\}$ (4.60)

Total Flowtime (F_{tot}) : $\sum_{i=1}^{n} F_i = \sum_{i=1}^{n} C_i - r_i$ (4.61)

For $r_i = 0$

Total Flowtime (F_{tot}): $\sum_{i=1}^{n} F_i = \sum_{i=1}^{n} C_i$ (4.62)

Analyses of the two algorithms reveal that the MDD algorithm favours the EDD rule at the beginning of the schedule, while towards the tail end of the schedule, the algorithm is biased towards the SPT rule. Thus, if the MDD and the SPT algorithms yield the same schedule, the two solutions are either optimal or near-optimal solutions. Furthermore, if the two solutions differ, the optimal schedule is a close neighbour to the schedules obtained. The algorithm is as follows:

Initialization

JobSet A = [J_1, J_2, J_3, …….J_n], set of given jobs

JobSet B = [0], set of scheduled job

JobSet C = [J_1', J_2', J_3', …….J_n'], set of unscheduled jobs, $J_j' = J_j$

STEP 1: Form JobSet D by arranging the JobSet A in the order of non-decreasing processing time. If there is a tie, break it with the due date

STEP2: Form JobSet E by exploring the modified due date algorithm.

STEP 3: Compute the LCOF function; LCOF1 and LCOF2 for JobSet D and JobSet E, respectively.

STEP 4: Set JobSet E as the required schedule (JobSet B), if LCOF2 is less than or equal to LCOF1, otherwise, set JobSet D as the required schedule

STEP 5: Compute the objective function of the required schedule

STEP 6: Stop.

ii. HeuIV

This algorithm is based on the two independent variables upon which the two criteria depend. These are the processing time and the due date. Furthermore, the SPT rule and the MDD algorithm yield the optimal and the near optimal schedules for the total flowtime and the total tardiness, respectively. The SPT rule explored the completion time, the MDD algorithm utilized the modified due date (called the priority index). Therefore, the two independent variables; the completion time and the modified due date can also be linked to the LCOF of the two criteria. The completion time depends on the processing time only, while the modified due date is a function of both the processing time and the due date. Thus, the processing time is the dominant independent variable. In this regard, the processing time is used to schedule the jobs, while the modified due date is used to break the tie in the processing time of a job i at time t. The algorithm is as follows:

Initialization

JobSet A = [J_1, J_2, J_3, …….J_n], set of given jobs

JobSet B = [0], set of scheduled job

JobSet C = [J_1’, J_2’, J_3’, …….J_n’], set of unscheduled jobs, J_j’ = J_j

STEP 1: Arrange the JobSet A in the order of increasing processing time

STEP2: Break the tie in step 1 by using the modified due date rule. If tie still exist, break arbitrarily

STEP 2: Compute the objective function of the schedule

STEP 3: Stop.

4.5.4 Proposed solution methods for the Class III problem

Two heuristics (HeuV and HeuVI) are proposed for this problem. The heuristics are now described.

i. HeuV

In order to minimise the composite function of total flowtime and total tardiness on a single processor with non-zero release dates, three parameters are involved; the processing time, the release date and the due date. The processing time and the release date affect the two objectives, while the due date affects only the tardiness. In this regard, the index defined as the sum of the due date, processing time and the release date (d+p+r) is used as the pivot to schedule the jobs, while all the other parameters are used to break the tie consecutively. The statement of the algorithm is as follows:

STEP 1: Initialization

JobSet A = [J_1, J_2, J_3, …….J_n], set of given jobs

JobSet B = [0], set of scheduled jobs

JobSet C = [J_1’, J_2’, J_3’, …….J_n’], set of unscheduled jobs, J_j’ = J_j, n=number of jobs

STEP 2: Compute the index defined as the sum of the due date, processing time and the release dates ($di+p_i+r_i$) for each of the jobs in JobSet A

STEP 3: Arrange the jobs in JobSet A in the order of non- decreasing index computed in Step 2 and put the jobs in JobSet B

STEP 4: If there is a tie in JobSet B, break the tie using the release date, processing time and the due date in that order. If tie still exists, break arbitrarily

STEP 5: Stop.

ii. HeuVI

This is a modification of the HeuV. It is based on the fact that the effect of the release dates and the processing time is more significant on the linear composite of objective function than the due date variable. This is because the processing time and the release date affect the two objectives, while the due date has an effect on the tardiness only. The steps of the heuristic now follows;

STEP 1: Initialization

JobSet A = [J_1, J_2, J_3, …….J_n], set of given jobs

JobSet B = [0], set of scheduled jobs

JobSet C = [J_1’, J_2’, J_3’, …….J_n’], set of unscheduled jobs, J_j’ = J_j , n=number of jobs

STEP 2: Compute the index defined as the sum of the processing time and the release date (p_i+r_i) for each of the jobs in JobSet A

STEP 3: Arrange the jobs in JobSet A in the order of increasing index computed in Step 2 and put the jobs in JobSet C

STEP 4: Arrange the jobs in JobSet A in the order of increasing release dates and put the jobs in JobSet D

STEP 5: Compute the LCOF function; LCOF1 and LCOF2 for both JobSet C and JobSet D, respectively

STEP 6: Set JobSet D as the required schedule (JobSet B), if LCOF2 is less than or equal to LCOF1. Otherwise, set JobSet C as the required schedule

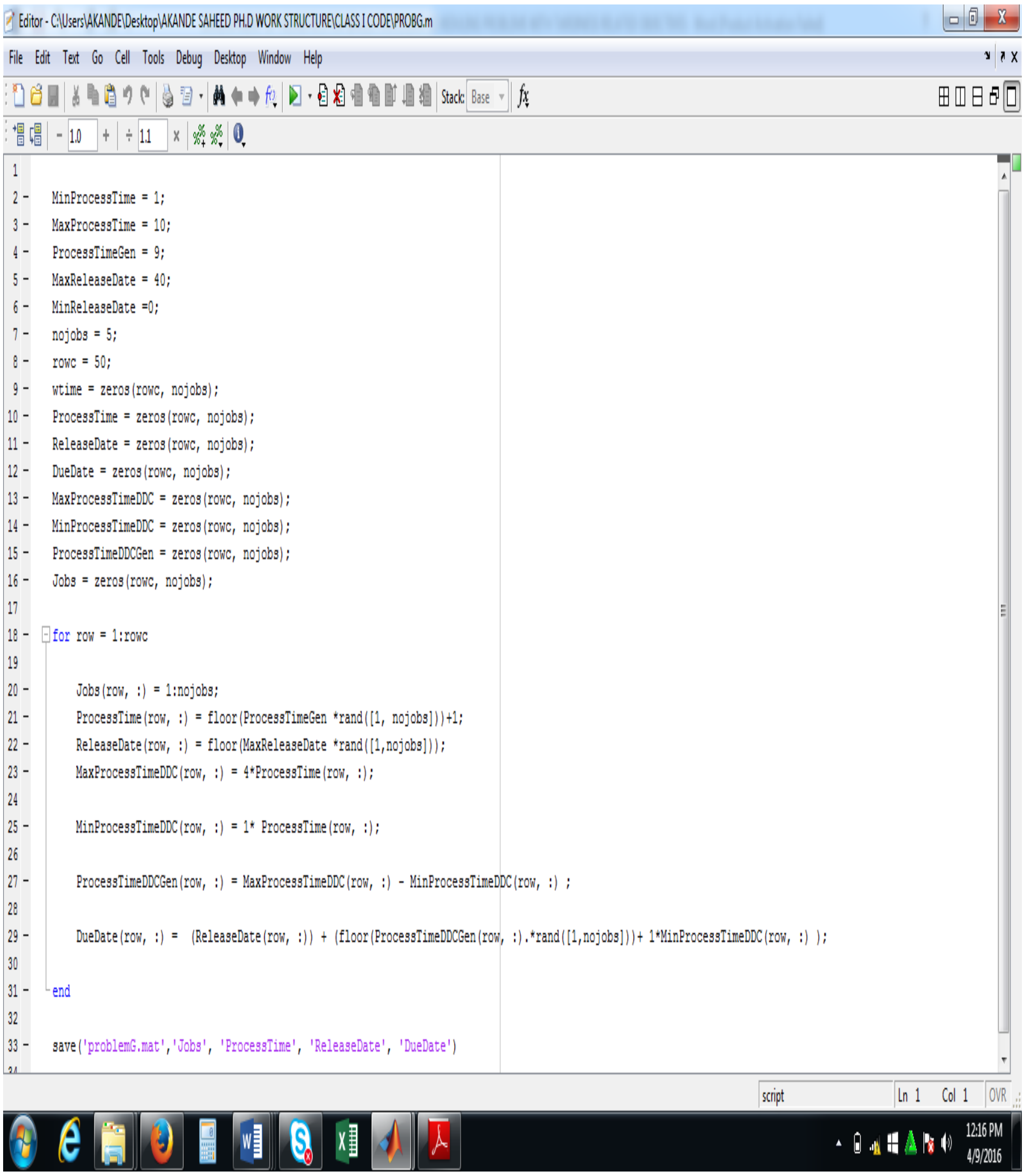
STEP 7: Stop.

4.6 Model Implementation

The proposed as well as the selected solution methods for each problem class were implemented to solve some single processor scheduling problems. These include the randomly generated problems and the real life problems collected from a firm.

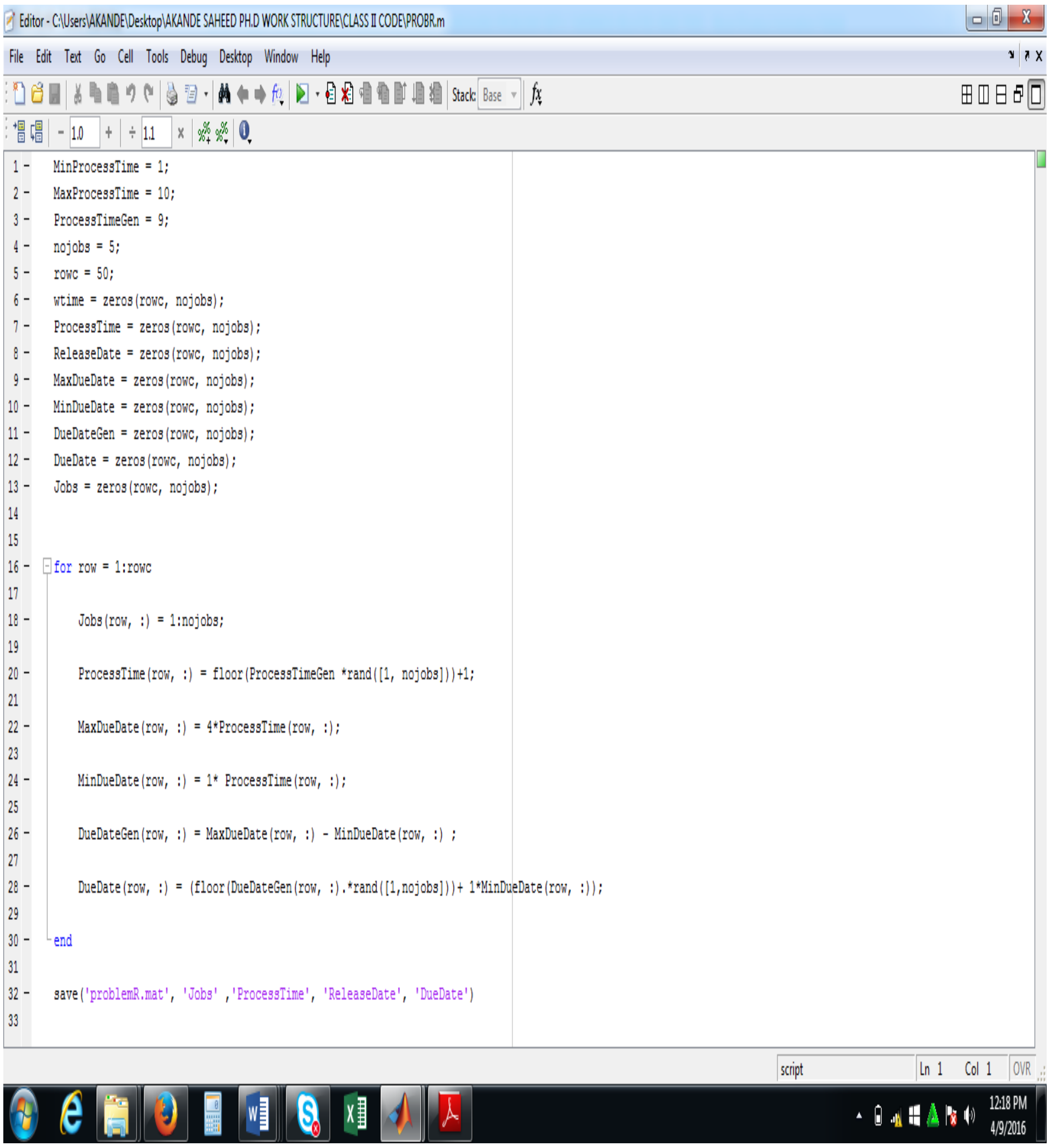
4.6.1 Random problem generation

The Gursel *et al.* (2012) concept was explored to generate a single processor scheduling problems randomly. The relation was explained in section 3.1. The random problem generation codes (written in MATLAB) is the same for the two dynamic environment problems (Class I and III) but differ for the static problem (Class II). This is because the release date is a variable in the former but zero in the latter. The random problem generation file was saved as the PROBG and the PROBR for the dynamic and static environment, respectively. The Figure 4.3 and 4.4 shows the screen shot of the PROBG and the PROBR



```
MinProcessTime = 1;
MaxProcessTime = 10;
ProcessTimeGen = 9;
MaxReleaseDate = 40;
MinReleaseDate =0;
nojobs = 5;
rowc = 50;
wtime = zeros(rowc, nojobs);
ProcessTime = zeros(rowc, nojobs);
ReleaseDate = zeros(rowc, nojobs);
DueDate = zeros(rowc, nojobs);
MaxProcessTimeDDC = zeros(rowc, nojobs);
MinProcessTimeDDC = zeros(rowc, nojobs);
ProcessTimeDDCGen = zeros(rowc, nojobs);
Jobs = zeros(rowc, nojobs);

for row = 1:rowc

    Jobs(row, :) = 1:nojobs;
    ProcessTime(row, :) = floor(ProcessTimeGen *rand([1, nojobs]))+1;
    ReleaseDate(row, :) = floor(MaxReleaseDate *rand([1,nojobs]));
    MaxProcessTimeDDC(row, :) = 4*ProcessTime(row, :);

    MinProcessTimeDDC(row, :) = 1* ProcessTime(row, :);

    ProcessTimeDDCGen(row, :) = MaxProcessTimeDDC(row, :) - MinProcessTimeDDC(row, :) ;

    DueDate(row, :) =  (ReleaseDate(row, :)) + (floor(ProcessTimeDDCGen(row, :).*rand([1,nojobs]))+ 1*MinProcessTimeDDC(row, :) );

end

save('problemG.mat','Jobs', 'ProcessTime', 'ReleaseDate', 'DueDate')
```

Figure 4.3: Random generation code for the Class I and Class III (PROBG)


Editor - C:\Users\AKANDE\Desktop\AKANDE SAHEED PH.D WORK STRUCTURE\CLASS II CODE\PROBR.m
File Edit Text Go Cell Tools Debug Desktop Window Help
Stack: Base
script Ln 1 Col 1 OVR
12:18 PM
4/9/2016

```matlab
MinProcessTime = 1;
MaxProcessTime = 10;
ProcessTimeGen = 9;
nojobs = 5;
rowc = 50;
wtime = zeros(rowc, nojobs);
ProcessTime = zeros(rowc, nojobs);
ReleaseDate = zeros(rowc, nojobs);
MaxDueDate = zeros(rowc, nojobs);
MinDueDate = zeros(rowc, nojobs);
DueDateGen = zeros(rowc, nojobs);
DueDate = zeros(rowc, nojobs);
Jobs = zeros(rowc, nojobs);


for row = 1:rowc

    Jobs(row, :) = 1:nojobs;

    ProcessTime(row, :) = floor(ProcessTimeGen *rand([1, nojobs]))+1;

    MaxDueDate(row, :) = 4*ProcessTime(row, :);

    MinDueDate(row, :) = 1* ProcessTime(row, :);

    DueDateGen(row, :) = MaxDueDate(row, :) - MinDueDate(row, :) ;

    DueDate(row, :) = (floor(DueDateGen(row, :).*rand([1,nojobs]))+ 1*MinDueDate(row, :));

end

save('problemR.mat', 'Jobs' ,'ProcessTime', 'ReleaseDate', 'DueDate')

```

Figure 4.4: Screen shot of a Random problem generation code for the Class II

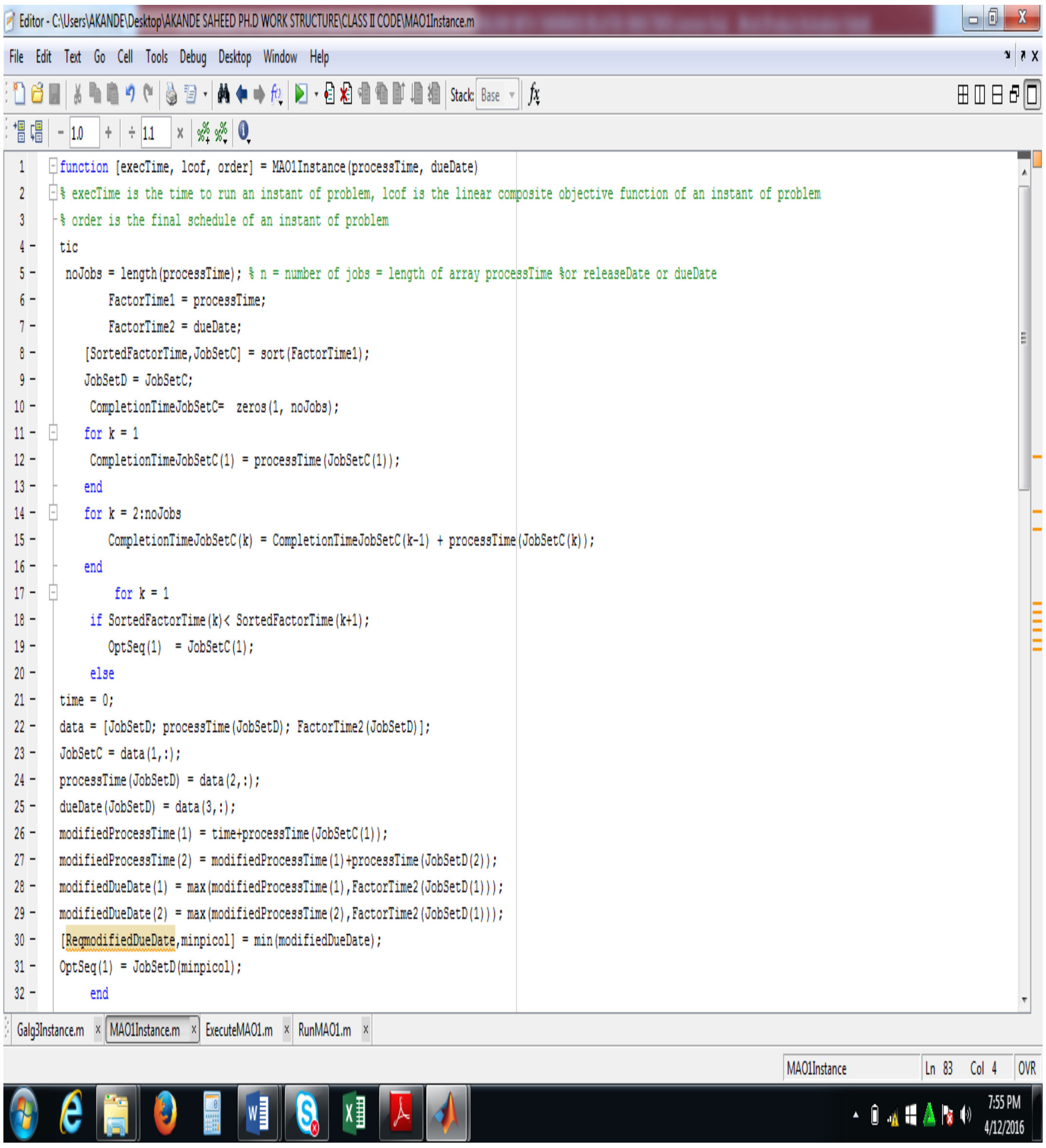
The generated problems were solved by the implemented solution methods. The coding of the solution on Matlab Programing was executed with the following files;

i. The single instance file, which solves an instance of the problem,

ii. The execution file, where the number of instances required for any size problem was loaded and solved to obtained the mean value of the objective function, and

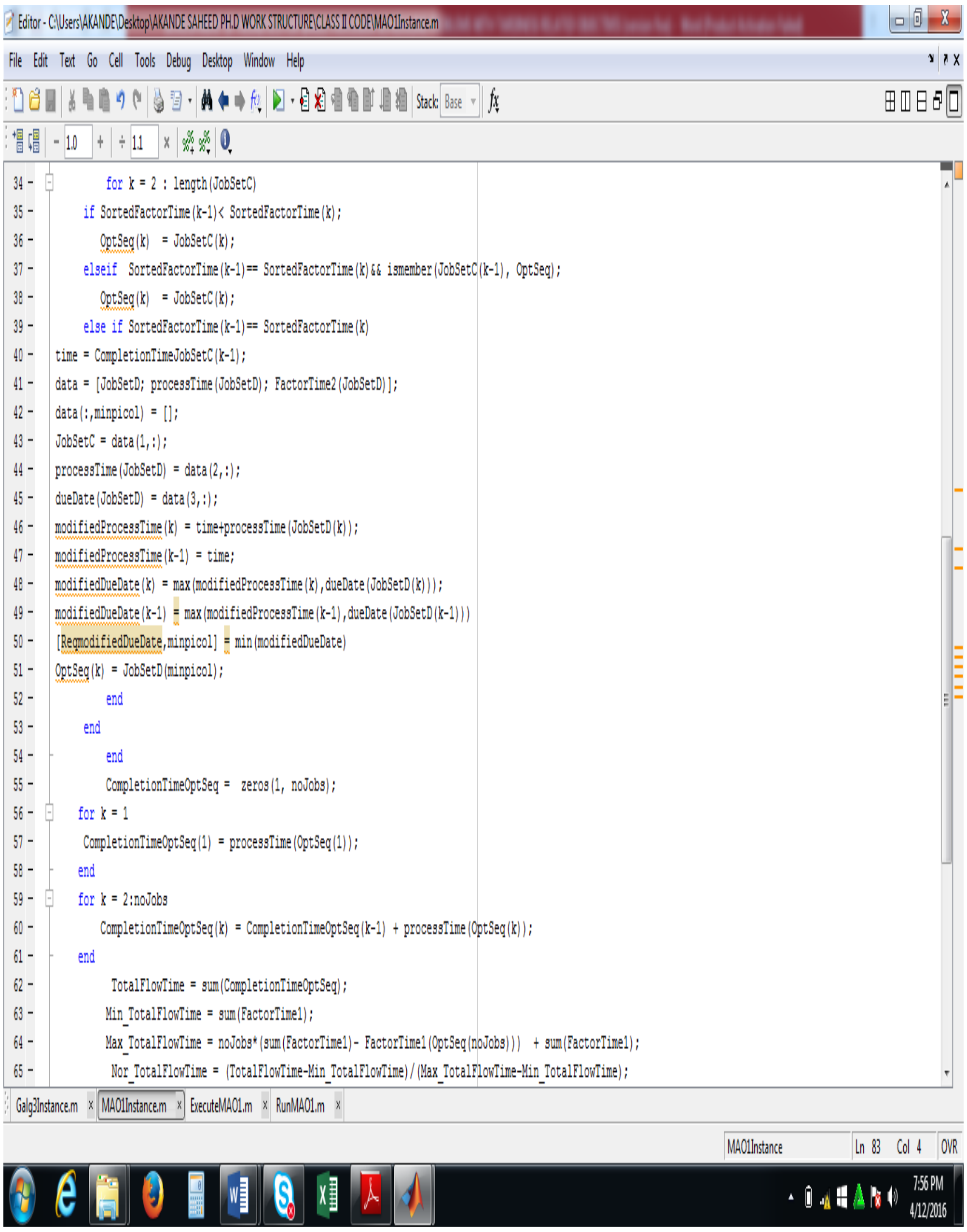
iii. The run problem file, where the single instant file and the execution file were loaded and run to give the output.

Figure 4.5- 4.7 show the screen shot of a single instance file, the execution file, and the run problem file for the HeuIV proposed for the Class II problem, respectively.



