On-line near-real-time scheduling of semiconductor wafer fabrication

Ang, Adeline Tee Hwee

2008

Ang, A. T. H. (2008). On-line near-real-time scheduling of semiconductor wafer fabrication.
Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/13462>

<https://doi.org/10.32657/10356/13462>

**ON-LINE NEAR-REAL-TIME SCHEDULING OF
SEMICONDUCTOR WAFER FABRICATION**

ANG TEE HWEE ADELINE

SCHOOL OF MECHANICAL AND AEROSPACE ENGINEERING

**A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy**

2008

ACKNOWLEDGMENTS

I would like to take this opportunity to express my heart-felt gratitude to a number of persons who have in one way or another make the completion of this PhD dissertation possible.

First, I would like to thank Assoc. Prof. Sivakumar Appa Iyer, my research supervisor, for his invaluable suggestions, advice and guidance; for his enthusiasm to share vast amount of industrial experiences; his effort in establishing all the industrial and research collaborations; his confidence in my research; and his extensive support and patience throughout the progress of this research.

I also appreciate our industrial collaborators from Chartered Semiconductor Manufacturing Ltd. (especially Mr. Tan Hiap Keong, Mr. Liu Sip Khean, Mr. Madan Mohan, Mr. Loh Choon Leng, Mr. Liu Feng, and Mr. Lim Kian Wee) for their helpful feedback on my research work, for the numerous training sessions and discussions that they have arranged for my better understanding of their operations, and the support they rendered towards the patent application (currently their trade secret).

It is also my privilege to have had numerous constructive discussions with Dr. Viswanath Kumar Ganesan (Singapore MIT Alliance), Dr. M. Mathirajan (Indian Institute of Science), Dr. Qi Chao (Singapore MIT Alliance), and Asst. Prof. Chang Seok Ho (NTU) in their respective area of expertise. I am thankful for their diverse perspectives and beneficial reviews.

To all the CSCM staff and friends known through this period of time and to all my close friends whom I have known for years, I thank you all for your great company, encouragement and support.

I would also like to acknowledge the support A*Star and NTU has given me through the A*STAR-NTU Graduate Fellowship scheme.

My feelings of gratitude also go towards my loving parents, siblings, and Auntie Linda with whom I owe more than words can describe. Finally yet importantly, I thank my husband Guangxian, who has wholeheartedly supported all my decisions, and for sticking with me during the good and bad times.

Adeline Ang

March 2008

TABLE OF CONTENTS

ACKNOWLEDGMENTS..... i
TABLE OF CONTENTS iii
ABSTRACT..... ix
LIST OF FIGURES..... xi
LIST OF TABLES..... xiii
LIST OF ABBREVIATIONS xvi
LIST OF PUBLICATIONS AND INDUSTRIAL CONTRIBUTION..... xviii

CHAPTER 1 INTRODUCTION	1
1.1 BACKGROUND	1
1.2 RESEARCH PROBLEM AND MOTIVATION	5
1.3 RESEARCH OBJECTIVE	9
1.4 RESEARCH SCOPE	10
1.5 RESEARCH CONTRIBUTION	11
1.6 ORGANIZATION OF REPORT	13

CHAPTER 2 LITERATURE REVIEW	14
2.1 PARTIAL TAXONOMY OF SCHEDULING	14
2.2 PREDICTIVE-REACTIVE SCHEDULING (PRS) STRATEGY	19
2.2.1 SCHEDULING/RESCHEDULING METHODOLOGY	20
2.2.1.1 Time Policies (When to Schedule/Reschedule?)	20
2.2.1.1.1 Periodic Scheduling/Rescheduling Time Policy ..	21
2.2.1.1.2 Event-driven Scheduling/Rescheduling Time Policy	23
2.2.1.1.3 Hybrid Scheduling/Rescheduling Time Policy	24
2.2.1.2 Decision policies (How to Schedule/Reschedule?)	26
2.2.1.2.1 Schedule Generation Techniques	26
2.2.1.2.2 Rescheduling Techniques	27
2.2.2 ROBUST SCHEDULING METHODOLOGY	30

2.3	COMPLETELY REACTIVE SCHEDULING (CRS) STRATEGY	32
2.3.1	DISPATCHING	33
2.3.1.1	Common Dispatching Rules and Classification.....	33
2.3.1.2	Review of Dispatching Decision Policies	34
2.3.1.2.1	Selection of Dispatching Rules	35
2.3.1.2.2	Aggregation of Dispatching Rules	38
2.3.1.3	Industrial Applications	40
2.3.2	AGENT-BASED SCHEDULING.....	42
2.3.3	PULL-BASED SCHEDULING	43
2.4	HYBRID REACTIVE SCHEDULING (HRS) STRATEGY	45
2.5	CONCLUSIONS DRAWN FROM LITERATURE REVIEW	47
 CHAPTER 3 TIME POLICIES OF SCHEDULING METHODOLOGIES ...		52
3.1	EXISTING PSR AND DISPATCHING TIME POLICIES	52
3.1.1	PSR TIME POLICY	53
3.1.2	DISPATCHING TIME POLICY	56
3.2	IDENTIFYING THE GAPS BETWEEN LITERATURE AND ACTUAL IMPLEMENTATION	57
3.3	BASIC CONCEPTS OF NEXTS TIME POLICY	60
3.4	PRELIMINARY COMPARISON OF TIME POLICIES USING AN ILLUSTRATIVE EXAMPLE	61
3.5	COMPARISON OF TIME POLICIES USING DISCRETE EVENT SIMULATION	64
3.5.1	OBJECTIVE & ASSUMPTIONS	64
3.5.2	PROBLEM DESCRIPTION	65
3.5.3	REQUIRED PROCESSING CAPACITY vs AVAILABLE MACHINE CAPACITY	67
3.5.4	METHODOLOGY EVENT-BASED REPRESENTATION.....	69
3.5.5	VERIFICATION & VALIDATION	73
3.5.6	EXPERIMENTAL RUNS	73
3.6	EXPERIMENTAL RESULTS AND DISCUSSION	76
3.7	SUMMARY	81

CHAPTER 4 VARIANTS OF NEXTS TIME POLICY AND A CASE STUDY
.....83

4.1	PROBLEM DESCRIPTION	84
4.2	DYNAMIC NEXTS TIME POLICY HEURISTICS.....	86
4.2.1	ALGORITHM OF THE FIXED TIME PERIOD POLICY	93
4.2.2	ALGORITHM OF THE LOADPORT-INITIATED TIME POLICY	96
4.2.3	ALGORITHM OF THE ADJUSTED TIME PERIOD POLICY	97
4.3	EXPERIMENTAL EVALUATION OF THE DYNAMIC NEXTS TIME POLICY HEURISTICS	98
4.3.1	COMPUTATIONAL EXPERIMENTS	98
4.3.2	RESULTS AND ANALYSIS OF THE NEXTS TIME POLICIES.	101
4.4	IMPLEMENTATION	106
4.5	SUMMARY	106

**CHAPTER 5 PRELIMINARY STUDY OF DECISION POLICIES FOR
MULTIOBJECTIVE SCHEDULING.....109**

5.1	PARETO OPTIMALITY AND DOMINANCE CONCEPTS	110
5.2	MULTIOBJECTIVE OPTIMIZATION AND SCHEDULING	111
5.2.1	INCLUSION OF RULES.....	115
5.3	MARKOV PROCESS MODELING FOR PRELIMINARY STUDY OF LOOK UPSTREAM, LOOK DOWNSTREAM, AND LOOK INTO QUEUE	116
5.3.1	MODEL DESCRIPTION.....	118
5.3.2	ASSUMPTIONS	121
5.3.3	DETAILED MARKOV MODELS	121
5.3.3.1	Control and Operational Rule Indicator Variable.....	122
5.3.3.2	Update of Buffer Levels B_1 TO B_6	123
5.3.3.3	Update of Buffer Levels B_7 TO B_{10}	126
5.3.3.4	State Space Reduction	128
5.3.3.5	Transition Probabilities.....	129
5.3.3.6	Update of Rewards for Machine.....	130
5.3.3.7	Numerical Solution	131
5.3.4	VERIFICATION	133

5.3.5	RESULTS & DISCUSSION	134
5.3.5.1	Varying Failure and Repair Probabilities (Same Machine Availability Across All Machines).....	134
5.3.5.2	Varying Machine Availability Across Machines.....	135
5.3.5.3	Varying Buffer Capacity	136
5.3.5.4	Discussion	137
5.4	CRITERIA SELECTION AND ANALYSIS	138
5.4.1	DISPATCHING CRITERIA	139
5.4.2	COMPUTATIONAL EXPERIMENTS	139
5.4.3	RESULTS AND DISCUSSION.....	143
5.4.3.1	Preliminary Evaluation and Analysis of Simulation Results	143
5.4.3.2	Comparison of Rules using Nonparametric Tests.....	146
5.4.3.3	Selection of Candidate Scheduling Criteria (Dispatching Rules)	147
5.4.3.4	Evaluation of Candidate Scheduling Criteria (Dispatching Rules)	149
5.5	CORRELATION ANALYSIS OF OBJECTIVES	155
5.6	SUMMARY	158
 CHAPTER 6 PROPOSED NEXTS DECISION POLICY		160
6.1	INCORPORATING USER PREFERENCES INTO DECISION-MAKING.....	160
6.1.1	DESIRABILITY FUNCTIONS	161
6.1.2	DESIRABILITY FUNCTION FOR A CRITERION TO BE MINIMIZED	162
6.1.3	DESIRABILITY FUNCTION FOR CRITERION WITH TARGET	163
6.2	REACTIVE WEIGHTS OPTIMIZATION.....	164
6.3	LOOK AHEAD SIMULATION-BASED GENETIC ALGORITHM WITH DESIRABILITY FUNCTION	168
6.3.1	DISCRETE EVENT SIMULATION	170
6.3.1.1	Determining Number of Replications for the Sub Simulation.	172
6.3.2	GENETIC ALGORITHM	174

6.3.2.1	Genetic Algorithm and Simulation Platform	175
6.3.2.2	Pareto-Based Versus Preference-Based Genetic Algorithm Approaches	175
6.3.2.3	Representation	176
6.3.2.4	Selection and Population Replacement.....	177
6.3.2.5	Crossover	178
6.3.2.6	Mutation.....	180
6.3.2.7	Fitness Evaluation.....	181
6.3.2.8	Stopping Criterion.....	182
6.3.2.9	GA Parameters Selection	186
6.4	COMPARISON OF RSM & LASGAD.....	188
6.4.1	RESPONSE SURFACE METHODOLOGY (RSM).....	188
6.4.2	COMPARISON OF RSM AND LASGAD FOR CRITERIA WITH USER SPECIFIED TARGET VALUES	189
6.4.3	COMPARISON OF RSM AND LASGAD FOR CRITERIA TO BE MINIMIZED	192
6.5	SUMMARY	194
CHAPTER 7 INDUSTRIAL APPLICATION: CASE STUDY		196
7.1	PROPOSED TIME AND DECISION POLICIES.....	196
7.2	CASE STUDY MODEL, DATA and EXPERIMENTATION	197
7.3	RESULTS AND DISCUSSION	200
CHAPTER 8 CONCLUSION AND RECOMMENDATIONS		206
8.1	SUMMARY OF THE RESEARCH	206
8.2	LIMITATIONS OF THE RESEARCH	211
8.3	DIRECTIONS FOR FUTURE RESEARCH.....	212
REFERENCES.....		214
APPENDICES.....		231
APPENDIX A DEFINITION OF SOME TERMINOLOGY USED		232
APPENDIX B SEMICONDUCTOR MANUFACTURING, WAFER FABRICATION, AND ITS SCHEDULING COMPLEXITIES		234

B.1	SEMICONDUCTOR MANUFACTURING.....	234
B.2	WAFER FABRICATION	236
B.3	COMPLEX INHERENT CHARACTERISTICS OF WAFER FAB.....	238
B.4	COMPLEXITIES DUE TO UNCERTAINTIES	239
	APPENDIX C T-TESTS RESULTS TO COMPARE NEXTS, PSR, AND DISPATCHING TIME POLICIES.....	242
	APPENDIX D CYCLE TIME AVERAGE DATA AND ANOVA ON RANKS RESULTS	246
	APPENDIX E CYCLE TIME AND TARDINESS RELATED PERFORMANCE OF DISPATCHING RULES	253
	APPENDIX F SIMULATION RESULTS FOR MOFCS AND MOFCS+ RULES	256
	APPENDIX G SIMULATION RESULTS FOR COMPARISON OF RSM & LASGAD	263
G.1	FITTING META-MODEL FOR THE AVERAGE CYCLE TIME RESPONSE	263
G.2	FITTING META-MODEL FOR THE STANDARD DEVIATION OF CYCLE TIME RESPONSE	264
G.3	FITTING META-MODEL FOR THE AVERAGE TARDINESS RESPONSE	266
G.4	NUMERICAL OPTIMIZATION RESULTS	268
	G.1.1 OPTIMIZATION FOR CRITERIA WITH SPECIFIED TARGETS	268
	G.1.2 OPTIMIZATION FOR CRITERIA TO BE MINIMIZED.....	269
	APPENDIX H SIMULATION RESULTS FOR THE INDUSTRIAL CASE STUDY	270

ABSTRACT

Very little research have considered the combined aspects of scheduling involving (i) a dynamic and stochastic shop floor, (ii) Sequence-Dependent Setup (SDS), (iii) transportation, and (iv) the consideration of user preferences to achieve the best/preferred trade-offs amongst the multiple contradicting objectives. Given these complexities, it is indeed challenging to determine when and how the next task (i.e. transportation, setup, processing of a lot) should be selected and executed on each machine such that the best/preferred trade-offs amongst the multiple conflicting objectives can be achieved based on user preferences in an online, near-real-time manner. In this research, the NEXt Task Scheduling (NEXTS) methodology, made up of a NEXTS time policy and a NEXTS decision policy, is proposed. The NEXTS time policy specifies when to schedule. In essence, the time policy stipulates the appropriate timing (i.e. before and as close as possible to the time when the machine is available) for the selection and transportation of the next lot. The NEXTS decision policy specifies how a scheduling decision is to be made. The decision policy is made up of a multiobjective rule (i.e. weighted aggregation of dispatching criteria/rule) that is applied for the selection of the next lot and a Look Ahead Simulation-based Genetic Algorithm with Desirability function (LASGAD) that optimizes the weights used in the multiobjective rule in accordance with user preferences. The multiple performance measures of interest in this research are average cycle time, standard deviation of cycle time, average tardiness, and standard deviation of tardiness. The NEXTS time policy is shown to perform better than the existing Periodic Scheduling/Rescheduling and dispatching time policies

when there are transportation and decision making times involved. Simulation results also show that on-line LASGAD is better at finding the optimal set of weights than the off-line response surface methodology with desirability function method. The novel NEXTS methodology contributes to the advancement of knowledge by making use of real-time data for effective on-line scheduling of a dynamic and stochastic shop floor with sequence-dependent setups, transportation times, and user specified preferences to obtain Pareto optimal performance in the face of multiple conflicting objectives.

LIST OF FIGURES

Figure 2.1: Partial taxonomy of scheduling.....	16
Figure 3.1: PSR methodology's time policy.....	53
Figure 3.2: A breakdown of the schedule generation duration.....	55
Figure 3.3: (a) Dispatching methodology's time policy with negligible next job selection duration, and (b) Dispatching methodology's time policy with next job selection duration.....	56
Figure 3.4: NEXTS methodology's time policy.....	61
Figure 3.5: Effect of (a) PSR time policy, (b) dispatching time policy, and (c) NEXTS time policy on the unexpected arrival of an urgent job.....	62
Figure 3.6: Event-based representation of PSR.....	70
Figure 3.7: Event-based representation of dispatching.....	70
Figure 3.8: Event-based representation of NEXTS.....	71
Figure 3.9: Moving averages for WIP.....	75
Figure 3.10: Percentage improvement in cycle time of NEXTS vs PSR and Dispatching across five levels of variability and disturbance.....	79
Figure 4.1: Partial shop floor.....	84
Figure 4.2: Gantt chart for the three proposed time policies.....	92
Figure 4.3: Average relative percentage deviation (<i>ARPD</i>) of the average cycle time (<i>avgCT</i>) chart.....	103
Figure 4.4: Maximum relative percentage deviation (<i>MRPD</i>) of the average cycle time (<i>avgCT</i>) chart.....	103
Figure 5.1: Pareto optimality.....	111
Figure 5.2: Markov model of a simplified job shop with no re-circulation.....	119
Figure 5.3: Definition of the policy-dependent term, $I_{i,j}^*(t)$	126
Figure 5.4: Update of the states of dummy buffers following a job departure from B ₃ or B ₄	127
Figure 5.5: Update of the states of dummy buffers following a job arrival at B ₃ and/or B ₄	128
Figure 5.6: Mean of average cycle time of rules under different problem configurations.....	143

Figure 5.7: Mean of standard deviation of cycle time of rules under different problem configurations	144
Figure 5.8: Mean of average tardiness of rules under different problem configurations.....	144
Figure 5.9: Mean of standard deviation of tardiness of rules under different problem configurations.....	144
Figure 5.10: Strong positive correlation between the average tardiness and standard deviation of tardiness objectives for $\{PI,DL\}$ problem configuration	157
Figure 5.11: Strong positive correlation between the average tardiness and standard deviation of tardiness objectives for $\{PI,DT\}$ problem configuration	158
Figure 6.1: Desirability function $d(x_i)$ for criterion x_i to be minimized	163
Figure 6.2: Desirability function $d(x_i)$ for criterion x_i with a specified target $T(x_i)$	164
Figure 6.3: Conceptual application of the reactive weights optimization.....	165
Figure 6.4: Overview of look-ahead simulation-based genetic algorithm with desirability function (LASGAD).....	169
Figure 6.5: Conceptual application of main and sub simulation modules	171
Figure 6.6: Flowchart of LASGAD	174
Figure 6.7: Blend crossover operator	179
Figure 6.8: Plot of $\left(1 - u^{(1-\gamma/\gamma_{\max})^\beta}\right)$ against γ th generation	181
Figure 6.9: Total desirability value versus number of generations.....	184
Figure 6.10: Average cycle time versus number of generations.....	185
Figure 6.11: Standard deviation of cycle time versus number of generations.....	185
Figure 6.12: Average tardiness versus number of generations	185
Figure 7.1: Performance for different methodology configuration (combination of the dispatching time policy with various decision policies) for wafer fab case study	202
Figure 7.2: Performance for different methodology configuration (combination of the NEXTS time policy with various decision policies) for wafer fab case study	202

LIST OF TABLES

Table 2.1: Brief explanation of the time and decision policies when used in the context of various methodologies	18
Table 2.2: Advantages and disadvantages of reviewed scheduling methodologies	47
Table 2.3: Summary of the closely related references	50
Table 3.1: Input variables with their corresponding values and/or distributions.....	68
Table 3.2: Degree of perturbation applied to arrivals and MTTF under different levels of variability and disturbance	68
Table 3.3: Available machine capacity under different levels of variability and disturbances	68
Table 3.4: Mean and 95% C.I. for the average cycle time.....	77
Table 3.5: Percentage improvement in cycle time of NEXTS vs PSR and Dispatching across five levels of variability and disturbance.....	79
Table 3.6: Summarized results on the tests of hypotheses between NEXTS time policy, and the PSR and dispatching time policies	80
Table 4.1: Description of notations and abbreviations used in the proposed NEXTS time policy heuristics	88
Table 4.2: Summary of the experimental design	100
Table 4.3: Mean and 95% confidence intervals for $avgCT$	102
Table 4.4: Average relative percentage deviation ($ARPD$) of average cycle time ($avgCT$) results averaged over 960 problem instances	106
Table 5.1: Brief description of some classical methods and algorithms.....	112
Table 5.2: Description of notations and abbreviations used in Markov process modeling	117
Table 5.3: Look Around rules that are applied to machine M_3	120
Table 5.4: Conditions considered in each Look Around rule to decide which job type is given priority for processing on machine M_3	125
Table 5.5: Transition probabilities of a particular machine M_i	130
Table 5.6: Average cycle time for different failure and repair probabilities (with same machine availability of 0.909)	134

Table 5.7: Average cycle time for varying machine variability across machines .	135
Table 5.8: Average cycle time for varying buffer capacity	136
Table 5.9: Dispatching criteria/rules	140
Table 5.10: Lot types, product mix, and processing times data	141
Table 5.11: δ values for loose and tight due date setting corresponding to each setup level.....	142
Table 5.12: Comparable rules based on Tukey Test results.....	147
Table 5.13: Rules that gives the best performance in each of the performance objective	148
Table 5.14: Weights used for the MOFCS and MOFCS+ rule.....	150
Table 5.15: Dominated and non-dominated solutions of the MOFCS+ and MOFCS rule for the $\{PI,DL\}$ and $\{PI,DT\}$ problem configurations	151
Table 5.16: Comparing MOFCS+ and MOFCS rules for the $\{PI,DL\}$ and $\{PI,DT\}$ problem configurations using the concept of domination across four objectives ($avgCT$, $sdCT$, $avgTARD$, and $sdTARD$).....	154
Table 5.17: Correlation coefficient and P-value (in bracket) for each pair of objectives	156
Table 6.1: Estimated number of replications required	173
Table 6.2: Values used in the desirability function for criterion with specified target	184
Table 6.3: 2-level experimental setup	186
Table 6.4: Mean and 95% confidence interval of simulation results obtained from the best chromosome for each set of GA parameters.....	188
Table 6.5: Optimized weights and its corresponding predicted performance and total desirability.....	190
Table 6.6: Simulated performance and total desirability using the optimized weights for criterion with target values	192
Table 6.7: Optimized weights and the predicted performance and total desirability... ..	193
Table 6.8: Simulated performance and total desirability using the optimized weights for criterion to be minimized	193
Table 7.1: Wafer fab case study model.....	198
Table 7.2: Summary of the experimental design for the wafer fab model.....	198

Table 7.3: Summary of the optimized weights found using LASGAD.....	200
Table 7.4: Percentage improvement	203

LIST OF ABBREVIATIONS

Strategies

CRS	Completely Reactive Scheduling
HRS	Hybrid Reactive Scheduling
PRS	Predictive-Reactive Scheduling

Methodologies

NEXTS	NEXt Task Scheduling
PSR	Periodic Scheduling/Rescheduling

Dispatching Rules

COVERT	Cost OVER Time
CR	Critical Ratio
EDD	Earliest Due Date
FCFS / FIFO	First-Come, First-Serve / First-In, First-Out
LPT	Longest Processing Time
LS / MS	Least Slack / Minimum Slack
SCT	Shortest Cycle Time
SPT	Shortest Processing Time
SRPT	Shortest Remaining Processing Time
STNV	Shortest Time until Next Visit
WINQ	Work (content) In Next Queue
SSU	Same / Smallest SetUp
SSU+	Same / Smallest SetUp Modified
MOFCS	Multi-Objective rule made up of FIFO, CR, and SSU rules
MOFCS+	Multi-Objective rule made up of FIFO, CR, and SSU+ rules

Look Around Components (used in Markov Process Modeling)

LD	Look Downstream
LD_B	Look Downstream at the Buffer states
LD_M	Look Downstream at the Machine states
LD_BM	Look Downstream at both the Buffer and Machine states
LQ_B	Look at the Queue's Buffer states
LU	Look Upstream
LU_B	Look Upstream at the Buffer states
LU_M	Look Upstream at the Machine states
LU_BM	Look Upstream at both the Buffer and Machine states

Event Notations (used in the Methodology Event-Based Representation)

PE	Processing End
PS	Processing Start
PSRT	Periodic Scheduling/Rescheduling Time
SE	Selection End
SGE	Schedule Generation End
SGS	Schedule Generation Start
SS	Selection Start

TE	Transportation End
TS	Transportation Start

Measures

avgCT	Average Cycle Time
sdCT	Standard Deviation of Cycle Time
avgTARD	Average Tardiness
sdTARD	Standard Deviation of Cycle Time
RPD	Relative Percentage Deviation
ARPD	Average Relative Percentage Deviation
MRPD	Maximum Relative Percentage Deviation

Others

AGV	Automated Guided Vehicle
AMHS	Automated Material Handling System
AutoSchedAP	AutoSched TM Accelerated Processing
DOE	Design Of Experiment
GA	Genetic Algorithm
IID	Independent and Identically Distributed
LASGAD	Look Ahead Simulation-based Genetic Algorithm with Desirability function
MES	Manufacturing Execution System
MTTF	Mean Time To Failure
MTTR	Mean Time To Repair
PM	Preventive Maintenance
RSM	Response Surface Methodology
RTD	Real Time Dispatcher TM
TTF	Time To Failure
TTR	Time To Repair
WIP	Work-In-Process

LIST OF PUBLICATIONS AND INDUSTRIAL CONTRIBUTION

JOURNALS

Ang, A. T. H., Sivakumar, A. I., Qi, C., 2008, "Criteria selection and analysis for single machine dynamic scheduling with multiple objectives and sequence-dependent setups", *Computers & Industrial Engineering* (Accepted manuscript (Ref: CAIE2366), unedited version available online: 31 Jul 2008).

Ang, A. T. H., Sivakumar, A. I., and Mathirajan, M., 2008, "Dynamic time policy for scheduling a discrete wafer fabrication tool using upstream information", *International Journal of Production Research* (Under review, Manuscript ID: TPRS-2008-IJPR-0176).

Ang, A. T. H., Sivakumar, A. I., Qi, C., 2008, "Look-ahead simulation-based genetic algorithm with desirability function for multiobjective scheduling", *IEEE Transactions on Semiconductor Manufacturing* (Under review, Manuscript Ref: #7011).

CONFERENCES

Ang, T. H., and Sivakumar, A. I., 2005, "Next task scheduling time policy for on-line near-real-time scheduling of semiconductor wafer fabrication", In *Proceedings of the International Conference on Modeling and Analysis of Semiconductor Manufacturing (MASM 2005)*, pp. 214-221.

Ang, A. T. H., Sivakumar, A. I., 2007, "Online multiobjective single machine dynamic scheduling with sequence dependent setups using simulation-based genetic algorithm with desirability function", In *Proceedings of the 2007 Winter Simulation Conference (WSC2007)*, Eds. Henderson, S. G., Biller, B., Hsieh, M.-H., Shortle, J., Tew, J. D., and Barton, R. R., pp. 1828-1834.

TRADE SECRET

Ang, T. H., Sivakumar, A. I., Tan, H. K., Loh, C. L., Lieu, S. K., and Liu, F., 2004, "Next optimum task decision support for wafer fabrication", Filed and accepted as *Trade Secret of Chartered Semiconductor Manufacturing Ltd., Singapore*, Ref: CS04/029.

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

In a continually evolving, market driven, and competitive environment, companies are placing more emphasis on market responsiveness to outdo competitors. While quoting a shorter delivery times can make a difference to securing orders in the global market, the ability to consistently fulfill customers' requirements and meeting committed on-time delivery dates are equally important, as failure to do so may result in significant loss of goodwill. In order to achieve these objectives together with high asset utilization, there is a need for effective production management decisions. Tasks and decisions involved in production management generally fall into one of the three levels: strategic planning, tactical planning, and operational control (Hitomi, 1996).

Strategic planning, aligned with corporate goals and management vision, typically focuses on long-term capacity and policy-related decisions such as capital investment for facility layout and added resource capacity, new product or process introductions and product-mix decisions. The planning horizon (length of the time span for which decisions are made) may vary from one to several years. Tactical planning is concerned with the matching of production to fluctuating demands by the generation of production plans (e.g. master schedule) in consideration of the availability of resources specified by strategic plans. This can involve a horizon of

weeks or months. Operational control (or shop floor scheduling) deals with short-term allocation and coordination of resources between operations of a few minutes to a few weeks. The decisions made at this level include job release into the shop floor, job scheduling, dispatching, preventive maintenance (PM), and expediting jobs through getting part-time help, overtime or sub-contracting, etc.

There is no doubt that manufacturing performance is an important determinant of success in the manufacturing industry and effective shop floor scheduling is a key aspect to improving system performance. There are generally two stages of operational control in shop floor scheduling: (i) release scheduling or job release, involving the control of the right types, amount and time point of release of jobs into the manufacturing plant; (ii) shop floor scheduling or dispatching, involving the continued control over the work-in-process (WIP) and the jobs that have been released into the plant. This entails deciding which job will be loaded next when the machine is available.

Although several studies (Wein, 1988; Johri, 1993; Sivakumar et al., 2003; Qi and Sivakumar, 2004) have shown that application of appropriate release policies has a greater effect on performance measures than dispatching, Adachi et al. (1988), Kumar (1994), DiSalvi et al. (1995), and Li et al. (1996) demonstrated that a good shop floor scheduling or dispatching policy is also necessary to significantly improve system performance. In this research, the focus is on the second stage of shop floor scheduling. After all, if shop floor scheduling or dispatching does not result in a reasonably good performance, job release however well executed is futile. The reverse is also true.

Relevant to the second stage of shop floor scheduling, there are generally two methods to make decisions pertaining to which job will be processed next when the machine is available:

- (i) The first method is to generate a predictive schedule for all jobs over a scheduling horizon using analytical/mathematical models, optimization models, scheduling/dispatching algorithms or heuristics and to process the jobs according to the sequence and timing specified by the schedule. Rescheduling may be performed to update or revise the schedule.
- (ii) An alternative method is to apply simple heuristics or rules to prioritize all the jobs that are waiting for processing on a workstation and to select the job with the highest priority index without following a pre-defined schedule. The next job to process is typically selected each time the machine becomes available.

The commonly used performance measures considered in published works include makespan, mean and standard deviation or variance of cycle time, machine utilization, queue time, and on-time delivery or due date related performance (e.g. tardiness, lateness), throughput rate, etc. Appendix A provides a definition of these performance measures.

The research community has, in recent years, made unparalleled advances in both the theoretical and practical aspects of scheduling. Despite these, the scheduling problem is far from solved (Smith, 2003). Shop floor scheduling is still an important research topic due to the impact it has over the performance of many manufacturing systems, particularly for complex systems like semiconductor manufacturing.

The overall semiconductor manufacturing flow can generally be divided into four stages: (i) wafer fabrication, (ii) wafer probe, (iii) assembly or packaging, and (iv) final testing. This research focuses on wafer fabrication because it is the most technologically complex and capital-intensive phase compared to the other stages. The facility where wafer fabrication takes place is commonly referred to as a wafer fab. A lot is a set of wafers that travels as a unit between tools (machines) and is the equivalent terminology for job in wafer fabrication. Hence, in this thesis, lot will be used in place of job whenever it is used in the context of wafer fabrication. Appendix B provides a detailed discussion of semiconductor manufacturing, wafer fabrication, as well as the complexities of scheduling in wafer fabrication.

To summarize, wafer fabrication is highly complex due to the presence of high product mix variability, variable order sizes, different routings with varying number of processing steps, large number of equipment types, sequence-dependent setups, re-entrant flow (i.e. multiple visits to the same tools/machines), and a multitude of other job-related, machine-related, and operations-related constraints. In addition, wafer fabrication is also highly dynamic and stochastic in nature. New orders may arrive randomly in varying quantities over time, and lot priorities and due dates may be modified at any processing steps. Machines may also fail unexpectedly and it may be difficult to predict the time required to repair the machine.

Given the complexities highlighted above, it is usual for a wafer fab to have lot cycle times that are two to ten times the theoretical minimum cycle times (when running close to full capacity) (Robinson, 2002). It is also not unusual for such wafer fabs to miss more than 20-30% of the scheduled deliveries (Dabbas, 1999). The complexities, coupled with the dynamic and stochastic nature of wafer

fabrication, pose great challenges for shop floor scheduling to minimize cycle times and due date related objectives.

1.2 RESEARCH PROBLEM AND MOTIVATION

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

Given the complexities, dynamic and stochastic nature of wafer fabrication, determine when and how the next task (i.e. transportation, setup, processing of a lot) should be selected and executed on each machine such that the best/preferred trade-offs amongst multiple conflicting objectives can be achieved based on user preferences in an online, near-real-time manner.

Most existing literature on scheduling has focused on optimal solutions for simple, static and deterministic problems. This implies that many inherent dynamic and stochastic characteristics of the manufacturing environment in which scheduling occurs have not been adequately dealt with. The lack of scheduling research to deal with the complexities, disturbances, and uncertainties described above is cited as a major reason for its low impact on industrial practice (Aytug et al., 2005).

A vast amount of the literature to determine and execute the next task on the shop floor is generally based on either a pre-determined schedule or dispatching rules (usually in the form of dispatch list). A schedule provides the optimal timing and sequence of tasks to be followed. However, it is generally not possible to execute the schedule exactly as generated in a highly dynamic environment as the stochastic occurrence of unexpected events results in substantial deviation between the generated schedule and the actual shop floor status. Even when rescheduling is performed to update or revise the schedule, the extensive computation is likely to

take a relatively long time to return a result, far more time than an operator or automated transporter can efficiently wait for instructions about what to process next.

An alternative to scheduling tasks in a complex manufacturing environment is through dispatching. Dispatching is commonly used in practice due to its robustness (i.e. ability to react to uncertainties), low computational requirements, ease of implementation, and intuitive appeal (i.e. easy to comprehend). However, in cases where transportation durations are non-negligible and have to be considered, dispatching is unable to allow for continuous processing which is especially important for ensuring starvation avoidance on bottleneck machines. This is because dispatching selects the next lot only when the machine is available, and the transportation that result thereafter will cause the machine to wait (in idle mode) for the next lot to arrive (Wright et al., 2002). This is an action performed too late. The need to consider the required transportation, by dispatching the next lot (provided it is available) before the machine becomes available, is generally impossible to accomplish with the existing dispatching time policy (Krishnaswamy and Nettles, 2005).

Another example in which the time component is important is the scheduling of lots through the chemical processing steps that involve a time link constraint. This time link constraint means that the wafers, once processed at a particular machine at certain process steps, cannot be delayed for more than a specified maximum waiting duration before the machine at the next processing step processes it. This also requires knowledge and pre-planning of steps that is generally hard to achieve with the existing dispatching time policy (Krishnaswamy and Nettles, 2005). The

above shows that the timing aspect of scheduling and its integration with transportation is generally lacking.

Simple dispatching rules (e.g. First-In, First-Out, Shortest Processing Time, Earliest Due Date) are often labeled as local and myopic. As such, some studies proposed the use of global rules (e.g. Dabbas (1999) proposed the global line balance rule to balance the line output for a high volume, high mix re-entrant flow shop). Each rule typically addresses an objective/criterion of interest. Hence, in studies that consider multi-objective optimization, many converted multiple objectives into a single objective optimization problem using a weighted combination of rules (Dabbas, 1999; Deb, 2001; Lin et al., 2005; Gupta, 2006). These studies either use response surface methodology with desirability function determine the optimal weights to use, or generate numerous Pareto optimal solutions for user to select and apply. These methods have been accomplished off-line. As such, more research can be conducted to implement a real-time optimization of the weights to adapt to the wafer fab's changing condition.

Review papers by Allahverdi et al. (1999), Yang and Liao (1999), and Zhu and Wilhelm (2006) have also shown that few papers have considered the combined aspects of scheduling involving (i) a dynamic and stochastic shop floor, (ii) Sequence-Dependent Setup (SDS), (iii) transportation, and (iv) the consideration of user preferences to achieve the best/preferred trade-offs amongst the multiple contradicting objectives. The unresolved issues, gaps, and complexities discussed above suggest that it is indeed challenging to schedule the next task at a machine level on-line, in near real time.

In the semiconductor industry, perhaps nowhere are the stakes higher than in manufacturing with wafer fab costing in excess of US\$2 billion (Watts, 2000). Given the extremely high capitalization, wafer fabs strive to optimize equipment utilization by keeping their equipment as productive as possible to achieve a higher throughput. Watts (2000) reported that even a mere 1% increase in utilization can generate half million dollars per month for any moderate-volume fab. Lyman (2002) states that it would already have a significant effect if the efficiency and throughput were improved by only one wafer a day. These provide justification for scheduling the fab to manage the WIP and process flow as effectively as possible in wafer fabrication.

This research work forms a significant part of a research initiative established with Chartered Semiconductor Manufacturing Ltd., which also includes the research work of Qi (2005) and Gupta (2006) that relate to job release, and Pareto-control for multi-objective scheduling respectively. Not only has Chartered Semiconductor Manufacturing Ltd. shown great interest in our research findings (presented in this thesis), this research collaboration has allowed a better appreciation of the gap between theoretical research and industrial practice and the importance of shop floor scheduling in wafer fabs. The knowledge that an opportunity exists for developing novel scheduling methods that are of practical relevance, coupled with an available avenue for validation of our proposed scheduling approach, provides much motivation for this research topic.

1.3 RESEARCH OBJECTIVE

Motivated by the significance of the shop floor scheduling in wafer fabrication, the main aim of this research is to devise a better scheduling methodology (made up of a time policy and a decision policy that specifies ‘when’ and ‘how’ to select the next task) to improve multiple performance criteria of the wafer fab in terms of (i) average cycle time, (ii) standard deviation of cycle time, (iii) average tardiness, and (iv) standard deviation of tardiness. This can be achieved via the following objectives:

- (i) To review and understand the existing scheduling methodologies, and to identify unresolved issues and gaps from the context of the time and decision policies;
- (ii) To define and present the conceptual framework of our proposed NEXt Task Scheduling (NEXTS) time policy, and justify its relative performance in comparison with the existing time policies;
- (iii) To develop practical variants of NEXTS time policies for implementation, and to validate their applicability in an industrial case study;
- (iv) To experimentally investigate which rules (whether look upstream, look downstream, or commonly used rules that look into queue) should be incorporated for next lot selection to be in line with improving the cycle time and due date-based performance measures stated above;
- (v) To identify how user preferences could be incorporated in the optimization of weights;
- (vi) To define and present the conceptual framework of our proposed NEXTS decision policy, and justify the relative performance of the on-

line approach used to optimize the weights used in the weighted aggregation of rules with existing approach; and

- (vii) To analyze the performances of the proposed NEXTS time policy and the NEXTS decision policy when they are used in tandem using an industrial case study.

1.4 RESEARCH SCOPE

The relevance of this research is in a highly dynamic and stochastic environment such as wafer fabrication that is characterized by multitude of complexities and uncertainties. The availability of test data from industrial collaborator Chartered Semiconductor Manufacturing Ltd. and the presence of inherent complex and challenging characteristics are the main reasons why this research focused on wafer fabrication.

The scheduling problem is restricted to the selection of the next task (i.e. transportation, setup, processing of a lot) for each machine before and as close as possible to the point where the machine is available in an on-line, near-real-time manner in order to support the realization of the required or expected user specified performance measures. Any machine in the fab can apply the proposed time policies although the proposed decision policies will require some adaptation concerning the selection of rules as no one rule (or a set of rules) will apply optimally in all environments.

Other scheduling decisions such as job release and job sizing are not considered in this research even though they could have an impact on system performance. In this research, it is assumed that there will always be a planned release from which it is

known when and which jobs are released into the manufacturing shop floor. The data sets used in this research work are reflective of actual fab data but may have been scaled by a factor for confidentiality issues.

Stankovic (1988) defined a real-time system as one whose correctness depends not only on the logical results of computation, but also on the time at which the results are produced. In our context, a scheduling system is considered real-time if it is able to deliver a Pareto optimal scheduling decision within one second. As the NEXTS scheduling procedures involve a finite amount of time (despite no significant delays, it is likely to take more than one second) to retrieve the on-line shop floor data, to make scheduling decision(s), and to update the shop floor of the scheduled decision(s) for execution, near-real-time is used for our application.

1.5 RESEARCH CONTRIBUTION

The original contributions of this research include:

1. Introduction of a partial taxonomy of scheduling for classifying scheduling methodologies that could be applied in dynamic and stochastic manufacturing environments and for understanding different scheduling methodologies from the context of the time and decision policies.
2. Development of a novel NEXTS time policy that can take transportation and/or decision delays into consideration by stipulating the timing for carrying out the next lot selection and transportation before and as close as possible to the time when the machine is available.

3. Designs of event-based representations to guide the customization of discrete-event simulation tools as these tools are typically unable to simulate the timing policies applied in the NEXTS and Periodic Scheduling/Rescheduling (PSR) methodologies without customization.
4. Development of a Markov process model to analytically study the benefits of incorporating upstream, downstream, and queue information (i.e. machine and buffer states in the context of Markov modeling) for next lot selection.
5. Identification of the importance of basing the selection of candidate criterion/rule not only on the criterion/rule's strength in optimizing an objective but also the trade-offs the criterion has concerning the other conflicting objectives.
6. Development of a novel decision policy that is made up of a MOFCS rule that can consider multiple conflicting objectives of interest and a Look Ahead Simulation-Based Genetic Algorithm with Desirability function (LASGAD) that utilizes the desirability function to incorporate user preferences to guide the optimization of weights (towards the user specified or desired performances) using a real-coded genetic algorithm.
7. Introduction of the concept of reactive optimization of the weights to continuously search for better weights automatically on-line in near real time to deal with the factory's changing condition.
8. Development of the NEXTS methodology that combines the proposed NEXTS time policy (i.e. fixed time period policy) and the NEXTS decision

policy (i.e. LASGAD with MOFCS rule) with promising results observed using an industrial case study.

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 from the context of the time and decision policies for dynamic and stochastic environments. In Chapter 3, the existing PSR and dispatching time policies are evaluated and compared against a NEXTS conceptual time policy. Chapter 4 presents an industrial-based discrete event simulation case study performed to compare three proposed detailed variants of the NEXTS time policies. Chapter 5 introduces multiobjective optimization and the Pareto optimality concept used in the NEXTS decision policy, evaluates the scheduling criteria and analyzes how the dispatching rules (i.e. look upstream, look downstream, and/or look into queue) should be selected. In Chapter 6, the Look-Ahead Simulation-based Genetic Algorithm with Desirability function used for the optimization of weights in the NEXTS decision policy is proposed and compared with commonly used off-line response surface methodology with desirability function. Chapter 7 presents an industrial case study to evaluate the performance when the proposed NEXTS time policy and decision policy are applied in tandem. Chapter 8 concludes the thesis and suggests directions for future research.

CHAPTER 2

LITERATURE REVIEW

Despite the availability of a vast amount of literature on scheduling methodologies, the question of when and how the next task should be scheduled, in an on-line, near-real-time manner for a dynamic and stochastic manufacturing environment with sequence-dependent setups and transportation times such that the best/preferred trade-offs amongst multiple conflicting objectives can be achieved or controlled, is a fruitful area for research.

This chapter reviews the existing methodologies and policies that are applicable in a dynamic and stochastic manufacturing environment and classifies them into three scheduling strategies in an adapted taxonomy. 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, as well as to highlight the potential contributions of this research.

2.1 PARTIAL TAXONOMY OF SCHEDULING

There has been an extensive body of research on shop floor scheduling, both from theoretical and practical perspectives. The existing literature can be broadly classified into two main areas: static and dynamic. In the static environment, there are a finite number of jobs to be scheduled and no further jobs are considered until the next scheduling instance or until all jobs considered in the scheduling horizon

are completed. In the dynamic case, jobs can arrive in a random manner during the scheduling horizon. However, it is observed that one frequent assumption made, which rarely holds in practice, is that the scheduling environment is static (Baker, 1974; French, 1982; Pinedo and Chao, 1999; Błażewicz et al., 2001; Pinedo, 2002).

The scheduling research can also be classified as either deterministic or stochastic (French, 1982; Pinedo and Chao, 1999; Pinedo, 2002). Deterministic models assume that all information related to the jobs is known with certainty. Conversely, stochastic models assume that not all job information is known exactly beforehand, only their distributions are known in advance. In a wafer fab, the dynamic and stochastic environment is a more realistic representation of the scheduling problem.

Apart from the two generalized classifications, some researchers, such as Suresh and Chaudhuri (1993), Jones and Rabelo (1998), and Jain and Meeran (1999), classify the scheduling literature based on scheduling approaches and solution techniques such as mathematical programming, decomposition methods (e.g. shifting bottleneck, rolling horizon procedures), artificial intelligence, neural networks, search methods (e.g. branch and bound, filtered beam search, tabu search, simulated annealing, genetic algorithms, fuzzy logic, simulation-based scheduling, etc.), heuristics, and dispatching rules (e.g. single, composite, multiobjectives). Pinedo and Chao (1999), Błażewicz et al. (2001), Pinedo (2002), and Onwubolu (2002) provide a good introduction on some of the above-mentioned schedule generation techniques. Zweben and Fox (1994), and Brown and Scherer (1995) include works describing a wide range of knowledge-based and AI approaches for generating and updating production schedules. Bagchi (1999) is a good reference for multiobjective scheduling methods using genetic algorithms.

In contrast to previous papers, the literature review discussed in this chapter is based on a different taxonomy that is more appropriate for classifying the existing knowledge that is relevant to this research. The purpose of proposing a partial taxonomy of scheduling is to clarify the contextual nature of our research in relation to the published work. The partial taxonomy of scheduling (see Figure 2.1) that classifies various scheduling methodologies for application in a dynamic and stochastic setting is partially adapted from the papers of Vieira et al. (2003) and Aytug et al. (2005).

Figure 2.1: Partial taxonomy of scheduling

In the existing literature, various scheduling strategies can achieve the common purpose of scheduling a dynamic and stochastic manufacturing shop floor. Predictive-Reactive Scheduling (PRS) strategy, for example, is a common categorization for the classification of scheduling methodologies that involves a

two-step process of scheduling. An initial schedule is generated in advance, and when necessary, rescheduling is carried out to generate a completely new schedule or to update the initial schedule) (Vieira et al., 2003; Aytug et al., 2005). Completely Reactive Scheduling (CRS, also known as dynamic scheduling) strategy is another commonly used categorization for the classification of scheduling methodologies that do not produce schedules but uses decentralized production control methods to dispatch jobs when the machine is available (Vieira et al., 2003).

Robust scheduling and scheduling/rescheduling methodologies involve generating an initial optimal/robust schedule in advance (i.e. periodic, event-driven, hybrid) followed by rescheduling using right-shift, partial/repair, or complete regeneration to update the initial schedule. Robust scheduling and scheduling/rescheduling methodologies can be classified under PRS while dispatching, agent-based, and pull-based scheduling methodologies can be classified under CRS. Sections 2.2 and 2.3 provide a review of the PRS- and CRS-related methodologies respectively.

In this taxonomy, the Hybrid Reactive Scheduling (HRS) strategy is added to accommodate methodologies that can neither be appropriately classified as absolutely a PRS nor CRS strategy. The author introduced the HRS classification for methodologies that are either (i) a hybrid of both PRS and CRS (e.g. switching between scheduling and dispatching, such as that studied by Matsuura et al. (1993)); or (ii) different from PRS and CRS but yet bear some resemblances to the partial aspects of each predictive-reactive and completely reactive scheduling (e.g. NEXt Task Scheduling (NEXTS) methodology proposed in this research). The new

HRS strategy category suggests that research may be lacking in this area. Section 2.4 provides a review for the HRS-related methodologies.

Iverson and Özgüner (1999) use the *time policy* and *decision policy* terminologies and they are therefore not new. In our context, the two terminologies generally refer to ‘*when to make scheduling decisions*’ and ‘*how to make scheduling decisions*’ respectively. Each methodology identified in Figure 2.1 can be defined by a time policy and a decision policy. Table 2.1 provides a brief explanation of what the time and decision policies refer to when used in the context of scheduling/rescheduling, dispatching and NEXTS methodologies.

Table 2.1: Brief explanation of the time and decision policies when used in the context of various methodologies

Methodology	Time Policy	Decision Policy
Scheduling/ Rescheduling	Refers to when to generate a new schedule and/or update the initial schedule (in short, when to schedule/reschedule) e.g. <i>in advance</i> (i.e. periodic, event-driven, hybrid)	Refers to how to generate a new schedule and/or update the initial schedule (in short, <i>how to schedule/reschedule</i>) e.g. optimal/robust initial schedule followed by reschedule using right-shift, partial/repair, or complete regeneration
Dispatching	Refers to when to select the next job for processing at a machine e.g. <i>when machine is available</i>	Refers to how to select the next job for processing at a machine e.g. dispatching rules
NEXt Task Scheduling (NEXTS)	Refers to when to select the next task for processing at a machine e.g. <i>before machine is available</i>	Refers to how to select the next task based on predictive knowledge/schedule for processing at a machine e.g. dispatching rules based on look-ahead schedule

As the relevance of this research is applicable in a highly dynamic and stochastic environment, this taxonomy is relevant for the review of existing scheduling methodologies that are capable of dealing with uncertainties and disruptions.

2.2 PREDICTIVE-REACTIVE SCHEDULING (PRS) STRATEGY

Scheduling (or predictive scheduling) is defined as a decision-making process that involves the allocation of scarce resources to tasks over some time period into the future with the goal of optimizing one or more objectives (Pinedo, 2002) while satisfying a set of domain constraints (Vieira et al., 2000a). Generally, these constraints include temporal constraints, decision makers' objectives, and inherent process and resource requirements. This decision-making process is therefore a process of creating a predictive schedule for a given set of jobs and resources over a scheduling horizon (Kempf, 1994). The schedule specifies, for each resource required for production, the planned start time and end time of each job assigned to that resource. After scheduling, the schedule is released to the shop floor for implementation.

Disturbances may occur frequently in real manufacturing systems (Stoop and Wiers, 1996). Depending on the frequency and the impact of unexpected events, considerable differences between the pre-determined schedule and its actual realization on the shop floor will eventually make it difficult or impossible for the degraded schedule to be precisely followed (Parunak, 1991). Under such circumstances, rescheduling (or reactive scheduling) will almost be mandatory to revise and update the existing production schedule in response to disruptions and changes (Yamamoto and Nof, 1985; Church and Uzsoy, 1992; Matsuura et al., 1993; Smith, 1994; Abumaizar and Svestka, 1997; Fang and Xi, 1997; Jain and Elmaraghy, 1997; Shafaei and Brunn, 1999a, 1999b; Cowling and Johansson, 2002; Vieira et al., 2003; Sabuncuoglu and Kizilisik, 2003).

In contrast, some researchers (Yellig and Mackulak, 1997; Shafaei and Brunn, 1999b, 2000) have described approaches for generating robust schedules (Section 2.2.2 reviews the robust scheduling methodology) that is expected to perform well even if disruptions occur.

2.2.1 SCHEDULING/RESCHEDULING METHODOLOGY

Scheduling/rescheduling is frequently termed as predictive-reactive scheduling in the literature. Based on the taxonomy (see Figure 2.1), scheduling/rescheduling is differentiated as a methodology that is classified under the predictive-reactive scheduling strategy. Scheduling/rescheduling methodology is a two-step process. First, a predictive schedule is generated for a scheduling horizon. Second, rescheduling is performed to update or revise the schedule when unexpected events invalidate the existing schedule. The scheduling/rescheduling research is mainly directed towards two main areas (Vieira et al., 2003): (i) time policies (when to schedule/reschedule?), and (ii) decision policies (how to schedule/reschedule?).

2.2.1.1 Time Policies (When to Schedule/Reschedule?)

Three types of rescheduling time policies have been studied: periodic, event-driven, and hybrid. Whether rescheduling is performed periodically, upon the occurrence of events, or under a hybrid of both, the points in time where rescheduling is performed are referred to as rescheduling points (Aytug et al., 2005). Any events that occur between rescheduling points are ignored until the next rescheduling point (Shafaei and Brunn, 1999a; Aytug et al., 2005).

2.2.1.1.1 Periodic Scheduling/Rescheduling Time Policy

Periodic time policy reschedules the facility periodically at a fixed or varied time interval. Most of the studies focused on fixed time interval rescheduling (Hashimoto et al., 1993; Kempf, 1994; Herrmann et al., 1995; Watt, 1998; Shafaei and Brunn, 1999a, 1999b, 2000; Sivakumar, 1999, 2001). In the fixed time interval method, rescheduling is performed at equally spaced rescheduling points (e.g. every shift, every day, etc.). Studies by Sabuncuoglu and Karabuk (1999), and Sabuncuoglu and Kizilisik (2003) use the variable time interval rescheduling. Variable time interval implies that the rescheduling points are not equally spaced. Rather, rescheduling points are determined based on the percentages of jobs processed or total processing time realized on all machines in the system.

