Scheduling manufacturing system with sequence dependent setup and maintenance

Hendra Teja Wirawan

2013

Hendra Teja Wirawan. (2013). Scheduling manufacturing system with sequence dependent setup and maintenance. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/64823>

<https://doi.org/10.32657/10356/64823>

121
3X
2/B/48

SCHEDULING MANUFACTURING SYSTEM WITH
SEQUENCE DEPENDENT SETUP AND MAINTENANCE

HENDRA TEJA WIRAWAN

(B. Eng., NTU)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

IN MANUFACTURING SYSTEMS & TECHNOLOGY (MST)

SINGAPORE-MIT ALLIANCE

NANYANG TECHNOLOGICAL UNIVERSITY

2013

DECLARATION

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Hendra Teja Wirawan

6 December 2013

ACKNOWLEDGEMENTS

This work could not have been done without contributions of many great people during my PhD study. I sincerely thank the following people for their kind help.

First and foremost, I would like to thank Assoc Prof Sivakumar Appa Iyer and Prof Stephen Graves, my research supervisors, for their invaluable suggestions, advices, and guidance; and their extensive support and patience throughout the progress of this research. Dr. John Tajan has also provided great assistance, particularly in the initial buildup of my research progress.

I also appreciate our industrial collaborators from STATS ChipPAC Pte Ltd, especially Mr. Douglas Chin, Mr. Cindy Palar, Mrs. Vivienne Tan, Mr. Long Chay Kang, Mr. Jason Toh, Mr. Robert De Quiroz, and Mr. Lim Ming Chew, for their help in facilitating all the discussions, observations, and data collections during my industrial attachment.

To all of my friends, especially my lab-mates Jeffry, Mohana, Tony, Saha, Roy, Rajeeb, Phuong, Sangsang, and Thu Ba, I thank you all for your great company, encouragement, and support. I would also like to thank Rong, Eveline, Ketan, Jason, and Andre for their supports and hospitality during my MIT visit.

My feelings of gratitude also go towards my loving parents and siblings with whom I owe more than words can describe. Finally yet importantly, I thank my wife, Ei Lene for her love and support throughout.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS i

TABLE OF CONTENTS..... iii

ABSTRACT..... x

SUMMARY xi

LIST OF FIGURES xiii

LIST OF TABLES..... xx

LIST OF SYMBOLS xxii

LIST OF ABBREVIATIONS xxiv

1. INTRODUCTION 1

1.1 BACKGROUND..... 1

1.2 RESEARCH PROBLEM AND MOTIVATION 3

1.3 RESEARCH OBJECTIVE..... 8

1.4 RESEARCH SCOPE 10

iii

1.5	RESEARCH CONTRIBUTION	13
1.6	ORGANIZATION OF REPORT	14
2.	LITERATURE REVIEW	16
2.1	TAXONOMY OF SCHEDULING LITERATURE	16
2.3	MAINTENANCE SCHEDULING LITERATURE	29
2.4	INTEGRATED SETUP AND MAINTENANCE SCHEDULING LITERATURE	37
2.5	SCHEDULING WITH TIME-DEPENDENT PROCESSING TIMES LITERATURE	39
2.6	DYNAMIC SETUP SCHEDULING LITERATURE	46
2.7	SUMMARY	51
3.	SETUP SCHEDULING WITH FIXED MAINTENANCE PERIOD AND STATIC JOB ARRIVAL.....	53
3.1	PROBLEM FORMULATION	54
3.2	PROBLEM COMPLEXITY	57
3.3	OPTIMAL SOLUTION: BRANCH AND BOUND ALGORITHM	60

- 3.3.1 Lower Bound 60
- 3.3.2 Branching Strategy..... 64
- 3.3.3 Branch and Bound Algorithm..... 64
- 3.3.4 Numerical Example of Branch and Bound Procedure 67
- 3.4 HEURISTIC SOLUTION: GRASP ALGORITHM..... 70
 - 3.4.1 Overview of GRASP meta-heuristics 70
 - 3.4.2 GRASP Algorithm..... 72
 - 3.4.3 Numerical Example of GRASP meta-heuristics..... 74
- 3.5 COMPUTATIONAL EXPERIMENT 78
 - 3.5.1 Design of Experiment 78
 - 3.5.2 Results Analysis..... 79
 - 3.5.2.1 Heuristics CPU time and delta percentage comparison..... 81
 - 3.5.2.2 CPU time of the optimal algorithm 84
 - 3.5.3 Lower Bound Effectiveness..... 86
 - 3.5.4 Simultaneous Setup and Maintenance 88

3.6	SUMMARY	89
4.	SETUP SCHEDULING WITH FLEXIBLE MAINTENANCE PERIOD AND STATIC JOB ARRIVAL	91
4.1	PROBLEM FORMULATION	92
4.2	QUALITY DETERIORATION MODEL	93
4.3	OPTIMAL SOLUTION: BRANCH AND BOUND ALGORITHM ..	101
4.3.1	Lower Bound	101
4.3.2	Branching Strategy	104
4.3.3	Branch and Bound Algorithm	105
4.3.4	Numerical Example of Branch and Bound Procedure	106
4.4	HEURISTIC SOLUTION	110
4.4.1	Overview	110
4.4.2	Heuristic Algorithms	111
4.4.2.1	Random Interchange Algorithm	111
4.4.2.2	Simulated Annealing Algorithm	112
4.4.2.3	GRASP Meta-heuristic	114

4.5	COMPUTATIONAL EXPERIMENT	115
4.5.1	Design of Experiment	115
4.5.2	Results Analysis.....	115
4.5.2.1	Heuristics delta percentage comparison	118
4.5.2.2	The Fixed vs. Flexible Maintenance Period	119
4.5.2.3	Early vs. Late Maintenance Period.....	122
4.5.2.4	Meta-heuristic performance: Fixed vs. Flexible Maintenance Period	125
4.6	EXAMPLE OF POSSIBLE SCENARIO	127
4.7	SUMMARY	130
5.	SETUP SCHEDULING PROBLEM WITH FIXED MAINTENANCE PERIOD AND DYNAMIC JOB ARRIVAL	132
5.1	PROBLEM FORMULATION	132
5.2	ROLLING HORIZON PROCEDURE.....	136
5.3	ROLLING HORIZON ALGORITHMS	139

5.3.1	Rolling Horizon procedure with Branch and Bound with limited job arrival information (RH-BBL).....	139
5.3.2	Rolling Horizon procedure with GRASP meta-heuristic (RH-GRASP) 146	
5.3.3	Shortest Completion Time (SCT)	147
5.3.4	Rolling Horizon procedure with Branch and Bound with complete job arrival information (RH-BBC)	151
5.4	COMPUTATIONAL EXPERIMENT	160
5.4.1	Design of Experiment	160
5.4.2	Results Analysis.....	162
5.5	ANALYSIS ON MINIMIZING AVERAGE OF THE JOB FLOW TIMES.....	165
5.5.1	Modified Lower bound	165
5.5.2	Computational Experiment	170
5.5.2.1	Design of Experiment.....	170
5.5.2.2	Results Analysis	171
5.6	SUMMARY	175

viii

6. INDUSTRIAL APPLICATION: CASE STUDY 177

6.1 PROPOSED POLICIES 177

6.2 CASE STUDY MODEL, DATA, AND EXPERIMENTATION..... 178

6.3 RESULTS AND DISCUSSION 181

6.3.1 Utilization Performance Comparison 181

6.3.2 WIP Average Performance Comparison..... 186

6.3.3 Effect of Forecast Window 188

6.4 SUMMARY 192

7. CONCLUSIONS AND RECOMMENDATIONS..... 193

7.1 SUMMARY OF THE RESEARCH 193

7.2 LIMITATION OF THE RESEARCH..... 197

7.3 DIRECTION FOR FUTURE RESEARCH 198

REFERENCES 200

ABSTRACT

Few researches in scheduling literature have considered setup and maintenance concurrently. This thesis analyzes sequence-dependent setup scheduling problem with non-resumable case of maintenance consideration to minimize makespan (maximizing utilization) on a single machine. Some parts of machine time are required for setup and maintenance activity, and in this study it is assumed that these two can be performed simultaneously. The analysis is divided into three parts, based on its job arrival and maintenance period assumption. Algorithms with dynamic job arrival assumption are tested on historical data of an actual semiconductor testing facility system. It is observed that the proposed algorithms are able to improve machine productive utilization by up to ~6% and average WIP by up to ~40%.

SUMMARY

In this research, a single machine scheduling problem with consideration of both sequence dependent setup and non-resumable case of maintenance activity is studied with specific assumption that setup and maintenance can be performed simultaneously on the machine. Minimizing makespan, which is equivalent with maximizing machine utilization, is the chosen objective function.

Based on the assumption of job arrival and maintenance period, this analysis is divided into three parts. First part of problem analysis dealt with an assumption of deterministic-static job arrival and fixed maintenance period. Branch & Bound technique is proposed to obtain an optimal solution of this strongly NP-Hard Problem. It is able to provide an optimal solution, but the computation time of this approach grows exponentially which affects its practicality for a problem of larger size. Hence, three meta-heuristics are also developed and proposed based on GRASP meta-heuristic. Due to its strong assumptions of static job arrival and fixed maintenance period, results of part 1 problem analysis does not provide an implementable solution to the scheduling problem; However, it provides a crucial basis for the subsequent part of the analysis which have more relaxed assumptions.

In the second part, fixed maintenance period assumption is relaxed by allowing maintenance period to be shifted earlier or later than machine's tool life. With flexible maintenance period, better makespan performance is obtained with the cost of certain yield loss. When the yield loss value is too high, delaying maintenance period to minimize makespan should be avoided. The cut-off value of yield loss can be calculated with the help of quality deterioration model which is developed in this analysis. Results of this Part 2 analysis are helpful in quantifying the cost and value of scheduling decision on whether or not maintenance activity is to be performed at a certain point in time, and this quantification results are used as the basis for managerial decision on the matter.

Dynamic version of the scheduling problem of minimizing makespan is discussed in Part 3 of problem analysis. Dynamic job arrival is modeled based on Rolling Horizon procedure, in which job arrival information for certain limited time (Forecast Window) is known. In Rolling Horizon procedure, scheduling decision is made periodically, once every time the job arrival information is obtained. Two implementable Rolling Horizon algorithms are proposed. Dynamic scheduling problem with objective function of minimizing the average of the job flow times (equivalent with WIP average) is also discussed in Part 3 of problem analysis. The Part 3 Algorithms are then tested on historical data of an actual semiconductor testing facility system. It is observed that the proposed algorithms are able to improve machine productive utilization by up to ~6% and average WIP by up to ~40%.

LIST OF FIGURES

Figure 1.1: Production, Setup, and Maintenance Correlation Chart..... 4

Figure 2.1: Scheduling analysis taxonomy 17

Figure 2.2: Setup scheduling problem classification by Allahverdi, et al. (1999) 23

Figure 2.3: Maintenance management literature classification by Garg &
Deshmukh (2006)..... 33

Figure 3.1: Problem Illustration..... 55

Figure 3.2: Part 1 Problem Formulation 55

Figure 3.3: Graphical representation of numerical example of Part 1 Problem ... 56

Figure 3.4: Graphical representation of optimal sequence 56

Figure 3.5: Problem Complexity..... 59

Figure 3.6: Completion time illustration..... 60

Figure 3.7: Branch & Bound Procedure Example 68

Figure 3.8: Optimal solution with Branch & Bound..... 75

Figure 3.9: GRASP Descending Heuristic Solution 75

Figure 3.10: GRASP Ascending Heuristic Solution..... 77

Figure 3.11: GRASP Random Heuristic Solution 77

Figure 3.12: Heuristics CPU time comparisons..... 81

Figure 3.13: Heuristics delta percentage comparisons 81

Figure 3.14: Average idle time comparison..... 83

Figure 3.15: Average setup time comparison 83

Figure 3.16: CPU time comparison between BB and GRASP-RH 85

Figure 3.17: CPU time and Number of Nodes across tool life T 86

Figure 3.18: Comparison of simultaneous vs. separate setup and maintenance... 89

Figure 4.1: Part 2 Problem Formulation 93

Figure 4.2: Additional time computation case 1 illustration..... 96

Figure 4.3: Additional time computation case 2 illustration..... 97

Figure 4.4: Fixed period assumption optimal solution 99

Figure 4.5: Flexible period assumption optimal solution 99

Figure 4.6: Deterioration Model 100

Figure 4.7: Feasible Solution Illustration..... 102

Figure 4.8: Solution with one maintenance period 108

Figure 4.9: Solution with two maintenance period 109

Figure 4.10: Heuristics delta percentage comparisons 118

Figure 4.11: Fixed maintenance period vs. Flexible maintenance period 121

Figure 4.12: Early vs. Late Maintenance Period..... 124

Figure 4.13: GRASP-RH Delta Percentage Comparison: Fixed vs. Flexible
Maintenance Period 126

Figure 4.14: Fixed GRASP-RH without idle time vs. Flexible GRASP-RH 126

Figure 4.15: Fixed GRASP-RH without idle time vs. Flexible GRASP-RH
without late maintenance 127

Figure 4.16: Optimal Solution with Fixed Maintenance Period Assumption..... 128

Figure 4.17: Optimal Solution with Flexible Maintenance Period Assumption. 129

Figure 5.1: Part 3 Problem Formulation 133

Figure 5.2: Example of job arrival assumption..... 135

Figure 5.3: Rolling Horizon Procedure..... 138

Figure 5.4: Job arrival of Rolling Horizon 1 (RH-BBL) 141

Figure 5.5: Static Optimal Sequencing of RH1 (RH-BBL)..... 142

Figure 5.6: Static Optimal Sequencing of RH1 – RH2 (RH-BBL) 143

Figure 5.7: Job arrival of Rolling Horizon 3 (RH-BBL) 143

Figure 5.8: Static Optimal Sequencing of RH1 – RH3 (RH-BBL) 144

Figure 5.9: Job arrival of Rolling Horizon 4 (RH-BBL) 144

Figure 5.10: Static Optimal Sequencing of RH1 – RH4 (RH-BBL) 145

Figure 5.11: RH-BBL Sequencing Solution 146

Figure 5.12: First decision point 148

Figure 5.13: First job sequencing decision 149

Figure 5.14: Second decision point..... 149

Figure 5.15: Second job sequencing decision..... 149

Figure 5.16: Third decision point..... 150

Figure 5.17: Third job sequencing decision..... 150

Figure 5.18: SCT Sequencing Solution 151

Figure 5.19: RH1 Job List..... 155

Figure 5.20: Optimal Solution of RH1 156

Figure 5.21: Optimal Solution of RH2 157

Figure 5.22: Optimal Solution of RH3 157

Figure 5.23: Optimal Solution of RH4 157

Figure 5.24: Optimal Solution of RH5 158

Figure 5.25: Optimal Solution of RH6 158

Figure 5.26: Optimal Solution of RH7 158

Figure 5.27: Optimal Solution of RH8 158

Figure 5.28: Optimal Solution of RH9 158

Figure 5.29: Optimal Solution of RH10 159

Figure 5.30: Optimal Solution of RH11 159

Figure 5.31: Optimal Solution of RH12 159

Figure 5.32: Optimal Solution of RH13 159

Figure 5.33: Optimal Solution of RH14 159

Figure 5.34: Delta Percentage Comparison (Makespan minimization)..... 163

Figure 5.35: Delta Percentage across Forecast Window on RH-BBL..... 164

Figure 5.36: Example of Branch & Bound Procedure with Modified Lower Bound
..... 168

Figure 5.37: Delta Percentage Comparison (average of the job flow times
minimization)..... 174

Figure 5.38: Delta Percentage across Forecast Window on MRH-BBL 175

Figure 6.1: IC Manufacturing Process Flow 179

Figure 6.2: Productive Utilization Comparison 183

Figure 6.3: Non-productive Utilization Comparison 183

Figure 6.4: Downtime Utilization Comparison 184

Figure 6.5: CPU time comparison between RH-BBL and RH-GRASP 185

Figure 6.6: WIP Average Comparison..... 187

Figure 6.7: Forecast window effect on RH-BBL Productive utilization 190

Figure 6.8: Forecast window effect on RH-GRASP Productive utilization 190

Figure 6.9: Forecast window effect on MRH-BBL WIP Average 191

Figure 6.10: Forecast window effect on MRH-GRASP WIP Average 191

LIST OF TABLES

Table 2.1: Research Problem Taxonomy 21

Table 3.1: List of the remaining time on respective block 76

Table 3.2: Part 1 Computational Experiment Results 78

Table 3.3: 2-way ANOVA results..... 84

Table 3.4: Comparison of LB and LB’ 88

Table 4.1: Part 2 Computational Experiment Results**Error! Bookmark not defined.**

Table 5.1: Part 3 Computational Experiment Results (Makespan minimization)..... 161

Table 5.2: Forecast Window Computational Experiment Results on RH-BBL 163

Table 5.3: Part 3 Computational Experiment Results (Average of The Job Flow Times
Minimization) 172

Table 5.4: Forecast Window Computational Experiment Results on MRH-BBL 174

Table 6.1: Experimental Design 180

Table 6.2: Utilization Performance Computational Experiment Result..... 182

Table 6.3: WIP Average Performance Computational Experiment Result 187

Table 6.4: Forecast Window Computational Experiment Result (Makespan minimization)
..... 189

Table 6.5: Forecast Window Computational Experiment Result (WIP average
minimization)..... 189

LIST OF SYMBOLS

n	Number of jobs
p_i	Processing time of job i
s_{ij}	Setup time, in which job i is processed just before job j
T	Tool life
m	Maintenance time
$C_{\max}$	Makespan
J_i	Job i
M	Maintenance activity
c_{ij}	Completion time, in which job i is processed just before job j
LB	Lower Bound
c_k	Column minima of column k of completion time matrix
s_k	Associated setup time for the column minima c_k

x_i	Additional time of job i
C_i	Flow time of job i
λ	Poisson arrival rate
α	Additional time rate

LIST OF ABBREVIATIONS

ANOVA	Analysis of Variance
BB	Optimal Solution Branch & Bound Algorithm
CBM	Condition Based Maintenance
CLSP	Capacitated Lot Sizing Problem
CMMS	Computerized Maintenance Management Systems
CSLP	Continuous Setup Lot Sizing problem
DLSP	Discrete Lot Sizing and Scheduling Problem
ECM	Effectiveness Centered Maintenance
EDD	Earliest Due Date Rule
ELSP	Economic Lot Scheduling Problem
EOQ	Economic Order Quantity
FW	Forecast Window
GLSP	General Lot Sizing Problem

GRASP	Greedy Randomized Adaptive Search Procedures
GRASP-AH	GRASP Ascending Heuristic
GRASP-DH	GRASP Descending Heuristic
GRASP-RH	GRASP Random Heuristic
LB	Lower Bound
LPT	Longest Processing Time First Rule
MILP	Mixed Integer Linear Programming
MRH-BBC	Rolling Horizon procedure with Branch & Bound with complete job arrival information with modified lower bound
MRH-BBL	Rolling Horizon procedure with Branch & Bound with limited job arrival information with modified lower bound
MRH-GRASP	Rolling Horizon procedure with GRASP meta-heuristic with modified objective function
NB	Next Best Rule
OEE	Overall Equipment Effectiveness
PLSP	Proportional Lot Sizing Scheduling Problem
PM	Preventive Maintenance

RBM	Risk Based Maintenance
RCM	Reliability Centered Maintenance
RF	Re-planning Frequency
RH	Rolling Horizon
RH-BBC	Rolling Horizon procedure with Branch & Bound with complete job arrival information
RH-BBL	Rolling Horizon procedure with Branch & Bound with limited job arrival information
RH-GRASP	Rolling Horizon procedure with GRASP meta-heuristic
SCT	Shortest Completion Time
PH	Planning Horizon
SMM	Strategic Maintenance Management
TPM	Total Productive Maintenance
TSP	Traveling Salesman Problem
ULS	Un-capacitated Lot Sizing problem

1. INTRODUCTION

1.1 BACKGROUND

In today's manufacturing industry, producing only one type of product is insufficient for catering to diverse and ever-changing customer demand. Combined with the fact that most manufacturing systems have limited machine capacity, it is inevitable that production management utilizes one machine to process more than one type of product. Hence, a portion of the overall machine capacity is required to perform setup activity. Setup activity includes removal of previous components, converting the machine, and/or fine-tuning the machine after it has been restarted.

Another scheduling aspect that is worth considering is the fact that a machine in any manufacturing system is unreliable, and it is impossible for the machine to be perpetually available. Machine unreliability can be caused by quality deterioration over time; hence maintenance has to be carried out within a certain time interval. This means that the machine is not available during that maintenance period. Maintenance activity is defined as a set of activities designed and performed in order to avoid possible breakdowns and makes up one of the deterministic causes of machine unavailability (Duffuaa, et al., 1998).

Two of the most common assumptions in scheduling literature are that setup activity is considered negligible, and that machines can run continuously without maintenance activity. However, these assumptions may not be accurate in actual industry scenarios, and while it simplifies the analysis, it negatively affects the quality of solutions for most of the applications.

This thesis analyzes a single machine sequence-dependent setup scheduling problem with maintenance consideration of non-resumable cases to minimize makespan. Makespan is closely related to machine utilization as utilization is the production rate per capacity and production rate is the number of jobs divided by the makespan (Hopp and Spearman, 1996). A specific assumption made in this study is that setup and maintenance activity can be performed simultaneously. In a semiconductor manufacturing system, chemical changes on the etching tools due to process changeover (setup activity) and chemical changes due to chemical life-time (maintenance activity) can be performed in a single changeover. Also, in any manufacturing system with a costly shut down period, changing machine components according to the job to be processed, e.g., robot arm switch (setup activity) and replacing old components with a new one, e.g., filter replacement (maintenance activity) can be performed in one shutdown period. There are many other examples of simultaneous setup and maintenance activity from actual industry applications, which imply that coordination between setup scheduling and maintenance scheduling is crucial.

In a manufacturing system with tight machine capacity, performing setup and maintenance simultaneously is always desirable to save certain amount of the precious machine time. Cost associated to this combined activity is mainly on additional manpower required to perform both activities simultaneously. Besides, a more thorough planning and scheduling coordination from the manufacturing system planner is also required. Developed solution in this study would provide an implementable decision making tool to help in doing the coordination among the machine activities.

Besides minimizing makespan (i.e. maximizing machine utilization), an analysis on minimizing the average of job flow times while assuming dynamic job arrival is also included in this study. The average the job flow times is closely related to the average WIP of a manufacturing system, and it is found to be a more robust measure for systems with dynamic job arrival.

1.2 RESEARCH PROBLEM AND MOTIVATION

The problem that is addressed in this research can be concisely stated as follows:

Consideration of production activity, setup activity, and maintenance activity of an unreliable single machine with sequence-dependent setup time, in order to determine which type of product should be processed next, when the change-over is to be executed, and when the maintenance needs to be performed on the

machine such that machine utilization can be maximized and/or average WIP can be minimized.

All three aspects being considered, i.e., production activity, setup activity, and maintenance activity are correlated to each other as shown in Figure 1.1.

Figure 1.1: Production, Setup, and Maintenance Correlation Chart

i. Production and Setup Activity

If setup activity is performed too often, the amount of machine capacity used for setup will be high; whereas restraining setup activity would induce a high cycle time on certain product types. Hence, production and setup activity is correlated in the context of a tradeoff between setup time and cycle time.

ii. Production and Maintenance Activity

If maintenance activity is performed too often, the amount of machine capacity used for maintenance will be high; whereas too much restraint on the maintenance activity would affect machine performance, which in turn, affects product yield. Hence, production and maintenance activity is

correlated in the context of a tradeoff between maintenance time and product yield.

iii. Setup and Maintenance Activity

In most cases, setup and maintenance activity can be performed simultaneously. As performing them simultaneously would free up some machine capacity, scheduling coordination between these two activities is necessary.

Solving the formulated problem is divided into three parts, as follows:

1. Problem Part 1: Deterministic and Static Job Arrival with Fixed Maintenance Period.

In this model, all jobs are assumed to be available and ready to be processed at the start time of scheduling. Periodic maintenance activity is scheduled at fixed intervals with no flexibility whatsoever.

2. Problem Part 2: Deterministic and Static Job Arrival with Flexible Maintenance Period.

This part uses the same assumption as Problem Part 1 on job arrival; however, maintenance activity is flexibly scheduled, and a tradeoff between machine time (capacity) and product yield exists.

3. Problem Part 3: Dynamic Job Arrival with Fixed Maintenance Period.

Dynamic job arrival is assumed. It is also assumed that future job arrival information for the next certain limited time horizon is known with accuracy.

Investigations into the significance of setup times can be found in various studies, and some of them have been listed in a review by Allahverdi, et al (1999):

- The significance of sequence-dependent setup time is apparent when a manufacturing system is operated at, or close to, its capacity (Wilbrecht and Prescott, 1969).
- Based on a survey of industrial managers, “It is observed that about three quarters of the managers reported at least some of their operations they schedule require sequence-dependent setup time while approximately 15% reported all operations requiring sequence-dependent setup times.” (Panwalkar, et al., 1973).
- In a cellular manufacturing shop, “It is determined that applications of both sequence-dependent setup procedures and group technology principles are able to increase output capacity.” (Flynn, 1987).
- Various factors in a manufacturing system have been examined by Krajewski, et al. (1987) and it was concluded that, in any production system, “simultaneous reduction of setup times and lot sizes is the most effective way to reduce inventory levels and improve customer service.”
- Wortman (1992) states that sequence-dependent setup time is an important aspect of effective manufacturing capacity management.

The importance of maintenance has been gradually recognized as it is a necessity to avoid machine breakdowns. Furthermore, in any operational budget, subsequently after costs of energy, costs of maintenance can be the biggest part

(Garg & Deshmukh, 2006). The increasing trend of maintenance significance is explained in a study by Dekker & Scarf (1998), with the following reasons:

- Continuous expansion of capital in various sectors leads to an increase in the needs for maintenance.
- The emergence of production philosophy such as Just in Time (JIT) increases the requirement for the reliable systems, which means random breakdown on the system has become less acceptable.
- Outsourcing trend for the maintenance activity, which forces manufacturing management to describe and schedule it precisely.

Both setup scheduling and maintenance scheduling have been well studied by the researchers. However, most existing research has studied each aspect independently; while the study of a system which considers the integration of setup and maintenance scheduling is considerably limited. There are only a few publications about single machine setup scheduling problems with a maintenance consideration to minimize makespan. Chen (2008) discusses this integrated problem, with an assumption of resumability while considering maintenance, which simplifies the problem since the inclusion of maintenance activities would not affect the job sequencing. Ángel-Bello, et al. (2011b) derived a mixed-integer programming model to achieve an optimal solution for the problem posed by non-resumable cases of integrated problem while Ángel-Bello, et al. (2011a) proposed meta-heuristic procedures to find a decent solution with short CPU times. All these previous studies do not provide implementable solutions as they have only

considered the problem while assuming deterministic and static job arrival, in which all jobs are assumed to be ready for processing at the time of scheduling. In addition, the analyses performed were limited to fixed and periodic maintenance activities, which is not always the case for most dynamic manufacturing systems. Last but not least, while setup and maintenance activity can be performed simultaneously in many real industrial systems, none of these previous studies take this into account.

1.3 RESEARCH OBJECTIVE

Setup and maintenance activity is especially significant in manufacturing system scheduling, especially in the area of machine utilization. The main aim of this research is to obtain a comprehensive understanding of setup and maintenance scheduling in a manufacturing system. Besides, this study also aim to produce an implementable solution to sequence multiple types of jobs on an unreliable bottlenecked resource in order to maximize its utilization and improve overall system performance. This can be achieved via the following objectives:

- i. To explore and understand the existing literature on manufacturing sequencing and scheduling involving setup and maintenance, and then to identify gaps between theoretical study and actual industry implementation.

- ii. To analyze setup scheduling problems in deterministic and static job arrival and an assumption of a fixed maintenance period, with the following sub-objectives:
 - (i). To study the complexity of solving it optimally to determine which facets of the problem contributes to its complexity.
 - (ii). To develop optimal and heuristic algorithms to solve the problem.
 - (iii). To compare performance among several heuristic algorithms and uncover the optimal solution experimentally.
 - (iv). To identify which factor will significantly affect algorithm performance in terms of solution quality and computation time.

Part 1 model algorithm performance is crucial, as it will be the basis for the subsequent models (Part 2 and Part 3). In Part 2, Part 1 model is extended to cater for system with flexible maintenance period, whereas in Part 3, Part 1 model is extended to cater for system with dynamic job arrival.

- iii. To analyze setup scheduling problems with deterministic and static job arrival and an assumption of a fixed maintenance period, with the following sub-objectives:
 - (i). To develop a quality-deterioration model that is useful in capturing the impact of shifting maintenance activity schedules on system performance.
 - (ii). To develop optimal and heuristic algorithms to solve the problem.