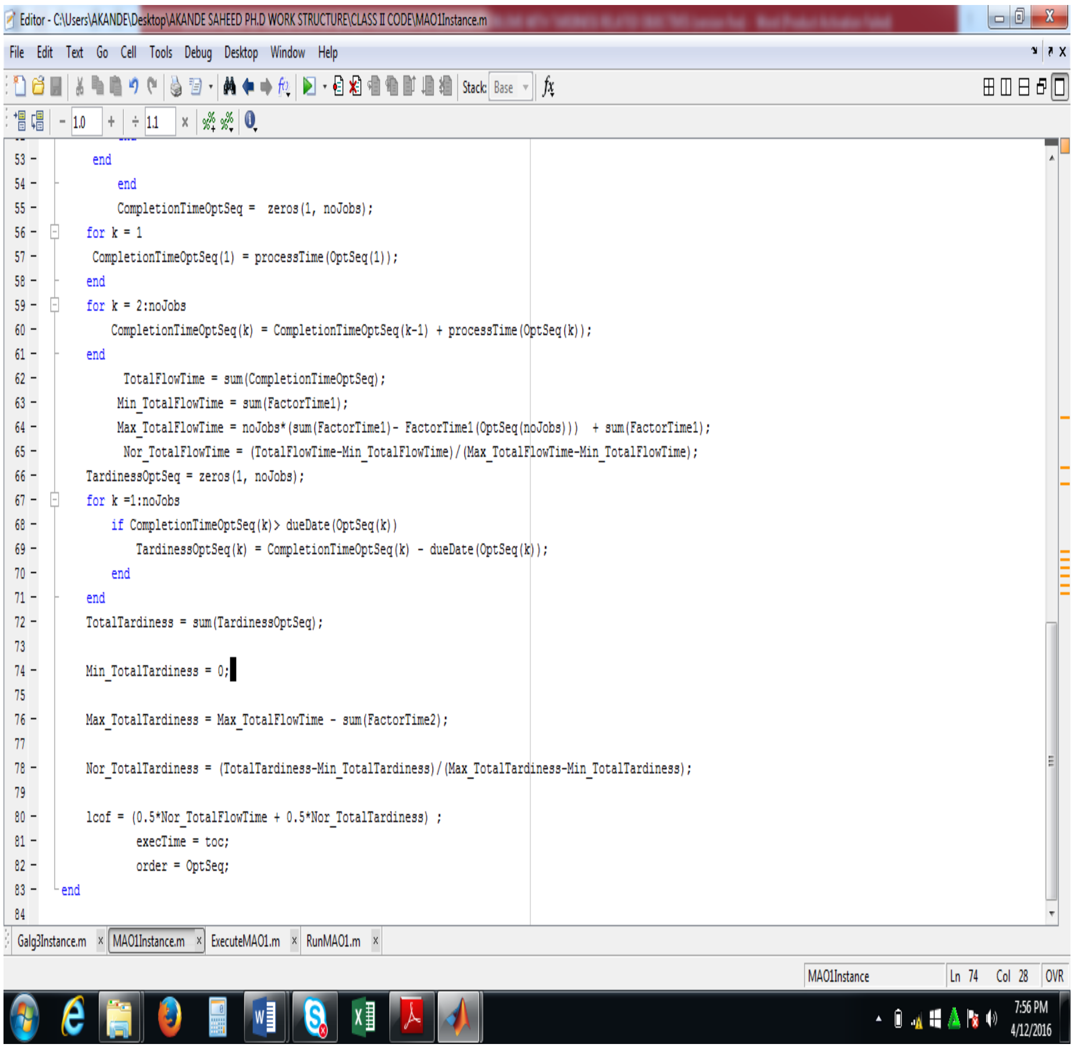
```
function [execTime, lcof, order] = MAO1Instance(processTime, dueDate)
% execTime is the time to run an instant of problem, lcof is the linear composite objective function of an instant of problem
% order is the final schedule of an instant of problem
tic
 noJobs = length(processTime); % n = number of jobs = length of array processTime %or releaseDate or dueDate
        FactorTime1 = processTime;
        FactorTime2 = dueDate;
    [SortedFactorTime,JobSetC] = sort(FactorTime1);
    JobSetD = JobSetC;
     CompletionTimeJobSetC=  zeros(1, noJobs);
    for k = 1
     CompletionTimeJobSetC(1) = processTime(JobSetC(1));
    end
    for k = 2:noJobs
        CompletionTimeJobSetC(k) = CompletionTimeJobSetC(k-1) + processTime(JobSetC(k));
    end
         for k = 1
     if SortedFactorTime(k)< SortedFactorTime(k+1);
        OptSeq(1)  = JobSetC(1);
     else
time = 0;
data = [JobSetD; processTime(JobSetD); FactorTime2(JobSetD)];
JobSetC = data(1,:);
processTime(JobSetD) = data(2,:);
dueDate(JobSetD) = data(3,:);
modifiedProcessTime(1) = time+processTime(JobSetC(1));
modifiedProcessTime(2) = modifiedProcessTime(1)+processTime(JobSetD(2));
modifiedDueDate(1) = max(modifiedProcessTime(1),FactorTime2(JobSetD(1)));
modifiedDueDate(2) = max(modifiedProcessTime(2),FactorTime2(JobSetD(1)));
[ReqmodifiedDueDate,minpicol] = min(modifiedDueDate);
OptSeq(1) = JobSetD(minpicol);
     end
```

Figure 4.5a: Single instance file of the HeuIV Heuristic


Editor - C:\Users\AKANDE\Desktop\AKANDE SAHEED PH.D WORK STRUCTURE\CLASS II CODE\MAO1Instance.m
File Edit Text Go Cell Tools Debug Desktop Window Help
Stack: Base
Galg3Instance.m
MAO1Instance.m
ExecuteMAO1.m
RunMAO1.m
MAO1Instance
Ln 83 Col 4 OVR
7:56 PM
4/12/2016

```
          for k = 2 : length(JobSetC)
      if SortedFactorTime(k-1)< SortedFactorTime(k);
         OptSeq(k)  = JobSetC(k);
      elseif  SortedFactorTime(k-1)== SortedFactorTime(k)&& ismember(JobSetC(k-1), OptSeq);
         OptSeq(k)  = JobSetC(k);
      else if SortedFactorTime(k-1)== SortedFactorTime(k)
time = CompletionTimeJobSetC(k-1);
data = [JobSetD; processTime(JobSetD); FactorTime2(JobSetD)];
data(:,minpicol) = [];
JobSetC = data(1,:);
processTime(JobSetD) = data(2,:);
dueDate(JobSetD) = data(3,:);
modifiedProcessTime(k) = time+processTime(JobSetD(k));
modifiedProcessTime(k-1) = time;
modifiedDueDate(k) = max(modifiedProcessTime(k),dueDate(JobSetD(k)));
modifiedDueDate(k-1) = max(modifiedProcessTime(k-1),dueDate(JobSetD(k-1)))
[ReqmodifiedDueDate,minpicol] = min(modifiedDueDate)
OptSeq(k) = JobSetD(minpicol);
          end
      end
          end
          CompletionTimeOptSeq =  zeros(1, noJobs);
     for k = 1
      CompletionTimeOptSeq(1) = processTime(OptSeq(1));
     end
     for k = 2:noJobs
         CompletionTimeOptSeq(k) = CompletionTimeOptSeq(k-1) + processTime(OptSeq(k));
     end
           TotalFlowTime = sum(CompletionTimeOptSeq);
          Min_TotalFlowTime = sum(FactorTime1);
          Max_TotalFlowTime = noJobs*(sum(FactorTime1)- FactorTime1(OptSeq(noJobs)))  + sum(FactorTime1);
           Nor_TotalFlowTime = (TotalFlowTime-Min_TotalFlowTime)/(Max_TotalFlowTime-Min_TotalFlowTime);
```

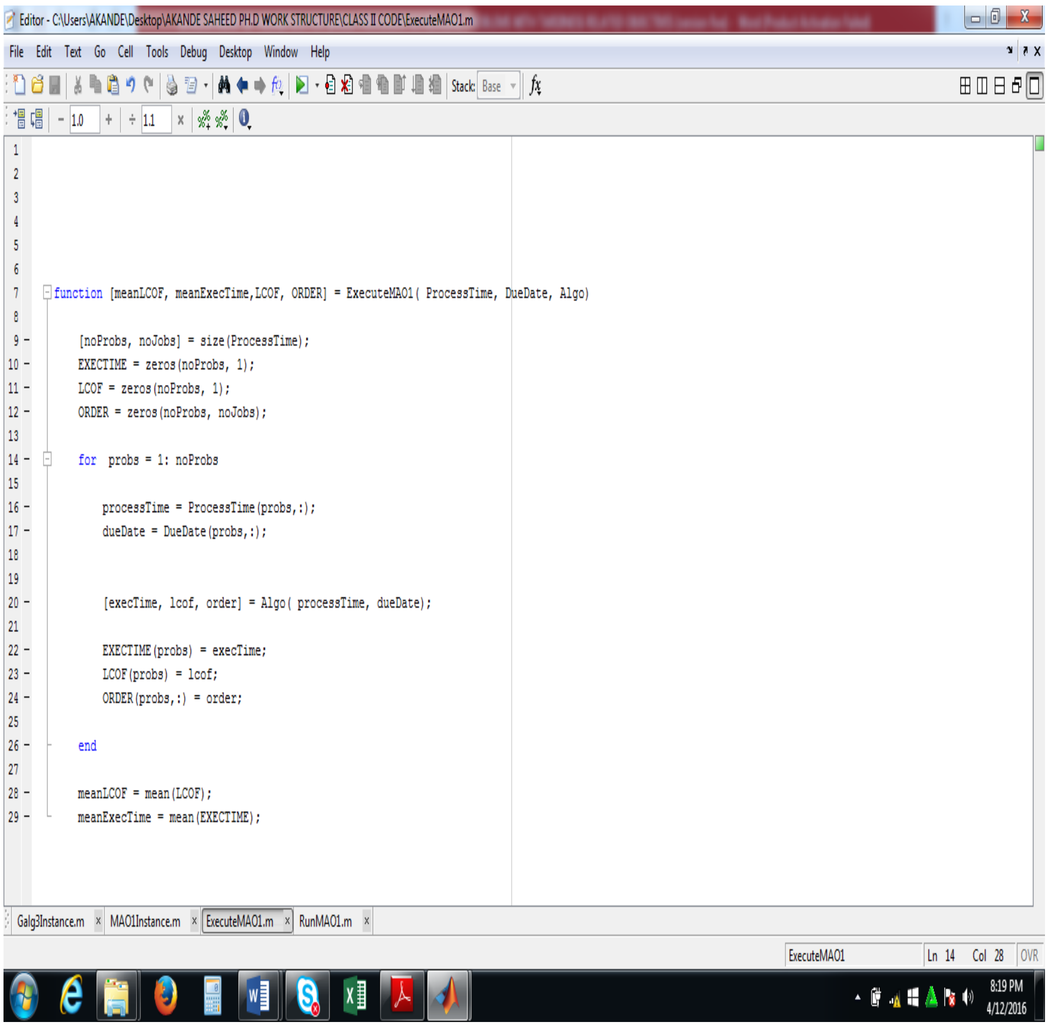
Figure 4.5b : Single instance file of the HeuIV



```
     end
         end
         CompletionTimeOptSeq =  zeros(1, noJobs);
    for k = 1
     CompletionTimeOptSeq(1) = processTime(OptSeq(1));
    end
    for k = 2:noJobs
        CompletionTimeOptSeq(k) = CompletionTimeOptSeq(k-1) + processTime(OptSeq(k));
    end
          TotalFlowTime = sum(CompletionTimeOptSeq);
         Min_TotalFlowTime = sum(FactorTime1);
         Max_TotalFlowTime = noJobs*(sum(FactorTime1)- FactorTime1(OptSeq(noJobs)))  + sum(FactorTime1);
          Nor_TotalFlowTime = (TotalFlowTime-Min_TotalFlowTime)/(Max_TotalFlowTime-Min_TotalFlowTime);
    TardinessOptSeq = zeros(1, noJobs);
    for k =1:noJobs
        if CompletionTimeOptSeq(k)> dueDate(OptSeq(k))
           TardinessOptSeq(k) = CompletionTimeOptSeq(k) - dueDate(OptSeq(k));
        end
    end
    TotalTardiness = sum(TardinessOptSeq);

    Min_TotalTardiness = 0;

    Max_TotalTardiness = Max_TotalFlowTime - sum(FactorTime2);

    Nor_TotalTardiness = (TotalTardiness-Min_TotalTardiness)/(Max_TotalTardiness-Min_TotalTardiness);

    lcof = (0.5*Nor_TotalFlowTime + 0.5*Nor_TotalTardiness) ;
           execTime = toc;
           order = OptSeq;
end
```

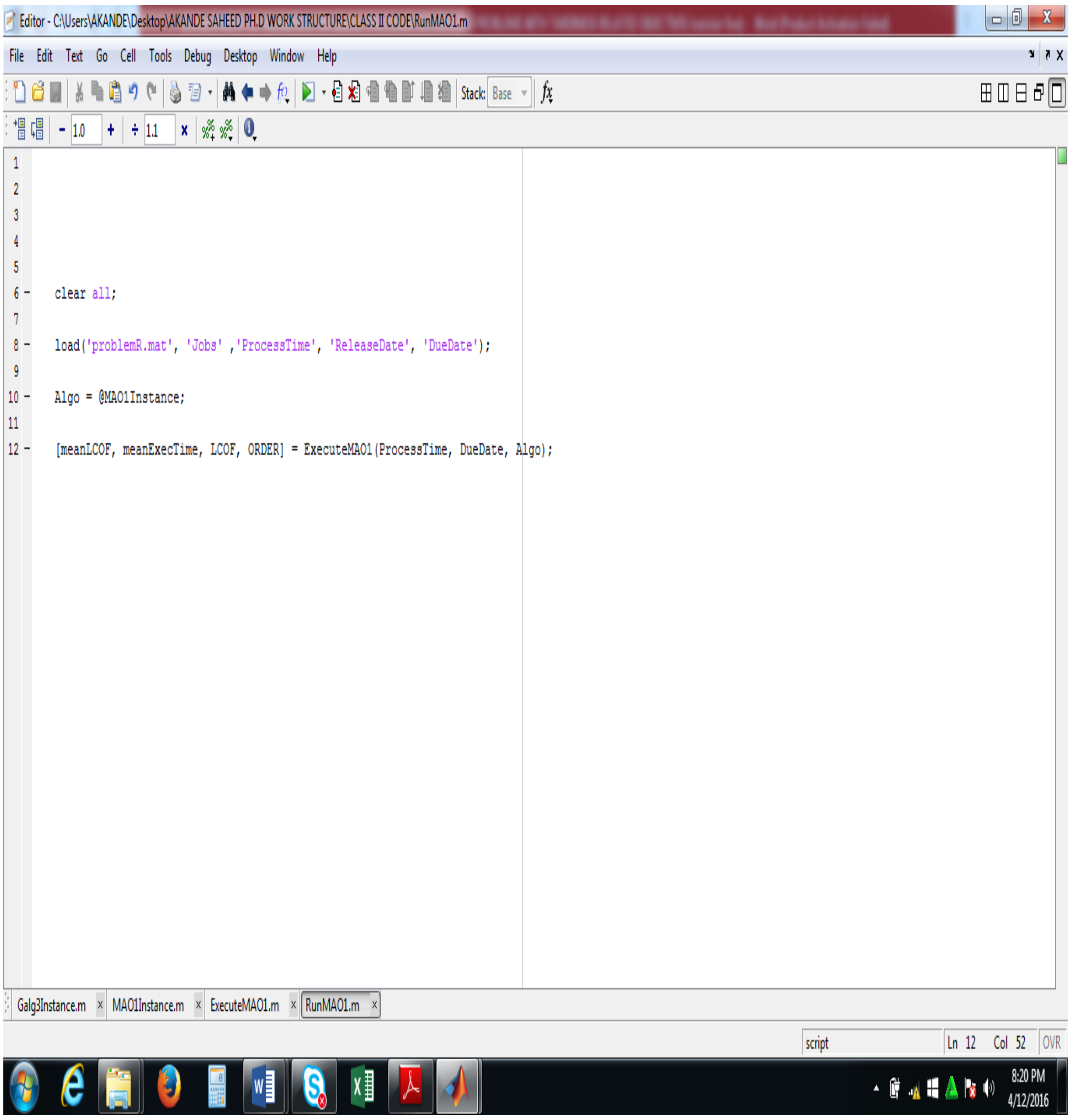
Editor - C:\Users\AKANDE\Desktop\AKANDE SAHEED PH.D WORK STRUCTURE\CLASS II CODE\MAO1Instance.m
File Edit Text Go Cell Tools Debug Desktop Window Help
Stack: Base
1.0
1.1
Galg3Instance.m
MAO1Instance.m
ExecuteMAO1.m
RunMAO1.m
MAO1Instance
Ln 74 Col 28 OVR
7:56 PM
4/12/2016

Figure 4.5c: Single instance file of the HeuIV


Editor - C:\Users\AKANDE\Desktop\AKANDE SAHEED PH.D WORK STRUCTURE\CLASS II CODE\ExecuteMAO1.m
File Edit Text Go Cell Tools Debug Desktop Window Help
Stack: Base
1.0
1.1
Galg3Instance.m
MAO1Instance.m
ExecuteMAO1.m
RunMAO1.m
ExecuteMAO1
Ln 14 Col 28 OVR
8:19 PM
4/12/2016

```matlab






function [meanLCOF, meanExecTime,LCOF, ORDER] = ExecuteMAO1( ProcessTime, DueDate, Algo)

    [noProbs, noJobs] = size(ProcessTime);
    EXECTIME = zeros(noProbs, 1);
    LCOF = zeros(noProbs, 1);
    ORDER = zeros(noProbs, noJobs);

    for  probs = 1: noProbs

        processTime = ProcessTime(probs,:);
        dueDate = DueDate(probs,:);


        [execTime, lcof, order] = Algo( processTime, dueDate);

        EXECTIME(probs) = execTime;
        LCOF(probs) = lcof;
        ORDER(probs,:) = order;

    end

    meanLCOF = mean(LCOF);
    meanExecTime = mean(EXECTIME);
```

Figure 4.6: The execution file of the HeuIV


Editor - C:\Users\AKANDE\Desktop\AKANDE SAHEED PH.D WORK STRUCTURE\CLASS II CODE\RunMAO1.m
File Edit Text Go Cell Tools Debug Desktop Window Help
Galg3Instance.m
MAO1Instance.m
ExecuteMAO1.m
RunMAO1.m
script
Ln 12 Col 52
OVR
8:20 PM
4/12/2016

```
clear all;

load('problemR.mat', 'Jobs' ,'ProcessTime', 'ReleaseDate', 'DueDate');

Algo = @MAO1Instance;

[meanLCOF, meanExecTime, LCOF, ORDER] = ExecuteMAO1(ProcessTime, DueDate, Algo);
```

Figure 4.7: The execution file of the HeuIV

4.7 The Real life Problem Size and the Current Firm Scheduling Policy

The data of received jobs for a period of four months were collected from a firm as discussed in section 3.3. Jobs customer details form where the data were obtained were pooled together for each month. The problem sizes varied depending on the number of job accumulation under consideration. For 5, 10, and 15 jobs accumulation, the problem sizes were determined by using the first 5, 10, and 15 jobs for each month. The baking and spraying unit of the company schedule the jobs using the First Come, First Served (FCFS), if the jobs on the queue were not received on the same date. This implies that jobs are arranged in the order of increasing release date**.** However, if the jobs on the queue were available on the same date, the firm explores the Shortest Processing Time (SPT) rule. This implies that jobs are arranged in the order of non-decreasing processing times.

Therefore, for the Class I and Class III problems, the implemented heuristics as well as the firm existing scheduling policy (FCFS) were coded and applied on the data. The same procedure was also carried out on the Class II with the Shortest Processing Time (SPT) rule being explored by the firm. However, a slight modification was made in the case of the Class II; the release dates of all the pooled jobs are set to zero.

The information gathered from the firm reveal that a tolerance of 1 to 2 days is usually given to jobs that require a maximum of two to three days to complete. Also, for rentage above five days, a week (5-7 days) tolerance is usually added. This is used to compute the due date from the processing time.

CHAPTER FIVE

RESULTS AND DISCUSSION

5.1 Introduction

The results obtained in solving the simulated and the real life problems by the implemented solution methods are discussed in this chapter. It involves evaluating the effectiveness and the efficiency of the solution methods through various comparative analyses of the objective function and the execution time, respectively.

5.2 Results Based on the Total Tardiness with Release Dates (2TRD)

The problem of minimizing the total tardiness of jobs with release dates on a single processor was considered as the Class I. The results are now presented and discussed.

5.2.1 Results of the selected heuristics from the literature

The simulated problems were solved by the heuristics selected among the solution methods found in the literature. Table 5.1 shows the mean of the total tardiness obtained.

Based on the results in Table 5.1, it can be inferred that for the problem ranges; $0 \leq n \leq 60$, the DMDD heuristic produced a better solutions, while for ranges; $80 \leq n \leq 1000$, the SPT rule performed better. Therefore, the DMDD and SPT heuristics were selected for comparison to the proposed heuristics and the optimal solution from the BB procedure.

Table 5.1: Mean of the total tardiness by solution methods and problem sizes

S/N	Problem sizes	MST	DMDD	EDD	ODD	SPT
1	5 x 1	48.64	2.82	2.82	48.64	38.42
2	7x1	95.25	15.12	15.12	95.25	79.56
3	8x1	137.45	17.4	18.1	137.45	112.62
4	10 x 1	236.82	44.75	45.82	236.82	175.25
5	15x1	498.16	169.32	171.72	498.16	409.18
6	20 x 1	961.94	427.5	413.16	961.94	728
7	25x1	1563	770.28	778.94	1563.2	1175.25
8	30x1	2202	1230	1241.75	2202	1705.15
9	40x1	3855.25	2450.2	2428	3855.25	2946
10	60x1	8803.35	6206	6268.2	8803	6521
11	80 x1	15848.2	11708.2	11833	15848	11534
12	100 x1	24571.1	18742.5	18970	24571.8	17981.7
13	150 x1	56134.1	44276.5	44776.1	56134	40324
14	200 x1	99492.1	80851.3	81742.2	99492.6	71485.2
15	300 x1	225690	186460	188840	225690	161070
16	400 x1	399540	334130	338160	399540	283210
17	500 x1	621480	519030	525160	621480	439550
18	1000 x1	2502600	2128400	2155700	2502600	1768400

5.2.2 Results of the proposed heuristics

The simulated problems solved by the selected solution methods from the literature were also implemented for the proposed heuristics and the BB method. Table 5.2 shows the total tardiness of the proposed heuristics, the BB algorithm and the two selected approaches from literature.

Table 5.2: Mean of the total tardiness (2TRD) by solution methods and problem sizes

		Proposed Heuristics		Existing Heuristics		Optimal
No	Sizes	HeuI	HeuII	DMDD	SPT	BB
1	5 x 1	4.32	0.96	2.82	38.42	0.26
2	7x1	12.5	5.96	15.12	79.56	4.98
3	8x1	14.54	12.00	17.4	112.62	10.25
4	10 x 1	33.9	29.18	44.75	175.25	21.56
	Mean	**16.32**	**12.03**	**20.0225**	**101.463**	**9.26**
5	15x1	150.46	146.2	169.32	409.1	84.14
6	20 x 1	407.74	394.25	407.5	728.2	285.5
7	25x1	768.25	757	770.28	1175.35	491.68
	Mean	**442.15**	**432.48**	**449.03**	**770.91**	**287.11**
8	30x1	1115.48	1245.1	1230.14	1705.78	
9	40x1	2410.00	2550.04	2450.00	2946.92	
10	60x1	5816.89	6462.00	6206.28	6521.75	
11	80 x1	11040.52	12162.10	11708.50	11533.10	
12	100 x1	16960.65	19522.00	18742.40	17981.90	
13	150 x1	39134.35	45759.10	44276.20	40324.50	
14	200 x1	70572.20	83296.10	80851.20	71485.10	
15	300 x1	160210.00	192230.00	186460.00	161070.00	
16	400 x1	282150.54	344000.00	334130.00	283210.00	
17	500 x1	438690.26	533700	519031	439550	
18	1000x1	1766400.48	2182100	2128401	1768401	
	Mean	**254045.56**	**311184**	**303044**	**254976**	

Table 5.2 shows that for the small-sized ($5 \leq n \leq 10$) and medium-sized ($10 \leq n \leq 25$) problems, the HeuII yielded a better result besides the optimal solution method (BB). Similarly, for the large-sized problems; $30 \leq n \leq 1000$, the HeuI produced a better results compared to other solution methods. The results were subjected to analytical tests in order to measure the effectiveness of the solution methods for proper ranking. Also, a high standard deviation implies that the values of objective function of problem instances (in such problem size) shows a larger difference as the number of jobs increases. This also justified the combinatorial nature of scheduling problems.

5.2.3 Approximation ratio test

This test shows the closeness of the heuristics to the optimal or the standard solution method. It gives the overall measure of how many times the optimal or the standard solution performed better than other implemented solutions. The BB results (the optimal) were used for benchmarking the small-sized and medium-sized results while the HeuI results were used as a standard solution for the problem ranges; $30 \leq n \leq 1000$. (See APPENDIX III, Table B, C and D for the approximation ratio Tables for the three problem sizes).

Figure 5.1 shows the plots of approximation ratios for all the solution methods in the problem ranges; $5 \leq n \leq 10$. The results shows that the heuristics are closer to the BB plot in the ranking order; HeuII, DMDD, HeuI, SPT. The Figure also shows that as the problem size increases, the differences between the heuristics decreases. Figure 5.2 shows the plots of approximation ratios for the solution methods for the medium-sized problems; $10 \leq n \leq 25$. The Figure shows that the HeuII outperformed the DMDD heuristic for the medium-sized problems. Therefore, the solution methods are closer to the BB plot in the ranking order; HeuII, HeuI, DMDD, SPT. The Figure also shows that as the problem size increases, the differences between the heuristics decreases.

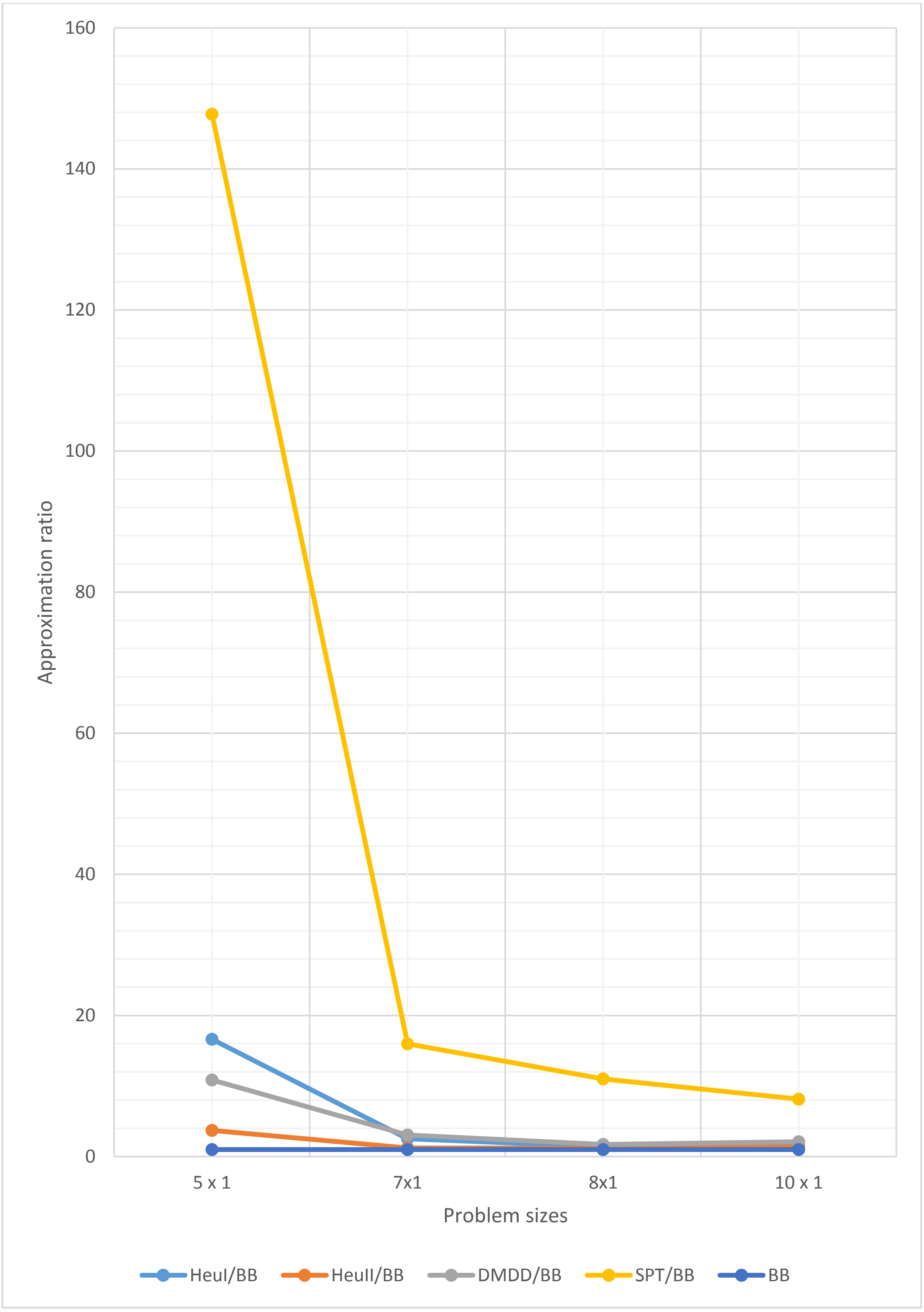


Figure 5.1: Plots of approximation ratio of all the solution methods for 5 ≤n ≤ 10

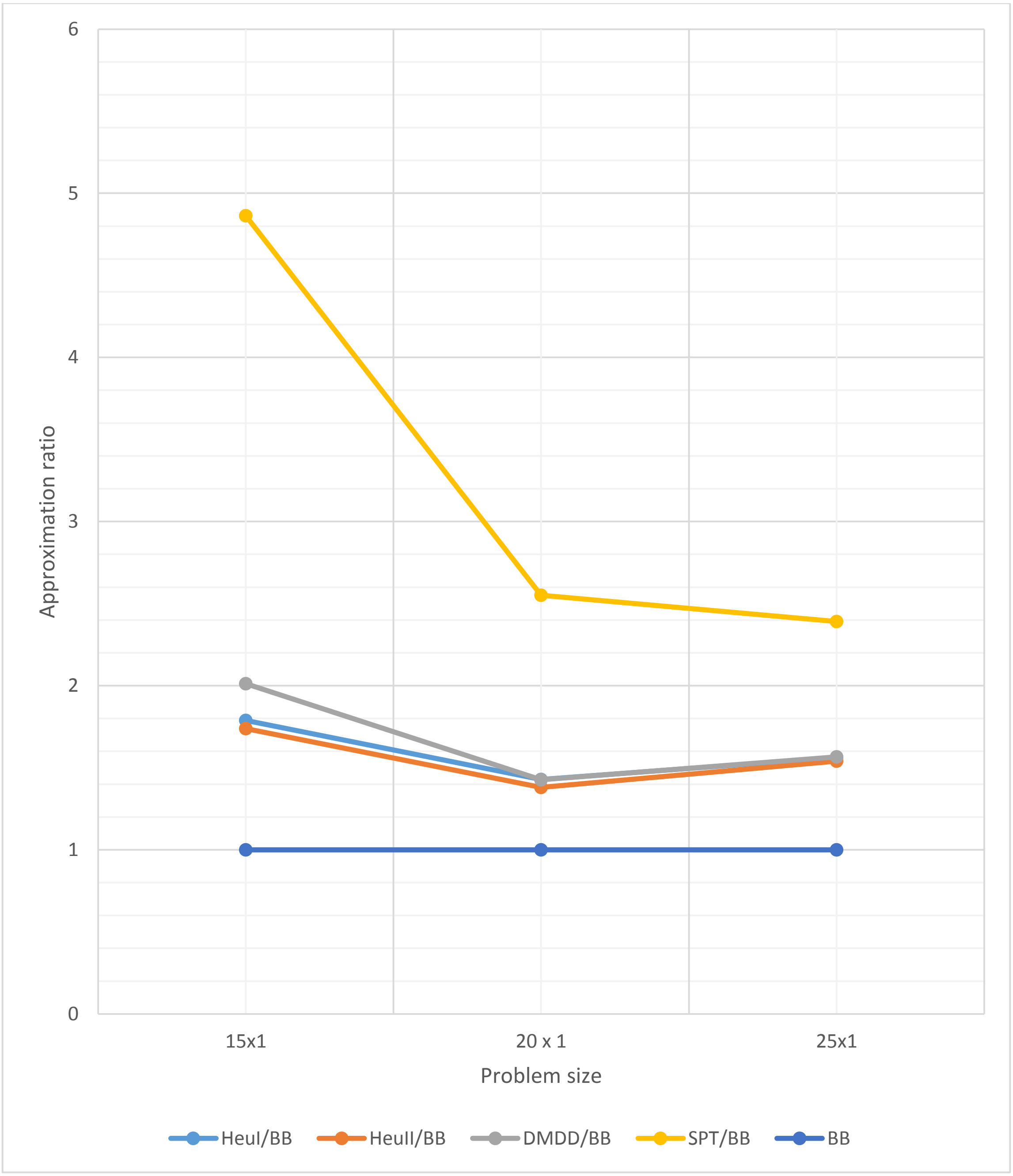


Figure 5.2: Plots of approximation ratio of all the solution methods for 10 ≤n ≤ 25

Furthermore, Table 5.3 and 5.4 show the overall means of approximation ratio of the solution methods for the small-sized and medium-sized problems, respectively.