Hashimoto et al. (1993) proposed an automatic scheduler for daily rescheduling (with rescheduling performed every day) in application specific integrated circuit (ASIC) manufacturing lines. Emphasis is placed on quick product turnaround time through lot-release control, control of the number of urgent lots in the line, and precise due date control. Kempf (1994) employed an artificial intelligence (AI) based tool that generates a production schedule for each shift of a semiconductor wafer fabrication plant. Herrmann et al. (1995) used a genetic algorithm to find good job shop schedules at the beginning of each shift in a semiconductor test facility to minimize the number of tardy jobs. Watt (1998) discussed the integration of simulation-based scheduling with MES at Sony Semiconductors to generate schedules automatically at periodic intervals for use by the shop floor operators.

Shafaei and Brunn (1999a, 1999b, 2000) integrated job release, routing, scheduling and frequent rescheduling to create robust periodic predictive schedule to achieve

performance in a dynamic job shop environment on the basis of a rolling time horizon with a cost-based performance measure using data sampled from distributions that are fitted from the actual collected data. The rolling time horizon approach solves the scheduling problem for a medium- to long-term horizon (i.e. multiple production periods) but implements only the schedule for the first period. Rescheduling is performed at subsequent period by taking into account new information from the shop floor. Simulation study results show that rescheduling frequently (using a shorter rescheduling interval) is effective in improving scheduling performance under tight due date conditions.

Sivakumar (1999, 2001) described the design and development of a deterministic discrete event simulation-based on-line multiobjective scheduling system to generate schedules and prediction reports for the near term at a specified frequency, initiated from real time status of the environment. They reported achieving a Pareto optimal cycle time distribution and machine utilization with the implementation of the system at a semiconductor backend using a periodic rescheduling time policy. *Pareto optimal* solutions refer to the boundary line of solutions that are better than the other solutions in the feasible objective space with respect to two or more conflicting objectives. See Section 5.1 for a brief discussion of the Pareto optimality concept.

The papers reviewed above have focused on a pre-determined, fixed periodic scheduling interval. Sabuncuoglu and Karabuk (1999) and Sabuncuoglu and Kizilisik (2003) reported that variable time interval method (determined based on the percentages of jobs processed or total processing time realized on all machines) is better than fixed time interval method because the variable time method is more

responsive to the state of the system. Sabuncuoglu and Karabuk (1999) and Sabuncuoglu and Kizilisik (2003) have shown that increasing the frequency of rescheduling (or decreasing the rescheduling interval) will also improve the performance of the system.

While periodic time policy may yield more *schedule stability* (measures the number of revisions or changes that a schedule undergoes during execution) and less *schedule nervousness* (i.e. more predictability) (Vieira et al., 2003), disregarding events that occurred between the rescheduling points may compromise system performance as these events may invalidate the schedule before the next rescheduling is performed. In addition, it is difficult to determine an optimal rescheduling interval in a changing environment.

2.2.1.1.2 Event-driven Scheduling/Rescheduling Time Policy

Event-driven time policy usually involves initial generation of schedule and subsequent rescheduling to update and revise the initial schedule on an event-driven basis, that is, upon the occurrence of events such as machine breakdowns, arrival of urgent jobs, etc. Most of the literature in rescheduling that uses this event-driven time policy is studied using a static environment (Vieira et al., 2003).

For dynamic rescheduling environments, Jain and ElMaraghy (1997) used genetic algorithms to obtain an initial schedule and performed local rescheduling only for those operations that are affected by the four different types of event-based uncertainties (i.e. unforeseen machine breakdowns, increased order priority, rush orders arrival and order cancellations).

In the work of Abumaizar and Svestka (1997), machine breakdowns, delays in the arrival of materials, arrival of rush jobs, and cancellation of orders, are generalized as machine breakdowns as they involve a disruption in the processing of operations on a machine or machines for a period of time. The Affected Operations Algorithm, based on a binary branching algorithm, is proposed to perform rescheduling upon machine breakdown. This algorithm minimizes both the increase in makespan and the deviation from the initial schedule, and is capable of managing both ‘interrupt-resume’ and ‘interrupt-repeat’ modes of interrupted operations.

Bierwirth and Mattfeld (1999) applied a genetic algorithm to create a new schedule every time a new job arrives. Vieira et al. (2000a, 2000b) have researched event-driven time policies that trigger rescheduling when the total number of job arrivals reaches a threshold for single and parallel machines systems respectively.

Event-driven time policy has the advantage of being able to react and redirect operations upon each disturbance. However, there is also a high chance for the system to be in a permanent state of rescheduling as events can occur in rapid succession in a large, dynamic and stochastic facility (Vieira et al., 2003), resulting in a ‘nervous’ system. The system is nervous (less stable) because the frequent revisions make the schedule less predictable and difficult to follow for execution.

2.2.1.1.3 Hybrid Scheduling/Rescheduling Time Policy

Hybrid time policy reschedules the facility both on a periodic basis (i.e. at each rescheduling point) and upon the occurrence of events (i.e. between two rescheduling points). Yamamoto and Nof (1985) used scheduling/rescheduling to generate an initial schedule of all operations in a system for a fixed time period in

advance using a tree search procedure based on the branch and bound method. Rescheduling is then performed to revise the initial schedule whenever a machine breaks down. They showed that their proposed approach is advantageous over fixed sequencing (i.e. no revision of schedule once it is generated) and priority dispatching (using longest total processing time for release and FCFS for dispatching). Periodically and when significant disruptions occur (i.e. upon the occurrence of events), Church and Uzsoy (1992) reschedule the entire facility from scratch.

Liao et al. (1996) formulated an integer-programming problem capable of capturing various salient features of a research and development pilot line of semiconductor wafer fabrication. Their scheduling methodology consists of three parts: (i) an efficient algorithm for developing a near-optimal daily schedule under nominal conditions using Lagrangian relaxation and network flow technique; (ii) fast rescheduling for timely adjustment of the nominal daily schedule to cope with disturbance and uncertainties (i.e. capacity gain/loss and lot hold/release) using a neighborhood search; and (iii) periodic rescheduling using the “rolling horizon scheme” and the nominal scheduling algorithm of (i). The performance of this study will be presented in Section 2.2.1.2 when Liao and coworkers’ work on ‘how to reschedule’ is reviewed.

Chacon (1998) described the use of periodic scheduling to schedule jobs at Sony Semiconductor. Manual rescheduling is performed in between the periodic rescheduling points in the event an unscheduled occurrence made the initial schedule significantly obsolete. Vieira et al. (2000b) studied a hybrid time policy

that triggers rescheduling when a machine breaks down and when a repair is completed.

While most studies involving periodic and event-driven time policies have used simulation experiments to evaluate the performance of the proposed approaches, Vieira et al. (2000a) presented analytical models that can satisfactorily predict important performance measures (average flow time and machine utilization) of one-machine systems under periodic and event-driven rescheduling in a dynamic environment. The scheduling algorithm uses a first-in first-out dispatching rule to sequence jobs and groups jobs with similar types to save setup time.

While the hybrid time policy may be able to combine the advantages of periodic and event-driven rescheduling time policies, it inherits their limitations as well. For example, it is still not clear what rescheduling intervals to use, which significant events that will trigger rescheduling, and how to reschedule (see Section 2.2.1.2). In fact, event-driven scheduling policy is still not well understood (let alone hybrid rescheduling policy).

2.2.1.2 Decision policies (How to Schedule/Reschedule?)

2.2.1.2.1 Schedule Generation Techniques

Schedule generation techniques (see next paragraph for references) that result in optimal, near-optimal, robust, or just reasonably good schedules include the use of mathematical programming, decomposition methods (e.g. shifting bottleneck, rolling horizon procedures), artificial intelligence, neural networks, search methods (e.g. branch and bound, filtered beam search, tabu search, simulated annealing,

genetic algorithms, fuzzy logic, simulation-based scheduling, etc.), heuristics, and dispatching rules (e.g. single, composite, multiobjectives).

The review of all of the above-mentioned schedule generation techniques is beyond the scope of this research. Only papers that suggest rescheduling techniques (see Section 2.2.1.2.2) or techniques that could generate robust schedules (see Section 2.2.2) will be reviewed, as these are most relevant to rescheduling. Readers can refer to Pinedo and Chao (1999), Błażewicz et al. (2001), Pinedo (2002), and Onwubolu (2002) for a good introduction on some of the above-mentioned schedule generation techniques. Zweben and Fox (1994), and Brown and Scherer (1995) include works describing a wide range of knowledge-based and AI approaches for generating and updating production schedules. Readers can refer to Bagchi (1999), Dimopoulos (2004), and Hart et al. (2005) for multiobjective scheduling methods using genetic algorithms. Section 2.3.1 provides a review of the dispatching rules.

2.2.1.2.2 Rescheduling Techniques

Rescheduling may involve right-shifting affected operations forward in time (right-shift), repairing the current affected schedule (repair/partial), or restarting the scheduling process from scratch (complete regeneration).

Right-shift rescheduling involves delaying the entire schedule by the duration of the interruption (Abumaizar and Svestka, 1997). The effect is generally to push the schedule forward in time. Scheduling/rescheduling methodologies that uses the periodic time policy typically involves complete regeneration of schedule at each

rescheduling point (Hashimoto et al., 1993; Kempf, 1994; Herrmann et al., 1995; Watt, 1998; Shafaei and Brunn, 1999a, 1999b, 2000; Sivakumar, 1999, 2001).

Yamamoto and Nof (1985) reschedule the entire set of operations from scratch when machine breaks down. Rescheduling is performed by first fixing those operations that have already been processed or are in processing and then revising all other non-fixed operations. Their rescheduling procedure is reported to take less than two minutes for six machines and 15 part types of varying quantities problem. Church and Uzsoy (1992) developed a hybrid time policy with dynamic job arrivals for single and parallel machines. In between regular rescheduling, only specific events will result in the entire facility being rescheduled and schedules are generated using the Earliest Due Date (EDD) rule to minimize maximum lateness.

Rossi and Dini (2000) proposed using a real-time genetic algorithm (with minimal number of genetic operations needed to reach the optimal) to reduce the time required to reschedule a FMS. Yang and Wu (2003) proposed an adaptive genetic algorithm to find new sub-optimal schedule of a large and complicated job shop FMS shortly after the interruption occurred. Both studies focus on reducing the time it requires to generate a new schedule following disruptions that render the initial schedule unusable.

For a larger and more complex facility such as semiconductor manufacturing, this procedure may be time consuming, and often results in response times unacceptable to the user. In view of this, schedule repair or partial rescheduling is suggested as favorable by researchers such as Efstathiou (1996), Liao et al. (1996), Jain and Elmaraghy (1997), Abumaizar and Svestka (1997), etc.

Efstathiou (1996) presented an anytime heuristic method for schedule repair in response to dynamic conditions. Anytime heuristics means that the repair procedure is interruptible to provide a partial solution when limited time is available. Liao et al. (1996) performed a complete regeneration under periodic rescheduling and partial adjustment of the periodically generated schedule under event-driven rescheduling. Liao et al. (1996) reported that daily (periodic) scheduling is able to generate schedule of high quality while the fast rescheduling algorithms are computationally efficient and resulted in good schedules with smooth adjustments from the original one for an R&D pilot line.

Jain and Elmaraghy (1997) and Abumaizar and Svestka (1997) have reported performing rescheduling only for the operations that are directly or indirectly affected by random disruptions. Abumaizar and Svestka (1997) have also demonstrated that their Affected Operations Rescheduling Algorithms produce superior performance over complete regeneration and right-shift rescheduling methods in most rescheduling scenarios.

Cowling and Johansson (2002) propose two quantitative measures of utility (measures the benefit gained by using a particular rescheduling approach) and stability (measures the destabilizing effect of local schedule change on the affected schedule) as a guide to decide whether to do nothing, to repair or to reschedule from scratch in response to production disturbances. The measures are studied using real time information for a single machine mathematical scheduling model. Hence, more work has to be done to demonstrate its practicality for use in industrial and commercial applications. In another study by Sabuncuoglu and Kizilisik

(2003), results showed that full rescheduling is generally better than partial rescheduling but at a cost of higher computational load.

Upasani et al. (2006) proposed a workcenter-based decomposition heuristic that partitions the workcenters in a fab into heavily loaded and lightly loaded classes and solves the global scheduling problem only for the heavily loaded workcenters. The problem reduction technique is demonstrated to generate acceptable global schedule in reasonable computational times. This represents a major step towards real-time scheduling/rescheduling unlike previous approaches that are generally limited by lengthy runtime and small problem size. However, additional experimentation involving the periodic rescheduling work by Barua et al. (2005) (review of Barua et al. (2005) is available in Section 2.4) would help to further investigate the performance of the proposed technique.

2.2.2 ROBUST SCHEDULING METHODOLOGY

While prior research may have concentrated on developing optimal or near-optimal schedules under the assumption of a static, deterministic environment and rescheduling when dynamic events invalidate the generated schedule, another group of researchers such as Yellig and Mackulak (1997), Mehta and Uzsoy (1998, 1999), O'Donovan et al. (1999), and Shafaei and Brunn (1999b, 2000) choose to deal with the rescheduling process by anticipating the unexpected occurrence of random events so that when disruptions occur, the necessary changes in the schedule are minimal (Pinedo, 2002). Vieira et al. (2003) defined schedule robustness as a measure of how much disruption would degrade the performance of the system as the schedule is executed.

Several studies include deliberate idle time into the schedule to absorb the impacts of breakdowns for single machine (Mehta and Uzsoy, 1998; O'Donovan et al., 1999) and job shop (Mehta and Uzsoy, 1999) scheduling problems. Mehta and Uzsoy (1998, 1999) and O'Donovan (1999) apply their approaches to minimizing maximum lateness and total tardiness, respectively. In these three studies, the insertion of idle time provides high predictability with minor sacrifices in single objective shop performance.

Yellig and Mackulak (1997) proposed the use of capacity hedging (capacity is held in reserve to protect against a stochastic event) to generate a robust schedule for application in a make-to-order environment. This scheduling strategy anticipates failure based on historical machine performance and maintains a static schedule in this stochastic environment by reserving machine capacity upfront. There is, however, a trade-off of throughput for schedule stability (i.e. stable on-time demand fulfillment), though not excessive. In addition, as unexpected breakdown is the only type of stochastic event considered, this schedule is unlikely to be robust in a manufacturing facility in which there are other stochastic events (e.g. increased order priority, urgent orders arrival and order cancellations, etc.).

Shafaei and Brunn (1999b) investigated the robustness of a number of scheduling rules in a dynamic and stochastic environment. The SPT-CR (combination of shortest processing time and critical ratio) dispatching rule with frequent rescheduling is efficient in generating robust predictive schedules. Since the shop loading (i.e. job release) and balancing (i.e. routing) activities are the main factors that influence the robustness of a schedule, they are equally as important as frequent rescheduling. Shafaei and Brunn (2000), in their further work, put forth a

framework to integrate job release, routing, and scheduling to create robust predictive schedule.

The use of optimal scheduling algorithms that require lengthy computation may require the schedule to be robust enough to deal with the uncertainties to avoid cases where the schedule is invalid even before it is generated or implemented in a dynamic and stochastic environment. This is because the whole wafer fab data (i.e. large amount of data) could not be possibly retrieved in real-time and the retrieval time involved also contributes to the aging of the retrieved data. In general, determining the best rescheduling solution remains an open research issue and, consequently, is the most difficult of the rescheduling process (Vieira et al., 2003).

2.3 COMPLETELY REACTIVE SCHEDULING (CRS) STRATEGY

Completely reactive scheduling (CRS) strategy, also known as dynamic scheduling, is different from predictive-reactive scheduling (PRS) strategy in that it does not produce schedules. Another difference is that scheduling determines when and which jobs use each of the facility resources in advance of resource availability, while dispatching decides which job a factory resource processes only upon the availability of the resource (Baker, 1998). CRS strategies usually use decentralized production control methods to dispatch jobs when necessary and use available information only at the required moment (Vieira et al., 2003). Dispatching, agent-based scheduling and pull-based scheduling are some methodologies classified under CRS. Out of the three, dispatching has been extensively implemented in industry (refer to Section 2.3.1.3).

2.3.1 DISPATCHING

Dispatching, an alternative to scheduling all jobs on each machine, involves making a decision as to which job is processed next at each machine every time a job finished processing on the machine (Morton and Pentico, 1993; Yurtsever and Pierce, 1998; Pinedo, 2002). In dispatching, jobs are not scheduled for processing. They are dispatched to resources without following a pre-defined schedule. The next job to seize a particular resource for processing is selected only when the resource becomes available using dispatching rule(s) (Vieira et al., 2000a; Pinedo, 2002). A dispatching rule is a rule that prioritizes all the jobs that are waiting for processing at a machine and the job with the highest rank or priority in the queue is usually selected for processing.

The terms dispatching rules, priority rules, scheduling rules, sequencing rules, and heuristics are often used interchangeably in the literature. This is because dispatching rules have been applied for the selection of the next job to be processed (in dispatching methodology) and the generation/regeneration of schedules (in scheduling/rescheduling methodology). In dispatching, the time policy typically refers to the selection of the next job when the machine is available using dispatching rules (i.e. decision policy).

2.3.1.1 Common Dispatching Rules and Classification

Dispatching rules such as First-In First-Out (FIFO), Earliest Due Date (EDD), Shortest/Longest Processing Time (SPT/LPT), Shortest Remaining Processing Time (SRPT), Critical Ratio (CR), Minimum Slack (MS), Same SetUp (SSU), Work In Next Queue (WINQ), Cost OVER Time (COVERT), etc., and their

variations have been researched. Panwalker and Iskander (1977) carry out the most comprehensive survey of dispatching rules in which 113 rules are presented, reviewed and classified. Other discussion and/or classification of dispatching rules can also be found in Blackstone et al. (1982), Pinedo and Chao (1999), Pinedo (2002), and Onwubolu (2002), etc.

There are various ways to classify dispatching rules. For example, distinction can be made between static and dynamic rules, local and global rules (Pinedo and Chao, 1999). Static rules are not time-dependent (e.g. FIFO, EDD). They are just function of job data, machine data, or both. Dynamic rules, on the contrary, are time-dependent (e.g. MS, COVERT). A local rule uses information pertaining only to either the queue where the job is waiting or machine where the job is queued. A global rule may use information regarding other jobs or machines within and outside its vicinity.

2.3.1.2 Review of Dispatching Decision Policies

Rules can be employed singly (i.e. same rule is used for all functions, or resources), or they may be used sequentially, usually in order of importance. The sequential usage is useful for resolving any ties that may arise after the application of a rule. Yurtsever and Pierce (1988) developed a system capable of data-driven decision support and dispatching for application in a wafer fab. In their system, a lot is selected from the queue based on three dispatch decisions applied in hierarchical importance.

Alternatively, different individual rule may be used for each associated functions, or resources. Lu et al. (1994), for example, used a combination of workload

regulation rules for release and fluctuation smoothing rules for dispatching to reduce the mean and variance of cycle-time in semiconductor manufacturing plant. The study by Jayamohan and Rajendran (2000) suggested the use of a single dynamic dispatching rule at every stage of a system incorporating the information of both the process time and due-date, as it is quite effective for minimizing the measures related to flow time and tardiness in shop scheduling. In another study, Min and Yih (2003a, 2003b) described the use of one rule for each function in order to obtain desired performance measures at the end of a certain production interval. The functions considered include: (i) lot release, (ii) selection of lots by critical machine, (iii) selection of lots by non-critical machine, (iv) selection of lot by stocker, and (v) selection of lots by vehicles on monorail.

Rather than applying existing dispatching rules, Geiger et al. (2006) proposed an autonomous learning approach to automatically discover effective priority dispatching rules based on genetic programming. Dispatching rules that are discovered by the search procedure are encoded into tree-based representations and evaluated in a variety of single machine environments with a simulation model.

2.3.1.2.1 Selection of Dispatching Rules

As dispatching rules are unable to adjust to the dynamics of the facility, a rule or a set of rules that provide unsurpassed performance at one point in time may not continue to do so as the facility evolves (Schorn-DiSalvi et al., 1995). Similar to scheduling/rescheduling, the selection process may occur only once, periodically triggered, event-driven, or a hybrid of both. These techniques include enumerative periodic selection (Wu and Wysk, 1989), state-dependent periodic selection (Tyan et al., 2002; Chen et al., 2004), artificial intelligence based on expert system (Ran

and Cheng, 1996), artificial/competitive neural networks (Min and Yih, 2003a, 2003b), genetic algorithms (Cardon et al., 2000), data mining (Yoshida and Touzaki, 1999; Li and Olafsson, 2005), discrete-event simulation (Kim et al., 2003), data envelopment analysis (Braglia and Petroni, 1999), etc., or various combinations of the above.

Wu and Wysk (1989) and Ishii and Talavage (1991) suggested the enumerative periodic method to select a best dispatching rule for application in the next short time period. Simulations are run periodically to evaluate the performance of the entire candidate dispatching rules according to the changes that have occurred in the system state. Pierreval and Mebarki (1997) proposed a ‘Shift From Standard Rules’ (SFSR) heuristic dispatching strategy that dynamically selects a dispatching rule from a pre-determined list of rules based on the operating conditions of the system (i.e. changing system states) each time a machine becomes available, in order to meet a primary objective and maintain good results on a secondary production objective.

Many studies employed the use of a decision tree for the dynamic selection of dispatching rules (Tyan et al., 2002; Chen et al., 2004; Wang et al., 2005; Li and Olafsson, 2005).

Tyan et al. (2002) developed a TOC-based state-dependent (TSD) dispatch rule that considers (i) three state variables (i.e. machine utilization, machine queue length, and contention factor), (ii) three dispatch rules (i.e. two boundary, shortest time to next visit, and fastest time for next visit), and (iii) mapping rules (i.e. a decision tree to select the dispatching rule based on the system states). Tyan et al. (2002),

however, did not discuss when to reselect the dispatching rule. In a similar further work by Chen et al. (2004), selection of rules is carried out on a 24-hour basis. They report that the time interval deserves further investigation.

Wang et al. (2005) developed a hybrid knowledge discovery model, using a combination of (i) decision tree (off-line rule learning module; constructed using ID3 algorithm) to determine an appropriate dispatching rule given a specific performance measure and system status, and (ii) a back-propagation neural network (on-line prediction module) to predict precisely the performance of the selected rule of a semiconductor final testing factory.

Li and Olafsson (2005) proposed a new framework for using data mining to discover dispatching rules from production data. The use of decision trees as dispatching rules are evaluated based on data created using four simple rules (i.e. earliest due date, weighted shortest processing time first, minimum slack, earliest release date) based on pair-wise comparison between two jobs as the target concept to be learned for a single machine problem. The decision tree algorithm can accurately discover how the jobs were dispatched by the particular rule, although the EDD and WSPT rules appear to be slightly more difficult to discover. As there may be more than two jobs in the queue for selection, it is still not known how this inductive learning approach will perform if comparisons between multiple jobs are considered.

Evolutionary methods typically applied a genetic algorithm to generate potential candidate chromosomes that encode the dispatching rule to be used in each machine/stage of the manufacturing facility. The candidate chromosomes are

typically evaluated using discrete-event simulation because the dynamic and stochastic scheduling problems considered are NP-hard. Examples of such studies include that of Lee et al. (1997) and Yang et al. (2007).

Min and Yih (2003b) developed a scheduler by integrating simulation and competitive neural network for the selection of dispatching rules for multiple dispatching decision variables in order to obtain the desired performance measures (i.e. mean of flow time, mean of slack time, and mean of total remaining processing time) given by a user for each production interval. The results of the study are promising based on the results of experiment. The consideration of transportation (via a vehicle-initiated dispatching policy), multiple objectives, and user preferences make this one of the more closely related paper as compared to other review papers in this section.

2.3.1.2.2 Aggregation of Dispatching Rules

There is hardly any single dispatching rule that dominates when more than one conflicting objectives are involved. Hence, emphasis is no longer only placed on the selection of rules but also on the application of rules simultaneously in the form of composite rule or a weighted combination of rules to optimize multiple performance measures of interest. In addition to Min and Yih (2003b), several previous research efforts reviewed in this section are also closely related.

Park et al. (2001) presented a method for simultaneously determining design and control parameters of an FMS with six workstations with the multiple performance objectives via full-factorial design of experiments, regression analysis, compromise programming, and discrete event simulation. Eight design and control parameters

(i.e. number of Automated Guided Vehicles (AGVs), speed of AGV, number of pallets, buffer sizes, routing scheduling rules, loading scheduling rules, sequencing rules, and AGV scheduling rules) are simultaneously determined by compromising four performance measures (i.e. minimizing job tardiness, maximizing system utilization, minimizing in-process inventory, and maximizing a production rate) that are formulated using regression analysis.

In a similar study, Gupta and Sivakumar (2005a, 2005b) proposed the use of Conjunctive Simulation Scheduling and Compromise Programming (i.e. a distance-based technique that could be viewed as a weighted aggregation with two weighting parameters) to achieve Pareto optimal solutions based on three multiobjectives (i.e. delivery accuracy, cycle time, machine utilization) for a single machine configuration. In their later works (Gupta, 2006; Sivakumar and Gupta, 2006), it was further extended to batch machine and a semiconductor back-end respectively.

Dabbas and his coworkers (Dabbas, 1999; Dabbas et al., 2001; Dabbas and Fowler, 2003; Dabbas et al., 2003) re-evaluated the optimal weights assignments of a linear weighted combination of four predetermined dispatching criteria (i.e. line balance, throughput, critical ratio, flow control) to adapt to the factory's changing condition. This approach has the objective of optimizing multiple performance measures using a mixture design of experiments (DOE), multiple response optimization and desirability function.

In a similar study, Lin et al. (2005) studied a parameterized dispatching rule using response surface methodology (with desirability function and geometric mean method) for semiconductor Logic IC sort operations that combines the attributes of

a lot (i.e. hot or normal lot, critical ratio, processing time, set-up times, waiting time, and machine flexibility) to maximize multiple performance measures (i.e. CLIP, CLIP of hot lot, set-up/changeover time, cycle time, and throughput). The use of desirability function approach (e.g. Dabbas, 1999; and Lin et al., 2005) has allowed the easy incorporation of user preferences into decision-making.

The use of weighted combination of rules to optimize multiple performance measures of interest based on the results from the papers reviewed in Section 2.3.1.2.2 have generally shown significant improvement over the use of single dispatching rule. Among the papers, only Dabbas and his coworkers (e.g. Dabbas et al., 2003) have reported a daily re-optimization of weights in an off-line manner in order to take into account the factory changing environment. They also reported that the incorporation of a real-time optimization of the weights within the scheduler is a fruitful area for further investigation.

2.3.1.3 Industrial Applications

Dispatching has been extensively implemented in wafer fabs. Chen et al. (2000) reported the use of a system that combines Manufacturing Execution System (MES), dispatching, and recipe management to reduce cycle time and balance production lines in Macronix International's FAB 1 facility in Hsinchu, Taiwan.

Real Time DispatcherTM (RTD), a commercialized tool from Brooks-PRI Automation, Inc., is implemented in many wafer fabs. RTD works by accessing the status of the shop floor via the Advanced Productivity FamilyTM (APF) Repository and executing the pre-assigned dispatching rule(s) whenever signaled to do so by the MES (i.e. when a shop floor event involving the availability of a resource, such

as a machine or an operator, triggers it). Brooks-PRI Automation's APF Technical Brief (2003) provides more information about the RTD.

Rippenhagen and Krishnaswamy (1998) reported the application of simulation-based scheduling using RTD in a wafer fab of AMD Inc. Similar application of the RTD in Sony's and Intel's semiconductor wafer fabs is reported by Watt (1998) and Tumlinson and Muzumdar (1999) respectively. In the application by Tumlinson and Muzumdar (1999), the dispatching rule used is separated into two parts to reduce the time it takes for RTD to return a result. The complex part runs periodically, and the simpler part is configured to run whenever lot dispatch requests are made.

Texas Instruments DMOS5 wafer fab have replaced their legacy dispatching system with RTD in the photolithography area (Williams et al., 2000) with rules configured to execute every three minutes. Baweja and La (2001) implemented RTD at Kilby FAB at Texas Instruments with dispatch algorithm geared towards a combined effect of keeping the bottleneck from starving through a bottleneck feed factor and prioritizing the lots based on critical ratio. Each lot in the fab is re-ranked every five minutes.

In summary, dispatching has been extensively implemented in practice due to its robustness, low computational requirements, ease of implementation, and intuitive appeal. However, it is also reported in the literature that dispatching rules compliantly sort the lots waiting in the queue based on limited data and is unaware of what is happening around it, both downstream and upstream (Kempf, 1994; Dabbas, 1999). Dispatching, in short, is no guarantee to global performance as it

may wander into local or sub-optimal solution space during execution for a problem of high complexity. Rather, it is more suited at finding consistent solutions with a fast response time.

2.3.2 AGENT-BASED SCHEDULING

Agent-based scheduling typically involves the use of multi-agent systems to solve a scheduling problem. Based on a rather different approach from dispatching, a multi-agent system is a distributed artificial intelligence system that embodies a number of autonomous intelligent agents to achieve common goals (Lim and Zhang, 2003). These agents are typically goal-directed with each trying to optimize their own objective function that may differ from the other agents. Bidding mechanism (e.g. Contract Net protocol) and mediation mechanism are used to resolve the conflicts between different agents. In such a system, each agent interacts and negotiates with each other to bid for jobs and makes decisions that can respond to the uncertainties and changes that occur within their distributed environment. Shen (2002) provided a helpful tutorial on distributed manufacturing scheduling using intelligent agents.

Lin and Solberg (1992) put forth a system where the part agents negotiate with machine agents to dispatch a routing decision based on a price-setting mechanism. The price a part agent pays or the price a resource agent charges is said to depend on the market conditions. Cardon et al. (2000) proposed a dynamic model based on the contract-net protocol with agent negotiations and agent paradigms implemented on parallel machines.

Xue et al. (2001) used the agent-based mechanism involving bidding and mediation mechanisms to allocate available resource and timing parameters values for jobs based on either the earliest-delivery-time-based (forward loading) or due-time-based scheduling (backward loading) policies. Lim and Zhang (2003) extended it further by introducing a framework, which integrates process planning and production scheduling for responsive manufacturing. They reported a preliminary case study involving six machine agents that can deal with unexpected events.

It is not clear how the objectives of the individual agents should be set to ensure good overall system performance. In addition, most of these approaches are tested with relatively few agents. The bids are usually evaluated based on simple information such as cost (Lin and Solberg, 1992), earliest delivery time, or earliest due time (Xue et al., 2001). The number of bids would grow linearly with the number of candidate machines in the system (Baker, 1998). As such, it is possible that the number of machines grows so large that bid congestion occurs in a large complex system like wafer fabrication in which hundreds to thousands of machines, jobs, and operators are common. While Baker (1998) suggested that practical filters could be placed into the system to make sure only machines that are likely to win the bidding actually make the bids, it is unsure if these filters would be outdated, and therefore require monitoring or revision in a highly dynamic and stochastic environment.

2.3.3 PULL-BASED SCHEDULING

Pull mechanisms are used in popular approaches such as Kanban (Krajewski et al., 1987; Martinelli and Valigi, 1998), Constant Work In Process (CONWIP)

(Spearman et al., 1990), and Drum-Buffer-Rope (DBR) (Goldratt and Fox, 1986; Schragenheim and Ronen, 1990). These pull approaches have been motivated by the concept of Just-In-Time (JIT) (Sohal et al., 1989; Weitzman and Rabinowitz, 2003). A pull strategy used the completion of one job to trigger the beginning of another job. Hence, pull systems are considered to be reactive and is a decentralized form of production control.

The most traditional pull algorithm is the (s, Q) policy, where if inventory falls below s , the economic order quantity, Q , of the item is ordered from production. These algorithms are called pull algorithms because production is initiated only after the product is pulled out of inventory (Baker, 1998).

The Kanban system is first developed and used by Toyota. It is based on the use of cards that authorize jobs production and jobs delivery in the plant. Once implemented, production is indirectly scheduled by the use of kanbans and rules governing their flows. Kanban-related studies may involve constant or dynamic (Martinelli and Valigi, 1998) allocation of kanbans for scheduling.

In DBR, the drum represents the pacing of the entire factory to the bottleneck resource. Upstream locations are tied to the bottleneck by a rope, which represents input control to the system. Inputs to the facility and all the workstations upstream from the bottleneck are the buffer that is designed to keep the bottleneck supplied, but not over-supplied (Goldratt and Fox, 1986; Schragenheim and Ronen, 1990; Pyke and Cohen, 1990). DBR can be viewed as an application using the theory of constraints.

While pull mechanisms have inherent inventory and lead time advantages attached to it, pull algorithms may not be appropriate for manufacturing facilities that make diverse products in varying volumes (Morton and Pentico, 1993; Baker, 1998). In adverse scenario, machine failure may even bring the production to a halt. In addition, there is no guarantee that pull systems will perform well under all circumstances. Furthermore, pull mechanisms are unable to consider multiobjectives normally considered in practice. In fact, a pure pull system may not be as favorable as push-pull systems as some researchers have indicated (Pyke and Cohen, 1990; Weitzman and Rabinowitz, 2003).

2.4 HYBRID REACTIVE SCHEDULING (HRS) STRATEGY

Very little is reported in literature relating to Hybrid Reactive Scheduling (HRS) strategies. Most scheduling studies are classified as either Predictive-Reactive Scheduling (PRS) or Completely Reactive Scheduling (CRS).

Matsuura et al. (1993) use an approach based on HRS, which is a hybrid of PRS and CRS strategies. Simulation is used in their study to investigate the dynamic and stochastic job shop scheduling problem by switching between scheduling and dispatching to deal with uncertainties. The uncertainties considered in their study include machine breakdowns, specification changes, and unexpected arrival of rush jobs. A branch and bound technique (without backtracking) generates a schedule at the beginning of the scheduling horizon. This schedule is followed until the occurrence of any one of the above-mentioned uncertainties, after which the approach is switched to dispatching using either the FCFS or the SPT rule.

Dispatching is switched back to scheduling at the end/beginning of each scheduling horizon.

In contrary to Matsuura et al. (1993), other studies (e.g. Roundy et al., 1991; Barua et al, 2005) use a global schedule to guide dispatching to deal with the changing shop floor rather than switch to dispatching. Roundy et al. (1991) proposed a price-directed approach that comprised of a global scheduler and a real-time dispatching module to minimize weighted tardiness for a scheduling problem. When a machine becomes available, the local dispatcher runs a fast algorithm to determine the job to process next based on the costs given by the global scheduler. The global schedule guides the dispatching of jobs by providing information on the relative priority of jobs and there is no requirement for the global schedule to be exactly feasible. They showed that their method performed well to dispatching rules with increasing shop complexity.

In a similar direction, Barua et al. (2005) proposed SB-DIS policy that uses the scheduled start time in the global, optimized-based schedule (generated using a shifting bottlenecks heuristic) to dispatch jobs for processing on machines in a production environment with significant random disruptions. Thus, if the job that should start next under the global schedule is not available, the available job with the earliest planned start time will start next. This is in contrast to the scheduling/rescheduling methods that usually adhere strictly to the global schedule, resulting in the machine becoming idle when the scheduled job is not available for processing.

The SB-DIS policy is evaluated using the due date related objectives (e.g. maximum lateness, average tardiness) and is tested to perform well against FIFO, EDD, and CR. This work offers a promising path to bring into implementation the large body of research on global factory scheduling algorithms that has remained mostly in the academic domain until now. However, further work is required to determine the frequency of recalculating the global schedule that achieves the best trade-off between solution quality and time. In addition, the effects of changing scheduling decisions (i.e. deviating from global schedule) need to be explored to determine how and under what conditions they affect the performance of the factory significantly.

2.5 CONCLUSIONS DRAWN FROM LITERATURE REVIEW

The literature reviewed with the aid of the proposed taxonomy (see Figure 2.1) is by no means exhaustive. Rather, it has provided a framework of the various scheduling methodologies that could be applied in a dynamic and stochastic environment. In an environment where there are fewer uncertainties, optimization-based predictive scheduling algorithms can outperform dispatching algorithms but the latter proves to have a better performance once the uncertainties exceed a certain threshold (Matsuura et al., 1993; Lawrence and Sewell, 1997; Sabuncuoglu and Kizilisik, 2003; Aytug et al., 2005). In fact, the performance of most, if not all, scheduling methods are very much problem/system dependent. Each methodology has its advantages and disadvantages. Table 2.2 summarizes the advantages and disadvantages derived from the scheduling methodologies reviewed.

Table 2.2: Advantages and disadvantages of reviewed scheduling methodologies

Methodology	Advantages	Disadvantages / Drawbacks
Scheduling/ Rescheduling [42 references]	<ul style="list-style-type: none"> Possible to generate optimal or near-optimal schedules Possible to deal with uncertainties and disturbances through rescheduling 	<ul style="list-style-type: none"> Extensive computational required (slow response time) Disregarding events that occur between the rescheduling points may compromise system performance Difficult to determine an optimal rescheduling interval Possibility of entering a permanent state of rescheduling that is triggered by events
Robust Scheduling [8 references]	<ul style="list-style-type: none"> Possible to deal with uncertainties and disturbances with the robust schedule Minimize the need for rescheduling 	<ul style="list-style-type: none"> Ability to absorb impacts of random events only if such events are considered in schedule generation It is not clear if optimality can be achieved and multiobjectives can be considered (since most studies focus on generating robust schedules with simple dispatching rules (e.g. FIFO, SPT, EDD)). It is not known to what degree of uncertainties and disturbances the robust schedule can absorb before a new robust schedule is needed.
Dispatching [45 references]	<ul style="list-style-type: none"> Intuitive (dispatching rules are easy to explain to users) Easy to implement on shop floor Low computational requirements, very fast response time (i.e. responsive to uncertainties and disturbances) Robust (is applicable under all uncertainties) 	<ul style="list-style-type: none"> Simple dispatching rules are generally myopic. Cannot support continuous processing as unnecessary idle time is incurred if next lot selection duration and transportation of the selected job to the machine are non-negligible. In studies involving the selection of rules, the optimal time period to re-select the rules to respond to the changing shop floor conditions is not typically addressed. Real-time optimal weighting of rules have not been addressed.
Agent-based Scheduling [6 references]	<ul style="list-style-type: none"> Agents can respond to uncertainties and disturbances that occur within their distributed environment through interaction and negotiation. 	<ul style="list-style-type: none"> Not clear how objectives of individual agents should be set to achieve Pareto optimal system performance Unsure if agent-based scheduling can achieve near-real-time responsiveness to uncertainties and disturbances May not work in a large complex system as bid congestion may occur since most studies have tested this methodology only for few numbers of agents.
Pull-based Scheduling [10 references]	<ul style="list-style-type: none"> Inherent inventory and lead time advantage 	<ul style="list-style-type: none"> No guarantee that existing pull systems will perform well, with near-real-time responsiveness under uncertainties and disturbances Published works are generally unable to consider multiple objectives that are normally considered in practice.
Scheduling-Dispatching [3 references]	<ul style="list-style-type: none"> Scheduling is switched to dispatching to deal with uncertainties and disturbances when generated schedule is invalidated [Matsuura et al., 1993] If the job that should start next under the global schedule is not available, another available job (based on the global schedule) will be dispatched. [Roundy et al., 1991; Barua et al., 2005] See other advantages from Scheduling/Rescheduling and Dispatching (where applicable) 	<p>Applies to study by Matsuura et al. (1993):</p> <ul style="list-style-type: none"> It is not known if there is an optimal period after which dispatching should be switched back to scheduling It is also not clear under the occurrence of which disturbances will scheduling be switched to dispatching Possibility of scheduling portion being redundant if it can only be implemented only for a very short period of time <p>Applies to all three references:</p> <ul style="list-style-type: none"> It is still not known how and under what conditions the dispatching part of their work will affect the performance of the factory (as opposed to the performance as expected from the global/optimal schedule). See other disadvantages from Scheduling/Rescheduling and Dispatching (where applicable)

As agent-based scheduling methodologies are mostly experimented with only a few agents, this methodology has not been proven to work in a large complex system. Published works of pull-based scheduling methodology are generally unable to consider multiple objectives that are normally considered in practice. In addition, it is not clear if existing agent-based and pull-based systems will perform well with near-real-time responsiveness under uncertainties and disturbances. These make both methodologies unattractive for applications in complex, highly dynamic and stochastic manufacturing facilities such as wafer fabrication.

Scheduling/rescheduling (classified under PRS) is able to achieve the required optimality but not in near real time. The variability in a fab is so high that drives the need for frequent rescheduling to which the extensive computational load required to achieve an optimized schedule makes it unsuitable for near-real-time applications. Additionally, the simplification and assumptions built into the theoretical research problem may result in the approach being unable to account sufficiently for the complexities and uncertainties present in actual facilities. These, according to fab practitioners at AMD (Krishnaswamy and Nettles, 2005), steer the industry towards dispatching. From the review, it is also evident that most near-real-time applications in wafer fabs are achieved with dispatching using proprietary dispatching systems or commercialized tools like RTD.

Dispatching (classified under CRS), on the other hand, is able to achieve near-real-time responsiveness (i.e. is reactive to disruptions and requires a low computation load) but without the required optimality as the pure application of simple dispatch rules is often labeled as a myopic approach that may result in local or sub-optimal solutions. Therefore, it is predicted that an adaptation of certain aspects of the PRS

and CRS methodologies into the proposed Next Task Scheduling (NEXTS) methodology (classified under HRS) would be desirable. For example, the use of global schedules to guide dispatching (as in scheduling-dispatching) or the careful selection and aggregation of rules (as in dispatching) that takes into consideration global system states would be a potential direction for the design of the NEXTS decision policy.

Table 2.3 summarizes the closely related reviewed works (see Sections 2.3.1 and 2.4) in terms of the number of objectives they addressed, whether user preferences could be taken into account, and whether sequence-dependent setups and transportation times are considered in their research.

Table 2.3: Summary of the closely related references

Methodology	Closely Related References	Objectives		Consider User Preferences	Seq-Dep Setups ¹	Tspt Times ²
		Single	Multi			
Completely Reactive Scheduling (CRS) involving the selection of rules	Li and Olafsson (2005); Geiger et al. (2006); Yang et al. (2007)	✓				
	Wang et al. (2005)	✓			✓	
	Pierreval and Mebarki (1997); Tyan et al. (2002); Chen et al. (2004)		✓			
	Min and Yih (2003)		✓	✓		✓
Completely Reactive Scheduling (CRS) involving the aggregation of selected rules	Dabbas (1999); Dabbas et al. (2001); Dabbas et al. (2003); Dabbas and Fowler (2003)		✓	✓ (offline via Desirability Function)		
	Park et al. (2001)		✓	✓ (offline)		✓
	Lin et al. (2005)		✓	✓ (offline via Desirability Function)	✓	
	Gupta (2006)		✓	✓ (offline via generating all Pareto optimal solutions)		
	Gupta and Sivakumar (2005a, 2005b); Sivakumar and Gupta (2006)		✓	✓ (offline via generating all Pareto optimal solutions)	✓	
Hybrid Reactive Scheduling (HRS)	Roundy et al. (1991); Matsuura et al. (1993); Barua et al. (2005)	✓				
	This Research		✓	✓	✓	✓
Remarks: ¹ Seq-Dep Setups: Sequence-Dependent Setups; ² Tspt Times: Transportation Times						

In dispatching studies, only two reviewed studies (Park et al., 2001; and Min and Yih, 2003) have taken the automated material handling systems into consideration. In the majority of the studies, transportation times are assumed to be negligible. The time component is important from the simulation and modeling, and industrial application perspectives (details are further elaborated in Section 3.2) and should therefore be dealt with to some degree of detail in dispatching studies. This time policy portion is lacking in the literature. This explains the intention of this research to divide the proposal of a methodology into two components (i.e. a time policy and a decision policy) to address the importance of both components.

The proposed NEXTS approach, made up of novel time and decision policies, to the best of the author's knowledge, has not been studied before. NEXTS may seem to be just another approach for scheduling but it is favorable as NEXTS is expected to address some of the gaps in the literature and also takes advantage of the partial valuable characteristics of scheduling/rescheduling and dispatching in a modified way. There is, therefore, a motivation to study beyond the deployment of existing approaches to develop a new methodology that considers transportation times, sequence-dependent setups, and user preferences for making Pareto-Optimal scheduling decisions at a machine level in an on-line, near-real-time approach. The proposed approach should be able to deal with the production uncertainties and disturbances effectively and consistently.

CHAPTER 3

TIME POLICIES OF SCHEDULING METHODOLOGIES

Each scheduling methodology identified in Figure 2.1 can be defined by a time policy and a decision policy. The time policy and decision policy generally refer to ‘*when to make scheduling decisions*’ and ‘*how to make scheduling decisions*’ respectively. This chapter focuses on the time policy aspects of scheduling methodologies. Section 3.1 examines the existing periodic scheduling/rescheduling (PSR) and dispatching time policies. The importance of the time component from the simulation (and modeling) and industrial application perspectives is then highlighted in Section 3.2. After studying the strengths and weaknesses of the existing time policies and the gaps between the existing literature and the needs of actual implementation, a conceptual time policy for use in the NEXt Task Scheduling (NEXTS) methodology is proposed in Section 3.3. The conjecture that the proposed time policy will offer a better performance than existing time policies is shown via an illustrative example in Section 3.4. The conjecture is further proven in Section 3.5 using a discrete event simulation study.

3.1 EXISTING PSR AND DISPATCHING TIME POLICIES

The time policy stipulates the timing for the execution of the decision policy to obtain the required scheduling decision(s). This section begins by first introducing the time policies used in PSR before the time policy used in dispatching is described.

3.1.1 PSR TIME POLICY

Figure 3.1 illustrates the PSR time policy. The points in time where rescheduling are performed are referred to as rescheduling points (Aytug et al., 2005) and are denoted by the symbol ' $\triangle$ ' in Figure 3.1. The time period taken to generate a schedule is termed as the schedule generation duration and is denoted as the duration between ' $\triangle$ ' and ' $\blacktriangle$ '. The horizon for which the schedule is generated is referred to as the scheduled horizon. The duration during which the schedule is applied is termed as the schedule usage horizon and the demarcation of a schedule usage horizon is indicated by two consecutive ' $\blacktriangle$ '. Periodic rescheduling cycle refers to an interval of time during which rescheduling is completed and restarted (i.e. interval between two rescheduling points marked by two consecutive ' $\triangle$ ').

Figure 3.1: PSR methodology's time policy

In the context of scheduling/rescheduling, the term 'time policy' refers to 'when to reschedule'. In PSR, the granularity of the decision made is for a set of tasks (whereas NEXTS and dispatching are concerned with scheduling only the next task each time the decision policy is executed). The dotted directional connector '----->'

in Figure 3.1 denotes when the scheduled tasks (in the form of a schedule) are processed as the shop floor evolves.

As the name PSR (periodic scheduling/rescheduling) suggests, scheduling decisions (in the form of a schedule) are made periodically. Although PSR methodology can be used to reschedule the facility periodically at a time interval that may be fixed or varied, Fowler et al. (2001) reported that PSR is typically used to develop optimal or near-optimal production schedules based on a fixed time interval of an entire shift (or day). The frequency of rescheduling is determined by the inverse of periodic rescheduling interval.

In actual implementation of PSR, the length of the schedule generation duration is dependent on (i) the duration it takes to take a snapshot of the factory status data, (ii) the duration to preprocess the collected data and generate a model for scheduling/rescheduling, (iii) the duration it takes for the scheduling/rescheduling procedures to generate a schedule (this depends on the complexity of the scheduling algorithms applied and the length of the scheduling horizon over which the generated schedule is applied), and (iv) the duration to upload or communicate the schedule to the shop floor for execution. A breakdown of the schedule generation duration into the four above-mentioned durations is shown in Figure 3.2.

Although schedule generation starts at ' $\triangle$ ', the new schedule would not be ready for use in the manufacturing shop floor until instance ' $\blacktriangle$ '. Hence, during the schedule generation duration (between ' $\triangle$ ' and ' $\blacktriangle$ '), the schedule of the previous periodic rescheduling cycle is followed. Due to a long schedule generation duration involved (in the range of two hours as observed in a back-end semiconductor

assembly plant in Singapore), some of the stochastic events that occur during this schedule generation duration may not be captured since the on-line data may have already been retrieved before their occurrence. This is one of the main drawbacks of PSR as it could result in suboptimal performance measures, or worst still, result in a generated schedule that could not be followed on the shop floor.

Figure 3.2: A breakdown of the schedule generation duration

For example, a job would already have been committed (during the schedule generation duration) to a machine for processing based on the binding decisions specified in the schedule of the previous periodic rescheduling cycle but this job may still be scheduled to be processed by yet another machine in the new schedule. In another example, the frozen jobs (i.e. jobs that have been assigned to machines earlier for processing, that may not be moved or preempted – Pinedo, 2002) may have completed earlier than expected but the new schedule still reflect this job as in the processing mode. These inconsistencies will confuse the operators and/or systems.

In cases where the job that should start next under the global schedule is not available, a decision has to be made whether to wait or to process the available job with the earliest planned start time. Waiting for the job to become available would result in the machine becoming idle. On the other hand, choosing to process the

next available job may also have an adverse impact on the performance measures since sequence-dependent setup may have been considered in the global schedule generation and a change in the sequence may require two setups instead of one (i.e. longer setup times incurred).

3.1.2 DISPATCHING TIME POLICY

Dispatching involves making near-real-time decisions as to which job from a set of jobs in queue is to be selected for processing at a machine when the machine is available. In dispatching, decisions are typically made much more frequently than in PSR since a job dispatch decision is made every time a machine becomes available. This also implies that the dispatching rule(s) is applied to select the next job from the available jobs in queue at the machine availability instance denoted by ‘ $\triangle$ ’ (see Figure 3.3). The job selection duration (if any) is indicated by the duration between ‘ $\triangle$ ’ and ‘ $\blacktriangle$ ’.

Figure 3.3: (a) Dispatching methodology's time policy with negligible next job selection duration, and (b) Dispatching methodology's time policy with next job selection duration

Simple heuristics (dispatching rules) such as first-in, first-out (FIFO), earliest due date (EDD), shortest processing times (SPT), same setup (SSU), etc., are typically used in the selection of the next job. When simple heuristics are used, the selection can be achieved in a short or negligible time. In this scenario, ‘ $\triangle$ ’ and ‘ $\blacktriangle$ ’ would come closer or coincide as shown in Figure 3.3(a). For superior rules that are slightly more computationally intensive, the selection duration is reflected in Figure 3.3(b).

The longer the selection and transport durations, the longer the machine idle time (see unshaded region in Figure 3.3). These idle times are typically not considered in the simulation of these rules. Hence, the application of superior rules may prove to be more detrimental than beneficial to the system performance due to a longer computation time (and therefore longer idle time) that effectively decreases the machine available capacity. The assumption of negligible transport duration when it is not valid will also lead to the same negative effect.

3.2 IDENTIFYING THE GAPS BETWEEN LITERATURE AND ACTUAL IMPLEMENTATION

In on-line, near-real-time application, it is necessary to be able to make a next lot selection decision within the available time (Dalal et al., 2003). Although the application of simple rules/heuristics to select the next lot (or job) may consume negligible decision time, application of complex rules/algorithms may be much more computationally intensive. Global rules that require the on-line retrieval of information of other lots or machine status may also have non-negligible decision time. These rules, when simulated for their performance using a discrete-event simulation software/program, are typically executed at a particular simulated time

(i.e. when the machine is/becomes available) with decision also being made at the same simulated time. Hence, the decision delays only show up with a longer simulation runtime but are not reflected in the performance measures (e.g. an increase in cycle time or a decrease in capacity caused by the need for the machine to wait in idle mode for the decision). This also implies that the simulation model is not a true representation of the actual implementation. The length of machine idleness can be further worsened (lengthened) if delivery involving non-negligible transportation duration is required after the decision is made and communicated.

From the industrial application perspective, the time component also needs to be dealt with to some degree of detail as it is important not only for automation purposes but also from a product, equipment, and process perspective (Krishnaswamy and Nettles, 2005). With negligible lot transport time (or delivery duration), it is intuitive to carry out the next lot selection at the latest possible time, that is, when the machine becomes available. In practice, transport times (or durations) between two different locations need not be negligible as observed in wafer fabrication plants and in studies that involve lot transportation or Automated Material Handling System (AMHS).