- (iii). To compare the performance of several heuristic algorithms and find an experimental optimal solution.
- iv. To compare fixed maintenance period analysis with an analysis based on an assumption of a flexible maintenance period in order to study the effect of the maintenance period flexibility on system performance.
- v. To analyze dynamic job arrival with a fixed maintenance period of setup scheduling problems with the following sub-objectives:
 - (i). To review and understand the Rolling Horizon approach, and assess its feasibility to solve a dynamic version of the problem.
 - (ii). To implement the optimal algorithm and several heuristic algorithms within the Rolling Horizon environment.
 - (iii). To compare performance among several heuristic algorithms and establish an experimental optimal solution.
 - (iv). To determine the effect of the length of future arrival information horizon (forecast window width) on the algorithm performance.
- vi. To validate and analyze the performance of the proposed algorithms for dynamic job arrival approach using industrial case study parameters.

1.4 RESEARCH SCOPE

A single machine production environment is assumed in this study. Previous studies on simultaneous consideration of production scheduling, sequence-

dependent setup scheduling, and maintenance scheduling are considerably limited. Hence, single machine analysis of the problem is still worth to consider. In addition, solutions obtained from single machine analysis can be implemented into a complex manufacturing system by decomposing the system into several single machine entities.

A semiconductor manufacturing system is the main focus of our analysis. It was chosen due to its highly dynamic characteristics, limited machine time, and has sufficient computing facilities for online decision making. Nevertheless, this study is relevant for any kind of manufacturing system with frequent setup activity and / or maintenance activity, which requires some kind of coordination of production, setup, and maintenance in order to optimize the utilization of a bottlenecked machine.

The tasks and decisions by production management are generally categorized into one of three levels: strategic planning, tactical planning, and operational control (Hitomi, 1996). The scheduling problem formulated in this study is focused at the lowest level of operational control, i.e., in deciding the sequence of activity to be performed on the machine in order to maximize machine utilization and/or minimize the average of the job flow times; whether to perform production activity, on which type of product, setup activity, or maintenance activity, while knowing that all these three activities are all necessary and correlate to each other.

The problems of setup activity model can be divided into batch and non-batch models (Allahverdi, et al., 2008). The batch model occurs when jobs are processed in batches, and a setup time precedes the processing of each batch. In this model, the jobs are divided into several families. A batch is a set of jobs from the same family, and batch arrangement is a part of the decision variables while families are known in advance. In the non-batch model, a setup time is incurred before each time a job is processed. Theoretically, this research can be classified as dealing with the non-batch / serial processing model; However in practice, the model used in this study can be applied in both environments. Non-batch model can also be regarded as a batch model, in which each family consists of a single job. Jobs within the same family in a batch model can also be viewed as a group of jobs with zero sequence-dependent setup time in a non-batch processing model.

Allahverdi, et al. (2008) also classified setup as anticipatory and non-anticipatory setup. Anticipatory setup can be started before the corresponding job is available for production, which is not the case for non-anticipatory setup. The anticipatory setup model is assumed in this study.

As machine utilization is the main concern in this analysis, setup cost is considered directly proportional to setup time; hence a schedule that has optimal setup time is also the one with optimal setup cost. However, there are also several cases in which the setup cost is more significant than setup time. These cases will not be covered in this study.

1.5 RESEARCH CONTRIBUTION

The original contributions of this research include:

1. Analysis of sequence-dependent setup scheduling problems, with maintenance consideration in which setup and maintenance activity are correlated as they can be performed simultaneously on the machine.
2. Complexity analysis of setup scheduling problem with a fixed maintenance consideration.
3. Development of Branch and Bound Algorithm with completion time based on a lower bound to attain optimal sequence of deterministic and static case of the formulated problem.
4. Development of various heuristic algorithms including Random Heuristic based on existing GRASP (Greedy Randomized Adaptive Search Procedures) meta-heuristic, Rolling Horizon procedure, etc.
5. Development of quality-deterioration model to translate quality deterioration in the system into processing time requirement of certain jobs.
6. Analysis of the impact of early and / or late maintenance on manufacturing system performance.
7. Implementation of Rolling Horizon procedure in solving dynamic case of the formulated problem, which allows the use of heuristic algorithms to

solve actual industrial cases of a single machine with future job arrival information for certain limited time horizon, is known.

8. Identification of critical parameters, which affect manufacturing system performance in the context of maximizing machine utilization and/or minimizing average WIP.

1.6 ORGANIZATION OF REPORT

The organization for the rest of this thesis is as follows:

Chapter 2 provides a literature review on scheduling methodologies involving setup and/or maintenance activity. Chapter 3 is an analysis of the first part of the problem, a case involving deterministic and static job arrival with a fixed period of maintenance. The problem complexity is analyzed, and an optimal branch and bound is proposed together with a GRASP Random Heuristic. Several other meta-heuristics are also introduced and compared. Chapter 4 analyzes a deterministic and static case of the formulated problem, with a flexible maintenance period. A modified version of the branch and bound algorithm is proposed and tested with several classic heuristics combined with a pre-processing heuristic developed to estimate the number of maintenance activities within a planning horizon. An analysis of a case involving a flexible maintenance period is performed with the help of a quality-deterioration model. Chapter 5 assumes a case of dynamic job arrival, where future job arrival information for a certain limited time horizon is

known in advance. In Chapter 6, the implementation of a developed algorithm from previous chapters is illustrated with a case study based on actual industry parameters. Chapter 7 concludes the thesis and suggests directions for future research.

2. LITERATURE REVIEW

While both setup and maintenance scheduling have been analyzed by many published studies, most of them treat each of the two aspects separately, as two independent systems. From the limited studies that combine both setup and maintenance scheduling, the solutions given are hard to implement, due to the strong assumptions that the analyses began with.

This chapter reviews the existing research on methodologies and policies that consider setup and / or maintenance scheduling in manufacturing systems. The papers reviewed in this chapter are not exhaustive but are sufficiently comprehensive for a critique of the existing knowledge to identify unresolved issues and gaps, in addition to highlighting the potential contributions of this research.

2.1 TAXONOMY OF SCHEDULING LITERATURE

There is an extensive body of research on manufacturing system scheduling, which can be broadly classified, as shown in Figure 2.1.

Figure 2.1: Scheduling analysis taxonomy

Based on the number of machines, scheduling analysis can be generally divided into *single machine analysis* and *multiple machine analysis*. Single machine models are essential not only for providing insight into the single machine environment, but also provide a foundation for a heuristic that is applicable to more complex manufacturing systems (Pinedo, 2008). Besides, a complex manufacturing system can be decomposed into several sub-problems that deal with a single machine system. As for multiple machine analysis, it can be considered as a generalization of single machine analysis. Many variations of multiple machine analysis can be found in literature, such as analyses involving parallel machines, flow shops, job shops, and open shops; in which each of them has its own further variations, e.g., parallel machines can be further classified as

identical parallel machines, uniform parallel machines, and unrelated parallel machines.

The scheduling research can also be classified as either *deterministic* or *stochastic* (French, 1982). Deterministic models assume that job arrival information is known with certainty. With a deterministic assumption, the assumption and solution algorithm used on the system determine the results. Conversely, stochastic models introduce the element of randomness into the system at one or many points of the model, where job arrival information is only known in terms of their distribution. While a stochastic model is a more realistic representation of the problem than a deterministic one, it is difficult to perform sensitivity analysis on a system with randomness. Hence the deterministic model is essential for understanding problematic structures, whereas the stochastic model helps to gain some realism when it is necessary (Starfield, 2005).

Job arrival can be assumed to be either *static* or *dynamic*. In the static environment, there are a predetermined number of jobs to be scheduled, and no further jobs are considered until the following scheduling instance, or until all jobs considered in the planning horizon are completed. In a dynamic case, jobs can arrive in a random manner during the planning horizon.

The setup activity has been previously thought of as insignificant and thus disregarded in most of scheduling literature. Even though this may be a reasonable assumption for certain scheduling problem applications, there are

many other cases in which setup consideration is fundamental in analyzing the scheduling problem. Setup can be broadly categorized as *sequence-independent setup* and *sequence-dependent setup* (Allahverdi, et al., 1999). This categorization is based on whether the setup time is dependent only on the job to be processed (sequence independent) or whether it is also dependent on the prior job (sequence dependent). An example of sequence independent setup application can be seen in a computer system application, in which different compilers are required for computer jobs (Bruno and Downey, 1978; B. Chen, 1993; Monma and Potts, 1989); No setup is induced if the subsequent job to be processed needs a compiler that is already resident in the memory. On the other hand, if the subsequent job needs a compiler that is not resident in the memory, a setup time that is only dependent on the time required for loading the new compiler is required. As for sequence-dependent setup applications, Conway, et al. (2003) describes printing industry applications where the setup activity in terms of machine cleaning activity depends on the color of the current and immediately subsequent jobs.

While most of the scheduling literature assumes that machines are continuously available for processing without the need to perform maintenance activity, this assumption may not be true in the actual industry setting. For those with maintenance considerations, scheduling analysis can be classified as *single maintenance activity*, when only one maintenance activity is allowed during the scheduling period, and *multiple maintenance activity*, when more than one maintenance activity is allowed.

Another method of classifying maintenance consideration is in terms of having a *fixed maintenance period* and a *flexible maintenance period*. One is the case when maintenance periods are fixed beforehand; the other is when the maintenance period is a part of decision making process, i.e., “scheduling processing of jobs and maintenance of a machine simultaneously.” (Ma, et al., 2010).

Preemption of jobs while processing may occur due to maintenance considerations in the system. There are two kinds of processing scenarios for scheduling problems with maintenance considerations, i.e., resumable case, in which a job that cannot be finished prior to the maintenance period can resume after the machine is available; and non-resumable case, in which the job has to be restarted rather than resumed, which implies that job cannot be stopped for maintenance, and maintenance has to be delayed till the end of job processing.

With regard to the formulated problem being analyzed in this study, Table 2.1 describes the position of our study on the scheduling analysis taxonomy as shown in Figure 2.1.

	Part 1 Problem	Part 2 Problem	Part 3 Problem
Number of machine	Single	Single	Single
Job arrival model	Deterministic	Deterministic	Deterministic
Job arrival assumption	Static	Static	Dynamic
Setup time assumption	Sequence dependent	Sequence dependent	Sequence dependent
Maintenance activity	Multiple	Multiple	Multiple
Maintenance period	Fixed	Flexible	Fixed
Preemption assumption	Non-resumable	Non-resumable	Non-resumable

Table 2.1: Research Problem Taxonomy

2.2 SETUP SCHEDULING LITERATURE

In this section, previous studies on setup scheduling problem without maintenance considerations are reviewed. Study in scheduling problems with setup consideration can be found in literature since the mid-1960s, with papers by Gilmore and Gomory (1964), Gavett (1965), Presby and Wolfson (1967), and Glassey (1968). In 1990s, it was observed that there was a tremendous growth of attention in solving the setup scheduling problem, whereby “an average of more than 40 papers per year have been added to the literature.” (Allahverdi, et al., 2008). This increase is primarily caused by the fact that a significant amount of savings can be obtained by carefully considering setup activity in manufacturing scheduling. Allahverdi, et al. (2008) has also listed out some of the many recent applications of setup scheduling implementation in various real industrial settings:

- “In a certain facility that produces toner for photocopiers and laser printers, it is observed that changing production from one toner to another, results in large setup times, which are generally in the order of days.” (Laguna, 1999).
- For the printed circuit boards manufacturing system with automated insertion machine, Schaller, et al.(2000) stated that one of its main concerns is setup scheduling problem.
- “In the textile industry, fabric types are assigned to looms equipped with wrap-chains. When the fabric type is changed on a machine, the wrap-chain must be replaced, and the time it takes depends on the current and previous fabric types.” (Gendreau, et al., 2001). Hence, sequence-dependent setup times are significant to the system
- In the semiconductor industry, Kim and Shin (2003) pointed out that “the machines used in the production should be adjusted whenever different types of wafers are processed; therefore different setup times are required depending on wafer sequences.”
- In an auto-making facility, the setup activity for the stamping plants involves the changing of heavy dies (Yi and Wang, 2003).
- In a label sticker label manufacturing company, “The first stage is a single high speed machine that is used to glue the surface material and liner together to produce label stickers. When the machine is changed over from jobs in one class to another one, a sequence-dependent setup time is required.” (H. T. Lin and Liao, 2003).

- In the tile industry, “The objective of the product grouping problem is to minimize the setup time in order to reduce the production time.” (Andres, et al., 2005).

Allahverdi, et al. (1999) adopted a basic classification of setup scheduling literature in their survey paper, by dividing it based on batch availability and sequence dependency. Figure 2.2 describes the position of our study on the mentioned classification, in which emphasis is given to the non-batch setup problem with sequence-dependent setup consideration.

Figure 2.2: Setup scheduling problem classification by Allahverdi, et al. (1999)

A tremendous amount of setup scheduling analysis is available in literature; however in this review section, focus is given to a single machine setup scheduling analysis to minimize makespan, which is equivalent with maximizing utilization as shown in section 1.1.

In setup scheduling analysis, a machine’s time is allocated only for production activity and setup activity. Hence, the problems of maximizing utilization are

equivalent with those of minimizing total setup time. In addition, since the problem of minimizing setup cost exists because the time has direct proportionality and a similar solution methodology with the problem of minimizing setup time, thus it will also be included in this review section.

Gilmore and Gomory (1964) considers a sequencing problem in a machine with single real variable that describes its state. A certain cost is incurred in changing the machine from one state to another, with specific cost assumptions which suit furnace machine applications. The temperature of the furnace is the stated variable, and various jobs are queued to be processed through a temperature cycle inside the machine. A certain job will be started at temperature (state) A and taken out of the furnace at temperature (state) B. The temperature of the furnace then needs to be changed for the next job, and cost function $f(x)$ is used to describe the cost of heating the furnace while another cost function $g(x)$ is for the cost of cooling it. Hence, in order to minimize the total setup cost, there is no advantage to be obtained by cycling the state variable, i.e., by increasing it and then decreasing it to its original state. This problem is solved as a special case of Traveling Salesman Problem (TSP) with an assumption that initial and final state of the system is known in order to transform it from a sequencing problem into a tour problem. A basic algorithm which includes permutation search and sequence interchange to find the minimum spanning tree is proposed to solve this specific problem.

The first study with an objective function of minimizing total setup time over a finite batch of jobs can be found on a journal paper by Gavett (1965). It deals with a single production facility with a set of non-identical jobs, in which the shutdown period as preparation for the subsequent job in sequence is regarded as the setup time. Setup time is determined by the substantial closeness between the jobs, e.g. "In the case of a printing press, the setup time between two jobs will depend on the similarity of their requirements for ink color, paper dimensions, and some other factors which demand press adjustments." Three basic heuristic rules are tested, and then compared with random sequencing and optimal solutions to assess their efficiency. The optimal solution is obtained by means of the TSP branch and bound algorithm with a reduced cost matrix previously developed by Little, et al. (1963) as the lower bound. The three heuristics, i.e., Next Best Rule (NB), in which an unassigned job that has the least setup time relative to the job which has just been completed is chosen; The Next Best Rule with Variable Origin (NB') applies the NB rule following the random selection of the initial job which goes after the job which has just been completed, and The Next Best Rule after Column Deduction (NB'') applies the NB rule following subtraction of the smallest value in each column of the matrix of setup time from all other values in the column. From computational experiments with the number of jobs up to $N = 20$, the delta percentage of three heuristics compared with the optimal solution are found to be relatively large, i.e., averaging of 57%, 34%, and 25% respectively. Certain practical problems of the approach were pointed out, i.e., the problem of obtaining a standard matrix of the setup time for computational purposes and also

the trouble of having access to computing amenities for making an online decision, which is not always feasible for every manufacturing environment.

The effectiveness of the three heuristics above is further analyzed in a study by Haynes, et al. (1973). The effectiveness was evaluated with respect to job size and setup time distribution. The number of jobs considered in this analysis is $N = 5, 7, 9$, and 11 due to computation time constraints; whereas normal, uniform, and gamma distribution were tested in the setup time parameter. Analysis was performed with three-factor Analysis of Variance (ANOVA). It is observed that the delta average of heuristics worsened significantly as the number of jobs increases from 5 to 11. The significance of setup time distribution is not as weighty because little distinction is evident, and this only occurs when 90% level of significance is accepted. For this criterion, the greatest delta is observed when setup times followed a uniform distribution.

A similar problem was solved by Presby and Wolfson (1967) using a staggered enumeration optimal algorithm. The algorithm starts with four-job sequences and constructs five-job sequences from these. From the five-job sequences, six-job sequences are constructed, and so on. At each stage, several non-optimal sequences are eliminated to make computations more feasible. With this staggered approach, the advantage is that it is possible to make occasional modifications in case there is any rush order during the process. Nevertheless, the enumeration process proposed is quite computation-intensive and only suitable

for a relatively small number of jobs queuing to be sequenced (2.5 – 3.0 minutes to solve a problem with $N=12$).

Glassey (1968) studied the changeover scheduling for n products on a single machine with a constant changeover loss for all products. A state space in vector form is defined in which a product changeover corresponds to a change in the direction of the path in the state space. Dynamic programming is formulated in order to determine the shortest path to minimize the changeover costs while meeting each job's respective due date. This approach was found to require a long enumeration time to find the solution. Hence, Mitsumori (1972) proposed a more efficient branch and bound algorithm than the earlier approach, in which a method of partitioning the set of feasible schedules to follow Glassey's method and an equation to calculate the lower bound of the objective function is presented. He also extended the earlier study by including the case in which the changeover loss varies depending on the subsequent product.

A variant of the setup scheduling problem was analyzed by Laguna, et al. (1991) by adding in a linear delay penalty into the objective function. The tabu search procedure was utilized to solve sequence-dependent setup scheduling problems, to minimize total setup cost and linear delay penalties. Three kinds of tabu search procedures are analyzed, i.e., tabu search with swap moves, tabu search with insertion moves, and tabu search hybrid procedure. A near-optimal solution was obtained for most cases. Laguna and Glover (1993) further improved this procedure by integrating target analysis. "Target analysis is meant to give certain

heuristic procedures the ability to *learn* what rules are best to solve a particular class of problems by means of hindsight analysis and pattern recognition techniques.” (Glover and Greenberg, 1989). Integrating this technique into tabu search procedures makes it possible to improve the heuristic performance both in terms of quality solutions and computation time.

The same problem above is then solved by different means, i.e., Greedy Randomized Adaptive Search Procedure (GRASP) methodology in the study by Feo, et al. (1996). GRASP is iterative meta-heuristics, consisting of a construction phase to build initial feasible solutions and a local search phase to find the local minimum. It is shown that GRASP is a powerful approach for solving combinatorial optimization problems. It is superior to previous tabu search procedures with respect to the solution values obtained and the computation time required for reaching the solution.

Ozgun and Brown (1995) developed a two-stage heuristic procedure for a single machine sequence-dependent setup scheduling problem with a symmetric setup time matrix. With a symmetric setup time matrix, the setup time required to do a changeover from a job i to job j is the same as the time required to do it from job j to job i . In the first stage of heuristic procedure, all the jobs are classified into groups using cluster analysis procedure, in which those jobs with relatively short setup time among each other are grouped as one. Afterwards, TSP is then solved with a classic TSP heuristic, the Lin and Kernighan heuristic. The Lin and Kernighan heuristic is a highly effective heuristic procedure to provide near-

optimal solutions for symmetric TSP, with a 0.46 average frequency of the optimum solution for problem size $N=30$ (S. Lin and Kernighan, 1973). As for the second stage, a particular purpose traveling salesman heuristic was developed to combine the already-sequenced groups. This particularly purposed heuristic begins with the formation of the setup matrix between groups and is followed by the application of a branch and bound algorithm. Computational experiment is only performed to compare the two-stage heuristic with a pure Lin and Kernighan heuristic, with no comparison performed with respect to the optimal solution. The two-stage heuristic was found to have only a 0.3% worse solution with 53.5% shorter computation time than the Lin and Kernighan heuristic. However, both of these heuristics are only applicable for the symmetric cases and no obvious comparisons towards an optimal solution can be found in the existing literature.

2.3 MAINTENANCE SCHEDULING LITERATURE

In this section, past research on maintenance scheduling problems without setup considerations is reviewed. Maintenance optimization generally consists of mathematical models intended to result either in an optimal equilibrium between costs and benefits of maintenance or the most suitable time to perform maintenance (Dekker and Scarf, 1998). It is a well-established area as suggested in the studies by Sherif (1982) and Valdez-Flores and Feldman (1989).

The importance of maintenance management tends to grow in all sectors of society due to various reasons (Dekker and Scarf, 1998).

- The continuous expansion of capital inventory includes machines being utilized in manufacturing systems, which require regular maintenance so as to remain suitable for use in processing jobs.
- The increase in requirements for the systems functioning (e.g., the just-in-time production philosophy), which require manufacturing systems with a minimized level of random breakdowns.
- The increasing trend of maintenance outsourcing, which lays stricter requirements on maintenance management, as most of the maintenance activity has to be explained and planned accurately and accordingly.

A comprehensive review paper by Garg and Deshmukh (2006) classified maintenance management literature into six parts, i.e., “maintenance optimization models, maintenance techniques, maintenance scheduling, maintenance performance measurements, maintenance information systems, and maintenance policies.” Four of them are further discussed as follows:

- *Maintenance optimization models*, which cover several aspects, including a depiction of a manufacturing system, function, and its significance; a deterioration model of the system in time and potential cost for the system; a description of the existing information system and available measures for management; and an objective function and an

optimization method, which help in finding the optimum equilibrium. Maintenance optimization models can be both qualitative (e.g., Total Productive Maintenance (TPM), Reliability Centered Maintenance (RCM), etc) and quantitative (e.g., various deterministic / stochastic models like Bayesian approach, Mixed Integer Linear Programming (MILP), etc).

- *Maintenance techniques*, which consist of the following:
 - Total Productive Maintenance (TPM), proactive and holistic way of solving maintenance problem, which essentially aims to identify issues as soon as possible and aims to prevent any issues before occurrence,
 - Computerized Maintenance Management Systems (CMMS), the use of a computer database, which provides the capabilities to keep, retrieve, and analyze information,
 - Reliability Centered Maintenance (RCM), which directs resources of maintenance at those components where reliability is crucial,
 - Maintenance outsourcing,
 - “Effectiveness Centered Maintenance (ECM), which is composed of people participation, quality improvement, maintenance strategy development, and performance measurement.” It is deemed more effective than the TPM or RCM approach.

- Strategic Maintenance Management (SMM), quantification models that incorporate technical, commercial, and operational characteristics from a business perspective.
- Risk Based Maintenance (RBM), makes certain a solid strategy for maintenance in meeting the dual objectives of reducing compromised product quality and a cost effective strategy.
- *Maintenance scheduling*, which includes several techniques, as follows:
 - Preventive Maintenance (PM) is maintenance that is performed periodically at a frequency determined by the course of time or the production's quantity being processed on the machine,
 - Condition Based Maintenance (CBM) is maintenance that is based on some measurement reading from the tool,
 - Predictive Maintenance, which consists of deciding whether or not to maintain a system according to its state.
- *Maintenance Performance Measurement*, which includes several aspects, as follows:
 - Overall Equipment Effectiveness (OEE), which evaluates and indicates how effectively a manufacturing system is utilized,
 - Performance measurement relationship with maintenance strategy,
 - Effect of maintenance-induced failures on operational effectiveness.

Figure 2.3: Maintenance management literature classification by Garg & Deshmukh (2006)

Figure 2.3 describes the position of this research study on the maintenance literature classification by Garg and Deshmukh (2006), with the following explanations:

- Only the quantitative model based on various optimization tools is to be considered in our analysis.
- The essence of the results of this study is useful for Strategic Maintenance Management (SMM), specifically helping manufacturing systems to sequence jobs to maximize utilization with maintenance activity considerations; While Problem Part 2, which considers flexible maintenance activity with trade-offs between machine capacity and product yield, is closely related with the Risk Based Maintenance (RBM) approach.

- The result of this study is mainly suitable for manufacturing systems with the Preventive Maintenance (PM) approach.
- Minimizing makespan is equivalent with maximizing machine utilization (OEE) and Part 2 of this analysis studies how maintenance-induced yield loss affects the operational effectiveness of the manufacturing system.

As mentioned above, a tremendous amount of maintenance scheduling analysis is available in literature; however in this review section, focus is to be given to single machine scheduling analysis to minimize makespan.

For makespan minimization problem with preemption allowed (resumable assumption), it is clear that every solution is optimal, in which it starts at time zero, and the machine never stays idle while some task is waiting to be processed (Schmidt, 2000). However, preemption is only of use when some job cannot be completed prior to the maintenance period.

As for the problem with no preemption allowed (non-resumable assumption), (Lee, 1996) shows that the problem with single maintenance period is NP Hard, by transforming the Partition Problem to the formulated problem. When multiple maintenance periods are allowed, it is proved to be NP Hard in the strong sense by transformation of the three-partition problem. It is also shown that the LPT heuristic, i.e., assigning the machine as many jobs as possible before the period of non-availability and then assigning the remaining jobs after the period in arbitrary order, has a tight bound of $1/3$.

Kacem (2009) considers the single machine scheduling problem with single fixed maintenance period, which includes a positive tail for every job. The consideration of a positive tail is motivated by those scheduling problems such that jobs have delivery times after their processing. A polynomial approximation algorithm with the worst case performance ratio of $3/2$ was proposed, and it was shown that the bound is tight.

Another variance of a single machine scheduling problem with single maintenance period was studied by Yang, et al. (2002). A single maintenance period $[s, t]$ was arranged previously; with s (t) being the earliest (latest) time at which the machine starts (stops) its maintenance. The maintenance time is constant r in which $r \leq t-s$. The problem is proved to be NP-Hard with Partition problem transformation, and some solvable cases are shown, i.e., if $\min_{1 \leq i \leq n} \{p_i\} > t - r$, in which jobs assigned in arbitrary order after $s + r$ is an optimal schedule; and if $\sum_{1 \leq i \leq n} p_i \leq t - r$, in which jobs assigned in arbitrary order before $t-r$ is an optimal schedule. A heuristic algorithm is also proposed with an average error percentage of 0.406%. This study was extended by J. S. Chen (2008) by considering multiple and periodic maintenance periods. Two optimal models based on Binary Integer Programming (BIP) were provided and able to solve up to 100 job instances. Additionally, an efficient heuristic is proposed for finding the near optimal solution for large sized problems, with the average percentage error below 1%. In a separate study, Xu, et al. (2009) showed that the worst case performance of Chen's heuristic algorithm is 2. They also showed that there is no

polynomial time approximation algorithm with a worst case performance of less than 2 unless $P = NP$.

The problem of fixed multiple maintenance periods was analyzed by Ji, et al. (2007), in which each maintenance activity is scheduled after a periodic time interval. It was shown that the worst case ratio of the classical LPT algorithm is 2 and that 2 is the best possible for all the polynomial time approximation algorithms unless $P = NP$. This result is useful as a benchmark for future research, in which it is only worth designing an approximation algorithm with a lower time complexity than the LPT algorithm while maintaining the worst case ratio of 2.

Low, et al. (2010) dealt with two maintenance stratagems simultaneously, that a machine should cease for maintenance after a periodic time interval or change tools after a fixed amount of jobs processed. A maintenance activity must be completed within a designated time window, which is assigned to each maintenance period. The problem was shown to be NP Hard in the strong sense. Six heuristic approaches were tested with no optimal solution proposed; hence performance comparison was only performed among the six heuristics. Further performance evaluation was done by designing an asymptotic worst case performance ratio. The optimal solution for the same problem was provided by Hsu, et al. (2010) with two-stage binary integer programming, which was able to solve the problem for up to 350 job instances. They also provided two heuristics with a total average percentage error below 1%.

2.4 INTEGRATED SETUP AND MAINTENANCE SCHEDULING LITERATURE

Single machine analyses minimizing makespan for both setup scheduling and maintenance scheduling have been well studied, as described in the last two sections. However, most of these studies consider each aspect independently; while studies involving a system which considers both setup and maintenance scheduling in an integrated system are considerably limited. There are only a few previous studies which consider the single machine setup scheduling problem with maintenance consideration to minimize makespan, and the earliest study found is from 2008. This shows that the study of this specific problem is still at an early stage of development with many aspects which still need to be further analyzed.

W. J. Chen (2008) discusses a single machine scheduling problem with multiple and periodic machine maintenance and sequence-dependent setup time to minimize the total setup time, which is equivalent with minimizing makespan. It is assumed that if a job is stopped for maintenance activity, the work performed on the job is not lost; hence the operation can be resumed once the machine is available (resumable assumption). It is also assumed that the split job will be resumed without setup if it is already installed, with no change in processing time for the split job. A heuristic procedure which consists of the “initialization procedure, Stinson heuristic procedure, and the iterative procedure” is proposed,

and its performance is compared with the optimal solution, which is obtained by a branch and bound algorithm. A near-optimal solution can be produced with the heuristic procedure. However, it is necessary to take note that having a resumable assumption in the analysis simplifies the problem tremendously, as the inclusion of maintenance activities has no effect on job sequencing. The same problem with a similar solution methodology is discussed by W. J. Chen (2009), by including a case study of machine maintenance and sequence-dependent setup consideration in a textile company.