Table 5.3: The overall means of approximation ratio for the small-sized problems

Solution methods	Overall means of approximation ratio
HeuI	5.53
HeuII	1.85
DMDD	4.42
SPT	45.71
BB	1.00

Table 5.4: The Overall means of approximation ratio for medium-sized problems.

Solution methods	Overall means of approximation ratio
HeuI	1.59
HeuII	1.55
DMDD	1.66
SPT	3.26
BB	1.00

The overall means of approximation ratio computed implies that the HeuII, DMDD, HeuI, and SPT are 1.85, 4.29, 5.53, and 45.71 times the optimal, respectively, on the average for the small–sized problems.

Figure 5.3 shows the plots of approximation ratio for all the solution methods for the large-sized problems; $30 \leq n \leq 1000$. The Figure shows that for job ranges $30 \leq n \leq 60$, the DMDD and HeuII are closer to the standard solution (HeuI). However, as the problem size increases to 80 jobs, the SPT method converges towards the standard solution, while other solution methods diverge. This implies that the SPT performed better than the DMDD and HeuII Heuristics. Furthermore, Table 5.5 shows the overall means of approximation ratio of all the solution methods for the large-sized problems.

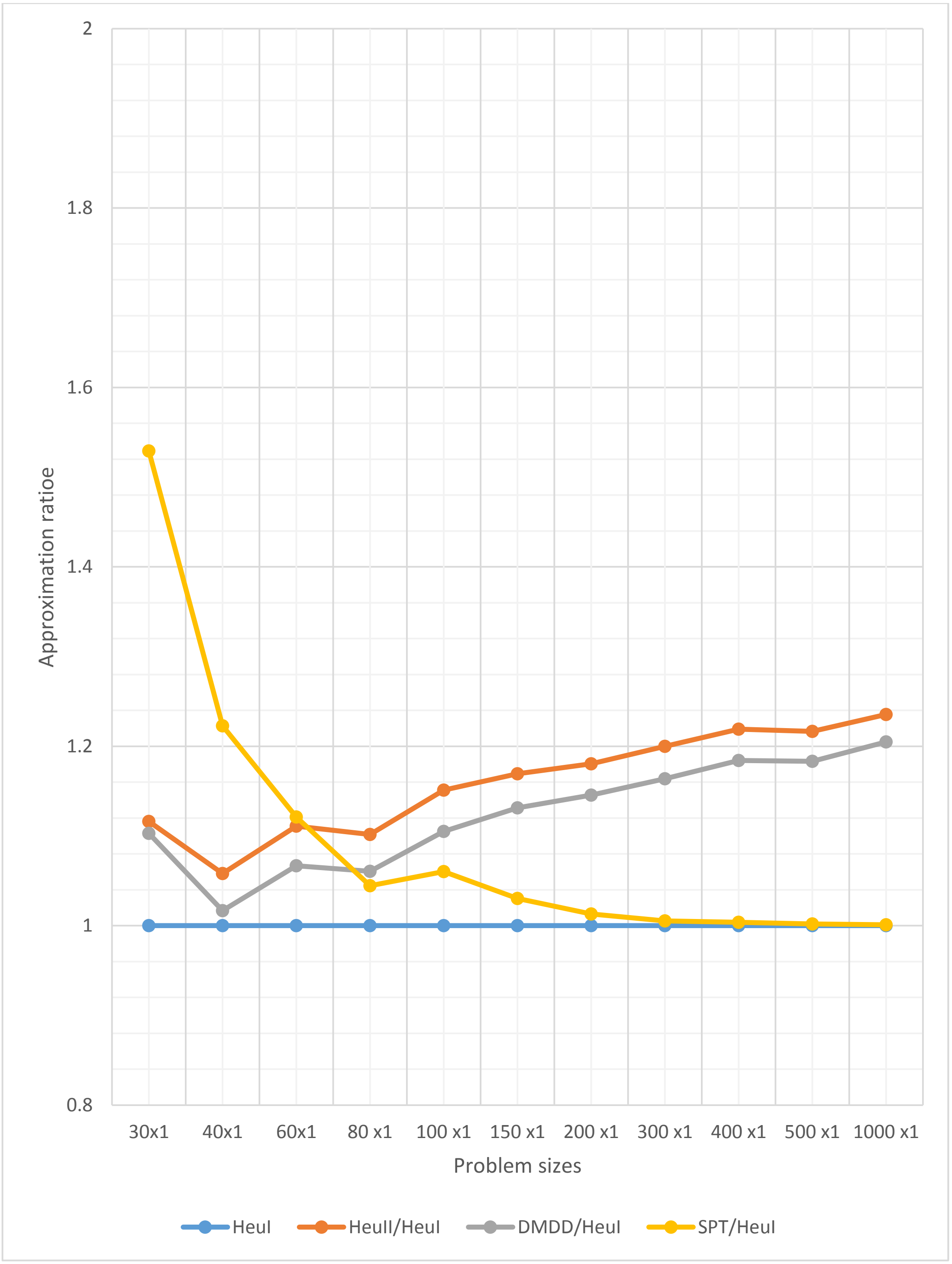


Figure 5.3: Plots of approximation ratio of all the solution methods for large-sized problems

Table 5.5: The overall means of approximation ratio for large sized problem

Solution methods	Overall means of approximation ratio
HeuI	1.00
HeuII	1.16
DMDD	1.12
SPT	1.09

5.2.4 The t-test

The results from the Table 5.2 and the approximation ratio test revealed that there are differences in the effectiveness of the solution methods. However, the analyses failed to ascertain whether the observed differences are significant or not. The statistical t-test of paired two samples for means was carried out, using Spreadsheet 2013 platform to determine the significance of the observed difference. Tables 5.6-5.8 show the results of the t-tests for the small-sized, medium-sized and the large-sized problems, respectively.

The t-test shows that for the small-sized problems, only the HeuII produced the results that are not significantly different ($\alpha > 0.05$) from the optimal, while other solution methods results are significantly different from the optimal ($\alpha < 0.05$). For the medium-sized problems, only the SPT solution method produced a result that is significantly different from the optimal. Table 5.8 shows that for the large-sized problems, the differences in the performance of all the implemented solution methods are not significantly different except for the SPT and HeuI. The HeuI performed significantly better than the SPT. This is because the poor performance of the SPT heuristic in the problem ranges $30 \leq n \leq 60$ overcome its divergence when the number of jobs exceeded 80.

Table 5.6: t-test for mean of total tardiness for the small-sized problems.

Solution methods	HeuI	HeuII	DMDD	SPT	BB
HeuI	--------	>0.05	>0.05	< 0.05*	< 0.05*
HeuII	>0.05	-------	>0.05	< 0.05*	> 0.05
DMDD	>0.05	>0.05	-------	< 0.05*	< 0.05*
SPT	<0.05*	< 0.05*	< 0.05*	----------	< 0.05*
BB	<0.05*	> 0.05	< 0.05*	<0.05*	--------

Note: *indicates significant result; Sample size = 50; -----indicates not necessary

Table 5.7: t-test for mean of total tardiness for the medium-sized problems.

Solution methods	HeuI	HeuII	DMDD	SPT	BB
HeuI	--------	>0.05	>0.05	< 0.05*	>0.05
HeuII	>0.05	-------	>0.05	< 0.05*	>0.05
DMDD	>0.05	>0.05	-------	< 0.05*	>0.05
SPT	<0.05*	<0.05*	<0.05*	--------	<0.05*
BB	> 0.05	> 0.05	> 0.05	< 0.05*	-------

Note: *indicates significant result; Sample size = 50; -----indicates not necessary

Table 5.8: t-test for mean of total tardiness for the large- sized problems

Solution methods	HeuI	HeuII	DMDD	SPT
HeuI	-------	>0.05	>0.05	<0.05*
HeuII	>0.05	-------	>0.05	>0.05
DMDD	>0.05	>0.05	--------	>0.05
SPT	<0.05*	>0.05	>0.05	------

Note: *indicates significant result; Sample size = 50; -----indicates not necessary

5.2.5 The Optimal Count (O.C) test

The test counts the number of times the heuristics yielded the same value of the total tardiness as the BB method. The test measures the chance of exploring optimal schedule in service, if a given heuristics is implemented. Table 5.9 shows the result of the count test. From the Table 5.9, it can be deduced that the number of times the heuristics yielded the optimal reduces as the problem sizes increases. Also, the HeuII produced the best results. The t-test result on the optimal count shows that, the HeuII is significantly better than other result. This is shown in Table 5.9b

Table 5.9a: The Optimal Count test

S/No	Problem Sizes	HeuI	HeuII	DMDD	SPT
1	5 x 1	54	78	58	0
2	7x1	44	60	42	0
3	8x1	30	56	24	0
4	10 x 1	22	48	10	0
	Mean Optimal count	**37.5**	**60.5**	**33.5**	**0**
5	15x1	6	14	2	0
6	20 x 1	0	0	0	0
7	25x1	0	0	0	0

Table 5.9b: t-test for the optimal count for the small- sized problems

Solution methods	HeuI	HeuII	DMDD	SPT
HeuI	-------	<0.05*	>0.05	<0.05*
HeuII	<0.05*	-------	<0.05*	<0.05*
DMDD	>0.05	<0.05*	--------	<0.05*
SPT	<0.05*	<0.05*	<0.05*	------

Note: *indicates significant result; -----indicates not necessary

5.2.6 The consistency test

The number of times the heuristic achieved optimality, the closeness of the heuristics to the optimal, as well as the significance of the observed differences to the optimal or the standard solution has been measured by the optimal count test, approximation ratio test and t-test, respectively. However, none of the tests measure the consistency of the solution methods in all the problem sizes in the fifty problem instances solved. To achieve this, the consistency test was carried out. The test involves counting the number of times each of the solution method yields the best result (the minimum mean value of total tardiness) in the fifty instances solved for each problem size. Table 5.10 shows the results of the consistency test.

Table 5.10: The Consistency test for all the problem sizes 5≤n ≤ 1000.

S/No	HeuI	HeuII	DMDD	SPT	Rank
5 x 1	2	100	4	0	HeuII, DMDD, HeuI, SPT
7x1	4	100	6	0	HeuII, DMDD, HeuI, SPT
8x1	12	96	0	0	HeuII, DMDD, HeuI, SPT
10 x 1	10	96	2	0	HeuII, DMDD, HeuI, SPT
15x1	26	82	0	0	HeuII, DMDD, HeuI, SPT
20 x 1	38	66	12	0	HeuII, DMDD, HeuI, SPT
25x1	50	54	4	0	HeuII ,DMDD, HeuI, SPT
30x1	66	40	52	0	HeuI, DMDD, HeuII, SPT
40x1	56	20	32	10	HeuI, HeuII, DMDD, SPT
60x1	70	24	14	24	HeuI, HeuII, SPT, DMDD
80 x1	80	2	10	56	HeuI, SPT , DMDD, HeuII
100 x1	82	0	0	72	HeuI, SPT , DMDD ,HeuII
150 x1	86	0	0	82	HeuI, SPT , DMDD ,HeuII
200 x1	86	0	0	80	HeuI, SPT , DMDD ,HeuII
250 x1	90	0	0	84	HeuI, SPT , DMDD ,HeuII
300 x1	90	0	0	84	HeuI, SPT , DMDD ,HeuII
400 x1	92	0	0	86	HeuI, SPT , DMDD ,HeuII
500 x1	94	0	0	90	HeuI, SPT , DMDD ,HeuII
1000 x1	98	0	0	90	HeuI, SPT , DMDD ,HeuII

Therefore, based on effectiveness, it can be inferred that

i. The HeuII is recommended for the small and medium-sized problems.
ii. The HeuI is recommended for the large- sized problems.

5.2.7 Real life results based on the total tardiness problem

Some real life single processor scheduling problems data were collected from VAL-VAL auto clinic to demonstrate the utility of the proposed models. All the implemented heuristics for simulated problems as well as the firm existing scheduling policy (FCFS) were applied on the data. The objective was to minimise the total tardiness. The results of the mean value of the total tardiness for the problem sizes are shown in Tables 5.11.

Table 5.11: The mean of the total tardiness by solution methods for real life problems

Problem size	Sample size	HeuI	HeuII	DMDD	SPT	FCFS
5x1	20	6.2	4.85	5.35	9.25	9.05
10x1	6	44.8	38.5	42.8	61	49
15x1	6	68.2	64.8	167.7	171.7	76.8

In order to recommend a scheduling policy for the firm, the percentage loss or gain of the existing FCFS policy with respect to each of the implemented solution methods was carried out. These are shown in Tables 5.12 – 5.14.

Table 5.12: Percentage gain or loss by adopting the HEUI policy

Problem Sizes	Sample size	HeuI	FCFS	Difference	% gain (total tardiness)	Remark
5x1	20	6.2	9.05	2.85	+46%	Gain
10x1	6	44.8	49	4.2	+9.40%	Gain
15x1	6	68.2	76.8	8.6	+12.7	Gain
Mean of percentage gain by adopting the HeuI policy					**22.7%**	**Gain**

Table 5.13: Average gain or loss by adopting the HEUII policy

Problem size	Sample size	HEUII	FCFS	Difference	%gain (total tardiness)	Remark
5x1	20	4.85	9.05	4.2	+86.6%	Gain
10x1	6	38.5	49	10.5	+27.3%	Gain
15x1	6	64.8	76.8	12	+19%	Gain

Mean of percentage gain by adopting the HeuII policy **44.3%**
Gain

Standard deviation of percentage gain by adopting the HeuII policy **36.9%**

Table 5.14: Average gain or loss by adopting DMDD

Problem size	Sample size	DMDD	FCFS	Difference	% gain or loss(total tardiness)	Remark
5x1	20	5.35	9.05	3.7	+69%	Gain
10x1	6	42.8	49	6.2	+15%	Gain
15x1	6	167.7	76.8	-90.9	-118%	Loss

Table 5.15: Average gain or loss by employing SPT

Problem sizes	Sample size	FCFS	SPT	Difference	% loss (total tardiness)	Remark
5x1	20	9.05	9.25	-0.25	-2.76%	Loss
10x1	6	49	61	-12	-24.5%	Loss
15x1	6	76.8	171.7	-94.6	-123%	Loss

The gains and losses from the Table 5.12 show that there is percentage gain in total tardiness by implementing the HeuI for the three problem sizes. Table 5.13 shows that the HeuII scheduling policy is better than the FCFS policy currently in practice. Also, the benefits of adopting the HeuII scheduling policy was found to be greater than that of the HeuI policy. Therefore, the HeuII scheduling policy is recommended over the HeuI policy. Table 5.14 shows that the DMDD scheduling policy is not the best method for the firm when the number of vehicles on the queue is fifteen jobs. Though, it was found that the policy is better than the FCFS employed by the firm when there are 5 and 10 vehicles on the queue. Table 5.15 shows that by adopting the SPT policy over the currently practiced FCFS policy would lead to loss of effectiveness. Thus, the SPT policy is not suited to the firm environment for jobs not received on the same date.

5.2.8 Results based on the execution time with respect to the total tardiness

The performance of the implemented solution methods were also measured based on the time required to solve an instance of the problem. Table 5.15 shows the mean execution time obtained for all the solution methods by problem sizes.

Table 5.16: Mean of the execution time by solution methods and problem sizes.

		Proposed Heuristics		Existing Heuristics		Optimal
S/No	Problem Sizes	HeuI	HeuII	DMDD	SPT	BB
1	5 x 1	0.00061	0.00063	0.00067	0.0002	288
2	7x1	0.00061	0.00064	0.00067	0.0002	962
3	8x1	0.00062	0.00065	0.00067	0.0002	1560
4	10 x 1	0.00064	0.00066	0.00069	0.0002	2979
5	15x1	0.00069	0.0007	0.00078	0.0002	4535
6	20 x 1	0.00071	0.00075	0.00086	0.00023	8727
7	25x1	0.00072	0.00074	0.00092	0.00026	17128
8	30x1	0.00075	0.00077	0.00099	0.00028	
10	40x1	0.00079	0.00079	0.00148	0.00029	
11	60x1	0.00083	0.00085	0.00196	0.00031	
12	80 x1	0.00087	0.00089	0.00277	0.00035	
13	100 x1	9.10E-05	9.3E-05	0.00362	3.70E-05	
14	150 x1	0.0095	0.0096	0.083	0.00398	
15	200 x1	0.00119	0.0012	0.0057	0.00041	
16	300 x1	0.0023	0.0011	0.0472	0.00048	
17	400 x1	0.00284	0.00065	0.0791	0.00057	
18	500 x1	0.00352	0.0075	0.0866	0.00067	
19	1000 x1	0.0049	0.0054	0.1321	0.00089	

Expectedly, the BB method has the highest execution time. The time complexity associated with this method limits its application for large-sized problems. The Table also shows that the SPT algorithm yielded the best results (the minimum execution time). Furthermore, the t-test and approximation ratio test were carried out to measure the efficiency of the solution methods.

5.2.9 The t-test

The t-tests was carried out to ascertain whether the differences observed between the execution times of the SPT and other heuristics were significant. The statistical t-test of paired two-samples for means was carried out, using spreadsheet 2013 data analysis. Table 5.16-5.18 show the results of t-test for the three problem sizes (small, medium and large sized problems)

Table 5.17: t-test for mean execution time for $5 \leq n \leq 10$ problems

Solution methods	HeuI	HeuII	DMDD	SPT	BB
HeuI	---------	>0.05	>0.05	<0.05*	<0.05*
HeuII	>0.05	-------	>0.05	<0.05*	<0.05*
DMDD	<0.05*	<0.05*	-------	<0.05*	<0.05*
SPT	<0.05*	<0.05*	<0.05*	----------	<0.05*
BB	<0.05*	<0.05*	<0.05*	<0.05*	---------

Table 5.18: t-test for mean execution time for 10 ≤n ≤ 25 problems

Solution methods	HeuI	HeuII	DMDD	SPT	BB
HeuI	---------	>0.05	<0.05*	<0.05*	<0.05*
HeuII	>0.05	-------	<0.05*	<0.05*	<0.05*
DMDD	<0.05*	<0.05*	-------	<0.05*	<0.05*
SPT	<0.05*	<0.05*	<0.05*	----------	<0.05*
BB	<0.05*	<0.05*	<0.05*	<0.05*	-----

Table 5.19: t-test for mean execution time for $30 \leq n \leq 1000$ problems

Solution methods	HeuI	HeuII	DMDD	SPT
HeuI	---------	>0.05	<0.05*	>0.05
HeuII	>0.05	-------	<0.05*	>0.05
DMDD	<0.05*	<0.05*	-------	<0.05*
SPT	>0.05	>0.05	<0.05*	----------

Note* indicates significant result; Sample size = 50 ; -----indicates not necessary

The results of the t-tests show that the execution time of SPT hare significantly better ($\alpha_{0.05}$ > 0.05) than other solution methods for the small-sized and the medium-sized problems. However, for large-sized problems, the execution time of SPT is not significantly different from that of the HeuI and HeuII. Also, the two proposed heuristics are significantly faster than the DMDD ($\alpha_{0.05}$< 0.05) for the medium-sized and large-sized problems.

5.2.10 The approximation ratio test for the execution time

The approximation ratio test was also used to compare the execution time of the heuristics. In this case, the SPT algorithm was used as the standard solution method (See APPENDIX 1, Table E for the approximation ratio Table). Figure 5.4-5.6 show the plots of approximation ratio for all the solution methods for the three problem classes.