In 300mm fully automated wafer fabrication plant, the machine has input-output automation through the incorporation of loadports (Krishnaswamy and Nettles, 2005). The loadports tend to provide a small buffer capacity for the machine and allow for continuous processing. For example, actions to bring a selected lot to a loadport can happen when another lot is being processed on the machine. This is to avoid incurring unnecessary machine idle time while waiting for lots to be delivered (Wright et al., 2002). This requires the scheduling of the next lot before

machine is available, which is generally impossible to accomplish with existing dispatching time policy (Krishnaswamy and Nettles, 2005). This is because the dispatching time policy selects the next lot only when the machine becomes idle and this is considered as an action performed too late.

In the case of PSR (periodic scheduling/rescheduling), the next lot selection decision is made in advance of implementation in the form of a generated schedule. By referencing the generated schedule, the next lot to be processed by a particular machine is immediately available. Hence, there is no problem transporting the lot to the machine to ensure continuous processing (provided the scheduled lot is available). However, the validity of the generated schedule is highly dependent upon the dynamism of the shop floor over which the schedule is applied. The more dynamic the shop floor, the faster the generated schedule deviates from the actual shop floor status (Vieira et al., 2003). In addition, PSR typically do not consider the effects of changing scheduling decisions on the performance of the shop floor. Hence, PSR may not be suitable for on-line, near-real-time implementation in highly dynamic environment.

In order to take into account the non-negligible transportation times, it is intuitive to bring the next lot selection forward in time so that the next lot selection and transportation can be carried out before the machine becomes available. Fargher et al. (1994) reported that committing an event a fixed amount of time in advance of the estimated start of processing is complicated because some of the lots that are committed might get scrapped before they arrive at the machine. It is, however, not stated in the paper how far in advance the events are committed because, intuitively, an event that is committed well ahead of its occurrence (e.g. in the form

of a schedule) could easily be invalidated by stochastic events in a dynamic environment. The conjecture is therefore to allow the decision-making (i.e. selection of the next lot) to be committed at the latest possible time (before machine is available) so that the decision has a much lower possibility of being invalidated. Based on the conjecture, a basic conceptual time policy for use in the proposed NEXTS methodology is presented in the next section.

3.3 BASIC CONCEPTS OF NEXTS TIME POLICY

In this section, a basic concept of the proposed NEXTS time policy is presented based on the conjecture that if the next lot selection is scheduled to occur before machine is available, it may reduce the unnecessary idleness that will result if next lot selection and transportation (if any) are performed only when the machine is available as in dispatching. In addition, the next lot selection will only be executed as close as possible to the time when a decision is required so as to reduce the likelihood of changing factory status or conditions invalidating the decision (as compared to PSR).

Whenever the machine begins to process a new job, the next ' Δ ' instance (refer to Figure 3.4) can be estimated by subtracting the selection and transportation duration from the expected processing completion time of this new job. It is assumed that the selection and transportation durations can be well estimated with their respective sample means obtained from the data repository in a wafer fab that is constantly updated with the shop floor status. In certain operations with nearly deterministic processing times (Johri, 1993), the completion time can be determined with certainty. In cases where processing time varies, it is still possible to keep

track of the progress of the processing to determine the probable/expected remaining processing time required to complete the processing.

Figure 3.4: NEXTS methodology's time policy

The next lot selection begins at instances marked by 'Δ' in Figure 3.4 (before machine is available) instead of when the machine is available (compare with the instances of 'Δ' in Figure 3.3). When the next lot is selected, if transportation is necessary, this requirement could be communicated at instance '▲' to the manufacturing personnel or to the automated transporter to move the selected lot to machine. When the next lot starts processing on the machine, the process repeats with the focus placed on determining the next 'Δ' instance to start the decision-making (i.e. selection of the next lot for machine).

3.4 PRELIMINARY COMPARISON OF TIME POLICIES USING AN ILLUSTRATIVE EXAMPLE

In order to understand the performance of the basic NEXTS time policy (refer to Section 3.3), a comparison will be made against the commonly used periodic scheduling/rescheduling (PSR) and dispatching time policies (see Section 3.1).

A simple one-machine problem with unexpected arrival of an urgent job (i.e. job with a tight due date) is presented for a preliminary comparison of the three time

policies. In wafer fabrication, unexpected arrival of an urgent job is common; this illustration is therefore representative of an actual situation. The same Earliest Due Date (EDD) rule is applied across all the three methodologies (i.e. PSR, Dispatching, NEXTS) and no pre-emption (i.e. interruption) is allowed. In this example, it is assumed that there are three jobs waiting in queue (i.e. Job 26, Job 78, and Job 17 specified in increasing order of time to due date), and that an urgent job (i.e. Urgent Job 10 – with earliest due date among the four jobs) arrives unexpectedly halfway through the processing of Job 26. The effect of the time policy used in PSR, dispatching and NEXTS is illustrated in Figure 3.5(a), (b), and (c) respectively.

(a) Effect of PSR time policy on the unexpected arrival of an urgent job

(b) Effect of dispatching time policy on the unexpected arrival of an urgent job

(c) Effect of NEXTS time policy on the unexpected arrival of an urgent job

Figure 3.5: Effect of (a) PSR time policy, (b) dispatching time policy, and (c) NEXTS time policy on the unexpected arrival of an urgent job

It can be seen from Figure 3.5(a) that PSR is unable to react to the unexpected arrival of Urgent Job 10 as this urgent job is taken into consideration only at the next rescheduling point unless manual interruption is carried out. The lack of responsiveness to unexpected occurrence of events is a drawback of PSR. In practice, job selection duration, transportation duration, and processing duration are usually non-negligible. Assuming that job selection duration and transportation duration are incurred for both dispatching (see Figure 3.5(b)) and NEXTS (see Figure 3.5(c)), it is not difficult to observe the drawback of performing job selection and transportation only upon machine availability (as in dispatching) in contrast to performing them prior to and closest to the point when the machine is available (as in NEXTS).

The makespan (completion time of the last job to leave the system) of dispatching is longer than that of NEXTS and PSR. Although the makespan for NEXTS and PSR is the same, the former is expected to perform better in terms of due date objectives, as the former is more reactive to the unexpected arrival of Urgent Job 10 and is able to process this urgent job immediately after Job 26 is completed. Through this example, it is shown that the proposed time policy for NEXTS has the potential to outperform the existing time policy used in dispatching and PSR. Recognizing that this single scenario does not provide sufficient statistical evidence to prove the performance of the proposed time policy, a simulation-based study involving various levels of disruptions is presented in the next section.

3.5 COMPARISON OF TIME POLICIES USING DISCRETE EVENT SIMULATION

As a mathematical model cannot accurately describe the complex interactions between various stochastic elements of wafer fabrication, simulation is used to obtain useful performance estimates in such system. Discrete event simulation is used to model the system as it evolves over time by a representation in which the state variables change instantaneously at different points in time.

A discrete-event simulation-based analysis is carried out to compare the differences between the system performance of using the NEXt Task Scheduling (NEXTS), Periodic Scheduling/Rescheduling (PSR), and dispatching time policies in a dynamic and stochastic environment with different levels of variability (i.e. disturbances). Fictitious data that is representative of the actual wafer fabrication data is used in this study.

3.5.1 OBJECTIVE & ASSUMPTIONS

The broad goal of this simulation study is to evaluate and compare the proposed NEXTS time policy against the commonly used PSR and dispatching time policies. In order to achieve the objective of our current simulation study, certain details are considered not essential and are excluded from the model. Some of these details (e.g. sequence-dependent setup time) would, however, be considered in subsequent chapters. The assumptions considered are:

- Operators, handlers, and hardware are not explicitly modeled in the study.
- Constant lot sizes (same number of wafers in each lot).
- No yield losses are considered.

- No rework is modeled.
- No sequence-dependent setup time.
- No preemption is allowed.

The selection duration involved in NEXTS and dispatching, the schedule generation duration for PSR, and the transportation duration (for all the three methodologies) are assumed to be known deterministically.

3.5.2 PROBLEM DESCRIPTION

As the respective time policy is applied in the same way whether there are one or many machines, a one-machine problem can be seen as an appropriate decomposition or a building block of a larger system. Though simple, the single machine problem is sufficient to provide valuable insights and statistical evidence concerning the performance of the three time policies.

This study involves a dynamic and stochastic single machine system that is subjected to varying levels of variability and disturbance in which lots with different types and sizes arrive dynamically for processing. The perturbation of the system is classified into five different levels with Level 1 having the lowest degree of variability (with reference to lots arrival time) and disturbance (with reference to MTTF) and Level 5 having the highest degree of variability and disturbance. The sequences in which the lots are processed are determined with a commonly used shortest processing time (SPT) dispatching rule. When there is a tie, First-In First-Out (FIFO) is applied. The same dispatching rules (SPT, then FIFO) are applied across all the methodologies.

In this study, four lot types (i.e. lot type I, II, III, and IV) are considered and released to the machine in proportional amounts. In PSR, the schedule is generated well ahead of time and as a result, the actual lot arrivals are not known in advance. As such, anticipated arrivals are used to generate the schedule. In this study, it is assumed that the anticipated lots' arrival times are computed based on the anticipated lots inter-arrival times that are taken to be independent and identically distributed (IID) exponential random variables with a mean of 45 minutes (fictitious data in the proper range based on actual wafer fabrication data). IID means that the inter-arrival times are independent of lots and that the inter-arrival times have the same probability distribution.

In a real setting, the anticipated arrivals are seldom realized and it is common that lots arrive earlier or later than expected. Hence, it is necessary to perturb the anticipated arrival times to account for the unexpected early and late arrival of lots. The actual simulated arrival time is computed by adding a random number (generated from a uniform distribution) to the anticipated arrival time (generated from an exponential distribution with mean 45 as discussed in the preceding paragraph). The random numbers (added to the anticipated exponentially distributed arrival times) introduced in the five levels (i.e. Level 1, 2, 3, 4, and 5) determines the degree of uncertainty and are uniformly generated from $U[-10, 10]$, $U[-60, 60]$, $U[-110, 110]$, $U[-160, 160]$, and $U[-210, 210]$ respectively.

As no sequence-dependent setups are considered, it is not necessary to consider process times and setup times separately. In this study, the processing times used are assumed to be inclusive of the setup times. The processing times, modeled using uniform distribution, are dependent on lot types but are IID for each lot type.

Each lot type has a different mean processing times (i.e. 50, 45, 40 and 35 minutes for lot type I, II, III and IV respectively). Again, the actual simulated processing times are not known in advance, hence, lots are ranked using the SPT rule based on their respective mean processing times. An interval of $U[-9, 9]$ is added to the mean processing times to account for operators' efficiencies in performing the setups (which include loading and unloading).

Machine breakdowns are also modeled. The time to failure and time to repair are random variables exponentially distributed with means denoted by MTTF (mean time to failure) and MTTR (mean time to repair) respectively. In this study, MTTR is taken to be 120 minutes. MTTF used in the five levels of disturbance (i.e. Level 1, 2, 3, 4, and 5) are 4320, 3600, 2880, 2160, and 1440 minutes respectively. This simply implies that machine fail less often at Level 1 (less disturbances) and fail more often at Level 5 (more disturbances) over a given period of time.

The values and distributions that are used for the three modeled methodologies and the factors that are being perturbed are summarized in Table 3.1 and Table 3.2 respectively.

3.5.3 REQUIRED PROCESSING CAPACITY vs AVAILABLE MACHINE CAPACITY

The time to failure and time to repair are random variables exponentially distributed with means denoted as $MTTF$ and $MTTR$ respectively. Hence, the machine availability A will be:

$$A = \frac{MTTF}{MTTF + MTTR} \quad (3.1)$$

The available machine capacity α is given by:

$$\alpha = MA \quad (3.2)$$

where M is the machine capacity in lots per unit time. Since this is a single machine problem and the machine can process one lot at a time, this machine is modeled as unary resource (i.e. resource of unit capacity, $M = 1$). Table 3.3 shows the computation of the available machine capacity α for the five levels of variability and disturbance.

Table 3.1: Input variables with their corresponding values and/or distributions

Variables / Parameters	Values / Distributions
Number of lot types	4
Dispatching rule	SPT followed by FIFO
Processing time (duration)*	Type I: $U[41,59]$ is equivalent to $50 + U[-9,9]$ Type II: $U[36,54]$ is equivalent to $45 + U[-9,9]$ Type III: $U[31,49]$ is equivalent to $40 + U[-9,9]$ Type IV: $U[26,44]$ is equivalent to $35 + U[-9,9]$
Expected Inter-arrival Time*	EXPO(45)
MTTR*	EXPO(120)
Selection Duration (for NEXTS & Dispatch)*	1 (Ideal case: 0)
Schedule Generation Duration (for PSR)*	60 (In view of the presence of improved PSR time policy: 1 and 25 minutes are included for a few cases; Ideal case: 0)
Transportation Duration*	4 (2 minutes is included for a few cases; Ideal case: 0)
NOTE: * Values are specified in minutes. EXPO(x) – Exponential Distribution with Mean = x and Variance = x^2 $U[L,U]$ – Continuous Uniform Distribution with Lower and Upper Bound given by L and U respectively.	

Table 3.2: Degree of perturbation applied to arrivals and MTTF under different levels of variability and disturbance

Factors Perturb:	In Increasing Level of Variability & Disturbance				
	Level 1	Level 2	Level 3	Level 4	Level 5
1) Arrivals	Expected Arrivals +				
	$U[-10,10]$	$U[-60,60]$	$U[-110,110]$	$U[-160,160]$	$U[-210,210]$
2) MTTF	EXPO(4320)	EXPO(3600)	EXPO(2880)	EXPO(2160)	EXPO(1440)

Table 3.3: Available machine capacity under different levels of variability and disturbances

	Level 1	Level 2	Level 3	Level 4	Level 5
MTTF	4320	3600	2880	2160	1440
MTTR	120	120	120	120	120
α	0.973	0.968	0.960	0.947	0.923

The mean processing time p for each lot type j is denoted as p_j . Since each lot type is equally likely, the average of the mean processing times $\bar{p}$ can be computed as:

$$\bar{p} = \frac{\sum_{j=1}^J p_j}{J} = \frac{50 + 45 + 40 + 35}{4} = 42.5 \quad (3.3)$$

where J is the number of lot types. As mentioned in Section 3.5.2, the inter-arrival times are modeled with an exponential distribution with mean $\tau = 45$. In this case, the required processing capacity γ can be calculated as follows:

$$\gamma = \frac{\bar{p}}{\tau} = \frac{42.5}{45} = 0.944 \quad (3.4)$$

It should be noted that the degree of disturbance in Level 5 is introduced with the required processing capacity ($\gamma = 0.944$) exceeding the available machine capacity ($\alpha = 0.923$) on purpose. This inclusion of this level will help to determine the system performance of the three time policies under such a condition if it should occur within a specified period.

3.5.4 METHODOLOGY EVENT-BASED REPRESENTATION

This section outlines how the above-mentioned problem using PSR, dispatching, and NEXTS is computer-represented. The discussion presented in this section is independent of the particular general-purpose programming language or discrete event simulation tools to be used. Without customization, commercial simulation tools are typically unable to simulate the timing policies applied in NEXTS and PSR methodologies. Hence, the event-based representations for the three scheduling methodologies are provided as a guide for customization.

The differences exhibited by a one-machine system modeled based on PSR, dispatching, and NEXTS methodologies are depicted by their event-based representation as shown respectively in Figure 3.6, Figure 3.7, and Figure 3.8.

Nodes in the figures represent events and the notations used are:

- SS – selection start / SE – selection end
- TS – transportation start / TE – transportation end
- PS – processing start / PE – processing end
- PSRT – periodic scheduling/rescheduling time
- SGS – schedule generation start / SGE – schedule generation end

Figure 3.6: Event-based representation of PSR

Figure 3.7: Event-based representation of dispatching

Figure 3.8: Event-based representation of NEXTS

The following descriptions of the event-based notations are made with respect to Figure 3.6, Figure 3.7, and Figure 3.8. Apart from these events, lot release and down events are also scheduled but are not shown in the figures for clarity of explanation. The first lot release and down events are scheduled during the start of the simulation, with subsequent of these events scheduled from the occurrences of their respective events. Solid arrows indicate that an event (represented as nodes) at the end of the arrow may be scheduled from the event node at the beginning of the arrow. For example,

$\textcircled{A} \rightarrow \textcircled{B}$ implies that event A will schedule the occurrence of event B, and

$\textcircled{A} \rightarrow \textcircled{A}$ implies that event A will reschedule itself.

Dotted arrows provide a form of conditional check to ensure that the event at the end of the arrow can be executed only upon the occurrence of the event at the beginning of the arrow. For example,

$\begin{matrix} \textcircled{A} \\ \textcircled{B} \end{matrix} \rightarrow \textcircled{C}$ implies that event A will schedule the occurrence of event C but event C will be executed only upon the occurrence of event B.

In this simulation study, efficiency of the simulation program is not sought after as much as the logical accuracy of the conceptual and programmed model. Some of

the events, though redundant, are introduced in the modeling to facilitate verification and validation through the use of ‘trace’ and ‘walk-through’ techniques.

The modeling of a system based on the dispatching methodology (see Figure 3.7) is the simplest and is typically achievable in most commercial discrete-event simulation tools with typically no customization. Since SS precedes SE, and SE precedes TS, etc. in dispatching, the sequence can be represented by $SS \prec SE \prec TS \prec TE \prec PS \prec PE$.

The difference between the NEXTS and dispatching time policy lies in when the SS event is scheduled. While the SS event is scheduled by the PS event in simulation, it should be noted that the SS event is not scheduled by the PS event in actual implementation. In NEXTS (see Figure 3.8), the SS event is scheduled such that TE event occurs immediately before or immediately after the PE event. In both dispatching and NEXTS, the SPT-FIFO rule is applied in the event routine of SS to select the next lot.

In the case of PSR (see Figure 3.6), the PSRT, SGS and SGE are the events that are used for schedule generation while the remaining events are used to simulate the implementation of the schedule. The PSRT event reschedules itself based on the rescheduling interval and this is used to keep track of the simulated time to perform schedule generation. In PSR, the SPT-FIFO rule is applied in the event routine of SGS. The scheduled lots (based on the schedule) are assigned an attribute (with ‘1’ assigned to the first lot in the schedule and incremented for the successive scheduled lots based on their processing order). When the TS event routine looks

for the next lot to be transported, it selects the lots (in queue) with the smallest attribute value (a lot without this attribute value implies that it has not been scheduled and will therefore not be considered in the selection). These attribute values are updated in the event routine of SGE (that is when the schedule is generated).

3.5.5 VERIFICATION & VALIDATION

The simulation program is constructed using Microsoft® Visual Basic 6.0 and Microsoft® Excel 2000. This program is debugged and verified in a number of iterations under a variety of settings of the input parameters using the ‘trace’ technique (Law, 2007). In a trace, the state of the simulated system (e.g. the contents of the event list, the state variables, certain statistical counters, etc.) are displayed just after each event occurs and are compared with hand calculations to check if the program is operating as intended. A ‘structured walk-through’ technique (Law, 2007) was performed at several meetings with Chartered Semiconductor Manufacturing Ltd. to further increase the validity and credibility of the conceptual model.

3.5.6 EXPERIMENTAL RUNS

Since the SPT rule minimizes the average cycle time for single machine problem, the three time policies shall be compared using average cycle time. As mentioned in Section 3.5.2, there are five levels of variability and disturbances over which NEXTS, PSR, and dispatching methodologies (time policies, to be more specific) are compared. For PSR, five levels of rescheduling interval varying from 60 minutes to one-day period (i.e. 60, 405, 750, 1095, 1440 minutes of rescheduling

interval), with schedule generation duration of 60 minutes (i.e. G60) and transportation duration of 4 minutes (T4) are investigated. The above cases are denoted as PSR_R60_G60_T4, PSR_R405_G60_T4, PSR_R750_G60_T4, PSR_R1095_G60_T4, and PSR_R1440_G60_T4 respectively.

In recent years, PSR policies have been able to achieve some major reductions in schedule generation time; for instance, Upasani et al. (2006) have shown that with appropriate problem modeling, the CPU time to use a global scheduling algorithm can be reduced by several orders of magnitude. In view of these recent progresses, another three sets of experiments are added (namely PSR_R25_G25_T4, PSR_R60_G1_T4, and PSR_R25_G1_T4) to facilitate understanding of how these improved PSR time policies compare to the NEXTS time policies under the application of the same decision policy. Since the minimum processing time for all lot types is 26 minutes, rescheduling interval of 25 minutes (i.e. R25) is considered. Reducing the rescheduling interval further (very frequent rescheduling) will result in extremely lengthy simulation runs and are therefore not considered.

As the PSR simulation runs are generally lengthy, transportation time of two minutes are only repeated for the NEXTS and PSR_R25_G1 cases (denoted as NEXTS_G1_T2 and PSR_R25_G1_T2 respectively). In addition, the ideal case where schedule generation and transportation times are zero minutes (denoted as Ideal_G0_T0) is also included to reflect the best possible average cycle times for each of the five levels.

As these simulation models characterize stochastic problem situations, 10 replications (with different seeds) are made to obtain acceptable confidence in the

output measures (i.e. mean cycle time). The same seeds are used for the different methodologies to allow three time policies to be compared in a more statistically efficient manner. In order to ensure unbiased determination of a warm-up period, ten replications of simulation run are considered for each of the NEXTS and PSR models (using schedule generation time of 60 minutes) with variability and disturbances ranging from Level 1 to 3 using Welch's graphical procedure (Law, 2007). The moving averages for WIP of these models are plotted in

Figure 3.9.

Figure 3.9: Moving averages for WIP

For models with Level 4 variability and disturbances, a very long warm-up period is expected as the required processing capacity ($\gamma = 0.944$) approaches the available machine capacity ($\alpha = 0.947$). As the main purpose of this simulation is to obtain a sense of difference in magnitude with respect to the baseline case (NEXTS time policy), and an indication of directions of improvement in system performance, a longer warm-up period is not sought after.

For models with Level 5 variability and disturbances, steady-state conditions will never be reached as the required processing capacity ($\gamma = 0.944$) exceeds the available machine capacity ($\alpha = 0.923$) in Level 5. Due to the inefficiency of the Dispatching time policy which resulted in 5 minutes (4 minutes of transportation duration and 1 minute of scheduling duration) of extra idle time, steady state will never be reached because arrival rate (i.e. $60/\text{interarrival time} = 60/45 = 1.33$) becomes larger than service rate (i.e. $60/(\text{processing time} + \text{idle time}) = 60/(42.5+5) = 1.26$). Hence, Level 5 models and dispatching models are not included in the determination of the warm-up period.

As observed from
Figure 3.9, the system stabilizes after 6000 minutes. Hence, a warmup period of 6000 minutes is used across all models. Statistical data is collected for 300 lots after statistical counters are reset (i.e. after warmup). A total of 650 experimental runs are carried out:

$$(2 \text{ NEXTS model} + 1 \text{ Dispatching model} + 9 \text{ PSR models} + 1 \text{ Ideal model}) \times (5 \text{ levels}) \times (10 \text{ replications}) = 650 \text{ runs}$$

The input data for these experimental runs can be found in Section 3.5.2.

3.6 EXPERIMENTAL RESULTS AND DISCUSSION

The three methodologies (PSR, Dispatching, and NEXTS) differ in terms of the time policy used. Since the same dispatching rule (SPT) is applied in this simulation study, the simulation results provide a comparison of the three methodologies based on purely the time policies applied. Table 3.4 presents the mean and 95% confidence interval (C.I.) for the average cycle time.

Table 3.4: Mean and 95% C.I. for the average cycle time

	In increasing level of variability and disturbance
	Level 1	Level 2	Level 3	Level 4	Level 5
Variability Introduced to Expected Arrivals:	UNIF(-10, 10)	UNIF(-60, 60)	UNIF(-110, 110)	UNIF(-160, 160)	UNIF(-210, 210)
MTTF Disturbance:	EXP(4320)	EXP(3600)	EXP(2880)	EXP(2160)	EXP(1440)
Ideal_G0_T0	344.08 ± 160.44	355.61 ± 152.84	373.58 ± 154.54	402.73 ± 144.23	497.15 ± 184.19
NEXTS_G1_T2	349.21 ± 154.96	360.94 ± 153.67	380.13 ± 151.23	410.75 ± 148.34	507.55 ± 188.87
NEXTS_G1_T4	350.95 ± 155.81	370.34 ± 152.79	394.63 ± 150.78	433.94 ± 147.95	574.65 ± 206.64
PSR_R25_G1_T2	355.51 ± 162.29	363.63 ± 156.55	384.54 ± 156.97	416.16 ± 146.22	512.86 ± 189.83
PSR_R25_G1_T4	357.32 ± 162.36	369.67 ± 156.54	400.31 ± 150.06	446.51 ± 161.68	585.87 ± 206.07
PSR_R25_G25_T4	358.4 ± 162.07	371.76 ± 156.12	400.95 ± 148.33	449.01 ± 161.26	588.79 ± 205.8
PSR_R60_G1_T4	373.05 ± 168.68	377.13 ± 158.81	405.13 ± 149.82	452.65 ± 156.78	591.28 ± 203.69
PSR_R60_G60_T4	374.20 ± 166.51	389.10 ± 150.15	414.95 ± 148.35	454.43 ± 148.19	601.24 ± 212.69
PSR_R405_G60_T4	433.20 ± 155.95	457.11 ± 160.16	486.30 ± 141.88	523.17 ± 148.31	640.84 ± 202.90
PSR_R750_G60_T4	561.99 ± 178.52	571.35 ± 181.01	606.37 ± 220.99	642.29 ± 170.39	689.43 ± 203.16
PSR_R1095_G60_T4	676.17 ± 258.10	743.12 ± 245.49	750.00 ± 276.72	752.51 ± 173.83	879.12 ± 217.28
PSR_R1440_G60_T4	879.43 ± 305.21	843.68 ± 275.18	816.96 ± 292.67	886.25 ± 194.99	965.62 ± 269.14
Dispatching_G1_T4	894.49 ± 341.03	912.53 ± 321.96	921.26 ± 317.00	972.75 ± 323.89	1069.1 ± 351.28

When rescheduling interval reduces from 1440 minutes to 25 minutes, mean cycle time decreases. As uncertainties and disturbances tend to make the schedule ineffective, frequent rescheduling has the effect of keeping the schedule updated with the shop floor status to allow the schedule to be followed to produce the intended performance (i.e. better cycle time performance). Shafaei and Brunn (1999b) also reported similar observation of better performance with frequent rescheduling.

A reduction in rescheduling interval (e.g. PSR_R60_G1_T4 to PSR_R25_G1_T4) results in larger improvement in mean cycle times as compared to a reduction in schedule generation time (e.g. PSR_R60_G60_T4 to PSR_R60_G1_T4, and PSR_R25_G25_T4 to PSR_R25_G1_T4). This arises because transportation in PSR is generally not incurred as scheduling decisions (in the form of a schedule) are already created and the next scheduled lot (already known) would be

transported ahead of time to the machine for processing. This will avoid cases of machine having to wait idling for the next lot to arrive.

It is also observed that the mean cycle times of PSR_R25_G1_T2/4 (i.e. rescheduling interval of 25 minutes with schedule generation time of 1 minute) approaches that of NEXTS. PSR_R1_G1 is not included in the experiments because the rescheduling at an interval of 1 minute is too frequent and this will result in extremely lengthy simulation runs. However, it is expected that the mean cycle time will approach that of NEXTS when the rescheduling interval is reduced to every 1 minute provided the same decision policy is applied. The percentage improvement of the mean cycle time using NEXTS (as compared to PSR and dispatching) are computed in Table 3.5 and are plotted in Figure 3.10.

A positive value implies that the NEXTS time policy has a better performance over the time policy that it is compared against. It is observed from Table 3.5 that the NEXTS time policy outperforms the PSR and dispatching time policies under most levels of variability and disturbance considered. As seen from Figure 3.10, the improvement is relatively constant across Level 1 to Level 4. However, a slightly smaller improvement is observed in Level 5. As mentioned in Section 3.5.6, the required processing capacity exceeds the available machine capacity. As such, the unsteady condition may have affected the magnitude of the results. In addition, the shortage of capacity could have imposed a limit on how much the NEXTS time policy can perform and this reasonably explains the smaller improvement over the PSR and dispatching time policies.

Table 3.5: Percentage improvement in cycle time of NEXTS vs PSR and Dispatching across five levels of variability and disturbance

		In increasing level of variability and disturbance →				
		Level 1	Level 2	Level 3	Level 4	Level 5
Variability Introduced to Expected Arrivals:		UNIF(-10, 10)	UNIF(-60, 60)	UNIF(-110, 110)	UNIF(-160, 160)	UNIF(-210, 210)
MTTF Disturbance:		EXP(4320)	EXP(3600)	EXP(2880)	EXP(2160)	EXP(1440)
% Improvement	NEXTS_G1_T2 vs PSR_R25_G1_T2	1.77	0.74	1.15	1.3	1.04
	NEXTS_G1_T4 vs PSR_R25_G1_T4	1.79	-0.18	1.42	2.82	1.92
	NEXTS_G1_T4 vs PSR_R25_G25_T4	2.08	0.38	1.58	3.36	2.41
	NEXTS_G1_T4 vs PSR_R60_G1_T4	5.93	1.81	2.6	4.13	2.82
	NEXTS_G1_T4 vs PSR_R60_G60_T4	6.21	4.82	4.90	4.51	4.42
	NEXTS_G1_T4 vs PSR_R405_G60_T4	18.99	18.98	18.85	17.06	10.33
	NEXTS_G1_T4 vs PSR_R750_G60_T4	37.55	35.18	34.92	32.44	16.65
	NEXTS_G1_T4 vs PSR_R1095_G60_T4	48.10	50.16	47.38	42.33	34.63
	NEXTS_G1_T4 vs PSR_R1440_G60_T4	60.09	56.10	51.70	51.04	40.49
	NEXTS_G1_T4 vs Dispatching_G1_T4	60.77	59.42	57.17	55.39	46.25

Figure 3.10: Percentage improvement in cycle time of NEXTS vs PSR and Dispatching across five levels of variability and disturbance

Two-tailed t -tests (tests of hypotheses of mean cycle time) are carried out between the NEXTS time policy (denoted by subscript A), and the PSR and dispatching time policies (denoted by subscript B). The null hypothesis is $H_0 : \mu_A - \mu_B = 0$ and

Table 3.6 summarizes whether to accept or to reject H_0 at 0.05 level of significance.

The detailed computation is presented in Appendix C.

Table 3.6: Summarized results on the tests of hypotheses between NEXTS time policy, and the PSR and dispatching time policies

		In increasing level of variability and disturbance →				
		Level 1	Level 2	Level 3	Level 4	Level 5
Variability Introduced to Expected Arrivals:		UNIF(-10, 10)	UNIF(-60, 60)	UNIF(-110, 110)	UNIF(-160, 160)	UNIF(-210, 210)
MTTF Disturbance:		EXP(4320)	EXP(3600)	EXP(2880)	EXP(2160)	EXP(1440)
Accept H_0 , $u_A = u_B$?	NEXTS_G1_T2 vs PSR_R25_G1_T2	Yes	Yes	Yes	Yes	Yes
	NEXTS_G1_T4 vs PSR_R25_G1_T4	Yes	Yes	Yes	Yes	Yes
	NEXTS_G1_T4 vs PSR_R25_G25_T4	Yes	Yes	Yes	Yes	Yes
	NEXTS_G1_T4 vs PSR_R60_G1_T4	Yes	Yes	Yes	Yes	Yes
	NEXTS_G1_T4 vs PSR_R60_G60_T4	Yes	Yes	Yes	Yes	Yes
	NEXTS_G1_T4 vs PSR_R405_G60_T4	Yes	Yes	Yes	Yes	Yes
	NEXTS_G1_T4 vs PSR_R750_G60_T4	Yes	Yes	Yes	Yes	Yes
	NEXTS_G1_T4 vs PSR_R1095_G60_T4	No	No	No	No	No
	NEXTS_G1_T4 vs PSR_R1440_G60_T4	No	No	No	No	No
	NEXTS_G1_T4 vs Dispatching_G1_T4	No	No	No	No	No

With reference to Table 3.6, H_0 is rejected for all levels of (i) NEXTS_G1_T4 versus PSR_R1095_G60_T4, (ii) NEXTS_G1_T4 versus PSR_R1440_G60_T4, and (iii) NEXTS_G1_T4 versus Dispatching_G1_T4. This means that there is statistical evidence to show that their mean cycle times are significantly different at 95% confidence level (i.e. at 0.05 level of significance). These results show that it is indeed desirable for decision-making to be committed before machine is available so that the bottleneck machine will not have to wait idling for the next lot to arrive. PSR, when rescheduled at a higher frequency, will give a slightly more comparable result to NEXTS. This explains why there are no statistical evidences to show that the mean cycle time for the following methodologies are significantly different: (i) NEXTS_G1_T2 versus PSR_R25_G1_T2, (ii) NEXTS_G1_T4 versus PSR_R25_G1_T4, (iii) NEXTS_G1_T4 versus PSR_R25_G25_T4, (iv) NEXTS_G1_T4 versus PSR_R60_G1_T4, (v) NEXT_G1_T4 versus

PSR_R60_G60_T4, (vi) NEXTS_G1_T4 versus PSR_R405_G60_T4, and (vii) NEXTS_G1_T4 versus PSR_R750_G60_T4.

The simulation experiments allowed for an evaluation of the benefit of the proposed time policy under a variety of conditions. Considering the (i) presence of transportation time, (ii) the dynamism of the environment over which the policies would be applied, and (iii) further work are required to test the performances of the improved PSR policy in which the schedule generation time is reduced, the experimental results show that it is worthy to exploit the proposed NEXTS time policy.

3.7 SUMMARY

The existing PSR and dispatching time policies are presented. The strength and weakness of the existing time policies and the gaps of existing literature to address the needs of actual implementation have also been addressed. A basic conceptual time policy for use in the NEXTS methodology is proposed and the conjecture that the proposed time policy will offer a better performance than existing time policies is shown using an illustrative example and further proven using a discrete-event simulation study. The proposed basic NEXTS time policy is shown to perform better than PSR and dispatching time policies in terms of mean cycle time in a one machine configuration. The simulation results also show considerable promise for the NEXTS time policy to be used in a dynamic and stochastic environment as NEXTS is robust to arrival and processing time variations, and machine breakdowns. This will be further evaluated in a wafer fab case study in Chapter 7.

This simulation results also suggest that it is worthwhile to study the detailed application of NEXTS time policy to address the timing aspects of scheduling that is generally lacking in the dispatching literature. Considering the PSR and dispatching time policies to have been outperformed by NEXTS time policy for the scenario that most matches the case considered in collaboration with Chartered Semiconductor Manufacturing Ltd., they will not be considered in the subsequent chapters. Instead, detailed variants of the NEXT time policy are proposed to consider a more practical problem in which loadports (buffer with limited capacity) at the machine to be scheduled are taken into account. The assumption of fixed transportation duration is also relaxed to accommodate for variable transportation duration to represent the transportation of lot from various physical locations to the machine.

CHAPTER 4

VARIANTS OF NEXTS TIME POLICY AND A CASE STUDY

In the last chapter, the conceptual time policy for making and executing a decision (i.e. selection of the next lot) at the latest possible time (before machine is available) is shown to perform favorably over existing PSR and dispatching time policies. The goal of this chapter is to propose detailed variants of the NEXTS time policies (built upon the conceptual time policy) that have the capability to be applied in semiconductor wafer fabrication and, in the process of doing so, identify the appropriate time pattern for the selection of the next lot.

The time policy research is motivated by an industrial application and will be discussed in relation to the real-life industrial application via a case study. The goal is not only to automate how to select the next lot but also when to select the next lot and invoke the required lot movement for a machine that processes lots arriving from different physical locations within the wafer fabrication shop floor.

The practical problem used in the case study is first described in Section 4.1. The proposed NEXTS time policies that have the capability to look upstream to consider information such as the arrival time of lots (jobs) to upstream buffers, and different transport duration from each upstream buffer to the machine for the dynamic scheduling of a discrete machine (or single lot processing tool) are then presented in Section 4.2. These proposed NEXTS time policies are subsequently

compared in Section 4.3 based on the average cycle time ($avgCT$) criterion. Since the decision aspect is an inseparable part of a scheduling methodology, it is realized by applying the Shortest Processing Time (SPT) rule to be in-line with the objective of minimizing average cycle time. Section 4.4 briefly discussed the possibility and the system architecture required for implementing the proposed NEXTS time policy in a wafer fab.

4.1 PROBLEM DESCRIPTION

The problem to be addressed is shown in Figure 4.1. The discrete machine (i.e. X) to be scheduled gets intermediate processed lots from any one of the m upstream buffers (i.e. $UB1, UB2, \dots, UBm$). These m upstream buffers hold intermediate lots that have each completed its preceding processing step on other machines. Machine X has a local buffer labeled as XB to hold lots arriving from upstream buffers. Transport time (or duration) between buffer XB and each of these m upstream buffers varies.

Figure 4.1: Partial shop floor

The problem described above could be observed in the wafer fabrication plant. One example is the dynamic scheduling of the costly lithography (or masking) machines. Machine X is considered as a particular lithography machine in this study. Buffer XB refers to loadports on the lithography machine. The loadports refers to the local buffers dedicated to the machine. Hence, the capacity of buffer XB can be set equal to the number of loadports.

Prior to the masking step, a lot may have undergone Oxidation and Diffusion, Etch and Clean Tech, Implantation, Film Deposition and Metallization, or Metrology process step. For more information of these wafer fabrication processes, refer to Section B.2 in Appendix B. The route taken could be (i) from previous upstream machine to current destination machine, or (ii) from previous upstream machine through stocker(s) (intermediate storage) and then to current destination machine (Kondo and Harada, 2005). The former is known as a direct machine-to-machine lot movement while the latter refers to a through-stocker lot movement.

As such, the m upstream buffers could refer to the loadports of the various machines used in the immediate previous step of lithography, or the stockers. These m upstream buffers (i.e. loadports and stockers) and the lithography machine could be positioned at various physical locations of the fab. This explains the consideration of variable transport times, $TT(i,j)$, to move lot j from upstream buffer i to machine X for processing.

In order to describe concisely the problem under study, the three-field notation by Graham et al. (1979) is used. For single machine scheduling problem $1|r_j|\Sigma C_j$ where the release date r_j is arbitrary and the total completion time ΣC_j is to be minimized,

even when all the parameters that define the problem instance are known in advance, the problem is strongly NP-hard. It is shown to be NP-complete by Lenstra et al. (1977). Hence, our research has mainly focused on designing heuristics for the problem.

4.2 DYNAMIC NEXTS TIME POLICY HEURISTICS

Based on the preliminary study of the basic conceptual time policy (see Section 3.3), it is identified that the proposed time policies should have the capability to look upstream to consider information such as the arrival time of lots to upstream buffers, and different transport duration from each upstream buffer to the machine for the dynamic scheduling of a discrete machine (or single lot processing tool). In most existing studies (e.g. Mao and Kincaid, 1994; Holthaus and Ziegler, 1997; Chen et al., 2004), the look-ahead feature is applied in the decision policies ('how to select the next lot'), whereas in our study, the 'look-ahead' is implemented in the time policy ('when to select the next lot'). The term 'look upstream' is coined to differentiate our usage of looking ahead in time based on upstream information to determine the appropriate time for next lot selection. Before the proposed time policies are presented, some of the relevant studies are presented.

Typical scheduling/rescheduling and dispatching studies generally do not consider the timing aspects associated with transportation issues. Typical automated material handling systems (AMHS) studies, on the other hand, mostly uses the FIFO rule and do not incorporate other dispatching rules for the selection of the next lot (e.g. Kondo and Harada, 2005; Lin et al., 2004). The AMHS studies are mostly focused

on scheduling from the transporter's perspective. However, some relevant timing aspects could be adapted for use from the machine's perspective.

In the AMHS studies by Kondo and Harada (2005), there are generally two types of decision points: (i) transportation of the next lot is triggered when the lot finishes processing on another tool used in the previous process step, and (ii) transportation of the next lot occurs whenever the loadport of a tool to be scheduled becomes empty. Recent routing (Jang et al., 2001), loadport buffering (Wright et al., 2005), and AMHS (Lin et al., 2004; Wakabayashi et al., 2004; Kondo and Harada, 2005) studies suggest or consider the transportation of the next lot to the machine whenever the loadport of the machine to be scheduled becomes available.

In a study by Chang (2004), the mechanism involves the use of precise 'forecast delivery time' and 'remaining process time' to decide the right time to trigger just-in-time (JIT) delivery for 300mm fab to fulfill continuous processing especially for high-throughput tools. The mechanism, as suggested, is to trigger the delivery when the 'forecast delivery time' is equal to the 'remaining process time'. However, no experimental or implementation results are presented.

The loadport initiated time policy is adapted from the transporter's perspective (based on the AMHS related studies) and formulated for use in scheduling from the machine's perspective. This loadport initiated time policy will be presented and compared against two other proposed time policies after the notations and abbreviations (see Table 4.1) and assumptions used in this case study are listed.

Table 4.1: Description of notations and abbreviations used in the proposed NEXTS time policy heuristics

Notation	Description
X	Machine to be scheduled.
XB	Machine X 's Buffer with capacity of one - This local buffer is used to hold a lot waiting for processing by machine X but does not include the lot that is being processed by machine X .
UB	Upstream machine's buffer
i	A particular upstream buffer with assumed unlimited capacity, $i \in \{1, \dots, m\}$.
n	Total number of lots to be processed by machine X .
j	A particular lot, $j \in \{1, \dots, n\}$.
$AT(i,j)$	(Step) Arrival Time of lot j to upstream buffer i .
$AT(XB,j)$	(Step) Arrival Time of lot j to XB .
$PT(j)$	(Step) Processing Time of lot j on machine X .
$TT(i,j)$	(Step) Transport time to move lot j from upstream buffer i to machine X for processing.
t	Current time.
η	Number of lots that has been processed by machine X .
$NLST$	Next Lot Selection Time (refers to the time when the next lot is to be selected).
$NLSP$	Next Lot Selection Period (refers to a <i>time period</i> for which the next lot selection occurs before machine X is available).
XAT	Machine X 's Available Time (refers to the time when machine X is available for processing the next lot).
$LVL(XB)$	XB 's Level, $LVL(XB) = \{0,1\}$. '0' indicates that XB is empty and '1' implies that XB is full (holding one lot).
$LVL(i)$	Upstream buffer i 's buffer level.
$LIST$	Contains an appropriately updated list of lots that has arrived at the upstream buffers but has not been processed by the machine at time t .
$NLIST$	Number of lots in the $LIST$.
$SEL(j)$	Selected Lot (implies that lot j is selected for processed next on machine X).
$PST(j)$	Processing (Step) Start Time of lot j on machine X .
$PCT(j)$	Processing (Step) Completion Time of lot j on machine X .
$CT(j)$	(Step) Cycle Time of lot j is defined as $CT(j) = PCT(j) - AT(i,j)$.
$avgCT$	(Step) Average Cycle Time.

The assumptions used in this study are summarized as follows:

- A set of n intermediate lots are to be scheduled.
- Each one of these n lots is expected to pass through any one of the respective m upstream buffers and is associated with a step arrival time, $AT(i,j)$ that denotes the arrival time of lot $j, j \in \{1, \dots, n\}$ to upstream buffer $i, i \in \{1, \dots, m\}$; and a step processing time, $PT(j)$ that denotes the processing time of lot j on the machine to be scheduled.
- Each job requires one processing operation (step) on the machine to be scheduled, and all jobs are independent.
- The machine to be scheduled can process only one lot at any one time and has an attached local buffer with capacity of one lot. This means that the local buffer can hold a lot waiting to be processed, not counting the lot that the machine is processing. In practice, this can be viewed as equivalent to a two-loadport configuration.
- Processing is non-preemptive.
- Each of the upstream buffers is assumed to have unlimited capacity.
- Transport times (or durations) between upstream buffers and the machine to be scheduled vary and $TT(i,j)$ denotes the step transport time (or duration) to move lot j from the upstream buffer i to the machine for processing.
- Negligible transport time is considered between the local buffer and the machine to which it is attached.
- All data are assumed to be known *a priori*. In practice, the required parameters for this problem could be obtained from the shop floor information system in real time with reasonable accuracy.

All the proposed dynamic time policies look upstream for next lot selection and each variant can be summarized as follows:

Variant 1: Fixed Time Period Policy

In this policy, a new lot is selected, from among all the lots available in the upstream buffers, for processing on machine X at a fixed time period before machine X is available. At each time of the next lot selection, no future arrival of lots to the m upstream buffers is considered. The fixed time period refers to the maximum among all the transport times between the m upstream buffers (whether empty or non-empty) and machine X . This could be viewed as a case where the lot selection procedure waits until the latest possible time before committing a lot to a specific machine for processing.

Variant 2: Loadport-Initiated Time Policy

This is similar to the fixed time period policy (see Variant 1) except that a new lot is selected as soon as XB (the loadport) becomes empty.

Variant 3: Adjusted Time Period Policy

This is an extension of the fixed time period policy (see Variant 1). The next lot selection is postponed (where possible) closer to the time (before) machine X is available. This is achieved by using an adjusted time period. The time period is adjusted based on the maximum of the transport times taken to deliver lots in the non-empty upstream buffers to machine X . This is to eliminate the cases where a constantly empty upstream buffer located far away from machine X dictates the time period, thereby causing the next lot selection to occur much earlier

than necessary. This could be viewed as a case where the lot selection procedure waits until the latest possible time before committing a lot to a specific machine for processing.

Figure 4.2 shows the Gantt chart from the collective perspective of machine X , lot j (referring to the current lot in process), and lot $j+1$ (referring to the next selected lot) for each of the three time policies presented above. The dependencies (relationship) between the listed events are indicated with arrows. From the Gantt chart, the processing of lot j starts at time t_A . Given that the processing time of lot j , $PT(j)$, can be predicted or is known a priori, the processing step completion time for lot j can be obtained using $t_B = t_A + PT(j)$. Time t_B also corresponds to the time when machine X is next available.

By looking upstream, the next lot selection is performed before time t_B . For the fixed time period policy, the next lot selection is invoked at time $t_C = t_B - NLSP$ where $NLSP$ is defined as the maximum of all transport times from each upstream buffer to machine X . In the case of the loadport-initiated time policy, when lot $j-1$ (the lot processed before lot j) leaves and when lot j begins processing on machine X , the loadport becomes available (empty). For the loadport-initiated time policy, the next lot selection takes place at time t_A .

In a move to practically push the decision point further forward in time, the adjusted time period policy is proposed. As the time policy is based on an adjusted time period, the machine X Gantt shows that the next lot selection can be triggered to occur at any time between t_C and t_D (both times inclusive). When next lot selection is invoked, the decision policy that is implemented will guide the

selection process. Suppose that lot $j+1$ is selected. After the selection, lot $j+1$ is transported to the available loadport XB (machine X 's buffer) for processing.

Figure 4.2: Gantt chart for the three proposed time policies

The detailed step-based procedures of the three proposed time policies are provided below. Discrete event simulation carried out to compare the performance of the proposed dynamic scheduling time policies is subsequently discussed in Section 4.3.

4.2.1 ALGORITHM OF THE FIXED TIME PERIOD POLICY

Step 1: Initialization

Set all variables to zero or empty where applicable.

$t = 0$; $NLST = 0$; $NLSP = 0$; $XAT = 0$; $LVL(XB) = 0$; $LVL(i) = 0$; $\eta = 0$;
 $LIST = \{\text{empty}\}$; $NLIST = 0$; $SEL(\text{null})$; $PST() = 0$; $PCT() = 0$; $CT() = 0$;
 and $avgCT = 0$.

Step 2: Next Lot Selection

Step 2.1 (Updating)

Update time (i.e. $t = NLST$) and retrieve the $LIST$ of lots that has already reached its respective upstream buffers and waiting for processing on machine X . The $LIST$ contains lots with $AT(i,j) \leq NLST$. The number of waiting lots is given by $NLIST$.

Step 2.2 (Re-adjust next lot selection time whenever necessary – part of time policy)

If $NLIST = \text{zero}$ (i.e. no lot is waiting in all upstream buffers)

Set $NLST$ to the next earliest arrival time of a lot;

Repeat Step 2.1;

Else

Continue to Step 2.3;

End If

Step 2.3 (Execute decision policy to select the next lot)

From the *LIST* of at least one lot ($NLIST \geq 1$), select the next lot based on the applied decision policy. In this study, SPT rule is applied. Hence, the lot with the shortest processing time is selected for processing next on machine X . Ties (if any) are resolved based on FIFO (i.e. arrival time to upstream buffer i). $SEL(j)$ is updated to keep track that lot j is selected.

Step 3: Updating & Determining Next Lot Selection Time

Step 3.1 (Update till the time the selected lot j reaches machine X 's buffer, XB)

Selected lot j is transported from upstream buffer i to machine X 's buffer, XB with a duration given by $TT(i,j)$. Since lot j is selected, it is removed from the *LIST* and place into machine X 's buffer, XB : $LVL(XB) = 1$.

Lot j 's arrival time to machine X 's buffer: $AT(XB,j) = t + TT(i,j)$, where $t = NLST$.

Update current time: $t = AT(XB,j)$.

Step 3.2 (Update processing step start time of selected lot j , $PST(j)$, and next machine X 's available time, XAT)

If machine is available before the selected lot arrives (i.e. when $AT(XB,j) \geq XAT$)

Machine X processes the selected lot j the moment it arrives at XB :

$PST(j) = t (= AT(XB,j))$ as has been updated in Step 3.1).

Else If selected lot arrives when machine X is still processing a previous selected lot, (i.e. when $AT(XB, j) < XAT$)

Machine X processes the selected lot j only after it completes processing the previous selected lot: $PST(j) = XAT$

End If

Next machine X 's available time will be the time it finishes processing the selected lot j : $XAT = PST(j) + PT(j)$

Step 3.3 (Update next lot selection time, $NLST$ – based on a fixed time period before machine's availability)

The $NLSP$ refers to the *fixed time period* that is set equal to the maximum among all the transport times between upstream machine buffers (regardless of buffer level) and the machine X . Since $NLSP$ is a fixed period, the $NLSP$ value only to be set once: $NLSP = \max\{TT(i, j), i \in \{1, \dots, m\}\}$

Computing the next lot selection time:

If $NLSP$ is shorter than or equal to the processing time of the selected job j

(i.e. $NLSP \leq PT(j)$)

$$NLST = XAT - NLSP$$

Else $NLST$ will be less than the current time, hence reset to current time

$$NLST = t, \text{ where } t = AT(XB, j)$$

End If

Step 3.4 (Update and compute performances)

Set machine X 's buffer level to zero: $LVL(XB) = 0$;

Increment the number of lots processed by one: $\eta = \eta + 1$;

Set the process step completion time of lot j to the machine's available

time: $PCT(j) = XAT$

Compute the cycle time of lot j : $CT(j) = PCT(j) - AT(i,j)$

Step 4: Stopping criterion

If $\eta = n$ (number of processed lots equal the total number of lots planned for the study),

Proceed to Step 5.

Else

Repeat Step 2.

End If

Step 5: Compute performances

Retrieve the cycle time performances for all completed lots and compute average cycle time, $avgCT$.

$$avgCT = \frac{1}{n} \sum_{j=1}^n CT(j)$$

4.2.2 ALGORITHM OF THE LOADPORT-INITIATED TIME POLICY

The loadport-initiated time policy is similar to the fixed time period policy (refer to Section 4.2.1) except for Step 3.3 where a new lot is selected as soon as machine X 's buffer (XB) is empty. Hence, only the modified step is presented below:

Step 3.3 (Update next lot selection time, $NLST$ – based on the availability of machine X 's buffer, XB)

Next lot selection occurs whenever XB becomes empty and this corresponds to the time when machine X finishes processing a lot

(assuming negligible transport time taken to transport the finished lot away). In Step 3.2 of the fixed time period policy, machine X 's available time, XAT has already been updated to include the processing time of the selected lot j . Hence, it is necessary to subtract $PT(j)$ from XAT to get the previous XAT : $NLST = XAT - PT(j)$;

4.2.3 ALGORITHM OF THE ADJUSTED TIME PERIOD POLICY

As the name of the adjusted time period policy suggests, the next lot selection is carried out based on an *adjusted time period* before machine X is available. The time period is adjusted based on the maximum of the transport times taken to deliver lots in the *non-empty* upstream buffers to machine X . The adjusted time period policy is an extension to the fixed time period policy (refer to Section 4.2.1) and only Step 2.1 and Step 3.3 of the fixed time period policy need to be modified. The modified step is presented below:

Step 2.1 (Updating)

Update time (i.e. $t = NLST$) and retrieve the *LIST* of lots that has already reached its respective upstream buffers. Since the next lot selection is no longer computed based on the maximum of all possible transport times, only lots that will reach machine X in time for processing are considered for the next lot selection. Hence, lots that meet the following conditions are included in the *LIST*: $(AT(i,j) \leq NLST) \text{ OR } (NLST + TT(i,j) \leq XAT)$

The number of lots in the *LIST* is given by $NLIST$.