The study of the non-resumable version of the above problem was initiated by Ángel-Bello, et al. (2011b). A mixed integer linear model was developed to solve the problem optimally, and in order to improve the computation time performance the author proposed two methods. By adding valid inequality to the model to strengthen the linear relaxation, and a heuristic procedure, a starting feasible solution is provided to the solver. Based on their computational experiment, the optimal model was able to solve up to 12 job instances. As for larger instances, (Ángel-Bello, et al. (2011a) proposed a meta-heuristic based on existing GRASP procedure.

The work in this dissertation is also trying to analyze manufacturing systems with the consideration of both sequence-dependent setup and multiple maintenance consideration under a non-resumable assumption. While previous work by Ángel-Bello, et al. (2011a) and Ángel-Bello, et al. (2011b) assume that setup activity and maintenance activity are two separate entities which are independent of each

other, our work here assumes that setup and maintenance activity are correlated to each other. The correlation exists such that assumption setup and maintenance activity can be performed simultaneously, which intensifies the need for scheduling coordination between these two activities. This assumption can be easily justified by many actual industry cases from various industries, which perform setup and maintenance activity simultaneously at a certain point of time.

2.5 SCHEDULING WITH TIME-DEPENDENT PROCESSING TIMES LITERATURE

Our Problem Part 2 deals with setup scheduling problems with a flexible maintenance period, which allows manufacturing systems to perform maintenance activities earlier or later than its usual period. When maintenance is performed later than usual, it is assumed that a machine's quality would deteriorate, which causes a slower rate of processing. Hence, the subsequent job to be processed on the machine would have a higher processing time than it is supposed to have. In literature, a similar kind of problem is also referred to as a scheduling problem with time-dependent processing times.

Scheduling with time dependent processing time has many applications, as mentioned in a review paper by Alidaee and Womer (1999):

- In the scheduling of steel rolling mills, while the ingot is waiting in a buffer between the furnace and rolling mills, when the temperature of an ingot has dropped below a certain temperature, and the ingot then needs to be reheated which requires additional processing time.
- In a military situation, the search for an object requires a longer time because of a worsening weather or growing darkness.
- In fire-fighting, more time is required to control the fire if there is a delay in starting the firefighting effort.

In all those cases above, any delay in tackling a task may result in an increased processing time.

In this section, previous research on scheduling problems with time-dependent processing time is discussed. While previous studies on this matter do not specifically consider setup and maintenance scheduling, the analysis findings and methodology in handling the time-dependent processing of a scheduling problem are valuable in solving the Problem Part 2.

Gupta and Gupta (1988) considers the makespan minimization problem of a static single facility problem with a non-resumable assumption, and the processing time of each job depends on its starting (or waiting) time in the sequence. The processing time function can be assumed to be linear, quadratic, or beyond. The general case of the problem has been proven to be an NP Complete problem by Garey and Johnson (1979). For problems with linear processing time function

$p'_i = a_i + b_i t_i$, it is proved that the schedule obtained by arranging jobs in a non-decreasing order of a_i/b_i value is optimal. It is difficult to establish an efficient rule to achieve the optimal schedule for more general functions, such as the quadratic function and beyond. The branch and bound approach is also not a good alternative as it is exceedingly difficult to develop general expressions for the lower bounds on makespan. Hence, the authors proposed using combinatorial approaches based on the dominance conditions in the sequence, by starting the list with n partial schedules each containing only one job, and then generating new partial schedules for each of these partial schedules, by augmenting each of the unscheduled jobs to the end of the partial schedule. The completion time is then calculated for each permutation and those with a minimum value are retained, then the procedure is repeated until a full schedule is obtained. In view of the high computation time required for this combinatorial approach, several heuristics based on the optimal rule of linear processing function are also proposed. The computational experiment was performed only for small size problem instances of up to 7 jobs.

A similar problem with another variance of linear processing time function was discussed by Kunnathur and Gupta (1990). The processing time of a job consists of a fixed part p_i and variable part v_i , i.e., $p'_i = p_i + v_i$; the variable part can be considered as the penalty for a job i , which depends on the start time S_i of the job, i.e., $b_i = \max\{0, \alpha_i(S_i - d_i)\}$; where d_i is the latest start time of job i . If the

job i does not start by d_i , then its processing time increases at the rate of α_i per unit time that S_i exceeds. Besides dynamic programming, the branch and bound approach is also proposed with the following lower bound:

$$LB_{\langle A \rangle} = \sum_{i \in B} p'_i + \sum_{i \in \langle A \rangle} [p'_i + \max\{0, \alpha_i (S_i - d_i)\}] \quad (1.1)$$

Where $\langle A \rangle$ is the sequence of jobs $A \subset J$ that has been scheduled to start after the jobs in $B = J - A$. In addition to optimal algorithms, heuristic solutions are based on the lower bound obtained above. The computational experiment for the branch and bound with the best heuristic rule was performed with up to 15 jobs.

The linear processing time function can also be assumed to have bounded deterioration, as seen in a study by Kubiak and van de Velde (1998), where the actual processing time $p'_i(t)$ of job i then depends on its start time t and can be defined as following:

$$p'_i(t) = \begin{cases} p_i, & \text{if } t \leq d, \\ p_i + \alpha_i(t - d), & \text{if } d < t < D, \\ p_i + \alpha_i(D - d), & \text{if } t \geq D \end{cases} \quad (1.2)$$

where p_i is the basic processing time of job i , α_i is the deterioration rate of job i , d is the common critical date, and D is the common maximum deterioration date. This problem was proven to be NP Hard by transforming the partition

problem. A dynamic programming algorithm and a branch and bound algorithm were developed based on the following optimal properties:

- The sequence of jobs that start before d (early jobs) does not affect the solution optimality,
- The sequence of jobs that start after D does not affect the solution optimality,
- The jobs that start after d but before D are sequenced in non-decreasing order of α_i/p_i (Gupta and Gupta, 1988),
- The jobs that start before d and completes after d has processing time no smaller than any of the early jobs
- If $p_k \leq p_l$ and $w_k \leq w_l$, then J_k precedes J_l

A study on non-linear deterioration model was first attempted by Mosheiov (1995) with a step deterioration processing time function, in which processing time increases at a certain time points only. Both single and multiple maintenance periods were analyzed. In the case of a single step deterioration, each job is characterized by a critical deadline, basic processing time (if a job starts prior to, or on its deadline), and new processing time (if the job is late); as for the case of the general case of multi-step deterioration, each job is characterized by a number of deadlines and several possible processing times, depending on the job's starting time. The problem was showed to be NP Complete using the knapsack problem transformation, and heuristic approaches based on Moore's algorithm and

Jackson's Lemma (Jackson, 1955) were proposed, which result in a solution somewhat equivalent with an Earliest Due Date (EDD) sequence. A similar problem was studied by Cheng and Ding (2001). A practical example of the step deterioration was given by considering the overtime payment rate in a manufacturing system of $\$(x + y)$ per hour while normal rate is $\$(x)$ per hour, in which $y = (\alpha)(x)$ with α can be considered as the deterioration rate of the model. The objective of the example is to minimize the total cost of production, which is proportional to the makespan of the intended model. Another non-linear model can be found in study by Wu, et al. (2009), who dealt with a piecewise linear model on the processing time function. A branch and bound approach was proposed to obtain the optimal solution. A heuristic approach was developed based on the optimal property of a non-decreasing sequence of the ratio between deterioration rate and basic processing time (Gupta and Gupta, 1988). A computational experiment is performed to compare the heuristic developed, and a classic Simulated Annealing (SA) algorithm. The SA, which has the advantage of avoiding getting trapped in a local optimal, was observed to result in a better solution.

Scheduling Problems with time dependent processing times and Group Technology Assumption

Together with a scheduling problem with time-dependent processing times, another important class of a scheduling problem closely correlated to setup

scheduling is to be discussed in this section. It is characterized by the group technology assumption, i.e., jobs are classified into groups where no machine setups are required between two consecutively scheduled jobs from the same group; however setup is required between jobs of different groups. In this assumption, all jobs from the same group are to be scheduled continuously. The group technology assumption can be considered as a special case of a general setup scheduling problem. Several previous publications provided an analysis of a scheduling problem with time-dependent processing times using the group technology assumption.

The first study was performed by Guo and Wang (2005), which divides n jobs into f groups to be processed on a single machine. Setup times between jobs from different groups s_i were assumed to be sequence independent, and the processing of the job may not be interrupted (non-resumable assumption). The processing time function is described as $p'_{ij} = p_{ij}(a + bt)$, where a and b are positive constants. The makespan minimization problem was proven to be polynomial time solvable. In the optimal algorithm, jobs in each group can be sequenced in any order; while the groups are to be sequenced in a non-decreasing order of

$s_i(1 + bA_i)/(bA_i)$ where $A_i = \sum_{k=1}^{n_i} b^{k-1} C_{n_i}^k(p_{i1}, p_{i2}, \dots, p_{in_i})$ and $C_n^i(p_1, p_2, \dots, p_n)$ is the sum of C_n^k items, where each item is the product of i numbers of $p_1, p_2, \dots, p_n$.

A similar result is also obtained for the case where the group setup time is not constant, i.e., it grows as they wait for the processing (Wu, et al., 2008) since

longer setup or preparation might be necessary as food quality deteriorates or the patient's condition worsens. Both processing times and group setup times of the jobs are a linear function of its starting time, according to the basic linear function, $p'_{ij} = \alpha_{ij}t$ and $s'_i = \beta_i t$. A case with linear function $p'_{ij} = \alpha_{ij}(a + bt)$ and $s'_i = \beta_i(a + bt)$ has also undergone similar analysis and produced a similar result of being solvable in polynomial time (Wang, et al., 2008). Further analysis of the same problem was performed by analyzing a particular case where the schedule of groups was given with a general processing time function of $p'_{ij} = a_{ij} + b_{ij}t$ and general setup time function of $s'_i = c_i + d_i t$. In this situation, the optimal schedule can be obtained by scheduling jobs in non-decreasing order of a_{ij}/b_{ij} .

2.6 DYNAMIC SETUP SCHEDULING LITERATURE

Our Problem Part 3 deals with setup scheduling problems with a dynamic job arrival assumption. In the literature, studies on dynamic setup scheduling can be traced back to the analysis of the *dynamic lot sizing problem*. The *lot sizing* model aims to determine the optimal timing and level of production (Jans and Degraeve, 2007). At one end of the spectrum of assumption, there exists the continuous time scale, continuous demand, and infinite time horizon. The well-known Economic Order Quantity (EOQ), which considers one type of product, and the Economic Lot Scheduling Problem (ELSP), which considers multiple types of product,

belongs to this group. At the other end of the spectrum, it is assumed that the time scale is discrete, demand is dynamic, and time horizon is finite. This type of problem is referred to as *dynamic lot sizing* (Jans and Degraeve, 2007).

The essence of dynamic lot sizing problem lies in finding a production plan that satisfies deterministic and dynamic demand and minimizes the total costs of production, setup, and inventory over a multi-period horizon. There are several dynamic lot sizing models that can be found in literature (Drexel and Kimms, 1997; Fleischmann and Meyr, 1997; Jans and Degraeve, 2007; Zipkin, 1991):

- The single type Un-capacitated Lot Sizing problem (ULS) is the simplest kind of dynamic lot sizing problem as only a single type of product is being considered. The classic Wagner-Whitin Procedure is one of the solutions available in solving this kind of problem (Wagner and Whitin, 1958).
- The multi type Capacitated Lot Sizing Problem (CLSP), where the planning interval is divided into periods, each with a specific demand, and the decision variables are the production quantities of each type of product in every period. In this model, more than one type of product can be produced on the same machine in one time period.
- The Continuous Setup Lot Sizing problem (CSLP) assumes that during each time period, at most one type of product can be produced on the machine with a continuous production policy, implying that capacity per period does not have to be exhausted entirely.

- The Proportional Lot Sizing Scheduling Problem (PLSP) relaxes the CSLP constraint of producing at most one product at each time period. In this model, at most two different types of products can be produced on the machine at each period. With this model, the unused remaining capacity that might occur in the CSLP model can be avoided as it can be used to produce the second type of product in the particular period.
- The Discrete Lot Sizing and Scheduling Problem (DLSP) has a similar formulation to CSLP. However, it also has a discrete production policy assumption, implying that a product must be produced at full capacity. It is based on an “all or nothing” assumption, i.e., in any period of time, either it processes one product at full capacity, or it is perfectly idle.
- The General Lot Sizing Problem (GLSP) is the modification of DLSP where each lot is uniquely assigned to a position number in order to define a sequence. With this, the drawbacks of the DLSP model due to a large number of time periods can be avoided.

The CLSP model is called a *large bucket model*, as several types of the product may be produced per period while CSLP, PLSP, DLSP, and GLSP are considered as *small bucket models*, as at most one type of product can be produced per period (Eppen and Martin, 1987). A major advantage of the *small bucket model* over the *large bucket model* is exact control of the sequence of lots, which opens the possibility of including a sequence-dependent setup (Fleischmann, 1994).

The setup scheduling problem discussed in our study has the closest form to the DLSP and GLSP model. The first DLSP model analysis with sequence-dependent setup was performed by Fleischmann (1994). The problem was formulated as an asymmetric travelling salesman problem with time windows. Due to the problem complexity, a heuristic approach was preferred. A basic initial heuristic was followed by an improving heuristic based on an Or-Opt operation, which consisted of cutting off a piece of the current tour and inserting it at another (earlier or later) position. The heuristic performance was compared with a tight lower bounds obtained by means of Lagrangian relaxations.

Gicquel, et al. (2009) proposed a new way of modeling the production system for DLSP with a sequence-dependent setup. Products in manufacturing systems are described as combinations of physical attributes, and to exploit this description, it is possible to reduce the size of the mixed integer program to be solved. A small illustrative example was given to describe the model, by considering a bottle fitting line where 4 types of products can be produced. A product is described by its corresponding bottle size (with two possible values) and the composition of the liquid to be used (also with two possible values). With that, the type of product to be considered are (1,1), (1,2), (2,1), and (2,2), as the idle period is described as (0,0). A setup cost matrix is generated for each attribute.

As for the GLSP model, the only study found that considered a sequence-dependent setup is a journal paper by Fleischmann and Meyr (1997). As the schedule is independent of predefined time periods, this model generates another

restricted time structure. With that, a 2-level time structure exists, i.e., the predefined time structure called micro-periods, and the restricted time structure, called macro-periods. The basic idea is to model the external dynamic of the system, such as demand and holding costs, using a fixed discrete time grid. As for the internal dynamics of the system which can occur anytime, control is exercised using decision variables and *independent* of the external dynamics. Both internal and external dynamics are linked by the assignment of micro-periods to macro-periods.

A multilevel time structure was also previously applied by Sharifnia, et al. (1991) in solving real time dynamic setup scheduling and flow control, which they called a *hierarchical framework*. The hierarchy structure was arranged based on the occurrence frequency of an event, with the least frequent events assigned to the highest level of hierarchy, and it is assumed that events from different hierarchies are independent of each other. Setup changes constitute one level of hierarchy in between machine failure state (above) and the actual dispatching (below). Each level schedules its own controllable event and determines the rates at which the controllable event of the lower level must be scheduled. At the setup level, the production rate of the part types in each setup and the exact timing of the setup changes to be determined. In order to determine the production rate of the part types in each setup, a *feedback control policy*, which is based on surplus state of the production system, was proposed. As for determining the exact timing of the setup changes, the *corridor policy* and a *limit cycle* are implemented. This

approach is one of the alternatives provided by the literature in handling dynamic job arrival. However, this model only provides a general framework of setup scheduling policy with no solution proposed for how to sequence the queuing jobs to the machine. Chapter 5 of this thesis proposes a method for sequencing jobs with a dynamic arrival into the machine in order to maximize machine utilization.

2.7 SUMMARY

This chapter reviews the current literature on single machine scheduling with setup and / or maintenance consideration. In doing so, a general taxonomy of scheduling literature is generated, which classifies previous studies based on general attributes such as type of manufacturing system, job arrival model and assumption, and setup and maintenance assumption. Subsequently, deeper analysis is performed by reviewing those literatures which either considers setup activity only or maintenance activity only. Those studies are classified by adopting literature taxonomy from the existing review papers on setup or maintenance scheduling. From all those taxonomies, the position of this study is described. Several gaps in the literature which need to be bridged is identified before this study can provide a comprehensive understanding of the importance of considering setup and maintenance activity in sequencing and scheduling jobs in order to utilize limited machine capacity maximally. Lastly, literature on scheduling with time-dependent processing times is reviewed for the benefit of

Part 2 problem analysis and literature on dynamic setup scheduling for Part 3 problem analysis.

Chapter 3 of this dissertation discusses the result of this analysis. In Chapter 4, the analysis is extended for the case of a flexible maintenance period, in which maintenance activity can be performed earlier or later with trade-off considerations between machine time (capacity) and product yield. Job arrival assumption of previous studies for analysis with consideration of setup and maintenance is of deterministic and static job arrival; the analysis is further extended in Chapter 5 into deterministic and dynamic job arrival analysis; hence the algorithm proposed is more implementable for actual industry application.

3. SETUP SCHEDULING WITH FIXED MAINTENANCE PERIOD AND STATIC JOB ARRIVAL

As mentioned in the previous chapter, the setup scheduling problem is divided into three parts based on the assumption of job arrival and the maintenance period. In this chapter, the first part of the analysis on setup scheduling problems with maintenance consideration is discussed. The simplest form of assumption is made, with a fixed maintenance period, i.e., once for every tool life T , and a deterministic-static job arrival. This analysis extends the previous studies on setup scheduling problems with a maintenance consideration by taking into account the fact that setup and maintenance activity can be performed simultaneously. The result of the analysis would be an essential basis for the analysis of subsequent parts with a relaxed assumption on maintenance period and/or job arrival.

3.1 PROBLEM FORMULATION

The scheduling problem is formulated as follows:

- There are n jobs to be scheduled on a single machine within a finite time horizon; all the jobs are available at time zero.
- Each job j has a distinctive processing time p_j , and each of them would require a different machine configuration with a sequence-dependent setup time s_{ij} , in which job i is processed just before job j .
- The jobs cannot be interrupted (non-resumable assumption).
- Tool is subject to tool life T (time-based), and once reached it cannot be used for production, and maintenance activity is to be performed. Maintenance time m is constant.
- Setup and maintenance activity can be performed simultaneously.
- At time zero, it is assumed that the machine has just undergone maintenance and can be setup for the first job in sequence (regardless of what is chosen), and that this setup time is not included in the makespan.
- Objective: To minimize makespan ($C_{\max}$).

Figure 3.1: Problem Illustration

Figure 3.2: Part 1 Problem Formulation

Figure 3.1 illustrates the formulated problem, and Figure 3.2 describes the Part 1 Problem formulation.

Numerical Example

- Number of Jobs: $n = 7$
- Tool life: $T = 16$
- Maintenance time: $m = 4$
- Processing time: $p_1 = 10; p_2 = 10; p_3 = 5; p_4 = 2; p_5 = 4; p_6 = 9; p_7 = 9$
- Setup time matrix:

s	J ₁	J ₂	J ₃	J ₄	J ₅	J ₆	J ₇
J ₁	-	2	4	1	1	2	4
J ₂	5	-	3	3	3	5	5
J ₃	1	5	-	2	5	5	4
J ₄	4	3	1	-	5	3	1
J ₅	5	3	5	3	-	2	4
J ₆	4	1	1	4	4	-	5
J ₇	4	3	2	2	4	3	-

Figure 3.3: Graphical representation of numerical example of Part 1 Problem

Figure 3.3 shows the graphical representation of the numerical example. In order to minimize makespan, the problem is solved optimally by means of a branch and bound algorithm as discussed in section 3.3. An optimal sequence for the problem with makespan value of 69 units of time is as follows:

$$\Rightarrow J_1-J_5-M-J_7-J_3-M-J_2-J_4-M-J_6$$

Figure 3.4: Graphical representation of optimal sequence

An optimal sequence is shown in Figure 3.4. In the first block, J_1 and J_5 are processed on the machine, which requires 15 units of time including the 1 unit of setup time s_{15} in between the two jobs. Afterwards, the machine is left idle for 1 unit of time before performing a fixed scheduled maintenance for 4 units of time on the machine. Setup activity s_{57} is performed simultaneously during this maintenance period, which happens such that s_{57} also requires 4 units of time as shown in the setup time matrix.

In between the second and third block, setup time $s_{32}=5$ requires a longer time than maintenance activity $m=4$, thus the first job of the third block, i.e., J_2 , can only be started after s_{32} is finished. While in between the third and fourth block, setup time $s_{46}=3$ is shorter than maintenance time, so J_6 can be started straight away after maintenance activity.

3.2 PROBLEM COMPLEXITY

The following theorem shows that our formulated problem (as described in section 3.1) is an NP-hard problem:

Theorem 3.1:

Problem of minimizing makespan $C_{\max}$ for a group of jobs $J_1, J_2, \dots, J_n$ on a single machine with non-resumable multiple period of maintenance

(non-availability constraint) and sequence-dependent setup consideration is NP-Hard.

Proof:

This formulated problem (noted P_1) consists of the following two subset problems (as shown in Figure 3.5):

1. Problem of minimizing makespan with non-resumable multiple periods of maintenance consideration.
2. Problem of minimizing makespan with a sequence-dependent setup consideration.

Subset problem 1 has been proven to be an NP Hard problem in the strong sense by transforming 3-partition problem (Lee, 1996).

As for subset problem 2, this is a variation of the classic Traveling Salesman Problem (TSP), which is known to be NP complete (Garey and Johnson, 1979). In this case, the problem of minimizing total setup time is equivalent with the problem of minimizing makespan.

Because the subset problem 1 is NP-Hard Problem and the subset problem 2 is NP-complete, our formulated problem P_1 is also proven to be NP-Hard. $\square$

Figure 3.5: Problem Complexity

Next, the problem complexity is analyzed further by determining the number of enumerations for the formulated problem. The total number of enumerations can be calculated as follows:

⇒ Total number of enumerations of sequencing n jobs with sequence-dependent setup time = $n!$

⇒ Total number of enumerations of allocating n jobs to its respective maintenance block:

$$\begin{aligned}
 & \begin{array}{ccc} n \text{ block} & (n-1) \text{ block} & 1 \text{ block} \\ \downarrow & \downarrow & \downarrow \end{array} \\
 & = \binom{n}{n} + \binom{n}{n-1} + \dots + \binom{n}{1} \\
 & = \binom{n}{n} + \binom{n}{n-1} + \dots + \binom{n}{1} + \binom{n}{0} - \binom{n}{0} \\
 & = 2^n - 1
 \end{aligned}$$

⇒ Hence, the total number of enumerations for a problem with n number of jobs $= (n!) \times (2^n - 1)$, which means with 10 numbers of jobs, more than 3 million enumerations are resulted.

3.3 OPTIMAL SOLUTION: BRANCH AND BOUND ALGORITHM

I propose a branch and bound procedure to obtain an optimal solution for the formulated problem as described in section 3.1

3.3.1 Lower Bound

The lower bound is calculated based on the sum of column minima of reduced matrix of completion time c . Completion time c_{ij} is defined as the total time required to perform setup on the machine and to process job j immediately after job i . A convention is made by associating job 0 to the maintenance activity. An example of the completion time matrix is shown below:

Figure 3.6: Completion time illustration

Setup time matrix:

s	J ₁	J ₂	J ₃
J ₁	-	3	4
J ₂	5	-	4
J ₃	4	4	-

As shown in Figure 3.6, completion time c_{12} can be calculated by summing up setup time s_2 and process time p_2 :

$$\Rightarrow c_{12} = s_{12} + p_2 = 3 + 4 = 7$$

Completion time matrix:

c	J ₁	J ₂	J ₃
J ₁	-	7	13
J ₂	13	-	13
J ₃	12	8	-

Job 0 (maintenance activity) is not included in the completion time matrix in order to avoid constant lower bound value of $c_{0j} = p_j$. Although $c_{0j} = p_j$ is a valid lower bound value for the problem, it is not a very tight lower bound and will induce a higher computation time.

With that, the lower bound is described on the following theorem:

Theorem 3.2:

$$\text{Let } LB = \sum_{k=1}^n c_k \quad (3.1)$$

c_k = Column minima of column k of completion time matrix, i.e.

$$\min(c_{1k}, c_{2k}, c_{3k}, \dots, c_{nk})$$

The quantity LB is a lower bound on the problem P_1 of minimizing $C_{\max}$.

Proof:

The sum of column minima $\sum_{k=1}^n c_k$ can be defined as follows:

$$\sum_{k=1}^n c_k = \min(c_{11}, c_{21}, c_{31}, \dots, c_{n1}) + \min(c_{12}, c_{22}, c_{32}, \dots, c_{n2}) + \dots + \min(c_{1n}, c_{2n}, c_{3n}, \dots, c_{nn})$$

$$\text{Let: } \min(c_{11}, c_{21}, c_{31}, \dots, c_{n1}) = c_1$$

$$\min(c_{12}, c_{22}, c_{32}, \dots, c_{n2}) = c_2$$

$$\vdots$$

$$\vdots$$

$$\min(c_{1n}, c_{2n}, c_{3n}, \dots, c_{nn}) = c_n$$

If we consider any feasible solution, its makespan can be calculated as follows:

$$\Rightarrow C_{\max} = c_{01} + c_{12} + \dots + c_{(r-1)0} + c_{0r} + \dots + c_{(n-1)n}$$

in which $c_{(r-1)0}$ represents the maintenance activity that happens after job $(r-1)$, plus any idle time between the end of job $(r-1)$ and the start of the maintenance task; while c_{0r} represents the time to complete task r given that it is the first task after a maintenance activity.

Since $c_{(r-1)0}$ has to be at least equal to the length of maintenance time m :

$$\Rightarrow c_{(r-1)0} \geq m$$

and c_{0r} has to be at least the processing time, plus any setup time that goes beyond the maintenance time:

$$\Rightarrow c_{0r} \geq p_r + \max(0, s_r - m)$$

in which p_r is processing time of job r , and s_r is associated setup time for the column minima.

Therefore:

$$\Rightarrow c_{(r-1)0} + c_{0r} \geq m + p_r + \max(0, s_r - m) = p_r + \max(s_r, m) \geq p_r + s_r = c_r$$

$$\Leftrightarrow c_{(r-1)0} + c_{0r} \geq c_r$$

Since $c_{12} \geq c_2; c_{23} \geq c_3; \dots; c_{(n-1)n} \geq c_n$:

$$\Rightarrow C_{\max} = c_{01} + c_{12} + \dots + c_{(r-1)0} + c_{0r} + \dots + c_{(n-1)n}$$

$$\Leftrightarrow C_{\max} \geq c_{01} + c_2 + \dots + c_r + \dots + c_n$$

$$\Leftrightarrow C_{\max} \geq \sum_{k=1}^n c_k$$

$$\Leftrightarrow C_{\max} \geq LB$$

The inequality above is still valid for cases with more than one maintenance event, as the more maintenance event being included on the sequence would only further increase the value of the left hand side of the inequality.

Hence, LB is a valid lower bound for the problem P_1 of minimizing makespan $C_{\max}$. $\square$

3.3.2 Branching Strategy

To maximize the branch and bound algorithm effectiveness, depth first and best lower bound first is adopted in the node selection procedure. This algorithm assigns jobs in a forward manner starting from the first position. It begins with an entirely undetermined schedule node $(-, -, \dots, -)$, branches the node into $(J_1, -, \dots, -)$, $(J_2, -, \dots, -)$, $\dots$, $(J_n, -, \dots, -)$; then chooses the node with the best lower bound, let's say $(J_1, -, \dots, -)$, and branches it further into $(J_1, J_2, -, \dots, -)$, $(J_1, J_3, -, \dots, -)$, $\dots$, $(J_1, J_n, -, \dots, -)$, and $(J_1, M, -, \dots, -)$. The process is repeated until we obtain the complete schedule. In each step, the node is pruned if it violates the tool life T requirement or if its lower bound is higher than the upper bound value. The upper bound value is initially obtained by finding feasible solution using certain meta-heuristics (to be discussed in Section 3.4), and continuously updated anytime a better feasible solution than the current solution is found during the branch and bound procedure. These steps continue until all nodes are explored to give us an optimal solution.

3.3.3 Branch and Bound Algorithm

Step 1: Obtain initial upper bound with meta-heuristic

Details of the meta-heuristic procedure are described in section 3.4.