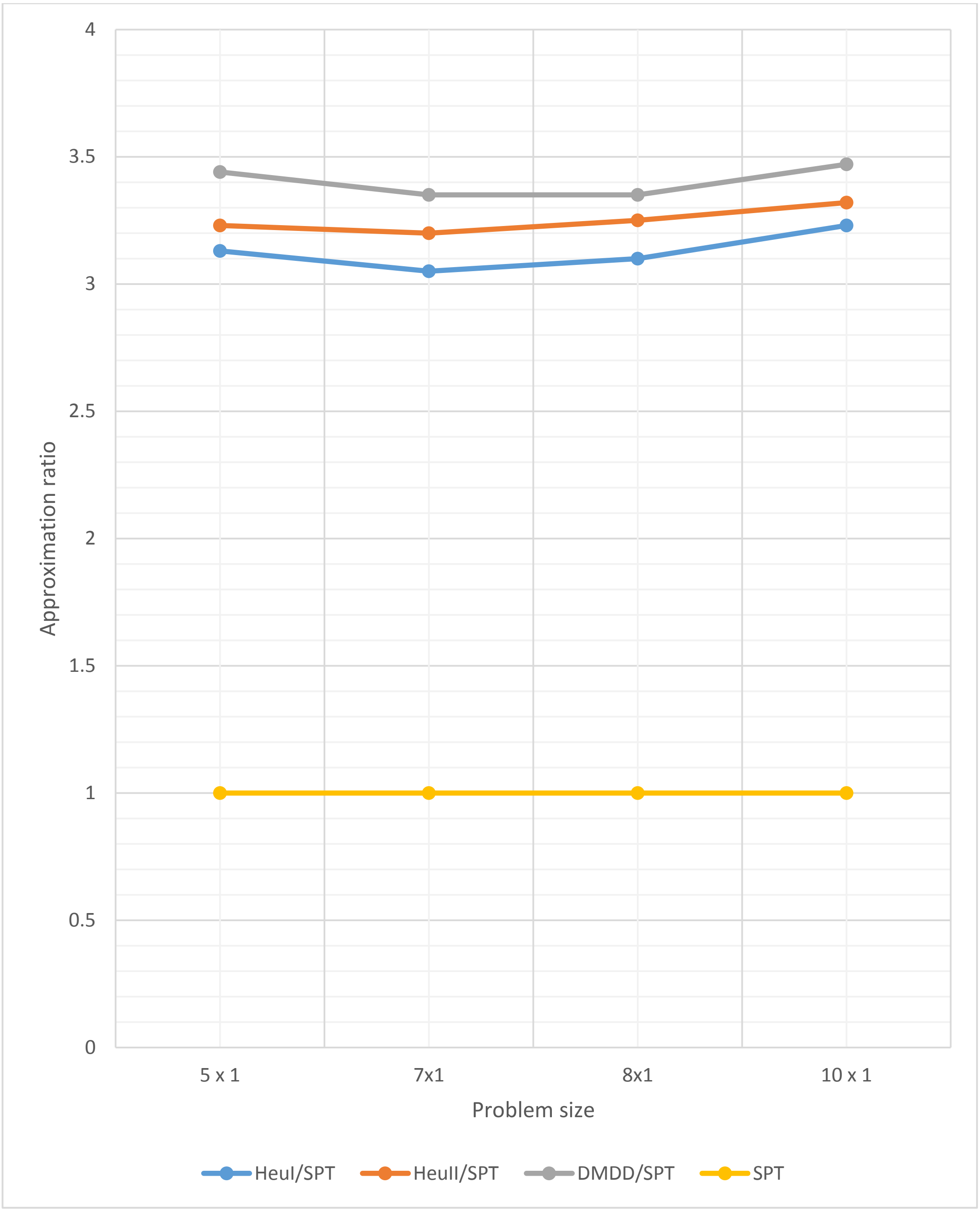


Figure 5.4: Plots of approximation ratio of solution methods for 5 ≤n ≤ 10

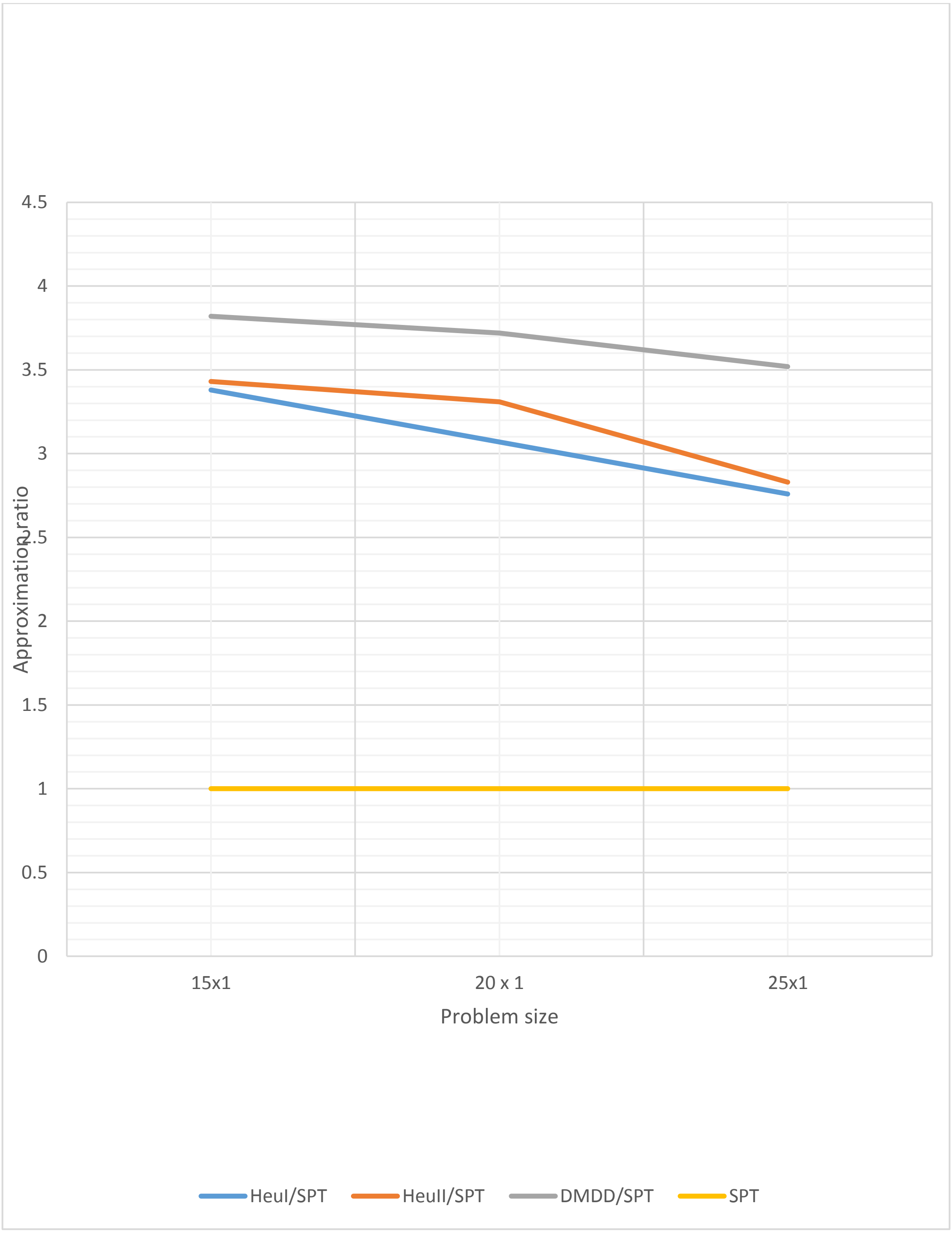


Figure 5.5: Plots of approximation ratio of solution methods for 15 ≤n ≤ 25

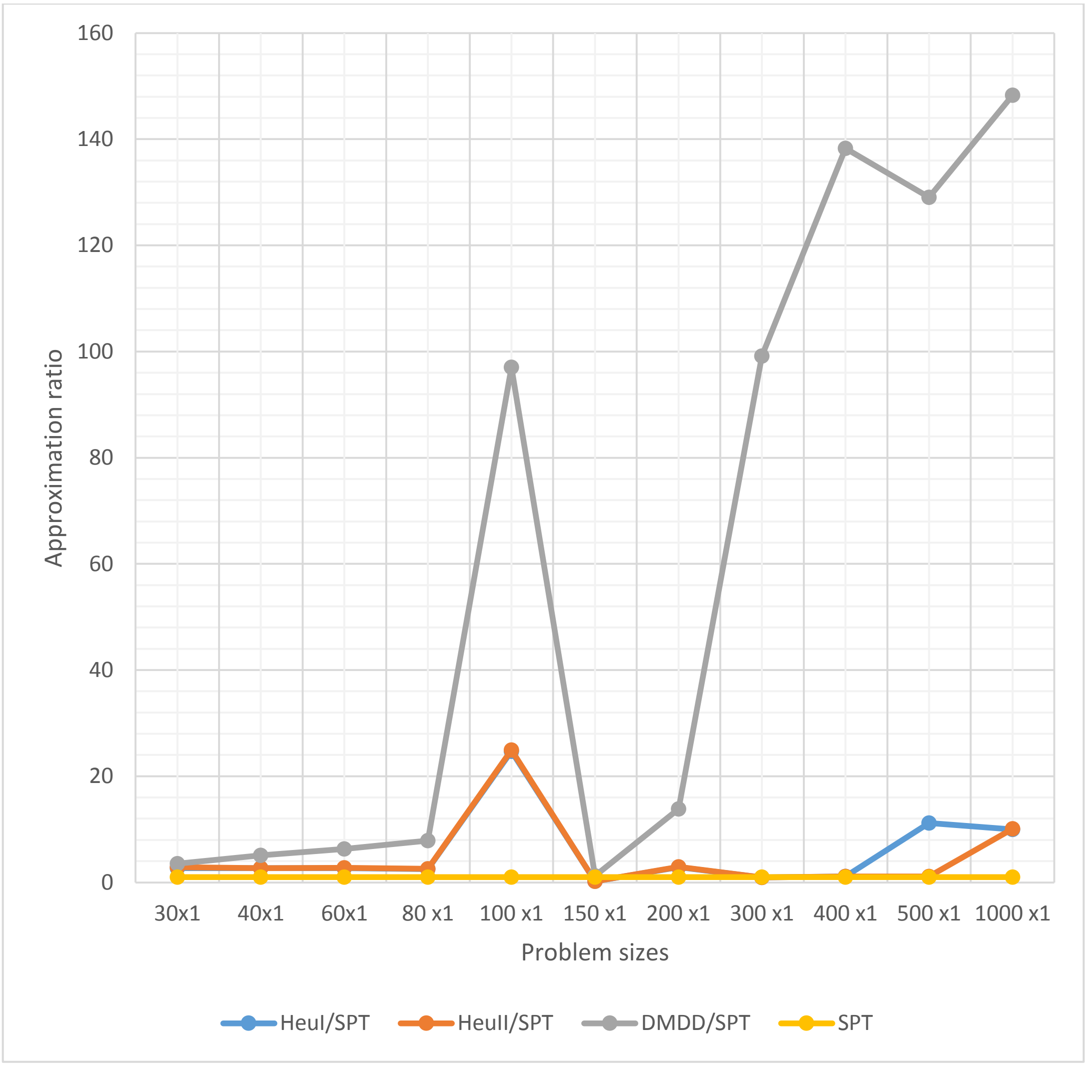


Figure 5.6: Plots of approximation ratio of solution methods for 30 ≤n ≤ 1000

The results show that as the problem size increases, the DMDD solution method diverges away from the standard solution. Thus, for the three problem sizes, the heuristics can be ranked in the order: SPT, HeuI, HeuII, DMDD. Furthermore, the overall means of approximation ratios (of the execution time) for all the solution methods for the three problem sizes are shown in Tables 5.20-5.22.

Table 5.20: The overall means of the approximation ratio for small sized problems

Solution methods	Overall means of approximation ratio
BB	7262500
HeuI	3.1
HeuII	3.23
DMDD	3.38
SPT	1.0

Table 5.21: The overall means of the approximation ratio for medium sized problems

Solution methods	Overall means of approximation ratio
BB	42333333
HeuI	3.1
HeuII	3.16
DMDD	3.73
SPT	1.0

Table 5.22: The overall means of the approximation ratio for large-sized problems

Solution methods	Overall means of approximation ratio
HeuI	3.53
HeuII	3.57
DMDD	60.93
SPT	1.00

The results based on the mean execution time show that the SPT heuristic is the most efficient solution method compare to other solution. However, with the exception of BB, no other solution requires prohibitive execution time.

Heuristics are being explored to obtain effective solution within an acceptable computation time. Though, SPT heuristic is the most efficient solution method, the heuristic is the least effective solution compared to other methods. Also, the two proposed heuristics; HeuII and HeuI, which are the most effective solution methods, have execution-time approximation ratio of HeuII(3.23, 3.1, 3.57), HeuI(3.1, 3.2, 3.53) with respect to the SPT for the small, medium and large sized problems respectively. However, the effectiveness approximation ratio of SPT are 45.71, 3.27 compared to HeuII (1.85, 1.55), HeuI(5.53,1.59) with respect to the optimal for the small and medium sized problems, respectively. Also, the HeuII yielded optimal in 60.5% for small-sized problems compare to 0% in SPT. In this regards, the performance of the two proposed heuristics with respect to the effectiveness and efficiency justified the selection of the HeuII and HeuI for the small and medium-sized and large sized problems, respectively.

Furthermore, the two proposed heuristics performed better than the DMDD. In terms of effectiveness and efficiency, the analytical tests reveal that the HeuII performed better than the DMDD for the small and medium-sized problems. For large-sized problems, the HeuI performed better than the HeuII and the DMDD. Thus, the HeuII is recommended over the DMDD for the small and medium sized problems, while the HeuI is recommended for the large-sized problems.

5.3 Results Based on the Bi-criteria Scheduling Problem

The bi-criteria scheduling problem of minimizing the composite function of total tardiness and total flowtime of jobs with zero release dates on single processor was explored as the Class II of this work. The dynamic variant of the problem was named the Class III. The results of the simulated problems and the real life problems for the bi-criteria approaches are presented in this section. For the simulated problems, the results include the mean of the normalized composite function of total tardiness and total flowtime of all the implemented solution methods for the two classes. The execution time are also presented. For the real life problems, the mean of the normalized composite objective for all the implemented solution methods were compared to that of the firm scheduling policy. The expected gain or loss for the firm, if a new scheduling policy were to be adopted, was computed and expressed as a percentage. This should aid the decision of the firm in adopting the most effective scheduling policy.

5.4 Results Based on the Static Variant of the Bi-criteria Problem (3TFZRD)

The mean values of the normalized total LCOF by solution methods and problem sizes are shown in Table 5.23.

Table 5.23: Mean value of the normalized total LCOF by solution methods

		Existing Heuristics		Implemented Heuristics	Proposed Heuristics		Optimal
S/N	Sizes	HeuA	HeuB	GAlg	HeuIII	HeuIV	BB
1	5x1	0.2900	0.5689	0.2975	0.2823	0.313	0.2823
2	7x1	0.3122	0.5598	0.3144	0.3015	0.3107	0.3015
3	8x1	0.3289	0.563	0.3168	0.3115	0.3225	0.3115
4	10x1	0.3451	0.5568	0.3293	0.3281	0.3304	0.3281
	Mean	**0.3191**	**0.5621**	**0.3145**	**0.3059**	**0.3192**	**0.3059**
5	15x1	0.357	0.5684	0.3353	0.3345	0.3355	0.3345
6	20x1	0.3669	0.571	0.344	0.3437	0.3441	0.3436
7	25x1	0.3648	0.5765	0.3413	0.3411	0.3413	0.3410
	Mean	**0.3629**	**0.5719**	**0.3402**	**0.33977**	**0.3403**	**0.3397**
8	30x1	0.3675	0.5777	0.3439	0.3438	0.3439	
9	40x1	0.3718	0.5776	0.348	0.3478	0.3479	
10	60x1	0.3708	0.5873	0.3476	0.3476	0.3477	
11	80 x1	0.3715	0.5891	0.3478	0.3478	0.3479	
12	100 x1	0.374	0.5848	0.3489	0.3489	0.3489	
13	150 x1	0.3746	0.5892	0.3509	0.3507	0.3508	
14	200 x1	0.3738	0.5891	0.3491	0.3491	0.3491	
15	300 x1	0.3751	0.59	0.3511	0.3511	0.3511	
16	400 x1	0.3773	0.588	0.3528	0.3501	0.3503	
17	500 x1	0.3761	0.5897	0.3519	0.3519	0.3519	
18	1000 x1	0.3766	0.5987	0.3523	0.3523	0.3523	
	Mean	**0.3736**	**0.5874**	**0.34948**	**0.34919**	**0.3492**	

The results obtained show that the solution methods can be ranked in the order: HeuIII, GAlg, HeuIV, HeuA, HeuB. The trends of the results for the small-sized, medium-sized and large -sized problems were subjected to analytical tests to measure the effectiveness of the heuristics.

5.4.1 The t-test

To ascertain whether the differences observed are significant, t-tests for paired two-samples for means were carried out, using the spreadsheet 2013 data analysis. Tables 5.24- 5.26 show the results of the t- tests for the small, medium and the large problem sizes, respectively. Furthermore, Table 5.27 and 5.28 show the overall means of approximation ratio of all the solution methods for the small-sized and the medium-sized problems, respectively.

Table 5.24: t-test for the mean value of the total LCOF for small-sized problems (5≤n≤10)

Solution method	GAlg	HeuIII	HeuIV	HeuA	HeuB	BB
GAlg	----	<0.05*	>0.05	<0.05*	<0.05*	<0.05*
HeuIII	<0.05*	------	>0.05	<0.05*	<0.05*	>0.05
HeuIV	>0.05	>0.05	-----	<0.05*	<0.05*	>0.05
HeuA	<0.05*	<0.05*	>0.05	--------	>0.05	<0.05*
HeuB	<0.05*	<0.05*	<0.05*	>0.05	--------	<0.05*
BB	<0.05*	>0.05	>0.05	<0.05*	<0.05*	---------

Table 5.25: t-test for mean for total LCOF for medium-sized problems (10 ≤n ≤ 25)

Solution method	GAlg	HeuIII	HeuIV	HeuA	HeuB	BB
GAlg	----	>0.05	>0.05	<0.05*	<0.05*	>0.05
HeuIII	<0.05*	------	>0.05	<0.05*	<0.05*	>0.05
HeuIV	>0.05	>0.05	-----	<0.05*	<0.05*	>0.05
HeuA	<0.05*	<0.05*	>0.05	--------	>0.05	<0.05*
HeuB	<0.05*	<0.05*	<0.05*	>0.05	--------	<0.05*
BB	<0.05*	>0.05	>0.05	<0.05*	<0.05*	---------

Table 5.26: t-test for total LCOF for large-sized problems (30 ≤n ≤ 1000)

Solution method	GAlg	HeuIII	HeuIV	HeuA	HeuB
GAlg	----	>0.05	>0.05	<0.05*	<0.05*
HeuIII	>0.05	------	>0.05	<0.05*	<0.05*
HeuIV	>0.05	>0.05	-----	<0.05*	<0.05*
HeuA	<0.05*	<0.05*	<0.05*	--------	>0.05
HeuB	<0.05*	<0.05*	<0.05*	>0.05	--------

Note* indicates significant result; Sample size = 50; -----indicates not necessary.

The t-tests results show that for the small and medium-sized problems, only the two proposed heuristics yielded results that are not significantly different from the optimal, while the heuristics found in the literature (the GAlg, HeuA and HeuB) produced results that are significantly different from the optimal. Also, the HeuIII, HeuIV, and GAlg are all significantly better than the HeuA and HeuB. In addition, the two existing heuristics (HeuA and HeuB) produced results that are significantly different ($p < 0.05$) from each other. The HeuA is significantly better than the HeuB. For the large-sized problems, the two proposed heuristics as well as the implemented GAlg are not significantly different from each other but are significantly better than the HeuA and HeuB.

5.4.2 The approximation ratio test

The BB solution results (the optimal) were used to compute the approximation ratio for other heuristics in the problem range $5 \leq n \leq 25$ (small and medium-sized problems). The HeuIII heuristic results were used to compute the approximation ratio for other solution methods for large-sized problem (See APPENDIX III, Tables F, G, and H for the approximation ratio Tables for the three problem sizes). Figure 5.7 and 5.8 show the plots of approximation ratios for the solution methods for the small-sized and medium-sized problems respectively. From the result, it can be observed that the HeuB plot is far away from the optimal (BB) plot compared to other solution methods.

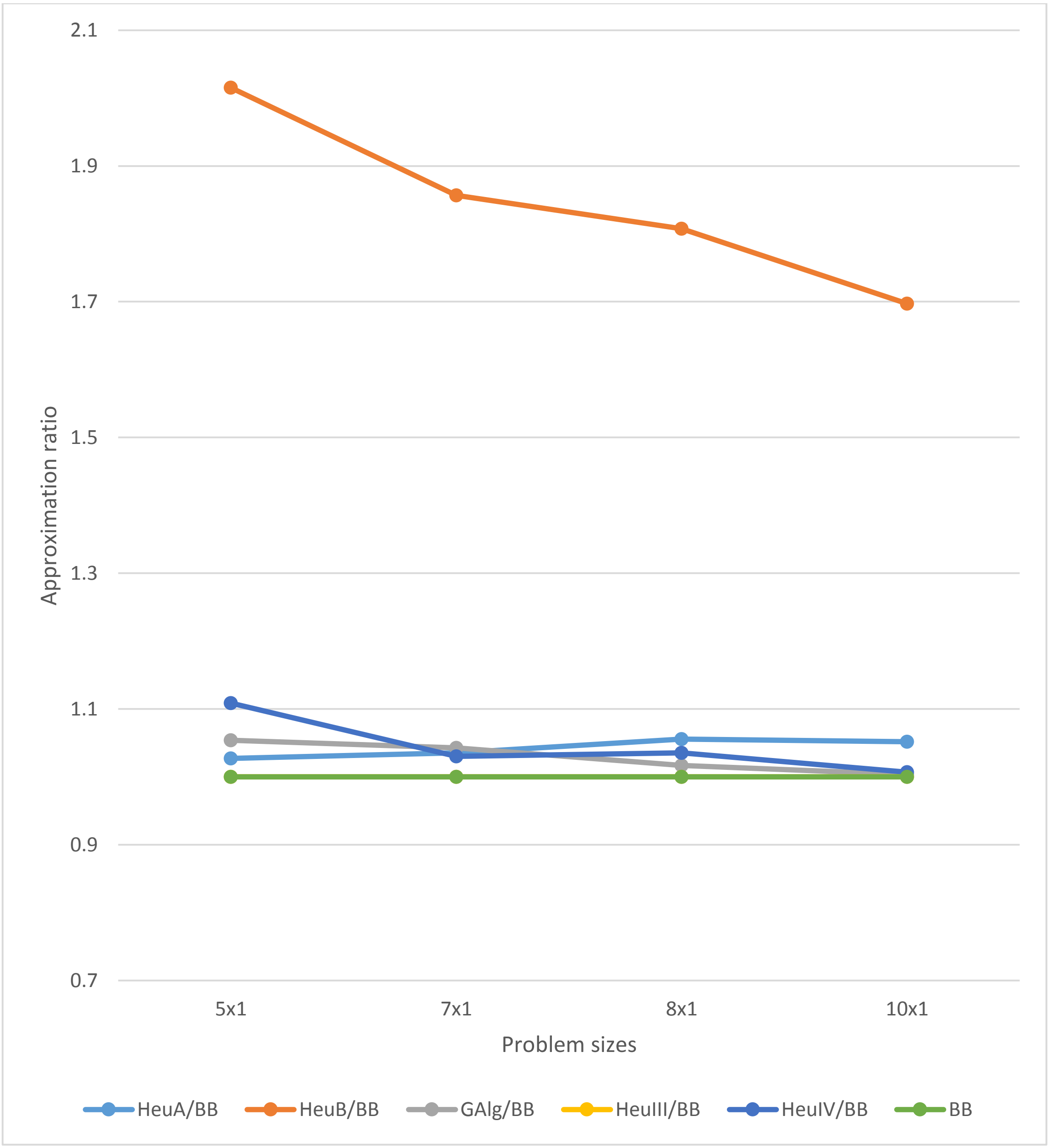


Figure 5.7: Plots of approximation ratio of solution methods for $5 \leq n \leq 10$

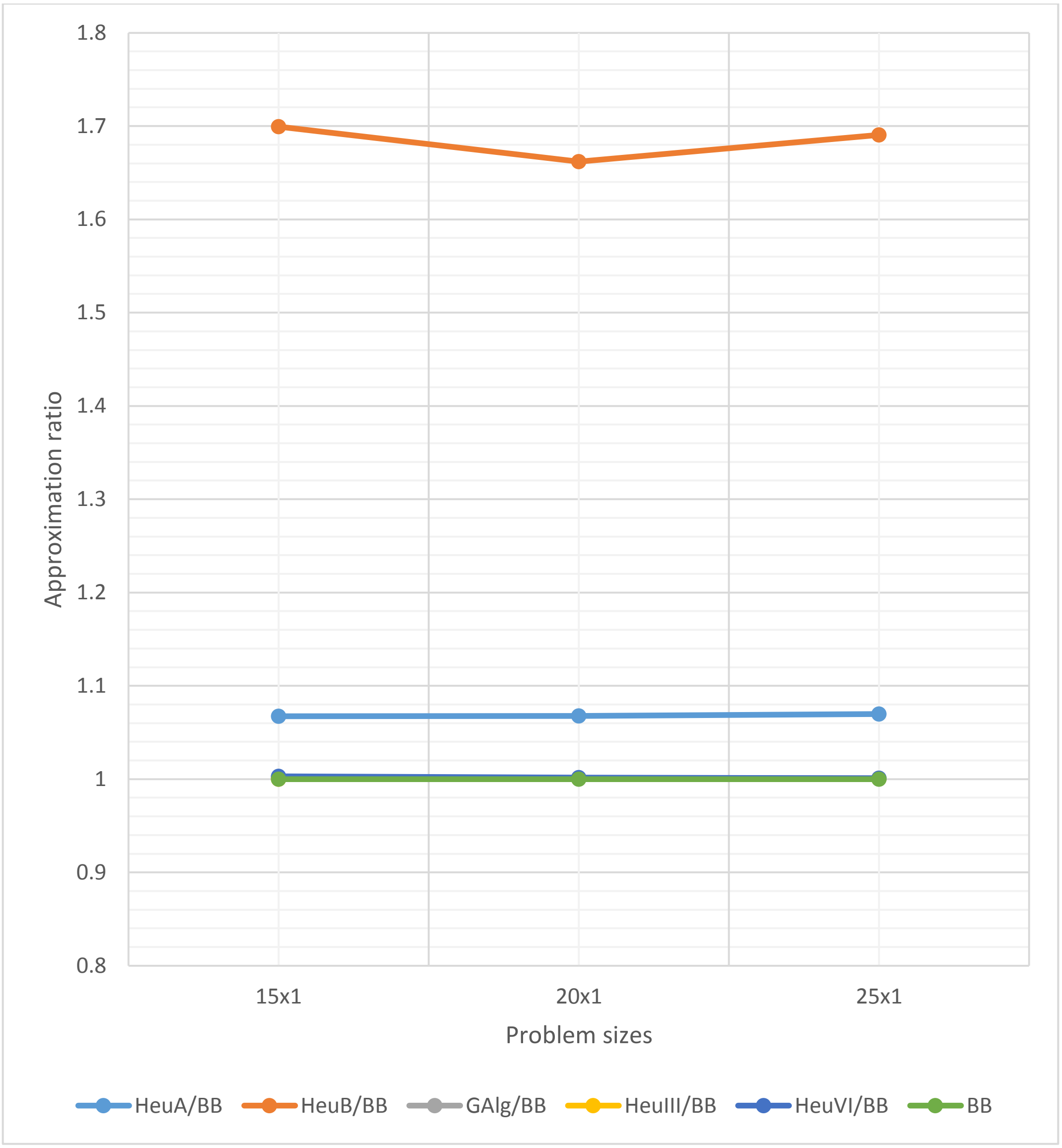


Figure 5.8: Plots of approximation ratio of solution methods for 15 ≤ n ≤ 25

Table 5.27: The overall means of approximation ratio for ranges; 5≤n ≤ 10.

Solution methods	Overall means of approximation ratio
HeuIII	1.00
HeuIV	1.04
GAlg	1.04
HeuA	1.05
HeuB	1.84
BB	1.00

Table 5.28: The overall means of approximation ratio for ranges; 15≤n ≤ 25.

Solution methods	Overall means of approximation ratio
HeuIII	1.0002
HeuIV	1.002
GAlg	1.01
HeuA	1.08
HeuB	1.68
BB	1.00

Similarly, Figure 5.9 shows the plots of approximation ratio for all the solution methods in the problem range; $30 \leq n \leq 1000$. The plots show that the HeuA and the HeuB show a wider difference compare to other heuristics.

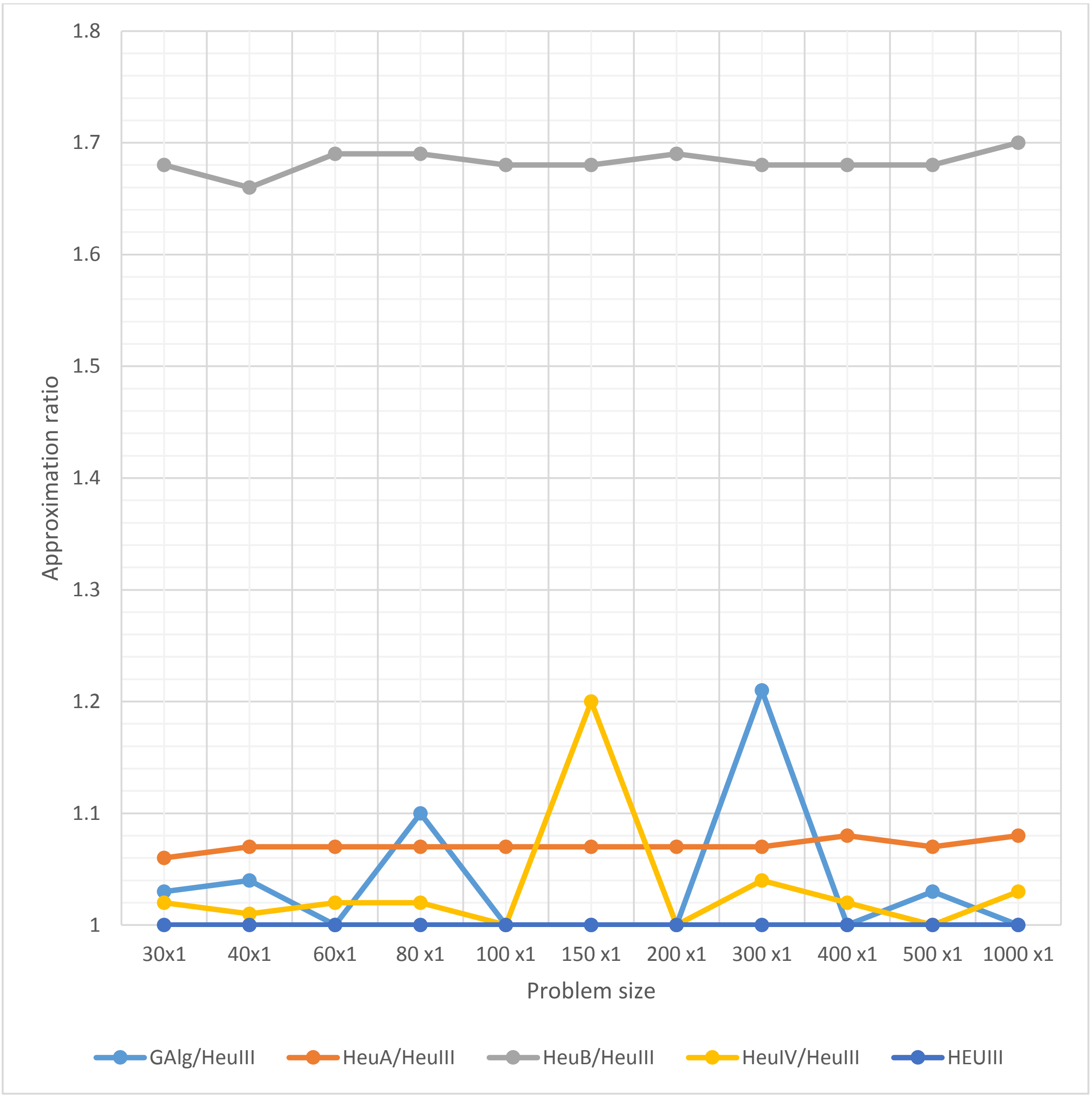


Figure 5.9: Plots of approximation ratio by problem sizes 30 ≤ n ≤ 1000

Furthermore, Table 5.29 shows the overall means of approximation ratio of all the solution methods for the large-sized problems.

Table 5.29: The overall means of approximation ratio for the ranges; $30 \leq n \leq 1000$

Solution methods	Overall means of approximation ratio
HeuIV	1.0003
GAlg	1.0009
HeuA	1.0700
HeuB	1.6800
HeuIII	1.0000

5.4.3 The Optimal Count (OC) test

The test count the number of time the heuristics yield the same value of the objective function as the BB method. Table 5.30 shows the result of count test.

Table 5.30: The Optimal Count test

S/No	Problem sizes	HeuIII	HeuIV	HEUA	HEUB	GAlg
1	5 x 1	96	78	26	0	84
2	7x1	96	76	16	0	84
3	8x1	94	76	12	0	84
4	10 x 1	90	72	10	0	80
	Mean	**94**	**75.5**	**16**	**0**	**83**
5	15x1	86	64	4	0	74
6	20 x 1	82	70	0	0	68
7	25x1	78	54	0	0	60

Based on the optimal count test results, the heuristics can be arranged in the order; HeuIII, GAlg, HeuIV, HeuA, HeuB. Furthermore, the result of t-test on the optimal count shows that the HeuIII result is significantly different from the other heuristics results. This is shown in Table 5.30b.

Table 5.30b : t-test for the Optimal count for the small-sized problem

Solution method	GAlg	HeuIII	HeuIV	HeuA	HeuB
GAlg	----	<0.05*	<0.05*	<0.05*	<0.05*
HeuIII	<0.05*	------	<0.05*	<0.05*	<0.05*
HeuIV	<0.05*	<0.05*	-----	<0.05*	<0.05*
HeuA	<0.05*	<0.05*	<0.05*	--------	>0.05
HeuB	<0.05*	<0.05*	<0.05*	<0.05*	--------

5.4.4 The consistency test

The consistency test was carried out to measure the tendency of a solution method to obtain best result in the fifty instances for all the problem sizes. The number of times each of the solution methods yielded the best results were expressed in percentage. This is shown in Table 5.31.