Step 3.3 (Update next lot selection time, $NLST$ – based on an adjusted time period before machine's availability)

The *adjusted time period* is set equal to the maximum of the transport times taken to deliver lots in the *non-empty* upstream buffers to machine X :

$$NLSP = \max \{ TT(i,j); i \in 1, \dots, m \mid LVL(i) \neq 0 \}$$

Computing the next lot selection time:

If $NLSP$ is shorter than or equal to the processing time of the selected job j (i.e. $NLSP \leq PT(j)$)

$$NLST = XAT - NLSP$$

Else $NLST$ will be less than the current time, hence reset to current time

$$NLST = t, \text{ where } t = AT(XB_j)$$

End If

4.3 EXPERIMENTAL EVALUATION OF THE DYNAMIC NEXTS TIME POLICY HEURISTICS

The three proposed time policy algorithms are coded and simulated using AutoSchedTM Accelerated Processing (AutoSchedAP) v7.2 by Brooks Automation Inc. Customizations are necessary and the simulation models are debugged and verified in a number of iterations using the ‘trace’ technique (Law, 2007). The computational experiments and the analysis carried out for evaluating the proposed heuristics are discussed in the subsequent sub-sections.

4.3.1 COMPUTATIONAL EXPERIMENTS

Three problem parameters required in the time policies are identified and included in the experimental design of the three time policies to schedule $n = 1000$ lots. No

distribution is fitted for processing times. However, the processing time (PT) parameter is uniformly distributed between $U[35,75]$ minutes. This is labeled as the medium (M) level. The narrow (N) and wide (W) levels are included to possibly cover other problem configuration possibilities. Uniform distribution is chosen because it is bounded in the range collected from an extrapolated set of historical data from Chartered Semiconductor Manufacturing Ltd.

Each lot is equally likely to arrive from any of the upstream buffer. Lots are assumed to follow a uniformly distributed inter-arrival time (AT) of $U[35,75]$ minutes, also labeled as the medium level. Apart from the narrow and wide levels, exponentially distributed AT with mean of 55 minutes is included as a possible fourth AT level labeled as exponential (E).

Since the m upstream buffers could refer to loadports or stockers as in direct machine-to-machine or through-stocker delivery respectively, the transport time (TT) is estimated from the observed historical data corresponding to the respective machine or stocker locations. In this study, three upstream buffers (i.e. $m = 3$) are considered. The Narrow with Same mean (NS) level, for example, implies that the transport time from each of the three upstream buffers to machine X follows a narrow uniformly distributed range of $U[9,11]$ minutes with the same mean value of 10 minutes. Conversely, the Wide with Different mean (WD) level implies that the transport time from each of the three upstream buffers to machine X follows a wide uniformly distributed range of $U[1,9]$, $U[6,14]$, and $U[11,19]$ minutes with different mean values of 5, 10, and 15 minutes respectively.

Table 4.2 summarizes the experimental design discussed above. For each problem configuration as described by $\{PT = N/M/W, AT = N/M/W/E, \text{ and } TT = NS/WS/ND/WD\}$, 20 problem instances (replications) are randomly generated to obtain acceptable confidence in the mean of average cycle time output, yielding a total of 960 ($= 3 \times 4 \times 4 \times 20$) instances. The same seeds for each problem instance are applied across the three time policies to allow comparison in a more statistically efficient manner. In this study, terminating simulation is assumed where $n = 1000$ lots are to be processed on machine X . All buffers are initially empty with machine X starting in idle mode. Without loss of generality, it is assumed that in any problem instance, there is one lot that arrives to any one of the upstream buffers at time zero.

Table 4.2: Summary of the experimental design

Factors	No. of Levels	Levels	Values
Processing Time (PT) of a lot on machine X	3	Narrow (N): Medium (M): Wide (W):	$U[50,60]$ $U[35,75]$ $U[20,90]$
Inter-Arrival Time (AT) of a lot to upstream buffer i	4	Narrow (N): Medium (M): Wide (W): Exponential (E):	$U[50,60]$ $U[35,75]$ $U[20,90]$ $Expo(55)$
Transport Time (TT) of a lot from upstream buffer i to machine X	4	Narrow, Same mean (NS): Wide, Same mean (WS): Narrow, Different mean (ND): Wide, Different mean (WD):	$U[9,11]; U[9,11]; U[9,11]$ $U[6,14]; U[6,14]; U[6,14]$ $U[4,6]; U[9,11]; U[14,16]$ $U[1,9]; U[6,14]; U[11,19]$
Number of problem configurations	$3 \times 4 \times 4 = 48$		
Problem instances per configuration	20		
Total problem instances per time policy	$48 \times 20 = 960$		

4.3.2 RESULTS AND ANALYSIS OF THE NEXTS TIME POLICIES

A Kruskal-Wallis One Way Analysis of Variance on Ranks (ANOVA on ranks) study shows that the differences in the mean *avgCT* values of each transport time level (i.e. *NS*, *WS*, *ND*, and *WD*) are not statistically significant at 95% confidence level for all the three time policies under different combinations of *PT* and *AT* levels. For brevity, ANOVA tables are not presented but the *avgCT* output data for 960 problem instances of each time policy, test statistics *H* and *P* values¹ obtained from ANOVA on rank tests are tabulated in Appendix D.

As such, the four levels of transport time are combined in our analysis. Each problem configuration is therefore made up of 80 problem instances (= 4 levels of transport time term $\times$ 20 instances). Kondo and Harada (2005) reported that the Automated Material Handling Systems (AMHS) delivery time does not add any additional waiting time to lot cycle time if

$$TU + TL \leq (\text{number of loadports} - 1) \times \text{tool cycle time}, \quad (4.1)$$

where *TU* is the time duration from unloading request to unloading completion, and *TL* is the time duration from determination of lot that should be delivered to the tool to completion of loading. Our problem configurations satisfy Equation (4.1). It should be noted that the definition of the buffer capacity used in this study (the lot in process does not occupy a loadport) varies slightly from the number of loadports used in Equation (4.1) and should be two instead of one when used in Equation (4.1). Our simulation results support the results reported in Kondo and Harada

¹ *P* value is the probability of being wrong in concluding that there is a true difference in the groups based on the test statistics *H* used. The ANOVA on Ranks test statistic *H* is computed by ranking all observations from smallest to largest without regard for treatment group. The average value of the ranks for each treatment group are computed and compared (see Appendix D).

(2005) since the transport time is not a significant term where mean $avgCT$ is concerned. In subsequent analysis, the transport time terms are grouped together.

The relative percentage deviation (RPD) is computed with respect to the best results (in terms of average cycle time, $avgCT$) in each of the 960 problem instances since there are no optimal or estimated optimal results to compare against.

$$RPD = \frac{\text{Results(Policy } \Upsilon) - \text{Results(Best Policy)}}{\text{Results(Best Policy)}} \times 100, \quad \Upsilon \in \{1, 2, 3\} \quad (4.2)$$

Table 4.3 presents the mean and 95% confidence interval for the $avgCT$ output. As mentioned above, each reading in Table 4.3 is based on 80 problem instances.

Table 4.3: Mean and 95% confidence intervals for $avgCT$

Problem Configuration	Mean of $avgCT$ with 95% Confidence Interval [in hours]		
	Fixed Time Period Policy	Loadport-Initiated Time Policy	Adjusted Time Period Policy
1. $\{PT=N, AT=N, TT\}$	2.285 $\pm$ 0.139	2.316 $\pm$ 0.142	2.290 $\pm$ 0.139
2. $\{PT=M, AT=N, TT\}$	3.681 $\pm$ 0.300	3.821 $\pm$ 0.305	3.703 $\pm$ 0.299
3. $\{PT=W, AT=N, TT\}$	4.774 $\pm$ 0.452	4.995 $\pm$ 0.457	4.800 $\pm$ 0.450
4. $\{PT=N, AT=M, TT\}$	4.574 $\pm$ 0.440	4.619 $\pm$ 0.441	4.580 $\pm$ 0.439
5. $\{PT=M, AT=M, TT\}$	4.567 $\pm$ 0.452	4.713 $\pm$ 0.456	4.587 $\pm$ 0.451
6. $\{PT=W, AT=M, TT\}$	5.402 $\pm$ 0.532	5.632 $\pm$ 0.536	5.430 $\pm$ 0.529
7. $\{PT=N, AT=W, TT\}$	6.080 $\pm$ 0.589	6.127 $\pm$ 0.590	6.085 $\pm$ 0.589
8. $\{PT=M, AT=W, TT\}$	6.534 $\pm$ 0.692	6.690 $\pm$ 0.694	6.549 $\pm$ 0.690
9. $\{PT=W, AT=W, TT\}$	6.043 $\pm$ 0.651	6.276 $\pm$ 0.654	6.069 $\pm$ 0.649
10. $\{PT=N, AT=E, TT\}$	15.228 $\pm$ 1.871	15.275 $\pm$ 1.872	15.231 $\pm$ 1.871
11. $\{PT=M, AT=E, TT\}$	12.660 $\pm$ 1.401	12.832 $\pm$ 1.403	12.669 $\pm$ 1.400
12. $\{PT=W, AT=E, TT\}$	10.576 $\pm$ 1.010	10.843 $\pm$ 1.012	10.591 $\pm$ 1.008

The score of the average RPD ($ARPD$) indicates the average performance of the proposed heuristic algorithm with respect to the best heuristic. The maximum RPD ($MRPD$) indicates the worst-case performance of the heuristic algorithms. Both $ARPD$ and $MRPD$ were computed for each of the 80 problem instances of the

problem configuration $\{PT, AT\}$. $ARPD$ and $MRPD$ of $avgCT$ are plotted in Figure 4.3 and Figure 4.4 respectively. The problem configuration labels of 1 to 12 in Figure 4.3 and Figure 4.4 correspond to the ones used in Table 4.3.

Figure 4.3: Average relative percentage deviation ($ARPD$) of the average cycle time ($avgCT$) chart

Figure 4.4: Maximum relative percentage deviation ($MRPD$) of the average cycle time ($avgCT$) chart

ARPD value of zero indicates that the heuristic algorithm performs the best in all the considered problem instances. The smaller the *ARPD* value, the better the performance of the heuristic algorithm. Similarly, *MRPD* value of zero indicates that the maximum of *avgCT* is the lowest among all the considered problem instances. Hence, the smaller the *MRPD* value, the better the performance of the heuristic algorithm.

The fixed time period policy performs better than the loadport-initiated time policy in terms of both *ARPD* and *MRPD* of *avgCT* in all tested problem configurations. In fact, the *ARPD* and *MRPD* of *avgCT* for the fixed time period policy is so small (i.e. $ARPD \leq 0.001$ and $MRPD \leq 0.055$) that it could not be seen in Figure 4.3 and Figure 4.4 respectively. With reference to Figure 4.2, the committing and moving of a selected lot to a specific machine at a processing operation occurs later in the fixed time period policy as compared to that of loadport-initiated time policy. As such, any lot arriving between time t_A and t_C with a shorter processing time than the lots that have already been waiting in the upstream buffers could be considered for selection in the fixed time period policy but not in the loadport-initiated time policy since the latter has already committed a lot to machine X at time t_A . Being able to process a job with a shorter processing time possibly explains for a better *avgCT* results observed when the fixed time period policy is applied.

From the above discussion and our initial conjecture, the adjusted time period policy is intuitively expected to perform better than the fixed time period policy if committing an event at the latest possible time practical of the situation will produce better results. However, the conjecture is proven wrong since Figure 4.3 and Figure 4.4 show that the fixed time period policy has a slightly better

performance than the adjusted time period policy in terms of both *ARPD* and *MRPD* of *avgCT*. In the move to practically push the decision point further forward in time in order to consider the lots that arrive at an even later time (i.e. between time t_C and t_D), thereby increasing the chance of selecting another lot with an even shorter processing time lot than those already in queue, an additional constraint need to be imposed for the adjusted time period policy. That is, only lot with transport duration shorter than or equal to the remaining processing time left on machine X would be selected so as not cause the machine to become idle while waiting for the next lot to arrive.

The tradeoff is, if no lot with the shortest processing time did arrive between times t_C and t_D , existing lots (those that have arrived before time t_C) with short processing times but with transport duration longer than the remaining processing time left on machine X could no longer be chosen for processing next. As the time duration between time t_A to a time between t_C and t_D is longer than that between t_C to a time between t_C and t_D based on the problem configuration used in this case study, the tradeoff has more probabilistic influence over its advantage, resulting in slightly more negative effects on the mean *avgCT* performance when the adjusted time period policy is applied.

Table 4.4 shows the *RPD* value averaged over all the 960 problem instances for each variant. Overall, the fixed time period policy is also superior to the loadport-initiated time policy and the adjusted time period policy, and this observation is similar to the ones observed with respect to the 12 problem configurations presented above.

Table 4.4: Average relative percentage deviation (*ARPD*) of average cycle time (*avgCT*) results averaged over 960 problem instances

	Variant 1	Variant 2	Variant 3
Average <i>RPD</i> of <i>avgCT</i> for 960 Instances	0.000	2.765	0.372

From the simulation results, it is observed that the fixed time period policy outperforms the loadport initiated time policy by approximately 0.3% to 4.6% in terms of average cycle time (computed from the mean of average cycle time presented in Table 4.3).

4.4 IMPLEMENTATION

The actual implementation of the time policy can be achieved by means of an advanced scheduling system that uses Activity Manager (AM), a commercial application of the Brooks Automation's Advanced Productivity FamilyTM (APF) suite, integrated with dispatching rule (decision) policies in Real Time Dispatcher (RTD) and Manufacturing Execution System (MES) as a platform to execute real time transactions for lot dispatching. It requires the replacement of the existing polling mechanism used by the MES with real time event tracking and execution. The MES-RTD architecture that could support the real-time event-based requirements of the time policies is integrated and implemented in Chartered Semiconductor Manufacturing Ltd (Mohd Azizi et al., 2005).

4.5 SUMMARY

Analyses of the three time policies lead to the following summarized observations (based on the problem configurations used in this case study):

- Commonly used loadport-initiated time policy is outperformed by both the fixed and adjusted time period policies.
- The loadport-initiated time policy that is addressed in recent AMHS related studies) is outperformed by the fixed time period policy by approximately 0.3% to 4.6% in terms of average cycle time.
- The general intuition of committing a decision (i.e. selection of next lot) only at the latest possible time so that the decision has a much lower possibility of being invalidated by the stochastic events on the shop floor is proven wrong since the adjusted time period policy does not perform better than the fixed time period policy in terms of average cycle time.

The conceptual NEXTS time policy has been shown to outperform commonly used PSR and dispatching time policies in Chapter 3 and its practicality, as demonstrated in this chapter, extends to discrete machines in 300mm fully automated wafer fab that ‘pulls’ lots from various loadports and stockers for processing. The use of ‘look upstream’ in the proposed NEXTS time policies to consider information such as dynamic job arrivals, and location-dependent transport times to determine the timing for next lot selection is viable.

With reference to the results of this study, the fixed time period policy is the best time policy for invoking the next lot selection. Next lot selection, when carried out any earlier (i.e. the loadport-initiated time policy) or later (i.e. the adjusted time period policy), results in poorer average cycle time performances. This study does not claim that the optimum performances for all systems are achieved using the fixed time period policy as the constraints and data sets may be vastly different. It does, nevertheless, confirm that the appropriate timing has to be identified based on

the concept of carrying out the next lot selection and delivery before and with practicable closeness to the time when the machine is available. The simulation results also indicate that there is a need to address the ‘when’ aspects in conjunction to the ‘how’ aspects of selecting the next lot for achieving better fab performance.

As the NEXTS methodology is made up of a time policy and a decision policy, having addressed and identified a time policy (i.e. the fixed time period policy) for use in the NEXTS methodology, a natural extension is to incorporate a better decision policy for the selection of lots. This is because if the decision policy does not give a good performance, the time policy however well executed is futile. Hence, the NEXTS decision policy related research work is addressed over the next two chapters before Chapter 7 evaluates the combination of the proposed time and decision policies using a case study that is built on the one presented in this chapter.

CHAPTER 5

PRELIMINARY STUDY OF DECISION POLICIES FOR MULTIOBJECTIVE SCHEDULING

This research is concerned with the design of a NEXt Task Scheduling (NEXTS) methodology that integrates time and decision policies to handle multiple conflicting objectives and consider user preferences for the selection of the next task in a dynamic and stochastic environment that includes the presence of sequence-dependent setups and transportation times. The transportation times have been addressed in the proposed NEXTS time policy. Hence, it is important to consider multiple conflicting objectives, user preferences, as well as sequence-dependent setups in the design of the proposed decision policy.

This chapter put forth the evaluation of all the groundwork and preliminary concepts that go into the design of the NEXTS decision policy. The Pareto optimality and dominance concepts commonly applied in multiobjective studies are first presented in Section 5.1 to facilitate subsequent discussions that refer to the Pareto and dominance concepts. Then multiobjective optimization is briefly introduced (see Section 5.2) followed by an evaluation of the strengths and weaknesses of the existing decision policies. The evaluation helps to identify the direction taken to design the NEXTS decision policy. In Section 5.3, Markov modeling is used to evaluate the possible inclusion of look upstream and look downstream components into the proposed decision policy before the criteria (in

the form of dispatching rules) are selected and analyzed in Section 5.4. Finally, a correlation analysis of objectives is presented in Section 5.5.

5.1 PARETO OPTIMALITY AND DOMINANCE CONCEPTS

It is rarely the case that there is a single point that simultaneously optimizes all the objective functions of multiobjective optimization problems. Therefore, achieving trade-offs are required to deal with the multiple conflicting objectives. Coello Coello (2005) reported that this ‘trade-off optimality’ notion is originally proposed by Edgeworth (1881) and later generalized by Pareto (1896; selected translated portions are available in Pareto (1966)). The notion is commonly known as *Pareto optimality*.

A vector of decision variables $\mathbf{t}^* \in F$ is *Pareto optimal* if there does not exist another $\mathbf{t} \in F$ such that $f_i(\mathbf{t}) \leq f_i(\mathbf{t}^*)$ for all $i = 1, \dots, k$ and $f_j(\mathbf{t}) < f_j(\mathbf{t}^*)$ for at least one j (Coello Coello, 2005). In other words, a solution vector is said to be Pareto optimal if there exists no solution vectors that would decrease one or more objective function values without causing a simultaneous increase in at least one other objective function value.

Suppose that there are two objective functions, $f_1(t)$ and $f_2(t)$, to be minimized. These two objective functions can be formulated as a single objective problem given by $Z_i \triangleq (f^1(t_i), f^2(t_i))$ where each Z_i corresponds to a point that can be plotted on the Cartesian coordinates as shown in Figure 5.1.

Figure 5.1: Pareto optimality

Most multi-objective optimization methods use the dominance concept to compare and search for non-dominated solutions. With reference to Figure 5.1, solutions Z_3 and Z_2 are dominated by solution Z_1 . Hence, it could be said that Z_1 is a better solution than Z_2 and Z_3 in the face of multi-objective optimization. The Pareto optimal solutions (i.e. solutions found to lie on the Pareto front/surface) constitute the set of non-dominated solutions for the entire search space considered.

5.2 MULTIOBJECTIVE OPTIMIZATION AND SCHEDULING

Multiojective optimization has been widely researched, published and reviewed (Hwang et al., 1980; Sawaragi et al., 1985; Tabucanon, 1988; Bagchi, 1999; Deb, 2001). The solution approaches to multiobjective optimization can be classified based on the way the initial multiobjective problem is transformed into (Bagchi, 1999). The classifications include: (i) conversion of secondary objectives into constraints, (ii) development of a single combined objective function, and (iii) treatment of all objectives as constraints. Hwang et al. (1980), alternatively, grouped solution methods according to the stage at which information is needed from the decision maker. The stages are classified into four categories: (i) no articulation of preference data, (ii) *a priori* articulation of preference information,

(iii) progressive articulation of preference information, and (iv) *a posteriori* articulation of preference information from the decision maker. Some of these classical methods and algorithms are briefly described in Table 5.1.

Table 5.1: Brief description of some classical methods and algorithms

Classical Methods	Description
ε -Constraint Method	This method involves optimizing one objective function and using all other objectives as constraints.
Compromise Programming	This is a distance-based technique that aims to identify solutions that are closest to the ideal solution.
Global Criterion Method	This method aims at minimizing the sum of deviations of the values of the individual objective functions from their respective singular objective optimum values as a ratio to that of the singular optima.
Goal Programming	This method aims at minimizing deviation variables (from goals) with assigned priorities and weights.
Minimum Deviation Method	This method aims at minimizing the sum of individual objective's fractional deviation. The fractional deviation of an objective refers to a ratio between the deviation of a value of that objective from its individual solution and its maximum deviation.
Weighted Aggregation Method	This method aims at minimizing a positively weighted convex sum of the objectives.

Although not all Pareto-optimal solutions can be found by some of these classical algorithms in nonconvex multiobjective optimization problems (e.g. Weight Aggregation Method) and that some problem knowledge (e.g. suitable weights) are required, these algorithms do have their advantages. For example, these algorithms have been proven to converge to the Pareto-optimal set (e.g. the ε -constraint method guarantees that every Pareto-optimal solution corresponds to an optimal solution to the resulting single-objective optimization problem for any convex or nonconvex multiobjective optimization problems) and that they are simple and easy to implement.

Multiobjective optimization is typically applied in scheduling/rescheduling (classified under PRS) to generate an optimal schedule. However, it is possible that the variability in a fab could be so high that the extensive computational load required for frequent rescheduling to achieve an optimal schedule makes it unsuitable for near-real-time applications. Additionally, the simplification and assumptions built into most of the existing theoretical research problems may result in the approach being unable to account sufficiently for the complexities and uncertainties present in actual facilities. These, according to fab practitioners at AMD (Krishnaswamy and Nettles, 2005), steer the semiconductor industry towards dispatching (Rippenhagen and Krishnaswamy, 1998; Watt, 1998; Tumlinson and Muzumdar, 1999; Williams et al., 2000; Baweja and La, 2001).

Dispatching (classified under CRS), on the other hand, is able to achieve near-real-time responsiveness (i.e. is reactive to disruptions and requires a low computation load) but without the required optimality as the pure application of simple dispatch rules is often labeled as a myopic approach that may result in local or sub-optimal solutions. Therefore, it is predicted that an adaptation of the partial aspects of the PRS and CRS methodologies into the proposed Next Task Scheduling (NEXTS) methodology (classified under HRS) would be desirable. For example, the use of global schedules to guide dispatching (as in scheduling-dispatching) and/or the careful selection and aggregation of rules (as in dispatching) that takes into consideration global system states could be potential directions for the design of the NEXTS decision policy.

Previous closely related studies (Park et al., 2001; Dabbas et al., 2001; Dabbas and Fowler, 2003; Dabbas et al., 2003; Lin et al., 2005; Gupta and Sivakumar, 2005a,

2005b; Sivakumar and Gupta, 2006) addressed the multiobjective optimization problem by applying rules simultaneously in the form of a weighted aggregation of rules to optimize multiple performance measures of interest. As can be observed from the above studies, the main strength of the weighted aggregation method is its computational efficiency, and its suitability to generate non-dominated solutions. Its main weakness is the difficulty to determine the appropriate weights that can appropriately scale the objectives when there is insufficient information about the problem (Coello Coello, 1999). This approach also does not generate all the Pareto optimal solutions in the presence of non-convex search spaces regardless of the weights used. However, this is not critical, as the goal of the research is not to identify the entire Pareto surface but to select one solution from the surface at each lot selection instance.

After evaluating the pros and cons from these recent studies reviewed above, the weighted aggregation method is practical for on-line, near-real-time dynamic and stochastic scheduling problems with multiple conflicting objectives. As such, the weighted aggregation method will be considered in the design of the NEXTS decision policy.

The research issues, after having decided on the use of the weighted aggregation method, have to do with (i) which rules to include for optimizing multiple objectives of interest, (ii) how user preferences should be incorporated into decision-making, and (iii) how the weights used in the weighted aggregation of rules can be optimized. These issues will be addressed keeping in mind the designed decision policy should be suitable for on-line, near-real-time application.

Issue (i) will be addressed in this chapter whereas issues (ii) and (iii) will be discussed in the next chapter.

5.2.1 INCLUSION OF RULES

Subbiah and Bodenstab (1992) reported that dispatching ignores the interdependencies that clearly exist between machines when each machine operates as an independent unit guided by dispatching rules that determines which job to process next in the queue. In fact, numerous studies have thereafter developed rules that make use of upstream and downstream information to address the machine dependency issue in decision-making. These rules are classified in this research as ‘look upstream’ and ‘look downstream’ rules.

For example, Yurtsever and Pierce (1998) have reported implementing the ‘Idle Time’ event in the Motorola’s Graphical Manufacturing Monitoring System (GraMMS) by referencing upstream information (i.e. future arrivals) to give privilege to an urgent job that would arrive at the end of the idle time period. Although the implementation of GraMMS in Motorola’s Advanced Products Research and Development Laboratory has resulted in productivity improvements, it is not known how much this idle event has contributed to the improvements.

The Shortest Time until Next Visit (STNV) dispatch rule (Wein, 1988) selects the job with the shortest expected processing time until its next visit to the same bottleneck station to prevent starvation at the bottleneck station. Work content In Next Queue (WINQ) gives preference to jobs that move on to queues with the least backlog, rather than to speed up a job now that could later be stopped in a congested queue (Haupt, 1989). The STNV and WINQ (and other variants) are

examples of look downstream rules as these rules make use of downstream information (i.e. length or expected waiting time in queue for subsequent downstream operations of a job)

Robinson et al. (1995) presented a control strategy that exploits both upstream and downstream information on the current and expected states of a semiconductor manufacturing facility to make intelligent batch size decisions in which significant improvements in cycle time is realized.

Judging from the reported benefits of considering look upstream and look downstream rules, some simplified form of these rules (based on machines and buffer states) are modeled using Markov process modeling methodology. Although a wide range of modeling techniques are available, most analytical models of manufacturing systems are Markovian in nature (Viswanadham and Narahari, 1992). The Markov process model is studied first before the simulation model is considered because the former is able to provide exact analytical results (i.e. no need for many simulation replications) for short production lines (Papadopoulos and Heavey, 1996). The major disadvantage of Markov modeling is the exponential growth of the state space as the system to be modeled increases in size.

5.3 MARKOV PROCESS MODELING FOR PRELIMINARY STUDY OF LOOK UPSTREAM, LOOK DOWNSTREAM, AND LOOK INTO QUEUE

The main idea behind these look upstream (or look-behind) and look downstream (or look-ahead) rules is to make efficient use of upstream and downstream information (e.g. future arrivals, WIP levels, etc.) for better decision-making to

improve factory performance. Dispatching rules that do not incorporate look upstream and look downstream at upcoming work tend to result in myopic decisions being made (Fowler et al., 2001).

The main objective of this Markov modeling study is to identify if there are any potential benefits to use look upstream and look downstream rules, and also to evaluate whether a Markov process model could be used as a look-ahead tool to predict some performances that can be used to guide the optimization of weights.

These rules are modeled using deterministic processing time simplified job shop model adapted from the transfer line model described in Gershwin (1994). A state-space modeling and analysis methodology proposed by Tan (2002) is subsequently applied to generate the state space models automatically. Table 5.2 summarizes the notations and abbreviations used in the description of the Markov process model.

Table 5.2: Description of notations and abbreviations used in Markov process modeling

Notations	Description
M_1 to M_6	Refers to a particular machine in the model
B_1 to B_6	Refers to a particular buffer in the model
B_7 to B_{10}	Refers to a particular dummy buffer used in the model
T_j	Job Type j
$n(M)$	Number of machines in the model
$n(B)$	Number of buffers in the model
α_i	Repair state of machine M_i , $\alpha_i \in \{0,1\}$. $\alpha_i = 1$ implies that M_i is up (operational), and $\alpha_i = 0$ implies that M_i is down (under repair).
N_i	Maximum capacity of buffer B_i
$n_i(t)$	Number of jobs in buffer B_i at time t
S	Initial state space
S'	Reduced state space
$X(t)$	Current state of the system at time t

Φ_{ij}	Indicator variable that describes the control and operational rules on machine M_i for processing job type T_j . If the flow of T_j out of M_i is not restricted, $\Phi_{ij} = 1$. Otherwise, $\Phi_{ij} = 0$.
$I_{ij}(t)$	Indicates whether a part of type T_j arrives at buffer B_i from upstream at time t . $I_{ij}(t)$ is policy independent (i.e. applies across all the Look Around rules – see Section 5.3.1)
$I_{ij}^*(t)$	Indicates whether a part of type T_j arrives at buffer B_i from upstream at time t . $I_{ij}^*(t)$ is policy-dependent.
$d_i(t)$	Refers to the job type of the jobs in the dummy buffers B_i
J	Total number of job types considered in the system
$\delta(t)$	Denotes the job type processed by machine M_3 at time t . $\delta(t) = 0$ implies no job is processed by M_3 at time t .
$D(t)$	State vector for the dummy buffers at time t
$q_{a,b}$	Transition probabilities between state a and state b
p_i	Probability of failure of machine M_i during a time unit
f_i	Probability of failure of an operational machine M_i during a time unit, $f_i = p_i \Phi_i$
r_i	Probability of a failed machine M_i being repaired during a time unit
Q	Probability matrix
W	Reward matrix used to denote the machine and performance measure of interest
$w_{X(t),i}$	Reward of machine M_i at a particular state $X(t)$
$\pi^{(k)}$	Probability vector obtained after k transitions
E	Production rate (or throughput) vector for all the machines
L	Average buffer levels vector
e_i	Production rate of machine M_i
l_i	Average buffer level of buffer B_i
μ_{CT}	Average cycle time for the simplified job shop
A_i	Availability of machine M_i

5.3.1 MODEL DESCRIPTION

The simplest deterministic processing time job shop model for studying the various Look Downstream (LD), Look Upstream (LU), and Look into Queue (LQ) rules can be represented using a simplified job shop (with no re-circulation) made up of

five machines (i.e. M_1 to M_5), six buffers (i.e. B_1 to B_6), and four dummy buffers (i.e. B_7 to B_{10}) as shown in Figure 5.2.

Figure 5.2: Markov model of a simplified job shop with no re-circulation

The term “Look Around rules” is coined to refer to all the LD, LU, and LQ rules. Machines M_1 to M_5 are subjected to breakdowns with probability of failure and being repaired during a time unit given by p_i and r_i respectively. As the release machines (i.e. M_a and M_b) are not subjected to breakdowns, they are considered in the update of buffers but are not included in the system structure to reduce the state space. Two job types (i.e. T_1 and T_2) are considered, each of which follows a predetermined route. Jobs of type T_1 follow route: $M_a \rightarrow B_1 \rightarrow M_1 \rightarrow B_3 \rightarrow M_3 \rightarrow B_5 \rightarrow M_4$ while jobs of type T_2 follow route: $M_b \rightarrow B_2 \rightarrow M_2 \rightarrow B_4 \rightarrow M_3 \rightarrow B_6 \rightarrow M_5$. After each job finishes its last processing step, it flows out of the system. In the model, machine M_3 is capable of processing either a T_1 or T_2 job at any one time. The decision as to which job type M_3 will process next has to do with the rule (see Table 5.3) that is used. For other machines, FIFO+ is applied (without the need to use dummy buffers).

Table 5.3: Look Around rules that are applied to machine M_3

Look Around Rules		Consider the states of	Resolve ties (if any) using (states of)
Look Upstream (LU)	LU_B	B_1 and B_2	FIFO+ (Dummy buffer B_7)
	LU_M	M_1 and M_2	FIFO+ (Dummy buffer B_7)
	LU_BM	B_1, B_2, M_1 and M_2	FIFO+ (Dummy buffer B_7)
Look into Queue (LQ)	LQ_B	B_3 and B_4	FIFO+ (Dummy buffer B_7)
	FIFO+	Dummy buffer B_7	-
Look Downstream (LD)	LD_B	B_5 and B_6	FIFO+ (Dummy buffer B_7)
	LD_M	M_4 and M_5	FIFO+ (Dummy buffer B_7)
	LD_BM	B_5, B_6, M_4 and M_5	FIFO+ (Dummy buffer B_7)

In this study, for simplicity, no decomposition methods (e.g. Gershwin, 1994 and Bonvik, 1996) are applied. However, this implies that buffer capacity has to be kept to a maximum of two in order to prevent an explosion of state space. This scenario could represent a case of strict tool-to-tool delivery (with no stockers to serve as storage between tools, or machine), with each tool having two loadports (local storage on tool) to hold a job in process and a waiting job. Two buffers B_3 and B_4 are used to differentiate the type of jobs waiting for processing by M_3 . With at most four jobs in buffers B_3 and B_4 , four dummy buffers are used to track the First-In, First-Out (FIFO+) of jobs waiting in B_3 and B_4 .

FIFO+ can be viewed as an improved version of FIFO such that if the job that is first in queue could not be processed because of blockage, other jobs in queue (of the other part type) would be processed instead if there is no blockage. The states of the dummy buffers represent the job type. The sequence of job types to be processed (if based on FIFO+) is listed in the order of the states of B_7, B_8, B_9 , followed by B_{10} . The state of B_7 indicates the job type that M_3 will process next (if M_3 is not down and/or blocked). FIFO+ is used to break ties in the Look Around rules (if required).

5.3.2 ASSUMPTIONS

The processing duration of each job at any machine is taken to be one time unit. All the stations start their processing operation at the same instant. Transportation times between machines and buffers are assumed to be negligible. Release machines (i.e. M_a and M_b) are never starved and the last machines (i.e. M_4 and M_5) are never blocked. It is also assumed that there are no sequence-dependent setups and no scrapping of jobs during processing. Operational failures are assumed (i.e. a machine will not breakdown if it is not processing any job). When a machine fails, the job it was processing is taken to its respective upstream buffer to wait till the machine is repaired. Repairs and failures are assumed to occur at the beginning of time units, whereas changes in buffer levels are assumed to take place at the end of the time units.

5.3.3 DETAILED MARKOV MODELS

Let $n(M) = 5$ and $n(B) = 6$ denote the number of machines and buffers in the model respectively. Let the repair state of Machine M_i be $\alpha_i \in \{0,1\}$, where $\alpha_i = 1$ and $\alpha_i = 0$ implies that M_i is up (operational) and down (under repair) respectively. Let the number of jobs in Buffer B_i (for $i = 1, \dots, 6$) at time t be $n_i(t)$, where $0 \leq n_i(t) \leq N_i$, and N_i is the maximum capacity of B_i . The repair state of the five machines are represented as a 5-tuple: $(\alpha_1, \dots, \alpha_5) \in A = \{0,1\}^5$ while the buffer state of the simplified job shop, made up of six buffers with capacity two and four dummy buffers with possible states of $\{0,1,2\}$, is given by $(n_1, \dots, n_{10}) \in B = \{0,1,2\}^{10}$. The state of the system at time t is described as $X(t) = (n_1, \dots, n_{10}, \alpha_1, \dots, \alpha_5)$, $X(t) \in S' = A \times B$ with state space S' having

$n(S') = 2^5 \times 3^{10} = 1,889,568$ states before state space reduction. The state space (see Section 5.3.3.4) after reduction is referred to as S where $n(S) < n(S')$.

Let the current state of the system be $X(t) = b = (n_1, \dots, n_{10}, \beta_1, \dots, \beta_5)$ and the next state be $X(t+1) = a = (n_1, \dots, n_{10}, \alpha_1, \dots, \alpha_5)$. For a given current state, all possible changes in the machine states are evaluated and the corresponding changes in the buffer levels are determined in accordance to policy and rules as described in Sections 5.3.3.1 to 5.3.3.3. The transition probability is then computed in Section 5.3.3.5 in accordance to the control and operational rules. The rewards for machines given in Section 5.3.3.6 are used to indicate the chosen states of the machines of interest for the computation of the performance measure. Section 5.3.3.7 sums up how the steady state performance measures are computed using the iterative power method.

5.3.3.1 Control and Operational Rule Indicator Variable

Let $\Phi_{i,j}$ be an indicator variable that describes the control and operational rules on machine M_i for processing job type T_j . When no job type differentiation is required, the indicator variable for machine M_i is represented as Φ_i . If the flow out of M_i is not restricted, $\Phi_{i,j}$ is 1, otherwise it is 0. Equations (5.1) to (5.6) are the same for all the Look Around rules listed in Table 5.3.

For release machines M_a and M_b :

$$\Phi_{a,1} = \begin{cases} 1 & \text{if } n_1(t) < N_1 \\ 0 & \text{otherwise} \end{cases} \quad (5.1)$$

$$\Phi_{b,2} = \begin{cases} 1 & \text{if } n_2(t) < N_2 \\ 0 & \text{otherwise} \end{cases} \quad (5.2)$$

For machines M_1 and M_2 :

$$\Phi_i = \Phi_{i,i} = \begin{cases} 1 & \text{if } n_i > 0, n_{i+2}(t) < N_{i+2} \quad \text{where } i = 1, 2 \\ 0 & \text{otherwise} \end{cases} \quad (5.3)$$

For machines M_3 :

$$\Phi_{3,1} = \begin{cases} 1 & \text{if } n_3 > 0, n_5(t) < N_5 \\ 0 & \text{otherwise} \end{cases} \quad (5.4)$$

$$\Phi_{3,2} = \begin{cases} 1 & \text{if } n_4 > 0, n_6(t) < N_6 \\ 0 & \text{otherwise} \end{cases} \quad (5.5)$$

$$\Phi_3 = \Phi_{3,1} \vee \Phi_{3,2} \quad (5.6)$$

For machines M_4 and M_5 :

$$\Phi_i = \Phi_{i,i-3} = \begin{cases} 1 & \text{if } n_{i+1}(t) > 0 \quad \text{where } i = 4, 5 \\ 0 & \text{otherwise} \end{cases} \quad (5.7)$$

The difference of each Look Around rule affects how the buffer levels are updated. The buffer levels of B_1 to B_6 increase or decrease by one unit, or remain unchanged depending on the repair state of its upstream and/or downstream machines, the operational rules (see Equations (5.1) to (5.7)), and the Look Around rule applied. The buffer levels of B_7 to B_{10} are updated in a slightly different manner. The following describes how buffer levels of B_1 to B_6 are updated.

5.3.3.2 Update of Buffer Levels B_1 TO B_6

Let both $I_{i,j}(t)$ and $I_{i,j}^*(t)$ indicate whether a part of type T_j arrives at buffer B_i from upstream at time t . $I_{i,j}(t)$ is policy-independent (i.e. applies across all Look

Around rules) and is defined in Equation (5.8). $I_{i,j}^*(t)$ is policy-dependent and will be subsequently addressed in accordance with each Look Around rule.

$$I_{i,j}(t+1) = \begin{cases} 1 & \text{if } \alpha_i(t+1) = 1 \text{ and } \Phi_{i,j}(t) = 1 \\ 0 & \text{otherwise} \end{cases} \quad (5.8)$$

Levels of buffer (i) B_1 , (ii) B_2 , (iii) B_3 and B_4 , and (iv) B_5 and B_6 are updated using Equations (5.9) to (5.12) respectively.

$$n_1(t+1) = n_1(t) + I_{a,1}(t+1) - I_{1,1}(t+1) \quad (5.9)$$

$$n_2(t+1) = n_2(t) + I_{b,2}(t+1) - I_{2,2}(t+1) \quad (5.10)$$

$$n_i(t+1) = n_i(t) + I_{i-2,i-2}(t+1) - I_{3,i-2}^*(t+1), \quad i = 3, 4 \quad (5.11)$$

$$n_i(t+1) = n_i(t) + I_{3,i-4}^*(t+1) - I_{i-1,i-4}(t+1), \quad i = 5, 6 \quad (5.12)$$

The following discusses how the values of $I_{i,j}^*(t)$ in Equations (5.11) and (5.12) are determined. Table 5.4 summarizes the conditions as applied to each Look Around rule to decide whether machine M_3 should process job of type T_1 or T_2 next. This is required in order to update the buffer levels of B_3 to B_6 appropriately. The logic of consideration for each LU, LQ, and LD policies (refer to Table 5.4) is chosen given the limitation of Markov models and does not represent the best possible implements available. However, it does provide an insight into the usage of various Look Around rules (based on machine and buffer states) for making a next job selection decision.

In LU_B, the priority is given to the job type that has more jobs waiting to be processed upstream at M_1 and M_2 . In LU_M, the priority is given to the job type that utilizes the upstream machine (refers to M_1 and M_2) that is not down. LU_BM

is a combination of both LU_B and LU_M. In LD_B, the priority is given to the job type that has less jobs waiting to be processed downstream at M_4 and M_5 . In LD_M, the priority is given to the job type that utilizes the downstream machine (refers to M_4 and M_5) that is not down. Similarly, LD_BM is a combination of both LD_B and LD_M. In LQ_B, the priority is given to the job type that has more jobs waiting in queue.

Table 5.4: Conditions considered in each Look Around rule to decide which job type is given priority for processing on machine M_3

Look Around Rules		Condition 1 (C1)	Condition 2 (C2)	Condition 3 (C3)
		If true, priority will be given to process T1 jobs over T2 jobs	If true, priority will be given to process T2 jobs over T1 jobs	For all except FIFO+: If true, use FIFO+ C1-C3 to decide / For FIFO+: M3 is left idle.
Look Upstream (LU)	LU_B	$n_1(t) > n_2(t)$	$n_1(t) < n_2(t)$	$n_1(t) = n_2(t)$
	LU_M	$\alpha_1(t+1) > \alpha_2(t+1)$	$\alpha_1(t+1) < \alpha_2(t+1)$	$\alpha_1(t+1) = \alpha_2(t+1)$
	LU_BM	$(\alpha_1(t+1) > \alpha_2(t+1)) \vee [(\alpha_1(t+1) = \alpha_2(t+1)) \wedge (n_1(t) > n_2(t))]$	$(\alpha_1(t+1) < \alpha_2(t+1)) \vee [(\alpha_1(t+1) = \alpha_2(t+1)) \wedge (n_1(t) < n_2(t))]$	$[(\alpha_1(t+1) = \alpha_2(t+1)) \wedge (n_1(t) = n_2(t))]$
Look into Queue (LQ)	LQ_B	$n_3(t) > n_4(t)$	$n_3(t) < n_4(t)$	$n_3(t) = n_4(t)$
	FIFO+	$n_7(t) = 1$	$n_7(t) = 2$	$n_7(t) = 0$
Look Downstream (LD)	LD_B	$n_5(t) < n_6(t)$	$n_5(t) > n_6(t)$	$n_5(t) = n_6(t)$
	LD_M	$\alpha_4(t+1) > \alpha_5(t+1)$	$\alpha_4(t+1) < \alpha_5(t+1)$	$\alpha_4(t+1) = \alpha_5(t+1)$
	LD_BM	$(\alpha_4(t+1) > \alpha_5(t+1)) \vee [(\alpha_4(t+1) = \alpha_5(t+1)) \wedge (n_5(t) < n_6(t))]$	$(\alpha_4(t+1) > \alpha_5(t+1)) \vee [(\alpha_4(t+1) = \alpha_5(t+1)) \wedge (n_5(t) > n_6(t))]$	$[(\alpha_4(t+1) = \alpha_5(t+1)) \wedge (n_5(t) = n_6(t))]$

The flow chart in Figure 5.3 summarizes how Table 5.4 can be used to determine the values of $I_{i,j}^*(t)$. If machine M_3 is not down, Condition 1 (C1) is first checked. If C1 is true, priority will be given to process job of type T_1 . This does not, however, mean that job of type T_1 will definitely be processed. If a job type that has been given priority could not be processed (e.g. because buffer B_5 is full and therefore blocks the processing of type T_1 job), the other job type will be considered for processing next provided the processing flow is not restricted. Equations (5.4) and (5.5) give the flow restriction for job type T_1 and T_2 respectively. If C1 is false,

Condition 2 (C2) is checked next. If C2 is true, priority will be given to process job of type T_2 . Similarly, if C2 is false, Condition 3 (C3) is then checked. If C3 is true for all the LU and LD policies, it implies that there is a tie between the selection to process T_1 and T_2 jobs. In this case, FIFO+ is used to break tie and C1 to C3 of FIFO+ is re-applied to determine the value of $I_{i,j}^*(t)$. If C3 is true for FIFO+ policy, it implies that there is no job to be processed.

Figure 5.3: Definition of the policy-dependent term, $I_{i,j}^*(t)$

5.3.3.3 Update of Buffer Levels B_7 TO B_{10}

Let the state of dummy buffer B_i (for $i = 7, \dots, 10$) at time t refers to the job type of the jobs, arranged in order of arrival to B_3 and B_4 , be denoted by $d_i(t)$, where $0 \leq d_i(t) \leq J$, and J is the total number of job types considered in the system. Let $D(t) = \{d_7(t), d_8(t), d_9(t), d_{10}(t)\}$ be the state vector for the dummy buffers at time t . Let $\delta(t) \in \{0, \dots, J\}$ denotes the job type processed by M_3 at time t . $\delta(t) = 0$ implies

that no job is processed whereas $1 \leq \delta(t) \leq J$ implies that a job of type $\delta(t)$ is processed by M_3 .

Since two job types (i.e. $J = 2$) are considered, $d_i(t) \in \{0, 1, 2\}$. For example, if FIFO+ is to be used, $D(t) = \{1, 1, 2, 0\}$ indicates that M_3 should process a T_1 job next if it is not down, and not blocked (i.e. B_5 is not full). If a T_1 job could not be processed due to blockage, T_2 will be processed instead provided that B_6 is not full so that M_3 will not be left to idle. Figure 5.4 and Figure 5.5 show how the states of the dummy buffers are updated at each time step following (i) a job departure from either B_3 or B_4 , and (ii) a job arrival to B_3 and/or B_4 respectively.

Figure 5.4: Update of the states of dummy buffers following a job departure from B_3 or B_4

```

If  $I_{1,1}(t+1) = 1 \vee I_{2,2}(t+1) = 1$ 
  Let  $\lambda(t) = \min \{i \in \Gamma : d_i(t) = 0\}$ , where  $\Gamma = \{7, 8, 9, 10\}$ 
  If  $I_{1,1}(t+1) > I_{2,2}(t+1)$ 
     $d_{\lambda(t)}(t+1) = 1$ 
  Else If  $I_{1,1}(t+1) < I_{2,2}(t+1)$ 
     $d_{\lambda(t)}(t+1) = 2$ 
  Else
     $d_{\lambda(t)}(t+1) = I_{1,1}(t+1)$ 
     $d_{\lambda(t)+1}(t+1) = 2(I_{2,2}(t+1))$ 
  End
End

```

Figure 5.5: Update of the states of dummy buffers following a job arrival at B_3 and/or B_4

5.3.3.4 State Space Reduction

In the Markov chain model, some of the states are never visited. Therefore, the state space can be reduced in size by eliminating the unvisited states in accordance to the following conditions:

$$(n_1, \dots, n_6) \in \{(n_1, \dots, n_6) \mid R_1 \wedge R_2 \wedge R_3\} \quad (5.13)$$

where R_1 , R_2 , and R_3 are defined as:

- The dummy buffers B_7 to B_{10} are used to store the job type of the jobs available in B_3 and B_4 in the FIFO+ sequence. If both B_3 and B_4 are not full, then $D(t) = \{d_7(t), d_8(t), d_9(t), d_{10}(t)\}$ will contain one or more zeros in it. These zeros will have to come after all the job types are stored within $D(t)$ in FIFO+ sequence (from left to right in the $D(t)$ array). For example, if there is only one job in B_3 and B_4 each and that job of type T_2 arrives before job of type T_1 , then $D(t) = \{2, 1, 0, 0\}$ and not $D(t) = \{0, 2, 1, 0\}$ or $D(t) = \{0, 0, 2, 1\}$. This is represented as R_1 .

$$R_1 : (d_i(t) = 0) \geq (d_j(t) = 0), \quad \forall i, j \in \{7, 8, 9, 10\} \text{ and } i > j \quad (5.14)$$

- Since the dummy buffers B_7 to B_{10} are used to store the job type of the jobs available in B_3 and B_4 , the number of jobs of type T_1 as represented in $D(t) = \{d_7(t), d_8(t), d_9(t), d_{10}(t)\}$ must add up to be the same as the number of jobs in buffer B_3 . Similarly, the number of jobs of type T_2 as represented in $D(t)$ must also add up to be the same as the number of jobs in buffer B_4 .

$$R_2 : \left[\sum_{i=7}^{10} (d_i(t) = 1) = n_3(t) \right] \wedge \left[\sum_{i=7}^{10} (d_i(t) = 2) = n_4(t) \right] \quad (5.15)$$

- $I_{1,1}(t)$ taking a value of '1' would indicate that a part of type T_1 arrives at buffer B_1 at time t and $I_{2,2}(t)$ taking a value of '1' would indicate that a part of type T_2 arrives at buffer B_2 at time t . In order to ensure the logical correctness of the model, R_3 is used to check that there are sufficient buffer spaces to hold the arriving jobs. Hence, the number of empty buffer spaces must be greater or equal to the number of arriving jobs.

$$R_3 : \sum_{i=7}^{10} (d_i(t) = 0) \geq I_{1,1}(t) + I_{2,2}(t) \quad (5.16)$$

5.3.3.5 Transition Probabilities

The probability matrix, $Q = \{q_{a,b}\}$, $a, b \in S$ describes the transition probabilities between states where

$$q_{a,b} = P[X(t+1) = a \mid X(t) = b] \quad \forall t, a, b. \quad (5.17)$$

In order to account for the operational failures, the probability of failure of an operational machine during a time period is computed as $f_i = p_i \Phi_i$ while the

probability of a failed machine being repaired during a time period is given by r_i . Table 5.5 shows the four possible transitions and their corresponding transition probabilities.

Table 5.5: Transition probabilities of a particular machine M_i

Current State of M_i (at time t)	Next State of M_i (at time $t+1$)	Transition Probability
0	0	$1 - r_i$
0	1	r_i
1	0	f_i
1	1	$1 - f_i$

Hence, if the current and next state of the system is denoted respectively as $X(t) = b = (n_1, \dots, n_{10}, \beta_1, \dots, \beta_5)$ and $X(t+1) = a = (n_1, \dots, n_{10}, \alpha_1, \dots, \alpha_5)$, the probability matrix Q can be computed using

$$q_{a,b} = \prod_{i=1}^5 \left(f_i^{1-\alpha_i} (1-f_i)^{\alpha_i} \right)^{\beta_i} \left((1-r_i)^{1-\alpha_i} r_i^{\alpha_i} \right)^{1-\beta_i}, \quad a \neq b, \quad (5.18)$$

with Equation (5.18) repeated $2^5 = 32$ times for all possible repair states in order to obtain all the transitions from a given state.

5.3.3.6 Update of Rewards for Machine

The $n(S) \times 5$ reward matrix, W is used to denote the machine and performance measure of interest and is denoted as

$$W = \{w_{X(t),i}\}, \quad X(t) \in S, \quad i = 1, \dots, n(M) \quad (5.19)$$

For example, if the production rate of the first machine of a production line is of interest, then the states where the first station is up and not blocked are indicated with a value of '1'. In our study, it is necessary to find the production rates of all machines in the simplified job shop. Hence, if machine M_i is operational and the

control and operational rules allows flow into the respective downstream buffer(s), the reward of this machine at a particular state $X(t)$ is 1, otherwise it is 0. That is,

$$w_{X(t),i} = \begin{cases} 1 & \text{if } \alpha_i = 1 \text{ and } \Phi_i = 1 \text{ at state } X(t) \\ 0 & \text{otherwise} \end{cases} \quad (5.20)$$

5.3.3.7 Numerical Solution

The generation of the state space models and the application of the decision policies are coded in MATLAB. The iterative power method is used for solving large Markov chains. For computational efficiency, the transition probability matrix, Q is transformed into a sparse matrix made up of non-zero transition probabilities between states. Let $n(Q)$ be the maximum size (length) of Q .