$\Rightarrow \text{upper_bound} := \text{heuristic_value}$

Step 2: Initialize Live list

Live list consists of Lower Bound value $LB(P_0)$, length of nodes $L(P_0)$, and node P_0 itself, e.g., node $P_0 = 3-0-2-1$ with length $L(P_0)$ of 4 and assumed lower bound value $LB(P_0) = 40$ will be defined as $(40, 4, 3-2-2-1)$ inside the Live list.

$\Rightarrow \text{Live} := \{(LB(1), 1, 1), (LB(2), 1, 2), \dots, (LB(n), 1, n)\}$

Repeat the following steps until $\text{Live} = \emptyset$:

Step 3: Select Node from Live list

Based on the branching strategy of depth first and best lower bound first, the node with lowest lower bound value $LB(P)$ and longest length $L(P)$ is chosen first. Once chosen, that respective node is removed from the Live list.

$\Rightarrow \text{Live} := \text{Live} \setminus \{(LB(P), L(P), P)\}$

Step 4: Branching

Branch on selected node P generating $P_1, P_2, \dots, P_b$ by adding in maintenance activity and those jobs that have not been included into the node.

For $1 \leq i \leq b$:

Step 5: Pruning

Fathom P_i if non-feasible due to tool life constraint, else:

If $LB(P_i) = f(X)$ for a feasible solution X
and $f(X) < \text{upper_bound}$:

$\text{upper_bound} := f(X); \text{solution} := X$

Go to EndBound

If $LB(P_i) \geq \text{upper_bound}$ then fathom P_i

else $\text{Live} := \text{Live} \cup \{LB(P_i), L(P_i), P_i\}$

EndBound

End of For

End of Repeat

Step 6: Obtain optimal solution and optimal value

$\text{optimal_solution} = \text{solution}$

$\text{optimal_value} = \text{upper_bound}$

3.3.4 Numerical Example of Branch and Bound Procedure

In order to describe the branch and bound technique more clearly, the following simple numerical example is solved with the procedure:

- Number of Jobs: $n = 3$
- Tool life: $T = 16$
- Maintenance time: $m = 4$
- Processing time: $p_1 = 8; p_2 = 8; p_3 = 10$
- Setup time matrix:

s	J ₀	J ₁	J ₂	J ₃
J ₀	-	0	0	0
J ₁	0	-	5	4
J ₂	0	3	-	4
J ₃	0	4	5	-

The completion time matrix can be obtained as follows:

c	J ₀	J ₁	J ₂	J ₃
J ₀	-	8	8	10
J ₁	4	-	13	14
J ₂	4	11	-	14
J ₃	4	12	13	-

The initial upper bound is calculated with a heuristic solution, and a value of 48 is obtained. The next step is to start the branching procedure, and then prune those nodes with a lower bound value higher than or equal than upper bound and those non-feasible nodes due to life constraint. Figure 3.7 describes the branching procedure, in which the underlined index number on the branch’s arrow describes the sequence of branching according to the branching strategy stated in section 3.3.2.

Figure 3.7: Branch & Bound Procedure Example

Lower Bound computation for node 1:

- Node's makespan = $c_{01} = 8$
- Reduced matrix of completion time is as follows. Column 1 is removed as job 1 is included in node 1:

c	J ₁	J ₂	J ₃
J ₁		13	14
J ₂	1	-	14
J ₃	12	13	-

The sum of column minima:

$$\begin{aligned}
 &= \sum_{k=1}^n c_k \\
 &= 0 + c_2 + c_3 \\
 &= c_{12} + c_{13} \\
 &= 13 + 14 = 27
 \end{aligned}$$

- Hence, lower bound of node 1 = $8 + 27 = 35$

Lower Bound computation for node 1-2:

- Node's makespan = $c_{01} + c_{12} = 21 > (T + m)$

Node 1-2 is pruned due to infeasibility as it violates the tool life constraint.

Lower Bound computation for node 2-1:

- Node's makespan = $c_{02} + c_{21} = 19$
- Reduced matrix of completion time is as following. Column 2 and column 1 are removed as both job 2 and job 1 are included in node 2-1. Row 2 is also removed as a setup activity is performed after job 2:

c	J ₁	J ₂	J ₃
J ₁	-	13	14
J ₂	11	-	14
J ₃	11	12	13

The sum of column minima:

$$= c_{13}$$

$$= 14$$

- Hence, lower bound of node of node 2-1 = $19 + 14 = 33$

In Figure 3.7, node 1-2, 1-3, 2-3, 3-1, 3-2, 2-1-0, and 2-1-3 are pruned due to infeasibility (tool life constraint), while node 1-0-2, 1-0-3, 2-0-1, 2-0-3, 3-0-1, and 3-0-2-0 are pruned because their lower bound value is higher than updated upper bound value of 40, which turned out to be the optimal value of the problem.

Hence, with a branch and bound procedure, an optimal solution is obtained as 3-0-2-1 with an optimal value of 40. This optimal value has also been further validated with a simple enumeration.

3.4 HEURISTIC SOLUTION: GRASP ALGORITHM

Branch and bound technique described in previous section is able to provide the optimal solution, but the computation time of this approach grows exponentially which affects its practicality for a problem of larger size. In order to obtain a decent solution with a short computation time, heuristic procedures are proposed.

3.4.1 Overview of GRASP meta-heuristics

To overcome the intractability of a branch and bound algorithm for a large number of job problems, three meta-heuristics are described in this section. The

meta-heuristics proposed are based on GRASP ideas developed by Feo and Resende (1995). It is known to be a powerful approach for solving a combinatorial optimization problem, and found to be superior to another powerful approach, the tabu search procedure, based on a study by T.A. Feo, et al. (1996). Another advantage of the GRASP meta-heuristics is its simplicity. Contrary to what occurs with other meta-heuristics such as a tabu search or genetic algorithms, which use a large number of parameters in their implementations, the GRASP only requires the adjustment of a single parameter, i.e., the maximum number of iterations.

“The GRASP is an iterative process, in which each of the iterations consist of two phases: a construction phase and a local search phase. The construction phase builds a feasible solution, whose neighborhood is investigated until a local minimum is found during the local search phase. The best overall solution is kept as the result.” (T. A. Feo and Resende, 1995)

In the formulated problem, machine time is to be shared to perform processing, maintenance, and setup, while the remainder is the idle time. As both processing times and maintenance periods are fixed, minimal makespan can be achieved by simply minimizing the sum of the setup and idle time. Based on this straightforward optimal property of the formulated problem, meta-heuristics are developed. Depending on the sequence of job insertion into the block, three meta-heuristics are derived as follows:

1. Descending Heuristic: Jobs are inserted into the block in descending order of the remaining time of the respective block.

The motivation behind proposing a descending heuristic is to minimize the amount of setup time by inserting jobs with the smallest completion time into the block.

2. Ascending Heuristic: Jobs are inserted into the block in ascending order of the remaining time of the respective block.

The motivation behind proposing an ascending heuristic is to minimize the amount of idle time by inserting jobs with the largest completion time into the block.

3. Random Heuristic: Jobs are inserted into the block randomly.

The motivation behind proposing a random heuristic is to broaden the search area for the solution as simultaneously minimizing both setup and idle time cannot be accomplished by either a descending heuristic or an ascending heuristic.

3.4.2 GRASP Algorithm

Step 1: Decide the maximum number of iterations

Step 2: Initialization

$\Rightarrow \text{iteration} = 0; \text{heuristic_solution} = \{\}; \text{heuristic_value} = \infty$

Step 3: Create job list A, based on available jobs to be processed

$$\Rightarrow A = \{J_1, J_2, \dots, J_n\}$$

Step 4: Open e^{th} block solution set S (e) and randomly transfer one job from set A into S (e). $e = 1, 2 \dots$ etc

$$\Rightarrow S(e) := J_i; A := A \setminus \{J_i\}$$

Step 5: Check set A. Go to Step 8 if $A = \emptyset$

Step 6: Check whether any of the jobs from set A can be inserted into S (e) without violating the tool life constraint of S (e). Go to Step 4 if none.

Step 7: Transfer jobs with smallest (Descending Heuristic) / largest (Ascending Heuristic) / randomly (Random Heuristic) completion time from A to S (e) and then perform local search procedure to improve the sequence within the block. Go to Step 5.

Step 8: Calculate the makespan value of S.

If $\text{makespan}(S) < \text{heuristic_value}$:

$$\text{heuristic_solution} = S$$

$$\text{heuristic_value} = \text{makespan}(S)$$

- Step 9:** Iteration = Iteration +1. Go to Step 2 if Iteration < maximum number of iteration
- Step 10:** Output final solution heuristic_solution and heuristic_value , then Stop.

3.4.3 Numerical Example of GRASP meta-heuristics

In order to describe the three heuristic procedures more clearly, the following simple numerical example is solved with the procedure:

- Number of Jobs: $n = 7$
- Tool life: $T = 16$
- Maintenance time: $m = 4$
- Processing time: $p_1 = 5; p_2 = 2; p_3 = 3; p_4 = 1; p_5 = 9; p_6 = 8; p_7 = 6$
- Setup time matrix:

s	J ₁	J ₂	J ₃	J ₄	J ₅	J ₆	J ₇
J ₁	-	1	2	2	2	1	4
J ₂	2	-	2	2	3	2	4
J ₃	2	5	-	3	5	5	3
J ₄	3	2	3	-	4	3	2
J ₅	1	3	1	2	-	1	4
J ₆	4	1	4	3	2	-	3
J ₇	2	2	3	5	1	4	-

Figure 3.8 shows an optimal solution by means of a branch and bound algorithm which gives a makespan value of 48:

Figure 3.8: Optimal solution with Branch & Bound

GRASP Descending Heuristic:

Figure 3.9 shows the heuristic of a GRASP descending heuristic:

Figure 3.9: GRASP Descending Heuristic Solution

A makespan value of 53 is obtained with this algorithm, which is 10.42% away from the optimal solution. The above solution is the best solution obtained with 70 iterations of the construction phase. At each block, the first job to be processed is selected randomly. In this case, J_2 is chosen to be first processed at block 1. Next, the remaining time on the respective block for each unprocessed job is calculated as shown in Table 3.1. If J_1 is to be processed after J_2 , the remaining time on the block 1 = $T - p_2 - c_{21} = 16 - 2 - 7 = 7$

Job	Completion time	Remaining time on respective block
J ₁	$c_{21} = 7$	7
J ₂	-	-
J ₃	$c_{23} = 5$	9
J ₄	$c_{24} = 3$	11
J ₅	$c_{25} = 12$	2
J ₆	$c_{26} = 10$	4
J ₇	$c_{27} = 10$	4

Table 3.1: List of the remaining time on respective block

For a descending heuristic, the job with the largest remaining time is to be processed first. Hence, J₄ is processed after J₂ as shown in Figure 3.9, if the local search procedure is not able to improve the makespan value by interchanging the sequence. This process is continued until all jobs are processed without violating the tool life constraint.

It can be observed that more idle time is generated by sequencing jobs in this way, as we always choose those jobs with the shortest completion time to be processed first. This observation is to be confirmed during computational experiments in section 3.5.2.1.

GRASP Ascending Heuristic:

Figure 3.10 shows the solution of a GRASP ascending heuristic with a makespan value of 52, i.e., 8.3% away from the optimal value of 48.

Figure 3.10: GRASP Ascending Heuristic Solution

The GRASP descending heuristic has an almost similar procedure, except in this case, the job with smallest remaining time is to be processed first.

It can be observed that this way of sequencing results in higher values of setup time being generated, as we always choose those jobs with the largest completion time to be processed first. In many cases, the jobs with the largest completion time would also mean that they have a higher setup time in comparison with the rest of the unprocessed jobs.

GRASP Random Heuristic:

Figure 3.11 shows the solution of a GRASP random heuristic with a makespan value of 49, which is only 2.08% away from the optimal value of 48. This solution is the best that can be obtained from 700 iterations being performed during the construction phase.

Figure 3.11: GRASP Random Heuristic Solution

The GRASP random heuristic broadens the search area as the restriction of sequencing jobs based on a descending or ascending order of the remaining time

on respective blocks is removed. The solution quality can be improved by increasing the number of iterations, which means a longer CPU time is required.

3.5 COMPUTATIONAL EXPERIMENT

In order to compare the performance of the branch and bound and three developed heuristic solutions of GRASP meta-heuristics, the algorithms were coded in Matlab and tested on an Intel Core™ 2 CPU with 3GHz and 3292MB RAM.

3.5.1 Design of Experiment

The processing time of jobs and a setup time matrix were generated randomly from an integer uniform distribution on (1, 10) and (1, 5) respectively while the maintenance time is fixed at $m = 4$. In order to test the effect of the range of the maintenance period, three values of tool life T are tested, i.e., $T=16$, $T=26$, and $T=36$. Six different numbers of jobs ($n = 6, 7, 8, 9, 10$, and 11) were considered, in which the total number of enumerations of the formulated problem goes up to 81.75 billion for $n=11$. From each combination of machine tool life and number of jobs, 30 random instances were generated; hence, a total of 540 random instances were tested for the following algorithms:

- i. Optimal Solution Branch and Bound Algorithm (BB)
- ii. GRASP Descending Heuristic (GRASP-DH)
- iii. GRASP Ascending Heuristic (GRASP-AH)

iv. GRASP Random Heuristic (GRASP-RH)

For GRASP-DH and GRASP-AH, the number of iterations for the construction phase is set at $10xn$, and the number of iterations for the local search procedure is at $5xn$. As for GRASP-RH, the number of iterations is set at $100xn$ and $5xn$ respectively.

3.5.2 Results Analysis

The overall result of the experiment is summarized in Table 3.1. A comparison of solution quality among heuristic algorithms (GRASP-DH, GRASP-AH, and GRASP-RH) is performed in terms of its closeness to the optimal solution obtained from BB. Delta percentage is used as the performance measure in this case, which is calculated with the following formula:

$$\text{Delta}(\%) = \frac{(\text{heuristic value} - \text{optimal value})}{\text{optimal value}} \times 100\% \quad (3.2)$$

Computational times (in seconds) of the four algorithms are also compared.

The results of the computational experiment are summarized in Table 3.2.

n	I	Branch & Bound (BB)				GRASP-DH				GRASP-AH				GRASP-RH			
		CPU time (s)		No of nodes		CPU time (s)		Delta (%)		CPU time (s)		Delta (%)		CPU time (s)		Delta (%)	
		Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max
n=6	I=16	0.67	4.02	213.23	1,673	0.36	0.75	11.09	34.00	3	out of 30	0.38	0.69	7.10	27.27	9	out of 30
	I=26	0.79	1.83	112.17	660	0.71	1.30	10.74	31.37	4	out of 30	0.66	1.14	4.80	18.18	9	out of 30
	I=36	1.36	2.19	68.83	330	1.42	2.19	9.37	41.18	9	out of 30	1.29	2.17	5.64	20.59	9	out of 30
n=7	I=16	1.49	7.44	713.07	2,731	0.54	1.08	15.87	34.00	2	out of 30	0.56	1.11	9.87	24.00	7	out of 30
	I=26	1.58	4.44	278.37	1,300	1.18	2.58	12.10	36.73	1	out of 30	1.20	1.48	4.66	12.50	8	out of 30
	I=36	2.10	4.73	240.43	629	2.12	4.70	10.95	23.68	2	out of 30	1.79	4.70	5.23	15.38	5	out of 30
n=8	I=16	4.50	34.16	1,693.77	7,723	0.87	1.66	16.12	25.00	1	out of 30	0.90	1.55	8.10	17.50	4	out of 30
	I=26	3.14	6.25	735.97	3,035	2.29	4.44	10.52	26.42	1	out of 30	2.06	3.05	6.31	19.64	5	out of 30
	I=36	3.74	8.63	517.67	1,333	3.74	9.13	8.85	25.71	2	out of 30	3.05	8.61	4.18	10.20	3	out of 30
n=9	I=16	66.22	1029.30	7,050.67	47,237	1.26	2.39	17.14	35.44	0	out of 30	1.29	2.28	6.94	23.64	3	out of 30
	I=26	6.52	23.73	1,830.13	7,107	2.99	4.58	13.49	26.42	0	out of 30	2.75	4.88	7.47	16.98	0	out of 30
	I=36	6.14	11.56	911.10	2,393	5.82	11.39	11.64	25.00	0	out of 30	4.63	8.00	5.33	16.67	2	out of 30
n=10	I=16	160.60	853.94	14,742.27	40,981	1.92	3.92	17.58	29.59	0	out of 30	1.82	2.91	8.82	15.71	0	out of 30
	I=26	24.27	80.13	5,531.37	13,320	4.09	8.64	14.53	25.97	0	out of 30	3.55	5.28	6.07	13.79	0	out of 30
	I=36	16.33	152.73	2,716.07	18,733	7.98	13.30	11.23	28.99	0	out of 30	7.04	9.56	6.04	12.50	0	out of 30
n=11	I=16	3287.63	12463.00	63,935.40	153,319	2.47	3.94	20.06	31.65	0	out of 30	2.34	3.38	10.23	16.67	0	out of 30
	I=26	731.40	14257.00	23,073.67	166,889	5.82	11.70	12.81	25.64	0	out of 30	5.17	7.00	6.21	14.55	0	out of 30
	I=36	119.03	1835.00	7,870.23	61,953	11.30	17.98	11.52	25.35	0	out of 30	9.70	14.78	6.58	14.08	2	out of 30
AVG		246.53		7346.36		3.16		13.09		2.79		6.64		29.57		3.43	

Table 3.2: Part 1 Computational Experiment Results

3.5.2.1 Heuristics CPU time and delta percentage comparison

Figure 3.12 and Figure 3.13 show the comparison among heuristics in terms of CPU time and delta percentage, respectively:

Figure 3.12: Heuristics CPU time comparisons

Figure 3.13: Heuristics delta percentage comparisons

The following observations are made:

- The descending heuristic requires a low CPU time with an average of 3.16s; however, it provides feasible solutions with poor makespan performance with an average delta percentage of 13.09%. The poor performance of this heuristic is due mainly to its tendency to produce

solutions with a high percentage of machine idle time, as shown in Figure 3.14.

- The ascending heuristic requires a low CPU time, similar to the descending heuristic, with an average of 2.79s. However, it is able to give feasible solutions with a moderate level of makespan performance with an average delta percentage of 6.64%. This approach is able to reduce the percentage of machine idle time. However, a relatively small search area due to ascending order restrictions limits this approach from getting a solution nearer to the optimal solution. Also, by choosing those jobs with the largest completion time to be processed first, a higher percentage of setup time is induced with this method of sequencing as shown in Figure 3.15.
- The random heuristic is able to provide a solution with the highest level of makespan performance compared to the other two meta-heuristics, with an average delta percentage of 3.43%. However, a high CPU time is required due to the random nature of the algorithm as a higher number of iterations are required in order to obtain a decent solution during the search. A bigger search area is key to this solution, with a trade-off in the higher CPU time required.

Figure 3.14 and figure 3.15 show the comparison of average idle time and average setup time among four algorithms, respectively.

The GRASP-DH, which sequences jobs by processing those jobs with the smallest completion time first, has a significantly higher idle time in comparison with the rest of the algorithms. However, the GRASP-AH, which sequences jobs with the largest completion time first, consistently has the highest setup time, with no significant differences. This explains the moderate makespan performance levels from GRASP-AH, since no obvious setup time differences are observed despite this algorithm doing worse than GRASP-DH in terms of minimizing setup time.

Figure 3.14: Average idle time comparison

Figure 3.15: Average setup time comparison

Further analysis was performed to determine the effect of the number of jobs n and tool life T . The Analysis of Variance (ANOVA) with 2-factors was performed with the following results:

ANOVA			
Source of Variation	GRASP-DH P-value	GRASP-AH P-value	GRASP-RH P-value
Number of Jobs n	0.275242	0.668710	0.014306
Tool Life T	0.000264	0.003818	0.000006

Table 3.3: 2-way ANOVA results

Based on ANOVA with 99% confidence level on the computational experiment data, the heuristics delta percentage is only significantly affected by tool life T (p -value < 0.01), as shown in Table 3.3. On the other hand, the effect of the number of jobs n is not statistically significant (p -value > 0.01). However, the effect of the number of jobs n on the random heuristic is found to be more apparent in comparison with the other two heuristics, which is expected due to the random nature of the algorithm.

3.5.2.2 CPU time of the optimal algorithm

The main drawback of the optimal algorithm is the high computation time which might be intractable for a large problem size. The CPU time comparison between the optimal algorithm BB and the GRASP-RH, which has the highest level of makespan performance, results in the following results shown in Figure 3.16:

Figure 3.16: CPU time comparison between BB and GRASP-RH

In comparison with the other two heuristics, GRASP-RH was found to have a higher CPU time requirement, which increased in a linear way across n . This is contrary to the branch and bound algorithm CPU time, which increases in an exponential way across n . Hence, the GRASP-RH would be a good alternative to solve scheduling problem with a large number of jobs which is intractable by means of branch and bound method, because it is able to provide a feasible solution with a decent level of makespan performance.

Another minor observation that can be made from this experiment is the CPU time trend of branch and bound algorithm across tool life T , as shown in Figure 3.17:

Figure 3.17: CPU time and Number of Nodes across tool life T

The BB algorithm takes a shorter time to solve a problem with larger tool life T, as a larger maintenance activity induces less maintenance activity required, which in turn results in a shorter height of the branch and bound tree and fewer nodes.

3.5.3 Lower Bound Effectiveness

The lower bound effectiveness is relatively measured by comparing its computation performance to another valid lower bound. While the proposed lower bound is calculated based on the sum of column minima (as described by Theorem 3.2 in Section 3.3.1), the second lower bound is calculated based on sum of processing time and estimated maintenance time required. The second lower bound is defined as follows:

$$LB' = \sum_{i=1}^n p_i + m \times \left\lceil \frac{\sum_{i=1}^n p_i}{T} \right\rceil \quad (3.3)$$

p_i = processing times of job i

m = maintenance times

T = tool life of the machine

The validity proof of the lower bound LB' is presented as follows:

Proof:

Makespan can be defined as the summation of all the processing times, setup times, maintenance times, and idle times of a sequence. Hence, if we remove setup times and idle times from this equation, we will obtain the following inequality:

$$C_{\max} \geq \sum_{i=1}^n p_i + m \times q,$$

in which q is defined as the number of maintenance activity in a sequence.

$$\text{Since } q \geq \left\lceil \frac{\sum_{i=1}^n p_i}{T} \right\rceil, \text{ hence } C_{\max} \geq \sum_{i=1}^n p_i + m \times \left\lceil \frac{\sum_{i=1}^n p_i}{T} \right\rceil \Leftrightarrow C_{\max} \geq LB'$$

Hence, LB' is a valid lower bound for the formulated problem P_1 of minimizing makespan $C_{\max}$. $\square$

The computation performance of branch and bound algorithm with lower bound LB was compared with the one with lower bound LB' . Five random sample for

each number of jobs $n=6, 7, 8, 9, 10$ and 11 were generated. Each of these random samples was solved with branch and bound algorithm with lower bound LB and LB' , which were coded in Matlab and tested on an Intel Core™ 2 CPU with 3GHz and 3292MB RAM. Upper limit of computation time is set at 1 hour. The results are summarized in the following Table 3.3:

Number of jobs	Average of computation time (seconds)	
	Lower bound LB	Lower bound LB'
$n=6$	1.38	1.93
$n=7$	2.7	23.87
$n=8$	2.75	1096.86
$n=9$	17.8	no solution within 1 hour
$n=10$	272.39	no solution within 1 hour
$n=11$	909.45	no solution within 1 hour

Table 3.4: Comparison of LB and LB'

The computation time of branch and bound algorithm with lower bound LB' increased much more rapidly than the one with lower bound LB . Neglecting sequence-dependent setup time on the lower bound calculation reduced the pruning efficiency of the branch and bound algorithm,, as shown from the experiment results.

3.5.4 Simultaneous Setup and Maintenance

In order to benchmark the simultaneous setup and maintenance against a process that does setup and maintenance as separate activities, a simple comparison is performed between the average makespan values of the two processes. Comparison is performed by means of branch and bound techniques on the same

data sets used in the above experiments. Delta percentage of ~10% is observed, which implies that based on this specific data sets, around 10% machine time can be saved by performing setup and maintenance simultaneously.

Figure 3.18: Comparison of simultaneous vs. separate setup and maintenance

3.6 SUMMARY

In this chapter, we discussed sequence-dependent setup scheduling problems with fixed maintenance consideration and deterministic static job arrival assumption, to minimize makespan on a single machine system. The branch and bound method is able to solve the formulated problem reasonably well. As larger tool life T induces shorter CPU time, the problem with a larger number of jobs can potentially be solved within a reasonable time for the case of larger tool life T .

Three meta-heuristics were also developed based on GRASP ideas and the

optimal properties of minimizing the sum of setup and idle time. An ascending heuristic (GRASP-AH) was found to be a good option if a quick solution with moderate level of makespan performance is intended. While the random heuristic (GRASP-RH) provides a solution with better makespan performance, it requires a relatively high CPU time. Nevertheless, it would still be a good alternative to the branch and bound method for solving a problem with a large number of jobs, as its CPU time increases in a linear way across n , while the branch and bound method increases in an exponential way. The results of this Part 1 problem analysis form the basis for subsequent analysis with flexible maintenance period (Chapter 4) and dynamic job arrival (Chapter 5).

4. SETUP SCHEDULING WITH FLEXIBLE MAINTENANCE PERIOD AND STATIC JOB ARRIVAL

The previous chapter discussed setup scheduling problems with deterministic and static job arrival and assumed that periodic maintenance activity is scheduled at fixed intervals with no flexibility whatsoever. In this chapter, the analysis is extended to consider setup scheduling problems with a flexible maintenance period, i.e., the maintenance period can be shifted flexibly earlier or later than tool life T with a tradeoff between machine time and product yield. The product yield is assumed to deteriorate if maintenance activity is not performed once tool life T is reached. The results of this analysis would be helpful as a quantification tool in determining when the best time to perform maintenance activity, in order to maximize machine utilization.

4.1 PROBLEM FORMULATION

The scheduling problem is formulated as follows:

- There are n jobs to be scheduled on a single machine within a finite time horizon; all the jobs are available at time zero.
- Each job j has a distinctive processing time p_j , and each of them would require a different machine configuration with a sequence-dependent setup time s_{ij} , in which job i is processed just before job j .
- The jobs cannot be interrupted (non-resumable assumption).
- *Tool has tool life T (time-based); once tool life is reached, certain machine quality (yield) deterioration is incurred. (The deterioration value is defined by the quality deterioration model of the application, one model based on semiconductor industry application is proposed in section 4.2).*
- Setup and maintenance activity can be performed simultaneously.
- At time zero, it is assumed that the machine has just undergone maintenance and can be setup for the first job in sequence (regardless of what is chosen), and that this setup time is not included in the makespan.
- Objective: To minimize makespan ($C_{\max}$).

Figure 4.1: Part 2 Problem Formulation

Figure 4.1 describes the Part 2 Problem Formulation. Flexible maintenance is introduced in this model, in which the maintenance period can be shifted flexibly earlier or later (T' and T''). A maintenance trade-off exists in this problem (noted problem P2), in which high maintenance time (capacity loss) is incurred when maintenance activity is performed too often; whereas refraining from maintenance activity would induce lower product yield or production rate.

This formulated problem P2 is strongly NP Hard, as shown in section 3.2.

4.2 QUALITY DETERIORATION MODEL

It is assumed that quality deterioration on the machine affects the production rate of the machine if it is being used for production beyond its tool life T . In this quality deterioration model, machine yield is translated into the machine's processing rate, i.e., reduction in product yield means an increment in the processing time requirement. The model developed is helpful in determining the

increment in processing time requirement if a job is processed in a deteriorated machine.

The quality deterioration model is described in the following formula:

Given that y is a number between 0 and 1 to describe the product yield of the machine after tool life T if maintenance activity is not performed before tool life T , then the processing time requirement of the deteriorated tool can be calculated as:

$$p' = p + \alpha p, \text{ in which } \alpha = \frac{1}{y} - 1 \quad (4.1)$$

p' = processing time requirement on deteriorated tool (units of time).

p = processing time requirement on normal tool with 100% product yield (units of time).

α = additional time rate (dimensionless constant).

Proof:

In a normal tool with 100% product yield, the processing time requirement can be defined as $p = \frac{S}{V}$, in which S is the production target (in units), i.e. the amount of jobs to be processed on the machine; and V is the machine processing rate for normal / non-deteriorated conditions (in units/units of time).

In a deteriorated machine, a higher processing time requirement is incurred. Let p' be the processing time requirement when a job is processed in a deteriorated machine, i.e. $p' = p + x$, in which x is the additional time in the processing time requirement.