Table 5.31: The consistency test

Sizes	GAlg	HeuIII	HeuIV	HeuA	HeuB	Rank
5x 1	64	98	48	30	0	HeuIII, GAlg, HeuA, GAlg, HEUIV, HEUB.
7x1	66	98	42	24	0	HEUIII, GAlg, HeuA, GAlg, HEUIV, HEUB.
8x1	74	98	40	22	0	HEUIII, GAlg, HEUA, GAlg, HEUIV, HEUB.
10x1	82	94	38	0	0	HEUIII, GAlg, HEUIV, HEUA, HEUB,
15x1	56	78	38	0	0	HEUIII, GAlg, HEUIV, HEUA, HEUB
20x 1	82	86	40	0	2	HEUIII, GAlg, HEUIV, HEUA, HEUB
25x1	98	90	44	0	0	HEUIII, GAlg, HEUIV, HEUA, HEUB
30x1	72	94	34	0	0	HEUIII, GAlg, HEUIV, HEUA, HEUB
40x1	74	100	34	0	0	HEUIII, GAlg, HEUIV, HEUA, HEUB
60x1	94	100	42	0	0	HEUIII, GAlg, HEUIV, HEUA, HEUB
80 x1	94	96	36	0	0	HEUIII, GAlg, HEUIV, HEUA, HEUB
100 x1	96	100	34	0	0	HEUIII, GAlg, HEUIV, HEUA, HEUB
150 x1	92	98	34	0	0	HEUIII, GAlg, HEUIV, HEUA, HEUB

Table 5.31 Cont'd

Sizes	GAlg	HeuIII	HeuIV	HeuA	HeuB	Rank
200 x1	100	100	58	0	0	HEUIII, GAlg, HEUIV, HEUA, HEUB
250 x1	94	100	86	0	0	HEUIII, GAlg, HEUIV, HEUA, HEUB
300 x1	96	100	92	0	0	HEUIII, GAlg, HEUIV, HEUA, HEUB
400 x1	100	100	98	0	0	HEUIII, GAlg, HEUIV, HEUA, HEUB
500 x1	98	100	96	0	0	HEUIII, GAlg, HEUIV, HEUA, HEUB
1000 x1	100	100	98	0	0	HEUIII, GAlg, HEUIV, HEUA, HEUB

5.4.5 Real life results based on static variant of the bicriteria problem

The real life single processor scheduling problems data collected from VAL-VAL auto clinic were also implemented to demonstrate the utility of the heuristics proposed for the Class II problem. All the implemented heuristics as well as the firm existing scheduling policy (SPT) for jobs available on the same date were applied on the data. The objective was to minimise the composite function of the total tardiness and total flowtime. The results of the total normalized composite function obtained by all the solution methods are shown in Table 5.32.

Table 5.32: The mean of normalized total LCOF for the real life problem

Problem Size	Sample size	HeuA	HeuB	HeuIII	HeuIV	GAlg	SPT
5x1	20	0.2556	0.6049	0.1376	0.1498	0.1378	0.5701
10x1	5	0.2796	0.5392	0.2774	0.2779	0.2776	0.5930
15x1	5	0.4169	0.6124	0.2975	0.3467	0.3012	0.6572

In order to advise the firm on the best scheduling policy that suit its environments, the gain or loss from the use of any other scheduling policies compared to the SPT were computed. These are shown in Tables 5.33 – 5.37.

Table 5.33: Average gain or loss by adopting the HEUA policy

Problem sizes	Sample size	HEUA	SPT	Difference	Remark
5x1	20	0.2556	0.5701	+0.3145	Gain
10x1	5	0.2796	0.5930	+0.3134	Gain
15x1	5	0.4169	0.6572	+0.2403	Gain

Table 5.34: Average gain or loss by adopting the HeuB policy

Problem sizes	Sample size	HeuB	SPT	Difference	Remark
5x1	20	0.6049	0.5701	-0.0348	Loss
10x1	5	0.5392	0.5930	+0.0538	Gain
15x1	5	0.6124	0.6572	+0.0448	Gain

Table 5.35: Average gain or loss by adopting the GAlg policy

Problem sizes	Sample size	GAlg	SPT	Difference	Remark
5x1	20	0.1378	0.5701	+0.4323	Gain
10x1	6	0.2776	0.5930	+0.3154	Gain
15x1	6	0.3012	0.6572	+0.356	Gain

Table 5.36: Average gain or loss by adopting the HeuIII policy

Problem sizes	Sample size	HeuIII	SPT	Difference	Remark
5x1	20	0.1376	0.5701	+0.4325	Gain
10x1	6	0.2774	0.5930	+0.3156	Gain
15x1	6	0.2975	0.6572	+0.3597	Gain

Table 5.37: Average gain or loss by adopting the HeuIV policy

Problem sizes	Sample size	HeuIV	SPT	Difference	Remark
5x1	20	0.1498	0.5701	+0.4203	Gain
10x1	6	0.2779	0.5930	+0.3151	Gain
15x1	6	0.3467	0.6572	+0.3105	Gain

However, since the objectives were normalized, the normalized average gains or losses is converted to percentage in order to make the best decision with regard to the preferred scheduling policy. The percentage gain or loss by adopting any new policy are shown in Tables 5.38-5.42.

Table 5.38: Percentage gain or loss by adopting the HEUA policy

Problem sizes	Sample size	HEUA	SPT	Difference	%gain (total tardiness)	Remark
5x1	20	0.2556	0.5701	0.3145	+123%	Gain
10x1	5	0.2796	0.5930	0.3134	+112%	Gain
15x1	5	0.4169	0.6572	0.2403	+57.7%	Gain
Percentage gain by adopting the HEUA policy					**97.6%**	

Table 5.39: Percentage gain or loss by adopting the HEUB policy

Problem sizes	Sample size	HEUB	SPT	Difference	%gain (total tardiness)	Remark
5x1	20	0.6049	0.5701	-0.0348	-5.7%	Loss
10x1	5	0.5392	0.5930	0.0538	+9.98%	Gain
15x1	5	0.6124	0.6572	0.0448	+7.35%	Gain
Percentage gain by adopting the HEUB policy					**3.88%**	

Table 5.40: Percentage gain or loss by adopting the GAlg policy

Problem sizes	Sample size	GAlg	SPT	Difference	%gain (total tardiness)	Remark
5x1	20	0.1378	0.5701	0.4323	+313%	Gain
10x1	6	0.2776	0.5930	0.3154	+113%	Gain
15x1	6	0.3012	0.6572	0.356	+118%	Gain

Table 5.41: Percentage gain or loss by adopting the HeuIII policy

Problem sizes	Sample size	HeuIII	SPT	Difference	%gain (total tardiness)	Remark
5x1	20	0.1376	0.5701	0.4325	+314%	Gain
10x1	6	0.2774	0.5930	0.3156	+113%	Gain
15x1	6	0.2975	0.6572	0.3597	+121%	Gain
Mean of Percentage gain or loss by adopting the HEUIII policy						**182.7%**
Mean of Percentage gain or loss by adopting the HEUIII policy						**±113.8%**

Table 5.42: Percentage gain or loss by adopting the HeuIV policy

Problem sizes	Sample size	HeuIV	SPT	Difference	%gain (time)	Remark
5x1	20	0.1498	0.5701	0.4203	+281%	Gain
10x1	6	0.2779	0.5930	0.3151	+112%	Gain
15x1	6	0.3467	0.6572	0.3105	+89.6%	Gain
Percentage gain or loss by adopting the HeuIV policy						**160.9**

Results obtained from Table 5.38 implied that by adopting the HEUA method over the existing SPT policy, the firm would experience positive changes (gain) in performance, which is expressed as the percentage change in time.

Results obtained from Table 5.39 implied that by adopting the HeuB policy over the existing SPT method, the firm would experience a loss in performance if the accumulated jobs is five but would experiences positive changes for ten and fifteen accumulated jobs. This is expressed as the percentage change in total tardiness. Results obtained from Table 5.40 implied that by adopting the GAlg heuristic over the existing SPT policy, the firm would experience positive changes (gain) in performance which is expressed as the percentage change in time. The percentage gain from the three problem sizes considered is greater than that of HeuA and HeuB. Thus; GAlg heuristics is recommended over HeuA and HeuB.

Results obtained from Table 5.41 implied that by adopting the HeuIII over the existing SPT policy, the firm would experience positive changes (gain) in performance which is expressed as the percentage change in time. The percentage gain from the three problem sizes considered is greater than that of GAlg, HeuA and HeuB. Thus; the HeuIII is recommended over the GAlg, HeuA and HeuB. Results obtained from Table 5.42 implied that by adopting the HeuIV over the existing SPT policy, the firm would experience positive changes (gain) in performance which is expressed as the percentage change in time. The percentage gain from the three problem sizes considered is lesser than that of the HeuIII and GAlg but greater than that of HeuA and HeuB. Therefore, HeuIII scheduling policy is recommended for the firm for jobs received on the same date.

5.4.6 Results based on the efficiency for the static variant of the problem

The efficiency of an algorithm is measured through the execution time required to solve an instance of a given problem. In order to show the overall performance of the solution methods, the efficiency was also considered. In this regard, the mean of execution time to complete each problem size was computed. Table 5.43 shows the mean of the total execution time by solution methods and problem sizes.

Table 5.43: Mean of the total execution time by solution methods and problem sizes

		Existing Heuristics		Proposed Heuristics		Implemented	Optimal
S/N	Sizes	HeuA	HeuB	HeuIII	HeuIV	GAlg	BB
1	5 x 1	0.00065	0.00081	0.0016	0.0027	0.0036	2486.25
2	7x1	0.0007	0.00083	0.0018	0.003	0.0037	3012.12
3	8x1	0.00072	0.00083	0.0018	0.0032	0.0037	3215.45
4	10 x 1	0.00076	0.00085	0.0019	0.0035	0.0039	3472.12
5	15x1	0.00087	0.0009	0.0021	0.0045	0.0041	5243.23
6	20 x 1	0.00092	0.00094	0.0023	0.0058	0.0045	7453.54
7	25x1	0.00098	0.00097	0.0027	0.0064	0.0056	9145.62
8	30x1	0.001	0.0011	0.003	0.0076	0.0068	
9	40x1	0.0015	0.0016	0.0033	0.0081	0.009	
10	60x1	0.0015	0.0018	0.009	0.0088	0.0073	
11	80 x1	0.002	0.0021	0.019	0.0114	0.0088	
12	100 x1	0.0027	0.0029	0.023	0.0155	0.0108	
13	150 x1	0.003	0.0037	0.017	0.023	0.0155	
14	200 x1	0.0039	0.0041	0.016	0.0284	0.0197	
15	300 x1	0.0048	0.0055	0.0226	0.0382	0.0283	
16	400 x1	0.008	0.008	0.0301	0.0517	0.0374	
17	500 x1	0.01	0.01	0.0395	0.0644	0.047	
18	1000 x1	0.0219	0.022	0.0955	0.1274	0.1136	

The results based on the execution time show that the HeuA has the lowest execution time while the BB has the highest execution time. However, to ascertain whether the differences observed between the execution time of all the heuristics are significant, statistical t-test of paired two-samples for means was carried out, using spreadsheet 2013 data analysis. Tables 5.44-5.46 show the results of the t-tests.

Table 5.44: t-test for the execution time for small-sized problems

Solution method	GAlg	HeuIII	HeuIV	HeuA	HeuB	BB
GAlg	----	>0.05	>0.05	<0.05*	<0.05*	<0.05*
HeuIII	>0.05	------	>0.05	<0.05*	<0.05*	<0.05*
HeuIV	>0.05	>0.05	-----	<0.05*	<0.05*	<0.05*
HeuA	<0.05*	<0.05*	<0.05*	-------	<0.05*	<0.05*
HeuB	<0.05*	<0.05*	<0.05*	<0.05*	<0.05*	<0.05*
BB	<0.05*	<0.05*	<0.05*	<0.05*	<0.05*	--------

Table 5.45: t-test for the execution time for medium-sized problems

Solution method	GAlg	HeuIII	HeuIV	HeuA	HeuB	BB
GAlg	----	>0.05	>0.05	<0.05*	<0.05*	<0.05*
HeuIII	>0.05	------	>0.05	<0.05*	<0.05*	<0.05*
HeuIV	>0.05	>0.05	-----	<0.05*	<0.05*	<0.05*
HeuA	<0.05*	<0.05*	<0.05*	------	<0.05*	<0.05*
HeuB	<0.05*	<0.05*	<0.05*	<0.05*	------	<0.05*
BB	<0.05*	<0.05*	<0.05*	<0.05*	<0.05*	-------

Table 5.46: t-test for the execution time for large-sized problems

Solution method	GAlg	HeuIII	HeuIV	HeuA	HeuB
GAlg	----	>0.05	>0.05	>0.05	>0.05
HeuIII	>0.05	------	>0.05	>0.05	>0.05
HeuIV	>0.05	>0.05	-----	>0.05	>0.05
HeuA	>0.05	>0.05	>0.05	>0.05	>0.05
HeuB	>0.05	>0.05	>0.05	>0.05	>0.05

Note* indicates significant result , Sample size = 50 ; -----indicates not necessary.

The results of the t-tests show that the execution time of the HeuA methods are significantly better than other solution method.

Moreover, approximation ratio test was also carried out. HEUA was used as the standard because it has the lowest execution time. Figures 5.10-5.12 show the plots of approximation ratios for all the implemented solution methods. (See APPENDIX 1, Table I for the approximation ratio Table).

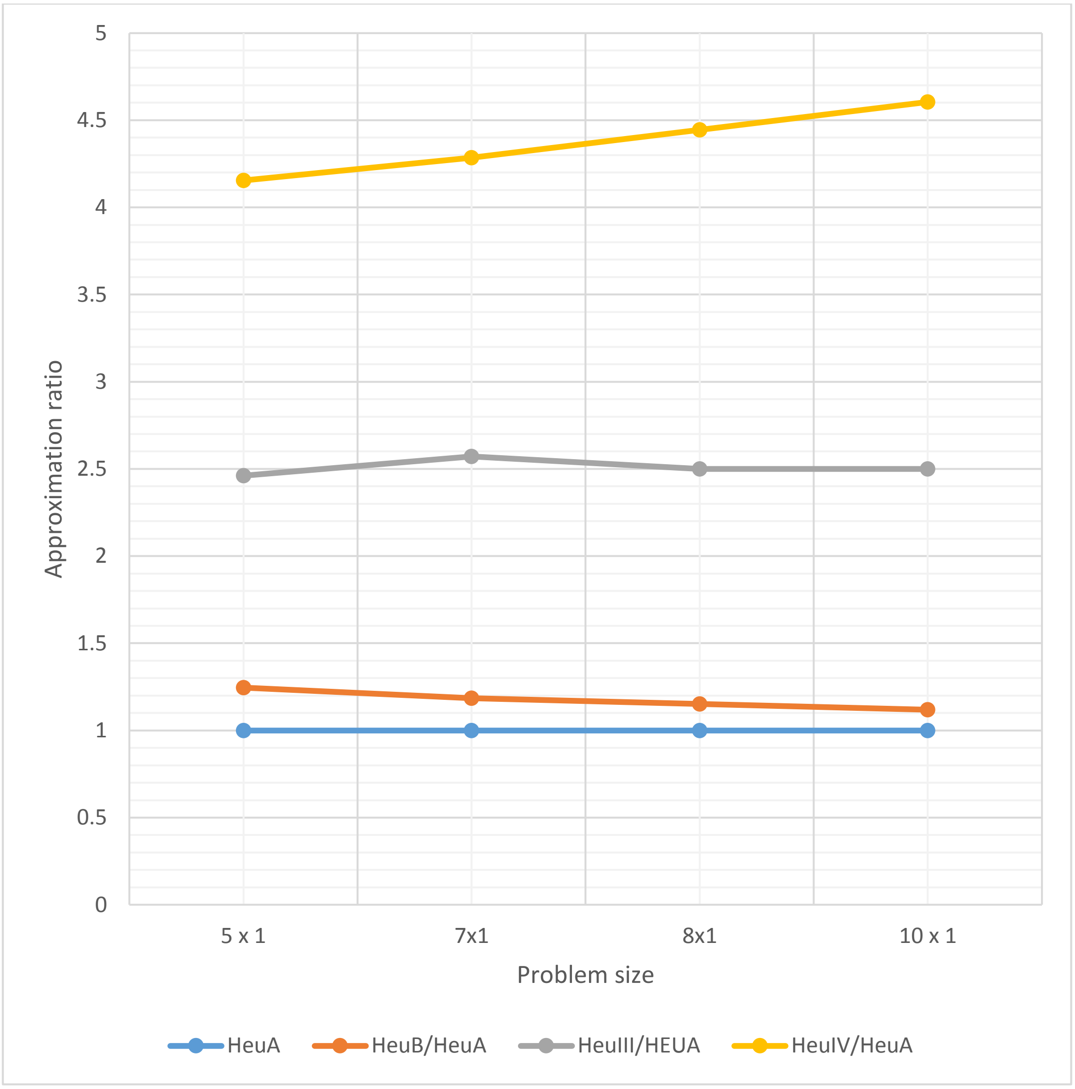


Figure 5.10: The plots of approximation ratio for the small-sized problems

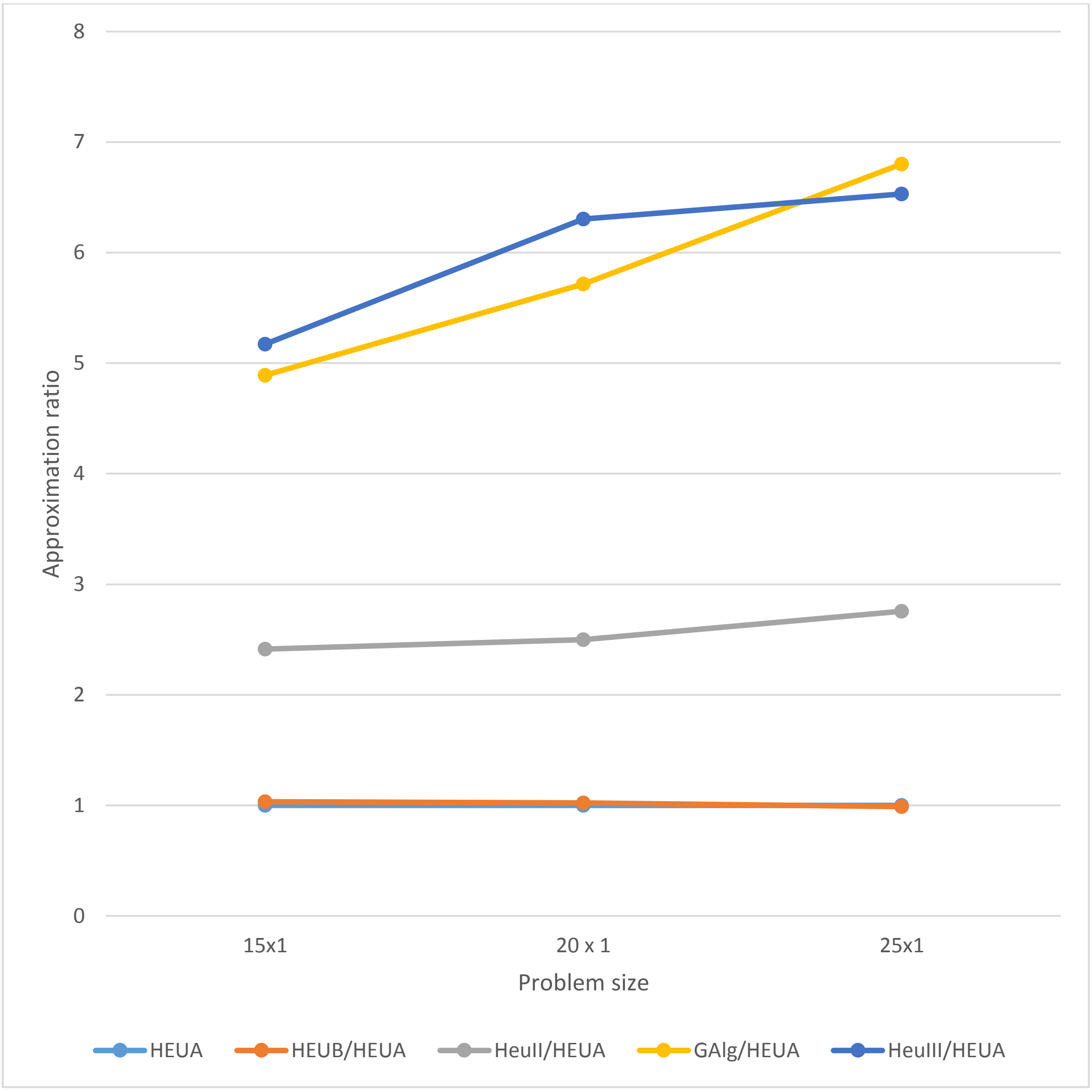


Figure 5.11: The plots of approximation ratio for the medium-sized problems

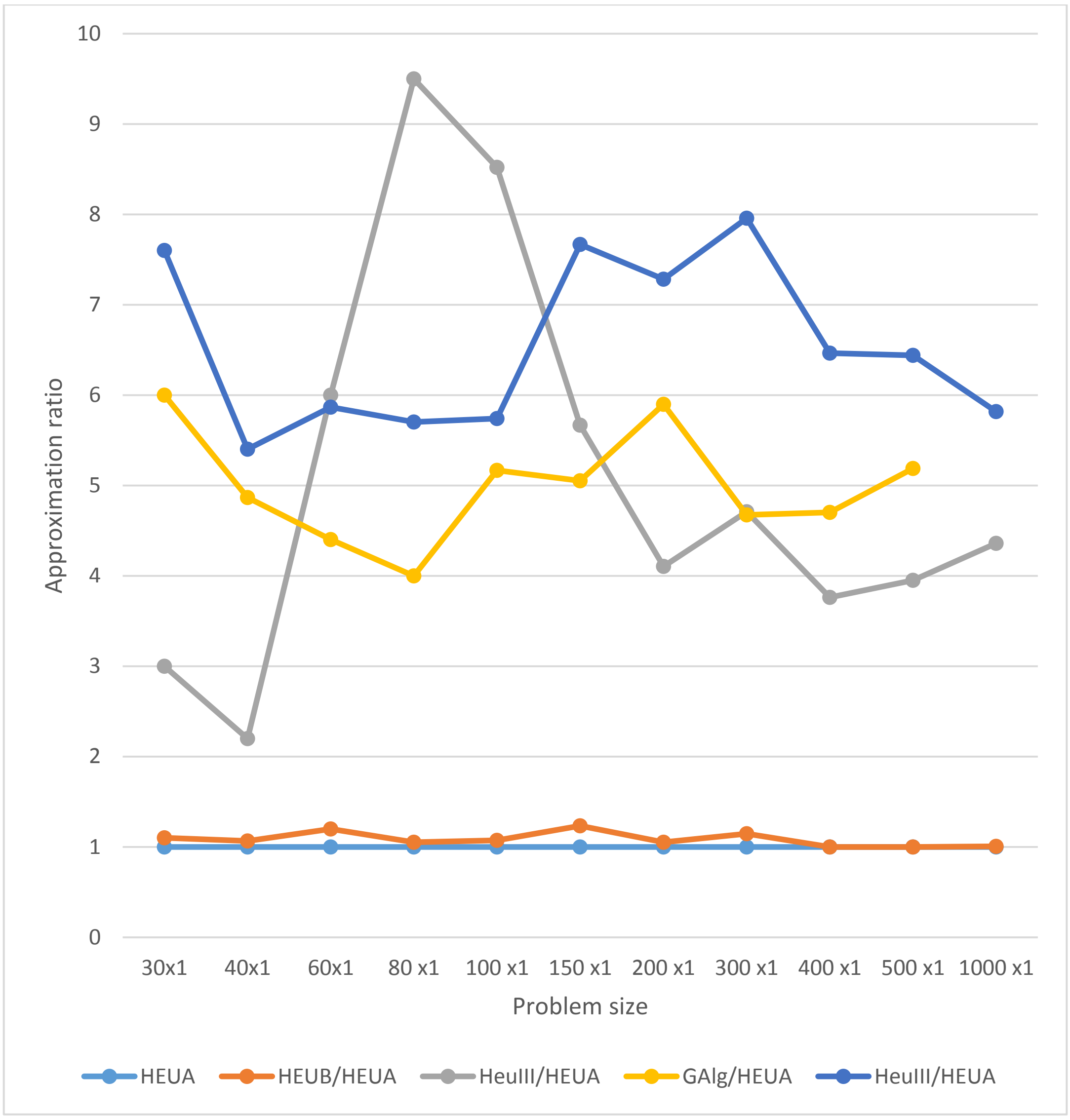


Figure 5.12: The plots of approximation ratio for the large-sized problems

Tables 5.47-5.52 show the overall means of approximation ratio of all the solution methods for the three problem sizes.

Table 5.47: The overall means of approximation ratio for the problems 5≤n ≤ 10

Solution methods	Overall means of approximation ratio
BB	4290627.574
HeuIV	4.37
HeuIII	2.51
GAlg	5.06
HeuB	1.18
HeuA	1.00

Table 5.48: The overall means of approximation ratio for the problem ($15 \leq n \leq 25$)

Solution methods	Overall means of approximation ratio
HeuIV	5.92
HeuIII	4. 18
GAlg	5.06
HeuB	1.08
HeuA	1.00

Table 5.49: The overall means of approximation ratio for the problem ranges, 30≤n ≤ 1000

Solution methods	Overall means of approximation ratio
HeuIV	6.31
HeuIII	3. 83
GAlg	5.00
HeuB	1.02
HeuA	1.00

The results based on the mean execution time show that the HeuA is the most efficient solution method compare to other solution. However, with the exception of BB, no other solution requires prohibitive execution time. Furthermore, the small values of approximation ratios for other heuristics for the three problem classes implies that the effectiveness with respect to the optimal should be given a higher priority in determine the recommended heuristic for the problem. However, the benefits of exploring the HeuIII heuristic is greater than that of HEUA. This is because in terms of effectiveness, the HeuIII heuristic is not significantly different from the optimal, yielded an approximation ratio of 1 and capable of yielding optimal results in 94% (in 50 problem instances) for small-sized problems while HEUA is significantly different from optimal, with A.R of 1.08 and optimal count of 16%. In this regards, for small, medium and large size problems, the proposed HeuIII is recommended.

5.5 Results Based on the Dynamic Variant of the Bi-criteria Problem

The scheduling problem of minimizing the composite function of total tardiness and total flowtime of jobs with non-zero release dates was explored as the Class III. This section presents the results obtained for the problem class.

5.5.1 Simulation results based on effectiveness

The mean values of the normalized total LCOF for the implemented solution methods for the problem class are shown in Table 5.50.

Table 5.50: Mean of normalized total LCOF by solution methods

	Implemented Heuristics		Proposed Heuristics		Optimal
Problem sizes	GAlg	DMDD	HeuV	HeuVI	BB
5 x 1	0.3437	0.2827	0.2360	0.2225	0.1601
7x1	0.3728	0.2453	0.2249	0.2002	0.1675
8x1	0.4011	0.2591	0.2466	0.209	0.1681
10 x 1	0.4325	0.2733	0.2326	0.2067	0.1698
Mean	**0.387525**	**0.2651**	**0.23503**	**0.2096**	**0.1664**
15x1	0.5734	0.3366	0.2831	0.2708	0.1786
20 x 1	0.6234	0.3788	0.3511	0.335	0.2624
25x1	0.6605	0.406	0.3843	0.3768	0.2821
Mean	**0.6191**	**0.3738**	**0.3395**	**0.32753**	**0.24103**
30x1	0.6762	0.4307	0.4164	0.4073	
40x1	0.7121	0.4596	0.453	0.4491	
60x1	0.512	0.7406	0.5094	0.5018	
80 x1	0.7583	0.5365	0.536	0.5292	
100 x1	0.7828	0.5532	0.5494	0.5465	
150 x1	0.8011	0.5785	0.5775	0.5725	
200 x1	0.8024	0.5886	0.588	0.5855	
300 x1	0.8151	0.6019	0.5998	0.5978	
400 x1	0.825	0.6054	0.6061	0.6052	
500 x1	0.8255	0.6088	0.6094	0.6086	
1000 x1	0.837	0.6167	0.6168	0.6167	
Mean	**0.7589**	**0.57459**	**0.55107**	**0.54729**	

Based on the results obtained, it can be deduced that the two proposed heuristics and the solution methods from the literature can be ranked in the order; HeuVI, HeuV, DMDD, GAlg. However, to ascertain whether the differences observed were significant, statistical t-test for paired two-samples for means were carried out, using the spreadsheet 2013 data analysis. Tables 5.51 -5.53 show the results of test of mean for three problem ranges.