The $1 \times n(Q)$ initial probability vector, $\pi^{(0)}$ is randomly generated such that all the $n(Q)$ elements in the vector sum up to one. For any integer k , the state of the system after k transitions is obtained by multiplying the probability vector obtained after $(k-1)$ transitions by Q , that is,

$$\pi^{(k)} = \pi^{(k-1)}Q = \pi^{(k-2)}Q^2 = \dots = \pi^{(0)}Q^k. \quad (5.21)$$

Since the Markov chain is finite, aperiodic, and irreducible, the probability vector $\pi^{(k)}$ converges to the stationary probability vector π regardless of the choice of initial vector. Hence,

$$\lim_{k \rightarrow \infty} \pi^{(k)} = \pi \quad (5.22)$$

For an ergodic Markov chain, the stationary probability vector is referred to as the *steady-state* probability vector, signifying that sufficient transitions have occurred as to erase the influence of the starting state. In MATLAB, convergence is deemed to have taken place once the following is satisfied:

$$\pi^{(k)} - \pi^{(k-1)} \leq 1 \times 10^{-5}. \quad (5.23)$$

The steady-state probability vector $\pi = \pi^{(k)}$ is obtained.

The production rate (or throughput) vector for all the machines (E) and average buffer levels vector for all buffers (L) are the steady state performance measures derived from the steady state probabilities using the equations below:

$$E = \{e_1, \dots, e_5\} = \pi \times W \quad (5.24)$$

The production rate of M_3 is effectively made up of the throughput of T1 and T2 jobs:

$$e_3 = e_{3,T1} + e_{3,T2}, \quad (5.25)$$

By conservation of flow,

$$e_{3,T1} = e_1 = e_4, \quad (5.26)$$

and

$$e_{3,T2} = e_2 = e_5, \quad (5.27)$$

should hold.

Each element in $L = \{l_1, \dots, l_6\}$ is computed using

$$l_i = \sum_{j=1}^{N_i} (\pi \times \bar{c}_i \times j), \quad i = 1, \dots, 6 \quad (5.28)$$

where $\bar{c}_i = \{c_{X(t),i}\}$ is a $n(S) \times 1$ vector corresponding to whether the value of the column vector of buffer B_i in state space matrix S is equal to the buffer level j . Let $c_{X(t),i}$ be a particular element in $\bar{c}_i$ such that

$$c_{X(t),i} = \begin{cases} 1 & \text{if } n_i = i, n_i \in X(t), X(t) \in S \\ 0 & \text{otherwise} \end{cases}. \quad (5.29)$$

Little's Law (Little, 1961) is subsequently applied to compute the average cycle time for part type T1 and T2 respectively.

$$\mu_{CT}^{T1} = \frac{l_1}{e_1} + \frac{l_3}{e_{3,T1}} + \frac{l_5}{e_4} = \frac{l_1 + l_3 + l_5}{e_4} \quad (5.30)$$

$$\mu_{CT}^{T2} = \frac{l_2}{e_2} + \frac{l_4}{e_{3,T2}} + \frac{l_6}{e_5} = \frac{l_2 + l_4 + l_6}{e_5} \quad (5.31)$$

The average cycle time for the simplified job shop is given by

$$\mu_{CT} = (\mu_{CT}^{T1} + \mu_{CT}^{T2})/2. \quad (5.32)$$

5.3.4 VERIFICATION

Intermediate results are traced for manual checking to ensure the coding is correct.

In addition, the conservation of flow is checked using Equations (5.25) to (5.27).

The isolated efficiency or availability A_i of machine M_i is given by

$$A_i = \frac{r_i}{r_i + p_i}, \quad i = 1, \dots, 5 \quad (5.33)$$

which represents the long-run fraction of time that M_i is operational. Therefore,

$$A_i \geq e_i \quad (5.34)$$

and it also holds. Further verification is performed via a similar simulation model.

There is less than 2% difference in the average cycle time between the results obtained from the markov process model and the steady-state simulation results (based on ten replications). This percentage difference is considered small considering the random variations present in simulation models.

5.3.5 RESULTS & DISCUSSION

The Markov process model presented above is coded using MATLAB. Experiments are carried out with different settings of the failure and repair probabilities, machine availability, and buffer capacities.

5.3.5.1 Varying Failure and Repair Probabilities (Same Machine Availability Across All Machines)

Different sets of failure and repair probabilities that give the same machine availability of 0.909 (computed using Equation (5.33)) across all machines are experimented. All the six buffers are assumed to have a capacity of two and the four dummy buffers are set to hold possible states of $\{0,1,2\}$. The average cycle time is computed using Equation (5.32) and is tabulated in Table 5.6. The best average cycle time for each set of failure and repair probabilities are presented in bold. As can be seen from Table 5.6, the LD_M (look downstream at machine states) rule performs the best in all the tested cases.

Table 5.6: Average cycle time for different failure and repair probabilities (with same machine availability of 0.909)

Failure Probabilities	Repair Probabilities	LD_BM	LD_B	LD_M	LU_BM	LU_B	LU_M	LQ_B	FIFO+
failure = [0.08 0.08 0.08 0.08 0.08 0.08];	repair = [0.8 0.8 0.8 0.8 0.8];	7.7857	7.8022	7.7744	7.8460	7.8047	7.8324	7.8042	7.8044
failure = [0.06 0.06 0.06 0.06 0.06 0.06];	repair = [0.6 0.6 0.6 0.6 0.6];	7.7829	7.8169	7.7766	7.8645	7.8211	7.8566	7.8200	7.8207
failure = [0.04 0.04 0.04 0.04 0.04 0.04];	repair = [0.4 0.4 0.4 0.4 0.4];	7.7965	7.8474	7.7930	7.8966	7.8501	7.8929	7.8497	7.8498
failure = [0.02 0.02 0.02 0.02 0.02 0.02];	repair = [0.2 0.2 0.2 0.2 0.2];	7.8222	7.8865	7.8214	7.9414	7.8876	7.9409	7.8874	7.8870
failure = [0.01 0.01 0.01 0.01 0.01 0.01];	repair = [0.1 0.1 0.1 0.1 0.1];	7.8390	7.9077	7.8387	7.9689	7.9077	7.9688	7.9078	7.9077
failure = [0.0067 0.0067 0.0067 0.0067 0.0067];	repair = [0.067 0.067 0.067 0.067 0.067];	7.8449	7.9146	7.8448	7.9786	7.9146	7.9784	7.9146	7.9146
failure = [0.005 0.005 0.005 0.005 0.005];	repair = [0.05 0.05 0.05 0.05 0.05];	7.8481	7.9177	7.8480	7.9834	7.9177	7.9834	7.9176	7.9177
Six buffers with capacity two and four dummy buffers with possible states of $\{0,1,2\}$. (i.e. Buffer: [2 2 2 2 2], Dummy Buffer: [2 2 2 2])									

5.3.5.2 Varying Machine Availability Across Machines

In this set of experiments, different combination of machine availability of 0.909 (i.e. failure probability, $p_i = 0.02$, repair probability, $r_i = 0.2$) and 0.714 (i.e. $p_i = 0.04$, $r_i = 0.1$) are applied across the machines. All the six buffers are assumed to have a capacity of two and the four dummy buffers are set to hold possible states of $\{0,1,2\}$. The results based on average cycle time are presented in Table 5.7.

Table 5.7: Average cycle time for varying machine variability across machines

Failure Probabilities	Repair Probabilities	LD_BM	LD_B	LD_M	LU_BM	LU_B	LU_M	LQ_B	FIFO+
failure = [0.02 0.02 0.02 0.02 0.02 0.02];	repair = [0.2 0.2 0.2 0.2 0.2 0.2];	7.8222	7.8865	7.8214	7.9414	7.8876	7.9409	7.8874	7.8870
failure = [0.02 0.02 0.02 0.02 0.04 0.04];	repair = [0.2 0.2 0.2 0.2 0.1 0.1];	8.1326	8.2814	8.1313	8.3314	8.2841	8.3294	8.2821	8.2831
failure = [0.02 0.02 0.02 0.04 0.02 0.02];	repair = [0.2 0.2 0.2 0.1 0.2 0.2];	8.1325	8.2816	8.1329	8.3316	8.2845	8.3322	8.2850	8.2859
failure = [0.02 0.02 0.02 0.04 0.04 0.04];	repair = [0.2 0.2 0.2 0.1 0.1 0.1];	8.7156	8.9122	8.7143	8.9562	8.9147	8.9547	8.9153	8.9148
failure = [0.02 0.04 0.02 0.02 0.02 0.02];	repair = [0.2 0.1 0.2 0.2 0.2 0.2];	7.7005	7.7557	7.7017	7.8695	7.7579	7.8686	7.7595	7.7593
failure = [0.02 0.04 0.02 0.02 0.02 0.04];	repair = [0.2 0.1 0.2 0.2 0.1 0.1];	7.9935	8.1250	7.9925	8.2252	8.1270	8.2242	8.1262	8.1264
failure = [0.02 0.04 0.02 0.04 0.02 0.02];	repair = [0.2 0.1 0.2 0.1 0.2 0.2];	8.2050	8.3313	8.2081	8.4255	8.3351	8.4267	8.3372	8.3389
failure = [0.02 0.04 0.02 0.04 0.04 0.04];	repair = [0.2 0.1 0.2 0.1 0.1 0.1];	8.7265	8.8988	8.7274	8.9823	8.9010	8.9813	8.9032	8.9027
failure = [0.04 0.02 0.02 0.02 0.02 0.02];	repair = [0.1 0.2 0.2 0.2 0.2 0.2];	7.7003	7.7556	7.6993	7.8694	7.7569	7.8677	7.7567	7.7558
failure = [0.04 0.02 0.02 0.02 0.04 0.04];	repair = [0.1 0.2 0.2 0.2 0.1 0.1];	8.2050	8.3308	8.2040	8.4251	8.3341	8.4225	8.3305	8.3314
failure = [0.04 0.02 0.02 0.04 0.02 0.02];	repair = [0.1 0.2 0.2 0.1 0.2 0.2];	7.9935	8.1255	7.9929	8.2250	8.1266	8.2260	8.1280	8.1283
failure = [0.04 0.02 0.02 0.04 0.04 0.04];	repair = [0.1 0.2 0.2 0.1 0.1 0.1];	8.7262	8.8989	8.7246	8.9821	8.9014	8.9810	8.9006	8.8995
failure = [0.04 0.04 0.02 0.02 0.02 0.02];	repair = [0.1 0.1 0.2 0.2 0.2 0.2];	7.6986	7.7477	7.6983	7.8843	7.7477	7.8835	7.7484	7.7486
failure = [0.04 0.04 0.02 0.02 0.04 0.04];	repair = [0.1 0.1 0.2 0.2 0.1 0.1];	8.1581	8.2695	8.1560	8.3881	8.2718	8.3853	8.2701	8.2696
failure = [0.04 0.04 0.02 0.04 0.02 0.02];	repair = [0.1 0.1 0.2 0.1 0.2 0.2];	8.1585	8.2699	8.1590	8.3884	8.2723	8.3893	8.2738	8.2748
failure = [0.04 0.04 0.02 0.04 0.04 0.04];	repair = [0.1 0.1 0.2 0.1 0.1 0.1];	8.8125	8.9634	8.8118	9.0646	8.9647	9.0642	8.9665	8.9654
Six buffers with capacity two and four dummy buffers with possible states of $\{0,1,2\}$. (i.e. Buffer: [2 2 2 2 2], Dummy Buffer: [2 2 2 2])									

Similarly, the best average cycle time for each set of failure and repair probabilities are tabulated in bold. As can be seen from Table 5.7, either the LD_M (look downstream at machine states) or the LD_BM (look downstream at both the buffer and machine states) rule performs the best in all the tested cases.

5.3.5.3 Varying Buffer Capacity

In this set of experiments, the machine availability is maintained at 0.909 (i.e. failure probability, $p_i = 0.02$, repair probability, $r_i = 0.2$) across the machines. Only the buffer capacity of B₁, B₂, B₅, and B₆ are varied between a value of two and three. The buffer capacity could not be increased any further due to the limitation of the size of state space in MATLAB. The buffer capacity of B₃, and B₄ are not set to three as an increase in one unit to the buffer capacity of either B₃ or B₄ will require an additional dummy buffer. The additional buffer will cause the state space to exceed the limits allowed in MATLAB. The results based on average cycle time are presented in Table 5.8.

Table 5.8: Average cycle time for varying buffer capacity

Buffers	Dummy Buffers	LD_BM	LD_B	LD_M	LU_BM	LU_B	LU_M	LQ_B	FIFO+
[2 2 2 2 2 2]	[2 2 2 2]	7.8222	7.8865	7.8214	7.9414	7.8876	7.9409	7.8874	7.8870
[3 2 2 2 2 2]	[2 2 2 2]	8.9323	8.9973	8.9315	9.0521	8.9954	9.0511	8.9982	8.9979
[2 3 2 2 2 2]	[2 2 2 2]	8.9323	8.9973	8.9315	9.0518	8.9954	9.0511	8.9982	8.9979
[2 2 2 2 3 2]	[2 2 2 2]	7.8209	7.8853	7.8223	7.9755	7.9190	7.9754	7.9169	7.9199
[2 2 2 2 2 3]	[2 2 2 2]	7.8209	7.8853	7.8220	7.9754	7.9186	7.9746	7.9161	7.9187
[3 3 2 2 2 2]	[2 2 2 2]	10.0424	10.1082	10.0416	10.1616	10.1095	10.1612	10.1091	10.1088
[2 2 2 2 3 3]	[2 2 2 2]	7.8203	7.8839	7.8231	8.0093	7.9497	8.0090	7.9447	7.9505
[3 2 2 2 3 2]	[2 2 2 2]	8.9294	8.9945	8.9308	9.0871	9.0283	9.0840	9.0264	9.0293
[2 3 2 2 2 3]	[2 2 2 2]	8.9294	8.9945	8.9305	9.0871	9.0282	9.0833	9.0255	9.0281
failure = [0.02 0.02 0.02 0.02 0.02]; repair = [0.2 0.2 0.2 0.2 0.2]									

Similarly, the best average cycle time for each set of failure and repair probabilities are tabulated in bold. As can be seen from Table 5.8, either the LD_M (look downstream at machine states) or the LD_BM (look downstream at both the buffer and machine states) rule performs the best in all the tested cases.

5.3.5.4 Discussion

The Markov process model can only consider dispatching rules that involves machine and buffer states. Hence, there is a limit to the rules that can be modeled. As such, it is highly unlikely to use Markov process model to predict the performances to guide the optimization of weights.

The general conclusion that can be drawn from the experimental results presented in Table 5.6 to Table 5.8 is that Look Downstream (LD) generally performs better than Look Upstream (LU) and Look into Queue (LQ) rules. One interesting observation noted is that Look Downstream at Machines' (LD_M) states and Look Downstream at both Buffers' and Machines' states (LD_BM) perform better (in terms of average cycle time) than Look Downstream at Buffers' (LD_B) states. This may suggest the possibility of a better performing rule (than existing rule such as the Work content In Next Queue rule) by incorporating machines' states into next lot selection.

The difference between each Look Around rule, however, is very small. The slight improvement is not sufficiently large enough to justify the design of a new rule that involves Look Downstream in the proposed decision policy. Instead, this research chooses to combine existing dispatching rules and focuses on where the major benefits are believed to lie, that is, on-line optimization of weights to achieve non-

dominated user preferred performances. Before the rules are combined using the weighted aggregation method, the criteria selection and analysis (see Section 5.4) is carried out to identify the rules to use.

5.4 CRITERIA SELECTION AND ANALYSIS

In most studies, the basic concept to consider multiple objectives is to combine different dispatching rules (or criteria) that have different strengths, such as the Shortest Processing Time (SPT) dispatching criterion with good cycle time performance, and Critical Ratio (CR) dispatching criterion with good on-time delivery performance, into a single rule (Dabbas et al., 2001).

Although it is theoretically possible to consider as many criteria (or rules) in a weighted aggregation manner, it is not practical for on-line applications that requires decision to be made with near-real-time responsiveness. Inclusion of many rules will increase the size of the search space (i.e. with one weight used for each rule, there is greater number of weights to be searched), thereby increasing the computational time required for the reactive optimization of weights. The rule of thumb is generally to select rules that have the best potential for offering a solution to the problem under consideration.

In this study, the performances of eight commonly used dispatching rules are evaluated and analyzed using a dynamic and stochastic serial (i.e. single lot processing) ion implanter machine with sequence-dependent setups (SDS). The main objective is to identify the criterion to be combined to achieve the best/preferred trade-off amongst the contradicting needs of cycle time and delivery accuracy.

In the process of doing so, this study will also highlight the importance of taking into account not only the criterion/rule's strength, but also its trade-off (i.e. degree at which a criterion optimizes a performance objective at the expense of other performance objectives) in the rule selection.

In addition, this section will also provide an insight of how commonly used setup conscious rules and conventional dispatching rules perform, before and after they are combined, with respect to multiple conflicting objectives. The measures related to the performance objectives considered in this study are namely (i) average cycle time (*avgCT*), (ii) standard deviation of cycle time (*sdCT*), (iii) average tardiness (*avgTARD*), and (iv) standard deviation of tardiness (*sdTARD*).

5.4.1 DISPATCHING CRITERIA

The dispatching criteria considered in this study are summarized in Table 5.9.

5.4.2 COMPUTATIONAL EXPERIMENTS

The assumptions considered are similar to those used in Section 3.5.1 except that sequence-dependent setups are considered. Machine breakdowns, however, are not considered, as it has no effect on the relative evaluation of the dispatching rules for a one-machine problem.

Three lot types (namely T1, T2, and T3) are considered and released to the machine in proportional amounts to promote greater probability of machine setups. Two sets of lot-type dependent expected average processing times $\bar{p}$ (in minutes) are considered: (i) Close processing times between lot types (i.e. T1: 45, T2: 50, T3:

55), and (ii) Far-apart processing times between lot types (i.e. T1: 35, T2: 50, T3: 65).

Table 5.9: Dispatching criteria/rules

Dispatching Criteria	Dispatching Rules	Description of Criteria/Rules
Minimize a_i	First-In, First-Out (FIFO)	Selects the lot with the earliest arrival time to the queue of the machine. FIFO is used in conjunction with other dispatching rules to break ties where applicable.
Minimize p_i	Shortest Processing Time (SPT)	Selects the lot with the shortest processing time.
Minimize $1/(RPT_i + EWT_i)^2$	Shortest Cycle Time (SCT)	Selects the lot with the least modified cycle time that is defined as $1/(RPT_i + EWT_i)^2$ (Sivakumar, 2001).
Minimize d_i	Earliest Due Date (EDD)	Selects the lot with the earliest due date.
Minimize $(d_i - t_{now})/RPT_i$	Critical Ratio (CR)	Selects the lot with the lowest critical ratio as defined by $(d_i - t_{now})/RPT_i$.
Minimize $(d_i - t_{now}) - RPT_i$	Least Slack (LS)	Selects the lot with the smallest slack. The slack is defined as $(d_i - t_{now}) - RPT_i$.
Minimize s_{ij}	Smallest SetUp (SSU)	Selects the lot that requires the smallest setup time.
Minimize $s_{ij}/n_i p_i$	Smallest SetUp Modified (SSU+)	Takes into account the relative cost of a particular setup by considering the potential runtime of available work in process that can make use of the setup on the machine (Sivakumar, 2001). Selects the lot that has the smallest SSU+ that is defined as $s_{ij}/n_i p_i$.
Notations/Abbreviations	Descriptions	
i	A particular lot among a set of lots waiting to be processed	
i'	A particular lot that has just completed its processing on the machine	
a_i	Arrival time of lot i to the queue of the machine	
p_i	Expected mean processing time of lot i	
d_i	Due date of lot i	
s_{ij}	Setup time between lot i' that has been processed and lot i considered for processing next	
RPT_i	Remaining mean processing time of lot i (For a single machine one operation system, $RPT_i = p_i$)	
EWT_i	Elapsed waiting time of lot i in the queue of the machine (i.e. $EWT_i = t_{now} - a_i$)	
t_{now}	Current time at which the rule is applied	
n_i	Number of lots in the queue that could use the same setup required by lot i	

The lot processing times in both sets are perturbed on two levels using a uniformly distributed interval of $U[-5,5]$ and $U[-15,15]$ minutes and labeled from P1 to P4 respectively in Table 5.10. The perturbation is included to account for machine and/or operator efficiencies, yield losses, and loading and unloading variability.

Table 5.10: Lot types, product mix, and processing times data

S/N	Variables/Parameters	Levels	Description
1	Number of lot types, T_j	-	3 (i.e. $T1$, $T2$, and $T3$)
2	Product Mix	-	1:1:1
3	Processing time, p_i [mins] (average processing time for each level = 50 mins)	$P1$ $P2$ $P3$ $P4$	$T1$: $45+U[-5,5]$, $T2$: $50+U[-5,5]$, $T3$: $55+U[-5,5]$; $T1$: $35+U[-5,5]$, $T2$: $50+U[-5,5]$, $T3$: $65+U[-5,5]$; $T1$: $45+U[-15,15]$, $T2$: $50+U[-15,15]$, $T3$: $55+U[-15,15]$; $T1$: $35+U[-15,15]$, $T2$: $50+U[-15,15]$, $T3$: $65+U[-15,15]$.

Lots arrive dynamically to the machine, with mean interarrival time of 53 minutes (based on the results of 10 simulation runs using the $P1$ processing time level to maintain an average WIP of six in a buffer with capacity of nine).

The proportion of average setup time to average processing time is set at 20%. The 20% level is comparable to the situation of a real job-shop (Kim and Bobrowski, 1997). Under this setting, sequence-dependent setup time of 10 minutes is applied when the type of the lot that has been processed is different from the type of the next lot to be processed. No setup is required to process consecutive lots of the same type.

The due date of lot i , d_i is determined by assigning a random flow allowance to each lot upon its arrival to generally represent the actual environment in which due dates are requested by the customer (Weng and Ren, 2006) and is defined as

$$d_i = a_i + (1 + u\delta) p_i, \quad (5.35)$$

where $u \in U(0,1)$, and δ is the due date tightness factor. In this study, loose and tight due date settings (denoted as *DL*, and *DT* respectively) are set parallel to those used in Kim and Bobrowski (1997). The δ value is adjusted based on 10 simulation runs to result in approximately 20% and 50% tardy lots for loose and tight due dates respectively when applying the FIFO dispatching rule at the *P1* processing time level. The δ values tabulated in Table 5.11 are applied across all the processing time levels.

Table 5.11: δ values for loose and tight due date setting corresponding to each setup level

Due Date Setting:	<i>Tight Due Date (DT)</i>	<i>Loose Due Date (DL)</i>
δ value	12	29

A model is constructed and simulated using AutoSchedTM Accelerated Processing (AutoSchedAP) v7.2 by Brooks Automation Inc. Some customizations of the dispatching rules used are necessary. The simulation models and rules are debugged and verified in a number of iterations using the ‘trace’ technique (Law, 2007).

The same seeds are used for the different dispatching rules to allow the rules to be compared in a more statistically efficient manner. The determination of a warm-up period is carried out using the Welch’s graphical procedure (Law, 2007) for each problem configuration. Performance measures are collected for a six-month period after statistical counters are reset (i.e. after warmup). Each problem configuration can be described using $\{P1/P2/P3/P4, DL/DT\}$, where the first and second component refers to the processing time level, and due date tightness level

respectively. 20 replications are randomly generated to obtain acceptable confidence in the output measures. Hence, eight problem configurations (i.e. 4 processing time levels $\times$ 2 due date tightness levels) with a total of 160 problem instances (i.e. 8 problem configurations $\times$ 20 replications) are carried out.

5.4.3 RESULTS AND DISCUSSION

5.4.3.1 Preliminary Evaluation and Analysis of Simulation Results

For brevity, the mean and 95% confidence interval of (i) average cycle time, (ii) standard deviation of cycle time, (iii) average tardiness, and (iv) standard deviation of tardiness for the eight dispatching rules under eight problem configurations are tabulated in Appendix E, and their means (based on 20 replications) are respectively plotted from Figure 5.6 to Figure 5.9. The 95% confidence intervals are omitted as they are too small (with respect to the mean values) to be clearly represented in the graphs.

Figure 5.6: Mean of average cycle time of rules under different problem configurations

Figure 5.7: Mean of standard deviation of cycle time of rules under different problem configurations

Figure 5.8: Mean of average tardiness of rules under different problem configurations

Figure 5.9: Mean of standard deviation of tardiness of rules under different problem configurations

SSU and SSU+ clearly perform better than the other dispatching rules in terms of average cycle time ($avgCT$) across all the problem configurations (see Figure 5.6). This is expected since setup times are dependent on the sequence and setup conscious rules capable of reducing the total setup time would result in a lower $avgCT$ performance. The other six rules result in similar $avgCT$ performance. In the presence of sequence-dependent setup (SDS), Shortest Processing Time (SPT) dispatching criterion that is known for its good cycle time performance no longer dominates. The performance of the rules in terms of $avgCT$ is not affected by variation in processing times and due date tightness of the system since each dispatching rules maintain almost the same average cycle time across all the problem configurations.

Figure 5.7 shows that FIFO and SCT are the better performing rules in terms of standard deviation of cycle time ($sdCT$) across eight problem configurations. This is because both rules attempt to equalize the waiting times of the lots (i.e. to minimize the variance of the waiting times), thereby, minimizing the standard deviation of cycle time. The performances of FIFO, SCT, and SSU rules in $sdCT$ seem to be much less susceptible to variations (in processing times and due date tightness) of the system as compared to due date related (i.e. EDD, CR, and LS) rules. For example, due date related rules have a much poorer performance in $sdCT$ under loose due date tightness conditions than tight due date tightness conditions.

In addition, the $sdCT$ performance of EDD, CR, and LS (see Figure 5.7) shows that the perturbations (of $U[-5,5]$, or $U[-15,15]$) on the processing times do not have much effect on $sdCT$ as compared to the closeness of the processing times between lot types. Hence, the performance for $P1$ and $P3$ processing times levels are usually

close. The same can be said for the $P2$ and $P4$ processing time levels. This applies across all the performance measures considered (i.e. $avgCT$, $sdCT$, $avgTARD$, and $sdTARD$).

The EDD, CR, and LS rules, as expected, are capable of minimizing $avgTARD$ and $sdTARD$ performance measures as can be seen from Figure 5.8 and Figure 5.9 respectively. In fact, it is later shown via correlation analysis (see Section 5.5) that both the $avgTARD$ and $sdTARD$ objectives are positively correlated and one may be dropped from the study.

5.4.3.2 Comparison of Rules using Nonparametric Tests

A nonparametric Kruskal-Wallis analysis of variance (ANOVA) on ranks test performed for all the problem configurations show that the differences between the treatment groups (i.e. dispatching rules) are statistically significant (i.e. P value ≤ 0.001 ; For more information on P value, see Appendix D). Pairwise multiple comparisons of the dispatching rules in each performance measure are then carried out using Tukey Test. Table 5.12 tabulates the best group of ‘comparable’ dispatching rules in which there is no statistical significance between/among them (i.e. P -values < 0.05).

As can be observed from Table 5.12, there is generally no statistically significant difference between the rank sums of (i) SSU and SSU+ that enclose the $avgCT$ comparison, (ii) FIFO, SCT, and SSU that enclose the $sdCT$ comparison, and (iii) EDD, CR, and LS that enclose both the $avgTARD$ and $sdTARD$ comparisons.

Table 5.12: Comparable rules based on Tukey Test results

Performance Measures	Comparable rules for each problem configuration denoted as $\{PTL, DTL\}^*$							
	$\{P1,DL\}$	$\{P2,DL\}$	$\{P3,DL\}$	$\{P4,DL\}$	$\{P1,DT\}$	$\{P2,DT\}$	$\{P3,DT\}$	$\{P4,DT\}$
<i>avgCT</i>	SSU, SSU+	SSU, SSU+	SSU, SSU+	SSU, SSU+	SSU, SSU+	SSU, SSU+	SSU, SSU+, SPT	SSU, SSU+
<i>sdCT</i>	FIFO, SCT, SSU	FIFO, SCT, SSU	FIFO, SCT, SSU	FIFO, SCT, SSU	FIFO, SCT, SSU	FIFO, SCT, SSU	FIFO, SCT, SSU	FIFO, SCT, SSU
<i>avgTARD</i>	CR, EDD, LS	CR, EDD, LS, SSU	CR, EDD, LS	CR, EDD, LS	CR, EDD, LS	CR, EDD, LS	CR, EDD, LS	CR, EDD, LS, SSU
<i>sdTARD</i>	CR, EDD, LS	CR, EDD, LS	CR, EDD, LS	CR, EDD, LS	CR, EDD, LS	CR, EDD, LS	CR, EDD, LS	CR, EDD, LS
<p>* <i>PTL</i>: Processing Times Level; <i>DTL</i>: Due date Tightness Level <i>avgCT</i> – Average Cycle Time; <i>sdCT</i> – Standard Deviation of Cycle Time; <i>avgTARD</i> – Average Tardiness; <i>sdTARD</i> – Standard Deviation of Tardiness</p>								

5.4.3.3 Selection of Candidate Scheduling Criteria (Dispatching Rules)

Table 5.13 tabulates the number of problem instances in which the dispatching rule gives the lowest value in the stated performance measure (to be minimized). ‘Rule (α/β)’ in Table 5.13 is interpreted as ‘Rule performed the best in α out of β problem instances’. There are 160 problem instances (i.e. 4 processing time levels $\times$ 2 due date levels $\times$ 20 replications).

With regards to the selection of candidate rules/criteria for combination in a weighted aggregation manner, some studies justify the selection of certain rules based on logical reasoning and/or prior studies’ results (e.g. Dabbas, 1999, Park et al., 2001, Lin et al., 2005), while others pick the best rule from among the comparable rules (i.e. not statistically different with respect to a particular criterion/objective) (e.g. Gupta, 2006). Since there is no statistically significant difference between the rank sums of SSU and SSU+ that enclose the average cycle time (*avgCT*) comparison (see Section 5.4.3.2), the number of problem instances in which SSU+ outperforms SSU as shown in Table 5.13 suggests that SSU+ should be selected over the SSU rule.

Table 5.13: Rules that gives the best performance in each of the performance objective

Performance Measures	Number of times the stated rule performs best out of a number of problem instances								
	{P1,DL}	{P2,DL}	{P3,DL}	{P4,DL}	{P1,DT}	{P2,DT}	{P3,DT}	{P4,DT}	Total
<i>avgCT</i>	SSU+ (16/20) SSU (4/20)	SSU+ (17/20) SSU (3/20)	SSU+ (19/20) SSU (1/20)	SSU+ (16/20) SSU (4/20)	SSU+ (19/20) SSU (1/20)	SSU+ (14/20) SSU (6/20)	SSU+ (17/20) SSU (3/20)	SSU+ (12/20) SSU (8/20)	SSU+ (130/160) SSU (30/160)
<i>sdCT</i>	FIFO (13/20) SCT (7/20)	FIFO (18/20) SCT (2/20)	FIFO (9/20) SCT (11/20)	FIFO (19/20) SCT (1/20)	FIFO (14/20) SCT (6/20)	FIFO (15/20) SCT (5/20)	FIFO (12/20) SCT (8/20)	FIFO (20/20) SCT (0/20)	FIFO (120/160) SCT (40/160)
<i>avgTARD</i>	EDD (8/20) CR (6/20) LS (6/20)	EDD (9/20) CR (5/20) LS (6/20)	EDD (7/20) CR (9/20) LS (4/20)	EDD (4/20) CR (7/20) LS (9/20)	EDD (5/20) CR (11/20) LS (4/20)	EDD (11/20) CR (6/20) LS (3/20)	EDD (6/20) CR (8/20) LS (6/20)	EDD (9/20) CR (2/20) LS (9/20)	EDD (59/160) CR (54/160) LS (47/160)
<i>sdTARD</i>	EDD (8/20) CR (7/20) LS (5/20)	EDD (8/20) CR (7/20) LS (5/20)	EDD (8/20) CR (8/20) LS (4/20)	EDD (6/20) CR (8/20) LS (6/20)	EDD (5/20) CR (10/20) LS (5/20)	EDD (2/20) CR (17/20) LS (1/20)	EDD (3/20) CR (11/20) LS (6/20)	EDD (1/20) CR (17/20) LS (2/20)	EDD (41/160) CR (85/160) LS (34/160)

However, the impact of SSU and SSU+ on other performance measures should also be taken into consideration apart from its ability to minimize *avgCT*. For example, the SSU rule results in a smaller standard deviation of cycle time (*sdCT*), average tardiness (*avgTARD*), and standard deviation of tardiness (*sdTARD*) than SSU+ in all the problem configurations. In fact, SSU also falls into the best group of comparable rules (see Table 5.12) with FIFO and SCT for the *sdCT* performance measures.

The *t*-tests (for data that meets the normality and equal variance requirements) and Mann-Whitney Rank Sum tests reveal that there is statistically significant difference in the mean values of *sdCT* and *sdTARD* between SSU and SSU+ at 95% confidence level. No significant differences were observed for other rules. Hence, FIFO is chosen for minimizing *sdCT* while CR is chosen for minimizing *avgTARD* and *sdTARD* in tandem.

In order to understand the difference in the resulting performances (measured using $avgCT$, $sdCT$, $avgTARD$, and $sdTARD$) between aggregating SSU or SSU+ with FIFO and CR to address multiple objectives (MO), the performance of the weighted rule made up of (i) FIFO, CR, SSU+, and (ii) FIFO, CR, SSU (hereafter known as the MOFCS+ and MOFCS rule respectively) will be simulated and evaluated in the next section.

5.4.3.4 Evaluation of Candidate Scheduling Criteria (Dispatching Rules)

With the purpose of pointing out the difference between selecting SSU+ or SSU as the criterion for minimizing $avgCT$ and to understand its causative influence over the three other objectives (i.e. $sdCT$, $avgTARD$, and $sdTARD$) it is not designed to minimize, the MOFCS and MOFCS+ will be compared using one processing time level (i.e. PI level) and two due date tightness levels (i.e. DL , DT) (see Section 5.4.2). Only one processing time level is considered as the various processing time levels has less effect on the four performance measures as compared to the two due date tightness levels.

Each of the criteria (presented in the form of dispatching rules) is normalized, weighted and combined into a single criterion. The individual rule index, $I_{R,i}$ is computed for each lot i in the queue at the time when machine is available to select the next lot, where R refers to the rules/criterion used, $R = \{SSU+ \text{ or } SSU, FIFO, CR\}$. The minimum and maximum index for each rule is denoted as $I_{R,min}$ and $I_{R,max}$ respectively. The weighted sum of normalized index, I_i for each lot i is computed as follows:

$$I_i = \sum_{R=1}^3 \left[w_R \frac{I_{R,\min} - I_{R,i}}{I_{R,\min} - I_{R,\max}} \right], \quad \sum_{R=1}^3 w_R = 1 \quad (5.36)$$

where weight w_R reflect the relative importance of the criterion associated with the rule. The lot with the lowest weighted sum of normalized index is selected for processing next. Table 5.14 lists the 19 sets of weights tested over the $\{PI, DL\}$ and $\{PI, DT\}$ problem configurations. Each set of weights is replicated 20 times.

Table 5.14: Weights used for the MOFCS and MOFCS+ rule

ID of Weights	w_{FIFO}	w_{CR}	w_{SSU+} / w_{SSU}	Remarks
1	0.000	0.333	0.667	A mix of CR and SSU/SSU+ rules
2	0.000	0.500	0.500	A mix of CR and SSU/SSU+ rules
3	0.000	0.667	0.333	A mix of CR and SSU/SSU+ rules
4	0.167	0.417	0.417	Involve all three rules
5	0.167	0.667	0.167	Involve all three rules
6	0.167	0.167	0.667	Involve all three rules
7	0.333	0.333	0.333	Involve all three rules
8	0.333	0.667	0.000	A mix of FIFO and CR rules
9	0.333	0.000	0.667	A mix of FIFO and SSU/SSU+ rules
10	0.417	0.167	0.417	Involve all three rules
11	0.417	0.417	0.167	Involve all three rules
12	0.500	0.000	0.500	A mix of FIFO and SSU/SSU+ rules
13	0.500	0.500	0.000	A mix of FIFO and CR rules
14	0.667	0.167	0.167	Involve all three rules
15	0.667	0.000	0.333	A mix of FIFO and SSU/SSU+ rules
16	0.667	0.333	0.000	A mix of FIFO and CR rules
17	1.000	0.000	0.000	Equivalent to FIFO rule
18	0.000	1.000	0.000	Equivalent to CR rule
19	0.000	0.000	1.000	Equivalent to SSU/SSU+ rule

In a way, the concept of dominance (see Section 5.1) allows the comparison of the MOFCS and MOFCS+ rules in relation to multiple performance measures (i.e. $avgCT$, $sdCT$, $avgTARD$, and $sdTARD$). The mean and 95% confidence interval for the multiple performance measures are documented in Appendix F. Table 5.15

summarizes the dominated and non-dominated solutions of the MOFCS+ and MOFCS rule for the $\{P1,DL\}$ and $\{P1,DT\}$ problem configurations.

Table 5.15: Dominated and non-dominated solutions of the MOFCS+ and MOFCS rule for the $\{P1,DL\}$ and $\{P1,DT\}$ problem configurations

ID of Weights	Weights			{P1,DL} problem configuration		{P1,DT} problem configuration	
	w_{FIFO}	w_{CR}	w_{SSU+}	(MOFCS+)	[MOFCS]	(MOFCS+)	[MOFCS]
1	0.000	0.333	0.667	✓	✓	✓	✓
2	0.000	0.500	0.500	✗ (4)	✓	✓	✓
3	0.000	0.667	0.333	✓	✓	✓	✓
4	0.167	0.417	0.417	✓	✓	✓	✓
5	0.167	0.667	0.167	✓	✓	✓	✓
6	0.167	0.167	0.667	✓	✓	✓	✓
7	0.333	0.333	0.333	✓	✓	✓	✓
8	0.333	0.667	0.000	✓	✓	✓	✓
9	0.333	0.000	0.667	✓	✗ [19]	✓	✓
10	0.417	0.167	0.417	✓	✓	✓	✓
11	0.417	0.417	0.167	✓	✓	✓	✓
12	0.500	0.000	0.500	✓	✓	✓	✓
13	0.500	0.500	0.000	✗ (11)	✗ [11]	✗ (11)	✗ [11]
14	0.667	0.167	0.167	✓	✓	✓	✓
15	0.667	0.000	0.333	✓	✓	✓	✓
16	0.667	0.333	0.000	✓	✓	✓	✗ [14]
17	1.000	0.000	0.000	✗ (14,15)	✗ [14,15]	✗ (14)	✗ [14,15]
18	0.000	1.000	0.000	✓	✓	✗ (5)	✗ [3,5]
19	0.000	0.000	1.000	✓	✓	✓	✗ [9]
Remarks ✗ (#...) - Denotes the solution is dominated by one or more solutions (ID enclosed in round brackets) from the MOFCS+ rule. ✗ [#...] - Denotes the solution is dominated by one or more solutions [ID enclosed in square brackets] from the MOFCS rule. ✓ - Denotes the solution is non-dominated by any solutions it is compared against.							

Every single solution of the MOFCS+ rule is compared with 18 other solutions within the same problem configuration. The same also applies for the MOFCS rule. A tick '✓' indicates that the particular solution is non-dominated. A cross '✗', however, indicates that the solution is dominated by at least one solution within the same problem configuration. The number(s) enclosed by the round or square

brackets denotes the ID(s) of the solution(s) that dominates the particular solution for the MOFCS+ and MOFCS rule respectively. For example, ‘✖ (4)’ in Table 5.15 means that the solution (i.e. $avgCT$, $sdCT$, $avgTARD$, and $sdTARD$) obtained using weight set 2 (i.e. $\{w_{FIFO} = 0, w_{CR} = 0.5, w_{SSU+} = 0.5\}$) is dominated by the solution obtained using weight set 4.

It should be noted that the purpose of Table 5.15 is not for the comparison of the MOFCS+ and MOFCS rules. Rather, some of the observations noted from Table 5.15 are important and have several implications as discussed below.

A theorem is reproduced from Miettinen (1999):

Theorem: *The solution to the problem*

$$\left. \begin{array}{ll} \text{Minimize} & F(\mathbf{x}) = \sum_{m=1}^M w_m f_m(\mathbf{x}) \\ \text{subject to} & g_j(\mathbf{x}) \geq 0, \quad j = 1, 2, \dots, J; \\ & h_k(\mathbf{x}) = 0, \quad k = 1, 2, \dots, K; \\ & x_i^{(L)} \leq x_i \leq x_i^{(U)}, \quad i = 1, 2, \dots, n. \end{array} \right\} \quad (5.37)$$

is Pareto-optimal if the weight is positive for all objectives.

This theorem is true for any multiobjective optimization problems (for the proof, refer to Miettinen (1999)). For example, in our problem, solutions corresponding to weight sets 4 to 7, 10 to 11, and 14 have a positive weight (greater than zero) for each objective and have also been observed to be Pareto-optimal. Although having positive weights for all objectives will ensure Pareto optimal solutions, it does not mean that Pareto optimal solutions cannot be obtained when one or more weights are non-positive (i.e. equal to zero). For example, solutions corresponding to weight sets 1 to 3, 8, 12, 15, 16, 18, and 19 for MOFCS rule in $\{PI, DL\}$ problem

configuration are also Pareto optimal among the tested weight sets despite not having positive weights for all the objectives.

Hence, one purpose of Table 5.15 is to highlight that there are many Pareto optimal solutions without positive weights for each objective. Since some set of weights will result in dominated solutions whether MOFCS+ or MOFCS is selected for use, it is therefore imperative for the NEXTS decision methodology to consider this so that dominated solutions will not be generated. This is achieved using the desirability function that will be further discussed in Section 6.1.1.

In addition, the individual rules (FIFO, CR, SSU/SSU+) may not make up the extreme points on the respective axis for each objective since one or more solutions (see solutions corresponding to weight sets 17 to 19) are dominated as can be seen from Table 5.15. These observations indicate that the heuristics FIFO, CR, SSU/SSU+ rules are only approximate minimizers for $sdCT$, $avgTARD$ or $sdTARD$, and $avgCT$ respectively.

The 19 solutions of MOFCS+ rule is then compared against the 19 solutions of the MOFCS rule across two different problem configurations. The results of the comparison based on the concept of dominance are summarized in Table 5.16. A tick ‘✓’ indicates that the particular solution is non-dominated by the other 37 solutions (i.e. remaining 18 solutions of the same rule + 19 other solutions of the other rule) within the same problem configuration. A cross ‘✗’, however, indicates that the solution is dominated by at least one solution within the same problem configuration. For example, solution corresponding to weight set 17 of the MOFCS rule in the $\{PI, DL\}$ problem configuration is dominated by solutions corresponding

to weight sets 14, 15, and 17 of the MOFCS+ rule and solutions corresponding to weight sets 14 and 15 of the MOFCS rule within the same problem configuration.

Table 5.16: Comparing MOFCS+ and MOFCS rules for the $\{P1,DL\}$ and $\{P1,DT\}$ problem configurations using the concept of domination across four objectives ($avgCT$, $sdCT$, $avgTARD$, and $sdTARD$)

ID of Weights	Weights			$\{P1,DL\}$ problem configuration		$\{P1,DT\}$ problem configuration	
	w_{FIFO}	w_{CR}	w_{SSU+}	(MOFCS+)	[MOFCS]	(MOFCS+)	[MOFCS]
1	0.000	0.333	0.667	× [1]	✓	× [1]	✓
2	0.000	0.500	0.500	× (4)	✓	× [4]	✓
3	0.000	0.667	0.333	✓	✓	× [3]	✓
4	0.167	0.417	0.417	✓	✓	× [4]	✓
5	0.167	0.667	0.167	✓	✓	× [5]	✓
6	0.167	0.167	0.667	× [6]	✓	× [6]	✓
7	0.333	0.333	0.333	✓	✓	× [7]	✓
8	0.333	0.667	0.000	✓	✓	✓	✓
9	0.333	0.000	0.667	× [6,9,19]	× [19]	× [6,9,19]	✓
10	0.417	0.167	0.417	× [10]	✓	× [7,10]	✓
11	0.417	0.417	0.167	× [11]	✓	× [11]	✓
12	0.500	0.000	0.500	× [10]	✓	× [10]	✓
13	0.500	0.500	0.000	× (11) & [11]	× (11) & [11]	× (11) & [11]	× (11) & [11]
14	0.667	0.167	0.167	× [14]	✓	× [14]	✓
15	0.667	0.000	0.333	× [15]	✓	× [15]	✓
16	0.667	0.333	0.000	✓	✓	× [14]	× [14]
17	1.000	0.000	0.000	× (14,15) & [14]	× (14,15,17) & [14,15]	× (14) & [14,15]	× (14) & [14,15]
18	0.000	1.000	0.000	✓	✓	× (5) & [3,5]	× (5) & [3,5]
19	0.000	0.000	1.000	✓	✓	× [1,6,9]	× [9]
Remarks × (#...) - Denotes the solution is dominated by one or more solutions (ID enclosed in round brackets) from the MOFCS+ rule. × [#...] - Denotes the solution is dominated by one or more solutions [ID enclosed in square brackets] from the MOFCS rule. ✓ - Denotes the solution is non-dominated by any solutions it is compared against.							

It can be observed from Table 5.16 that the MOFCS+ rule for the $\{P1,DL\}$ problem configuration contains only eight non-dominated solutions out of a total of 19 solutions. On the contrary, the MOFCS rule performed better with only three dominated solutions. For the $\{P1,DT\}$ problem configuration, there is only one

non-dominated solutions for the MOFCS+ rule while there are 14 non-dominated solutions for the MOFCS rule.

The comparison of MOFCS+ and MOFCS rules for the $\{PI, DL\}$ and $\{PI, DT\}$ problem configurations using the concept of domination across only three objectives (i.e. $avgCT$, $sdCT$, and $avgTARD$) shows similar outcome (see Table F.5 in Appendix F). This result is presented because, as it will be shown in Section 5.5, the number of objectives considered could be reduced from four to three.

The fact that there is a statistically significant difference in the mean values of $sdCT$ and $sdTARD$ between SSU and SSU+ rule at 95% confidence level has provided a good indication that the SSU rule will prevail over the SSU+ rule. The MOFCS rule has a larger solution space for the decision maker to choose from since it constitutes a larger set of non-dominated solutions for both tested problem configurations. It also suggests that the results presented in Gupta and Sivakumar (2005a, 2005b) and Sivakumar and Gupta (2006) could be further improved by switching the SSU+ rule to SSU rule.

5.5 CORRELATION ANALYSIS OF OBJECTIVES

A review paper by Hart et al. (2005) on multiobjective evolutionary scheduling reported that it is generally quite unclear about the extent of correlation between/among different objectives occurring in scheduling problems that will render multiobjective approaches unnecessary. The only work involving a thorough correlation analysis of fourteen different scheduling-related objectives for flowshop problems reviewed in Hart et al. (2005) is by Hapke et al. (2002).

The work by Hapke et al. (2002) is aimed at understanding in what situations multiobjective approaches are really required. For example, if uncorrelated makespan and maximum lateness are both important objectives in a particular case, then the problem is best treated as a proper multiobjective problem to deal with the tradeoff between these two objectives. Conversely, if the two objectives of interest are highly, positively correlated (both tend to increase/decrease together), there are likely to be few or even no tradeoff solutions, then a representative objective can be used (the other highly, positively correlated objective may be dropped) to reduce the dimensionality of the problem (Padmanabhan, 2005; Goel et al., 2007).

The correlation coefficients and P values, computed with the normalized objective values from all the simulation runs described in Section 5.4.2 using the Pearson Product Moment Correlation, are presented in Table 5.17. The correlation coefficient quantifies the strength of the association between the objectives while the P value is the probability of being wrong in concluding that there is a true association between the objectives. The smaller the P value, the greater the probability that the objectives are correlated.

Table 5.17: Correlation coefficient and P-value (in bracket) for each pair of objectives

	sdCT	avgTARD	sdTARD
avgCT	0.141 (3.57E-12)	-0.0784 (0.00012)	-0.432 (1.87E-109)
sdCT	-	-0.627 (3.71E-262)	-0.699 (0)
avgTARD	-	-	0.872 (0)

From Table 5.17, it can be observed that the average cycle time (*avgCT*) objective is weakly correlated with the standard deviation of cycle time (*sdCT*), and the

average tardiness (*avgTARD*) objectives since the correlation coefficients are small. The *avgCT* and *sdTARD* (standard deviation of tardiness) pair, the *sdCT* and *avgTARD* pair, and the *sdCT* and *sdTARD* pair are almost negatively correlated (i.e. conflicting as one objective tends to improve at the expense of the other).

On the other hand, there appears to be strong positive correlation between the pair of *avgTARD* and *sdTARD* objectives. Positively correlated objectives improve or deteriorate together as can be observed in Figure 5.10 and Figure 5.11 for the respective $\{PI, DL\}$ and $\{PI, DT\}$ problem configurations. The correlation plots for other pair of objectives can be found in Appendix F (see Figure F.1 to Figure F.10). There is, therefore, a choice to drop either the *avgTARD* or *sdTARD* objective from the study. In our research, the *sdTARD* is dropped as it is easier for the decision maker to comprehend the *avgTARD* objective and specify the target *avgTARD* to achieve.

Figure 5.10: Strong positive correlation between the average tardiness and standard deviation of tardiness objectives for $\{PI, DL\}$ problem configuration

Figure 5.11: Strong positive correlation between the average tardiness and standard deviation of tardiness objectives for $\{PI, DT\}$ problem configuration

5.6 SUMMARY

This chapter describes the Pareto optimality and dominance concept that is applied in multiobjective optimization problems. The selection of criteria (or dispatching rules) is performed to reduce the search space for the weights and in the process of the selection, the results also highlight the importance to base the criterion/rule selection not only on the criterion/rule's strength in optimizing an objective but also the trade-offs the criterion has with regards to the other objectives.

If the selection of criteria (rules) is based only on its ability to optimize the objective of interest (a common practice), the SSU+ rule could have been selected. However, the experiments showed that SSU, though not the best rules for the tested problem configurations, achieved its aim of minimizing $avgCT$ with less cost in terms of $sdCT$, $avgTARD$, and $sdTARD$ performance measures. The comparison of the MOFCS and MOFCS+ rules using the concept of dominance shows that MOFCS has fewer dominated solutions and indeed performs better than MOFCS+

in the face of multi-objective optimization. The MOFCS is applied in the NEXTS Decision Policy (see the next chapter).

A correlation analysis of the objectives shows that one of the tardiness-related (either average tardiness or standard deviation of tardiness) is redundant and can be dropped. In addition, the weakly correlated and negatively correlated objectives suggest the presence of conflicting solutions in which tradeoffs must be made.

As brought up earlier in Section 5.2, the research issues, after having decided on the use of the weighted aggregation method, have to do with (i) which rules to include for optimizing multiple objectives of interest, (ii) how user preferences should be incorporated into decision-making, and (iii) how the weights used in the weighted aggregation of rules can be optimized. Only issue (i) has been addressed so far. Issues (ii) and (iii) will be discussed in the next chapter.

CHAPTER 6

PROPOSED NEXTS DECISION POLICY

This chapter begins by addressing how user preferences can be incorporated into decision-making via the use of desirability functions (see Section 6.1). The proposed concept for the reactive optimization of weights is discussed in Section 6.2. The proposed NEXTS decision policy is made up of the MOFCS rule that is used for the next lot selection and the Look Ahead Simulation-based Genetic Algorithm with Desirability function (LASGAD) used to optimize the weights used in the MOFCS rule (see Section 6.3). In Section 6.4, LASGAD is compared against the commonly used Response Surface Methodology with Desirability function.