In this model, we assume that the processing time requirement increment is determined by the additional time rate α , i.e. $x = \alpha p \Rightarrow p' = p + \alpha p$

If we assume that the deteriorated tool has a product yield of y , the processing time requirement can be defined as $p' = \frac{S}{(y)(V)}$

$$\Rightarrow V \times p = (y)(V) \times p'$$

$$\Leftrightarrow p' = \frac{p}{y} = p + \frac{1-y}{y} p = p + \left(\frac{1}{y} - 1\right)p$$

Since p' is defined as $p' = p + \alpha p$, hence $\alpha = \frac{1}{y} - 1$

So, given that the product yield of the machine would deteriorate into y if maintenance activity is not performed before tool life T , then the processing time requirement of the deteriorated tool can be calculated as: $p' = p + \alpha p$, in which

$$\alpha = \frac{1}{y} - 1. \quad \square$$

The quality deterioration model described in the above formula is a tool to determine the increment of the processing time requirement on a deteriorated

machine. In calculating the increment (additional time), there are two cases to be considered:

Case 1: A job process is started after tool life is reached and/or exceeded:

Figure 4.2: Additional time computation case 1 illustration

Assume job i is processed on a deteriorated tool after tool life is reached, as shown in Figure 4.2. The normal processing time requirement of job i if it is processed in a machine with 100% yield is defined as p_i . However, this processing time requirement would increase into p_i' if it is processed in a deteriorated tool.

Hence:

$$p_i' = p_i + x_i \Rightarrow p_i' = p_i + \alpha p_i \quad (4.2)$$

Case 2: A job is started before tool life is reached

Figure 4.3: Additional time computation case 2 illustration

Assume job i is processed on a deteriorated tool before tool life is reached as shown in Figure 4.3. The normal processing time requirement of job i if it is processed in a machine with 100% yield is defined as p_i . However, this processing time requirement would increase into p_i' if it is processed in a deteriorated tool.

Let x_i be the additional time in processing time requirement, since job i is still processed in normal machine condition for $(p_i - d)$ units of time, hence:

$$p_i' = p_i + x_i \Rightarrow p_i' = p_i + \alpha d_i \quad (4.3)$$

In order to further describe the quality deterioration model, the following numerical example illustrates the calculation of process time increments due to the lower product yield of a deteriorated machine.

Numerical Example

- Number of Jobs: $n = 7$

- Tool life: $T = 20$
- Maintenance time: $m = 4$
- Processing time: $p_1 = 10; p_2 = 3; p_3 = 7; p_4 = 5; p_5 = 9; p_6 = 8; p_7 = 5$

$\alpha = 0.1$, i.e., yield drop to ~90% if maintenance is not performed after tool life T (translation of α value into yield drop value is based on the yield deterioration model formula).

Setup time matrix:

	J_1	J_2	J_3	J_4	J_5	J_6	J_7
J_1	-	5	1	3	5	2	2
J_2	3	-	2	5	1	1	5
J_3	4	3	-	3	4	1	5
J_4	4	5	4	-	2	4	3
J_5	5	1	2	5	-	2	3
J_6	4	2	1	3	3	-	5
J_7	1	5	1	2	4	1	-

In the following segment, the optimal solutions for both fixed period assumption and flexible period assumption obtained by means of branch and bound algorithms are presented:

Optimal Solution (Fixed Maintenance Period assumption):

Optimal Solution: J_4 - J_5 - J_2 -M- J_1 - J_6 -M- J_7 - J_3 ($C_{\max} = 62$)

Figure 4.4: Fixed period assumption optimal solution

For the fixed period assumption, as the maintenance period has to take place exactly once every tool life T , the machine yield is maintained at 100% throughout the process as shown in Figure 4.4.

Optimal Solution (Flexible Maintenance Period assumption)

Optimal Solution: $J_7-J_1-J_3-M-J_4-J_5-J_2-J_6$ ($C_{\max} = 58.2$)

Figure 4.5: Flexible period assumption optimal solution

For the flexible period assumption, we can choose to continue processing jobs on the machine beyond its tool life, with the expense of a lower yield due to the deteriorated machine. Part of J_3 and the entire of J_6 are processed beyond the tool life T ; hence it is processed with machine yield of $\sim 90\%$, which implies a lower production rate. The additional time of $x_3 = (\alpha\alpha)_3 = (0.1)(4) = 0.4$ and

$x_6 = (\alpha\alpha)(_6) = (0.1)(8) = 0.8$, are required. Maintenance activity is only performed after J_3 has been processed, and machine yield would be back to 100% afterwards, until the subsequent tool life is reached.

Note:

In this quality deterioration model, yield loss is assumed to follow a step deterioration model as shown in Figure 4.5 for purposes of simplicity and clarity. A more complex model, e.g., a linear or quadratic yield loss model (as shown in Figure 4.6); can be incorporated into this quality deterioration model. It will not be covered in this study, because analyzing different yield loss model would not give much significance or insight on how to solve the formulated scheduling problem. Also, the quality deterioration model proposed in this thesis is more relevant for semiconductor industry application; however the scheduling method that is developed in the next section is applicable for any kind of quality deterioration model, depending on the industry application.

Figure 4.6: Deterioration Model

4.3 OPTIMAL SOLUTION: BRANCH AND BOUND ALGORITHM

In this formulated problem P2, an optimal solution is also obtained by means of a branch and bound procedure.

4.3.1 Lower Bound

The lower bound from the previous chapter (Section 3.3.1) is proven to be still valid for the second part of the problem. The lower bound formula is described in the following theorem as follows:

Theorem 4.2:

$$\text{Let } LB = \sum_{k=1}^n c_k \quad (4.4)$$

c_k = Column minima of column k of completion time matrix, i.e.

$$\min(c_{1k}, c_{2k}, c_{3k}, \dots, c_{nk})$$

The quantity LB is a lower bound on the problem P₂ of minimizing $C_{\max}$.

Proof:

The sum of column minima $\sum_{k=1}^n c_k$ can be defined as follows:

$$\sum_{k=1}^n c_k = \min(c_{11}, c_{21}, c_{31}, \dots, c_{n1}) + \min(c_{12}, c_{22}, c_{32}, \dots, c_{n2}) + \dots + \min(c_{1n}, c_{2n}, c_{3n}, \dots, c_{nn})$$

Let:

$$\min(c_{11}, c_{21}, c_{31}, \dots, c_{n1}) = c_1$$

$$\min(c_{12}, c_{22}, c_{32}, \dots, c_{n2}) = c_2$$

$$\vdots$$

$$\vdots$$

$$\min(c_{1n}, c_{2n}, c_{3n}, \dots, c_{nn}) = c_n$$

Additional time = x

Consider a feasible solution as shown in the following:

Figure 4.7: Feasible Solution Illustration

If we consider any feasible solution as shown in Figure 4.7, its makespan can be calculated as follows:

$$\Rightarrow C_{\max} = c_{01} + c_{12} + \dots + c_{(r-1)0} + c_{0r} + \dots + c_{(n-1)n} + x$$

in which $c_{(r-1)0}$ represents the maintenance activity that happens after job $(r - 1)$, plus any idle time between the end of job $(r - 1)$ and the start of the maintenance task; while c_{0r} represents the time to complete task r given that it is the first task after a maintenance activity.

Since $c_{(r-1)0}$ has to be at least equal to the length of maintenance time m :

$$\Rightarrow c_{(r-1)0} \geq m$$

and c_{0r} has to be at least the processing time, plus any setup time that goes beyond the maintenance time:

$$\Rightarrow c_{0r} \geq p_r + \max(0, s_r - m)$$

in which p_r is processing time of job r , and s_r is associated setup time for the column minima.

Therefore:

$$\Rightarrow c_{(r-1)0} + c_{0r} \geq m + p_r + \max(0, s_r - m) = p_r + \max(s_r, m) \geq p_r + s_r = c_r$$

$$\Leftrightarrow c_{(r-1)0} + c_{0r} \geq c_r$$

Since $c_{12} \geq c_2; c_{23} \geq c_3; \dots; c_{(n-1)n} \geq c_n$:

$$\Rightarrow C_{\max} = c_{01} + c_{12} + \dots + c_{(r-1)0} + c_{0r} + \dots + c_{(n-1)n} + x$$

$$\Leftrightarrow C_{\max} \geq c_{01} + c_{12} + \dots + c_{r-1} + \dots + c_n$$

$$\Leftrightarrow C_{\max} \geq \sum_{k=1}^n c_k$$

$$\Leftrightarrow C_{\max} \geq LB$$

The inequality above is still valid for cases with more than one maintenance event, as the more maintenance event being included on the sequence would only further increase the value of the left hand side of the inequality.

$C_{\max}$ in the above inequality described makespan of one maintenance event, while the result presented in the thesis considered the overall makespan of the problem (consists of several maintenance event). In flexible maintenance case, the number of maintenance event is lower than the fixed one, because in each maintenance period we are allowed to go beyond tool life T . Please refer to the numerical example in section 4.2 for illustration.

Hence, LB is also a valid lower bound for the problem P_2 of minimizing $C_{\max}$. $\square$

4.3.2 Branching Strategy

The same branching strategy as the part 1 branch and bound procedure is applied, i.e., depth first and best lower- bound first. More details are described in section 3.3.2.

4.3.3 Branch and Bound Algorithm

In the part 2 branch and bound algorithm, the search for an optimal solution is performed by considering the possibility of solutions with the number of maintenance periods at 1 up to the solution with the maximum number of maintenance periods. The maximum number of maintenance periods is calculated based on the sequencing solution, which is obtained via the nearest neighbor method.

Another difference from the part 1 branch and bound algorithm is that the tool life constraint on the part 2 problem is based on the maximum value between normal tool life and extended tool life. The overall algorithm is shown as follows:

Step 1: Determine maximum number of maintenance periods based on sequencing solution, obtained via nearest neighbor method:

$$MP = \left\lceil \frac{NM}{T} \right\rceil \quad (4.5)$$

MP = maximum number of maintenance periods

NM = makespan value of nearest neighbor sequencing solution

For $1 \leq k \leq MP$:

Step 2: Calculate modified tool life T_m

$$T_m = \max\left(T, \frac{NM}{k}\right) \quad (4.6)$$

Step 3: Initialize live list (as described in section 3.3.3)

Repeat the following steps until Live = {}:

Step 4: Select Node from Live list (as described in section 3.3.3)

Step 5: Branching (as described in section 3.3.3)

For $1 \leq i \leq b$ do step 6

Step 6: Pruning based on modified tool life T_m (as described in section 3.3.3)

End of For

End of Repeat

End of For

Step 7: Obtain optimal solution and optimal value

4.3.4 Numerical Example of Branch and Bound Procedure

In order to describe the part 2 branch and bound technique more clearly, the following simple numerical example is solved with the procedure:

- Number of Jobs: $n = 6$
- Tool life: $T = 20$

- Maintenance time: $m = 4$
- Processing time: $p_1 = 4; p_2 = 1; p_3 = 7; p_4 = 2; p_5 = 8; p_6 = 5$
- $\alpha=0.1$, i.e., yield drop to ~90% if maintenance is not performed after tool life T
- Setup time matrix:

s	J ₀	J ₁	J ₂	J ₃	J ₄	J ₅	J ₆
J ₀	-	0	0	0	0	0	0
J ₁	0	-	3	2	4	2	2
J ₂	0	1	-	4	2	2	5
J ₃	0	3	4	-	2	4	3
J ₄	0	3	1	4	-	2	2
J ₅	0	2	2	5	5	-	2
J ₆	0	1	4	3	3	3	-

- Completion time matrix:

c	J ₀	J ₁	J ₂	J ₃	J ₄	J ₅	J ₆
J ₀	-	4	1	7	2	8	5
J ₁	4	-	4	9	6	10	7
J ₂	4	5	-	11	4	10	10
J ₃	4	7	5	-	4	12	8
J ₄	4	7	2	11	-	10	7
J ₅	4	6	3	12	7	-	7
J ₆	4	5	5	10	5	11	-

The nearest neighbor method is implemented in order to determine the maximum number of maintenance periods. The nearest neighborhood method was chosen, as it is a quick solution to obtain any feasible solution. The starting point of the nearest neighborhood procedure is chosen randomly. After 70 iterations, the solution of J6-J1-J2-J4-J5-J3 with a makespan value of 40 is obtained. Hence, the

maximum number of maintenance periods: $MP = \left\lceil \frac{40}{20} \right\rceil = 2$

The next step is to determine whether the solution with one or two maintenance periods would result in an optimal solution.

Solution with one maintenance period:

Modified tool life: $T_m = \max(20, \frac{40}{1}) = 40$, which means we are allowed to extend tool life up to 40 units of time for the solution with one maintenance period by applying the branch and bound procedure, as in the problem part 1 (as described in section 3.3.4). However, the modified tool life T_m instead of tool life T is used. The following solution was obtained, with a makespan value of 36:

$$\Rightarrow J_5-J_6-J_1-J_3-J_4-J_2$$

Figure 4.8: Solution with one maintenance period

J₃, J₄, and J₂ are performed beyond tool life T . Hence, the additional times x_3 , x_4 , and x_2 are incurred as shown in Figure 4.8. The additional times are calculated as follows: $x_3 = (0.1)(7) = 0.7$; $x_4 = (0.1)(2) = 0.2$; $x_2 = (0.1)(1) = 0.1$

Solution with two maintenance periods:

Modified tool life: $T_m = \max(20, \frac{40}{2}) = 20$

The following solution is obtained with a makespan value of 37:

$\Rightarrow J_6-J_1-J_3-J_0-J_4-J_2-J_5$

Figure 4.9: Solution with two maintenance period

As the maintenance period is assumed to be flexible, the second maintenance period is shifted earlier to avoid a unit of idle time as shown in Figure 4.9.

Optimal solution:

From above, we can see that the solution for one maintenance period was found to have a better makespan value, i.e., 36.11 units of time. Hence, during the iteration within the algorithm, the solution with one maintenance period would be chosen as our optimal solution for the problem.

In case the problem is slightly modified, in which the additional time rate is changed from $\alpha=0.1$ into $\alpha=0.2$, i.e., yield drops to 83% if maintenance is not performed after tool life T , the solution with one maintenance period will result in

a solution with a makespan value of 37.5. In this case, the solution with two maintenance periods (makespan = 37) would be our optimal solution.

4.4 HEURISTIC SOLUTION

To overcome the intractability of the branch and bound algorithm for larger problem sizes, three meta-heuristic processes are described in this section.

4.4.1 Overview

With flexible maintenance consideration, a schedule without idle time is the dominant schedule for minimizing makespan. Hence, a heuristic is developed where the only purpose is to sequence the jobs (with a sequence-dependent setup) and maintenance activities in the schedule to minimize makespan. Complicated constructive heuristics, as in the case of problem part 1, is not necessary; nevertheless the GRASP meta-heuristic remains to be tested for comparison purposes. Along with it, classic heuristics with straightforward procedures, such as Random Interchange and Simulated Annealing are tested.

The following three heuristics are tested in order to obtain a solution with a shorter computation time for the problem part 2:

1. Random Interchange (Croes, 1958).
2. Simulated Annealing (Kirkpatrick, et al., 1983)
3. GRASP Meta-heuristic (T.A. Feo, et al., 1996)

4.4.2 Heuristic Algorithms

4.4.2.1 Random Interchange Algorithm

Step 1: Determine the maximum number of maintenance periods MP based on a sequencing solution, which is obtained via nearest neighbor method (as described in section 4.3.3).

Step 2: Decide the maximum number of iterations

Step 3: Initialization:

$\Rightarrow N = 1$; iteration = 0; heuristic_solution = {}; heuristic_value = ∞

Step 4: Initialize random sequence

Step 5: 2-opt random interchange which results in sequence S and then calculates the makespan of the new sequence S

If makespan(S) < heuristic_value :

heuristic_solution = S

heuristic_value = makespan(S)

Step 6: Iteration = Iteration +1. Go to Step 5 if Iteration < maximum number of iteration

Step 7: N=N+1. Go to Step 4 if N $\leq$ MP

Step 8: Output final solution `heuristic_solution` and `heuristic_value`, then
Stop.

4.4.2.2 Simulated Annealing Algorithm

Step 1: Determine the maximum number of the maintenance period MP based on sequencing solution, which is obtained via nearest neighbor method (as described in section 4.3.3).

Step 2: Decide the maximum number of iteration1, maximum number of iteration2, and temperature reduction constant TRC (*Note: A separate experiment has been performed to determine the optimum value of TRC*).

Step 3: Initialization:

$\Rightarrow N = 1; \text{iteration1} = 0; \text{iteration2} = 0; \text{temperature} = 100;$

$\text{heuristic_solution} = \{\}; \text{heuristic_value} = \infty$

Step 4: Initialize random sequence

Step 5: Temperature reduction:

$\Rightarrow T = T - \text{TRC} \quad (4.7)$

Step 6: 2-opt random interchange which results in sequence S and then calculate makespan of the new sequence S

if makespan(S) < heuristic_value :

 heuristic_solution = S

 heuristic_value = makespan(S)

else

 Calculate acceptance probabilities:

$$P = e^{-\left(\frac{\text{makespan} - \text{heuristic_value}}{T}\right)} \quad (4.8)$$

And then generate a random number r between 0 and 1

if r > P :

 Revert S to the sequence before the 2-opt interchange

Step 7: iteration2=iteration2+1. Go to Step 6 if Iteration2 < maximum number of iteration2

Step 8: iteration1=iteration1+1. Go to Step 5 if Iteration2 < maximum number of iteration2

Step 9: N=N+1. Go to Step 4 if N ≤ MP

Step 10: Output final solution `heuristic_solution` and `heuristic_value`, then
Stop.

4.4.2.3 GRASP Meta-heuristic

Step 1: Determine the maximum number of maintenance periods `MP` based on the sequencing solution, obtained via nearest neighbor method (as described in section 4.3.3).

Step 2: Initialization:

$$\Rightarrow N = 1$$

Step 3: Calculate modified tool life T_m (as described in section 4.3.3).

Step 4: Part 1 GRASP algorithm (as described in section 3.4.2)

Step 5: $N=N+1$. Go to Step 4 if $N \leq MP$

Step 6: Output final solution `heuristic_solution` and `heuristic_value`, then
Stop.

In order to compare the performance of these three developed heuristic solutions and the branch and bound algorithm, they were coded in Matlab and tested on an Intel Core™ 2 CPU with 3GHz and 3292MB RAM. The details of this computational experiment are discussed on the section 4.5.

4.5 COMPUTATIONAL EXPERIMENT

4.5.1 Design of Experiment

The processing time of jobs and setup time matrix were generated randomly from an integer uniform distribution on (1, 10) and (1, 5) respectively while maintenance time was fixed at $m = 4$. In order to test the effect of the range of the maintenance period, three values were input for additional time rate, i.e., $\alpha=0.1$ (equivalent with machine yield of 91%), $\alpha=0.2$ (equivalent with machine yield of 83%), and $\alpha=0.3$ (equivalent with machine yield of 77%). Five different number of jobs ($n = 6, 7, 8, 9$, and 10) were considered. From each combination of machine tool life and number of jobs, 30 random instances were generated; hence, a total of 450 random instances were tested for the following algorithms:

- i. Optimal Solution Branch and Bound Algorithm (BB)
- ii. Random Interchange Heuristic (RI)
- iii. Simulated Annealing Heuristic (SA)
- iv. GRASP Random Heuristic (GRASP-RH)

4.5.2 Results Analysis

The overall result of the experiment is summarized in Table 4.1. A comparison of solution quality among heuristic algorithms (GRASP-DH, GRASP-AH, and GRASP-RH) was performed in terms of their closeness to the optimal solution

obtained from BB. The delta percentage was used as the performance measure in this case, which was calculated using the following formula:

$$\text{Delta}(\%) = \frac{(\text{heuristic value} - \text{optimal value})}{\text{optimal value}} \times 100\% \quad (4.9)$$

The results of the computational experiment are summarized in Table 4.1.

n	α	Branch & Bound (BB)			Random Interchange (RI)			Simulated Annealing (SA)			GRASP-RH		
		CPU time (s)	No of nodes		CPU time (s)	Delta (%)		CPU time (s)	Delta (%)		CPU time (s)	Delta (%)	
		Mean	Max	Mean	Max	Mean	Max	Mean	Max	Optimal solution	Mean	Max	Optimal solution
n=6	$\alpha=0.1$	2.38	3.20	586.93	990	5.00	9.31	0.05%	0.90%	27 out of 30	5.41	9.73	27 out of 30
	$\alpha=0.2$	2.58	3.45	569.57	1,025	6.00	9.33	0.04%	1.26%	29 out of 30	6.42	9.72	29 out of 30
	$\alpha=0.3$	2.65	3.59	515.00	1,128	6.60	9.27	0.01%	0.35%	29 out of 30	7.03	9.81	29 out of 30
n=7	$\alpha=0.1$	8.13	14.50	2,453.90	4,755	9.18	12.95	0.34%	3.56%	22 out of 30	9.70	13.38	22 out of 30
	$\alpha=0.2$	8.21	13.28	2,223.00	4,254	10.95	17.95	0.22%	1.74%	24 out of 30	11.35	18.61	24 out of 30
	$\alpha=0.3$	7.58	10.97	1,807.97	3,319	10.99	18.02	0.13%	2.34%	27 out of 30	11.52	18.50	28 out of 30
n=8	$\alpha=0.1$	40.61	102.80	8,414.80	15,498	13.64	24.42	0.14%	1.50%	25 out of 30	14.27	25.14	25 out of 30
	$\alpha=0.2$	39.18	71.97	7,983.80	12,625	16.76	24.34	0.76%	2.37%	11 out of 30	17.41	24.98	12 out of 30
	$\alpha=0.3$	27.88	52.27	5,927.27	10,078	18.16	24.41	0.40%	1.82%	20 out of 30	18.85	26.47	25 out of 30
n=9	$\alpha=0.1$	1,002.54	4,242.73	43,231.10	93,996	21.64	32.00	0.26%	2.30%	23 out of 30	22.46	32.69	23 out of 30
	$\alpha=0.2$	797.38	3,471.75	39,069.67	86,721	24.90	32.14	1.49%	4.19%	5 out of 30	25.61	32.61	16 out of 30
	$\alpha=0.3$	587.03	4,249.88	30,498.27	95,911	26.19	32.19	1.20%	3.70%	8 out of 30	26.76	32.50	15 out of 30
n=10	$\alpha=0.1$	16,228.78	49,844.28	195,330.23	432,605	30.82	52.98	0.97%	2.87%	9 out of 30	31.49	52.84	11 out of 30
	$\alpha=0.2$	17,408.39	204,043.23	177,859.83	618,214	34.54	53.41	2.71%	5.91%	0 out of 30	35.11	53.95	9 out of 30
	$\alpha=0.3$	7,793.59	63,249.34	118,784.27	348,668	36.54	53.59	2.39%	4.85%	1 out of 30	37.04	53.34	16 out of 30
AVG		2,930.46		42,350.37		18.13		0.74%			18.70		
											15.04		0.62%

Table 4.1: Part 2 Computational Experiment

4.5.2.1 Heuristics delta percentage comparison

Figure 4.10 shows a comparison among the heuristic methods in terms of delta percentage:

Figure 4.10: Heuristics delta percentage comparisons

The following observations are made:

- The random interchange heuristic has good makespan performance for a low number of jobs; however an increase in the number of jobs increases the delta percentage significantly. The significant makespan performance drop across the problem size is mainly due to the random nature of the solution, which means a greater number of iterations is required to achieve good solutions in the larger solution space, in a random way.
- The simulated annealing heuristic has the best makespan performance among the three tested solutions, with the delta percentage hovering below 1% for all additional time rates being tested. A proper choice of acceptance probabilities and annealing schedule is necessary for the heuristic to achieve a good solution in an effective manner.

- The GRASP heuristic has a decent makespan performance and the makespan performance drop across problem size is less apparent compared to the random interchange heuristic. The construction phase diminishes the random nature of the solution at a certain level, which might explain the less apparent makespan performance drop across number of the jobs, however, at the same time also restricts the search area of solutions.

4.5.2.2 The Fixed vs. Flexible Maintenance Period

The next comparison is between the makespan value obtained from analyses with a fixed periodic maintenance and those obtained from analysis with a flexible periodic maintenance. The comparison was done by calculating the delta percentage between those two makespan values as follows:

$$\text{delta percentage} = \frac{\text{makespan1} - \text{makespan2}}{\text{makespan1}} \times 100\% \quad (4.10)$$

makespan1 = makespan value with fixed maintenance period assumption

makespan2 = makespan value with flexible maintenance period assumption

Figure 4.11 depicts the delta percentage trend for the number of jobs $n=6, 7, 8, 9,$ and 10 for various values of additional time rate α of 0.1, 0.2, and 0.3; in which X-axes represent the 30 random instances and Y-axes represent the delta percentages.

Figure 4.11: Fixed maintenance period vs. Flexible maintenance period

It can be observed that the delta percentage is always larger than or equal to zero, which means allowing for flexible maintenance periods would result in better performance in terms of minimizing makespan.

Another observation is that the delta percentage monotonically decreasing across additional time rate α . Hence, there will always be a certain value of α for which delaying the maintenance period during a scheduling decision would not be favorable for minimizing makespan. As higher α implies a larger yield drop if the yield drop is found to be too high, delaying the maintenance period would not be a wise decision. Knowing the α cut-off value would be helpful in making a managerial decision about whether or not maintenance activity is to be delayed.

4.5.2.3 Early vs. Late Maintenance Period

Further analysis was performed to further understand whether the values of the flexible maintenance period are originated from being able to perform maintenance activity early or performing it late. In order to quantify the values of early maintenance and late maintenance respectively, each random instance was solved with an assumption of “flexible maintenance by only allowing early maintenance” to determine the delta percentage of early maintenance; and “flexible maintenance by only allowing late maintenance” to determine the delta percentage of late maintenance.

The overall average delta percentage was 6.09%, with a 4.17% delta percentage from performing early maintenance and 1.92% from performing late maintenance. Although early maintenance has higher delta percentage than late maintenance, the trend varies from one random instance to another, and it also depends on the value of additional time rate α . The detailed comparisons are shown in the following Figure 4.12:

Figure 4.12: Early vs. Late Maintenance Period

This model is able to provide a quantification model to support managerial decisions regarding maintenance activity. The cost of performing early maintenance and late maintenance are dependent on the industry application, e.g.:

- Early maintenance implies a higher frequency of maintenance activity in the long run, which means higher maintenance costs. Managerial decisions need to be made about whether it is worth it to perform maintenance early by considering how expensive the maintenance cost is and how much machine utilization can be saved by performing it early.
- In certain manufacturing systems, late maintenance might result in a lower production yield. Managerial decisions have to be made about whether product yield or machine utilization is the most critical parameter to be looked into at that point in time.

4.5.2.4 Meta-heuristic performance: Fixed vs. Flexible Maintenance Period

Meta-heuristics being tested for Problem Part 2 (flexible maintenance period) are found to have a much lower delta percentage compared to those being tested on Problem Part 1 (fixed maintenance period), even for the same GRASP-RH meta-heuristic that is being tested for both Problem Part 1 and Part 2. Figure 4.12 shows the GRASP-RH delta percentage comparison between fixed maintenance period assumption and flexible maintenance period assumption.

Figure 4.13: GRASP-RH Delta Percentage Comparison: Fixed vs. Flexible Maintenance Period

Two factors are identified as the cause of the significant gap:

1. Idle time (which does not exist in Problem Part 2) amplified the delta percentage of the Part 1 meta-heuristic.

When idle time components in the Part 1 solution (both optimal and meta-heuristic) are removed, the gap is reduced by more than 3%. As shown in Figure 4.14, in which the gap reduction is highlighted by the arrows.

Figure 4.14: Fixed GRASP-RH without idle time vs. Flexible GRASP-RH

- 2. Late maintenance (which is not allowed in Problem Part 1) reduces makespan and also compresses the delta percentage on the Part 2 meta-heuristic.

When late maintenance is not allowed on Part 2 solutions (both optimal and meta-heuristic), there is no more gap to be observed as shown in Figure 4.15.