Table 5.51: t-test for the mean value of the total LCOF for small-sized problems

Solution methods	GAlg	HeuVI	HeuV	DMDD	BB
GAlg	----	<0.05*	<0.05*	<0.05*	<0.05*
HeuV	<0.05*	<0.05*	-------	<0.05*	<0.05*
HeuVI	<0.05*	------	<0.05*	<0.05*	<0.05*
DMDD	<0.05*	<0.05*	<0.05*	--------	<0.05*
BB	<0.05*	<0.05*	<0.05*	<0.05*	---

Table 5.52: t-test for the mean value of the total LCOF for medium-sized problems

Solution methods	GAlg	HeuV	HeuVI	DMDD	BB
GAlg	----	<0.05*	<0.05*	<0.05*	<0.05*
HeuV	<0.05*	-------	<0.05*	<0.05*	<0.05*
HeuVI	<0.05*	<0.05*	------	<0.05*	<0.05*
DMDD	<0.05*	<0.05*	<0.05*	--------	<0.05*
BB	<0.05*	<0.05*	<0.05*	<0.05*	---

Table 5.53: t-test for the mean value of the total LCOF for large-sized problems

Solution methods	GAlg	HeuV	DMDD	HeuVI
GAlg	----	<0.05*	<0.05*	<0.05*
HeuV	<0.05*	-------	>0.05	<0.05*
HeuVI	<0.05*	<0.05*	>0.05	------
DMDD	<0.05*	>0.05	---------	>0.05

Note:* indicates significant result; Sample size = 50; -----indicates not necessary.

Tables 5.51 and 5.52 show that for the small-sized and medium-sized problems, all the heuristics yielded results that are significantly different from the optimal. For the three problem sizes, the proposed heuristic; HeuVI yielded results that are significantly better than other solution methods.

The approximation ratio test was also carried out. The BB results (the optimal solution) were used for benchmarking for the problem sizes $5 \leq n \leq 25$ (small and medium sized problems), while the HeuVI results were used for the problem sizes; $30 \leq n \leq 1000$. Figures 5.13-5.15 show the approximation ratio for the three problem sizes, respectively. (See Appendix II1, Tables J, K and L for the approximation Table for the three problem sizes).

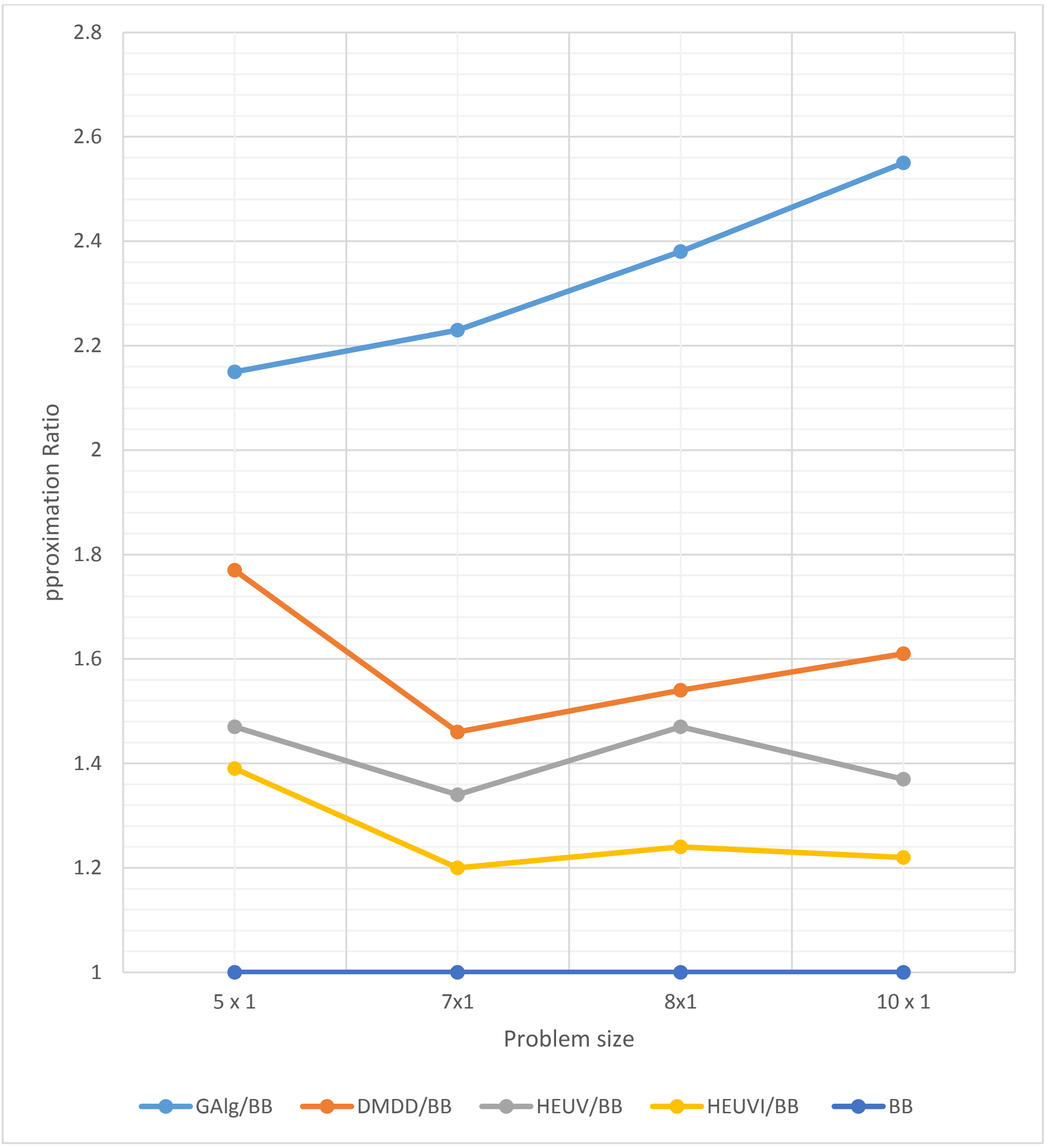


Figure 5.13: Plots of approximation ratio for problem sizes 5 ≤n ≤ 10

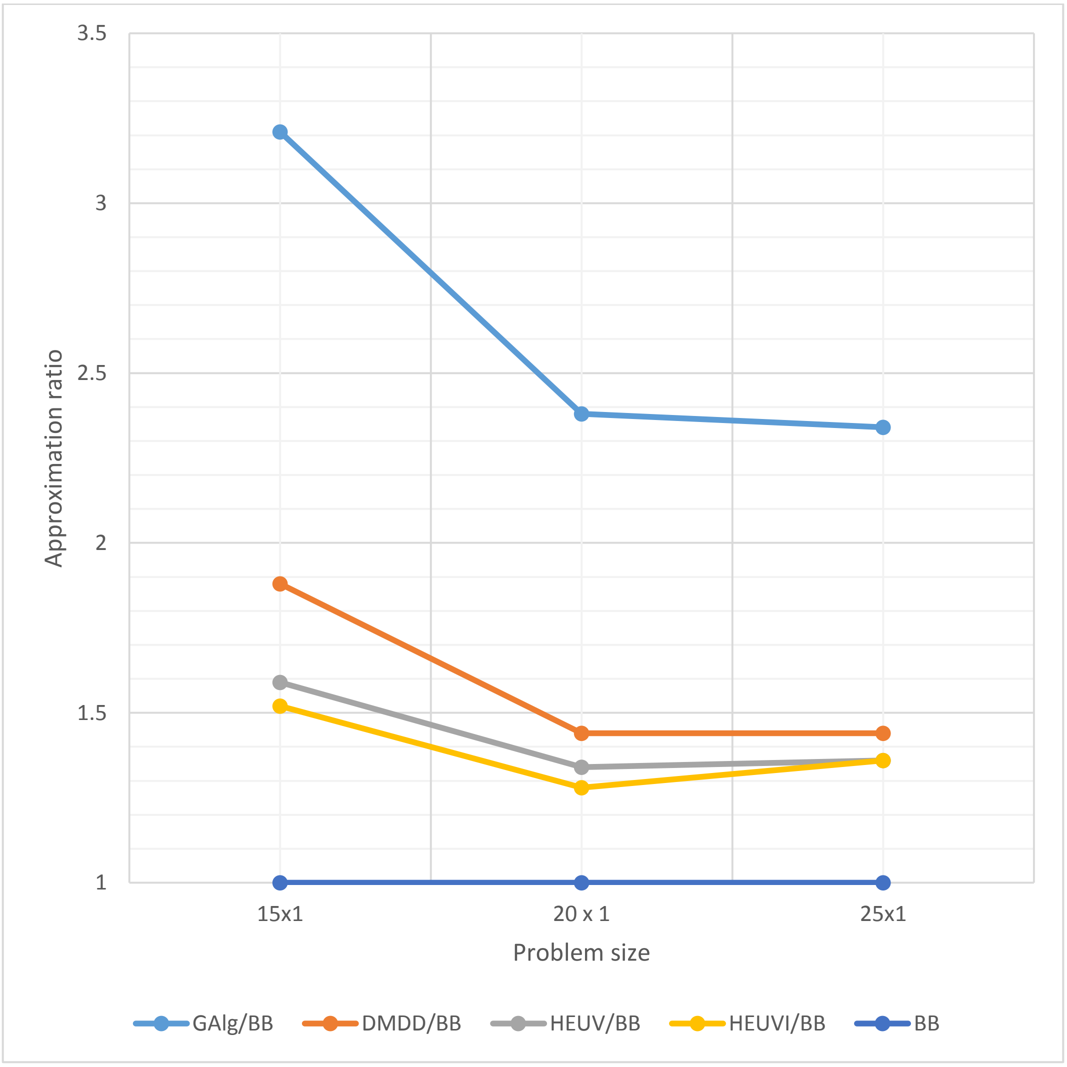


Figure 5.14: Plots of approximation ratio for problem sizes 15 ≤n ≤ 25

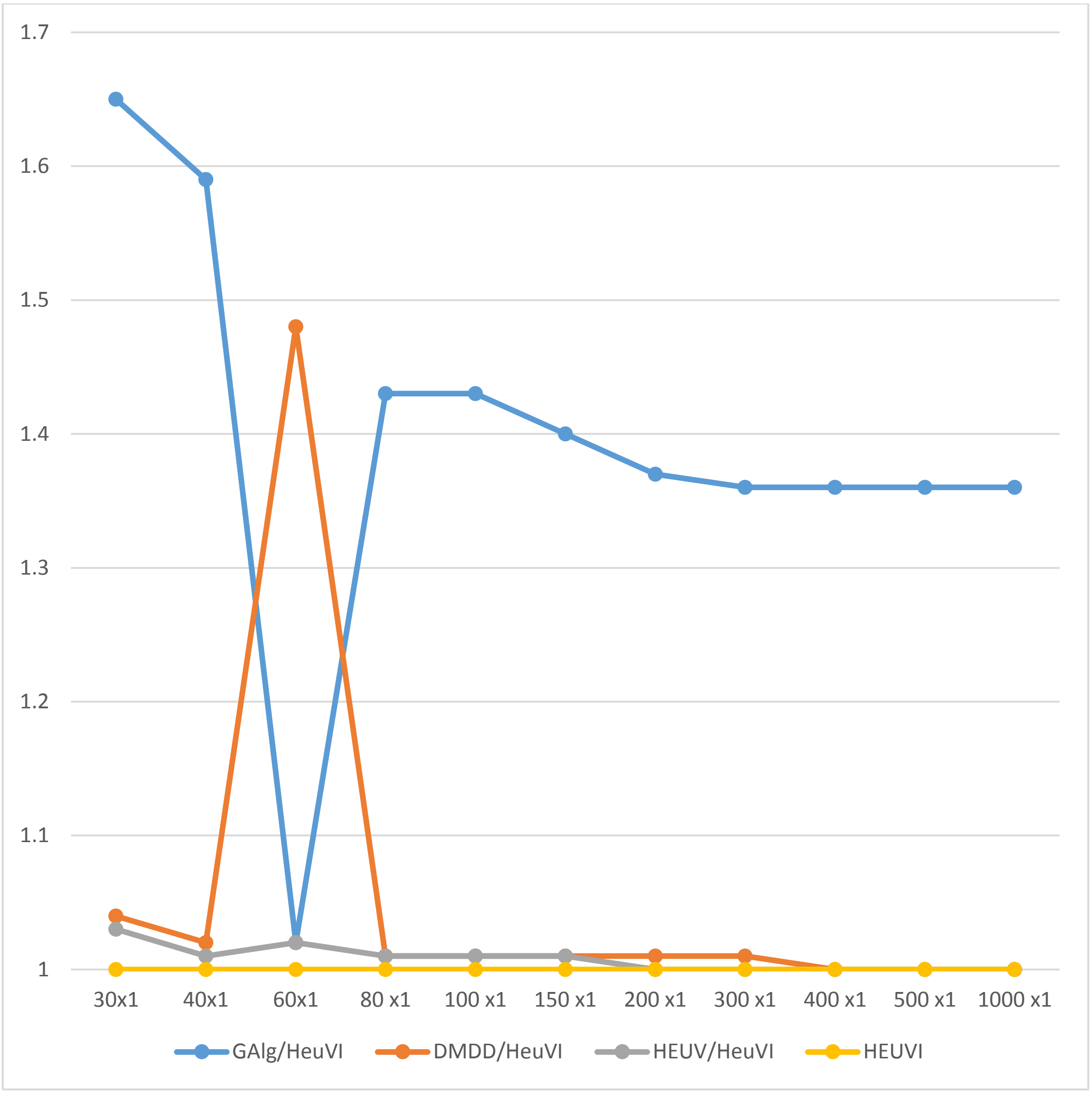


Figure 5.15: Plots of approximation ratio for problem sizes 30 ≤n ≤ 1000

From the results, it can be inferred that for problem sizes, $30 \leq n \leq 40$, the GAlg yielded the worst result. However, for $30 < n < 80$, the GAlg performed better than the DMDD.

Furthermore, Tables 5.54-5.56 show the overall means of approximation ratio of all the solution methods for the three problem sizes. Table 5.57a shows the results of the optimal count test for the heuristics. The result of t-test on the optimal count is shown in Table 5.57b

Table 5.54: The overall mean of approximation ratio for the problem sizes; $5 \leq n \leq 10$

Solution method	Overall means of approximation ratio
HeuVI	1.26
HeuV	1.41
DMDD	1.60
GAlg	2.33
BB	1.00

Table 5.55: The overall mean of approximation ratio for the problem sizes; $10 \leq n \leq 25$

Solution method	Overall means of approximation ratio
HeuVI	1.38
HeuV	1.43
DMDD	1.60
GAlg	2.64
BB	1.00

Table 5.56: The overall mean of approximation ratio for problem sizes; $30 \leq n \leq 1000$

Solution method	Overall means of approximation ratio
HeuV	1.006
GAlg	1.37
DMDD	1.05
HeuVI	1.00

Table 5.57: The Optimal Count test

S/No	Problem Sizes	HeuV	HeuVI	DMDD	GAlg
1	5 x 1	14	28	6	0
2	7x1	6	16	4	0
3	8x1	4	12	2	0
4	10 x 1	2	8	2	0
	Mean	**6.5**	**16**	**3.5**	**0**
	Standard deviation	**5.25**	**7.12**	**1.91**	**0**
5	15x1	0	0	0	0
6	20 x 1	0	0	0	0
7	25x1	0	0	0	0

Table 5.57b : The t-test for the optimal count result

Solution methods	GAlg	HeuV	DMDD	HeuVI
GAlg	----	<0.05*	<0.05*	<0.05*
HeuV	<0.05*	-------	>0.05	<0.05*
HeuVI	<0.05*	<0.05*	<0.05*	------
DMDD	<0.05*	>0.05	---------	<0.05*

Furthermore, in order to measure the consistency of the solution methods, the number of time each of the methods yielded the minimum value of the LCOF from the fifty problem instances were computed and expressed in percentages. This is shown in the Table 5.58. It can be deduced that the ranking order from the consistency test is in agreement with the results from the other tests.

Table 5.58: The consistency test

S/NO	GAlg	HeuVI	DMDD	HeuV	Rank
5 x 1	38	78	52	70	HeuVI, HeuV, DMDD, GAlg
7x1	24	74	44	66	HeuVI, HeuV, DMDD, GAlg
8x1	16	60	36	50	HeuVI, HeuV, DMDD, GAlg
10 x 1	12	52	30	44	HeuVI, HeuV, DMDD, GAlg
15x1	0	68	10	62	HeuVI, HeuV, DMDD, GAlg
20 x 1	0	74	5	64	HeuVI, HeuV, DMDD, GAlg
25x1	0	76	0	42	HeuVI, HeuV, DMDD, GAlg
30x1	0	68	0	54	HeuVI, HeuV, DMDD, GAlg
40x1	4	74	14	70	HeuVI, HeuV, DMDD, GAlg
60x1	8	64	10	60	HeuVI, HeuV, DMDD, GAlg
80 x1	0	78	0	70	HeuVI, HeuV, DMDD, GAlg
100 x1	0	70	0	66	HeuVI, HeuV, DMDD, GAlg
150 x1	0	76	0	72	HeuVI, HeuV, DMDD, GAlg
200 x1	0	78	0	74	HeuVI, HeuV, DMDD, GAlg
250 x1	0	80	0	76	HeuVI, HeuV, DMDD, GAlg
300 x1	0	76	0	72	HeuVI, HeuV, DMDD, GAlg
400 x1	0	74	0	70	HeuVI, HeuV, DMDD, GAlg
500 x1	0	80	0	76	HeuVI, HeuV, DMDD, GAlg
1000 x1	0	82	0	78	HeuVI, HeuV, DMDD, GAlg

5.5.2 Results based on the mean execution time

The execution time of an algorithm is the time taken to solve an instance of a given problem by the algorithm. Table 5.59 shows the mean of the total execution time by the solution methods and the problem sizes.

Table 5.59: Mean of the total execution time by solution methods and problem sizes

		Implemented		Proposed		
S/N	Problem Sizes	GAlg	DMDD	HeuV	HeuVI	BB
1	5 x 1	0.0064	0.00069	0.00021	0.00024	40.34
2	7x1	0.0069	0.00072	0.00026	0.00029	1924
3	8x1	0.0071	0.00072	0.00027	0.00032	3671
4	10 x 1	0.0076	0.00073	0.00029	0.00035	6542.3
5	15x1	0.0081	0.00084	0.00036	0.00049	12789.5
6	20 x 1	0.0093	0.00091	0.00045	0.00064	19876.5
7	25x1	0.0152	0.00095	0.00057	0.00076	30546.3
8	30x1	0.016	0.0010	0.00067	0.0011	
9	40x1	0.0237	0.0016	0.00086	0.0081	
10	60x1	0.0223	0.00207	0.00089	0.0011	
11	80 x1	0.0265	0.00298	0.001	0.0098	
12	100 x1	0.0307	0.00372	0.0029	0.0079	
13	150 x1	0.0454	0.00498	0.0035	0.0087	
14	200 x1	0.0533	0.0067	0.0014	0.0097	
15	300 x1	0.0808	0.0772	0.002	0.017	
16	400 x1	0.1032	0.0891	0.0028	0.029	
17	500 x1	0.1528	0.125	0.0032	0.035	
18	1000 x1	0.1943	0.1721	0.0045	0.048	

Results in the Table 5.59 shows that the HeuVI has the minimum execution time for the problem sizes considered. However, to ascertain whether the differences observed between the execution time of the HeuV and other heuristics are significant, t-test was also carried out. Tables 5.60-5.62 show the results of the t-test.

Table 5.60: The t-test for mean execution time for 5 ≤n ≤ 10 problems

Solution method	GAlg	DMDD	HeuV	HeuVI	BB
GAlg	----	<0.05*	<0.05*	<0.05*	<0.05*
DMDD	<0.05*	------	<0.05*	<0.05*	<0.05*
HeuV	<0.05*	<0.05*	-------	>0.05	<0.05*
HeuVI	<0.05*	<0.05*	>0.05	------	<0.05*
BB	<0.05*	<0.05*	<0.05*	<0.05*	-------

Table 5.61: The t-test for mean execution time for $10 \leq n \leq 25$ problems

Solution method	GAlg	DMDD	HeuV	HeuVI	BB
GAlg	----	<0.05*	<0.05*	<0.05*	<0.05*
DMDD	<0.05*	------	<0.05*	<0.05*	<0.05*
HeuV	<0.05*	<0.05*	-------	>0.05	<0.05*
HeuVI	<0.05*	<0.05*	>0.05	------	<0.05*
BB	<0.05*	<0.05*	<0.05*	<0.05*	--------

Table 5.62: The t-test for mean execution time for 30 $\leq$n $\leq$ 1000 problems

Solution method	GAlg	DMDD	HeuV	HeuVI
GAlg	----	<0.05*	<0.05*	<0.05*
DMDD	<0.05*	------	<0.05*	<0.05*
HeuV	<0.05*	<0.05*	-------	>0.05
HeuVI	<0.05*	<0.05*	>0.05	------

Note: *indicates significant result; Sample size = 50; -----indicates not necessary

The results show that the execution time of the heuristics are significantly different from each other ($p < 0.05$) except the HeuV and HeuVI. The execution time of HeuV and HeuVI are not significantly different from each other ($p > 0.05$).

Furthermore, the approximation ratio test was also carried out. The HeuV was used as the standard because it has the lowest execution time. Figures 5.16-5.18 show the plots of approximation ratio of the methods (See Appendix 1, Table H for the approximation Table). Furthermore, Tables 5.63-5.65 show the overall approximation ratio of the solution methods with respect to the HeuV.