In the study of the NEXTS decision policy, the proposed NEXTS time policy is not considered to avoid being bogged down with too much details at the initial stage in which new methods are being proposed and evaluated. The most common dispatching time policy is applied. This implies that the next lot selection is performed when the machine is available.

6.1 INCORPORATING USER PREFERENCES INTO DECISION-MAKING

The user preference refers to the decision maker's preferred operating point/region on the Pareto surface. The use of preferences is evident in multiple criteria decision making (Cvetković and Parmee, 2002). Preferences can be expressed *a priori*, *a*

posteriori, or in an interactive way (Horn, 1997; Coello Coello, 2000). Preferences expressed *a priori* imply that the decision maker defines his/her preferences in advance (before performing the search). When preferences are expressed *a posteriori*, it means that the search is performed first to (typically) generate more than one non-dominated solutions, from which the decision maker will specify his/her preference by picking one solution to apply. In the interactive case, preferences can change and a set of pre-specified preferences is used to guide the search until new preferences are supplied.

The desirability function approach is one of the most widely accepted and used methods in industry for dealing with the optimization of multiple criteria problems (Fuller and Scherer, 1998; Dabbas et al., 2003; Tyan et al., 2004; Lin et al., 2005; Pasandideh and Niaki, 2006). Evidence of this acceptance is demonstrated by the inclusion of desirability function optimization in one of the most popular commercially available software packages for experimental design, Design Expert® by Stat-Ease, Inc. The desirability function for criterion with a target can be used to allow user specified preferences to be set as the targets (see Section 6.1.3). Steuer (1999) provides a proof that the desirability approach can facilitate the selection of a Pareto-optimal solution out of the Pareto set.

6.1.1 DESIRABILITY FUNCTIONS

The concept of desirability was first introduced by Harrington Jr. (1965) and the flexibility of the desirability functions further improved by Derringer and Suich (1980). The multiple performance criteria (i.e. average cycle time, standard deviation of cycle time, and average tardiness) are each expressed as a desirability

function and combined into a single objective function. Depending on whether a particular criterion x_i is to be maximized, minimized, or assigned a target value, different desirability functions introduced by Derringer and Suich (1980) have been commonly applied to transform each criterion x_i into a desirability value $d(x_i)$, where $0 \leq d(x_i) \leq 1$. The total (or overall) desirability, D , is defined as the geometric mean of the individual desirability value:

$$D = (d(x_1) \times d(x_2) \times \dots \times d(x_n))^{1/n}, \quad (6.1)$$

where n denotes the number of the performance criteria. Total desirability D reflects the desirable ranges for each criterion $d(x_i)$. The desirable ranges are from zero (least desirable) to one (most desirable). Hence, the objective is to maximize D . The rationale behind using the geometric mean is that if any criterion has an undesirable desirability value (i.e. $d(x_i)$) at some shop floor operating conditions, then the total desirability D is zero regardless of the values taken on by other criteria.

6.1.2 DESIRABILITY FUNCTION FOR A CRITERION TO BE MINIMIZED

The desirability function for a criterion to be minimized (Derringer and Suich, 1980) is given by

$$d(x_i) = \begin{cases} 1, & x_i \leq L(x_i), \\ \left[\frac{U(x_i) - x_i}{U(x_i) - L(x_i)} \right]^\omega, & L(x_i) < x_i < U(x_i), \\ 0, & x_i \geq U(x_i). \end{cases} \quad (6.2)$$

where $L(x_i)$, and $U(x_i)$ are the lower and upper value of the i^{th} criterion respectively such that $L(x_i) \leq x_i \leq U(x_i)$, and the exponent ω determines the degree of acceptance of a value above the minimum. The desirability function $d(x_i)$ for criterion x_i to be

minimized, with respect to different exponent ω value, is plotted in Figure 6.1. The desirability function shown in Equation (6.2) is used when the decision maker does not specify any preferences and intends to minimize all performance criteria simultaneously.

Figure 6.1: Desirability function $d(x_i)$ for criterion x_i to be minimized

6.1.3 DESIRABILITY FUNCTION FOR CRITERION WITH TARGET

The commonly used desirability function for criterion with a target (Derringer and Suich, 1980) is given by

$$d(x_i) = \begin{cases} 0, & x_i \leq L(x_i), \\ \left[\frac{x_i - L(x_i)}{T(x_i) - L(x_i)} \right]^{\omega_1}, & L(x_i) < x_i \leq T(x_i), \\ \left[\frac{U(x_i) - x_i}{U(x_i) - T(x_i)} \right]^{\omega_2}, & T(x_i) < x_i < U(x_i), \\ 0, & x_i \geq U(x_i), \end{cases} \quad (6.3)$$

where $L(x_i)$, $U(x_i)$, and $T(x_i)$ are the lower, upper, and target value of the i^{th} criterion respectively such that $L(x_i) \leq x_i \leq U(x_i)$, and the exponents ω_1 and ω_2 determine how strict the target value is desired. The desirability function $d(x_i)$ for criterion x_i

with a target $T(x_i)$, with respect to different exponents ω_1 and ω_2 values, is plotted in Figure 6.2. When the decision maker has a preferred operating point, the desirability function shown in Equation (6.3) is used and the preferences can be set as the target value to meet. In this way, user preferences can be considered.

Figure 6.2: Desirability function $d(x_i)$ for criterion x_i with a specified target $T(x_i)$

6.2 REACTIVE WEIGHTS OPTIMIZATION

In the previous chapter, the First-In, First-Out (FIFO), Critical Ratio (CR), and Same SetUp (SSU) rules are selected for combination using a weighted aggregation method. This weighted MOFCS rule is applied in the NEXTS Decision Policy for the next lot selection. Due to the dynamism of the environment in which the MOFCS rule is applied, the weights that provide optimal performance at a given point in time will not continue to do so eternally.

The optimization of weights for the MOFCS rule can be performed periodically (time-driven), upon the occurrence of events (event-driven), or under a hybrid of both depending on the dynamism of the factory and how responsive it must be to

the state of the system. Apart from the change of dispatching rules, performance objectives, and user preferences that necessitate the re-optimization of weights, it is not known to the best of our knowledge what degree of disruptions and factory changes, or what kind of disruptions or unexpected events would invalidate the weights. A simple way to overcome this difficulty is to continuously search for better weights. Figure 6.3 shows the conceptual application of the reactive optimization of weights.

Figure 6.3: Conceptual application of the reactive weights optimization

In order to determine the weights for the aggregated rule, existing related studies (Deb, 2001; Lin et al., 2005) typically use the response surface methodology (RSM) (Myers and Montgomery, 1995) and desirability function approach (see Section 6.1.1). RSM is used to find the approximate relationship (i.e. fit a meta-model) between each response and independent variables so as to transform a complex model into a simplified (usually low-order) polynomial model. Each response's polynomial model is then transformed into a desirability function, and subsequently combined using the total desirability function to convert a multi-objective problem into a single objective optimization problem.

The fitting of meta-model for each response is typically carried out off-line by the decision maker(s) (Deb, 2001; Lin et al., 2005). The solution quality is therefore dependent on good approximation models for the responses. RSM usually works well when one response is considered, but is typically not considered in complex cases that involve non-polynomial and higher-order or multi-modal functions (Pasandideh and Niaki, 2006).

Different search algorithms (Hooke and Jeeves, 1962; Goldberg, 1989; Deb, 2001; Michalewicz and Fogel, 2004) ranging from classical optimization algorithms (e.g. exhaustive search, hill climbing methods, pattern search, branch and bound, etc.), to modern search algorithms (e.g. simulated annealing, tabu search), and evolutionary algorithms (i.e. genetic algorithm) have been researched and applied to solve various problems. Enumerating all possible solutions to determine the desired or optimal weights is impractical in any real-world problems because the number of alternative solutions is prohibitively large. Hence, there is a need to apply heuristics for search. Classical optimization algorithms are generally unable to handle analytically intractable complex problems (e.g. stochastic scheduling with multiobjectives and sequence-dependent setup) that may involve non-polynomial and higher-order or multi-modal functions.

Simulated annealing, tabu search, and genetic algorithm have the capability to avoid converging to the local optima and generally speaking, are potential techniques for determining the desired or optimal weights of a weighted rule. However, simulated annealing and tabu search rely on a single solution as the basis for future exploration (iteration). Hence, unexpected occurrence of events (especially the major ones that may affect the final search results) cannot be

accounted for without restarting the search. This prohibits the implementation of time-adaptive solutions that may be necessary for on-line, near-real-time applications. Evolutionary algorithms are inherently adaptive, and individuals in the population are able to adapt to the current environment (Michalewicz and Fogel, 2004). As such, evolutionary techniques do not require restarting every time something changes. In addition, evolutionary algorithms are suitable for parallel implementations as they are explicitly parallel. This can help to reduce the runtime.

Genetic algorithm (GA – for more details, refer to Section 6.3.2) have also been extensively used as search and optimization tools in various problem domains due to their broad applicability, ease of use, and global perspective (Goldberg, 1989; Bagchi, 1999). With the use of GA, no model needs to be generated. In addition, GA does not need to operate in a geometrically defined solution space. Most importantly, GAs have also demonstrated an ability to operate with noisy observations, such as the results generated by a simulation model (Goldberg, 1989). Hence, genetic algorithm is proposed for optimizing the weights used in this study.

Each GA run (involving 400 five-replicate simulation runs, refer to Section 6.3.2.8) takes approximately 210 minutes on an Intel Pentium M processor of 1.6 GHz with 1.5 GB of RAM. Due to the length of simulation time required, this research has not investigated the reactive optimization of weights in depth. However, acceptable results with re-optimization of weights carried out after every 500 and 1000 lots using a provisional binary coded GA (in which shorter simulation run length is used to cut down on the simulation run time) are reported in Ang and Sivakumar (2007).

In the design of the NEXTS decision policy, preferences are integrated with a multiobjective design search and optimization environment where search and optimization is performed by a real-coded GA. The optimized weights are then substituted into the MOFCS rule for the selection of the next lot. The proposed NEXTS decision policy is discussed in detailed in the next section.

6.3 LOOK AHEAD SIMULATION-BASED GENETIC ALGORITHM WITH DESIRABILITY FUNCTION

The NEXTS decision policy specifies how a scheduling decision is to be made. The decision policy is made up of a MOFCS rule that is applied for the selection of the next lot and a Look Ahead Simulation-based Genetic Algorithm with Desirability function (LASGAD) that optimizes the weights used in the MOFCS rule in accordance with user preferences. Since the decision makers are better at pinpointing the preferred operating point(s) than providing a set of suitable weights to apply, it would be desirable to have the weights of each criterion automatically chosen on-line in near-real-time to deal with the factory's changing condition.

The overview of the proposed Look-Ahead Simulation-based Genetic Algorithm with Desirability function (LASGAD) is summarized in Figure 6.4. The purpose of the main simulation model is to simulate the real-time factory environment in which LASGAD is applied. In actual implementation, the main simulation model is not required and LASGAD can be implemented on-line in the wafer fab.

Figure 6.4: Overview of look-ahead simulation-based genetic algorithm with desirability function (LASGAD)

Genetic algorithm (GA) is used to search for the desired or optimized weights for the combinatorial rule applied for the next job selection (see Step 1A in Figure 6.4). When GA is invoked, the sub simulation model (i.e. a simulation model embedded within another simulation model) is automatically generated and updated with the jobs and machines attributes and status available from the main simulation model (see Step 1B). The purpose of the sub simulation model is to evaluate the fitness of chromosomes that encode the weights for the MO rule (see Step 2). For each set of applied weights, the sub simulation model looks ahead in time and provides the performance estimates of the system over the time window covered by the simulation replication. Simulation is used because stochastic scheduling with multiobjectives and sequence-dependent setup (SDS) is analytically intractable.

The multiple conflicting performance measures returned by the sub simulation model (see Step 3 in Figure 6.4) is then combined using the appropriate desirability function into one objective function (see Step 4). This single objective function returns the fitness (in the form of total desirability value) for each chromosome (see Step 5) to the GA to be utilized in selection of the next generation of weights to be evaluated. Steps 2 to 5 are iterative until convergence or number of generations is met. After GA completes its search, the optimized weights are returned and applied in the MOFCS rule used in main simulation model (see Step 6).

The desirability function used in LASGAD has been addressed in Section 6.1.1. Discrete event simulation and genetic algorithms used in LASGAD are described in details in Sections 6.3.1 and 6.3.2 respectively. Section 6.4 compares the performance obtained when weights are optimized by response surface methodology (RSM) with desirability function and LASGAD.

6.3.1 DISCRETE EVENT SIMULATION

Figure 6.5 shows the conceptual application of discrete event simulation in LASGAD. There are two simulation modules created in AutoSchedAP. The main simulation module, used to simulate what is occurring on the factory real-time, is purely for evaluating the LASGAD, and is not required for actual implementation. The $\{PI, DL\}$ problem configuration (see Section 5.4.2) and the selected MOFCS rule that is made up of FIFO, CR, and SSU rules (refer to Section 0) are used in the main simulation model. The optimization of weights used in the MOFCS rule can be performed periodically (time-driven), upon the occurrence of events (event-driven), or under a hybrid depending on the dynamism of the factory and how

responsive it must be to the state of the system. The sub simulation module is used to look-ahead in simulated time to get the estimates of the steady state performances for the evaluation of each set of weights.

Figure 6.5: Conceptual application of main and sub simulation modules

Suppose at simulation time t_l (refer to Figure 6.5), the genetic algorithm (GA) is invoked to carry out weights optimization. At the same time GA is invoked, a sub simulation model is automatically generated. Auto-model generation has been implemented and used in the semiconductor industry (Sivakumar and Chong, 2000, Sivakumar and Gupta, 2006). Hence, the generation of a discrete-event simulation model on-line, in near-real-time is definitely viable. In this research, the sub simulation model is automatically generated and updated with the machine and WIP status present in the main simulation model at time t_l . The sub simulation model uses the same MOFCS rule, and lot arrival and processing time distributions (with different random number streams) as the main simulation model.

6.3.1.1 Determining Number of Replications for the Sub Simulation

Due to the embedded random variations in discrete-event simulations, the length of sub simulation and a certain number of replications are required to obtain an adequate confidence interval in $avgCT$, $sdCT$, and $avgTARD$ in order to estimate the fitness of each set of weights. Due to the computational requirement of simulation runs in on-line application, a preliminary study is conducted to determine the minimum number of replications required to establish a reasonable confidence in the output measures.

Given an initial number of replications, n_o , it is assumed, for each simulation output performance measure, that h_o is the resulting half-width of the 95% confidence interval. The required number of replications, n , to achieve a desired half-width, h , is estimated using (Law, 2007):

$$n = n_o \times (h_o / h)^2. \quad (6.4)$$

In this study, it is desired to have the half-width of the 95% confidence interval to be less than 5% in each of the simulation performance criterion (set as h). In a preliminary study, 20 replications (i.e. $n_o = 20$) of simulation for 30,000 lots are carried out to obtain the resulting half-width, h_o . The estimated number of replications, n required is given in Table 6.1 for each of the three dispatching rules (i.e. FIFO, CR, SSU).

From Table 6.1, it can be observed that it is insufficient to look at only one performance criterion to determine the number of simulation replications. If it is based solely on the average or standard deviation of cycle time (i.e. $avgCT$ or $sdCT$)

performance criterion, a one-replicate simulation might be sufficient to meet the required confidence interval. However, if the CR^2 rule is applied, more than four simulation replications are necessary. This is so possibly because the resulted average tardiness ($avgTARD$) value is very low and this makes the $avgTARD$ response more susceptible to the embedded random variations in a discrete-event simulation than $sdTARD$.

Table 6.1: Estimated number of replications required

		Performance Criterion:		
		$avgCT$	$sdCT$	$avgTARD$
Rules:	FIFO	0.21	0.14	0.99
	CR	0.20	0.12	4.18
	SSU	0.27	0.21	1.76

For example, the MOFCS rule with the set of applied weights $\{w_{FIFO} = 0.167, w_{CR} = 0.667, w_{SSU} = 0.167\}$ resulted in a mean $avgTARD$ and $sdTARD$ of 1.176 and 8.627 respectively for the $\{PI, DL\}$ problem configuration and the same (even slight) variation to the tardiness of one or more lots is likely to affect $avgTARD$ more. This is a concern as the proposed NEXTS decision policy will make use of look-ahead simulations to provide predicted performance in the selected objectives for GA to optimize the weights of the MOFCS rule. If it is possible to achieve the desired results via the $avgTARD$ objective, it is possible to do so via the $sdTARD$ objective. This further explains why, between the two positively correlated responses (see Section 5.5), the $sdTARD$ objective is dropped instead of the $avgTARD$ objective. The simulation results based on this set of weights (i.e. $\{w_{FIFO} = 0.167, w_{CR} = 0.667, w_{SSU} = 0.167\}$) will be used subsequently to represent user specified target values.

² The CR rule is equivalent to applying a weight vector of $\{0,1,0\}$ to the MOFCS rule where the order of weights in the vector is applied correspondingly to FIFO, CR, and SSU respectively.

As such, the evaluation of each chromosome will be based on the averaged performance of five sub simulation replications.

6.3.2 GENETIC ALGORITHM

Figure 6.6 shows the genetic search process of LASGAD. In GAs, each individual solution is called a chromosome. LASGAD begins with an initial population of randomly generated and/or pre-defined elite chromosomes in the initialization step. Through the use of genetic operators (i.e. selection, crossover, and mutation), modifications to the chromosomes of randomly selected solutions are systematically introduced to generate a new population of candidate solutions.

Figure 6.6: Flowchart of LASGAD

One cycle (iteration) of these GA operations to produce offspring is referred to as a generation in the GA terminology. The new generation of chromosomes is then evaluated using replications of simulation runs. Multiple conflicting performance measures obtained from the simulation runs are then combined into one objective function (i.e. total desirability) that is used as a measure of the fitness of each chromosome. Termination is tested at the end of each generation. If the termination

criterion is not met, the three GA operators iteratively operate upon the population to move it towards the optimization of certain criteria (Goldberg, 1989). The procedure is continued till the termination criterion is met. Each of the GA operators, sub-simulation and desirability computation components as shown in Figure 6.6 will be described in the following sub-sections.

6.3.2.1 Genetic Algorithm and Simulation Platform

The simulation is carried out using AutoSchedTM Accelerated Processing (AutoSchedAP) v7.2 by Brooks Automation Inc. As AutoSchedAP is built with a C++ class library, this makes GALib (Wall, 1996), a well documented C++ library of GA objects, easily integrated into AutoSchedAP. GALib is highly customizable and this allows the easy incorporation of any representation, genetic operators, and functionality required in this study.

6.3.2.2 Pareto-Based Versus Preference-Based Genetic Algorithm Approaches

Between using (i) Pareto-based approaches such as NSGA-II (Deb et al., 2002) to generate many non-dominated solutions in the population and then applying user preferences to select one candidate solution, and (ii) aggregation/preference-based approaches to generate only one solution by transforming the multiobjective problem into a single objective optimization problem using the desirability function, the latter is chosen for use in this research.

For on-line, near-real-time application, the number of generations and population size should ideally be kept to a minimum as multi-replicate simulations for the evaluation of each chromosome can be rather time-consuming. For Pareto-based approaches, the emphasis of *all* non-dominated solutions in a population may not

produce enough selection pressure for a small population to move towards the user specified target fast enough. In addition, for a large number of objectives, there is a need for an exponentially larger number of points (i.e. larger population size) to represent a higher-dimensional Pareto-optimal front (Deb and Sundar, 2006). Since the decision maker typically has a preferred operating point (based on historical data or off-line study results), and would not be present physically to evaluate each and every trade-off solutions after every run (i.e. automation preferred), the selected preference-based approach would be adequate (Deb, 2001).

6.3.2.3 Representation

In a binary-coded GA, the higher the required precision, the larger the string length. The string length also increases with the number of decision variables. For large binary strings, the population size requirement is also large (Goldberg et al., 1992), and this increases the computational complexity of the algorithm. Herrera et al. (1998) discuss the drawbacks of binary coding and highlight the advantages of real-coding. The current problem could be addressed using binary-coded GA as shown by Ang and Sivakumar (2007). However, a real parameter (or real-coded) GA is eventually selected for use in this study as real numbers far outperform the binary-to-decimal representation for most problems (Wall, 1996). The use of a real parameter GA also allows for future consideration of more decision variables.

As such, each chromosome contains an array of real numbers representing a set of corresponding weights for each rule used in a weighted aggregation approach. For example, in the multi-objective rule made up of FIFO, CR, and SSU rules (known as MOFCS rule), the first real number in the array represents the weight used for

the FIFO rule, and the two following real numbers the weights used for the CR and SSU rules respectively.

The actual value of each weight is not important. Rather, it is the relative magnitude of the weights when compared with another weight that is important (Dabbas et al., 2003). For example, the three weights could be $\{0.05, 0.10, 0.15\}$ or they could be $\{5, 10, 15\}$. In both cases, the same results are obtained because the ratio among the weights is the same for each set. In fact, they could be similarly normalized to $\{0.167, 0.333, 0.5\}$ with their sums equal to one. In this study, the value of the weights are kept between zero and one and subsequently normalized with their sums equal to one when used in the MOFCS rule.

6.3.2.4 Selection and Population Replacement

The selection process consists of determining from which chromosomes the next population would be generated. This study applies a tournament selector that uses the roulette wheel method to select two individuals and then picks the one with the higher fitness. Since it is possible that the fittest chromosomes of the population may fail to produce offspring that are at least as fit in the next generation, to prevent fitter chromosomes from being lost in the evolutionary process, an elitist strategy is adopted by copying some of the best individuals from one generation to the next (Goldberg, 1989).

The advantage of an elitist strategy over traditional probabilistic reproduction is that the best solution is monotonically improving from one generation to the next. The potential downside is population convergence to a local minimum. This can, however, be overcome by using higher mutation rates. The commonly known $(\mu+\lambda)$

selection strategy is applied in which the μ parents and λ offspring compete for survival and the μ best of offspring and old parents are selected as parents of the next generation.

In this study, $\mu = \lambda$ is used, that is, there is same number of offsprings as parents, and there is higher chance for a number of best performing chromosomes to be chosen to form the next generation. With a larger number of best performing chromosomes, GA performance can be improved by increasing mutation and crossover rates without worrying that the higher rates will destroy the fittest strings.

Apart from the elitist (replacement) strategy used from one generation to another, a global replacement strategy can also be considered to bring the best individual from one GA run to another GA run (i.e. another instance when GA is invoked). While changes occur inevitably in dynamic environments, the severity of change may be gradual with some form of structure in their evolution (Branke, 2002). Hence, by retaining the best individual, computational efficiency is improved. Apart from the best individual carried forward from previous GA run, the remaining individuals are randomly generated at the beginning of each GA run (except for the first run where equal weights are applied).

6.3.2.5 Crossover

In this study, the blend crossover (BLX- α) operator introduced by Eshelman and Schaffer (1993) is applied in the real-parameter GA with some modifications to ensure that the weights for the MOFCS rule are bounded within the specified range. Crossover is not applied to all pairs of individuals selected for mating. Instead, a

random choice is made in accordance to a pre-specified crossover probability p_c (one of the GA parameters).

For the i^{th} decision variable in two parent solutions $x_i^{(1)}$ and $x_i^{(2)}$, the BLX- α produces an offspring (child) that takes a value of x_i' that is randomly generated from the interval $[x_i^{(1)} - \alpha(x_i^{(2)} - x_i^{(1)}), x_i^{(2)} + \alpha(x_i^{(2)} - x_i^{(1)})]$, where $x_i^{(1)} < x_i^{(2)}$. If the bounds are exceeded, the value of x_i' is adjusted as follows:

$$x_i' = \begin{cases} x_{i,L} & \text{if } x_i' < x_{i,L} \\ x_{i,U} & \text{if } x_i' > x_{i,U} \end{cases} \quad (6.5)$$

where $x_{i,L}$ and $x_{i,U}$ are the respective lower and upper bounds on the value of decision variable i .

The BLX- α operator is illustrated in Figure 6.7. In a number of test problems, Eshelman and Schaffer (1993) have reported that BLX-0.5 (with $\alpha = 0.5$) performs better than BLX operators taking any other α values. In our study, $\alpha = 0.5$ is also selected to ensure the balance between exploitation and exploration of the search space. This balance is to facilitate a thorough search and is necessary to prevent GA from converging to a local minimum and from being unable to escape from a local minimum.

Figure 6.7: Blend crossover operator

6.3.2.6 Mutation

The mutation operator defines the procedure for mutating each chromosome. After the crossover operation, the child chromosomes are subjected to the mutation operation with a mutation probability of p_m . This mutation probability determines whether or not each element in the chromosome should be mutated. The non-uniform mutation introduced by Michalewicz (1992) is used.

The new value of the i^{th} decision variable, x_i'' after mutation at generation γ is given as

$$x_i'' = \begin{cases} x_i' - f(\gamma, x_i' - x_{i,L}), & \text{if } \tau = -1 \\ x_i' + f(\gamma, x_{i,U} - x_i'), & \text{if } \tau = +1 \end{cases} \quad (6.6)$$

where τ takes a value of -1 or 1, each with a probability of 0.5 and function $f(\gamma, \xi)$ is defined as

$$f(\gamma, \xi) = \xi \times \left(1 - u^{(1-\gamma/\gamma_{\max})^\beta}\right) \quad (6.7)$$

where u is a random number in $[0,1]$, $\gamma_{\max}$ is the maximum number of allowed generations, and β is a user-defined parameter that determines the degree of dependency on the number of iterations.

In this method, the probability of creating a solution closer to the parents is more than the probability of creating one away from it. The non-uniform mutation scheme acts like a uniform distribution in the early generations, but allows for a focused search in later generations (Deb, 2001). This can be observed in Figure 6.8 where the $\left(1 - u^{(1-\gamma/\gamma_{\max})^\beta}\right)$ value gets smaller with generations and eventually becomes zero when it approaches the specified maximum number of generations.

The number of generations is determined to be 20 (see Section 6.3.2.8) and hence, Figure 6.8 illustrates the value of $\left(1 - u^{(1-\gamma/\gamma_{\max})^\beta}\right)$ for up to 20 generations.

The non-uniform mutation generally avoids premature convergence and makes the search considerably faster and more accurate (Michalewicz, 1992). In this study, the relative size of the weights matters (see Section 6.3.2.3). As a change in each weight will affect the relative size of all weights applied, the mutation of each weight is limited to a maximum of 50% of its original value (i.e. before mutation). This is to avoid applying too much mutation that will delay convergence and make the search random (Kim, 2003). Hence, $\beta = 1$ is used in this study.

Figure 6.8: Plot of $\left(1 - u^{(1-\gamma/\gamma_{\max})^\beta}\right)$ against γ th generation

6.3.2.7 Fitness Evaluation

Discrete-event simulation and the desirability function approach are used to evaluate the fitness (or performance) of each chromosome. The fitness measure is based on average cycle time ($avgCT$), standard deviation of cycle time ($sdCT$), and average tardiness ($avgTARD$) which are the identified key performance indices of

this study (see Section 5.5). The values of these performance indices are obtained using simulation, and are each expressed as a desirability function and combined into a single objective function to give the fitness for each chromosome. The details of the sub simulation and desirability function are described in Sections 6.3.1 and 6.1.1 respectively.

6.3.2.8 Stopping Criterion

There are several stopping criteria that have been cited in the literature (Aytug and Koehler, 2000; Safe et al., 2004). A common criterion is to stop when all generations reach the same solution. Another common approach is to stop when the average fitness (solution quality) of several consecutive generations do not show significant improvement. Allowing for computational efficiency, this study uses a maximum number of simulation runs as the stopping criterion (also used by Azadivar and Tompkins (1999), and Yang et al. (2007)). This criterion is often associated with a discrete-event simulation model that usually requires significant computational time.

In Yang et al. (2007), it is reported that the evaluation of one chromosome requires one-replicate of the simulation run. However, in our study, five replications of the sub simulation are necessary (see Section 6.3.1.1) to achieve the required confidence interval. Thus, the total number of five-replicate simulation runs required is a multiple of the population size, and the number of GA search generations.

In order to estimate the required number of five-replicate simulation runs, existing rule of thumb is referenced and the GA parameters are chosen based on a few trial

simulation runs. Population size (P), crossover probability (p_c), and mutation probability (p_m) are fixed at 20, 0.6, and 0.3 respectively for this study. The population size and crossover probability setup is commonly seen in the literature (Yang et al., 2007). With regards to the mutation probability, Harvey (1995) suggests that there should be approximately no mutation 36.8% (i.e. $100 \times 1/e$) of the time. Since p_m is set at 0.3, this implies that there is an estimated $100 \times (1-p_m)^3 = 34.3\%$ chance of no mutation. The mutation probability is also close to the rule of thumb of 1/number of variables (i.e. $1/3 = 0.333$).

Since there is no real user preferred or optimal data based on the $\{PI, DL\}$ problem configuration, the experiments to determine the stopping criterion are best performed by comparing the results to a known set of performance measures based on a pre-selected set of weights. The performance measures that corresponds to weights set of $\{w_{FIFO} = 0.167, w_{CR} = 0.667, w_{SSU} = 0.167\}$, as retrieved from Table F.3 in Appendix F based on earlier experiments in Section 5.4.3.4, represent the user specified target values. The target values, lower and upper bounds, and the values for the two exponents ω_1 and ω_2 used in the desirability function for criterion with specified target (refer to Equation (6.4)) is summarized in Table 6.2.

For the stopping criterion experiments, the optimization of weights is only performed once. This means that LASGAD is executed only once within the main simulation model. The main simulation model uses an equally weighted $\{0.333, 0.333, 0.333\}$ MOFCS rule to select the initial lots until the newly optimized weights are available and applied. The number of generations is varied from one to 20 with increments of one. The experiment, thus, has 20 different GA parameter

combinations for the stopping criterion experiments. Each GA parameter setup is repeated five times (i.e. the main simulation model is executed five times).

Table 6.2: Values used in the desirability function for criterion with specified target

Variables	Criterion x_i		
	<i>avgCT</i>	<i>sdCT</i>	<i>avgTARD</i>
Lower bound, $L(x_i)$	30	0	0
Upper bound, $U(x_i)$	350	300	70
Target value, $T(x_i)$	296.291	246.997	1.176
ω_1	0.1	0.1	0.1
ω_2	1	1	1

Figure 6.9 shows the mean of the best total desirability value and its associated standard deviation for each GA parameter setup while Figure 6.10 to Figure 6.12 respectively display the mean and standard deviation of *avgCT*, *sdCT*, and *avgTARD* performances that go into the computation of total desirability for each GA parameter setup. The solutions begin to converge after the tenth generation, but a better total desirability value (see Figure 6.9) is obtained around the sixteenth generation. Thus, the number of generations of 20 in combination with the population size of 20 will be a conservative estimate as the stopping criterion, which results in a total number of 400 five-replicate simulation runs.

Figure 6.9: Total desirability value versus number of generations

Figure 6.10: Average cycle time versus number of generations

Figure 6.11: Standard deviation of cycle time versus number of generations

Figure 6.12: Average tardiness versus number of generations

6.3.2.9 GA Parameters Selection

Some studies such as Pongcharoen et al. (2002) and Yang et al. (2007) have used response surface methodology to optimize the GA parameters that also includes the ratio of population size, P to number of generations, G (i.e. P/G) in addition to crossover and mutation probabilities. The main aim of the simulation experiments is to illustrate the feasibility and capability of the proposed NEXTS decision policy (i.e. how LASGAD works to optimize the weights in the MOFCS rule that is used for the selection of the next lot). Hence, the need to spend excessive hours of computation time on fine tuning parameters at this experimental stage is circumvented although sufficient care is taken to ensure reasonable GA parameters are applied. This is to ensure a relatively good balance is achieved between intensification (exploitation of good solutions currently at hand for intensive search in a particular region of the solution space) and diversification (exploration of the total search space). Crossover does the former, and mutation the latter.

Although the crossover and mutation probabilities (i.e. $p_c = 0.6$; $p_m = 0.3$) have already shown promising results in the stopping criterion experiments (refer to Section 6.3.2.8), the crossover and mutation probabilities are varied over another level (i.e. $p_c = 0.9$; $p_m = 0.6$) (see Table 6.3).

Table 6.3: 2-level experimental setup

GA Parameter	Levels	
	1	2
p_c	0.6	0.9
p_m	0.3	0.6

Crossover probability of 0.9 is not uncommon and has also been considered in many studies (e.g. Pongcharoen et al., 2002; Yang et al., 2007). With the use of elitist reproduction method (for intensification) to ensure that high mutation rates will not destroy the fittest strings, the mutation rate cannot be too low or there would be insufficient exploitation. Preliminary tests involving a mutation probability of $p_m = 0.1$ are unable to achieve the user specified target values as insufficient exploitation could have resulted in population convergence to a local minimum. The use of non-uniform mutation (see Section 6.3.2.6) still allows a focused search in the later generations even with a high level of mutation probability, since the amount of mutation approaches zero with increasing generations. As such, a higher level of mutation probability of 0.6 is tested instead.

The target values, lower and upper bounds, and the values for the two exponents ω_1 and ω_2 used in the proposed desirability function for criterion with target (refer to Equation (6.4)) used in the stopping criterion experiments (see Table 6.2) is similarly applied for the GA parameter selection experiments. Each combination of GA parameter setup is repeated five times and the mean and 95% confidence interval of the simulation results is tabulated in Table 6.4.

The results among the four sets of tested GA parameters (i.e. p_c and p_m) are not significantly different. However, the combination $p_c = 0.6$ and $p_m = 0.3$ is selected for use since it resulted in the highest total desirability value and optimized weights are the closest to the weight set $\{w_{FIFO} = 0.167, w_{CR} = 0.667, w_{SSU} = 0.167\}$ that corresponds to producing the user-specified target performance values.

Table 6.4: Mean and 95% confidence interval of simulation results obtained from the best chromosome for each set of GA parameters

GA Parameters	$avgCT$	$sdCT$	$avgTARD$	Total Desirability, D	W_{FIFO}	W_{CR}	W_{SSU}
$p_c = 0.6,$ $p_m = 0.3$	292.871 $\pm$ 1.099	244.811 $\pm$ 0.476	1.260 $\pm$ 0.042	0.99881 $\pm$ 0.00027	0.167 $\pm$ 0.009	0.658 $\pm$ 0.004	0.175 $\pm$ 0.010
$p_c = 0.6,$ $p_m = 0.6$	292.041 $\pm$ 2.470	244.233 $\pm$ 0.825	1.264 $\pm$ 0.069	0.99864 $\pm$ 0.00059	0.172 $\pm$ 0.020	0.658 $\pm$ 0.006	0.170 $\pm$ 0.025
$p_c = 0.9,$ $p_m = 0.3$	292.403 $\pm$ 1.618	245.734 $\pm$ 0.619	1.300 $\pm$ 0.064	0.99870 $\pm$ 0.00043	0.153 $\pm$ 0.015	0.658 $\pm$ 0.006	0.189 $\pm$ 0.018
$p_c = 0.9,$ $p_m = 0.6$	292.804 $\pm$ 1.263	245.246 $\pm$ 1.021	1.254 $\pm$ 0.014	0.99872 $\pm$ 0.00020	0.163 $\pm$ 0.010	0.658 $\pm$ 0.005	0.179 $\pm$ 0.009

6.4 COMPARISON OF RSM & LASGAD

In this section, the optimized weights and results obtained using the commonly used response surface methodology (RSM) with desirability function and the proposed look-ahead simulation-based genetic algorithm with desirability function (LASGAD) method are compared for the case where all criteria are to be optimized (i.e. minimized) and the case where user target performances are specified for all the criteria.

6.4.1 RESPONSE SURFACE METHODOLOGY (RSM)

As the weights used in this research are dependent, mixture experiments are used as they can address the issue where the decision variables considered in the experiment must add to 100% (Cornell, 1995; Dabbas, 1999). Table 5.14 shows the 19 sets of weights (candidate points) for the mixture design that is generated for a full cubic model. Design-Expert® Software (Version 6) by Stat-Ease Inc. is used to fit the meta-models to each of the simulation response data (i.e. $avgCT$, $sdCT$, and

avgTARD performances based on 20 replications) presented in Table F.3 (see Appendix F). Design-Expert utilizes a simplex steepest ascent method for finding the optimum. A starting point is randomly selected and the steepest ascent/descent algorithm is applied to search along steepest slope up/down to a maximum/minimum. The response surface methodology with desirability function approach that uses the steepest ascent method will be hereafter known in short as RSM, for brevity.

The analysis of variance (ANOVA) is performed for each response to determine the statistically significant factors (i.e. w_{FIFO} , w_{CR} , w_{SSU}) before the least squares analysis is carried out to estimate the significant parameters for the meta-models. The detailed procedural steps to fit meta-models for the responses can be referred in Cornell (1995), Dabbas (1999), and Dabbas et al. (2003). The ANOVA, and sequential sum of squares tables, as well as the fitted meta-model for each response can be found in Appendix G.

6.4.2 COMPARISON OF RSM AND LASGAD FOR CRITERIA WITH USER SPECIFIED TARGET VALUES

The individual desirability for each criterion with specified target is computed using Equation (6.4) and the total desirability is computed using Equation (6.2). The target values, upper and lower bounds, and the two exponents ω_1 and ω_2 used to compute the individual desirability are found at the bottom of Table 6.5. Based on the fitted meta-models ('reduced' and 'not reduced'), the optimized set of weights (best solution) together with its corresponding predicted performance and total desirability are presented in Table 6.5.

‘Not reduced’ implies that the fitted meta-model is based on the highest order polynomial (where the additional terms are significant and the model is not aliased) while ‘reduced’ implies that the fitted meta-model is a reduced model in which insignificant model terms (not counting those required to support hierarchy) are removed to improve the model (resulting in a larger F Value). The optimized weights (and their corresponding predicted performance) found using LASGAD for five separate runs are also tabulated in Table 6.5.

Table 6.5: Optimized weights and its corresponding predicted performance and total desirability

Fitted Meta-Model	Weights			Predicted Performance			Predicted Total Desirability
	w_{FIFO}	w_{CR}	w_{SSU}	$avgCT$	$sdCT$	$avgTARD$	
RSM Meta-Model (Not Reduced)	0.08	0.68	0.25	284.981	246.997	1.17602	0.99855
RSM Meta-Model (Reduced)	0.07	0.68	0.25	284.596	246.997	1.176	0.99850
LASGAD - Run1	0.183	0.661	0.157	294.312	244.582	1.214	0.99924
LASGAD - Run2	0.171	0.651	0.178	293.784	244.54	1.34	0.99856
LASGAD - Run3	0.161	0.656	0.183	292.944	245.722	1.254	0.99903
LASGAD - Run4	0.157	0.661	0.183	291.196	244.34	1.252	0.99859
LASGAD - Run5	0.163	0.660	0.177	292.12	244.872	1.24	0.99864
Target	0.167	0.667	0.167	296.291	246.997	1.176	-
Lower Bound	-	-	-	30	0	0	
Upper Bound	-	-	-	350	300	70	-
Parameter, $\omega_1 = 0.1$; Parameter, $\omega_2 = 1$							

From the results presented in Table 6.5, the optimized weights found using RSM is approximately $\{0.07, 0.68, 0.25\}$ while those found using LASGAD are very close to the targeted weight set of $\{0.167, 0.667, 0.167\}$. As there is a possibility that different weight sets could produce results close to the user specified target performances, all the weight sets in Table 6.5 are simulated using the same

$\{PI, DL\}$ problem configuration (described in Section 5.4.2) for 20 replications. The simulated performances and total desirability are presented in Table 6.6. The percentage difference of the weights and performances to the target values are shown in brackets. A negative percentage indicates that the value is smaller than the target value and vice versa.

Based on the total desirability computed using the simulated performance criteria (i.e. $avgCT$, $sdCT$, $avgTARD$) presented in Table 6.6, it can be said that LASGAD performs better than RSM in optimizing the weights based on a user specified target value for all the scheduling criteria since the total desirability is closer to a value of one (most desirable). In addition, smaller percentage differences between the predicted and simulated performance measures are observed in LASGAD than RSM.

Although the set of weights $\{0.167, 0.667, 0.167\}$ and its corresponding performances forms a design point in RSM, it is still possible that similar weights cannot be reproduced (obtained via numerical optimization using desirability function and the fitted meta-models) when the same $avgCT$, $sdCT$, and $avgTARD$ response values are specified as the target. This is clearly undesirable. The experimental results have shown that LASGAD performs better than RSM and has the capability to optimize the weights used in the MOFCS rule based on user specified target values.

Table 6.6: Simulated performance and total desirability using the optimized weights for criterion with target values

Methods	Weights (% Difference from Target)			Simulated Performance (% Difference from Target)			Computed Total Desirability
	w_{FIFO}	w_{CR}	w_{SSU}	$avgCT$	$sdCT$	$avgTARD$	
RSM	0.070 (-58.1%)	0.68 (1.95%)	0.25 (49.7%)	291.018 ± 4.346 (-1.78%)	254.550 ± 2.323 (3.06%)	1.289 ± 0.071 (9.61%)	0.932 ± 0.028
LASGAD - Run1	0.183 (9.58%)	0.661 (-0.90%)	0.157 (-5.99%)	296.528 ± 4.626 (0.08%)	244.969 ± 2.124 (-0.82%)	1.221 ± 0.062 (3.83%)	0.963 ± 0.030
LASGAD - Run2	0.171 (2.40%)	0.651 (-2.40%)	0.178 (6.59%)	294.096 ± 4.468 (-0.74%)	243.555 ± 2.009 (-1.39%)	1.345 ± 0.067 (14.37%)	0.971 ± 0.024
LASGAD - Run3	0.161 (3.59%)	0.656 (-1.65%)	0.183 (9.58%)	294.300 ± 4.463 (-0.68%)	245.185 ± 2.137 (-0.73%)	1.292 ± 0.076 (9.86%)	0.968 ± 0.027
LASGAD - Run4	0.157 (-5.99%)	0.661 (-0.90%)	0.183 (9.58%)	294.059 ± 4.465 (-0.75%)	246.595 ± 2.171 (-0.16%)	1.278 ± 0.068 (8.67%)	0.966 ± 0.028
LASGAD - Run5	0.163 (-2.40%)	0.660 (-1.05%)	0.177 (5.99%)	294.651 ± 4.529 (-0.55%)	245.970 ± 2.190 (-0.42%)	1.255 ± 0.067 (6.72%)	0.965 ± 0.029
Target	0.167	0.667	0.167	296.291	246.997	1.176	1.000

6.4.3 COMPARISON OF RSM AND LASGAD FOR CRITERIA TO BE MINIMIZED

The individual desirability for each criterion with specified target and the total desirability are computed using Equation (6.3) and (6.2) respectively. The upper and lower bounds, and the exponent ω used to compute the individual desirability are found at the bottom of Table 6.7. RSM (both the non-reduced and reduced meta-models are considered) and LASGAD (five separate runs are carried out) are used to minimize the $avgCT$, $sdCT$, and $avgTARD$ responses and the optimized weights obtained (including the corresponding predicted responses and total desirability) are presented in Table 6.7. All the weight sets in Table 6.7 are simulated using the $\{PI, DL\}$ problem configuration (described in Section 5.4.2) for

20 replications. The simulated performances and total desirability are presented in Table 6.8.

Table 6.7: Optimized weights and the predicted performance and total desirability

Methods	Weights			Predicted Performance			Predicted Total Desirability
	W_{FIFO}	W_{CR}	W_{SSU}	$avgCT$	$sdCT$	$avgTARD$	
RSM (Meta-Model Not Reduced)	0.37	0.20	0.43	263.190	156.943	16.935	0.46115
RSM (Meta-Model Reduced)	0.31	0.12	0.57	256.009	163.868	19.820	0.45716
LASGAD - Run1	0.414	0.202	0.384	259.996	151.82	15.188	0.47734
LASGAD - Run2	0.436	0.201	0.363	262.476	147.944	15.066	0.47731
LASGAD - Run3	0.384	0.199	0.417	256.576	158.476	15.970	0.47370
LASGAD - Run4	0.359	0.289	0.352	260.330	158.408	12.318	0.47759
LASGAD - Run5	0.401	0.244	0.355	262.290	152.264	13.842	0.47660
Lower Bound	-	-	-	30	0	0	
Upper Bound	-	-	-	350	300	70	-
Parameter, $\omega = 1$							

Table 6.8: Simulated performance and total desirability using the optimized weights for criterion to be minimized

Methods	Weights			Simulated Performances*			Computed Total Desirability*
	W_{FIFO}	W_{CR}	W_{SSU}	$avgCT$	$sdCT$	$avgTARD$	
RSM (Meta-Model Not Reduced)	0.37	0.20	0.43	262.471 $\pm$ 3.885	161.790 $\pm$ 1.298	17.492 $\pm$ 0.646	0.455 $\pm$ 0.010
RSM (Meta-Model Reduced)	0.31	0.12	0.57	254.370 $\pm$ 3.676	173.242 $\pm$ 1.506	22.764 $\pm$ 0.788	0.440 $\pm$ 0.009
LASGAD - Run1	0.414	0.202	0.384	266.992 $\pm$ 3.955	153.167 $\pm$ 1.081	16.196 $\pm$ 0.548	0.460 $\pm$ 0.010
LASGAD - Run2	0.436	0.201	0.363	269.375 $\pm$ 3.827	148.855 $\pm$ 1.059	15.976 $\pm$ 0.545	0.460 $\pm$ 0.009
LASGAD - Run3	0.384	0.199	0.417	263.794 $\pm$ 3.878	159.201 $\pm$ 1.023	16.937 $\pm$ 0.682	0.457 $\pm$ 0.010
LASGAD - Run4	0.359	0.289	0.352	267.957 $\pm$ 3.926	160.076 $\pm$ 1.179	13.014 $\pm$ 0.541	0.460 $\pm$ 0.009
LASGAD - Run5	0.401	0.244	0.355	268.659 $\pm$ 3.902	153.960 $\pm$ 1.039	14.438 $\pm$ 0.564	0.461 $\pm$ 0.009

* Based on 20 simulation replications

Due to difference in the value of exponent(s) used between the setting where criterion are to meet the specified target value (refer to Section 6.4.2) and the setting where criterion is to be minimized (this Section), the total desirability values between the two cases cannot be compared. Only those within the same setting are comparable. As can be seen from Table 6.8, all five separate runs of LASGAD resulted in higher total desirability values than RSM (whether the meta-models are reduced or not). The experimental results imply that better weights could be found using LASGAD than RSM for the scenario in which all criteria are to be minimized.

6.5 SUMMARY

It has been shown how user preferences can be incorporated into decision-making via the use of desirability functions. In order to prevent any disruptions or occurrence of unexpected events from invalidating the weights used in the MOFCS rule, it is essential to continuously search for better weights. The concept for the reactive optimization of weights used in the MOFCS rule is also proposed.

The NEXTS decision policy is able to consider multiple conflicting objectives using the MOFCS rule. LASGAD utilizes the desirability function to incorporate user preferences to guide the optimization of weights (towards the desired performances) used in the MOFCS rule automatically on-line in near real time to deal with the factory's changing condition. The optimized weights found are substituted into the MOFCS rule to make the next lot selection decision. In the evaluation of the NEXTS decision policy, sequence-dependent setups have also been considered.

When the NEXTS decision policy (LASGAD + MOFCS rule) is compared against existing decision policy (Response Surface Methodology, RSM + MOFCS rule), the proposed NEXTS decision policy showed better performances. In the case where user preferences (target values) are specified, RSM is unable to find the set of weights that would produce the user specified performances. In the case where all criteria are to be minimized, the proposed NEXTS decision policy turns out performance measures with a higher total desirability value. The experimental results imply that better weights could be found using LASGAD than RSM whether all criteria are to be minimized or to meet user specified targets.

The proposed NEXTS time policy and the proposed NEXTS decision policy have shown promising results when studied on their own. It is anticipated that even better performance could be obtained when both the policies are applied in tandem in a case study (Chapter 7).

CHAPTER 7

INDUSTRIAL APPLICATION: CASE STUDY

The NEXt Task Scheduling (NEXTS) methodology is made up of a NEXTS time policy and a NEXTS decision policy. This chapter begins by summarizing the proposed time and decision policies that will be combined. The combined policies are evaluated using a representative wafer fab model. The case study model, data, and experimentation are summarized in Section 7.2 while the results and discussion are presented in Section 7.3.

7.1 PROPOSED TIME AND DECISION POLICIES

The time policy specifies when to schedule or when to make a scheduling decision to select the next task. The fixed time period policy is applied in NEXTS time policy as it has the best performance as compared to the loadport-initiated time policy and the adjusted time period policy. In the fixed time period policy, a new lot is selected, from among all the lots available in the upstream buffers, for processing on a machine at a fixed time period before the machine is available. The fixed time period policy can be viewed as the case where the lot selection procedure waits as long as practical before committing a lot to a specific machine for processing.

The decision policy, on the other hand, specifies *how* to schedule or how a scheduling decision is being made. The decision policy is made up of a Look

Ahead Simulation-based Genetic Algorithm with Desirability function that searches and provides the optimal weights to the MOFCS rule (i.e. weighted aggregation of FIFO, CR, and SSU rules). The MOFCS rule is then applied to select the next lot for processing on the tool (machine).

7.2 CASE STUDY MODEL, DATA and EXPERIMENTATION

In this case study, the NEXTS time policy is applied to all the tools. However, due to the length of runtime required per run of LASGAD (approximately 250 minutes on an Intel Pentium M processor of 1.6GHz with 1.5GB of RAM without the use of distributed and parallel simulation techniques) to obtain the optimized weights, the same MOFCS rule (i.e. same optimized weight set) is applied to all non-batching tools. Hence, for each set of weights (in GA), the sub simulation model looks ahead in time and provides the performance estimates of the entire fab over the time window covered by the simulation replication.

For batching tools, lots are ranked using critical ratio. Lots at steps with the same setup can be batched. All batching tools in the wafer fab model shall use the following batching logic: A batch is created when

- (i) a specified maximum batch size is reached; or
- (ii) a specified minimum batch size is reached and the pre-specified wait duration for maximum batch size to form duration has been exceeded by the oldest lot in the potential batch.

Due to confidentiality issues as per the non-disclosure agreement signed with Chartered Semiconductor Manufacturing Ltd., the details of the model will not be

presented. Table 7.1 summarizes the components that are included in the wafer fab case study model. The experimental design is summarized in Table 7.2.

Table 7.1: Wafer fab case study model

Modeled Components	Details
Problem Description	<ul style="list-style-type: none"> • The wafer fab model is reduced and modified from an actual wafer fab model (for confidentiality issues). • The reduced wafer fab model includes 70 different tools that are used across the Diffusion, Lithography, Etch, Clean Tech, Implantation, Thin Film, and Metrology processes. • 11 representative product types that spans across the 65-130nm technology are considered, each of which requires approximately 710-1220 processing steps with re-entrant flow. • Limited capacity buffer on the machine (known as load ports) are considered. • Preventive maintenance, unscheduled downtime, and sequence-dependent setups (where applicable) are considered. • Due date of each lot is determined by assigning a flow allowance upon its arrival. • There is no differentiation of lots (i.e. normal lots, priority lots (hot or bullet lot), rework lots, qualification lots). • Rework, operators, time-link constraints, and yield losses are not modeled.
Performance Measures (Responses) Collected and Objectives	<ul style="list-style-type: none"> • Average cycle time, <i>avgCT</i> (To minimize) • Standard deviation of cycle time, <i>sdCT</i> (To minimize) • Average tardiness, <i>avgTARD</i> (To minimize)

Table 7.2: Summary of the experimental design for the wafer fab model

Variables/Parameters	Distribution/Values
Number of lot types	11
Raw Processing Time (<i>PT</i>) per lot	Lot Type 1: 22.0days / Lot Type 2: 23.8days Lot Type 3: 19.7days / Lot Type 4: 17.8days Lot Type 5: 20.0days / Lot Type 6: 22.4days Lot Type 7: 21.1days / Lot Type 8: 26.6days Lot Type 9: 26.9days / Lot Type 10: 27.1days Lot Type 11: 23.2days.