Figure 4.15: Fixed GRASP-RH without idle time vs. Flexible GRASP-RH without late maintenance

4.6 EXAMPLE OF POSSIBLE SCENARIO

Let’s take an example of a semiconductor manufacturing machine, in which nine batches of wafers are to be processed on a batch cleaning machine. Each batch consists of 50 wafers, and each wafer is expected to yield 1000 chips, which are to be subsequently packaged in the next stage of the semiconductor manufacturing process. The batch cleaning machine is assumed to be a bottleneck

machine in the system due to its limited capacity, so its machine utilization is regarded as critical. The following is the information about the nine batches:

- Processing time: $p_1=60\text{mins}$, $p_2=70\text{mins}$, $p_3=60\text{mins}$, $p_4=70\text{mins}$, $p_5=50\text{mins}$, $p_6=100\text{mins}$, $p_7=30\text{mins}$, $p_8=100\text{mins}$, $p_9=20\text{mins}$
- Maintenance time: $m=40\text{mins}$;
- Tool life: $T=200\text{mins}$;
- Yield drop to 91% if maintenance is not performed after tool life T ;
- Setup time matrix (in minutes):

s	B ₁	B ₂	B ₃	B ₄	B ₅	B ₆	B ₇	B ₈	B ₉
B ₁	-	50	30	10	20	40	30	10	10
B ₂	40	-	20	20	10	10	40	30	10
B ₃	50	30	-	40	20	50	50	30	30
B ₄	50	20	40	-	50	10	20	10	30
B ₅	30	30	30	10	-	50	20	30	10
B ₆	20	50	20	50	50	-	20	10	50
B ₇	30	20	50	40	20	10	-	40	50
B ₈	50	50	20	30	20	50	30	-	50
B ₉	10	30	30	50	10	20	10	20	-

Optimal Solution (Fixed Maintenance Period assumption)

Optimal Solution: B₂-B₈-M-B₆- B₃-M-B₁-B₄-B₇-M-B₅-B₉ ($C_{\max} = 800\text{ mins}$)

Figure 4.16: Optimal Solution with Fixed Maintenance Period Assumption

Optimal Solution (Flexible Maintenance Period assumption)

Optimal Solution: B_7 - B_6 - B_8 - B_3 -M- B_2 - B_5 - B_9 - B_1 - B_4 ($C_{\max} = 701$ mins)

Figure 4.17: Optimal Solution with Flexible Maintenance Period Assumption

With a fixed period assumption, machine yield is maintained as maintenance activity is performed regularly at every tool life T ; as for a solution with a flexible maintenance period assumption, there is an occasion in which maintenance activity is delayed in order to save 99 mins of machine utilization. However, by delaying maintenance, a certain amount of yield loss is incurred (i.e., $0.09 \times 3 \times 50 \times 1000 = 13,500$ chips). The decision of whether to choose 99 mins of machine time or 13,500 chips is a managerial decision to be made depending on the criticality of machine time at the moment, product due date, customer priority, current product price, raw material cost, and other possible considerations.

In a semiconductor manufacturing system, late maintenance is penalized with certain amount of yield loss. With the help of quality deterioration model developed in Section 4.2, this yield loss value is translated into the extension of process time requirement. For any other industry applications which directly penalize late maintenance with longer processing time requirement, the

implementation of the scheduling algorithms presented in this chapter would have been more straightforward.

4.7 SUMMARY

In this chapter, we discussed the sequence-dependent setup scheduling problem with flexible maintenance consideration and deterministic static job arrival assumption to minimize makespan on a single machine system.

The optimal branch and bound algorithm and simulated annealing heuristic algorithm would be helpful in scheduling decisions involving production, setup, and maintenance consideration, in which the decision of performing or delaying maintenance activity is an additional consideration with a trade-off between machine capacity and production yield. While the optimal algorithm provides the best solution possible of minimizing makespan / maximizing utilization with a fairly long computation time, the simulated annealing heuristic algorithm provides a satisfactory solution with a 1% delta average compared to the optimal solution. The optimal algorithm is also useful in providing benchmark for how well the heuristic solution performs, as its exponential growth CPU time makes it non-practical as a day-to-day operational algorithm. Due to its static assumption, this algorithm can only be applied on the condition that every job is readily available to be processed into the machine. In Chapter 5, an analysis with dynamic job arrival is performed.

Comparison studies between the flexible maintenance period assumption analysis and the one with a fixed maintenance period assumption shows that delaying maintenance activity is beneficial up to a certain yield loss limit. By calculating this limit with the algorithm proposed, the procedure of decision making is made much simpler. However, the reliability of the limit is dependent on the translation process from product yield into the machine processing rate.

5. SETUP SCHEDULING PROBLEM WITH FIXED MAINTENANCE PERIOD AND DYNAMIC JOB ARRIVAL

While the extension of the basic setup scheduling problem is done on a fixed maintenance period assumption to a flexible one in the previous chapter, the basic setup scheduling problem is extended by assuming a dynamic job arrival assumption instead of a static one in this chapter. Dynamic job arrival is depicted with a rolling horizon model by assuming that job arrival for the next certain limited time horizon is known.

5.1 PROBLEM FORMULATION

The scheduling problem is formulated as follows:

- *Dynamic job arrival within a finite time horizon, based on a rolling horizon model with an assumption that job arrival for the next certain limited time horizon is known, defined by the width of Forecast Window (FW).*

- Each job j has a distinctive processing time p_j , and each of them would require a different machine configuration with sequence-dependent setup time s_{ij} , in which job i is processed just before job j .
- The jobs cannot be interrupted (non-resumable assumption).
- The tool is subject to tool life T (time-based), and once reached it cannot be used for production, in which maintenance activity has to be performed. Maintenance time m is constant.
- Setup and maintenance activity can be performed simultaneously.
- At time zero, it is assumed that the machine has just undergone maintenance and can be setup for the first job in sequence (regardless of what is chosen), and that this setup time is not included in the makespan or the job flow times.
- Objective: To minimize makespan ($C_{\max}$) and average of the job flow times ($(\sum_{i=1}^n C_i)/n$).

Figure 5.1: Part 3 Problem Formulation

Figure 5.1 describes the Part 3 Problem formulation. This formulated problem is strongly NP Hard, as shown in section 3.2.

Job arrival assumption:

Job arrival assumption is described with the following example:

- Assume that job arrival is generated based on a Poisson distribution with arrival rate $\lambda = 1$, and the job arrival for the next 10 units of time is known as shown in Figure 5.2, and job arrival information is periodically updated every 10 units of time
 - At time $t=0$, perfect job arrival information for $t = 0 - 10$ is known
 - At time $t=10$, perfect job arrival information for $t = 0 - 20$ is known
 - At time $t=20$, perfect job arrival information for $t = 0 - 30$ is known

Figure 5.2: Example of job arrival assumption

Despite analysis being performed in a dynamic environment, minimizing makespan is also analyzed as an objective function with the following justifications:

- The essence of this research study is to find the best way of allocating machine capacity for production, setup, and maintenance activity, i.e. maximizing machine utilization for production activity with availability constraint due to setup and maintenance activity, in which minimizing makespan is equivalent with maximizing machine utilization (Hopp and Spearman, 1996).

Since $\text{production rate} = \frac{\text{number of jobs}}{\text{makespan}}$ and $\text{utilization} = \frac{\text{production rate}}{\text{capacity}}$,

hence:

$$\Rightarrow \text{utilization} = \frac{\text{number of jobs}}{(\text{makespan}) \times (\text{capacity})}$$

- Studies on the single machine scheduling problem with consideration of both setup and maintenance activity is currently still at the early stage of development, in which analysis of the basic objective function of minimizing makespan is still required.
 - There are only two papers on the single machine scheduling problem with consideration of both setup and maintenance activity (with resumable assumption), which are the studies by W. J. Chen (2008) and W. J. Chen (2009).
 - There are only two papers on the single machine scheduling problem with consideration of both setup and maintenance activity (with the non-resumable assumption), which are the studies by Ángel-Bello, et al.(2011a) and Ángel-Bello, et al.(2011b).

5.2 ROLLING HORIZON PROCEDURE

In the rolling horizon procedure, the sequencing/scheduling decision is made periodically, once for each time the subsequent job arrival information is obtained (Baker and Peterson, 1979). A rolling schedule is formed by decomposing the time horizon into a multi-period problem and then solving it period by period. The first period's decision is solved and implemented, then one period later, the multi-

period model is updated and the process is repeated. This model only requires information about the limited number of time periods in the future.

There are practical benefits of working with this procedure:

- The procedure will require a shorter computation time as the large multi-period is decomposed into a number of smaller sub-problems.
- Forecasts pertaining to the near future tend to be more accurate than for the distant future, hence with this procedure only job arrival information with sufficient reliability is considered.

This procedure is further described with the following illustration:

- Consider the job arrival discussed in section 5.1. The finite horizon with 30 unit time is decomposed into three rolling horizons, Rolling Horizon 1 ($t = 0-10$), Rolling Horizon 2 ($t = 11-20$), and Rolling Horizon 3 ($t = 21-30$) as shown in Figure 5.3 below:

Figure 5.3: Rolling Horizon Procedure

- At time $t=0$, only job arrival information from Rolling Horizon 1 ($t = 0 - 10$) is known. The static sequence for these jobs is then solved. Based on the sequence solution obtained, job processing on the machine for $t = 0 - 10$ is carried out. Those jobs which have not been processed after $t = 10$ are carried forward to the next rolling horizon.
- At time $t=10$, job arrival information from Rolling Horizon 2 ($t = 11 - 20$) is gathered. The static sequence for jobs from Rolling Horizon 2 together with those unprocessed jobs from the previous rolling horizon is then solved. Based on the static sequence solution obtained, job processing on the machine for $t = 10 - 20$ is carried out. Those jobs which have not been processed after $t = 10$ are carried forward to the next rolling horizon.

- These processes are repeated until the entire lot of incoming jobs is sequenced to be processed into the machine.

5.3 ROLLING HORIZON ALGORITHMS

The following algorithms were developed and tested in solving the formulated dynamic scheduling problems:

1. Rolling Horizon procedure with Branch and Bound with limited job arrival information (RH-BBL)
2. Rolling Horizon procedure with GRASP meta-heuristic (RH-GRASP)
3. Shortest Completion Time (SCT)
4. Rolling Horizon procedure with Branch and Bound with complete job arrival information (RH-BBC)

5.3.1 Rolling Horizon procedure with Branch and Bound with limited job arrival information (RH-BBL)

The solution is obtained by solving the static optimal sequence on each rolling horizon by means of branch and bound algorithms. The total amount of setup time and idle time is used as the lower bound. The details of the algorithm are described as follows:

- Data input:** Processing time p , Setup time s , maintenance time m , Tool life T , Planning Horizon PH , Forecast Window FW
- Step 1:** Initialize $roll = 1$, $makespan = 0$, $R = FW$
- Step 2:** Initialize $Job_list = \emptyset$
- Step 3:** Include those jobs that are carried forward from previous rolling horizon into Job_list
- Step 4:** Include jobs with release time satisfies the following condition into Job_List : $release\ time + p \leq makespan + R - m$
- Step 5:** Solve for static optimal sequence of jobs inside Job_list with branch and bound algorithm (as described in section 3.3.3), then calculate and update makespan of sequence
- Step 6:** Carry forward jobs inside Job_list that has not been processed after R (based on static optimal sequence from Step 4) to the next Rolling Horizon
- Step 7:** $roll = roll + 1$, if $roll \leq \frac{SH}{R}$ go to Step 2
- Step 8:** Stop

In order to clearly describe the above procedure, a simple numerical example is given as follows:

Numerical Example

- Planning horizon = 480;
- Poisson job arrival with arrival rate: $\lambda = 0.075$;
- Maintenance time: $m = 4$;
- Tool life: $T = 16$;
- Forecast Window $FW = 40$
- Re-planning Frequency $RF =$ once every 40

Rolling Horizon 1:

At rolling horizon 1, i.e., $t = 0 - 40$, it is known that there will be 5 jobs arrival, with the following information:

- Job arrival is described in the following Figure 5.4:

Figure 5.4: Job arrival of Rolling Horizon 1 (RH-BBL)

- Processing time: $p_1 = 8$; $p_2 = 5$; $p_3 = 10$; $p_4 = 6$; $p_5 = 4$
- Setup time matrix:

s	J ₁	J ₂	J ₃	J ₄	J ₅
J ₁	-	1	1	4	4
J ₂	3	-	1	2	1
J ₃	2	2	-	4	4
J ₄	2	2	3	-	1
J ₅	2	1	3	4	-

At $t=0$, we made a sequencing / scheduling decision for RH1. Static Optimal Sequencing for Rolling Horizon 1 is obtained with the branch and bound method for the 5 jobs, as shown in the following Figure 5.5:

Figure 5.5: Static Optimal Sequencing of RH1 (RH-BBL)

J_1 is to be processed as soon as it arrives, followed by J_2 and J_3 ; while J_4 and J_5 are carried forward into the subsequent Rolling Horizon RH2. This sequence would result in a total amount of setup and idle time of 9 units of time for RH1.

Rolling Horizon 2:

At Rolling Horizon 2, where $t = 40 - 80$, there is no job arrival to the system. Hence we will only sequence J_4 and J_5 , which are carried forward from RH1.

At $t=40$, we made a decision for RH2 as shown in the figure below, which results in 22 units of setup + idle time, as shown in the following Figure 5.6:

Figure 5.6: Static Optimal Sequencing of RH1 – RH2 (RH-BBL)

Rolling Horizon 3:

At Rolling Horizon 3, where $t = 80 - 120$, another 6 jobs arrive with the following information:

- Job arrival is shown in the following Figure 5.7:

Figure 5.7: Job arrival of Rolling Horizon 3 (RH-BBL)

- Processing time: $p_6 = 3$; $p_7 = 1$; $p_8 = 4$; $p_9 = 2$; $p_{10} = 4$; $p_{11} = 7$
- Setup time matrix:

s	J_6	J_7	J_8	J_9	J_{10}	J_{11}
J_6	-	4	2	1	3	1
J_7	3	-	2	1	3	2
J_8	3	4	-	1	1	3
J_9	4	3	4	-	3	1
J_{10}	3	2	3	2	-	1
J_{11}	1	4	3	3	4	-

Static Optimal Sequencing for Rolling Horizon 3 is shown in the following Figure 5.8:

Figure 5.8: Static Optimal Sequencing of RH1 – RH3 (RH-BBL)

Rolling Horizon 4:

At Rolling Horizon 4, where $t = 120 - 160$, 4 more jobs arrived, and since we have J_{11} from RH3 being carried forward to RH4, we have 5 jobs to be considered:

- Job arrival is shown in the following Figure 5.9:

Figure 5.9: Job arrival of Rolling Horizon 4 (RH-BBL)

- Processing time: $p_{11} = 7$; $p_{12} = 3$; $p_{13} = 5$; $p_{14} = 9$; $p_{15} = 6$
- Setup time matrix:

s	J_{11}	J_{12}	J_{13}	J_{14}	J_{15}
J_{11}	-	4	2	3	4
J_{12}	2	-	3	3	4
J_{13}	2	3	-	4	2
J_{14}	3	3	2	-	2
J_{15}	3	4	2	4	-

Static Optimal Sequencing for Rolling Horizon 4 is shown in the following Figure

5.10:

Figure 5.10: Static Optimal Sequencing of RH1 – RH4 (RH-BBL)

These procedures are continued for the rest of the rolling horizons, RH5, RH6, ...,

RH 12, and the overall solution of sequencing decisions with a makespan value of

469 are obtained. The overall solution is described in the following Figure 5.11:

Figure 5.11: RH-BBL Sequencing Solution

5.3.2 Rolling Horizon procedure with GRASP meta-heuristic (RH-GRASP)

As in RH-BBL, the planning horizon is divided into several rolling horizons. However, instead of applying the branch and bound method, the solution is obtained by the GRASP heuristic procedure to minimize the total amount of setup and idle times of the respective rolling horizon. The details of the algorithm are described as follows:

Data input: Processing time p , Setup time s , maintenance time m , Tool life T , Planning Horizon PH , Forecast Window FW

Step 1: Initialize $roll = 1$, $makespan = 0$, $R = FW$

Step 2: Initialize $Job_list = \{ \}$

Step 3: Include those jobs that are carried forward from previous rolling horizon into Job_list

- Step 4:** Include jobs with release time satisfies the following condition into Job_List: $\text{release time} + p \leq \text{makespan} + R - m$
- Step 5:** Solve for static sequence of jobs inside Job_list with GRASP meta-heuristic (as described in section 3.4.2), then calculate and update makespan of sequence
- Step 6:** Carry forward jobs inside Job_list that has not been processed after R (based on static sequence from Step 4) to the next Rolling Horizon
- Step 7:** $\text{roll} = \text{roll} + 1$, if $\text{roll} \leq \frac{SH}{R}$ go to Step 2
- Step 8:** Stop

5.3.3 Shortest Completion Time (SCT)

The Shortest Completion Time (SCT) Algorithm is used to represent typical sequencing practices being implemented in manufacturing systems. In this algorithm, jobs are sequenced in such a way that the available job (i.e., job has already arrived into the system) with the shortest completion time is to be processed first on the machine. Future job arrival information is not utilized in this algorithm. In a typical testing facility system, sequencing is done manually with no known certain algorithms implemented. Also, low level job sequencing is

performed by only considering the jobs that have already been made available at the testing facility system, with no coordination between the testing department and assembly department, as both work as an independent entity within their manufacturing system.

In order to clearly describe the above procedure, a simple numerical example is given as follows:

Numerical Example

The SCT algorithm is applied to the same numerical example that was previously solved with RH-BBL.

Decision 1 ($t=5$):

The first decision is made once the first job has arrived into the system, as shown in Figure 5.12:

Figure 5.12: First decision point

- Job list = $\{J_1\}$
- Processing time: $p_1 = 8$;

Since J_1 is the only job available for processing during the decision time, it is immediately processed once it arrives into the system; as shown in Figure 5.13:

Figure 5.13: First job sequencing decision

Decision 2 (t=13):

Next decision time comes once J_1 finishes being processed, as shown in Figure 5.14:

Figure 5.14: Second decision point

- Job list = $\{J_2, J_3\}$
- Processing time: $p_2 = 5, p_3 = 10$;
- Setup time: $s_{12} = 1, s_{13} = 1$;

Both J_2 and J_3 cannot be completed before the maintenance period. Hence they can only be processed after the maintenance period. Since J_2 has a shorter processing time, it would also have a shorter completion time. Hence it would be processed next. 1 unit setup time is performed simultaneously with maintenance activity. Figure 5.15 illustrates the sequencing decision of the second job:

Figure 5.15: Second job sequencing decision

Decision 3 ($t=25$):

Figure 5.16 shows the third decision point, just after J_2 is finish processed.

Figure 5.16: Third decision point

- Job list = $\{J_3\}$
- Processing time: $p_3 = 10$;
- Setup time: $s_{23} = 1$;

J_3 is processed after setup activity s_{23} is performed. This third job sequencing decision is illustrated by Figure 5.17:

Figure 5.17: Third job sequencing decision

These procedures continue for the rest of the decision time, and the overall solution of sequencing decisions is obtained as shown in Figure 5.18, with a makespan value of 469.

Figure 5.18: SCT Sequencing Solution

5.3.4 Rolling Horizon procedure with Branch and Bound with complete job arrival information (RH-BBC)

In this algorithm, it is assumed that we have complete knowledge about job arrival information, not only for a certain limited time (forecast window) as in the case of the previous three algorithms above. Due to this assumption, this algorithm is not practical, and the result of this algorithm is only used to assess

the quality of the solutions from the other three algorithms. The result of this algorithm is a good benchmark to assess the other algorithms, as it can be taken as the ideal solution that can be achieved with the given job arrival information.

Solving this problem optimally with algorithms from Problem Part 1 requires an enormous amount of computation time due to a large number of jobs being expected; hence the rolling horizon procedure is incorporated in order to shorten the computation time. This rolling horizon algorithm is similar to the one with limited job arrival information from section 5.3.1, except that the length of the rolling horizon is made flexible, and no carrying forward of the job is performed in order to keep it an ideal solution. The R value is progressively increased from L , $2L$, $3L$, and so on until the value of the static optimal makespan is less than the R value, in which $L = \text{Tool life } T + \text{Maintenance time } m$. The details of the procedure are shown as follows:

Data input: Processing time p , Setup time s , maintenance time m , Planning Horizon PH

Step 1: Initialize $roll = 1$, $R = L$, $makespan = 0$, $span = 0$

Step 2: Initialize $Job_list = \{ \}$

Step 3: Include jobs with release time satisfying the following condition into Job_List : $release\ time + p \leq span + R - m$

- Step 4:** Solve for static optimal sequence of jobs inside Job_list with branch and bound algorithm (as described in section 3.3.3), then calculate makespan of sequence
- Step 5:** If makespan $\leq R$, save solution, span = span+R. Else, $R = R+L$ and go to Step 2
- Step 6:** roll = roll + 1, if roll $\leq \frac{SH}{R}$ go to Step 2
- Step 7:** Stop

In order to clearly describe the above procedure, a simple numerical example is given as follows:

Numerical Example

The RH-BBC algorithm is applied to the same numerical example that is previously solved with RH-BBL. In this algorithm, it is assumed that the following job arrival information, including its respective processing time and setup time, is known beforehand.

- Number of jobs = 37

- Job arrival:

r ₁	r ₂	r ₃	r ₄	r ₅	r ₆	r ₇	r ₈	r ₉	r ₁₀	r ₁₁	r ₁₂	r ₁₃	r ₁₄	r ₁₅	r ₁₆	r ₁₇	r ₁₈	r ₁₉
5	9	13	27	32	84	89	99	104	111	115	122	130	140	146	161	163	174	190

r ₂₀	r ₂₁	r ₂₂	r ₂₃	r ₂₄	r ₂₅	r ₂₆	r ₂₇	r ₂₈	r ₂₉	r ₃₀	r ₃₁	r ₃₂	r ₃₃	r ₃₄	r ₃₅	r ₃₆	r ₃₇
203	223	229	238	285	291	298	300	308	318	322	340	384	385	411	420	426	456

- Processing time:

p ₁	p ₂	p ₃	p ₄	p ₅	p ₆	p ₇	p ₈	p ₉	p ₁₀	p ₁₁	p ₁₂	p ₁₃	p ₁₄	p ₁₅	p ₁₆	p ₁₇	p ₁₈	p ₁₉
8	5	10	6	4	3	1	4	2	4	7	3	5	9	6	2	10	10	2

p ₂₀	p ₂₁	p ₂₂	p ₂₃	p ₂₄	p ₂₅	p ₂₆	p ₂₇	p ₂₈	p ₂₉	p ₃₀	p ₃₁	p ₃₂	p ₃₃	p ₃₄	p ₃₅	p ₃₆	p ₃₇
1	3	7	1	3	2	7	3	4	1	4	6	6	6	7	9	3	9

- Setup time matrix:

Job 1 - 20:

	J ₁	J ₂	J ₃	J ₄	J ₅	J ₆	J ₇	J ₈	J ₉	J ₁₀	J ₁₁	J ₁₂	J ₁₃	J ₁₄	J ₁₅	J ₁₆	J ₁₇	J ₁₈	J ₁₉	J ₂₀
J ₁	-	1	1	4	4	4	4	4	3	4	4	1	2	2	4	2	4	3	1	2
J ₂	3	-	1	2	1	4	1	2	3	2	3	1	4	3	3	4	3	2	4	4
J ₃	2	2	-	4	4	1	2	4	1	1	3	2	4	4	1	4	4	4	1	4
J ₄	2	2	3	-	1	2	3	4	1	1	3	3	2	1	2	3	4	4	1	3
J ₅	2	1	3	4	-	3	1	3	3	1	1	1	1	2	2	3	2	2	4	4
J ₆	1	2	2	1	3	-	4	2	1	3	1	4	4	2	4	3	2	4	1	4
J ₇	3	3	3	3	1	3	-	2	1	3	2	2	2	1	1	3	1	3	4	1
J ₈	3	3	2	3	2	3	4	-	1	1	3	3	3	2	2	4	2	1	1	4
J ₉	4	4	1	3	2	4	3	4	-	3	1	1	2	3	3	3	1	3	3	2
J ₁₀	1	3	1	3	4	3	2	3	2	-	1	3	2	1	3	1	4	3	3	4
J ₁₁	1	1	2	3	2	1	4	3	3	4	-	4	2	3	4	2	2	2	4	4
J ₁₂	1	2	4	3	1	2	2	1	1	4	2	-	3	3	4	4	2	2	2	2
J ₁₃	4	3	3	3	3	2	4	4	2	4	2	3	-	4	2	1	3	4	2	2
J ₁₄	4	2	1	2	2	3	4	1	2	1	3	3	2	-	2	3	1	3	1	3
J ₁₅	2	2	1	3	2	4	4	1	4	2	3	4	2	4	-	3	1	1	3	2
J ₁₆	1	1	1	2	3	2	4	2	2	4	1	2	1	1	4	-	1	4	2	4
J ₁₇	1	2	3	4	2	3	4	3	2	2	1	3	4	4	4	4	-	1	4	3
J ₁₈	1	2	1	3	4	4	4	4	4	2	1	3	1	1	4	3	4	-	2	1
J ₁₉	3	3	4	3	2	4	4	3	4	4	1	1	1	4	4	1	3	4	-	3
J ₂₀	4	4	4	1	3	2	1	4	4	4	1	3	2	4	4	3	4	3	2	-

Job 21 – 37:

	J ₂₁	J ₂₂	J ₂₃	J ₂₄	J ₂₅	J ₂₆	J ₂₇	J ₂₈	J ₂₉	J ₃₀	J ₃₁	J ₃₂	J ₃₃	J ₃₄	J ₃₅	J ₃₆	J ₃₇
J ₂₁	-	4	4	2	1	2	1	3	2	2	4	3	2	3	4	3	4
J ₂₂	3	-	1	1	3	3	4	1	2	3	3	4	1	2	2	4	2
J ₂₃	3	3	-	4	1	4	1	4	4	4	2	3	2	4	1	4	1
J ₂₄	2	2	2	-	1	4	3	2	4	3	1	2	2	3	2	4	2
J ₂₅	1	2	3	1	-	1	3	4	1	3	4	3	4	3	4	1	4
J ₂₆	1	4	1	3	4	-	3	3	4	1	1	2	4	1	2	3	2
J ₂₇	1	2	1	2	1	3	-	1	1	1	1	4	3	4	1	1	3
J ₂₈	2	3	1	4	2	1	4	-	4	4	3	3	4	2	2	4	2
J ₂₉	2	2	2	4	3	3	3	3	-	3	3	2	1	2	2	3	4
J ₃₀	1	4	3	3	4	2	3	1	2	-	3	4	2	2	4	4	3
J ₃₁	4	1	3	3	1	1	2	1	3	1	-	4	1	4	2	1	1
J ₃₂	3	3	4	4	4	1	4	4	1	2	3	-	2	3	2	1	4
J ₃₃	2	2	4	2	4	1	4	1	3	3	3	1	-	1	1	4	1
J ₃₄	2	2	3	4	3	3	2	1	1	2	3	3	1	-	1	4	3
J ₃₅	4	3	1	2	1	3	1	4	1	1	3	1	3	4	-	4	3
J ₃₆	4	1	4	4	2	2	3	1	4	1	3	1	4	2	3	-	3
J ₃₇	4	4	3	4	2	4	3	1	2	2	4	4	3	4	1	2	-

Roll=1:

- $R=L \Rightarrow \text{Job_List} = \{J_1, J_2\}$; since $r_1+p_1 \leq R - \text{maintenance}$ and $r_2+p_2 \leq R - \text{maintenance}$.

Figure 5.19: RH1 Job List

By using a branch and bound algorithm, an optimal sequence of $\{J_1, J_2\}$ is found to minimize makespan, since the makespan value is larger than R , value of R is to be increased, i.e., $R=R+L$.

- $R = 2L \Rightarrow \text{Job_List} = \{J_1, J_2, J_3, J_4, J_5\}$

makespan value is larger than R ; hence value of R is to be increased, i.e., $R=R+L$.

- $R = 3L \Rightarrow \text{Job_List} = \{J_1, J_2, J_3, J_4, J_5\}$

Optimal sequence: $J_1-M-J_2-J_3-M-J_4-J_5-M$; with makespan = R , thus save solution, span = $3L = 60$.

Optimal Solution of roll 1 with $R = 3L = 60$ is shown in Figure 5.20:

Figure 5.20: Optimal Solution of RH1

Roll=2:

- $R=L \Rightarrow \text{Job_list} = \{ \};$

Optimal sequence: M ; save solution, span = span + $L = 80$.

Optimal Solution of roll 2 with $R = L = 20$ is shown in Figure 5.21:

Figure 5.21: Optimal Solution of RH2

Roll=3:

- $R=L \Rightarrow \text{Job_List} = \{J_6, J_7\}$

Optimal sequence: J_6 - J_7 -M; save solution, $\text{span} = \text{span} + L = 100$.

Optimal Solution of roll 3 with $R = L = 20$ is shown in Figure 5.22:

Figure 5.22: Optimal Solution of RH3

Roll = 4 ($R = L$):

Optimal Solution of roll 4 with $R = L = 20$ is shown in Figure 5.23:

Figure 5.23: Optimal Solution of RH4

Roll = 5 ($R = 5L$):

Optimal Solution of roll 3 with $R = 5L = 100$ is shown in Figure 5.24:

Figure 5.24: Optimal Solution of RH5

Optimal solution of RH6 – RH14 are described in Figure 5.25 – 5.33:

Roll = 6:

Figure 5.25: Optimal Solution of RH6

Roll = 7:

Figure 5.26: Optimal Solution of RH7

Roll = 8:

Figure 5.27: Optimal Solution of RH8

Roll = 9:

Figure 5.28: Optimal Solution of RH9

Roll = 10:

Figure 5.29: Optimal Solution of RH10

Roll = 11:

Figure 5.30: Optimal Solution of RH11

Roll = 12:

Figure 5.31: Optimal Solution of RH12

Roll = 13:

Figure 5.32: Optimal Solution of RH13

Roll = 14:

Figure 5.33: Optimal Solution of RH14

In this problem instance, the global optimal makespan value is equal to the local optimal makespan value, i.e., makespan = 469.