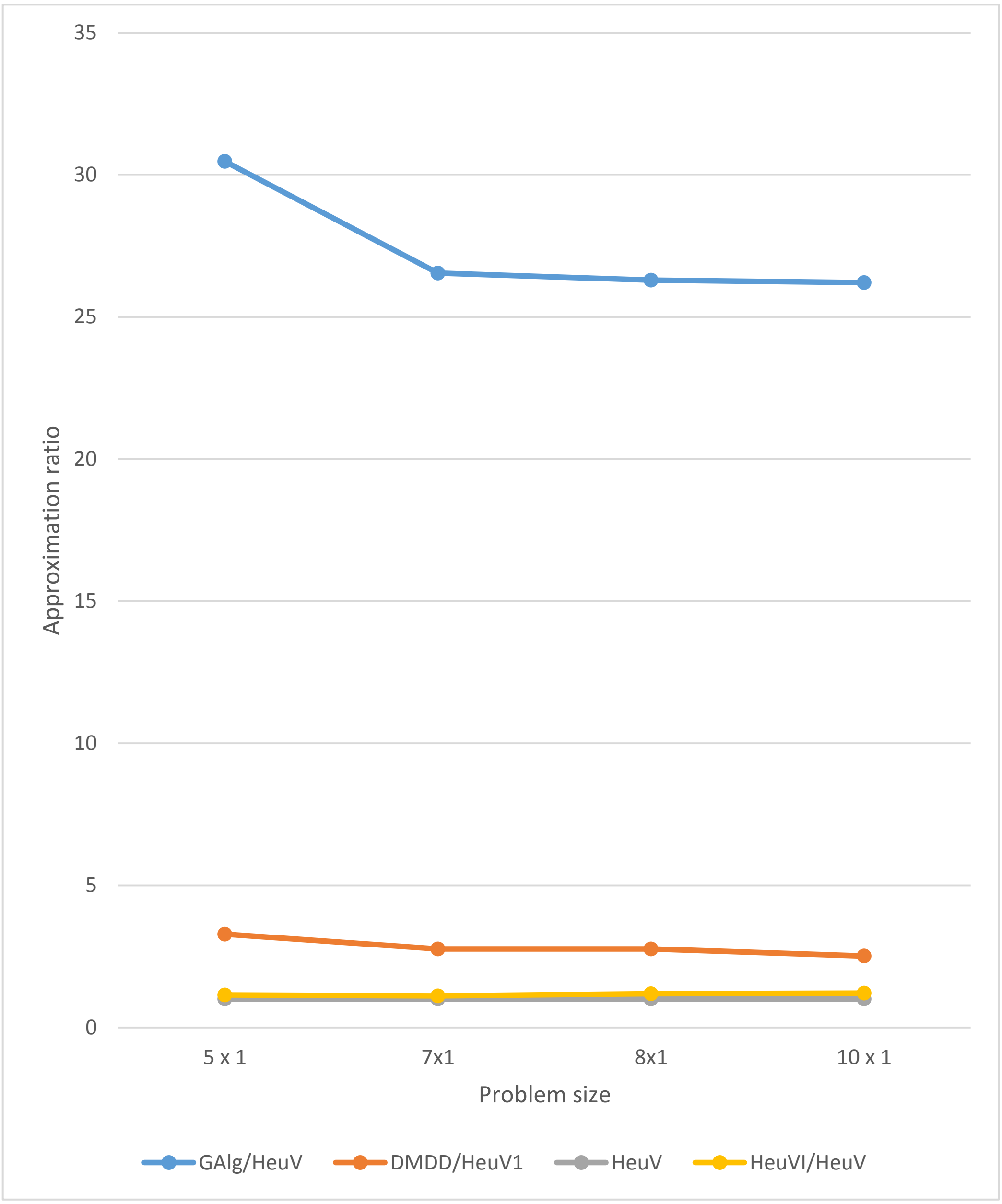


Figure 5.16: Plots of approximation ratio for problem sizes $5 \leq n \leq 10$

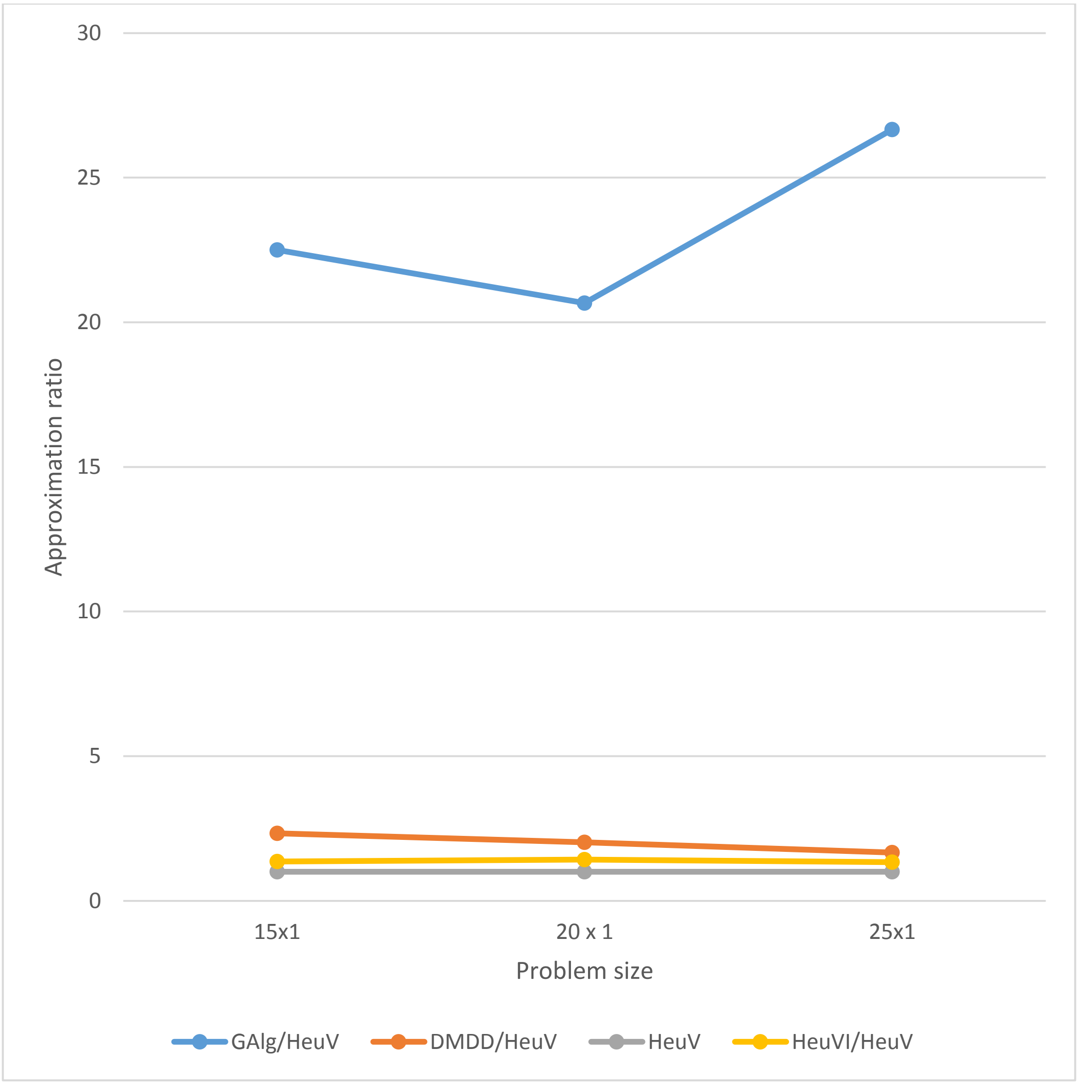


Figure 5.17: Plots of approximation ratio for problem sizes 15 ≤n ≤ 25

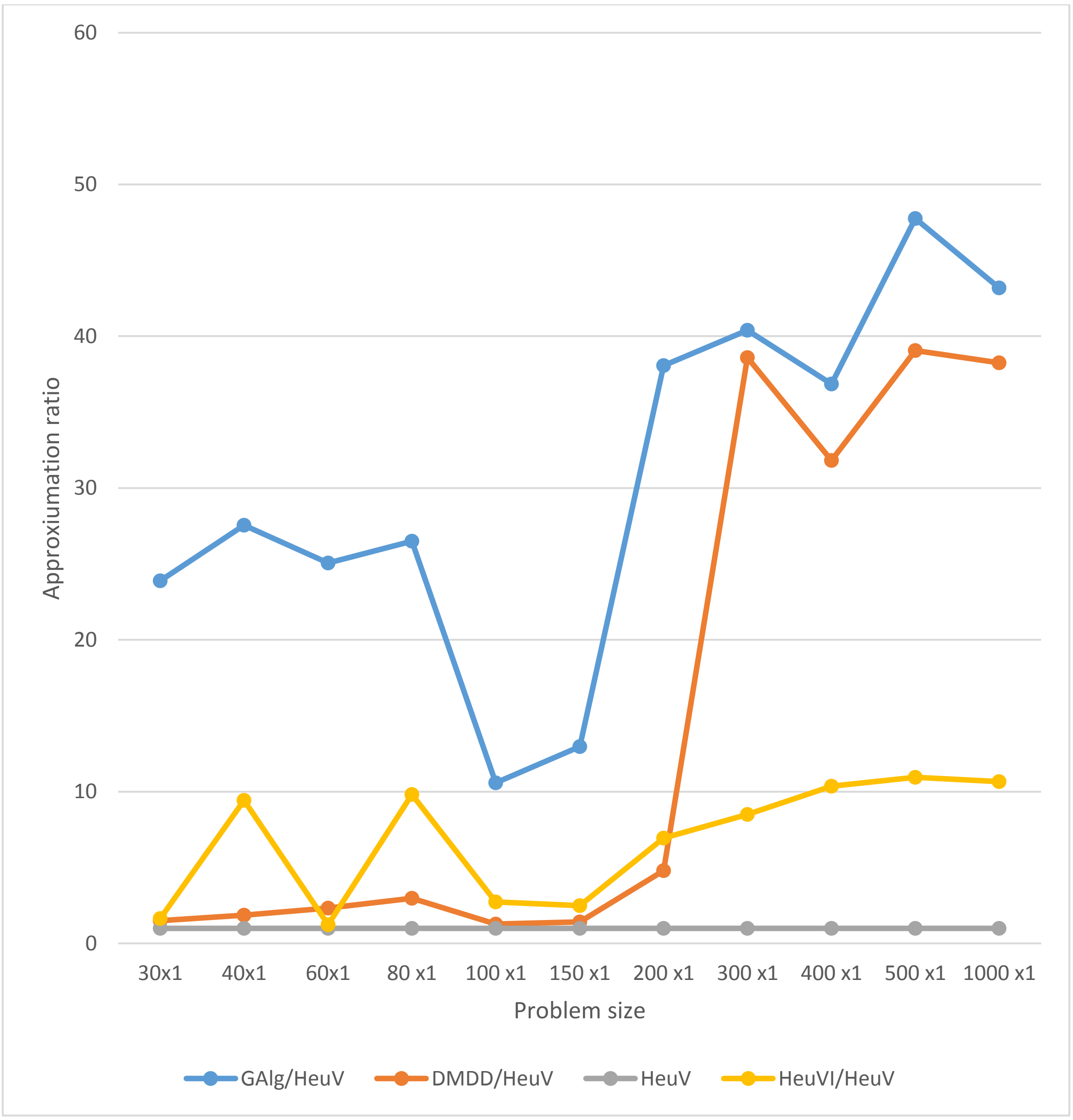


Figure 5.18: Plots of approximation ratio for problem sizes 30 ≤n ≤ 1000

Table 5.63: The overall mean of approximation ratio for the problem sizes; $5 \leq n \leq 10$

Solution Method	Overall means of approximation ratio
BB	1.1×10^7
HEUVI	1.16
DMDD	2.84
GAlg	27.38
HEUV	1.00

Table 5.64: The overall mean of approximation ratio for the problem sizes; 15 ≤n ≤ 25

Solution method	Overall means of approximation ratio
BB	4.4×10^7
HEUVI	1.37
DMDD	2.01
GAlg	23.28
HEUV	1.00

Table 5.65: The overall mean of approximation ratio for the problem sizes; 30 ≤n ≤ 1000

Solution method	Overall means of approximation ratio
HeuVI	6.79
DMDD	14..90
GAlg	30.26
HeuV	1.00

The results based on the mean execution time show that the proposed heuristic, HeuV is the most efficient solution method compare to other solution methods. For the three problems sizes, HeuVI is the closest to the standard efficient solution (HeuV). Therefore, comparing the effectiveness and efficiency of the two proposed HeuVI and HeuV, it can be inferred that HeuVI should be selected in preference to the HeuV solution method.

5.5.3 Real life results based on the dynamic variant of the bicriteria Problem

The implemented solution methods for the Class III problem as well as the scheduling policy (FCFS) employed by the firm for jobs received on different dates were applied on the real life data collected from the firm. The objective was to minimise the composite function of total tardiness and total flowtime. The results of the total normalized composite function obtained by all the solution methods are shown in Table 5.66.

Table 5.66: The mean of normalized total LCOF real life problem

Problem Size	Sample size	HEUVI	HEUV	DMDD	GAlg	FCFS
5x1	20	0.1076	0.1306	0.1461	0.1539	0.1308
10x1	5	0.2619	0.2936	0.3879	0.4930	0.3299
15x1	5	0.3682	0.4251	0.5243	0.5671	0.4876

In order to advise the firm on the best scheduling policy that suit its environments, the gain or loss from the use of any other scheduling policies compared to the FCFS were computed. These are shown in Tables 5.67 – 5.70.

Table 5.67: Average gain or loss by adopting the HeuV policy

Problem sizes	Sample size	HEUV	FCFS	Difference	Remark
5x1	20	0.1308	0.1310	+0.0002	Gain
10x1	5	0.2936	0.3299	+0.0363	Gain
15x1	5	0.4251	0.4876	+0.0625	Gain

Table 5.68: Average gain or loss by adopting the HeuVI policy

Problem sizes	Sample size	HeuVI	FCFS	Difference	Remark
5x1	20	0.1076	0.1310	+0.0234	Gain
10x1	5	0.2619	0.3299	+0.068	Gain
15x1	5	0.3862	0.4876	+0.1014	Gain

Table 5.69: Average gain or loss by adopting the GAlg policy

Problem sizes	Sample size	GAlg	FCFS	Difference	Remark
5x1	20	0.1539	0.1310	-0.0229	Loss
10x1	5	0.4930	0.3299	-0.1631	Loss
15x1	5	0.5671	0.4876	-0.0795	Loss

Table 5.70: Average gain or loss by adopting the DMDD policy

Problem sizes	Sample size	DMDD	FCFS	Difference	Remark
5x1	20	0.1461	0.1310	-0.0153	Loss
10x1	5	0.3879	0.3299	-0.058	Loss
15x1	5	0.5243	0.4876	-0.0367	Loss

However, since the objectives were normalized, the normalized average gains or losses were converted to percentage in order to make the best decision with regard to the preferred scheduling policy. The percentage gains or losses by adopting a new policy are shown in Tables 5.71-5.74.

Table 5.71: Percentage average gain or loss by adopting the HEUVpolicy

Problem sizes	Sample size	HEUV	FCFS	%gain (time)	Remark
5x1	20	0.1308	0.1310	+0.16%	Gain
10x1	5	0.2936	0.3299	+0.12%	Gain
15x1	5	0.4251	0.4876	+0.15%	Gain

Table 5.72: Percentage Average gain or loss by adopting the HeuVI policy

Problem sizes	Sample size	HeuVI	FCFS	%gain (time)	Remark
5x1	20	0.1076	0.1310	+22%	Gain
10x1	5	0.2619	0.3299	+26%	Gain
15x1	5	0.3862	0.4876	+68%	Gain

Mean of Percentage gain by adopting the HeuVI policy 39%

Standard deviation of Percentage gain by adopting the HEUVI policy ±25.4%

Table 5.73: Percentage gain or loss by adopting the GAlg policy

Problem Size	Sample size	GAlg	FCFS	% loss(time)	Remark
5 x 1	20	0.1461	0.1308	-17%	Loss
10x1	5	0.3879	0.3299	-49%	Loss
15x1	5	0.5243	0.4876	-16%	Loss

Table 5.74: Percentage average gain or loss by adopting the DMDD policy

Problem sizes	Sample size	DMDD	FCFS	% loss	Remark
5x1	20	0.1461	0.1308	-11.7%	Loss
10x1	5	0.3879	0.3299	-17.5%	Loss
15x1	5	0.5243	0.4876	-7.5%	Loss

Results obtained from Table 5.71 implied that by adopting the HeuV over the existing FCFS policy, the firm would experience positive changes (gain) in performance, which is expressed as the percentage change in time.

The results in Table 5.72 show that by adopting the HeuVI scheduling policy over the existing FCFS policy, the firm would experience a marked positive change (gain) in performance, which is expressed as the percentage change in time. The expected percentage gains in adopting the HeuVI policy was found to be far better than that of the HeuV policy. Thus, the HeuVI scheduling policy is recommended over the HEUV policy and the existing FCFS policy. Table 5.73 shows that by adopting the GAlg scheduling policy over the existing FCFS policy, the firm would experience a marked loss in their effectiveness. Table 5.74 shows that by adopting the DMDD scheduling policy over the existing FCFS policy, the firm would also experience a marked loss of effectiveness. Therefore, HeuVI scheduling policy is recommended for the firm for jobs with different release dates.

CHAPTER SIX

SUMMARY, CONCLUSIONS AND RECOMMENDATIONS

6.1 Introduction

This chapter highlights the summary of this work, the contribution of the study to knowledge and its benefits to the industrial policy. Conclusions were drawn based on the results obtained in this study and recommendations for future research are also outlined.

6.2 Summary

This study proposed some heuristics for scheduling problems with total tardiness related objectives. Based on the number of objectives and the subjected constraints, the problems solved were grouped into three classes:

CLASS 1: A scheduling problem of minimizing the 2TRD,

CLASS II: A bi-criteria scheduling problem of minimizing the 3TFZRD, and

CLASS III: A bi-criteria scheduling problem of minimizing the 3TFRD.

Six heuristics; HEUI, HEUII, HEUIII, HEUIV, HEUV and HEUVI were proposed. The HEUI and HEUII were proposed for the 2TRD problem and compared to the DMDD and the SPT from the literature. For the 3TFZRD problem, the HEUIII and HEUIV were compared to the HEUA and HEUB from the literature. The GAlg for solving multicriteria scheduling problems was also implemented for the problem. Due to the complexity status of the 3TFRD scheduling problem, algorithm were sparse that solves the problem. Nevertheless, the DMDD as well as the Generalized Algorithm (GAlg) were implemented for the problem and compared to the HEUV and HEUVI proposed for the problem. The BB was implemented for the problems for benchmarking. Fifty instances each for problem sizes ranging from 5-1000 jobs were randomly generated for evaluation, using MATLAB. Due to the prohibitive execution time, the BB was explored for problem sizes not exceeding twenty-five jobs. Statistically, a sample is referred to as large sample size if the sample size does not fall below 30 ($n \geq 30$). Exploring this fact, the simulated problems were grouped into the small-sized problems ($5 \leq n \leq 10$), medium-sized problems ($15 \leq n \leq 25$), and large-sized problems ($30 \leq n \leq 1000$).

Some real life problems involving 5, 10 and 15 jobs were also explored. For the 3TFZRD and 3TFRD bicriteria problems, the two objectives were expressed as the composite function and normalized. For the 3TFZRD problem, the existing normalization procedure in the literature was explored while a new procedure was proposed for the static scheduling constraints and applying to the 3TFZRD problem. The proposed corollary states that in order to determine the extreme values of a scheduling criterion for which a known optimal schedule or solution exists, the schedules corresponding to the two extreme values are notional. The corollary is based on the fact that for single processor scheduling problems, optimal or near-optimal solutions exist for some single criteria scheduling problems with zero release dates. The mean objective function and the mean execution time for all the implemented heuristics for each problem class were subjected to the Optimal Count (O.C), t-test ($p<0.05$), approximation Ratio (A.R) tests and consistency test to measure the effectiveness and efficiency of the heuristics for proper ranking.

Experimental results as well as the real life results show that the HEUII is recommended for small-and-medium sized problems, while the HEUI heuristic is recommended for the large-sized problems for 2TRD problem. For the 3TFZRD problem, the HEUIII is recommended for the small, medium and large sized problems. HEUVI is also recommended for the 3TFRD problem.

6.3 Contribution to Knowledge

Exploring scheduling approaches to minimise total tardiness related objectives in a single processor environment are NP hard problems with numerous application to industrial practices. Thus, the problem is of interest to researchers and industrialists. This work has contributed to the state of knowledge by proposing an effective and efficient scheduling heuristics for solving 2TRD problem. The heuristic (HEUII) yielded the results that are not significantly different from the optimal for small and medium sized problems ($n \leq 25$).

Similarly, two effective and efficient heuristics for the bicriteria scheduling problem of minimizing 3TFZRD were also proposed. In terms of effectiveness, the two proposed heuristics are not significantly different with respect to the optimal for number of jobs not exceeding twenty five jobs. Also the proposed heuristics are statistically better than the heuristics found in the literature. For small-sized problems, one of the proposed heuristics (HEUIII) yielded an optimal count of 90% on the average.

The need for normalization of objectives in multicriteria scheduling problems and the required procedure for non-zero release dates constraint has been established in the literature. However, the normalization procedure for multicriteria scheduling problems under static condition is a missing link. This work fills the gap.

Customer satisfaction as well as efficient resources utilization is a challenge in production and servicing firms. This study shows that there will be a significant boost in the effectiveness without compromising the efficiency of a firm exploring the first come first serve schedule policy, if the proposed policy in this work is adopted. This was reflected in the percentage gain observed when the utility of the proposed heuristics with real life data was demonstrated for the 3TFRD. Furthermore, for some firms like shipping firms, auto repair firm, auto painting firms, etc, where numerous jobs are available simultaneously for processing, there will be a significant boost in performance, if the proposed policies were explored compared to the SPT. This was reflected in the percentage gain observed when the utility of the proposed heuristics with real life data was demonstrated for the 3TFZRD.

6.4 Future Research and Recommendation

Although the solution methods proposed in this work yielded good results. Further research can be carried out by imposing other constraints on all the problem classes. Also, more solution methods can be explored, especially for the Class II and Class III problems.

6.5 Conclusion

Based on the results obtained, it was concluded that for the problem of minimizing the 2TRD, the heuristic HEUII is recommended if the number of job does not exceed 25. This is because the method produced a result (in terms of effectiveness) that is not significantly different from the optimal for small problem sizes. Though, the method requires a significantly different execution time from the most efficient method (SPT), the HEUII heuristic is polynomially-time bound. However, if the number of accumulated jobs is greater than twenty-five (25), the heuristic HEUI is recommended. This is because the method yielded the most effective results compared to other heuristics. For the bicriteria scheduling problem of minimizing 3TFZRD, the heuristics; HEUIII is recommended. This is because the solution method yielded results that are not significantly different from the optimal for small sized problems ($n \leq 25$) and also produced an optimal of 90% on the average. Also, the method obtains a polynomial-time bound. For the bicriteria scheduling problem of minimizing 3TFZRD, the HEUVI heuristic is recommended. Though the model yields results that are significantly different from the optimal. The approximation ratio tests for the small and medium sized problems show that the method is effective in solving the problem. In terms of effectiveness, the method is also significantly better than other solution method.

REFERENCES

Anderson, E. J. and Nyirenda, J. C. 1990. Two new rules to minimise tardiness in a job shop. *International Journal of Production Research* 28: 2277–2292.

Ali, M.D. 2016. Approximation algorithms to solve simultaneous multicriteria scheduling problems. *International Journal of Application or Innovation in Engineering & Management* 5.1: 1- 6.

Abdullah, H. F. 2010. Multicriteria scheduling problems to minimise total tardiness subject to maximum earliness or tardiness. *Ibn Al- Haitham Journal for Pure and Applied Science* 23.1: 1- 6.

Akande, S., Oluleye, A.E., and Oyetunji E.O. 2014. Reducibility of some multi criteria scheduling problems to bi-criteria scheduling problems. *Proceedings of the 2014 International Conference on Industrial Engineering and Operations Management*. Indonesia. 642-651.

Akande, S., Oluleye, A.E., and Oyetunji, E.O. 2015. On the reducibility of some multi criteria scheduling problems to bi-criteria scheduling problems. *International Conference on Industrial Engineering and Operations Management*. Dubai. 1-8.

Akande, S., Oluleye, A.E., and Oyetunji, E.O. 2015. Normalization of composite objective function for multicriteria scheduling problems with zero release dates. *Proceedings of the 2015 International Conference on the Challenge of Productivity and National Development, Nigeria Institute of Industrial Engineers*. Nigeria. 115-128.

Al-harkan, I. *Algorithms for Sequencing and Scheduling.* [online]
Available from: faculty.ksu.edu.sablog
(http://http://faculty.ksu.edu.sa/ialharkan/IE428) [Accessed: May, 21st 2013].

Akinyemi, O.O., Oluleye, A.E. and Akinbinu, A.F. 2002. Job scheduling against due dates using a single algebraic sequencing rule: A Case Study of an Electrical Repair Workshop. *Nigerian Journal of Engineering Management* 3.4: 1-6.

Bao, Z., Wang, W., Wang, P., and Quanke, P. 2012. Research on production scheduling system with bottleneck based on multi-agent. 2012 *International Conference on Applied Physics and Industrial Engineering*, Netherland. 27-37.

Bansal, N. and Kulkarni, J. 2015. Minimizing flow-time on unrelated machines. *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing.* Portland. 92-102.

Bianchi, L., Marco, D., Luca, M.G., and Walter, J. G. 2009. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing: an International Journal* 8.2: 239–287.

Blum, C. and Roli, A. 2003. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys* 35.3: 268–308.

Brucker, P. 2007. *Scheduling Algorithms,* 5th ed. Springer: Heidelberg Publisher.

Brian D. H and Daniel T. V. 2007. *Essential MATLAB for Engineers and Scientists*. 3rd ed. Amsterdam : Butterworth-Heinemann.

Baptiste P., Carlier J. And Jouglet A 2004. A branch and bound procedure for minimizing total tardiness with arbitrary release date. *European Journal of Operational Research* 15.8: 595–608.

Branke, J. Belton, V., Eskelinen, P., Greco, S., Molina, J., Ruiz, F. and Slowinski, R. 2008. *Multi objective Optimization: Interactive and Evolutionary Approaches*. 1st ed. Verlag Berlin Heidelberg : Springer

Bean, J.C. and Hall, D.H. 1985. Accuracy of the Modified Due Date Rule, Technical Report 85-10, Department of Industrial and Production Engineering, Michigan University. 1-7.

Baker, K.R. and Bertrand, J.W. 1982. A dynamic priority rule for scheduling against due-dates. *Journal of Operational Management* 3.1: 37-42.

Baker, K.R and Trietsch, D. 2013. *Principles of sequencing and scheduling*. Ist ed. Canada: John Wiley & Sons

Cox, J.F., Blackstone, J.H., and Spencer, M.S. 1992. *APICS Dictionary, American Production and Inventory Control Society* 1st ed. Virginia: Falls Church.

Chu, C. and Portmann, M.C. 1992. Some new efficient methods to solve the 1/ri/ $\sum_{i=1}^{n} T_i\,(\min)$ scheduling problem. *European Journal of Operational Research* 58: 404–413.

Cochran, J.K., Shwu, H. and Fowler, J.W. 2003. A multi-population genetic algorithm to solve multi-objective scheduling problems for parallel machines. *Computers and Operations Research* 30.7: 1087– 1102.

Della Croce F., Tsouki, A., and Moraıtis, P. [online] Why is difficult to make decisions under multiple criteria
Available from: www.math-info.univ-paris5.fr
[Accessed: April16th, 2016].

Chu, C. 1992. A Branch-And-Bound algorithm to minimise total flow time with unequal release dates. *Naval Research Logistics* 39: 859–875.

Du, J. and Leung, Y.T. 1990. Minimizing total tardiness on one machine is NP-Hard. *Mathematics of Operations Research* 15.3: 483–495.

Deb, K., 2001. *Multi-Objective optimization using evolutionary algorithms*. 1st Ed. Canada : John Wiley.

Ehrgott, M., and Gandibleux, X., 2000. A Survey and Annotated Bibliography of Multi Objective Combinatorial Optimization. *Operation Research Spectrum*. 22: 425–460.

Erenay, F.S., Sabuncuoglu, I., Toptal, A. and Tiwari, M.K. 2010. New solution methods for single machine bi-criteria scheduling problem: minimization of average flow time and number of tardy jobs. *European Journal of Operation Research* 201.1: 89 – 98.

Eren T. 2007. A multi-criteria scheduling with sequence-dependent setup times. *Applied Mathematical Sciences* 1.58: 2883 – 2894.

Weisstein, E W. Complexity Theory. A Wolfram Web Resource. http://mathworld.wolfram.com/ComplexityTheory.html [Accessed: June, 21st 2015].

French, S. 1982. *Sequencing and Scheduling*, 1st Ed. Ellis USA : Horwood Limited

Farhad, K., and Vahid, K. 2009. A heuristic approach for scheduling of multi-criteria unrelated parallel processors. *World Academy of Science, Engineering and Technology* 59: 253-261

Fera, M., Fruggiero, F., Lambiase, A., Martino, G., and Nenni, M.E., 2015. Production Scheduling Approaches for Operations management. [online] (http://creativecommons.org/licenses/by/3.0), [Accessed: December 21st 2015].

Gantt, H. 1916. *Industrial Leadership New Haven*. Ist Ed. USA: Yale University Press

Ghosh D. S. and Nagi, R. 2007. Batch splitting in an assembly scheduling environment. *International Journal of Production Economics*, 105.2: 372-384.

Gürsel, A., Yang, S. X., Alhawari, O. I., Santos, J., and Vazquez, R. 2012. A genetic algorithm approach for minimizing total tardiness in single machine scheduling. *International Journal of Industrial Engineering and Management* 3.3: 163-171

Haidar, Y.K. and Tariq, S.A. 2011. Single machine scheduling to minimise three hierarchically criteria. *Journal of Basrah Research Sciences* 37.4: 263-271.

Heidi, A. And David, W. 2006. Multiple Objective Scheduling Problems: Determination of Pruned Pareto Sets, Technical Report. Department Of Industrial and Systems Engineering, Rutgers University, 76-85.

Hoogeveen, H. 2005. Multi Criteria scheduling. *European Journal of Operational Research* 167: 592-623.

Ilhem, B., Julien, L. and Patrick, S. 2013. A survey on optimization metaheuristics. *Journal of Information Sciences* 23.7: 82–117.

Jeffrey, W. H. 2008. A history of production scheduling. *International Series in Operations Research and Management Science* 1-22.

Kanet, J.J. and Hayya, J.C. 1982. Priority dispatching with operation due-dates in a job shop. *Journal of Operations Management* 2: 155–163.

Kaplanoglu, V. 2014. Multi-Agent based approach for single machine scheduling with sequence-dependent setup times and machine maintenance. *Journal of Applied Soft Computing* 23: 165-179.

Karger, D.R., Phillips S.J., and Torng, E. 1996. A better algorithm for an ancient scheduling problem. *Journal of Algorithms* 20:400–430.

M’Hallah, R. 2007. Minimizing total earliness and tardiness on single machine using hybrid heuristic. *Computers and Operations Research* 34: 3126–3142.

Miyazaki, S.1981. Combined scheduling system for reducing job tardiness in a job shop. *International Journal of Production Research* 19: 201–211.

Michael, W., Michael, V. H., Ralf, H., Joachim, M. and Eberhard, A. 2015. Real time bottleneck detection and prediction to prioritize fault repair in interlinked production lines. Procedia CIRP. *Understanding the life cycle implications of manufacturing*. 37: 140 – 145.

Muhlemann, A.P., Lockett, A. G., and. Farn, C. I. 1982. Job shop scheduling heuristics and frequency of scheduling. *International Journal of Production Research* 20: 227–241.

Nagar A, Haddock J., and Heragu S., 1995. Multiple and bicriteria scheduling: a literature survey. *European Journal of Operational Research* 81.1: 88-104.

Naidu J.T. 2003. A Note on a Well-Known Dispatching Rule to Minimise Total Tardiness. *International Journal of Management Science* 31.2 : 137–140.

Neda, K. and Hamid D. 2015. A branch and bound method for solving multi-factory supply chain scheduling with batch delivery. *Journal of Expert Systems with Applications* 42.1: 238-245.

Oyawale, F. A. 2006. *Statistical methods: An introduction*.1st ed. Nigeria : International Publisher Ltd.

Oyetunji, E. O., Oluleye, A. E. and Akande, S.A. 2012. Approximation algorithms for minimizing sum of flow time on single machine with release dates. *International Journal of Modern Engineering Research* 2.3: 687-696.

Oyetunji, E.O., 2009. Some common performance measures in scheduling problems. *Research Journal of Applied Science, Engineering and Technology* 1.2: 6-9.

Oyetunji, E.O. and Oluleye, A.E. 2008. Heuristics for minimizing the total completion time and number of tardy jobs simultaneously on single machine with release time. *Research Journal of Applied Sciences* 3.2: 147–152.

Oyetunji, E.O. and Oluleye, A.E. 2009. Evaluating solution methods to bicriteria scheduling problems. *Advanced Materials Research* 62.64: 577–584.

Oyetunji, E.O. and Oluleye, A.E. 2010. Hierarchical minimization of total completion time and number of tardy jobs criteria. *Asian Journal of Information Technology* 7.4: 157-161.

Oyetunji, E.O. 2011. Assessing solution methods to mixed multi-objectives scheduling problems. *International Journal of Industrial and Systems Engineering* 9.2: 213-226.

Oyetunji, E.O. and Oluleye, A.E. 2012. A Generalized Algorithm for solving multicriteria scheduling problems. *Advanced Materials Research* 36.7: 653-666.

Oladokun, V.O., Charles-Owaba O.E, and Olaosebikan F.O. 2011. A bi-criteria algorithm for the simultaneous minimization of makespan and number of tardy jobs on single machine with sequence dependent set-up time. *Research Journal of Applied Science, Engineering and Technology* 3.9: 1048-1051.