Raw Processing Time (PT) per lot per step	Approx. 3.2 to 440 mins depending on process steps.
Lot Release Policy	1 to 8 days between lot releases for each lot type.
Wafer starts	No data available for a reduced subset of tools used. Approximately 110 wafer starts per day / 3,300 wafer starts per month.
Transport Time (TT) of a lot	<ul style="list-style-type: none"> • $U[4,6]$ mins – applied when Dispatching Time Policy is used • Perturbation of $U[0,1.5]$ mins is introduced to NEXTS Time Policy to account for any variabilities that may cause the next lot to be delivered later than scheduled.
Due date	No data available. Set arbitrarily to arrival time of lot + δ , where δ takes value between 50 to 125 days depending on lot types.
Setup time	<ul style="list-style-type: none"> • Sequence-dependent setup time of $U[3,6]$ mins for a particular lithography tool group • 0.3 to 53 mins for other tools (some setups are sequence dependent, some setups are always needed)
Preventive Maintenance (PM)	Each tool may be subjected to one or more of the following: <ul style="list-style-type: none"> • Bi-Yearly: 27-48hrs / Yearly: 3-168hrs • Half-Yearly: 4-64hrs / Quarterly: 3-39hrs • Half-Quarterly: 1-6.5hrs / Monthly: 1-24hrs • Weekly: 1-51hrs
Unscheduled Down Time	<ul style="list-style-type: none"> • Actual PM duration differs from tool to tool. • MTTF: 12.7-23.9hrs (average: 21.6hrs across tool types)
Time Policies Applied	<ul style="list-style-type: none"> • MTTR: 0.1-11.3hrs (average: 2.4hrs across tool types) • Dispatching Time Policy • NEXTS Fixed Time Policy
Decision Policies* Applied	<ul style="list-style-type: none"> • FIFO / SPT / EDD / CR / LS / SSU / WINQ applied to all tools • MOFCS and LASGAD are applied to all non-batching tools, while CR and Same Setup are applied to all batching tools (hereafter known as MOFCS+CR rule for brevity)
Desirability Function	<p>When Dispatching Time Policy is applied:</p> <ul style="list-style-type: none"> • $LB(avgCT) = 190days$, $UB(avgCT) = 290days$ • $LB(sdCT) = 30days$, $UB(sdCT) = 90days$ • $LB(avgTARD) = 90day$, $UB(avgTARD) = 190days$ • $\omega = 1$ <p>When NEXTS Time Policy is applied:</p> <ul style="list-style-type: none"> • $LB(avgCT) = 60days$, $UB(avgCT) = 90days$ • $LB(sdCT) = 0day$, $UB(sdCT) = 15days$ • $LB(avgTARD) = 0day$, $UB(avgTARD) = 6days$ • $\omega = 1$

Genetic Algorithms

- $p_c = 0.6$; $\alpha = 0.5$
- $p_m = 0.3$; $\beta = 1$
- Population size, $P = 20$
- Number of generations, $N = 20$

*FIFO – First-In, First-Out / SPT – Shortest Processing Time / EDD – Earliest Due Date / CR – Critical Ratio / LS – Least Slack / SSU – Same SetUp / WINQ – Work In Next Queue
 $U[a,b]$ – Uniformly distributed between a and b minutes
 $LB(x)$ – Lower bound of criterion x / $UB(x)$ – Upper bound of criterion x

7.3 RESULTS AND DISCUSSION

Different combinations of the time policies (i.e. NEXTS and dispatching time policies) and decision policies (i.e. FIFO, SPT, EDD, CR, LS, SSU, WINQ, MOFCS rules) listed in Table 7.2 are studied. The optimized weight set found using LASGAD is tabulated in Table 7.3.

Table 7.3: Summary of the optimized weights found using LASGAD

Time & Decision Policies Configurations	w_{FIFO}	w_{CR}	w_{SSU}
Dispatching Time Policy & MOFCS+CR (without weights updating; optimized at the beginning of Q1)	0.1	0.86	0.04
NEXTS Time Policy & MOFCS+CR (optimized at the beginning of Q1)	0.64	0.36	0.0
NEXTS Time Policy & MOFCS+CR (optimized at the beginning of Q3)	0.0	0.0	1.0
NEXTS Time Policy & MOFCS+CR (optimized at the beginning of Q4)	0.56	0.27	0.17

Due to the lengthy runtime, the updating of weights is only performed twice. The first set of optimized weights is search at the beginning of Quarter 1 (Q1) and applied across a six-month period (Q1 and Q2) after warm-up. Subsequently, the weights are updated at the beginning of Quarter 3 and 4 (i.e. Q3 and Q4) and applied across Q3 and Q4 respectively. In order to compare the differences between

updating the weights and not updating, the performances for the following cases are collected:

- (i) MOFCS+CR (1) – refers to the case where the optimized weights $\{w_{\text{FIFO}} = 0.64, w_{\text{CR}} = 0.36, w_{\text{SSU}} = 0.0\}$ determined in Q1 are applied in the MOFCS rule throughout the year (Q1 to Q4). This is the case where the weights are not updated throughout the one year period considered.
- (ii) MOFCS+CR (2) – refers to the case where the optimized weights $\{w_{\text{FIFO}} = 0.64, w_{\text{CR}} = 0.36, w_{\text{SSU}} = 0.0\}$ determined in Q1 are applied in the MOFCS rule for Q1 and Q2, and the optimized weights $\{w_{\text{FIFO}} = 0.0, w_{\text{CR}} = 0.0, w_{\text{SSU}} = 1.0\}$ determined in Q3 are applied in the MOFCS rule for Q3 and Q4. This is the case where the weights are updated only once (i.e. every six months).
- (iii) MOFCS+CR (3) – refers to the case where the optimized weights $\{w_{\text{FIFO}} = 0.64, w_{\text{CR}} = 0.36, w_{\text{SSU}} = 0.0\}$ determined in Q1 are applied in the MOFCS rule for Q1 and Q2, the optimized weights $\{w_{\text{FIFO}} = 0.0, w_{\text{CR}} = 0.0, w_{\text{SSU}} = 1.0\}$ determined in Q3 are applied in the MOFCS rule for Q3, and the optimized weights $\{w_{\text{FIFO}} = 0.56, w_{\text{CR}} = 0.27, w_{\text{SSU}} = 0.17\}$ determined in Q4 are applied in the MOFCS rule for Q4. This is the case where the weights are updated twice (i.e. once after six months and another time after three months).

The average cycle time (avgCT), standard deviation of cycle time (sdCT), average tardiness (avgTARD), and total desirability (D) performances for the combination of Dispatching and NEXTS time policies with various decision policies are plotted

in Figure 7.1 and Figure 7.2 respectively. The actual figures used are available in Table H.1 and H.2 respectively (refer to Appendix H).

Figure 7.1: Performance for different methodology configuration (combination of the dispatching time policy with various decision policies) for wafer fab case study

Figure 7.2: Performance for different methodology configuration (combination of the NEXTS time policy with various decision policies) for wafer fab case study

The total desirability, D , in Figure 7.1 and Figure 7.2 are not comparable as they are computed using different bounds. However, comparison of $avgCT$, $sdCT$, and $avgTARD$ shows that the use of NEXTS time policy improved the performances by at least 40%, 62%, 97% respectively (see Table 7.4). The large observed percentage improvement arises due to the cumulative effect of incurring transportation delays (e.g. include retrieval from storage buffers, waiting for AGV, and the delivery from storage buffer to tool) that result in idle time at each tool for each lot that is processed. However, it should be noted that the magnitude of improvement does not translate directly to that of another fab. In addition, the magnitude of improvement will be reduced if the transportation time considered is less. The simulation results from this case study, however, are able to reaffirm that switching from dispatching to NEXTS time policy (in a fab where delays occur when a lot travels between tools/storages in its processing route) will result in an improvement in $avgCT$, $sdCT$, and $avgTARD$ performances.

Table 7.4: Percentage improvement

Decision Policies	avgCT	sdCT	avgTARD
FIFO	76.67	88.71	99.15
SPT	39.82	89.63	96.76
EDD	64.05	62.39	99.83
CR	64.73	73.22	99.98
LS	66.44	68.07	99.89
SSU	76.95	83.80	98.97
WINQ	70.22	89.89	99.05
MOFCS+CR	67.60	80.32	99.46

As can be observed in Figure 7.1, some total desirability values (D) are negative. Negative D values mean that one or more performance measures (i.e. $avgCT$, $sdCT$, and $avgTARD$) have exceeded their respective upper bounds used in the desirability

function computation. For the same dispatching time policy used, only SPT and MOFCS+CR decision policies are not dominated (refer to Table H.1 in Appendix H). The MOFCS+CR decision policy outperformed other methodologies by approximately 0.7% to 177% with respect to total desirability, D .

When the same NEXTS time policy is used, the following decision policies are not dominated (refer to Table H.2 in Appendix H):

- SPT,
- CR,
- LS,
- MOFCS+CR (1) with no weights updating,
- MOFCS+CR (2) with weights updated once, and
- MOFCS+CR (3) with weights updated twice are not dominated

There are no significant differences in the total desirability values (D) between the three MOFCS+CR decision policies and the MOFCS+CR decision policies outperformed the other decision policies in total desirability (D) by at least 7.9%. Despite the inclusion of weights updating into the simulation experiments, it is still not conclusive how much benefit is gained by updating the weights in real time as opposed to identifying a single set of weights that works well on average. More studies are required to identify the benefits.

It is important to note that there are many other dispatching rules that can be used in wafer fabs and the author does not deny that there may be better rules developed by other researchers. Despite so, in most cases, no single dispatching rule consistently dominates others in all situations. Hence, the goal of the proposed methodology is not to devise new dispatching rule(s) or come up with a fixed set of rules that

applies across all fabs as these rules are likely to perform differently under different settings and performance criteria. Rather, the decision policy dictates the need to take necessary steps to select rules (i.e. to base the final criterion/rule selection not only the criterion/rule's strength in optimizing an objective but also the trade-offs the criterion has with regards to the other objectives) for the problem under study before the rules are combined using the weighted aggregation method.

A larger percentage improvement is observed from the time policy switch than the decision policy switch. From the simulation results, the tandem application of the proposed NEXTS time policy (based on the fixed time period policy) and decision policy (MOFCS+CR rule with weights optimized using LASGAD) shows considerable promise for NEXTS methodology to be implemented to schedule tools in the wafer fabrication plant.

CHAPTER 8

CONCLUSION AND RECOMMENDATIONS

This chapter summarizes the findings and the limitations of this research. Recommendations for potential further research in this area are also proposed.

8.1 SUMMARY OF THE RESEARCH

The existing scheduling methodologies that could be applied in a dynamic and stochastic environment have been reviewed with the aid of the proposed taxonomy from the context of the time and decision policies. From the review, it is evident that the decision policy aspect of scheduling (how to make a scheduling decision) has been addressed much more frequently than the time policy aspect of scheduling (i.e. when to make a scheduling decision). The time policy component, though important from the simulation and modeling, and industrial application perspectives, is lacking in the literature. It should therefore be dealt with to some degree of detail in dispatching studies, motivating this research to divide the proposal of a methodology into two components (i.e. a time policy and a decision policy) to address the importance of both the components.

In this research, the NEXt Task Scheduling (NEXTS) methodology, made up of novel time and decision policies, is proposed. While NEXTS may seem to be just another approach for scheduling, NEXTS is designed to take transportation times, sequence-dependent setups, and user preferences into consideration for making

Pareto optimal scheduling decisions (i.e. selecting the next task based on a Pareto Control) at a machine level in an on-line, near-real-time approach such that the production uncertainties and disturbances are dealt with effectively and consistently.

The NEXTS time policy resembles the commonly used dispatching time policy in that it employs the decentralized approach to dispatch jobs when necessary. However, as opposed to making the next lot selection only when the machine is available, NEXTS adapts the scheduling/rescheduling time policy approach by scheduling the next lot selection to occur before the machine is available (but as close as possible to the time when the machine is available). The basic conceptual NEXTS time policy is shown to perform better than the existing periodic scheduling/rescheduling (PSR) and dispatching time policies in terms of average cycle time by at least 4.4% and 46%, respectively. The simulation results also show considerable promise for the NEXTS time policy to be used in a dynamic and stochastic environment as NEXTS is robust to arrival and processing time variations, and machine breakdowns.

The practicality of the NEXTS time policy is further extended (via a case study) to discrete machines in 300mm fully automated wafer fab that ‘pulls’ lots from various loadports and stockers for processing. The case study shows that the loadport-initiated time policy that is addressed in recent AMHS related studies is outperformed by both the fixed and adjusted time period policies. In fact, the fixed time period policy outperforms the loadport-initiated time policy by approximately 0.3% to 4.6% in terms of average cycle time across all the tested problem configurations. The general intuition of committing a decision (i.e. selection of next

lot) only at the latest possible time so that the decision has a much lower possibility of being invalidated by the stochastic events on the shop floor is proven wrong since the adjusted time period policy does not perform better than the fixed time period policy in terms of average cycle time.

This research does not claim that the optimum performances for all systems are achieved using the fixed time period policy as the constraints and data sets may be vastly different. It does, nevertheless, confirm that the appropriate timing has to be identified based on the concept of carrying out the next lot selection and delivery before and with practicable closeness to the time when the machine is available so that transportation and/or decision delays can be considered without causing the machine to become idle unnecessarily.

Evaluation and analysis of the recent studies show that the use of the weighted aggregation method is a realistic approach frequently applied in practical scheduling problems and will therefore be considered in the design of the NEXTS decision policy to address realistic problems with multiple conflicting objectives. A Markov process model is designed to identify if there are any potential benefits to use look upstream and look downstream rules. The Markov process model can only consider dispatching rules that involve machine and buffer states. Hence, this imposes a restriction on the complexity of rules that can be modeled. As such, it is unlikely for the Markov process model to represent the weighted rule and therefore, unable to predict the performances to guide the optimization of weights for the next lot selection.

Experimental results show that Look Downstream (LD) generally performs better than Look Upstream (LU) and Look into Queue (LQ) rules. One interesting observation is that Look Downstream at Machines' (LD_M) states and Look Downstream at both Buffers' and Machines' states (LD_BM) perform better (in terms of average cycle time) than Look Downstream at Buffers' (LD_B) states. This may suggest the possibility of a better performing rule (than existing rule such as the Work content In Next Queue rule) by incorporating the machines' states into the next lot selection. However, the observed difference between these rules is not sufficiently large enough to justify the design of a new rule that involves Look Downstream in the proposed decision policy. Instead, this research chooses to combine existing dispatching rules and focus on where the major benefits are believed to lie, that is, optimization of weights to achieve non-dominated user preferred solutions for on-line, near-real-time application.

The experimental study for the selection of the criteria to be combined using weighted aggregation has highlighted that basing the criterion/rule selection solely on the criterion/rule's strength in optimizing an objective is insufficient. The trade-offs the criterion has with respect to the other conflicting objectives are also important and should be considered. A correlation analysis of the objectives shows that one of the tardiness-related (either average tardiness or standard deviation of tardiness) is redundant and can be dropped. In addition, the weakly correlated and negatively correlated objectives are identified that reaffirms the presence of conflicting solutions in which tradeoff has to be dealt with.

The criterion selection study suggests that the NEXTS decision policy is able to consider multiple conflicting objectives of interest using the MOFCS rule (i.e.

weighted aggregation of the FIFO, CR, and SSU rules). Look ahead simulation-based genetic algorithm with desirability function (LASGAD) utilizes the desirability function to incorporate user preferences to guide the optimization of weights (towards the desired performances) used in the MOFCS rule automatically on-line in near real time to deal with the factory's changing condition. The optimized weights found are substituted into the MOFCS rule to make the next lot selection decision. In the evaluation of the NEXTS decision policy, sequence-dependent setups have also been considered. The NEXTS decision policy (LASGAD + MOFCS rule) is shown to have a better performance over existing decision policy (Response Surface Methodology, RSM + MOFCS rule) in both cases where user preferences (target values) are specified and where all criteria are to be minimized.

The NEXTS methodology, made up of the proposed NEXTS time policy (based on the best performing fixed time period policy) and the NEXTS decision policy (MOFCS rule with LASGAD), is evaluated using an industrial wafer fab case study. Better performance is observed when the NEXTS time policy and decision policy are applied in tandem than when they are applied individually, with a larger percentage improvement observed from the switch of the time policy from that of dispatching to NEXTS. The simulation results indicate that there is a need to address the 'when' aspects in conjunction with the 'how' aspects of selecting the next lot for achieving better fab performance.

In short, the main aim of this research, to devise a better scheduling methodology (made up of a time policy and a decision policy that specifies 'when' and 'how' to select the next task) to improve multiple performances of the wafer fab in terms of

(i) average cycle time, (ii) standard deviation of cycle time, (iii) average tardiness, and (iv) standard deviation of tardiness, has been achieved.

8.2 LIMITATIONS OF THE RESEARCH

This research presents an initial effort to investigate the use of improved time and decision policies for multiobjective scheduling in a dynamic environment. The limitations of the research are summarized as follows:

1. The study of the time and decision policies is limited to a single discrete machine. The time policy can be applied in the same manner whether there are one or many machines. However, the same cannot be said for the decision policy since different rules may have to be used to optimize the same performance objectives.
2. The implementation of NEXTS time policy may require more computational resources than the commonly used Dispatching time policy.
3. Transportation times and processing times are used in the proposed time policies to determine the time to invoke the transportation of the next lot to the machine. It is assumed that these data could be obtained from the shop floor information system in real time with reasonable accuracy. This may or may not be true. In order to retrieve on-line data in real time, it has to be carried out preferably without or with minimal blockage and the need to queue for data retrieval. Hence, the issue of data being up-to-date, and the rate of data retrieval and update need to be further addressed in its implementation.

4. As the GA used is a preference-based multi-objective optimization procedure, it transforms a multi-objective problem into a single-objective problem. Hence, only one optimum solution is generated. There is no issue if all criteria are to be optimized simultaneously. However, when user preferences have to be considered, it requires the decision maker to have knowledge of the Pareto surface or non-dominated solutions. If the decision maker specifies a dominated solution as the target solution (either because the Pareto surface has already shifted or due to a lack of knowledge), GA will find a solution that is close to the specified target. This solution is most likely going to be dominated.

8.3 DIRECTIONS FOR FUTURE RESEARCH

As with any new methods, continual improvement and refinement of the NEXTS methodology is essential and the proposed directions for future research can be summarized as:

1. Distributed and parallel simulation techniques, or grid computing, can be used to run multiple simulation runs in parallel. This can be used to shorten the GA runtime. When the runtime is shortened, the preference-based GA used in this research can be replaced by a Pareto-based GA such as NSGA-II (Deb et al., 2002). The population size can be increased and there will be no worries of not producing enough selection pressure for a small-sized population to move towards the user specified target. The use of Pareto-based approaches will maintain a population of non-dominated solutions. Hence, even though the decision maker specifies a dominated solution as the target solution (either because the Pareto surface has already shifted or

due to a lack of knowledge), one of the non-dominated solutions that is closest to the target will be picked.

2. Despite the inclusion of weights updating (performed only twice in the wafer fab case study due to the lengthy simulation run time involved) into the simulation experiments, it is still not conclusive how much benefit is gained by updating the weights in real time as opposed to identifying a single set of weights that works well on average. More studies applying the distributed and parallel simulation techniques can be used to identify the benefits.
3. The distributed and parallel simulation techniques should be used to reduce the run time so that frequent re-optimization (updating) of weights can be studied. This will help to evaluate the stability of the methodology (i.e. if the frequency of weight re-optimization will have any impact on stability) and to identify any drawbacks (if any).
4. In order to fully automate the decision-making process, autonomous learning or data mining approaches could be used to discover effective priority dispatching rules automatically.

REFERENCES

- Abumaizar, R. J., and Svestka, J. A., 1997, "Rescheduling job shops under random disruptions", *International Journal of Production Research*, Vol. 35, No. 7, pp. 2065-2082.
- Adachi, T., Moodie, C. L., and Talavage, J. J., 1988, "A pattern recognition-based method for controlling a multi-loop production system", *International Journal of Production Research*, Vol. 26, No. 12, pp. 1943-1957.
- Adams, J., Balas, E., and Zawack, D., 1988, "The shifting bottleneck procedure for job shop scheduling", *Management Science*, Vol. 34, No. 3, pp. 391-401.
- APF (Advanced Productivity FamilyTM) Technical Brief*, 2003, Brooks-PRI Automation, Inc.
- Allahverdi, A., Gupta, J. N. D., and Aldowaisan, T., 1999, "A review of scheduling research involving setup considerations", *Omega*, Vol. 27, No. 2, pp. 219-239.
- Ang, A. T. H., Sivakumar, A. I., 2007, "Online multiobjective single machine dynamic scheduling with sequence dependent setups using simulation-based genetic algorithm with desirability function", In *Proceedings of the 2007 Winter Simulation Conference (WSC2007)*, Eds. Henderson, S. G., Biller, B., Hsieh, M.-H., Shortle, J., Tew, J. D., and Barton, R. R., pp. 1828-1834.
- Aytug, H., and Koehler, G. J., 2000, "New stopping criterion for genetic algorithms", *European Journal of Operational Research*, Vol. 126, No. 3, pp. 662-674.
- Aytug, H., Lawley, M. A., McKay, K., Mohan, S., and Uzsoy, R., 2005, "Executing production schedules in the face of uncertainties: A review and some future directions", *European Journal of Operational Research*, Vol. 161, No. 1, pp. 86-110.
- Azadivar, F., and Tompkins, G., 1999, "Simulation optimization with qualitative variables and structural model changes: A genetic algorithm approach", *European Journal of Operational Research*, Vol. 113, No. 1, pp. 169-182.
- Bagchi, T. P., 1999, *Multiobjective Scheduling by Genetic Algorithms*, Boston : Kluwer Academic Publishers.
- Baker, A. D., 1998, "A survey of factory control algorithms that can be implemented in a multi-agent heterarchy: Dispatching, scheduling and pull", *Journal of Manufacturing Systems*, Vol. 17, No. 4, pp. 297-320.

- Baker, K. R., 1974, *Introduction to Sequencing and Scheduling*, New York, Wiley.
- Barua, A., Raghavan, N., Upasani, A., and Uzsoy, R., 2005, "Implementing global factory schedules in the face of stochastic disruptions", *International Journal of Production Research*, Vol. 43, No. 4, pp. 793-818.
- Baweja, G. S., and La, H. T., 2001, "Real-time lot dispatching for semiconductor fab", *Future Fab International – Manufacturing Productivity: Methods and Enablers*, Issue 11, Section 4.
- Bierwirth, C., and Mattfeld, D. C., 1999, "Production scheduling and rescheduling with genetic algorithms", *Evolutionary Computation*, Vol. 7. No. 1, pp. 1-17.
- Blackstone, J. H., Phillips, D. T., and Hogg, G. L., 1982, "A state-of-the-art survey of dispatching rules for job shop operations", *International Journal of Production Research*, Vol. 20, No. 1, pp. 27-45.
- Błażewicz, J., Ecker, K. H., Pesch, E., Schmidt, G., and Węglarz, J., 2001, *Scheduling Computer and Manufacturing Processes*, 2nd Ed., Berlin; New York : Springer.
- Bonvik, A. M., 1996, *Performance analysis of manufacturing systems under hybrid control policies*, Doctoral Dissertation, Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA.
- Braglia, M., and Petroni, A., 1999, "Data envelopment analysis for dispatching rule selection", *Production Planning & Control*, Vol. 10, No. 5, pp. 454-461.
- Branke, J., 2002, *Evolutionary Optimization in Dynamic Environments*, Boston : Kluwer Academic Publishers.
- Brown, D. E., and Scherer, W. T., 1995, *Intelligent Scheduling Systems*, Kluwer Academic Publishers, Norwell, Massachusetts.
- Cardon, A., Galinho, T., and Vacher, J.-P., 2000, "Genetic algorithms using multi-objectives in a multi-agent system", *Robotics and Autonomous Systems*, Vol. 33, No. 2, pp. 179-190.
- Chacon, G. R., 1998, "Using simulation to integrate scheduling with the manufacturing execution system", *Future Fab International*, pp. 63-66.
- Chang, J. Y. C., 2004, "Just-in-time AMHS delivery for 300mm FAB", In *Proceedings of Semiconductor Manufacturing Technology Workshop*, pp. 25-28.
- Chen, J. C., Chen, C.-W., Tai, C.-Y., and Tyan, J. C., 2004, "Dynamic state-dependent dispatching for wafer fabrication", *International Journal of Production Research*, Vol. 42, No. 21, pp. 4547-4562.

- Chen, Y.-J., Su, Y.-J., Hong, M.-S., Wang, I., 2000, "Real-time dispatching reduces cycle time", *Semiconductor International*.
- Church, L. K., and Uzsoy, R., 1992, "Analysis of periodic and event-driven rescheduling policies in dynamic shops", *International Journal of Computer Integrated Manufacturing*, Vol. 5, No. 3, pp. 153-163.
- Coello Coello, C. A., 1999, "An updated survey of evolutionary multiobjective optimization techniques: state of the art and future trends", In *Proceedings of the 1999 Congress on Evolutionary Computation*, Vol. 1, pp. 3-13.
- Coello Coello, C. A., 2000, "Handling preferences in evolutionary multiobjective optimization: a survey", In *Proceedings of the 2000 Congress on Evolutionary Computation*, Vol. 1, pp. 30-37.
- Coello Coello, C. A., 2005, "Recent trends in evolutionary multiobjective optimization", In *Evolutionary multiobjective optimization: theoretical advances and applications*, eds. Abraham, A., Jain, L., Goldberg, R., New York: Springer, pp. 7-32.
- Cornell, J. A., 1995, "Fitting models to data from mixture experiments containing order factors", *Journal of Quality Technology*, Vol. 27, No. 1, pp. 13-33.
- Cowling, P., and Johansson, M., 2002, "Using real time information for effective dynamic scheduling", *European Journal of Operational Research*, Vol.139, No. 2, pp.230-244.
- Cvetković, D., and Parmee, I. C., 2002, "Preferences and their application in evolutionary multiobjective optimization", *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 1, pp. 42-57.
- Dabbas, R., 1999, *A New Scheduling Approach Using Combined Dispatching Criteria in Semiconductor Manufacturing Systems*, Doctoral Dissertation, Department of Industrial Engineering, Arizona State University, Tempe.
- Dabbas, R., Chen, H.-N., Fowler, J. W., and Shunk, D., 2001, "A combined dispatching criteria approach to scheduling semiconductor manufacturing systems", *Computers and Industrial Engineering*, Vol. 39, No. 3, pp. 307-324.
- Dabbas, R., and Fowler, J. W., 2003, "A new scheduling approach using combined dispatching criteria in wafer fabs", *IEEE Transactions on Semiconductor Manufacturing*, Vol. 16, No. 3, pp. 501-510.
- Dabbas, R., Fowler, J. W., Rollier, D. A., McCarvilles, D., 2003, "Multiple response optimization using mixture-designed experiments and desirability functions in semiconductor scheduling", *International Journal of Production Research*, Vol. 41, No. 5, pp. 939-961.

- Dalal, M., Groel, B., and Frieditis, A., 2003, "Real-time decision making using simulation", In *Proceedings of the 2003 Winter Simulation Conference*, Eds. Chick, S., Sánchez, P. J., Ferrin, D., and Morrice, D. J., pp. 1456-1464.
- Deb, K., 2001, *Multi-Objective Optimization using Evolutionary Algorithms*, John Wiley & Sons.
- Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T., 2002, "A fast and elitist multiobjective genetic algorithm: NSGA-II", *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, pp. 182-197.
- Deb, K., and Sundar, J., 2006, "Reference point based multi-objective optimization evolutionary algorithms", In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation Conference (GECCO'06)*, pp. 635-642.
- Derringer, G., and Suich, R., 1980, "Simultaneous optimization of several response variables", *Journal of Quality Technology*, Vol. 12, No. 4, pp. 214-219.
- Dimopoulos, C., 2004, "A review of evolutionary multiobjective optimization applications in the area of production research", In *Congress on Evolutionary Computation (CEC2004)*, Vol. 2, pp. 1487-1494.
- DiSalvi, E. S., Liquigan, R., Peters, B. A., and Smith, J. S., 1995, "Development of an intelligent scheduler for semiconductor manufacturing facilities", In *Proceedings of 1995 IEEE/SEMI Advanced Semiconductor Manufacturing Conference and Workshop*, pp.159-165.
- Edgeworth, F. Y., 1881, *Mathematical Physics*, P. Keagan, London, England.
- Efstathiou, J., 1996, "Anytime heuristic schedule repair in manufacturing industry", In *IEE Proceedings of Control Theory and Applications*, Vol. 143, No. 2, pp. 114-124.
- Eshelman, L. J., and Schaffer, J. D., 1993, "Real-coded genetic algorithms and interval-schemata", In *Foundations of Genetic Algorithms 2 (FOGA-2)*, pp. 187-202.
- Fang, J., and Xi, Y., 1997, "A rolling horizon job shop rescheduling strategy in the dynamic environment", *The International Journal of Advanced Manufacturing Technology*, Vol. 13, No. 3, pp. 227-232.
- Fargher, H. E., Kilgore, M. A., Kline, P. J., and Smith, R. A., 1994, "A planner and scheduler for semiconductor manufacturing", *IEEE Transactions on Semiconductor Manufacturing*, Vol. 7, No. 2, pp. 117-126.
- Fowler, J. W., Carlyle, W. M., Runger, G. C., Gel, E. S., Mason, S. J., and Rose, O., 2001, "A new approach for scheduling semiconductor wafer fabs", *Semiconductor Fabtech*, Edition 15, pp. 39-41.

- French, S., 1982, *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*, New York, Wiley.
- Fuller, D., and Scherer, W., 1998, "The desirability function: Underlying assumptions and application implications", In *IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 4, pp. 4016-4021.
- Geiger, C. D., Uzsoy, R., and Aytug, H., 2006, "Rapid modeling and discovery of priority dispatching rules: An autonomous learning approach", *Journal of Scheduling*, Vol. 9, No. 1, pp. 7-34.
- Gershwin, S. B., 1994, *Manufacturing Systems Engineering*, Englewood Cliffs, New Jersey: PTR Prentice Hall.
- Goel, T., Vaidyanathan, R., Haftka, R. T., Shyy, W., Queipo, N. V., and Tucker, K., 2007, "Response surface approximation of Pareto optimal front in multi-objective optimization", *Computer Methods in Mechanics and Engineering*, Vol. 196, No. 4-6, pp. 879-893.
- Goldberg, D. E., 1989, *Genetic Algorithms in Search, Optimization & Machine Learning*, Reading, Mass. : Addison-Wesley Pub. Co.
- Goldberg, D. E., Deb, K., and Clark, J. H., 1992, "Genetic algorithms, noise, and the sizing of populations", *Complex Systems*, Vol. 6, No. 4, pp. 333-362.
- Goldratt, E., and Fox, R., 1986, *The Race*, North River Press, Great Barrington, MA.
- Graham, R. L., Lawler, E. L., Lenstra, J. K., and Rinnooy Kan, A. H. G., 1979, "Optimization and approximation in deterministic sequencing and scheduling: A survey", *Annals of Discrete Mathematics*, Vol. 5, pp. 287-326.
- Gupta, A. K., and Sivakumar, A. I., 2005a, "Single machine scheduling with multiple objectives in semiconductor manufacturing", *International Journal of Advanced Manufacturing Technology*, Vol. 26, No. 9-10, pp. 950-958.
- Gupta, A. K., and Sivakumar, A. I., 2005b, "Multi-objective scheduling of two-job families on a single machine", *Omega*, Vol. 33, No. 5, pp. 399-405.
- Gupta, A. K., 2006, *Multiobjective Dynamic Scheduling of Serial and Batch Processes in Semiconductor Manufacturing*, Doctoral Dissertation, Department of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore.
- Hapke, M., Jaskiewicz, A., and Kurowski, K., 2002, "Multi-objective genetic local search methods for the flow shop problem", In *Advances in Nature-Inspired Computation: The PPSN VII Workshops, PEDAL*, University of Reading, UK, pp. 22-23.

- Harrington Jr., E. C., 1965, "The desirability function", *Industrial Quality Control*, Vol. 21, No. 10, pp. 494-498.
- Hart, E., Ross, P., and Corne, D., 2005, "Evolutionary scheduling: a review", *Genetic Programming and Evolvable Machines*, Vol. 6, No. 2, pp. 191-220.
- Harvey, I., 1995, *The Artificial Evolution of Adaptive Behaviour*, Doctoral Dissertation, School of Cognitive and Computing Sciences, University of Sussex, Revised Copy.
- Hashimoto, C., Takeda, T., Tazawa, S., and Sakurai, T., 1993, "Practical scheduling and line optimization technology for ASIC manufacturing lines", *IEEE Transactions on Components, Hybrids, and Manufacturing Technology*, Vol. 16, No. 4, pp. 407-411.
- Haupt, R., 1989, "A survey of priority rule-based scheduling", *OR Spectrum*, Vol. 11, No. 1, pp. 3-16.
- Herrera, F., Lozano, M., and Verdegay, J. L., 1998, "Tackling real-coded genetic algorithms: operators and tools for behavioural analysis", *Artificial Intelligence Review*, Vol. 12, No. 4, pp. 265-319.
- Herrmann, J. W., Lee, C.-Y., and Hinchman, J., 1995, "Global job shop scheduling with a genetic algorithm", *Production and Operations Management*, Vol. 4, No. 1, pp. 30-45.
- Hitomi, K., 1996, *Manufacturing Systems Engineering: A unified approach to manufacturing technology, production management, and industrial economics*, 2nd Ed., London : Taylor and Francis.
- Holthaus, O., and Ziegler, H., 1997, "Look ahead job demanding for improving job shop performance", *OR Spectrum*, Vol. 19, No. 1, pp. 23-29.
- Hooke, R., and Jeeves, T. A., 1962, "Direct search solution of numerical and statistical problems", *Journal of the Association of Computing Machinery*, Vol. 8, No. 2, pp. 212-229.
- Horn, J., 1997, "Multicriterion decision making", Eds. Bäck, Thomas, Fogel, D., and Michalewicz, Z., In *Handbook of Evolutionary Computation*, Vol. 1, pp. F1.9:1 – F1.9:15, IOP Publishing Ltd. and Oxford University Press.
- Hwang, C. L., Paidy, S. R., Yoon, K., and Masud, A. S. M., 1980, "Mathematical programming with multiple objectives: A tutorial", *Computers & Operations Research*, Vol. 7, No. 1-2, pp. 5-31.
- Ishii, N., Talavage, J. J., 1991, "A transient based real time scheduling algorithm in FMS", *International Journal of Production Research*, Vol. 29, No. 12, pp. 2501-2520.

- Iverson, M., and Özgüner, F., 1999, "Dynamic, competitive scheduling of multiple DAGs in a distributed heterogeneous environment", *Distributed Systems Engineering*, Vol. 6, No. 3, pp. 112-120.
- Jain, A. K., and ElMaraghy, 1997, "Production scheduling/rescheduling in flexible manufacturing", *International Journal of Production Research*, Vol. 35, No. 1, pp. 281-309.
- Jain, A. S., and Meeran, S., 1999, "Deterministic job-shop scheduling: Past, present and future", *European Journal of Operational Research*, Vol. 113, No. 2, pp. 390-434.
- Jang, J., Suh, J., Park, N., and Liu, R., 2001, "A look-ahead routing procedure for machine selection in a highly informative manufacturing system", *International Journal of Flexible Manufacturing Systems*, Vol. 13, No. 3, pp. 287-308.
- Jayamohan, M. S., and Rajendran, C., 2000, "A comparative analysis of two different approaches to scheduling in flexible flow shops", *Production Planning & Control*, Vol. 11, No. 6, pp. 572-580.
- Johri, P. K., 1993, "Practical issues in scheduling and dispatching in semiconductor wafer fabrication", *Journal of Manufacturing Systems*, Vol. 12, No. 6, pp. 474-486.
- Jones, A., and Rabelo, L. C., 1998, *Survey of Job Shop Scheduling Techniques*, NISTIR, National Institute of Standards and Technology, Gaithersburg, MD.
- Kempf, K. G., 1994, "Intelligently scheduling semiconductor wafer fabrication", In: *Intelligent Scheduling*, Eds. Zweben, M., and Fox, M. S., San Francisco, California, Morgan Kaufmann Publishers, Chpt. 18, pp. 517-544.
- Kim, J. T., 2003, "Fitness costs of mutation rate adaptation and its application to optimization of dynamic objective functions", Eds. Dosch, W., and Lee, R. Y., In *Proceedings of the ACIS Fourth International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'03)*, Schmidt Römhild, Lübeck, Germany, pp. 68-74.
- Kim, S. C., and Bobrowski, P. M., 1997, "Scheduling jobs with uncertain setup times and sequence dependency", *Omega*, Vol. 25, No. 4, pp. 437-447.
- Kim, Y.-D., Shim, S.-O., Choi, B. and Hwang, H., 2003, "Simplification methods for accelerating simulation-based real-time scheduling in a semiconductor wafer fabrication facility", *IEEE Transactions on Semiconductor Manufacturing*, Vol. 16, No. 2, pp. 290-298.

- Kondo, H., and Harada, M., 2005, "Study for realizing effective direct tool-to-tool delivery", In *IEEE International Symposium on Semiconductor Manufacturing (ISSM 2005)*, pp. 21-24.
- Krajewski, L. J., King, B. E., Ritzman, L. P., and Wong, D. S., 1987, "Kanban, MRP, and shaping the manufacturing environment", *Management Science*, Vol. 33, No. 1, pp. 39-57.
- Krishnaswamy, S., and Nettles, S., 2005, "Scheduling challenges for 300mm manufacturing", In *Proceedings of the International Conference on Modeling and Analysis of Semiconductor Manufacturing (MASM 2005)*, pp. 231-241.
- Kumar, P. R., 1994, "Scheduling semiconductor manufacturing plants", *IEEE Control Systems Magazine*, Vol. 14, No. 6, pp. 33-40.
- Law, A. M., 2007, *Simulation modeling and analysis*, 4th Ed., Boston: McGraw-Hill.
- Lawrence, S. R., and Sewell, E. C., 1997, "Heuristic, optimal, static, and dynamic schedules when processing times are uncertain", *Journal of Operations Management*, Vol. 15, No. 1, pp. 71-82.
- Lee, C.-Y., Piramuthu, S., and Tsai, Y.-K., 1997, "Job shop scheduling with a genetic algorithm and machine learning", *International Journal of Production Research*, Vol. 35, No. 4, pp. 1171-1191.
- Lenstra, J. K., Rinnooy Kan, A. H. G., and Brucker, P., 1977, "Complexity of machine scheduling problems", *Annals of Discrete Mathematics*, Vol. 1, pp. 343-362.
- Li, S., Tang, T., and Collins, D. W., 1996, "Minimum inventory variability schedule with application in semiconductor fabrication", *IEEE Transactions on Semiconductor Manufacturing*, Vol. 9, No. 1, pp. 145-149.
- Li, X., and Olafsson, S., 2005, "Discovering dispatching rules using data mining", *Journal of Scheduling*, Vol. 8, No. 6, pp. 515-527.
- Liao, D.-Y., Chang, S.-C., Pei, K.-W., and Chang, C.-M., 1996, "Daily scheduling for R&D semiconductor fabrication", *IEEE Transactions on Semiconductor Manufacturing*, Vol. 9, No. 4, pp. 550-561.
- Lim, M. K., and Zhang, Z., 2003, "A multi-agent based manufacturing control strategy for responsive manufacturing", *Journal of Materials Processing Technology*, Vol. 139, No. 1, pp. 379-384.
- Lin, G. Y.-J., and Solberg, J. J., 1992, "Integrated shop-floor control using autonomous agents", *IEEE Transactions*, Vol. 24, No. 3, pp. 57-71.

- Lin, J. T., Wang, F. K., and Kuo, P. C., 2005, "A parameterized-dispatching rule for a Logic IC sort in a wafer fabrication", *Production Planning & Control*, Vol. 16, No. 5, pp. 426-436.
- Lin, J. T., Wang, F. K., and Young, J. R., 2004, "Virtual vehicle in the connecting transport automated material-handling system (AMHS)", *International Journal of Production Research*, Vol. 42, No. 13, pp. 2599-2610.
- Little, J. D. C., 1961, "A proof for the queueing formula: $L=\lambda W$ ", *Operations Research*, Vol. 9, pp. 383-387.
- Lu, S. C. H., Ramaswamy, D., and Kumar, P. R., 1994, "Efficient scheduling policies to reduce mean and variance of cycle-time in semiconductor manufacturing plants", *IEEE Transactions Semiconductor Manufacturing*, Vol. 7, No. 3, pp. 374-388.
- Lyman, J., 2002, "Innovative scheduling techniques speed chip fabrication", *NewsFactor Network, NewsFactor SciTech – Tech Innovation & Discovery*, July 30.
- Mao, W., and Kincaid, R. K., 1994, "A look-ahead heuristic for scheduling jobs with release dates on a single machine", *Computers and Operations Research*, Vol. 21, No. 10, pp. 1041-1050.
- Martinelli, F., and Valigi, P., 1998, "A dynamic control problem for a two-part type pull manufacturing system", In *Proceedings of the 1998 IEEE International Conference on Robotics and Automation*, pp. 2753-2758.
- Matsuura, H., Tsubone, H., and Kanezashi, M., 1993, "Sequencing, dispatching, and switching in a dynamic manufacturing environment", *International Journal of Production Research*, Vol. 31, No. 7, pp. 1671-1688.
- Mehta, S. V., and Uzsoy, R. M., 1998, "Predictable scheduling of a job shop subject to breakdowns", *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 3, pp. 365-378.
- Mehta, S. V., and Uzsoy, R. M., 1999, "Predictable scheduling of a single machine subject to breakdowns", *International Journal of Computer-Integrated Manufacturing*, Vol. 12, No. 1, pp. 15-38.
- Michalewicz, Z., 1992, *Genetic Algorithms + Data Structures = Evolution Programs*, Berlin: Springer-Verlag.
- Michalewicz, Z., and Fogel, D. B., 2004, *How to Solve It: Modern Heuristics*, 2nd ed., New York: Springer.
- Miettinen, K., 1999, *Nonlinear Multiobjective Optimization*, Kluwer, Boston.

- Min, H. S., and Yih, Y., 2003a, "Development of a real-time multi-objective scheduler for a semiconductor fabrication system", *International Journal of Production Research*, Vol. 41, No. 10, pp. 2345-2364.
- Min, H. S., and Yih, Y., 2003b, "Selection of dispatching rules on multiple dispatching decision points in real-time scheduling of a semiconductor wafer fabrication system", *International Journal of Production Research*, Vol. 41, No. 16, pp. 3921-3941.
- Mohd Azizi, C., Ung, T. T., Thangappan, K., and Lim, K. W., 2005, "Cycle time reduction through event based management approach in 300mm wafer foundry", In *3rd International SEMATECH Manufacturing Initiative (ISMI) Symposium on Manufacturing Effectiveness*, Accessible online at: <http://ismi.sematech.org/ismisymposium/past/III/abstracts/08982006345.pdf>
- Morton, T. E., and Pentico, D. W., 1993, *Heuristic Scheduling Systems with Applications to Production Systems and Project Management*, New York, Wiley.
- Myers, R. H., and Montgomery, D. C., 1995, *Response Surface Methodology: Process and Product Optimization using Designed Experiments*, New York: Wiley.
- O'Donovan, R., Uzsoy, R., and McKay, K. N., 1999, "Predictable scheduling of a single machine with breakdowns and sensitive jobs", *International Journal of Production Research*, Vol. 37, No. 18, pp. 4217-4233.
- Onwubolu, G. C., 2002, *Emerging Optimization Techniques in Production Planning and Control*, London, Imperial College Press.
- Padmanabhan, D., 2005, "Techniques for construction of Pareto frontier approximations", *46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials Conference*, AIAA 2005-1817.
- Panwalker, S., and Iskander, W., 1977, "A survey of scheduling rules", *Operations Research*, Vol. 25, No. 1, pp. 45-61.
- Papadopoulos, H. T., and Heavey, C., 1996, "Queueing theory in manufacturing systems analysis and design: A classification of models for production and transfer lines", *European Journal of Operational Research*, Vol. 92, No. 1, pp. 1-27.
- Pareto, V., 1896, *Cours D'Economie Politique*, Vol. I and II, F. Rouge, Lausanne.
- Pareto, V., 1966, *Sociological Writings*, Ed. Finer, S. E., Oxford: B. Blackwell.
- Park, T., Lee, H., and Lee, H., 2001, "FMS design model with multiple objectives using compromise programming", *International Journal of Production Research*, Vol. 39, No. 15, pp. 3513-3528.

- Parunak, H. V. D., 1991, "Characterizing the manufacturing scheduling problem", *Journal of Manufacturing Systems*, Vol. 10, No. 3, pp. 241-258.
- Pasandideh, S. H. R., and Niaki, S. T. A., 2006, "Multi-response simulation optimization using genetic algorithm within desirability function framework", *Applied Mathematics and Computation*, Vol. 175, No. 1, pp. 366-382.
- Pierreval, H., and Mebarki, N., 1997, "Dynamic selection of dispatching rules for manufacturing system scheduling", *International Journal of Production Research*, Vol. 35, No. 6, pp. 1575-1591.
- Pinedo, M., and Chao, X., 1999, *Operations Scheduling with Applications in Manufacturing and Services*, London, Irwin/McGraw-Hill.
- Pinedo, M., 2002, *Scheduling: Theory, Algorithms, and Systems*, 2nd Ed., Upper Saddle, N.J., Prentice Hall.
- Pongcharoen, P., Hicks, C., Braiden, P.M., and Stewardson, D.J., 2002, "Determining optimum Genetic Algorithm parameters for scheduling the manufacturing and assembly of complex products", *International Journal of Production Economics*, Vol. 78, No. 3, pp. 311-322.
- Pyke, D. F., and Cohen, M. A., 1990, "Push and pull in manufacturing and distribution systems", *Journal of Operations Management*, Vol. 9, No. 1, pp. 24-43.
- Qi, C., 2005, *Closed-Loop Job Release Based on WIPLOAD Control in Semiconductor Wafer Fabrication*, Doctoral Dissertation, Department of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore.
- Qi, C., and Sivakumar, A. I., 2004, "Simulation based cause and effect analysis in semiconductor wafer fabrication", *Journal of The Institution of Engineers*, Singapore, Vol. 44, No. 4, pp. 53-65.
- Ran, W., and Cheng, W., 1996, "An on-line scheduler for semiconductor fabrications", In *IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 1, pp.238-243.
- Rippenhagen, C., and Krishnaswamy, S., 1998, "Implementing the theory of constraints philosophy in highly reentrant systems", Eds. Medeiros, D.J., Watson, E.F., Carson, J.S., and Manivannan, M.S., In *Proceedings of 1998 Winter Simulation Conference*, pp. 993-996.
- Robinson, J. K., 2002, "Understanding and improving wafer fab cycle times", *Semiconductor FabTech*, Vol. 17,
<http://www.fabtime.com/files/FabTech_2002.pdf>.

- Robinson, J. K., Fowler, J. W., and Bard J., 1995, "The use of upstream and downstream information in scheduling semiconductor batch operations", *International Journal of Production Research*, Vol. 33, No. 7, pp. 1849-1870.
- Rossi, A., and Dini, G., 2000, "Dynamic scheduling of FMS using a real-time genetic algorithm", *International Journal of Production Research*, Vol. 38, No. 1, pp. 1-20.
- Roundy, R. O., Maxwell, W. L., Herer, Y. T., Tayur, S. R., and Getzler, A. W., 1991, "A price directed approach to real-time scheduling of production operations", *IIE Transactions*, Vol. 23, No. 2, pp. 149-160.
- Sabuncuoglu, I., and Karabuk, S., 1999, "Rescheduling frequency in an FMS with uncertain processing times and unreliable machines", *Journal of Manufacturing Systems*, Vol. 18, No. 4, pp. 268-283.
- Sabuncuoglu, I., and Kizilisik, O. B., 2003, "Reactive scheduling in a dynamic and stochastic FMS environment", *International Journal of Production Research*, Vol. 41, No. 17, pp. 4211-4231.
- Safe, M., Carballido, J., Ponzoni, I., and Brignole, N., 2004, "On stopping criteria for genetic algorithms", Eds. Bazzan, A. L. C., and Labidi, S., *Advances in Artificial Intelligence – SBIA 2004 (Lecture Notes in Computer Science)*, Springer-Verlag Berlin Heidelberg, Vol. 3171, pp. 405-413.
- Sawaragi, Y., Nakayama, H., and Tanino, T., 1985, *Theory of Multiobjective Optimization*, Academic Press, Orlando, FL.
- Schorn-DiSalvi, E., Liquigan, R., Peters, B.A., and Smith, J.S., 1995, "Development of an intelligent scheduler for semiconductor manufacturing facilities", In *IEEE/SEMI Proceedings of Advanced Semiconductor Manufacturing Conference and Workshop (ASMC95)*, pp. 159-165.
- Schrageheim, E., and Ronen, B., 1990, "Drum-buffer-rope shop floor control", *Production and Inventory Management Journal*, Vol. 31, No. 3, pp. 18-22.
- Shafaei, R., and Brunn, P., 1999a, "Workshop scheduling using practical (inaccurate) data Part 1: The performance of heuristic scheduling rules in a dynamic job shop environment using a rolling time horizon approach", *International Journal of Production Research*, Vol. 37, No. 17, pp. 3913-3925.
- Shafaei, R., and Brunn, P., 1999b, "Workshop scheduling using practical (inaccurate) data Part 2: An investigation of the robustness of scheduling rules in a dynamic and stochastic environment", *International Journal of Production Research*, Vol. 37, No. 18, pp. 4105-4117.

- Shafaei, R., and Brunn, P., 2000, "Workshop scheduling using practical (inaccurate) data Part 3: A framework to integrate job releasing, routing, and scheduling functions to create a robust predictive schedule", *International Journal of Production Research*, Vol. 38, No. 1, pp. 85-99.
- Shen, W., 2002, "Distributed manufacturing scheduling using intelligent agents", *IEEE Intelligent Systems*, Vol. 17, No. 1, pp. 88-94.
- Shikalgar, S. T., Fronckowiak, D., and MacNair, E. A., 2002, "300mm wafer fabrication line simulation model", In *Proceedings of the 2002 Winter Simulation Conference*, Eds. Yücesan, E., Chen, C.-H., Snowdon, J. L., and Charnes, J. M., pp. 1365-1368.
- Sivakumar, A. I., 1999, "Optimization of cycle time and utilization in semiconductor test manufacturing using simulation based, on-line, near-real-time scheduling system", In *Proceedings of the 1999 Winter Simulation Conference*, Eds. Farrington, P. A., Nembhard, H. B., Sturrock, D. T., and Evans, G. W., pp. 727-735.
- Sivakumar, A. I., 2001, "Multiobjective dynamic scheduling using discrete event simulation", *International Journal of Computer Integrated Manufacturing*, Vol. 14, No. 2, pp. 154-167.
- Sivakumar, A. I., and Chong, C. S., 2000, "Development of an object oriented simulation engine for on-line simulation and optimization", In *Proceedings of the 33rd Annual Simulation Symposium (SS2000)*, pp. 291-298.
- Sivakumar, A. I., Qi, C., Tan, H. K., and Lieu, S. K., 2003, "Job release control in semiconductor wafer fabrication", *Industrial Engineering & Management Systems*, Vol. 2, No. 2, pp. 169-177.
- Sivakumar, A. I., and Gupta, A. K., 2006, "On-line multiobjective Pareto optimal dynamic scheduling of semiconductor back-end using conjunctive simulated scheduling", *IEEE Transactions on Electronics Packaging Manufacturing*, Vol. 29, No. 2, pp. 99-109.
- Smith, S. F., 1994, "OPIS: A methodology and architecture for reactive scheduling", In: *Intelligent Scheduling*, Eds. Zweben, M., and Fox, M. S., San Francisco, California, Morgan Kaufmann Publishers, Chpt 2, pp. 29-66.
- Smith, S. F., 2003, "Is scheduling a solved problem?", *Proceedings of First Multi-Disciplinary International Conference on Scheduling Theory and Applications (MISTA)*, Nottingham, UK.
- Sohal, A. S., Keller, A.Z., and Fouad, R. H., 1989, "A review of literature relating to JIT", *International Journal of Operations and Production Management*, Vol. 9, No. 3, pp. 15-25.