In order to compare the performance of the four algorithms developed in this section, they were coded in Matlab and tested on an Intel Core™ 2 CPU with

3GHz and 3292MB RAM. The design of experiment, more problem instances, and the result of the experiments are discussed in Section 5.4.

5.4 COMPUTATIONAL EXPERIMENT

5.4.1 Design of Experiment

The processing time of jobs and setup time matrix were generated randomly from an integer uniform distribution on (1, 10), and (1, 4) respectively, while the maintenance time is fixed at $m=4$ and tool life is $T=16$. Job arrival data is generated based on the Poisson arrival. In order to test the effect of the range of arrival rate, three values of arrival rate λ are tested: $\lambda = 0.05$ (equivalent with ~50% non-idle utilization), $\lambda = 0.075$ (equivalent with ~62% non-idle utilization), and $\lambda = 0.1$ (equivalent with ~71% non-idle utilization) with a planning horizon of 480 unit time. From each arrival rate condition, 30 random instances are generated, and each problem instance is solved with the following algorithms:

- i. Rolling Horizon procedure with Branch and Bound with limited job arrival information (RH-BBL)
- ii. Rolling Horizon procedure with GRASP meta-heuristic (RH-GRASP)
- iii. Shortest Completion Time (SCT)
- iv. Rolling Horizon procedure with Branch and Bound with complete job arrival information (RH-BBC)

5.4.2 Results Analysis

The overall result of the experiment is summarized in Table 5.1. A comparison of solution quality among algorithms is performed in terms of its closeness to the ideal solution obtained from RH-BBC. Delta value and Delta percentage are used as the performance measure in this case, which is calculated with the following formulas:

- Delta value = heuristic makespan – ideal makespan (5.1)

- Delta percentage = $\frac{\text{Delta value}}{\text{ideal_makespan}} \times 100\%$ (5.2)

Results of the computational experiment are summarized in Table 5.1, with the following observations made:

- Very low delta percentage from RH-BBL solution with only up to average of 0.03% from 30 random instances of Poisson job arrival with $\lambda = 0.1$, while considerably higher delta percentage was observed from SCT solution with 3.61% of average delta percentage.
- RH-GRASP solution performance was in between SCT solution and RH-BBL solution as shown in Figure 5.34. It was also observed that a more distinct delta percentage difference between SCT and the other two algorithms was expected for a higher arrival rate. The Computational Experiment with a higher arrival rate is outlined in Chapter 6, making use

of historical data from a real semiconductor testing facility system and machine utilization as performance measures.

Figure 5.34: Delta Percentage Comparison (Makespan minimization)

Further analysis of local optimal solutions is performed by varying the width of the forecast window on RH-BBL, i.e., FW = 20, 40, 60, and 80; with results of 30 random instances as shown in the following Table 5.2:

λ	FW = 20									
	Non-production time		Idle time		CPU time		Delta value		Delta percentage	
	Mean value	Max value	Mean value	Max value	Mean value	Max value	Mean value	Max value	Mean value	Max value
0.05	350.30	419.00	241.13	312.00	0.20	0.52	0.00	0.00	0.00%	0.00%
0.075	307.00	352.00	185.83	237.00	0.38	4.72	0.27	5.00	0.05%	0.98%
0.1	270.90	312.00	138.07	188.00	0.35	1.34	0.33	3.00	0.07%	0.59%

λ	FW = 40									
	Non-production time		Idle time		CPU time		Delta value		Delta percentage	
	Mean value	Max value	Mean value	Max value	Mean value	Max value	Mean value	Max value	Mean value	Max value
0.05	350.30	419.00	240.90	317.00	0.23	1.20	0.00	0.00	0.00%	0.00%
0.075	306.83	352.00	183.07	234.00	0.87	14.30	0.10	3.00	0.02%	0.65%
0.1	270.70	312.00	137.50	188.00	7.83	199.25	0.13	3.00	0.03%	0.59%

λ	FW = 60									
	Non-production time		Idle time		CPU time		Delta value		Delta percentage	
	Mean value	Max value	Mean value	Max value	Mean value	Max value	Mean value	Max value	Mean value	Max value
0.05	350.30	419.00	242.17	318.00	1.74	17.97	0.00	0.00	0.00%	0.00%
0.075	306.73	352.00	186.00	240.00	4.16	49.59	0.00	0.00	0.00%	0.00%
0.1	270.70	312.00	138.63	186.00	15.12	213.08	0.13	3.00	0.03%	0.59%

λ	FW = 80									
	Non-production time		Idle time		CPU time		Delta value		Delta percentage	
	Mean value	Max value	Mean value	Max value	Mean value	Max value	Mean value	Max value	Mean value	Max value
0.05	350.30	419.00	242.33	316.00	10,852.06	324,198.39	0.00	0.00	0.00%	0.00%
0.075	306.73	352.00	185.87	240.00	556.24	14,249.52	0.00	0.00	0.00%	0.00%
0.1	270.60	309.00	138.17	183.00	2,323.59	35,489.41	0.03	1.00	0.01%	0.20%

Table 5.2: Forecast Window Computational Experiment Results on RH-BBL

Although it's not always the case, we can observe that RH-BBL has a tendency to perform better with a wider forecast window (as shown in Figure 5.35); which can hypothetically be explained due to the fact that more information is included into the procedure. However, in this experiment, it is hard to gauge the significance of extending the forecast window length due to the small delta percentage value of the algorithm. A similar experiment was performed in Chapter 6 in order to evaluate its significance further, based on historical data of an actual manufacturing system.

Another aspect to be considered is that the forecast window width is dependent on the ability of the manufacturing system to provide job arrival information. A wider forecast window would require the system to have an extensive look-ahead capability. Also, the width of a forecast window will affect the frequency of scheduling activity on the system. The narrower forecast window implies that more frequent scheduling activity needs to be performed than on the wider one.

Figure 5.35: Delta Percentage across Forecast Window on RH-BBL

5.5 ANALYSIS ON MINIMIZING AVERAGE OF THE JOB FLOW TIMES

In this section, similar analysis with a different objective function is performed. While the previous study was done with a focus on minimizing makespan, the average of the job flow times is the objective function to be minimized in this section. Similar sets of rolling horizon algorithms, i.e. RH-BBC, RH-BBL, and RH-GRASP were modified and tested to solve the problem; and the SCT algorithm is still the benchmark to depict typical scheduling practice of an actual manufacturing system. The lower bounds of the RH-BBC and RH-BBL algorithms are modified to cater for the new objective function. As for RH-GRASP, we only need to modify the algorithm's objective function calculation procedure due to the combinatory nature of the algorithm.

5.5.1 Modified Lower bound

The modified lower bound for both the RH-BBC and RH-BBL to cater for the objective function of minimizing average of the job flow times are developed as follows:

Let PS be a partial sequence in which k jobs are scheduled; and US be the set of unscheduled jobs. For each unscheduled job i , in which $k+1 \leq i \leq n$, there are two cases to be considered, which are:

1. If $r_{[k+1]} \geq C_{\max}(\text{PS})$:

The sum of the completion time of the first $(k+1)$ job can be calculated as:

$$\sum_{l=1}^{k+1} C_l = \sum_{l=1}^k C_l + s_{[k][k+1]} + p_{[k+1]} \geq \sum_{l=1}^k C_l + p_{[k+1]}$$

Hence, the sum of the completion time of the first $(k+i)$ job is:

$$\sum_{l=1}^{k+i} C_l \geq \sum_{l=1}^k C_l + \sum_{l=1}^i p_{[k+1]}$$

Lower Bound:

$$\text{LB} = \sum_{l=1}^k C_l + \sum_{x \in \text{US}} p_{[x]} \quad (5.3)$$

2. If $r_{[k+1]} < C_{\max}(\text{PS})$:

The sum of the completion time of the first $(k+1)$ job can be calculated as:

$$\sum_{l=1}^{k+1} C_l = \sum_{l=1}^k C_l + C_{\max}(\text{PS}) + s_{[k][k+1]} + p_{[k+1]} - r_{[k+1]}$$

Hence, the sum of the completion time of the first $(k+i)$ job is:

$$\sum_{l=1}^{k+i} C_l \geq \sum_{l=1}^k C_l + C_{\max}(\text{PS}) + \sum_{l=1}^i (s_{[k][k+1]} + p_{[k+1]} - r_{[k+1]})$$

Lower Bound:

$$\text{LB} = \sum_{l=1}^k C_l + C_{\max}(\text{PS}) + \sum_{x \in \text{US}} (s_{[k][x]} + p_{[x]} - r_{[x]}) \quad (5.4)$$

The term $s_{[k][x]} + p_{[x]} - r_{[x]}$ on the lower bound formula above represents the minimum completion time contribution of each unscheduled job x . This minimum completion time contribution can be obtained by assuming that job x is sequenced just after job k .

Numerical Example of Branch and Bound Procedure with Modified Lower Bound

In order to describe the modified lower bound more clearly, the following simple numerical example is solved with a branch and bound procedure:

- Number of Jobs: $n = 3$
- Tool life: $T = 16$
- Maintenance time: $m = 4$
- Processing time: $p_1 = 9; p_2 = 7; p_3 = 4$
- Release time: $r_1 = 11; r_2 = 17; r_3 = 30$
- Setup time matrix:

s	J ₁	J ₂	J ₃
J ₁	-	1	2
J ₂	4	-	4
J ₃	4	3	-

The branch and bound procedure is described in Figure 5.36, in which node 1, 2, 3, 0-1-2, 0-2-1, 0-1-3-2, and 0-2-3-1 are pruned due to infeasibility (tool life constraint). Feasible solution is obtained on node 0-1-3-0-2 with lower bound value of 53; hence we obtain upper bound value of 53 and the rest of the nodes

with lower bound value higher than 53 are pruned, i.e. node 0-3,0-1-0, 0-2-0, and 0-2-3-0. Hence an optimal sequence is 0-1-3-0-2 with a sum of the job flow times of 53.

Figure 5.36: Example of Branch & Bound Procedure with Modified Lower Bound

The following are examples of lower bound computation for several nodes shown above:

Lower Bound computation for node 0-1:

- Partial Sequence PS = 0-1

$$C_{\max}(\text{PS}) = 20 + 9 = 29$$

- Job 1 is completed at $t=29$, hence job 1 flow time:

$$C_1 = 29 - r_1 = 29 - 11 = 18$$

- Since $r_2 < C_{\max}$ and $r_3 \geq C_{\max}$:

$$\Rightarrow LB = C_1 + (C_{\max} + s_{12} + p_2 - r_2) + (p_3)$$

$$\Leftrightarrow LB = 18 + (29 + 1 + 7 - 17) + 4 = 42$$

Lower Bound computation for node 0:

- Partial Sequence PS = 0

$$C_{\max}(PS) = 20$$

- Since $r_1, r_2 < C_{\max}$ and $r_3 \geq C_{\max}$:

$$\Rightarrow LB = (C_{\max} + s_{01} + p_1 - r_1) + (C_{\max} + s_{02} + p_2 - r_2) + (p_3)$$

$$\Leftrightarrow LB = (20 + 9 - 11) + (20 + 7 - 17) + 4 = 32$$

Lower Bound computation for node 0-1-3-0-2:

- Job 1 is completed at $t=29$, hence job 1 flow time:

$$C_1 = 29 - r_1 = 29 - 11 = 18$$

Job 3 is completed at $t=35$, hence job 3 flow time:

$$C_3 = 35 - r_3 = 35 - 30 = 5$$

Job 2 is completed at $t=47$, hence job 2 flow time:

$$C_2 = 47 - r_2 = 47 - 17 = 30$$

- Hence, lower bound:

$$\Rightarrow LB = 18 + 5 + 30 = 53$$

5.5.2 Computational Experiment

In order to compare the performance of the algorithms, they were coded in Matlab and tested on an Intel Core™ 2 CPU with 3GHz and 3292MB RAM.

5.5.2.1 Design of Experiment

The same set of data from previous analysis on minimizing makespan is used in this experiment. The details of the experimental design are described in section 5.4.1. 30 random instances were generated for each experimental condition, and each of the problem instances was solved with the following algorithms:

- Rolling Horizon procedure with Branch and Bound with limited job arrival information with modified lower bound (MRH-BBL)
- Rolling Horizon procedure with GRASP meta-heuristic with modified objective function (MRH-GRASP)
- Shortest Completion Time (SCT)

- iv. Rolling Horizon procedure with Branch and Bound with complete job arrival information with modified lower bound (MRH-BBC)

5.5.2.2 Results Analysis

The overall result of the experiment is summarized in Table 5.3. Comparison of solution quality among algorithms was performed in terms of its closeness to the ideal solution obtained from MRH-BBC. The Delta value and Delta percentage were used as the performance measures in this case, which is calculated with the following formulas:

- $\text{flowtime average} = (\sum_{i=1}^n C_i)/n$ (5.5)

- $\text{Delta value} = \text{heuristic flowtime average} - \text{ideal flowtime average}$ (5.6)

- $\text{Delta percentage} = \frac{\text{Delta value}}{\text{ideal flowtime average}} \times 100\%$ (5.7)

λ	MRH-BBC										
	Number of jobs		Avg of processing time		Average of jobs flow time			CPU time			No. of Optimal solution
	Mean value	Max value	Mean value	Max value	Mean value	Max value	Max value	Mean value	Max value	Max value	
0.05	24.07	32.00	5.44	6.63	12.16	17.50		2.61	38.72		
0.075	33.77	44.00	5.44	6.18	14.56	21.52		163.18	2,996.90		
0.1	42.57	50.00	5.12	5.90	15.34	20.00		6,983.54	93,720.75		
λ	MRH-BBL (Forecast Window = 40)										
	Number of jobs		Avg of processing time		Average of jobs flow time			CPU time			No. of Optimal solution
	Mean value	Max value	Mean value	Max value	Mean value	Max value	Max value	Mean value	Max value	Max value	
0.05	24.07	32.00	5.44	6.63	12.29	17.62		0.50	2.48	1.93	18 out of 30
0.075	33.77	44.00	5.44	6.18	14.77	21.58		1.41	6.25	1.35	16 out of 30
0.1	42.57	50.00	5.12	5.90	16.21	22.24		5.16	30.53	3.47	6 out of 30
λ	MRH-GRASP										
	Number of jobs		Avg of processing time		Average of jobs flow time			CPU time			No. of Optimal solution
	Mean value	Max value	Mean value	Max value	Mean value	Max value	Max value	Mean value	Max value	Max value	
0.05	24.07	32.00	5.44	6.63	13.17	17.88		0.01	0.02	3.71	6 out of 30
0.075	33.77	44.00	5.44	6.18	15.89	25.85		0.01	0.03	8.39	0 out of 30
0.1	42.57	50.00	5.12	5.90	17.60	23.06		0.03	0.06	5.05	0 out of 30
λ	SCT										
	Number of jobs		Avg of processing time		Average of jobs flow time			CPU time			No. of Optimal solution
	Mean value	Max value	Mean value	Max value	Mean value	Max value	Max value	Mean value	Max value	Max value	
0.05	24.07	32.00	5.44	6.63	13.95	20.35		0.01	0.02	6.87	2 out of 30
0.075	33.77	44.00	5.44	6.18	18.59	36.12		0.01	0.03	14.61	0 out of 30
0.1	42.57	50.00	5.12	5.90	23.08	57.63		0.01	0.03	38.86	0 out of 30

Table 5.3: Part 3 Computational Experiment Results (Average of The Job Flow Times minimization)

Results of the computational experiment are summarized in Table 5.3, with the following observations are made:

- Processing time averages at ~5 units of time; while the average of the job flow times ranges from ~12 – 23 units of time, depending on arrival rate and the sequencing algorithm applied. The excess time is due to setup time, maintenance time, and queuing time on the system.
- The considerably low delta percentage from the MRH-BBL solution with an average of only up to 5.46% from 30 random instances of an arrival rate of 0.1 (equivalent with 71% non-idle utilization), while 48.09% of average delta was observed from the SCT solution; which showed that there is a lot of room for improvement from the current scheduling practice.
- The MRH-GRASP solution performance is in between the SCT solution and the MRH-BBL solution as shown in Figure 5.37, and the CPU time of MRH-GRASP is much lower than MRH-BBL.
- It was also observed that a more distinct delta percentage difference between SCT and the other two algorithms was expected for a higher arrival rate, as higher arrival rate would create more room for improvement to the system. A Computational Experiment with a higher arrival rate is presented in Chapter 6, making use of historical data from a real semiconductor testing facility system and WIP average as performance measures.

Figure 5.37: Delta Percentage Comparison (average of the job flow times minimization)

A further analysis of the local optimal solution was performed by varying the width of the forecast window on the MRH-BBL, so that $FW = 20, 40, 60$, and 80 . Similar trends as the one with objective function of makespan minimization (as described in section 5.4.2) were observed, as shown in Table 5.4 and Figure 5.58.

λ	FW = 20									
	Avg of processing time		Avg of jobs flow time		CPU time		Delta value		Delta percentage	
	Mean value	Max value	Mean value	Max value	Mean value	Max value	Mean value	Max value	Mean value	Max value
0.05	5.44	6.63	12.30	17.62	0.50	2.63	0.13	1.93	1.05%	14.52%
0.075	5.44	6.18	14.94	21.58	0.76	3.78	0.37	1.98	2.47%	11.37%
0.1	5.12	5.90	16.38	21.02	4.35	62.19	1.05	4.07	7.09%	35.31%
λ	FW = 40									
	Avg of processing time		Avg of jobs flow time		CPU time		Delta value		Delta percentage	
	Mean value	Max value	Mean value	Max value	Mean value	Max value	Mean value	Max value	Mean value	Max value
0.05	5.44	6.63	12.29	17.62	0.50	2.48	0.12	1.93	0.96%	14.52%
0.075	5.44	6.18	14.77	21.58	1.41	6.25	0.20	1.35	1.19%	5.26%
0.1	5.12	5.90	16.21	22.24	5.16	30.53	0.87	3.47	5.46%	18.48%
λ	FW = 60									
	Avg of processing time		Avg of jobs flow time		CPU time		Delta value		Delta percentage	
	Mean value	Max value	Mean value	Max value	Mean value	Max value	Mean value	Max value	Mean value	Max value
0.05	5.44	6.63	12.27	17.50	1.14	7.00	0.11	1.89	0.42%	7.33%
0.075	5.44	6.18	14.68	21.58	5.64	48.94	0.12	1.29	0.78%	6.85%
0.1	5.12	5.90	15.82	22.04	25.31	249.34	0.46	3.27	3.03%	19.13%
λ	FW = 80									
	Avg of processing time		Avg of jobs flow time		CPU time		Delta value		Delta percentage	
	Mean value	Max value	Mean value	Max value	Mean value	Max value	Mean value	Max value	Mean value	Max value
0.05	5.44	6.63	12.19	17.62	3.42	70.33	0.03	0.27	0.25%	3.51%
0.075	5.44	6.18	14.66	21.55	10.37	62.10	0.10	1.32	0.58%	7.02%
0.1	5.12	5.90	15.78	20.47	87.22	658.34	0.45	2.64	2.79%	15.12%

Table 5.4: Forecast Window Computational Experiment Results on MRH-BBL

Figure 5.38: Delta Percentage across Forecast Window on MRH-BBL

5.6 SUMMARY

In this chapter, we discussed a sequence-dependent setup scheduling problem with fixed maintenance consideration and deterministic dynamic job arrival assumption to minimize makespan on a single machine system. Specific assumptions based on the Rolling Horizon model were made, in which job arrival information for a certain limited time (Forecast Window) is known. Based on this limited information, sequencing/scheduling decision is made periodically, for once every time a batch of job arrival information is gathered. With that, 4 rolling horizon algorithms were developed with various purposes. The RH-BBC algorithm is considered as a non-practical procedure, which provides an ideal solution that can be obtained with certain job arrival information given. The RH-BBL algorithm and RH-GRASP algorithm are two practical heuristic algorithms

which are able to provide solutions close to the RH-BBC ideal solution, and the SCT algorithm is used to represent a typical sequencing procedure practice from a manufacturing system.

The same analysis was also performed on a problem with an objective function of minimizing average of the job flow times with a similar trend of result is observed. In the next chapter, these algorithms are tested on the historical data of an actual testing facility manufacturing system for a case study.

6. INDUSTRIAL APPLICATION: CASE STUDY

The rolling horizon procedure is incorporated with the developed heuristic algorithms, the branch and bound heuristic and the GRASP meta-heuristics to solve a dynamic setup scheduling problem based on an actual industrial case problem from a semiconductor testing facility manufacturing system. This chapter begins by briefly summarizing the proposed policies to be applied on the testing facility model. The testing facility model is based on the historical data of a STATS ChipPAC manufacturing system. The case study model, data, and experimentation are summarized in Section 6.2 while the results and discussion are presented in Section 6.3.

6.1 PROPOSED POLICIES

The dynamic scheduling problem of the testing facility was solved with a rolling horizon framework previously discussed in Chapter 5. The dynamic scheduling problem was decomposed into a series of static problems, which were then solved by certain heuristic algorithms. In this case, the branch and bound heuristic and GRASP meta-heuristic were applied to solve the static problems, which are

deciding the sequence among processing jobs (includes determining which job), and performing setup, and / or maintenance activity on the machine. By solving the decomposed problem period by period, only limited job arrival information per period was required. Finally, a rolling schedule was obtained to determine the sequencing and scheduling of machine activity of each period. One period after another, the multi-period model was updated with the unfinished job from the previous period, and job arrival information of the subsequent period. The process was repeated until the problem was resolved.

6.2 CASE STUDY MODEL, DATA, AND EXPERIMENTATION

STATS ChipPAC is a contract manufacturer for semiconductor assembly and testing with various types of products. It serves various customers in the dynamic nature of the semiconductor industry. In this case study, the proposed scheduling policy described in section 6.1 was applied to solve the dynamic scheduling problem of a single tester machine which was used to process the PS800 platform of STATS ChipPAC testing facility. The objective function of concern was to maximize machine utilization at the moment of analysis. PS800 has high demand, and hence maximum utilization of the machine is crucial for the manufacturing system's performance.

The problem formulation of the case study is described as follows:

- The tester facility model is reduced and modified from the STATS ChipPAC actual testing facility model.
- Job arrival information for the next few days of several products is known with perfect accuracy as jobs are originated from the STATS ChipPAC assembly facility (70% of STATS ChipPAC test flow comes from its own assembly facility, as shown in Figure 6.1).

Figure 6.1: IC Manufacturing Process Flow

- Analysis is limited within a finite time horizon.
- Single machine system analysis.
- Sequence-dependent setup activity is required during the operation, primarily to adjust temperature or change handler, depending on the product to be tested.
- Scheduled preventive maintenance is considered while unscheduled maintenance (random breakdown) is not modeled.

- Setup and scheduled maintenance can be performed simultaneously, e.g., temperature adjustment (setup activity) can be performed while performing height calibration (maintenance activity).
- Once started, jobs cannot be interrupted (non-resumable assumption).
- Rework and priority of lots is not considered.

The experimental design based on historical data is summarized in Table 6.1:

Variables / Parameters	Distribution / Values
Processing time	U[1, 8] hours (integer): <ul style="list-style-type: none">• Processing time = (test time + index time) x (lot size)• Test time + index time is averaged at 3 seconds - calculated based on die release historical data, i.e., test time + index time = die release hour / die release quantity)• Lot size is estimated to vary between 1000 and 10000 units, based on PS800 job release historical data
Setup time	Sequence dependent setup time of U[1, 3] hours - estimated based on technician remarks on machine history archive
Preventive Maintenance	Machine is subjected to 3 hours maintenance time, once for each 17 hours tool life: <ul style="list-style-type: none">• 3 hours maintenance time is calculated from average of machine downtime - based on technician remarks on machine history archive• 17 hours tool life is calculated from estimated maintenance time and downtime machine utilization, i.e., tool life = downtime hour each day (maintenance time*24) - based on OPE historical data
Job arrival	Poisson distribution: <ul style="list-style-type: none">• Various arrival rates are to be evaluated, i.e., $\lambda = 0.05, 0.075, 0.1, 0.125, \text{ and } 0.15$• Information of job arrival is limited up to a finite planning horizon of 480 hours

Table 6.1: Experimental Design

6.3 RESULTS AND DISCUSSION

6.3.1 Utilization Performance Comparison

30 random instances were randomly generated and tested for various arrival rate values ($\lambda = 0.05, 0.075, 0.1, 0.125, \text{ and } 0.15$) to evaluate the utilization performance of the following heuristic algorithms:

- i. Rolling Horizon Procedure with Branch and Bound with limited job arrival information (RH-BBL)
- ii. Rolling Horizon Procedure with GRASP meta-heuristic (RH-GRASP)

The utilization performances of the above algorithms were compared with the Shortest Completion Time (SCT) algorithm, which depicts the typical sequencing practice being implemented in manufacturing systems. In the STATS ChipPAC testing facility, sequencing is done manually with no specific algorithm while low level job sequencing is performed only by considering the jobs that are already available at the testing facility system, and no coordination work is performed between the testing facility and its immediate upstream, the assembly facility.

It is assumed that the STATS ChipPAC testing facility is able to obtain job arrival information with perfect accuracy at 40 hours in advance from their assembly facility. Hence, in this experiment the forecast window (FW) is set at 40 hours. Job arrival information from the assembly facility is only given to the testing facility once every 40 hours. The re-planning frequency (RF) is set at one time for every forecast window of 40 hours.

Machine utilization is classified into three types, as standard performance measurements of the algorithm evaluation process:

- 1) *Productive utilization* includes the amount of machine time being used for production (job processing).
- 2) *Non-productive utilization* includes the amount of machine time being used for setup activity plus the amount of total machine idle time.
- 3) *Downtime utilization* includes the amount of machine non-availability, which in this case solely due to scheduled maintenance. Although it is not available for production, setup activity can still be performed during scheduled maintenance. In this calculation, when setup and maintenance are being performed simultaneously, that portion of time is considered as downtime utilization.

The computational experiment is performed by compiling and solving the three algorithms in Matlab on, and the results are based on 30 random instances of each condition obtained; as shown in the following Table 6.2:

λ	RH-BBL				RH-GRASP				SCT		
	Utilization (Mean)			CPU time (s)	Utilization (Mean)			CPU time (s)	Utilization (Mean)		
	Prod.	Non- prod.	Down- time		Prod.	Non- prod.	Down- time		Prod.	Non- prod.	Down- time
0.05	23.22%	62.12%	14.66%	0.19	23.13%	62.11%	14.76%	0.25	23.14%	62.17%	14.68%
0.075	34.47%	50.87%	14.66%	0.83	34.33%	50.90%	14.76%	2.35	34.24%	51.08%	14.68%
0.1	45.33%	40.00%	14.67%	171.76	45.19%	40.10%	14.71%	11.32	44.47%	40.89%	14.64%
0.125	56.51%	28.80%	14.68%	414.64	56.16%	29.10%	14.74%	20.06	53.23%	32.06%	14.71%
0.15	60.90%	24.41%	14.69%	1,008.74	59.89%	25.35%	14.75%	95.14	54.14%	31.19%	14.67%

Table 6.2: Utilization Performance Computational Experiment Result

Figure 6.2, Figure 6.3, and Figure 6.4 show the comparison of Productive Utilization, Non-productive Utilization, and Downtime Utilization respectively.

Figure 6.2: Productive Utilization Comparison

Figure 6.3: Non-productive Utilization Comparison

Figure 6.4: Downtime Utilization Comparison

From Figure 6.2, it can be seen that the RH-BBL and RH-GRASP always give a better productive utilization to the single machine system. At a low arrival rate, no significant utilization difference is noted between the two evaluated algorithms and SCT. There is a 0.01% average delta between the RH-GRASP and SCT, and a 0.08% average delta between the RH-BBL and SCT at an arrival rate of 0.05. However, at higher intensity, the significance of the evaluated algorithms in improving machine utilization is more apparent. A 5.75% average delta between the RH-GRASP and SCT, and a 6.76% average delta between the RH-BBL and SCT at an arrival rate of 0.15.

Based on the OPE historical data of STATS ChipPAC, its testing facility runs with ~50% productive utilization on average. This level of productive utilization is more or less equivalent with $\lambda = 0.125$ and $\lambda = 0.15$ in our computational

experiment, in which the RH-GRASP gives 2.93 - 5.75% and the RH-BBL gives 3.26 – 6.78% better utilization compared to the typical sequencing method of the SCT.