Panneerselvam, R., 2007. *Design and Analysis of Algorithm.*1st ed. India : Prentice Hall Limited.

Parviz, F., 2009. A hybrid multi objective algorithm for flexible job shop scheduling. *World Academy of Science, Engineering and Technology* 50: 551-556

Papadimitriou, C.H., 1994. *Computational Complexity*. 1st ed. New York : Addison-Wesley.

Pinedo, M.L., 2008. *Scheduling – Theory, Algorithms and Systems*.1st ed. New York : Springer

Potts C.N., and Van Wassenhove, L.N. 1982. A decomposition algorithm for the single-machine total tardiness problem. *Operations Research Letters* 1: 177–181

Phillips, C., Stein, C., and Wein, J. 1998. Minimizing average completion time in the presence of release dates. *Mathematical Programming* 82: 199–224.

Rajendran, C., 1995. Theory and methodology of heuristics for scheduling in flow shop with multiple objectives. *European Journal of Operation Research* 82: 540 −555.

Rahimi V. R. and Mirghorbani S. M. 2007. A multi-objective particle swarm for flow shop scheduling problem. *Combinatorial Optimization* 13.1:79–102

Roscoe, E.S., and Freark, D.G., 1971. *Organization for Production*. 5th ed. Illinois: Irwin, Inc.

Sipser, M., 1997. *Introduction to the Theory of Computation*. 1st ed. California: Belmont.

Skinner, W., 1985. *The Taming of The Lions: how manufacturing leadership evolved*. 2nd ed. Boston : Harvard Business School Press.

Sen, T. and Dillepan, O.1999. Bicriteria scheduling problems involving total tardiness and total flow time. *Journal of Information and Optimization Science* 20.2: 155-170.

Smith, W. E., 1956. Various Optimisers for single-stage production. *Naval Research Logistic Quarter* 3.1: 59-66.

Tapan P. B., (2012). *Multiobjective Scheduling by Genetic Algorithms*. 1st edition, USA: Springer Science & Business Media.

Tabucanon, M. T. and Cenna, A. A. 1991. Bicriteria scheduling problem in a job shop with parallel processor. *International Journal of Production Economics* 25.3:95-101.

Taha, H. 2007. *Operation Research; An Introduction*. 8th Ed. USA: Pearson Prentice Hall.

T'Kindt, V. and Billiant, J.C. 2005. *Multicriteria scheduling, theory, model and algorithm*. 2nd ed. USA: Mc Graw Hill.

Ulungu, E.L., Teghem, J., Fortemps, P. H., and Tuyttens, D., 1999. MOSA Method: A tool for solving problems. *Journal of Multi-Criteria Decision Analysis* 8: 221–236.

Van Wassenhove, L. N. and Gelders, F. 1980. Solving a bicriteria scheduling problem. *European Journal of Operational Research* 4.1: 42-48.

Vladimír, M.R. and Sudhakara, P., 2010. Flow shop scheduling algorithm to minimise completion time for n-jobs m-machines problem. *Technical Gazette* 17.3: 273-278.

Yang, H., Sun, S., and Wang, S. 2006. A heuristic algorithm for dynamic identify bottleneck machine. *Manufacturing Automation* 9: 21-24.

APPENDICES

APPENDIX A1 : Validity of the Normalization Procedure

This example will justify the need for normalization as well as the validity of the proposed corollary. Consider a single machine, 4 jobs bicriteria scheduling problem for minimizing the total flowtime and the total number of tardy jobs. The processing time and the due date of each jobs as shown in Table A.

Table A: Hypothetical problem to illustrate the need for normalization

Job	P (hour)	D(hour)
1	3.0	5
2	5.5	4
3	3.5	6
4	7.0	10

SPT algorithm yields optimal solution for the total flow time only. In this work, SPT schedule is 1, 3, 2, 4. The schedule can be represented by chant chart as shown in Figure A

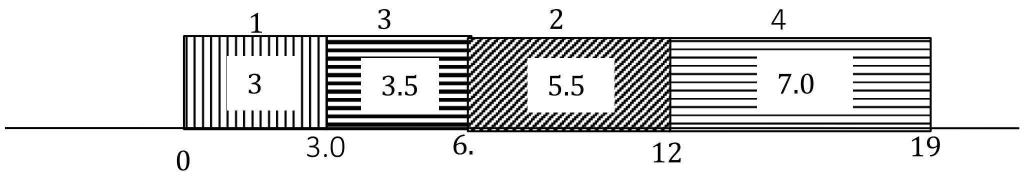


Figure A1: The Gantt chart of the SPT schedule.

The optimal total flowtime produced by SPT = 3.0 + 6.5 + 12 + 19 = 40.5hour

The total number of tardy jobs produced by SPT = 3jobs

Assuming the two criteria have equal weight of 0.5.

Without normalization, the composite objective function is given by;

$F(X,Y) = \alpha X + \beta Y$ = 0.5(40.5) hour + 0.5(3)jobs

20.25hours + 1.5job = 21.75 (hour +job)

Therefore, this is an unbalanced LCOF since the unit of flow time and number of tardy jobs are not the same. Similarly, the SPT does not yield a good result for total tardiness problem. However, this becomes insignificant in the LCOF because flow time objective dominates over that of the number of tardy jobs. Thus the LCOF is biased or skewed towards the flow time.

With normalization

Argument against the corollary

SPT algorithm yields optimal solution for the total flow time only. In this work, SPT schedule is 1, 3, 2, 4. The schedule can be represented by chant chart as shown in Figure A2

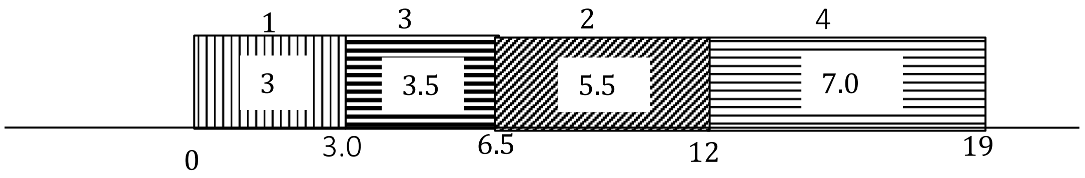


Figure A2. The Gantt chart of the SPT schedule.

The optimal total flowtime produced by SPT = 3.0 + 6.5 + 12 + 19 = 40.5hour

The minimum possible total flow time, $F_k = P_1 \ + \left\{\sum_{i=1}^{k-1} P_i + \ \sum_{i=2}^{n} P_i \ \right\}$

$$= \ 3.0 + (3 + 3.5\ ) + (3 + 3.5 + 5.5\ ) + (3 + 3.5 + 5.5 + 7.0) = 40.5hour$$

The maximum possible total flow time = $P_1 + \sum_{k=2}^{n}\left\{\sum_{i=1}^{k-1} P_i + P_k\right\}$

$$= \ 3.0 \ + (\ (3.0 \ + 3.5) \ + \ (3.0 \ + 3.5 \ + 5.5) \ + \ (3.0 \ + 3.5 \ + 5.5 \ \ + \ 7.0)$$

$$3.0 \ + \ 6.5 \ + \ 12 \ + \ 19 \ = \ 40.5 \text{ hours}$$

The normalized value of flow time is given by;

$$X_N = \frac{X - X_{min}}{X_{max} - X_{min}} \ = \frac{40.5-40.5}{40.5-40.5} \ = \frac{0hours}{0hours} \ = \text{indeterminate}$$

Argument in favour of the corollary

The minimum possible total flowtime, $F_{tot}^{min} = \sum_{1}^{n} P_i = 3.0 + \ 3.5 + 5.5 + 7.0 = 19$

The maximum possible total flowtime = $n\{\sum_{i=1}^{n-1} P_i\} \ + \ \sum_{i=1}^{n} P_i$

$$4(3 + 3.5 + 5.5) \ + \ (3 + 3.5 + 5.5 \ + 7.0) \ = \ 4(12) \ + \ 19 \ = \ 48 \ + 19$$
$$= \ 67hours$$

The normalized value of flow time is given by;

$$X_N = \frac{X - X_{min}}{X_{max} - X_{min}} \ = \frac{40.5-19}{67-19} \ = \frac{21.5hours}{48hours} \ = 0.4479$$

The total number of tardy jobs produced by SPT = 3

The minimum possible number of tardy jobs = 0

The maximum possible number of tardy jobs = number of jobs= 4

The normalized value of total number of tardy jobs is given by;

$$X_N = \frac{X - X_{min}}{X_{max} - X_{min}} \ = \frac{3-0}{4-0} \ = \frac{3jobs}{4jobs} \ = 0.75$$

The composite objective function;

F(X,Y) = α X + β Y = (0.5) 0.4479 + (0.5) 0.75 = 0. 4515

This is a balanced LCOF since both the flow time and the number of tardy jobs have been converted to a dimensionless parameter by normalization. Similarly, none of the objectives dominates the other. The high value of the normalized total number of tardy jobs observed is due to the fact that SPT does not produce a good result for scheduling problems to minimise the number of tardy jobs. Similarly, the lower value of normalized flow time observed is due to the fact that SPT yields the optimal for flow time single machine scheduling problem with zero release dates. Thus, a balanced and unbiased composite objective function is obtained by normalization.

APPENDIX AII: Code Listing : Pseudocode of the scheduling problem generation for SPTTP with release dates problems (CLASS I)

Code Written by : Saheed Akande

Language : MATrix LABoratory

Date : June 2016

Project: Ph.D Thesis - DEVELOPMENT OF HEURISTICS FOR SINGLE PROCESSOR SCHEDULING PROBLEMS WITH TARDINESS- RELATED PERFORMANCE MEASURES

Department : Industrial and Production Engineering

```
MinProcessTime = 1;
MaxProcessTime = 10;
ProcessTimeGen = 9;
MaxReleaseDate = 40;
MinReleaseDate =0;
nojobs = 5;
rowc = 5;
wtime = zeros(rowc, nojobs);
ProcessTime = zeros(rowc, nojobs);
ReleaseDate = zeros(rowc, nojobs);
DueDate = zeros(rowc, nojobs);
MaxProcessTimeDDC = zeros(rowc, nojobs);
MinProcessTimeDDC = zeros(rowc, nojobs);
ProcessTimeDDCGen = zeros(rowc, nojobs);
Jobs = zeros(rowc, nojobs);

for row = 1:rowc

    Jobs(row, :) = 1:nojobs;
    ProcessTime(row, :) = floor(ProcessTimeGen *rand([1, nojobs]))+1;

    ReleaseDate(row, :) = floor(MaxReleaseDate *rand([1,nojobs]));

    MaxProcessTimeDDC(row, :) = 4*ProcessTime(row, :);

    MinProcessTimeDDC(row, :) = 1* ProcessTime(row, :);

    ProcessTimeDDCGen(row, :) = MaxProcessTimeDDC(row, :) -
      MinProcessTimeDDC(row, :) ;

    DueDate(row, :) =  (ReleaseDate(row, :)) +
(floor(ProcessTimeDDCGen(row, :).*rand([1,nojobs]))+
1*MinProcessTimeDDC(row, :) );

end

save('problemG.mat','Jobs', 'ProcessTime', 'ReleaseDate', 'DueDate')
```

APPENDIX AII: Code Listing : Pseudocode of the single instance solution for Heu IIfor SPTTP with release dates problems (CLASS I)

Code Written by : Saheed Akande

Language : MATrix LABoratory

Date : June 2016

Project: Ph.D Thesis - DEVELOPMENT OF HEURISTICS FOR SINGLE PROCESSOR SCHEDULING PROBLEMS WITH TARDINESS- RELATED PERFORMANCE MEASURES

Department : Industrial and Production Engineering

```
function [execTime, sumtard, order] =
AA6Instance(processTime,releaseDate, dueDate)
% execTime is the time to run an instant of problem
% lcof is the linear composite objective function of an instant of
problem
% order is the final schedule of an instant of problem
%



tic

noJobs = length(processTime); % n = number of jobs = length of array
processTime
                                   %or releaseDate or dueDate

        FactorTime = dueDate;

        FactorTime1 = processTime + releaseDate;

    [SortedFactorTime,JobSetB] = sort(FactorTime);

    [SortedFactorTime1,JobSetC] = sort(FactorTime1);

    CompletionTimeJobSetB =  zeros(1, noJobs);

    for k = 1

    CompletionTimeJobSetB(1) = processTime(JobSetB(1)) +
releaseDate(JobSetB(1));

    end

    for k = 2:noJobs

        if  CompletionTimeJobSetB(k-1)> releaseDate(JobSetB(k))

        CompletionTimeJobSetB(k) = CompletionTimeJobSetB(k-1) +
processTime(JobSetB(k));
```

```
        else
            CompletionTimeJobSetB(k) = releaseDate(JobSetB(k))+
processTime(JobSetB(k));
        end
    end

    TardinessJobSetB = zeros(1, noJobs);

    for k =1:noJobs

        if CompletionTimeJobSetB(k)> dueDate(JobSetB(k))

            TardinessJobSetB(k) = CompletionTimeJobSetB(k) -
dueDate(JobSetB(k));

        end
    end

    SumTardJobSetB = sum(TardinessJobSetB);

     CompletionTimeJobSetC =  zeros(1, noJobs);

    for k = 1

    CompletionTimeJobSetC(1) = processTime(JobSetC(1))+
releaseDate(JobSetC(1));

    end

    for k = 2:noJobs

        if CompletionTimeJobSetC(k-1) > releaseDate(JobSetC(k))

                CompletionTimeJobSetC(k) = CompletionTimeJobSetC(k-1)
+ processTime(JobSetC(k));

        else

            CompletionTimeJobSetC(k) =  releaseDate(JobSetC(k))  +
processTime(JobSetC(k));

        end

    end


    TardinessJobSetC = zeros(1, noJobs);

    for k =1:noJobs

        if CompletionTimeJobSetC(k)> dueDate(JobSetC(k))

            TardinessJobSetC(k) = CompletionTimeJobSetC(k) -
dueDate(JobSetC(k));

        end
    end
    SumTardJobSetC = sum(TardinessJobSetC);


    OptSeq = zeros(1, noJobs);

    for k =  1 : noJobs

       if TardinessJobSetB(k) - TardinessJobSetC(k) <= 0

       OptSeq(k) = JobSetB(k);

       else

            OptSeq(k) = JobSetC(k);
       end

     end


    N = OptSeq;

    JobSetU = RTW.unique(OptSeq);

    L = length(JobSetU);

    NoRepJobs = noJobs - L;

    [JobNotSch1, indexINB] = setdiff(JobSetB, OptSeq);


    [JobNotSch2, indexINC] = setdiff(JobSetC, OptSeq);

      FOptSeq = [JobSetU,JobNotSch2];

    CompletionTimeFOptSeq =  zeros(1, noJobs);

     for k = 1

     CompletionTimeFOptSeq(1) = processTime(FOptSeq(1))+
releaseDate(FOptSeq(1)) ;

    end

    for k = 2:noJobs

        if CompletionTimeFOptSeq(k-1)> releaseDate(FOptSeq(k-1))

        CompletionTimeFOptSeq(k) = CompletionTimeFOptSeq(k-1) +
processTime(FOptSeq(k));
        else

         CompletionTimeFOptSeq(k) = processTime(FOptSeq(k))+
releaseDate(FOptSeq(k)) ;
        end
    end
    TardinessFOptSeq = zeros(1, noJobs);

    for k =1:noJobs

        if CompletionTimeFOptSeq(k)> dueDate(FOptSeq(k));

            TardinessFOptSeq(k) = CompletionTimeFOptSeq(k) -
dueDate(FOptSeq(k));
        end
    end

    if sum(TardinessJobSetC) > sum(TardinessFOptSeq)

        sumtard = sum(TardinessFOptSeq);

            execTime = toc;

            order = FOptSeq;

    else

            sumtard = sum(TardinessJobSetC) ;

            execTime = toc;

            order = JobSetC;
    end

end
```

APPENDIX AII: Code Listing : Pseudocode of the Execution file for HeuII for SPTTP with release dates problems (CLASS I)

Code Written by : Saheed Akande

Language : MATrix LABoratory

Date : June 2016

Project: Ph.D Thesis - DEVELOPMENT OF HEURISTICS FOR SINGLE PROCESSOR SCHEDULING PROBLEMS WITH TARDINESS- RELATED PERFORMANCE MEASURES

Department : Industrial and Production Engineering

```
function [meanSUMTARD,meanExecTime,SUMTARD,ORDER] =
ExecuteAA6(ProcessTime,ReleaseDate, DueDate, Algo)

    [noProbs, noJobs] = size(ProcessTime);
    EXECTIME = zeros(noProbs, 1);
    SUMTARD = zeros(noProbs, 1);
    ORDER = zeros(noProbs, noJobs);

    for  probs = 1: noProbs

        processTime = ProcessTime(probs,:);
        dueDate = DueDate(probs,:);
        releaseDate = ReleaseDate(probs,:);

        [execTime, sumtard, order] = Algo(processTime,releaseDate,
dueDate);

        EXECTIME(probs) = execTime;
       SUMTARD(probs) = sumtard;
        ORDER(probs,:) = order;

    end

    meanSUMTARD = mean(SUMTARD);
    meanExecTime = mean(EXECTIME);
```

APPENDIX AII: Code Listing : Pseudocode of the Run Problem for HeuII for SPTTP with release dates problems (CLASS I)

Code Written by : Saheed Akande

Language : MATrix LABoratory

Date : June 2016

Project: Ph.D Thesis - DEVELOPMENT OF HEURISTICS FOR SINGLE PROCESSOR SCHEDULING PROBLEMS WITH TARDINESS- RELATED PERFORMANCE MEASURES
Department : Industrial and Production Engineering

```
clear all;

load('problemG.mat','Jobs', 'ProcessTime', 'ReleaseDate', 'DueDate');

Algo = @AA6Instance;

[meanSUMTARD, meanExecTime, SUMTARD, ORDER] =
ExecuteAA6(ProcessTime,ReleaseDate,DueDate, Algo);
```

APPENDIX AII: Code Listing : Pseudocode of the scheduling problem generation for static problems (CLASS II)

Code Written by : Saheed Akande

Language : MATrix LABoratory

Date : June 2016

Project: Ph.D Thesis - DEVELOPMENT OF HEURISTICS FOR SINGLE PROCESSOR SCHEDULING PROBLEMS WITH TARDINESS- RELATED PERFORMANCE MEASURES
Department : Industrial and Production Engineering

```
 MinProcessTime = 1;
MaxProcessTime = 10;
ProcessTimeGen = 9;
nojobs = 200;
rowc = 50;
wtime = zeros(rowc, nojobs);
ProcessTime = zeros(rowc, nojobs);
ReleaseDate = zeros(rowc, nojobs);
MaxDueDate = zeros(rowc, nojobs);
MinDueDate = zeros(rowc, nojobs);
DueDateGen = zeros(rowc, nojobs);
DueDate = zeros(rowc, nojobs);
Jobs = zeros(rowc, nojobs);


for row = 1:rowc

    Jobs(row, :) = 1:nojobs;

    ProcessTime(row, :) = floor(ProcessTimeGen *rand([1, nojobs]))+1;

    MaxDueDate(row, :) = 4*ProcessTime(row, :);

    MinDueDate(row, :) = 1* ProcessTime(row, :);

    DueDateGen(row, :) = MaxDueDate(row, :) - MinDueDate(row, :);

    DueDate(row, :) = (floor(DueDateGen(row, :).*rand([1,nojobs]))+
1*MinDueDate(row, :));

end

save('problemR.mat', 'Jobs' ,'ProcessTime', 'ReleaseDate', 'DueDate')
```

APPENDIX A111: Tables of Approximation Ratio:

The approximation ratio Tables for the three problem classes solved and the problem sizes in each case were given in Tables B –M

Table B: The approximation ratio for the small-sized problem of Class 1

S/No	Problem Sizes	HEUI	HEUII	DMDD	SPT	BB
1	5 x 1	16.62	3.69	10.85	147.77	1
2	7x1	2.51	1.197	3.04	15.98	1
3	8x1	1.42	1.17	1.7	10.99	1
4	10 x 1	1.57	1.35	2.08	8.13	1

Table C: The approximation ratio for the medium-sized problems for Class 1

S/No	Problem Sizes	HEUI	HEUII	DMDD	SPT	BB
3	15x1	1.78	1.73	1.99	4.85	1
4	20 x 1	1.43	1.38	1.43	2.55	1
5	25x1	1.56	1.54	1.57	2.39	1

Table D: The approximation ratio for the large –sized problems (30 ≤ n ≤ 1000) for Class 1

S/No	Problem Sizes	HEUI	HEUII	DMDD	SPT
6	30x1	1.00	1.17	1.1	1.53
7	40x1	1.00	1.06	1.02	1.22
8	60x1	1.00	1.11	1.07	1.12
9	80 x1	1.00	1.1	1.06	1.05
10	100 x1	1.00	1.15	1.11	1.06
11	150 x1	1.00	1.17	1.13	1.03
12	200 x1	1.00	1.18	1.15	1.01
13	300 x1	1.00	1.19	1.16	1.01
14	400 x1	1.00	1.22	1.18	1
15	500 x1	1.00	1.22	1.18	1
16	1000 x1	1.00	1.24	1.2	1

Table E: The approximation ratio of execution time for Class 1 problem sizes 5 ≤ n ≤ 1000.

S/No	Problem Sizes	HEUI	HEUII	DMDD	SPT	BB
1	5 x 1	3.13	3.23	3.44	1.00	1476923
2	7x1	3.05	3.2	3.35	1.00	4810000
3	8x1	3.1	3.25	3.35	1.00	7800000
4	10 x 1	3.23	3.32	3.47	1.00	1.5E+07
5	15x1	3.38	3.43	3.82	1.00	2.2E+07
6	20 x 1	3.07	3.31	3.72	1.00	3.8E+07
7	25x1	2.76	2.83	3.52	1.00	
8	30x1	2.66	2.84	3.51	1.00	
9	40x1	2.71	2.71	5.09	1.00	
10	60x1	2.66	2.72	6.28	1.00	
11	80 x1	2.48	2.54	7.89	1.00	
12	100 x1	24.66	24.93	97.05	1.00	
13	150 x1	0.24	0.24	1.21	1.00	
14	200 x1	2.89	2.91	13.83	1.00	
15	300 x1	0.91	0.91	99.16	1.00	
16	400 x1	1.12	1.13	138.29	1.00	
17	500 x1	11.21	1.12	129.06	1.00	
18	1000 x1	9.99	10.1	148.26	1.00	

Table F: The approximation ratio for Class II problem sizes 5≤n≤ 10

S/N	Problem sizes	GAlg	HEUA	HEUB	HEUIII	HEUIV	BB
1	5x1	1.05	1.03	2.01	1.00	1.11	1.00
2	7x1	1.04	1.04	1.86	1.00	1.03	1.00
3	8x1	1.03	1.06	1.81	1.00	1.04	1.00
4	10x1	1.00	1.05	1.70	1.00	1.01	1.00

Table G: The approximation ratio for medium-sized problems 15≤n≤ 25 (Class II)

S/N	Problem sizes	GAlg	HEUA	HEUB	HEUIII	HEUIV	BB
5	15x1	1.00	1.07	1.70	1.00	1.00	1.00
6	20x1	1.00	1.07	1.66	1.00	1.00	1.00
7	25x1	1.00	1.07	1.69	1.00	1.00	1.00

Table H: The approximation ratio for large-sized problems (30≤n≤ 1000)

S/N	Problem sizes	GAlg	HEUA	HEUB	HEUIV	HEUIII
6	30x1	1.03	1.06	1.68	1.02	1.00
7	40x1	1.04	1.07	1.66	1.01	1.00
8	60x1	1.00	1.07	1.69	1.02	1.00
9	80 x1	1.10	1.07	1.69	1.02	1.00
10	100 x1	1.00	1.07	1.68	1.00	1.00
11	150 x1	1.00	1.07	1.68	1.2	1.00
12	200 x1	1.00	1.07	1.69	1.00	1.00
13	300 x1	1.21	1.07	1.68	1.04	1.00
14	400 x1	1.00	1.08	1.68	1.02	1.00
15	500 x1	1.03	1.07	1.68	1.00	1.00
16	1000 x1	1.00	1.08	1.70	1.03	1.00

Table I: The approximation ratio of execution time for Class 1I problem sizes

S/N	Problem Sizes	GAlg	HEUB	HEUA	Heu III	HeuIV	BB
1	5 x 1	5.54	1.25	1	2.46	3.33	3069444
2	7x1	5.29	1.19	1	2.57	4.29	4303029
3	8x1	5.14	1.15	1	2.5	4.4	4465903
4	10 x 1	5.13	1.12	1	2.52	4.12	4084847
5	15x1	4.71	1.03	1	2.41	5	5825811
6	20 x 1	4.89	1.02	1	2.5	6.17	7929298
7	25x1	5.71	0.99	1	2.76	6.59	9428474
8	30x1	6.8	1.1	1	3	6.91	
9	40x1	6	0.9	1	2.2	6.75	
10	60x1	4.87	1.2	1	3	4.89	
11	80 x1	4.4	0.95	1	4.5	6	
12	100 x1	4	0.92	1	4.41	6.2	
13	150 x1	5.17	1.23	1	5.6	6.22	
14	200 x1	4.8	1	1	3.9	6.93	
15	300 x1	4.42	0.89	1	3.53	6.95	
16	400 x1	4.62	1	1	3.72	6.38	
17	500 x1	4.7	1	1	3.95	6.44	
18	1000 x1	5.19	1	1	4.36	5.79	

Table J: The approximation ratio for small-sized problem (5≤n≤ 10) of Class III

Problem Size	GAlg	DMDD	HEUV	HEUVI	BB
5 x 1	2.15	1.77	1.47	1.39	1.00
7x1	2.23	1.46	1.34	1.2	1.00
8x1	2.38	1.54	1.47	1.24	1.00
10 x 1	2.55	1.61	1.37	1.22	1.00

Table K: The approximation ratio for Class III problem sizes 15≤n≤ 25

Problem sizes	GAlg	DMDD	HEUV	HEUVI	BB
15x1	3.21	1.88	1.59	1.52	1.00
20 x 1	2.38	1.44	1.34	1.28	1.00
25x1	2.34	1.44	1.36	1.36	1.00

Table L: The approximation ratio for large-sized problems (30≤n≤ 1000) for Class III.

Problem Sizes	GAlg	DMDD	HEUV	HEUVI
30x1	1.65	1.04	1.03	1.00
40x1	1.59	1.02	1.01	1.00
60x1	1.02	1.48	1.02	1.00
80 x1	1.43	1.01	1.01	1.00
100 x1	1.43	1.01	1.01	1.00
150 x1	1.4	1.01	1.01	1.00
200 x1	1.37	1.01	1.00	1.00
300 x1	1.36	1.01	1.00	1.00
400 x1	1.36	1.00	1.00	1.00
500 x1	1.36	1.00	1.00	1.00
1000 x1	1.36	1.00	1.00	1.00

Table M: The approximation ratio for Class III problem sizes 5≤n≤ 1000 (for efficiency)

Problem Sizes	GAlg	DMDD	HEUV	HEUVI	BB
5 x 1	30.48	3.29	1.00	1.14	192095
7x1	26.54	2.77	1.00	1.12	7400000
8x1	26.29	2.67	1.00	1.19	13596296
10 x 1	26.21	2.52	1.00	1.21	2.3E7
15x1	22.5	2.33	1.00	1.36	3.6E7
20 x 1	20.67	2.02	1.00	1.42	4.4E7
25x1	26.67	1.67	1.00	1.33	5.4E7
30x1	23.88	1.49	1.00	1.64	
40x1	27.56	1.86	1.00	9.42	
60x1	25.06	2.33	1.00	1.24	
80 x1	26.5	2.98	1.00	9.8	
100 x1	10.59	1.28	1.00	2.72	
150 x1	12.97	1.42	1.00	2.49	
200 x1	38.07	4.79	1.00	6.93	
300 x1	40.4	38.6	1.00	8.5	
400 x1	36.86	31.82	1.00	10.36	
500 x1	47.75	39.06	1.00	10.93	
1000 x1	43.18	38.24	1.00	10.67	