A contrasting trend was observed on non-productive utilization (Figure 6.3), while there was no noticeable difference in downtime utilization (Figure 6.4) as maintenance activity is more or less fixed to be performed only once every tool life T .

The comparison between the RH-BBL and RH-GRASP can be seen from the delta chart on the secondary axis of Figures 6.2 to 6.4. The RH-BBL gives a higher utilization improvement than the RH-GRASP, in which a higher arrival rate would further increase the improvement difference. However, in terms of computation time, the RH-BBL is more costly and not feasible for a situation with a very high arrival rate. Figure 6.5 shows the computation time comparison between the two evaluated algorithms:

Figure 6.5: CPU time comparison between RH-BBL and RH-GRASP

6.3.2 WIP Average Performance Comparison

The same sets of data from section 6.3.1 were used to evaluate the utilization performance of the following heuristic algorithms:

- i. Rolling Horizon Procedure with Branch and Bound with limited job arrival information with modified lower bound (MRH-BBL)
- ii. Rolling Horizon Procedure with GRASP meta-heuristic with modified objective function (MRH-GRASP)

The WIP average performances of the above algorithms were also compared with Shortest Completion Time (SCT) algorithm, which depicts the typical sequencing practices being implemented in manufacturing systems. It is assumed that the STATS ChipPAC testing facility is able to obtain job arrival information with perfect accuracy at 40 hours in advance from their assembly facility. Hence, in this experiment, the forecast window (FW) is set at 40 hours. Job arrival information from assembly facility is only given to the testing facility once every 40 hours. The re-planning frequency (RF) is set at once for every forecast window of 40 hours.

The value of the WIP average is calculated with the following formula:

$$\text{WIP average} = \frac{\text{sum of jobs flowtimes}}{\text{makespan}} = \frac{\sum_{i=1}^n C_i}{C_{\max}} \quad (5.8)$$

The computational experiment was performed by compiling and solving the three algorithms in Matlab, and the following results based on 30 random instances of each condition were obtained; as shown in the following Table 6.3:

λ	MRH-BBL		MRH-GRASP		SCT
	Average WIP	CPU time (s)	Average WIP	CPU time (s)	Average WIP
0.05	0.43	0.22	0.46	0.11	0.47
0.075	0.76	1.58	0.83	0.44	0.87
0.1	1.27	505.57	1.39	1.19	1.64
0.125	2.02	842.33	2.36	3.26	3.36
0.15	2.59	16,483.79	3.01	6.03	4.59

Table 6.3: WIP Average Performance Computational Experiment Result

Figure 6.6: WIP Average Comparison

From Figure 6.6, it can be seen that the MRH-BBL and MRH-GRASP always give a better WIP average performance on the single machine system. At a low arrival rate, no significant WIP average difference can be found between the two evaluated algorithms and SCT. There was no average delta between the MRH-

GRASP and SCT, and a 0.04 average delta between the MRH-BBL and SCT at an arrival rate of 0.05. However, at higher intensity, the significance of the evaluated algorithms in improving WIP average performance is more apparent. There is a 1.58 average delta between the RH-GRASP and SCT, and a 2.00 average delta between the RH-BBL and SCT at an arrival rate of 0.15.

Based on the OPE historical data of STATS ChipPAC, its testing facility runs with ~50% productive utilization, on average. This level of productive utilization is more or less equivalent to $\lambda = 0.125$ and $\lambda = 0.15$ in our computational experiment, in which the MRH-GRASP gives 1.01 – 1.58 (30.05 – 34.42%) and the MRH-BBL gives a 1.34 – 2.00 (39.88 – 43.575%) lower WIP average compared to the typical sequencing method of the SCT.

6.3.3 Effect of Forecast Window

In principle, extending the forecast window by including additional future periods will provide some incremental benefits simply because a greater amount of relevant information is brought into the analysis. At the same time, enlarging it incurs some increased computation time, and more importantly, a greater amount of coordination with an assembly facility system is required in order to gain more extensive job arrival information. To evaluate the significance of the forecast window effect, 30 random instances were generated and tested for each

combination of various arrival rates ($\lambda = 0.05, 0.075, 0.1$, and 0.125) and forecast windows ($FW = 20, 40$, and 60) on the two evaluated algorithms.

A computational experiment was also performed with the same computer specification. Based on the experimental 30 random instances for each of the various combinations, the following results on Table 6.4 and Table 6.5 were obtained:

λ	FW	RH-BBL				RH-GRASP			
		Utilization (Mean)			CPU time (s)	Utilization (Mean)			CPU time (s)
		Prod.	Non- prod.	Down- time		Prod.	Non- prod.	Down- time	
0.05	20	23.19%	62.12%	14.69%	0.16	23.13%	62.11%	14.76%	0.06
	40	23.22%	62.12%	14.66%	0.19	23.13%	62.11%	14.76%	0.25
	60	23.22%	62.12%	14.66%	1.15	23.13%	62.11%	14.76%	0.72
0.075	20	34.47%	50.87%	14.66%	0.24	34.33%	50.90%	14.76%	0.46
	40	34.47%	50.87%	14.66%	0.83	34.33%	50.90%	14.76%	2.35
	60	34.47%	50.87%	14.66%	122.75	34.34%	50.90%	14.76%	6.87
0.1	20	45.33%	40.00%	14.67%	0.33	45.19%	40.10%	14.71%	1.72
	40	45.33%	40.00%	14.67%	171.76	45.19%	40.10%	14.71%	11.32
	60	45.33%	40.00%	14.67%	661.54	45.18%	40.11%	14.71%	24.07
0.125	20	56.47%	29.03%	14.75%	2.26	56.21%	29.03%	14.75%	8.96
	40	56.51%	29.10%	14.74%	414.64	56.16%	29.10%	14.74%	20.06
	60	56.06%	29.20%	14.74%	10,294.09	56.06%	29.20%	14.74%	137.21

Table 6.4: Forecast Window Computational Experiment Result (Makespan minimization)

λ	FW	MRH-BBL		MRH-GRASP	
		Average WIP	CPU time (s)	Average WIP	CPU time (s)
0.05	20	0.43	0.34	0.47	0.06
	40	0.43	0.22	0.46	0.11
	60	0.40	0.21	0.46	0.18
0.075	20	0.77	1.25	0.85	0.21
	40	0.76	1.58	0.83	0.44
	60	0.76	4.50	0.82	0.74
0.1	20	1.28	46.80	1.41	0.64
	40	1.27	505.57	1.39	1.19
	60	1.26	624.25	1.38	2.37
0.125	20	2.09	52.16	2.31	1.83
	40	2.02	842.33	2.36	3.26
	60	2.01	2,585.31	2.34	5.37

Table 6.5: Forecast Window Computational Experiment Result (WIP average minimization)

Figure 6.7 and Figure 6.8 show the forecast window effect on the productive utilization of the RH-BBL and RH-GRASP respectively, while Figure 6.9 and Figure 6.10 show the forecast window effect on the WIP average of the MRH-BBL and MRH-GRASP:

Figure 6.7: Forecast window effect on RH-BBL Productive utilization

Figure 6.8: Forecast window effect on RH-GRASP Productive utilization

Figure 6.9: Forecast window effect on MRH-BBL WIP Average

Figure 6.10: Forecast window effect on MRH-GRASP WIP Average

It is observed that an extension of the forecast window has no significant effect on the utilization improvement of the machine. This implies that job arrival information with an extensive horizon and lengthy computation time are not crucial in the effort of maximizing machine utilization by means of these two algorithms.

6.4 SUMMARY

The rolling horizon algorithms are tested on a dynamic scheduling problem based on real industrial case data from a semiconductor testing facility manufacturing system. The performances of the algorithms were measured in terms of machine utilization value, which are classified as Productive, Non-productive, and Downtime Utilization. It is observed that the RH-GRASP gives 2.93 - 5.75% and the RH-BBL gives 3.26 - 6.78% better productive utilization compared to the SCT, which is used to represent the typical sequencing procedure of a testing facility system. In terms of the WIP average, the MRH-GRASP gives 1.01 - 1.58 (30.05 - 34.42%) and the MRH-BBL gives 1.34 - 2.00 (39.88 - 43.575%) lower WIP average compared to SCT. It is also found that job arrival information with a considerably short horizon length is sufficient to achieve decent sequencing and scheduling decisions for the dynamic scheduling problem.

The proposed algorithms showed promising results based on the experimental settings that have been performed in this study, with an arrival rate of up to 0.15. It is anticipated that further distinctive productive utilization improvement could be obtained when the arrival rate is increased.

7. CONCLUSIONS AND RECOMMENDATIONS

7.1 SUMMARY OF THE RESEARCH

In this research, a single machine scheduling problem with integrated consideration of sequence-dependent setup and non-resumable case of maintenance activity was studied with the specific assumption that setup and maintenance can be performed simultaneously on the machine. Minimizing makespan, which is equivalent with maximizing machine utilization, was the chosen objective function. The analysis was divided into three parts according to the assumed job arrival and maintenance period.

The first part of the problem analysis dealt with an assumption of deterministic-static job arrival and fixed maintenance period. The branch and bound technique was proposed, to obtain an optimal solution to this strongly NP-Hard Problem. The lower bound was calculated based on the column minima of the completion time matrix, and a depth-first and best lower-bound first branching strategy was applied. Despite it being able to provide an optimal solution, the computation time of this approach grows exponentially, which affects its applicable practicality for a problem of a larger size. Hence, three meta-heuristics were developed and

proposed based on the GRASP meta-heuristic, which are the Descending Heuristic (GRASP-DH), the Ascending Heuristic (GRASP-AH), and the Random Heuristic (GRASP-RH). The GRASP-DH is inferior in comparison to the other two meta-heuristics due to the high percentage of idle time. Hence, the choice is between the GRASP-AH, which provides moderate accuracy with a delta average of 6.64% within extremely short computation times, or the GRASP-RH which provides higher accuracy with a delta average of 3.43% with a relatively longer computation time due to its random nature.

Due to its strong assumptions of static job arrival and fixed maintenance period, the results of the Problem Part 1 analysis does not provide an implementable solution to the scheduling problem. However, it provides a crucial basis for the subsequent part of the analysis which had more relaxed assumptions to provide more implementable solutions.

In the second part, the fixed maintenance period assumption was relaxed by allowing the maintenance period to be shifted earlier or later than the machine's tool life. Besides proposing an optimal solution and several heuristic solutions as in the case for the Problem Part 1, analysis was also performed to compare a fixed maintenance period vs. a flexible maintenance period and an early maintenance period vs. a late maintenance period. With a flexible maintenance period, better makespan performance is obtained with the cost of certain yield loss. When the yield loss value is too high, delaying the maintenance period to minimize makespan should be avoided. The cut-off value of yield loss can be calculated

with the help of a quality deterioration model which was developed during the analysis. Further analysis suggests that the value of a flexible maintenance period not only comes from allowing late maintenance, but that early maintenance also contributes to reducing the makespan value. From an average delta percentage of 6.09%, 4.17% comes from performing early maintenance and 1.92% from late maintenance. Both early and late maintenance have their respective costs, which is dependent on the industry application. The results from the Part 2 analysis are helpful in quantifying the cost and value of scheduling decisions, on whether or not maintenance activity is to be performed at a certain point in time, and these quantification results are used as the basis for managerial decisions on the matter.

A dynamic version of the scheduling problem minimizing makespan is discussed in Part 3 of the problem analysis. Dynamic job arrival is modeled based on a rolling horizon procedure, in which job arrival information for a certain limited time (Forecast Window) is known. In the rolling horizon procedure, a scheduling decision is made periodically, once every time the job arrival information is obtained. Two implementable rolling horizon algorithms were proposed, which are a rolling horizon procedure with branch and bound with limited job arrival information (RH-BBL) and a rolling horizon procedure with a GRASP meta-heuristic (RH-GRASP). Their performance is compared with an ideal but non-practical solution obtained from a rolling horizon procedure with branch and bound, with complete job arrival information (RH-BBC). A comparison is also made for the Shortest Completion Time (SCT) algorithm, which depicts a simple

scheduling procedure that is usually implemented in a typical manufacturing system. The practicality of these two rolling horizon algorithms is further extended via a case study based on the historical data of a STATS ChipPAC testing facility system. It was observed that the RH-GRASP gives 2.93 - 5.75% and the RH-BBL gives 3.26 – 6.78% better productive utilization than those obtained with the SCT.

The dynamic scheduling problem with the objective function of minimizing the average of the job flow times is also discussed in Part 3 of the problem analysis. Previous rolling horizon algorithms were modified to cater for the new objective function, resulting in a rolling horizon procedure with branch and bound with limited job arrival information and a modified lower bound (MRH-BBL) and a rolling horizon procedure with a GRASP meta-heuristic with a modified objective function (MRH-GRASP). Their performance is compared with an ideal but non-practical solution obtained from a rolling horizon procedure with branch and bound with complete job arrival information and modified lower bound (MRH-BBC). A comparison is also drawn with the Shortest Completion Time (SCT) algorithm. The practicality of these two rolling horizon algorithms is further extended via a case study based on the historical data of a STATS ChipPAC testing facility system. It was observed that the MRH-GRASP gives 1.01 – 1.58 and the MRH-BBL gives 1.34 – 2.00 lower WIP average than those obtained with the SCT.

7.2 LIMITATION OF THE RESEARCH

This research presents an initial effort to solve scheduling problems with integrated consideration of both sequence-dependent setups and non-resumable cases of maintenance activity. The limitations of the research are summarized as follows:

1. The solution of the scheduling problem is only limited to single machine systems. This solution is still relevant to multiple machine systems, but certain procedures for assigning jobs to each of the machines need to be incorporated into the solution.
2. The problem analysis is only limited to the effort of minimizing makespan (maximizing machine utilization) and minimizing the average of the job flow times (minimizing WIP average). From a theoretical point of view, these two performance measures are the most sensible objective functions due to the nature of the problem. However, in practice, they are not the only concerns in making scheduling decisions.
3. The implementation of proposed solutions requires more computational resources than the commonly used simple dispatching policy.
4. In the quality deterioration model, we require a reliable yield loss model to determine the amount of yield loss incurred when maintenance is not performed beyond tool life. If this yield loss model exists, it can be incorporated into our quality deterioration model.

5. In the rolling horizon procedure, periodical scheduling decisions are only made once for each time subsequent job arrival information is obtained and the sequencing decisions of each rolling horizon are frozen once solved by the algorithm. This policy is easier to implement when fewer resources are required. However, potential further improvement could be obtained by allowing more scheduling points and continually updating the sequencing decisions. This will require more complex algorithms, more computational resources, and job arrival information fed continuously into the system.

7.3 DIRECTION FOR FUTURE RESEARCH

As with any new methods, continual improvement and refinement of the solution proposed in this research is essential, and the proposed directions for future research can be summarized as follows:

1. Job assignment policy can be developed and incorporated into our proposed solution. Hence, the solution framework can also be implemented for multi-machine manufacturing systems.
2. The problem analysis can be extended to cater for a different kind of objective function, such as minimizing lateness / tardiness for manufacturing systems in which on-time delivery is considered critical, or many other possible objective functions.

3. The rolling horizon procedure can be extended by allowing more scheduling points with a continuous update of sequencing decisions (As discussed in point 5 of section 7.2).
4. Quantitative analysis to compare benefit (machine utilization) and cost (e.g. additional manpower) associated with performing setup and maintenance simultaneously. Cost function can be created and compared with the benefit, to develop a decision making tool to help manager in choosing between simultaneous or separate setup-maintenance.

REFERENCES

- Alidaee, B., & Womer, N. (1999). Scheduling with time dependent processing times: review and extensions. *Journal of the Operational Research Society*, 711-720.
- Allahverdi, A., Gupta, J. N. D., & Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. *Omega*, 27(2), 219-239.
- Allahverdi, A., Ng, C., Cheng, T., & Kovalyov, M. Y. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187(3), 985-1032.
- Andres, C., Albarracín, J. M., Tormo, G., Vicens, E., & García-Sabater, J. P. (2005). Group technology in a hybrid flowshop environment: A case study. *European Journal of Operational Research*, 167(1), 272-281.
- Ángel-Bello, F., Álvarez, A., Pacheco, J., & Martínez, I. (2011a). A heuristic approach for a scheduling problem with periodic maintenance and sequence-dependent setup times. *Computers & Mathematics with Applications*, 61(4), 797-808.
- Ángel-Bello, F., Álvarez, A., Pacheco, J., & Martínez, I. (2011b). A single machine scheduling problem with availability constraints and sequence-dependent setup costs. *Applied Mathematical Modelling*, 35(4), 2041-2050.
- Baker, K. R., & Peterson, D. W. (1979). An analytic framework for evaluating rolling schedules. *Management Science*, 25(4), 341-351.

- Bruno, J., & Downey, P. (1978). Complexity of task sequencing with deadlines, set-up times and changeover costs. *Siam Journal on Computing*, 7(4), 393-404.
- Chen, B. (1993). A better heuristic for preemptive parallel machine scheduling with batch setup times. *Siam Journal on Computing*, 22(6), 1303-1318.
- Chen, J. S. (2008). Scheduling of nonresumable jobs and flexible maintenance activities on a single machine to minimize makespan. *European Journal of Operational Research*, 190(1), 90-102.
- Chen, W. J. (2008). Sequencing heuristic for scheduling jobs with dependent setups in a manufacturing system. *The International Journal of Advanced Manufacturing Technology*, 38(1), 176-184.
- Chen, W. J. (2009). Scheduling with dependent setups and maintenance in a textile company. *Computers & Industrial Engineering*, 57(3), 867-873.
- Cheng, T., & Ding, Q. (2001). Single machine scheduling with step-deteriorating processing times. *European Journal of Operational Research*, 134(3), 623-630.
- Conway, R. W., Maxwell, W. L., & Miller, L. W. (2003). *Theory of Scheduling*: Dover.
- Croes, G. (1958). A method for solving traveling-salesman problems. *Operations Research*, 6(6), 791-812.
- Dekker, R., & Scarf, P. A. (1998). On the impact of optimisation models in maintenance decision making: the state of the art. *Reliability Engineering & System Safety*, 60(2), 111-119.

- Drexel, A., & Kimms, A. (1997). Lot sizing and scheduling—survey and extensions. *European Journal of Operational Research*, 99(2), 221-235.
- Duffuaa, S. O., Raouf, A., & Campbell, J. D. (1998). *Planning and Control of Maintenance Systems: Modeling and Analysis*: John Wiley & Sons.
- Eppen, G. D., & Martin, R. K. (1987). Solving multi-item capacitated lot-sizing problems using variable redefinition. *Operations Research*, 35(6), 832-848.
- Feo, T. A., & Resende, M. G. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2), 109-133.
- Feo, T. A., Sarathy, K., & McGahan, J. (1996). A GRASP for single machine scheduling with sequence dependent setup costs and linear delay penalties. *Computers & Operations Research*, 23(9), 881-895.
- Fleischmann, B. (1994). The discrete lot-sizing and scheduling problem with sequence-dependent setup costs. *European Journal of Operational Research*, 75(2), 395-404.
- Fleischmann, B., & Meyr, H. (1997). The general lotsizing and scheduling problem. *OR Spectrum*, 19(1), 11-21.
- Flynn, B. (1987). The effects of setup time on output capacity in cellular manufacturing. *International Journal of Production Research*, 25(12), 1761-1772.
- French, S. (1982). *Sequencing and scheduling: an introduction to the mathematics of the job-shop*: Ellis Horwood.

- Garey, M., & Johnson, D. (1979). Computers and intractability.
- Garg, A., & Deshmukh, S. (2006). Maintenance management: literature review and directions. *Journal of Quality in Maintenance Engineering*, 12(3), 205-238.
- Gavett, J. W. (1965). Three heuristic rules for sequencing jobs to a single production facility. *Management Science*, 11(8), B-166-B-176.
- Gendreau, M., Laporte, G., & Guimarães, E. M. (2001). A divide and merge heuristic for the multiprocessor scheduling problem with sequence dependent setup times. *European Journal of Operational Research*, 133(1), 183-189.
- Gicquel, C., Miègeville, N., Minoux, M., & Dallery, Y. (2009). Discrete lot sizing and scheduling using product decomposition into attributes. *Computers & Operations Research*, 36(9), 2690-2698.
- Gilmore, P. C., & Gomory, R. E. (1964). Sequencing a one state-variable machine: A solvable case of the traveling salesman problem. *Operations Research*, 12(5), 655-679.
- Glassey, C. (1968). Minimum change-over scheduling of several products on one machine. *Operations Research*, 16(2), 342-352.
- Glover, F., & Greenberg, H. J. (1989). New approaches for heuristic search: A bilateral linkage with artificial intelligence. *European Journal of Operational Research*, 39(2), 119-130.

- Guo, A. X., & Wang, J. B. (2005). Single machine scheduling with deteriorating jobs under the group technology assumption. *International Journal of Pure and Applied Mathematics*, 18(2), 225-231.
- Gupta, J. N. D., & Gupta, S. K. (1988). Single facility scheduling with nonlinear processing times. *Computers & Industrial Engineering*, 14(4), 387-393.
- Haynes, R., Komar, C., & Byrd, J. (1973). The effectiveness of three heuristic rules for job sequencing in a single production facility. *Management Science*, 19(5), 575-580.
- Hitomi, K. (1996). *Manufacturing Systems Engineering: A Unified Approach to Manufacturing Technology, Production Management and Industrial Economics*: CRC.
- Hopp, W. J., & Spearman, M. L. (1996). *Factory physics: foundations of manufacturing management*: Irwin.
- Hsu, C. J., Low, C., & Su, C. T. (2010). A single-machine scheduling problem with maintenance activities to minimize makespan. *Applied Mathematics and Computation*, 215(11), 3929-3935.
- Jackson, J. R. (1955). *Scheduling a Production Line to Minimize Maximum Tardiness*: Defense Technical Information Center.
- Jans, R., & Degraeve, Z. (2007). Meta-heuristics for dynamic lot sizing: A review and comparison of solution approaches. *European Journal of Operational Research*, 177(3), 1855-1875.

- Ji, M., He, Y., & Cheng, T. (2007). Single-machine scheduling with periodic maintenance to minimize makespan. *Computers & Operations Research*, 34(6), 1764-1770.
- Kacem, I. (2009). Approximation algorithms for the makespan minimization with positive tails on a single machine with a fixed non-availability interval. *Journal of Combinatorial Optimization*, 17(2), 117-133.
- Kim, C. O., & Shin, H. J. (2003). Scheduling jobs on parallel machines: a restricted tabu search approach. *The International Journal of Advanced Manufacturing Technology*, 22(3), 278-287.
- Kirkpatrick, S., Jr., D. G., & Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, 220(4598), 671-680.
- Krajewski, L. J., King, B. E., Ritzman, L. P., & Wong, D. S. (1987). Kanban, MRP, and shaping the manufacturing environment. *Management Science*, 33(1), 39-57.
- Kubiak, W., & van de Velde, S. (1998). Scheduling deteriorating jobs to minimize makespan. *Naval Research Logistics (NRL)*, 45(5), 511-523.
- Kunnathur, A. S., & Gupta, S. K. (1990). Minimizing the makespan with late start penalties added to processing times in a single facility scheduling problem. *European Journal of Operational Research*, 47(1), 56-64.
- Laguna, M. (1999). A heuristic for production scheduling and inventory control in the presence of sequence-dependent setup times. *IIE transactions*, 31(2), 125-134.

- Laguna, M., Barnes, J. W., & Glover, F. W. (1991). Tabu search methods for a single machine scheduling problem. *Journal of Intelligent Manufacturing*, 2(2), 63-73.
- Laguna, M., & Glover, F. (1993). Integrating target analysis and tabu search for improved scheduling systems. *Expert Systems with Applications*, 6(3), 287-297.
- Lee, C. Y. (1996). Machine scheduling with an availability constraint. *Journal of Global Optimization*, 9(3), 395-416.
- Lin, H. T., & Liao, C. J. (2003). A case study in a two-stage hybrid flow shop with setup time and dedicated machines. *International Journal of Production Economics*, 86(2), 133-143.
- Lin, S., & Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2), 498-516.
- Little, J. D. C., Murty, K. G., Sweeney, D. W., & Karel, C. (1963). An algorithm for the traveling salesman problem. *Operations Research*, 11(6), 972-989.
- Low, C., Ji, M., Hsu, C. J., & Su, C. T. (2010). Minimizing the makespan in a single machine scheduling problems with flexible and periodic maintenance. *Applied Mathematical Modelling*, 34(2), 334-342.
- Ma, Y., Chu, C., & Zuo, C. (2010). A survey of scheduling with deterministic machine availability constraints. *Computers & Industrial Engineering*, 58(2), 199-211.
- Mitsumori, S. (1972). Optimum production scheduling of multicommodity in flow line. *Systems, Man and Cybernetics, IEEE Transactions on*(4), 486-493.

- Monma, C. L., & Potts, C. N. (1989). On the complexity of scheduling with batch setup times. *Operations Research*, 37(5), 798-804.
- Mosheiov, G. (1995). Scheduling jobs with step-deterioration; minimizing makespan on a single-and multi-machine. *Computers & Industrial Engineering*, 28(4), 869-879.
- Ozgur, C., & Brown, J. (1995). A two-stage traveling salesman procedure for the single machine sequence-dependent scheduling problem. *Omega*, 23(2), 205-219.
- Panwalkar, S., Dudek, R., & Smith, M. (1973). *Sequencing research and the industrial scheduling problem*. Paper presented at the Symposium on the Theory of Scheduling and its Applications.
- Pinedo, M. (2008). *Scheduling: Theory, Algorithms, and Systems*: Springer Science + Business Media.
- Presby, J., & Wolfson, M. (1967). An algorithm for solving job sequencing problems. *Management Science*, 13(8), B-454-B-464.
- Schaller, J. E., Gupta, J. N. D., & Vakharia, A. J. (2000). Scheduling a flowline manufacturing cell with sequence dependent family setup times. *European Journal of Operational Research*, 125(2), 324-339.
- Schmidt, G. (2000). Scheduling with limited machine availability. *European Journal of Operational Research*, 121(1), 1-15.

- Sharifnia, A., Caramanis, M., & Gershwin, S. B. (1991). Dynamic setup scheduling and flow control in manufacturing systems. *Discrete Event Dynamic Systems*, 1(2), 149-175.
- Sherif, Y. S. (1982). Reliability analysis: optimal inspection and maintenance schedules of failing systems. *Microelectronics Reliability*, 22(1), 59-115.
- Starfield, T. (2005). Discussion: Deterministic or Stochastic. Retrieved from http://www.uvm.edu/rsenr/vtcfwru/spreadsheets/pom/pdf/lecture_deterministic-or-stochastic_transcript.pdf
- Valdez-Flores, C., & Feldman, R. M. (1989). A survey of preventive maintenance models for stochastically deteriorating single-unit systems. *Naval Research Logistics (NRL)*, 36(4), 419-446.
- Wagner, H. M., & Whitin, T. M. (1958). Dynamic Version of the Economic Lot Size Model. *Management Science*, 5(1), 89-96.
- Wang, J. B., Lin, L., & Shan, F. (2008). Single-machine group scheduling problems with deteriorating jobs. *The International Journal of Advanced Manufacturing Technology*, 39(7), 808-812.
- Wilbrecht, J. K., & Prescott, W. B. (1969). The influence of setup time on job shop performance. *Management Science*, 16(4), B-274-B-280.
- Wortman, D. (1992). Managing capacity: getting the most from your firms assets. *Industrial Engineering*, 24(2), 47-49.

- Wu, C. C., Shiau, Y. R., Lee, L. H., & Lee, W. C. (2009). Scheduling deteriorating jobs to minimize the makespan on a single machine. *The International Journal of Advanced Manufacturing Technology*, 44(11), 1230-1236.
- Wu, C. C., Shiau, Y. R., & Lee, W. C. (2008). Single-machine group scheduling problems with deterioration consideration. *Computers & Operations Research*, 35(5), 1652-1659.
- Xu, D., Yin, Y., & Li, H. (2009). A note on “scheduling of nonresumable jobs and flexible maintenance activities on a single machine to minimize makespan”. *European Journal of Operational Research*, 197(2), 825-827.
- Yang, D. L., Hung, C. L., Hsu, C. J., & Chern, M. S. (2002). Minimizing the makespan in a single machine scheduling problem with a flexible maintenance. *Journal of the Chinese Institute of Industrial Engineers*, 19(1), 63-66.
- Yi, Y., & Wang, D. (2003). Soft computing for scheduling with batch setup times and earliness-tardiness penalties on parallel machines. *Journal of Intelligent Manufacturing*, 14(3), 311-322.
- Zipkin, P. H. (1991). Computing optimal lot sizes in the economic lot scheduling problem. *Operations Research*, 39(1), 56-63.