- Spearman, M. L., Woodruff, D. L., and Hopp, W. J., 1990, "CONWIP: A pull alternative to Kanban", *International Journal of Production Research*, Vol. 28, No. 5, pp. 879-894.
- Stankovic, J., 1988, "Misconceptions about real time computing: A serious problem for next generation systems", *IEEE Computer*, Vol. 21, No. 10, pp. 10-19.
- Steuer, D., 1999, "Multi-criteria-optimisation and desirability indices", Technical Report 20, Universität Dortmund, FB Statistik, Vogelpothsweg 87, 44221 Dortmund.
<<http://citeseer.ist.psu.edu/cache/papers/cs/6865/http:zSzzSzwww.statistik.uni-dortmund.dezSzsfb475zSzberichteSztr20-99.pdf/steuer99multicriteriaoptimisation.pdf>>
- Stoop, P. P. M., and Wiers, V. C. S., 1996, "The complexity of scheduling in practice", *International Journal of Operations & Production Management*, Vol. 16, No. 10, pp. 37-53.
- Subbiah, V., and Bodenstab, F., 1992, "Application of real-time expert systems in semiconductor manufacturing", In *Proceedings of the 1992 IEEE/SEMI Advanced Semiconductor Manufacturing Conference and Workshop*, pp. 90-95.
- Suresh, V., and Chaudhuri, D., 1993, "Dynamic scheduling – A survey of research", *International Journal of Production Economics*, Vol. 32, No. 1, pp. 53-63.
- Tabucanon, M. T., 1988, *Multiple Criteria Decision Making in Industry*, Amsterdam (The Netherlands); New York: Elsevier.
- Tan, B., 2002, "State-space modeling and analysis of pull-controlled production systems", In *Analysis and Modeling of Manufacturing Systems*, Eds. Gershwin, S. B., Dallery, Y., Papadopoulos, C. T., and Smith, J. M., Boston: Kluwer Academic Publishers.
- Tumlinson, J., and Muzumdar, N., 1999, "Extending APF/RTD beyond MES data and lot dispatch", In *AutoSimulations Symposium '99 – Driving the Golden Spike*.
- Tyan, J. C., Chen, J. C., and Wang F.-K., 2002, "Development of a state-dependent dispatch rule using theory of constraints in near-real-world wafer fabrication", *Production Planning & Control*, Vol. 13, No. 3, pp. 253-261.
- Tyan, J. C., Du, T. C., Chen, J. C., and Chang, I.-H., 2004, "Multiple response optimization in a fully automated FAB: an integrated tool and vehicle dispatching strategy", *Computers & Industrial Engineering*, Vol. 46, No. 1, pp. 121-139.

- Upasani, A. A., Uzsoy, R., and Sourirajan, K., 2006, "A problem reduction approach for scheduling semiconductor wafer fabrication facilities", *IEEE Transactions on Semiconductor Manufacturing*, Vol. 19, No. 2, pp. 216-225.
- Vieira, G. E., Herrmann, J. W., and Lin, E., 2000a, "Analytical models to predict the performance of a single-machine system under periodic and event-driven rescheduling strategies", *International Journal of Production Research*, Vol. 38, No. 8, pp. 1899-1915.
- Vieira, G. E., Herrmann, J. W., and Lin, E., 2000b, "Predicting the performance of rescheduling strategies for parallel machine system", *Journal of Manufacturing Systems*, Vol. 19, No. 3, pp. 256-266.
- Vieira, G. E., Herrmann, J. W., and Lin, E., 2003, "Rescheduling manufacturing systems: A framework of strategies, policies, and methods", *Journal of Scheduling*, Vol. 6, No. 1, pp. 39-62.
- Viswanadham, N., and Narahari, Y., 1992, *Performance Modeling of Automated Manufacturing Systems*, Englewood Cliffs, New Jersey: Prentice-Hall, Inc.
- Wakabayashi, T., Watanabe, S., Kobayashi, Y., Okabe, T., and Koike, A., 2004, "High-speed AMHS and its operation method for 300-mm QTAT fab", *IEEE Transactions of Semiconductor Manufacturing*, Vol. 17, No. 3, pp. 317-323.
- Wall, M., 1996, *GAlib: A C++ Library of Genetic Algorithm Components (User Guide)*, Version 2.4, Revision B, Mechanical Engineering Department, Massachusetts Institute of Technology, <http://lancet.mit.edu/ga/>.
- Walpole, R. E., Myers, R. H., Myers, S. L., and Ye, K., 2002, *Probability & Statistics for Engineers and Scientists*, 7th Ed., Prentice Hall, New Jersey.
- Wang, K.-J., Chen, J. C., and Lin, Y.-S., 2005, "A hybrid knowledge discovery model using decision tree and neural network for selecting dispatching rules of a semiconductor final testing factory", *Production Planning & Control*, Vol. 16, No. 7, pp. 665-680.
- Watt, D. G., 1998, "Integrating simulation based scheduling with MES in a semiconductor fab", In *Proceedings of the 1998 Winter Simulation Conference*, Eds. Medeiros, D. J., Watson, E. F., Carson, J. S., and Manivannan, M. S., pp. 1713-1715.
- Watts, H., 2000, "Improving fab performance", *Future Fab International*, Vol. 9, http://www.future-fab.com/documents.asp?grID=212&d_ID=1191.
- Wein, L. M., 1988, "Scheduling semiconductor wafer fabrication", *IEEE Transactions on Semiconductor Manufacturing*, Vol. 1, No. 3, pp. 115-130.

- Weitzman, R., and Rabinowitz, G., 2003, "Sensitivity of 'push' and 'pull' strategies to information updating rate", *International Journal of Production Research*, Vol. 41, No. 9, pp. 2057-2074.
- Weng, M., and Ren, H., 2006, "An efficient priority rule for scheduling job shops to minimize mean tardiness", *IIE Transactions*, Vol. 38, No. 9, pp. 789-795.
- Williams, L., Novakovich, S., and Davis, J., 2000, "Real time dispatcher replaces legacy dispatching system in Texas Instruments' DMOS5 wafer fab", In *AutoSimulations Symposium 2000 – Discovering New Frontiers*.
- Wood, S. C., 2000, "Factory modeling", In *Handbook of Semiconductor Manufacturing Technology*, Eds. Nishi Y., Doering, R., Marcel Dekker, Inc., New York, Basel, pp. 1103-1122.
- Wright, R., Shopbell, M., Rust, K., and Sigireddy, S., 2002, "Effects of metrology load port buffering in automated 300mm factories", Eds. Yücesan, E., Chen, C.-H., Snowdon, J. L., and Charnes, J. M., In *Proceedings of the 2002 Winter Simulation Conference*, pp. 1359-1364.
- Wu, S.-Y.D., and Wysk, R.A., 1989, "An application of discrete-event simulation to on-line control and scheduling in flexible manufacturing", *International Journal of Production Research*, Vol. 27, No. 9, pp. 1603-1623.
- Xue, D., Sun, J., and Norrie, D. H., 2001, "An intelligent optimal production scheduling approach using constraint-based search and agent-based collaboration", *Computers in Industry*, Vol. 46, No. 2, pp. 209-231.
- Yamamoto, M., and Nof, S. Y., 1985, "Scheduling/rescheduling in the manufacturing operating system environment", *International Journal of Production Research*, Vol. 23, No. 4, pp. 705-722.
- Yang, H.-H., and Wu, Z.-M., 2003, "The application of adaptive genetic algorithms in FMS dynamic rescheduling", *International Journal of Computer Integrated Manufacturing*, Vol. 16, No. 6, pp. 382-397.
- Yang, T., Kuo, Y., and Cho, C., 2007, "A genetic algorithms simulation approach for the multi-attribute combinatorial dispatching decision problem", *European Journal of Operational Research*, Vol. 176, No. 3, pp. 1859-1873.
- Yang, W.-H., and Liao, C.-J., 1999, "Survey of scheduling research involving setup times", *International Journal of Systems Science*, Vol. 30, No. 2, pp. 143-155.
- Yellig, E. J., and Mackulak, G. T., 1997, "Robust deterministic scheduling in stochastic environments: The method of capacity hedge points", *International Journal of Production Research*, Vol. 35, No. 2, pp. 369-379.

- Yoshida, T., and Touzaki, H., 1999, "A study on association among dispatching rules in manufacturing scheduling problems", In *Proceedings of 7th IEEE International Conference on Emerging Technologies and Factory Automation*, Vol. 2, pp. 1355-1360.
- Yurtsever, T., and Pierce, N. G., 1998, "Computerized manufacturing monitoring and dispatch system", *Computers Industrial Engineering*, Vol. 35, No. 1-2, pp. 137-140.
- Zhu, X, and Wilhelm, W. E., 2006, "Scheduling and lot sizing with sequence-dependent setup: A literature review", *IIE Transactions*, Vol. 38, No. 11, pp. 987-1007.
- Zweiben, M., and Fox, M. S., 1994, *Intelligent Scheduling*, Morgan Kaufmann Publishers, San Francisco.

APPENDICES

APPENDIX A

DEFINITION OF SOME TERMINOLOGY USED

Bottleneck	Bottleneck is defined as any entity whose capacity is less than the demand placed on it (Baweja and La, 2001).
Completely Reactive Scheduling (CRS) or Dynamic Scheduling	CRS strategy is used to classify scheduling methodologies that do not produce schedules in which the jobs are scheduled to process in advance but uses decentralized production control methods to dispatch jobs when necessary (Vieira et al., 2003).
Cycle Time (also called Throughput Time)	Cycle time refers to the time it takes for a job to go from start to finish (this time includes processing time, transportation time and the time spent waiting in queue).
Dispatching Decision Policy	Dispatching decision policy refers to how to select the next job (from those jobs waiting in queue) for processing at a machine. e.g. Dispatching rules
Dispatching Time Policy	Dispatching time policy refers to when to select the next job (from those jobs waiting in queue) for processing at a machine. e.g. when machine is available
Hybrid Reactive Scheduling (HRS) Strategy	HRS strategy is put forth to accommodate methodologies that can neither be appropriately classified as absolutely a PRS nor CRS strategy. HRS strategy is used to classify scheduling methodologies that are either (i) a hybrid of both CRS and PRS (e.g. switching between scheduling and dispatching, such as that studied by Matsuura et al. (1993)); or (ii) different from PRS and CRS but yet bare some resemblance to the partial aspects of each predictive-reactive and dynamic scheduling (e.g. the NEXTS methodology proposed in this research).
Lateness	Lateness is defined as the difference between the completion time and the due date of a job. Lateness of a job is positive when a job is completed late and is negative when it is completed early (Pinedo, 2002). <i>Lateness = Completion Time – Due Date</i>
Machine Utilization	Utilization refers to a fraction of time that a machine is processing work. The actual meaning of utilization is vague, unless the activities that constitute “processing” and the time units are defined (Wood, 2000).
Makespan	The makespan is defined as the completion time of the last job to leave the system. A minimum makespan usually implies a high utilization of the machine(s) (Pinedo, 2002).
Next Task Scheduling (NEXTS) Decision Policy	Made up of rules and algorithms that are able to consider the complexities of the fab and incorporate on-line and look-ahead data for making informed next task scheduling decisions for each machine in order to optimize multiple conflicting performance measures (e.g. average and standard deviation of

	cycle time, average and standard deviation of tardiness) in accordance to user specified preference(s).
Next Task Scheduling (NEXTS) Time Policy	Specifies the timing of the execution of the scheduling logic and algorithm that allows the next task to be determined and executed on each machine on-line, in near-real-time to consider transportation and decision delays and to deal with the unavoidable uncertainties and disturbances effectively and consistently in a highly dynamic and stochastic environment.
Predictive-Reactive Scheduling (PRS) Strategy	PRS strategy is used to classify scheduling methodologies that involves a two-step process of scheduling (generating an initial schedule and when necessary, rescheduling to generate a new schedule or to update the initial schedule) (Vieira et al., 2003; Aytug et al., 2003).
Queue Time	The amount of time spent waiting in queue.
Scheduling/Rescheduling Decision Policy	Scheduling/rescheduling decision policy refers to how to generate a new schedule and/or update the initial schedule (in short, how to schedule/reschedule) e.g. Right-shift, partial/repair, complete regeneration
Scheduling/Rescheduling Time Policy	Scheduling/rescheduling time policy refers to when to generate a new schedule and/or update the initial schedule (in short, when to schedule/reschedule) e.g. Periodic, event-driven, hybrid
Tardiness	Tardiness is defined as the positive difference between the completion time and the due date of a job. Tardiness is never negative. (Pinedo, 2002). $Tardiness = \max(Completion\ Time - Due\ Date, 0)$
Throughput	Throughput (also called throughput rate) is the amount of work processed by a set of resources over some time period (Wood, 2000).
Work-In-Process (WIP)	Work-in-process refers to the amount of jobs that are undergoing processing in the shop floor.

APPENDIX B

SEMICONDUCTOR MANUFACTURING, WAFER FABRICATION, AND ITS SCHEDULING COMPLEXITIES

B.1 SEMICONDUCTOR MANUFACTURING

Semiconductor manufacturing typically consists of four stages: (i) wafer fabrication, (ii) wafer probe, (iii) assembly or packaging, and (iv) final testing. Figure B.1 shows these four stages and the transition of silicon wafers to the integrated circuits (ICs). In wafer fabrication, successive operations are used to build up layers of various patterned materials on the silicon wafers to form the multiple layers of circuit patterns on a single chip. Each wafer may contain many chips (or dies). In wafer probe, the individual dies are tested electronically using thin probes and the rejected dies are marked. The wafers are then moved to the assembly/packaging stage where the wafers are cut into individual die. The defective dies are discarded; the good dies are identified and are mounted onto lead frames, wire bonded and subsequently covered with a hard plastic compound to protect the die. After the assembly cycle, the packaged ICs are subjected to a final test and the devices are sorted into good or reject trays, tubes or bins.

Figure B.1: A simplified semiconductor manufacturing flow

Wafer fabrication operations are carried out in clean rooms (also called wafer fabs). A lot is a set of wafers that travels as a unit between workstations and is the equivalent terminology for job in wafer fabrication. The number of wafers in a lot is termed as lot size. A lot typically contains a dozens of wafers. Every lot has its own process flow defined in a run sheet that is made up of a series of recipes. Each recipe corresponds to a processing step and includes detailed processing and setup related information. Each process flow can involve as many as 600 steps. At any given time, a lot is either being processed by a machine, waiting in a queue, waiting to rejoin the wafers that may have been split for rework, or waiting for instruction to proceed as they may be on hold for various reasons.

Wafer fabrication and wafer probe are generally referred to as the front-end operations while assembly and final testing are the back-end operations (see Figure B.1). Actual wafer fabrication cycle time typically falls between 5 to 40 days (Johri, 1993). Cycle time of the assembly operation in a typical back-end usually falls in the range of 3 to 6 days while typical test operation cycle time is in the region of 1.5 to 15 days (Sivakumar, 1999).

This research will focus on wafer fabrication of the semiconductor front end because of the long cycle time, extreme capitalization required to create and run the

fab and the extreme complexity of the manufacturing that occurs in a wafer fab as compared to the other stages.

B.2 WAFER FABRICATION

The specific process followed to make a product varies according to the type and the company that is manufacturing the chips. However, all wafer fabrication usually involves these basic iterative process steps³ (see Figure B.2):

Figure B.2: Iterative wafer processing steps (re-entrant flow)

(i) Oxidation and Diffusion

Oxidation is a process by which a layer of silicon oxide with controlled thickness is formed on the surface of the silicon wafer at an elevated temperature. Oxide layer is typically used as a mask against ion implantation, as isolation for devices, and/or as pad, gate and capacitor oxide. The process of diffusion is used to introduce chosen dopants into selected regions of a semiconductor crystal.

³ More details can be obtained from the following websites:

(i) <http://www.infras.com/Tutorial/sld001.htm>, (ii) <http://www.micron.com/k12/semiconductors/fab.html>

(ii) (Photo)Lithography or Masking

Lithography encompasses the complete process of transferring an image from a photographic mask to a resultant pattern on a wafer. The wafer is uniformly coated with a thick light-sensitive liquid and exposed to an ultraviolet light source. Photoresist exposed by the transparent areas of the mask will harden and unexposed photoresist will be removed with a developer solution.

(iii) Etch and Clean Tech

The etch process is used to remove required portion(s) of the exposed nitride layer that is/are not protected by the hardened photoresist. The clean tech process is then used to remove the hardened photoresist.

(iv) Implantation

Ion implantation is a method of introducing energized ions (dopants) in a controlled and precise manner into unprotected areas of silicon wafers to create positive and negative areas on the wafer to encourage or discourage the flow of current.

(v) Film Deposition and Metallization (or Thin Film)

This process is used to provide conducting regions within the device, as electrical insulation between metals, and for protection from the environment. Chemical vapor deposition is used for depositing these layers of thin films.

(vi) Metrology (Inspection and Measurement)

The wafer is inspected and the physical, dimensional, and/or electrical properties of materials are quantified to identify defects and to guide future operations (e.g. continue, rework, scrap).

B.3 COMPLEX INHERENT CHARACTERISTICS OF WAFER FAB

It is generally difficult to schedule the next optimum task for each machine in a complicated semiconductor wafer fabrication plant. In a typical wafer fab, the simultaneous production of multiple product types within the fab at any given time is common. The process of each product type requires about 200-600 processing steps with re-entrant flow (multiple visits to the workstations).

In a fab, there are a large number of equipment types involving hundreds of machines. As it makes more economic sense to share expensive machines with all the lots that require processing on these machines, lots at different stages of their manufacturing cycle therefore need to compete for the use of the same machines. The re-entrant flow is one of the complex characteristics of a wafer fab from the next task scheduling perspective. In addition, switching between different product types may require lengthy setup or changeover time.

There are a number of types of lots that are present in the fab at any one time. For example, normal lots, priority lots (hot or bullet lot), rework lots, qualification lots, lots for new product evaluation and R&D evaluation, etc. The qualification lots are used to qualify the machines on a regular basis to ensure that the machines are running within specifications. The frequency and the type of qualification vary according to machine type. Qualification lots decrease the availability of the machines due to non-product processing and may increase the factory cycle time if not scheduled properly (Shikalgar et al., 2002). Apart from the scheduling of different types of lots, preventive maintenance for each individual machine also needs to be scheduled based on different criteria, such as calendar time, number of

wafers or lots processed, amount of time spent processing, or even based on radio frequency that vary with different recipes used.

The nature of some chemical processes requires a time link constraint to be imposed on certain process steps. This time link constraint means that the wafers, once processed at a particular machine at certain process steps, cannot be delayed for more than a specified maximum waiting duration before it is processed by the machine at its next process step.

Different machine characteristics may pose yet another source of constraint on the manufacturing process. Machines can be classified as wafer-by-wafer, batch, or serial. For example, wafer-by-wafer machines (e.g. lithography machines) process wafer pieces by pieces while batch processing machines (e.g. furnaces) perform the operations by grouping (i.e. batching) of multiple lots together based on one or more criteria for processing simultaneously. Serial machines (e.g. wetbenches – sinks for cleaning) typically are able to process more than one lot at the same time in a sequential stepwise manner (e.g. there may be more than one sinks and lots are moved from one sink to another within the same wetbench). In addition, some machines may not have the capabilities to meet the stringent requirements specified by the customers. As such, these wafers can only be processed by specific, possibly higher end, machines.

B.4 COMPLEXITIES DUE TO UNCERTAINTIES

Wafer fabrication is highly dynamic and stochastic in nature. New lots may arrive randomly in varying quantities over time and there may be unpredicted arrival of urgent orders. Priorities and due date of existing lots could be modified in any stage

of the production. Process and setup times are stochastic due to varying operators' and machines' capabilities and efficiencies. Apart from machine unavailability, machine may fail when it is least expected and it may be difficult to predict the time to repair the machine.

During the manufacturing cycle, an unknown number of wafers may also be scrapped or sent for rework. Quality problems (i.e. yield losses) may render the common use of fixed processing time specified in per lot basis to be less accurate since the fixed processing time is taken irrespective of the number of wafers in a lot. Other inherent complexities include disturbances such as the random occurrence of unavailability of processing materials, handlers, hardware, machines and operators.

Suresh and Chaudhuri (1993) classified uncertainties in a wafer fabrication into four categories: (i) job-related, (ii) machine-related, (iii) operations-related, and (iv) others. They summarized most of the dynamic events that could occur within a wafer fab under these categories (see Table B.1).

The inherent characteristics, uncertainties and disturbances make wafer fabrication highly complex, stochastic and dynamic. It is thus a significant challenge to develop effective scheduling methods for use in wafer fabrication. In fact, most, if not all studies have only considered a very small number of the uncertainty causes (Aytug et al., 2003).

Table B.1: Classification of dynamic events (Suresh and Chaudhuri, 1993)

Category	Events considered
Job related	Jobs arrive at random
	Process times are non-deterministic
	Due-dates are changing
	Dynamic priorities
	Demand changes
Machine related	Machine breakdowns
	Machine non-availability
	Loading limits
	Machine blocking/deadlock
	Capacity conflicts
Operations related	Quality rejects
	Yield losses
Others	Operator non-availability
	Raw material – late deliveries
	Defective material
	Dynamic routing

APPENDIX C

T-TESTS RESULTS TO COMPARE NEXTS, PSR, AND DISPATCHING TIME POLICIES

In order to find out (estimate) if there are statistically significant difference between the mean cycle times between the NEXTS time policy (denoted by subscript A) versus the PSR and dispatching time policies (denoted by subscript B), tests of hypotheses of means are carried out (Law, 2007). The null hypothesis is $H_0 : \mu_A - \mu_B = 0$ and the alternative hypothesis is $H_1 : \mu_A - \mu_B \neq 0$. Assuming the populations from which the samples of size $n_A (=10)$ and $n_B (=10)$ are collected from the replications are normal with means μ_A and μ_B with unknown variances, the test statistic t is given by (Walpole et al., 2002):

Case I ($\sigma_A^2 = \sigma_B^2$):

$$t = \frac{(\bar{x}_A - \bar{x}_B) - (\mu_A - \mu_B)}{s_p \sqrt{1/n_A + 1/n_B}} \quad (C.1)$$

where pooled sample standard deviation, s_p , is given by:

$$s_p = \sqrt{\frac{s_A^2(n_A - 1) + s_B^2(n_B - 1)}{n_A + n_B - 2}} \quad (C.2)$$

with degrees of freedom,

$$\nu = n_A + n_B - 2. \quad (C.3)$$

Case II ($\sigma_A^2 \neq \sigma_B^2$):

$$t = \frac{(\bar{x}_A - \bar{x}_B) - (\mu_A - \mu_B)}{\sqrt{\frac{s_A^2}{n_A} + \frac{s_B^2}{n_B}}} \quad (C.4)$$

with approximate degrees of freedom,

$$\nu = \frac{(s_A^2/n_A + s_B^2/n_B)^2}{\left[(s_A^2/n_A)^2/(n_A - 1)\right] + \left[(s_B^2/n_B)^2/(n_B - 1)\right]} \quad (C.5)$$

H_0 is rejected if the t -value (calculated using Equation (C.1) or (C.5)) falls into the critical region given by $|t| > t_{\alpha/2, \nu}$ where the degrees of freedom, ν , is calculated based on Equation (C.3) or (C.5) depending on whether $\sigma_A^2 = \sigma_B^2$ or $\sigma_A^2 \neq \sigma_B^2$.

The two-sided variances hypothesis test is used to determine whether the test statistic from Case I or Case II should be used (null hypothesis, $H_0 : \sigma_A^2 = \sigma_B^2$; alternative hypothesis, $H_1 : \sigma_A^2 \neq \sigma_B^2$). H_0 is rejected if the f -value ($f = \frac{s_A^2}{s_B^2}$) falls into the critical region given by $f < f_{1-\alpha/2}(\nu_A, \nu_B)$ and $f > f_{\alpha/2}(\nu_A, \nu_B)$ where $\nu_A = n_A - 1$ and $\nu_B = n_B - 1$.

Table C.1 summarizes the test statistics and the conclusion (i.e. whether to accept or to reject H_0) of the tests of hypotheses of variances (f -test) and means (t -test). For the 2-tailed f -test (hypothesis of variances), the critical region is $f < 0.314$ and $f > 3.18$ based on 0.1 level of significance (i.e. $\alpha = 0.1$).

Table C.1: Results of the tests of hypotheses of variances and means

		In increasing level of variability and disturbance →				
		Level 1	Level 2	Level 3	Level 4	Level 5
	Variability Introduced to Expected Arrivals:	UNIF(-10, 10)	UNIF(-60, 60)	UNIF(-110, 110)	UNIF(-160, 160)	UNIF(-210, 210)
	MTTF Disturbance:	EXP(4320)	EXP(3600)	EXP(2880)	EXP(2160)	EXP(1440)
NEXTS_G1_T2 vs PSR_R25_G1_T2	$f = s_A^2/s_B^2$	0.92	0.97	0.96	0.93	0.93
	Accept $H_0: \sigma_A^2 = \sigma_B^2$?	Yes	Yes	Yes	Yes	Yes
	S_p	221.27	212.13	212.97	199.02	258.90
	d.o.f, v	18	18	18	18	18
	Test Statistic, t	-0.06	-0.03	-0.05	-0.06	-0.05
	Accept $H_0: \mu_A = \mu_B$?	Yes	Yes	Yes	Yes	Yes
NEXTS_G1_T4 vs PSR_R25_G1_T4	$f = s_A^2/s_B^2$	0.92	0.95	1.01	0.84	1.01
	Accept $H_0: \sigma_A^2 = \sigma_B^2$?	Yes	Yes	Yes	Yes	Yes
	S_p	222.45	216.23	210.28	216.64	288.48
	d.o.f, v	18	18	18	18	18
	Test Statistic, t	-0.06	0.01	-0.06	-0.13	-0.09
	Accept $H_0: \mu_A = \mu_B$?	Yes	Yes	Yes	Yes	Yes
NEXTS_G1_T4 vs PSR_R25_G25_T4	$f = s_A^2/s_B^2$	0.92	0.96	1.03	0.84	1.01
	Accept $H_0: \sigma_A^2 = \sigma_B^2$?	Yes	Yes	Yes	Yes	Yes
	S_p	222.24	215.94	209.08	216.33	288.29
	d.o.f, v	18	18	18	18	18
	Test Statistic, t	-0.08	-0.01	-0.07	-0.16	-0.11
	Accept $H_0: \mu_A = \mu_B$?	Yes	Yes	Yes	Yes	Yes
NEXTS_G1_T4 vs PSR_R60_G1_T4	$f = s_A^2/s_B^2$	0.85	0.93	1.01	0.89	1.03
	Accept $H_0: \sigma_A^2 = \sigma_B^2$?	Yes	Yes	Yes	Yes	Yes
	S_p	227.00	217.84	210.11	213.09	286.83
	d.o.f, v	18	18	18	18	18
	Test Statistic, t	-0.22	-0.07	-0.11	-0.23	-0.13
	Accept $H_0: \mu_A = \mu_B$?	Yes	Yes	Yes	Yes	Yes
NEXTS_G1_T4 vs PSR_R60_G60_T4	$f = s_A^2/s_B^2$	0.88	1.04	1.03	1.00	0.94
	Accept $H_0: \sigma_A^2 = \sigma_B^2$?	Yes	Yes	Yes	Yes	Yes
	S_p	225.42	211.76	209.09	206.99	293.14
	d.o.f, v	18	18	18	18	18
	Test Statistic, t	-0.23	-0.20	-0.22	-0.22	-0.20
	Accept $H_0: \mu_A = \mu_B$?	Yes	Yes	Yes	Yes	Yes
NEXTS_G1_T4 vs PSR_R405_G60_T4	$f = s_A^2/s_B^2$	1.00	0.91	1.13	1.00	1.04
	Accept $H_0: \sigma_A^2 = \sigma_B^2$?	Yes	Yes	Yes	Yes	Yes
	S_p	217.91	218.81	204.66	207.08	286.27
	d.o.f, v	18	18	18	18	18
	Test Statistic, t	-0.84	-0.89	-1.00	-0.96	-0.52
	Accept $H_0: \mu_A = \mu_B$?	Yes	Yes	Yes	Yes	Yes
NEXTS_G1_T4 vs PSR_R750_G60_T4	$f = s_A^2/s_B^2$	0.76	0.71	0.47	0.75	1.03
	Accept $H_0: \sigma_A^2 = \sigma_B^2$?	Yes	Yes	Yes	Yes	Yes
	S_p	234.23	234.15	264.46	223.06	286.46
	d.o.f, v	18	18	18	18	18
	Test Statistic, t	-2.01	-1.92	-1.79	-2.09	-0.90
	Accept $H_0: \mu_A = \mu_B$?	Yes	Yes	Yes	Yes	Yes
NEXTS_G1_T4 vs PSR_R1095_G60_T4	$f = s_A^2/s_B^2$	0.36	0.39	0.30	0.72	0.90
	Accept $H_0: \sigma_A^2 = \sigma_B^2$?	Yes	Yes	No	Yes	Yes
	S_p	298.02	285.83	-	225.64	296.41
	d.o.f, v	18	18	14	18	18
	Test Statistic, t	-2.44	-2.92	-2.55	-3.16	-2.30
	Accept $H_0: \mu_A = \mu_B$?	No	No	No	No	No
NEXTS_G1_T4 vs PSR_R1440_G60_T4	$f = s_A^2/s_B^2$	0.26	0.31	0.27	0.58	0.59
	Accept $H_0: \sigma_A^2 = \sigma_B^2$?	No	No	No	Yes	Yes
	S_p	-	-	-	241.95	335.42
	d.o.f, v	13	14	13	18	18
	Test Statistic, t	-3.49	-3.40	-2.90	-4.18	-2.61
	Accept $H_0: \mu_A = \mu_B$?	No	No	No	No	No
NEXTS_G1_T4 vs Dispatching_G1_T4	$f = s_A^2/s_B^2$	0.21	0.23	0.23	0.21	0.35
	Accept $H_0: \sigma_A^2 = \sigma_B^2$?	No	No	No	No	Yes
	S_p	-	-	-	-	402.87
	d.o.f, v	13	13	13	13	18
	Test Statistic, t	-3.28	-3.44	-3.39	-3.42	-2.74
	Accept $H_0: \mu_A = \mu_B$?	No	No	No	No	No

For the 2-tailed t -test (hypothesis of means), the critical region for Case I ($\sigma_A^2 = \sigma_B^2$) is given by $|t| > t_{0.025, 10+10-2} (= 2.101)$ at 0.05 level of significance. The case of $\sigma_A^2 \neq \sigma_B^2$ occurs in Level 3 of NEXTS_G1_T4 vs PSR_R1095_G60_T4, in Level 1 to Level 3 of NEXTS_G1_T4 vs PSR_R1440_G60_T4, and in Level 1 to Level 4 of NEXTS_G1_T4 vs Dispatching_G1_T4 (see Table C.1). For cases where $\sigma_A^2 \neq \sigma_B^2$, the critical region for $\nu = 13$ and $\nu = 14$ is given by $|t| > t_{0.025, 13} (= 2.160)$ and $|t| > t_{0.025, 14} (= 2.145)$ respectively at 0.05 level of significance.

APPENDIX D

CYCLE TIME AVERAGE DATA AND ANOVA ON RANKS RESULTS

The ANOVA on Ranks test statistic H (see last row of Table D.1 to D.12) is computed by ranking all observations from smallest to largest without regard for treatment group. The average value of the ranks for each treatment group are computed and compared. For small sample sizes, the actual distribution of H is used. If H is small, the average ranks observed in each treatment group are approximately the same and can be concluded that the data is consistent with the null hypothesis that all the samples were drawn from the same population. This implies that there is no significant difference between the cycle time average (*avgCT*) performance of each transport time level (i.e. *NS*, *WS*, *ND*, and *WD*). If H is a large number, the variability among the average ranks is larger than expected from random variability in the population, and it can be concluded that the samples were drawn from different populations (i.e., the differences between the groups are statistically significant).

The P value given in the last row of Table D.1 to D.12 is the probability of being wrong in concluding that there is a true difference between the groups based on the test statistic H used. It can be concluded that there are significant differences between the groups with at least 95% confidence when $P < 0.05$. Since all P values are much larger than 0.05, there is no statistically significant difference among the simulation cycle time average output result for different transportation levels.

Table D.1: $avgCT$, H , and P values for $\{PT=N, AT=N, TT\}$ problem configuration

Variant 1				Variant 2				Variant 3			
N,N,NS	N,N,WS	N,N,ND	N,N,WD	N,N,NS	N,N,WS	N,N,ND	N,N,WD	N,N,NS	N,N,WS	N,N,ND	N,N,WD
1.750	2.641	2.531	1.651	1.768	2.680	2.566	1.670	1.752	2.646	2.536	1.655
3.439	2.719	1.664	2.802	3.493	2.769	1.674	2.851	3.448	2.730	1.665	2.804
2.397	1.999	2.465	2.330	2.443	2.025	2.496	2.363	2.407	2.009	2.470	2.338
2.797	3.671	2.650	2.190	2.840	3.720	2.693	2.213	2.805	3.678	2.657	2.196
1.621	1.554	2.231	1.614	1.631	1.555	2.280	1.619	1.623	1.553	2.234	1.616
2.032	1.403	2.260	1.690	2.062	1.405	2.298	1.706	2.038	1.404	2.267	1.697
1.492	1.636	1.735	1.661	1.499	1.647	1.748	1.676	1.493	1.639	1.738	1.667
1.587	1.948	1.909	3.539	1.598	1.984	1.931	3.588	1.590	1.955	1.915	3.540
2.491	2.525	3.408	1.637	2.549	2.579	3.456	1.640	2.492	2.538	3.414	1.637
2.040	1.473	1.943	3.727	2.070	1.476	1.973	3.781	2.047	1.474	1.953	3.732
1.561	2.824	1.509	2.710	1.573	2.872	1.518	2.768	1.562	2.831	1.512	2.711
3.322	1.758	2.413	2.583	3.370	1.775	2.464	2.629	3.330	1.764	2.418	2.596
2.710	2.100	1.602	1.729	2.762	2.128	1.604	1.748	2.711	2.107	1.602	1.733
2.753	3.381	2.086	2.043	2.795	3.425	2.119	2.077	2.764	3.387	2.096	2.050
2.118	2.571	2.297	2.167	2.154	2.624	2.326	2.194	2.120	2.589	2.298	2.170
2.293	3.778	1.769	2.475	2.330	3.834	1.781	2.510	2.296	3.784	1.773	2.479
2.008	3.250	3.367	2.089	2.041	3.286	3.419	2.130	2.018	3.252	3.376	2.102
1.803	3.516	2.044	1.804	1.828	3.568	2.076	1.822	1.809	3.521	2.049	1.807
1.744	2.430	1.851	2.755	1.755	2.456	1.867	2.793	1.745	2.438	1.854	2.759
1.520	1.984	3.264	1.966	1.523	2.017	3.308	1.996	1.521	1.995	3.274	1.971
$H = 1.456, P = 0.692$				$H = 1.420, P = 0.701$				$H = 1.441, P = 0.696$			

Table D.2: $avgCT$, H , and P values for $\{PT=M, AT=N, TT\}$ problem configuration

Variant 1				Variant 2				Variant 3			
M,N,NS	M,N,WS	M,N,ND	M,N,WD	M,N,NS	M,N,WS	M,N,ND	M,N,WD	M,N,NS	M,N,WS	M,N,ND	M,N,WD
2.682	2.078	2.817	7.645	2.803	2.160	2.953	7.814	2.703	2.102	2.841	7.654
1.912	1.703	5.491	2.720	1.974	1.753	5.645	2.851	1.928	1.717	5.507	2.734
3.643	3.540	6.221	5.687	3.826	3.704	6.384	5.830	3.677	3.602	6.226	5.713
2.143	4.577	4.417	4.311	2.240	4.764	4.576	4.454	2.168	4.590	4.431	4.323
3.509	2.293	5.337	4.276	3.645	2.420	5.496	4.412	3.521	2.329	5.343	4.299
2.583	4.573	2.402	3.102	2.718	4.715	2.508	3.215	2.614	4.599	2.418	3.122
3.124	3.701	4.390	2.669	3.259	3.847	4.509	2.778	3.145	3.721	4.397	2.683
2.221	3.853	2.326	4.234	2.344	3.999	2.420	4.366	2.245	3.873	2.339	4.237
3.085	2.991	2.152	3.197	3.242	3.133	2.255	3.332	3.103	3.015	2.187	3.216
6.192	5.609	5.880	3.498	6.347	5.787	6.050	3.634	6.208	5.611	5.889	3.521
6.092	2.592	2.839	3.505	6.298	2.732	2.946	3.644	6.098	2.645	2.854	3.534
2.704	2.674	2.242	4.546	2.848	2.823	2.334	4.674	2.722	2.718	2.276	4.551
6.163	2.577	2.670	3.650	6.354	2.687	2.814	3.796	6.166	2.600	2.714	3.681
2.509	2.303	2.450	5.538	2.649	2.439	2.553	5.707	2.537	2.332	2.467	5.559
3.524	2.383	2.707	3.209	3.696	2.502	2.830	3.362	3.541	2.409	2.727	3.240
3.594	2.322	2.708	4.832	3.794	2.416	2.821	5.002	3.638	2.340	2.729	4.844
4.190	4.376	2.630	3.377	4.373	4.552	2.747	3.529	4.243	4.408	2.661	3.406
5.504	2.547	6.534	5.716	5.681	2.679	6.693	5.874	5.522	2.582	6.551	5.738
3.385	6.071	5.091	2.559	3.522	6.257	5.226	2.689	3.401	6.082	5.106	2.599
4.350	4.235	2.984	1.844	4.537	4.416	3.109	1.913	4.378	4.271	2.992	1.860
$H = 2.887, P = 0.409$				$H = 2.765, P = 0.429$				$H = 2.878, P = 0.411$			

Table D.3: $avgCT$, H , and P values for $\{PT=W, AT=N, TT\}$ problem configuration

Variant 1				Variant 2				Variant 3			
W,N,NS	W,N,WS	W,N,ND	W,N,WD	W,N,NS	W,N,WS	W,N,ND	W,N,WD	W,N,NS	W,N,WS	W,N,ND	W,N,WD
2.889	2.713	2.220	4.512	3.053	2.859	2.359	4.753	2.911	2.747	2.259	4.553
5.332	2.131	4.366	2.544	5.563	2.267	4.557	2.695	5.384	2.179	4.386	2.569
2.989	8.734	4.513	7.335	3.208	9.003	4.739	7.549	3.026	8.736	4.526	7.344
2.494	3.721	7.680	6.982	2.718	3.939	7.940	7.222	2.553	3.756	7.702	6.987
5.972	9.484	6.582	6.420	6.248	9.736	6.803	6.641	5.980	9.497	6.591	6.439
3.382	3.048	6.401	10.46	3.654	3.230	6.661	10.74	3.386	3.082	6.429	10.46
2.554	5.580	4.779	3.009	2.760	5.827	5.009	3.176	2.593	5.590	4.796	3.041
2.596	6.368	3.374	4.712	2.785	6.609	3.578	4.915	2.643	6.392	3.396	4.730
4.310	3.394	3.917	4.517	4.546	3.639	4.137	4.743	4.352	3.431	3.927	4.536
2.951	6.661	8.580	2.479	3.159	6.923	8.754	2.657	2.964	6.672	8.593	2.519
5.999	8.960	7.126	2.744	6.272	9.227	7.383	2.933	6.013	8.974	7.130	2.811
3.253	3.359	5.420	10.49	3.463	3.567	5.661	10.74	3.267	3.392	5.429	10.495
8.128	5.656	3.480	3.989	8.387	5.902	3.684	4.188	8.158	5.672	3.505	4.018
6.890	8.265	3.823	3.438	7.195	8.530	4.047	3.693	6.899	8.274	3.852	3.479
2.927	4.163	4.907	3.201	3.121	4.377	5.129	3.387	2.974	4.175	4.923	3.238
5.285	5.372	6.101	2.191	5.538	5.635	6.353	2.300	5.325	5.397	6.127	2.222
3.261	4.236	3.735	2.647	3.498	4.447	3.944	2.824	3.311	4.244	3.758	2.690
4.211	3.864	3.274	2.656	4.457	4.099	3.493	2.823	4.222	3.920	3.306	2.696
3.163	2.538	6.749	6.576	3.361	2.721	6.959	6.813	3.199	2.595	6.753	6.584
4.083	4.780	3.922	4.332	4.300	5.024	4.159	4.589	4.118	4.828	3.966	4.367
$H = 4.301, P = 0.231$				$H = 4.059, P = 0.255$				$H = 4.377, P = 0.224$			

Table D.4: $avgCT$, H , and P values for $\{PT=N, AT=M, TT\}$ problem configuration

Variant 1				Variant 2				Variant 3			
N,M,NS	N,M,WS	N,M,ND	N,M,WD	N,M,NS	N,M,WS	N,M,ND	N,M,WD	N,M,NS	N,M,WS	N,M,ND	N,M,WD
2.855	5.195	2.938	2.636	2.890	5.247	2.980	2.668	2.858	5.202	2.943	2.639
4.377	6.590	8.024	3.434	4.433	6.632	8.071	3.482	4.389	6.592	8.024	3.447
3.581	6.331	2.943	3.420	3.623	6.387	2.977	3.448	3.587	6.332	2.946	3.423
5.370	2.707	2.345	3.156	5.428	2.748	2.368	3.193	5.379	2.718	2.346	3.158
4.381	7.933	3.226	2.447	4.430	7.983	3.263	2.476	4.389	7.933	3.232	2.455
3.557	9.508	8.182	3.153	3.609	9.561	8.231	3.193	3.566	9.513	8.182	3.158
3.054	2.739	2.744	7.238	3.089	2.779	2.781	7.286	3.062	2.744	2.752	7.240
5.520	2.509	11.39	4.278	5.579	2.546	11.45	4.324	5.525	2.515	11.39	4.285
2.250	4.361	5.900	5.766	2.282	4.403	5.945	5.819	2.261	4.370	5.900	5.773
2.267	3.399	3.893	1.952	2.306	3.447	3.942	1.973	2.279	3.408	3.898	1.961
2.555	6.300	6.055	4.359	2.593	6.362	6.107	4.405	2.560	6.305	6.058	4.364
5.333	4.217	4.143	5.204	5.388	4.275	4.176	5.251	5.335	4.226	4.151	5.206
2.855	5.643	4.380	7.620	2.897	5.697	4.431	7.674	2.869	5.646	4.392	7.623
2.428	3.116	3.581	5.377	2.459	3.153	3.625	5.425	2.436	3.122	3.587	5.384
2.174	3.422	2.360	4.879	2.204	3.467	2.390	4.928	2.185	3.424	2.366	4.889
2.981	3.907	3.832	4.585	3.027	3.954	3.879	4.636	2.996	3.910	3.838	4.596
6.378	3.811	3.892	7.359	6.433	3.865	3.943	7.413	6.384	3.815	3.900	7.366
4.964	3.789	5.208	3.049	5.010	3.835	5.249	3.082	4.968	3.798	5.213	3.051
8.625	3.865	5.955	8.916	8.673	3.919	6.007	8.967	8.629	3.875	5.958	8.915
7.575	2.626	7.477	3.584	7.633	2.662	7.522	3.625	7.583	2.634	7.477	3.585
$H = 1.599, P = 0.660$				$H = 1.575, P = 0.665$				$H = 1.582, P = 0.664$			

Table D.5: $avgCT$, H , and P values for $\{PT=M, AT=M, TT\}$ problem configuration

Variant 1				Variant 2				Variant 3			
M,M,NS	M,M,WS	M,M,ND	M,M,WD	M,M,NS	M,M,WS	M,M,ND	M,M,WD	M,M,NS	M,M,WS	M,M,ND	M,M,WD
4.660	3.591	2.900	4.822	4.819	3.709	3.036	4.966	4.687	3.605	2.940	4.837
6.953	5.778	7.598	6.511	7.144	5.926	7.761	6.622	6.967	5.782	7.608	6.513
3.723	8.905	2.933	3.251	3.904	9.106	3.063	3.348	3.752	8.905	2.948	3.257
2.785	2.419	5.449	5.198	2.933	2.518	5.588	5.363	2.809	2.438	5.451	5.224
3.458	2.536	3.332	4.206	3.602	2.639	3.470	4.361	3.474	2.565	3.349	4.217
7.445	7.315	3.812	5.034	7.628	7.479	3.950	5.180	7.463	7.333	3.842	5.042
6.914	3.744	2.135	5.603	7.071	3.907	2.218	5.735	6.936	3.774	2.154	5.613
3.176	3.406	10.17	4.599	3.323	3.535	10.34	4.754	3.196	3.421	10.17	4.616
2.491	3.456	3.099	4.079	2.609	3.635	3.235	4.235	2.529	3.477	3.120	4.095
2.358	3.003	3.871	6.498	2.479	3.126	3.991	6.642	2.388	3.043	3.889	6.502
2.495	3.440	2.171	5.362	2.622	3.606	2.253	5.538	2.520	3.470	2.191	5.396
3.381	2.805	6.255	4.802	3.542	2.936	6.410	4.955	3.417	2.824	6.264	4.816
3.351	2.080	3.444	2.713	3.509	2.176	3.573	2.855	3.373	2.101	3.457	2.751
6.045	5.267	3.042	3.261	6.240	5.431	3.173	3.402	6.079	5.271	3.082	3.290
4.308	4.700	5.699	2.569	4.452	4.832	5.855	2.671	4.350	4.718	5.721	2.596
5.005	4.700	5.670	2.989	5.171	4.859	5.821	3.138	5.027	4.725	5.675	3.032
4.870	3.137	2.522	2.986	5.054	3.262	2.642	3.092	4.892	3.161	2.550	2.992
9.513	2.933	4.376	7.607	9.686	3.068	4.502	7.763	9.528	2.958	4.384	7.619
4.757	4.446	2.550	10.47	4.924	4.621	2.659	10.65	4.780	4.468	2.565	10.47
3.827	5.802	4.835	11.97	4.002	5.981	5.001	12.14	3.848	5.821	4.854	11.98
$H = 2.846, P = 0.416$				$H = 2.885, P = 0.410$				$H = 2.829, P = 0.419$			

Table D.6: $avgCT$, H , and P values for $\{PT=W, AT=M, TT\}$ problem configuration

Variant 1				Variant 2				Variant 3			
W,M,NS	W,M,WS	W,M,ND	W,M,WD	W,M,NS	W,M,WS	W,M,ND	W,M,WD	W,M,NS	W,M,WS	W,M,ND	W,M,WD
15.06	3.021	5.714	5.294	15.33	3.246	5.941	5.520	15.06	3.062	5.728	5.309
3.035	4.474	2.806	9.462	3.252	4.715	2.973	9.686	3.061	4.505	2.832	9.476
6.057	4.591	3.026	2.364	6.323	4.774	3.225	2.532	6.114	4.610	3.071	2.416
2.576	3.781	5.259	3.248	2.755	4.015	5.498	3.482	2.621	3.821	5.312	3.315
3.554	6.328	3.653	2.564	3.802	6.569	3.869	2.736	3.602	6.346	3.687	2.601
3.671	8.596	8.015	4.147	3.920	8.862	8.285	4.417	3.713	8.614	8.020	4.180
6.619	8.908	10.26	3.355	6.901	9.134	10.52	3.585	6.619	8.936	10.28	3.405
3.746	6.854	3.606	7.528	3.974	7.061	3.785	7.814	3.776	6.868	3.640	7.552
2.808	4.745	5.511	3.361	3.035	4.999	5.711	3.555	2.873	4.791	5.522	3.386
3.703	7.848	5.206	3.644	3.951	8.097	5.453	3.860	3.759	7.849	5.228	3.689
4.304	4.539	3.851	4.774	4.559	4.780	4.063	4.955	4.352	4.575	3.870	4.800
3.038	5.508	3.278	9.855	3.231	5.773	3.490	10.10	3.074	5.539	3.327	9.860
3.227	7.610	8.276	3.018	3.472	7.869	8.523	3.209	3.284	7.613	8.299	3.063
3.563	3.644	7.489	4.670	3.786	3.849	7.730	4.868	3.589	3.693	7.497	4.706
10.44	6.002	6.248	4.925	10.72	6.244	6.510	5.146	10.45	6.036	6.269	4.935
3.744	5.085	7.622	7.492	3.986	5.282	7.873	7.749	3.788	5.118	7.629	7.492
5.427	6.529	3.809	8.328	5.643	6.767	4.006	8.595	5.431	6.567	3.834	8.335
6.919	10.46	8.688	2.944	7.180	10.71	8.905	3.159	6.932	10.46	8.692	2.973
6.656	4.461	2.514	4.490	6.935	4.688	2.682	4.718	6.668	4.491	2.547	4.517
6.927	3.203	3.445	7.157	7.202	3.402	3.642	7.418	6.942	3.242	3.480	7.175
$H = 2.538, P = 0.469$				$H = 2.602, P = 0.457$				$H = 2.656, P = 0.448$			

Table D.7: $avgCT$, H , and P values for $\{PT=N, AT=W, TT\}$ problem configuration

Variant 1				Variant 2				Variant 3			
N,W,NS	N,W,WS	N,W,ND	N,W,WD	N,W,NS	N,W,WS	N,W,ND	N,W,WD	N,W,NS	N,W,WS	N,W,ND	N,W,WD
7.418	11.04	4.887	2.630	7.473	11.10	4.927	2.657	7.425	11.05	4.891	2.636
4.875	5.069	11.88	6.835	4.930	5.112	11.93	6.886	4.882	5.075	11.88	6.839
7.973	4.288	5.458	5.798	8.031	4.333	5.508	5.844	7.974	4.291	5.463	5.803
9.779	9.672	7.261	10.04	9.843	9.731	7.312	10.10	9.780	9.673	7.262	10.04
3.452	4.402	4.495	5.747	3.496	4.446	4.534	5.792	3.457	4.411	4.496	5.749
11.21	4.706	4.854	2.786	11.27	4.750	4.898	2.820	11.21	4.713	4.859	2.795
3.474	2.929	9.000	6.828	3.520	2.961	9.049	6.885	3.483	2.934	9.001	6.832
4.051	3.653	3.521	2.955	4.101	3.696	3.562	2.989	4.059	3.659	3.529	2.960
10.11	3.943	4.854	8.637	10.17	3.980	4.888	8.686	10.11	3.949	4.858	8.637
4.298	5.216	2.622	4.120	4.340	5.267	2.652	4.155	4.304	5.226	2.628	4.122
5.783	3.147	5.732	4.548	5.839	3.185	5.770	4.595	5.789	3.159	5.736	4.554
4.929	4.253	11.99	3.154	4.979	4.290	12.05	3.186	4.940	4.259	11.99	3.160
4.554	7.120	5.779	2.351	4.614	7.165	5.830	2.372	4.565	7.121	5.789	2.355
5.343	4.925	7.542	4.483	5.394	4.975	7.592	4.524	5.348	4.931	7.546	4.487
4.288	3.171	8.311	8.802	4.339	3.208	8.365	8.840	4.299	3.176	8.315	8.806
5.572	5.242	7.890	4.355	5.634	5.291	7.935	4.401	5.579	5.245	7.895	4.364
4.179	3.119	11.80	6.777	4.223	3.161	11.85	6.839	4.187	3.130	11.80	6.787
4.022	4.542	8.914	10.56	4.062	4.591	8.969	10.604	4.028	4.547	8.917	10.56
5.406	6.725	11.47	6.605	5.462	6.774	11.52	6.652	5.417	6.727	11.48	6.609
6.718	9.465	3.257	12.79	6.766	9.518	3.297	12.844	6.721	9.475	3.266	12.79
$H = 4.595, P = 0.204$				$H = 4.561, P = 0.207$				$H = 4.628, P = 0.201$			

Table D.8: $avgCT$, H , and P values for $\{PT=M, AT=W, TT\}$ problem configuration

Variant 1				Variant 2				Variant 3			
M,W,NS	M,W,WS	M,W,ND	M,W,WD	M,W,NS	M,W,WS	M,W,ND	M,W,WD	M,W,NS	M,W,WS	M,W,ND	M,W,WD
4.342	4.901	5.049	4.481	4.502	5.063	5.198	4.661	4.372	4.917	5.059	4.508
8.226	3.080	14.94	4.834	8.389	3.230	15.13	4.968	8.227	3.121	14.96	4.851
10.54	2.797	14.91	8.293	10.72	2.912	15.08	8.462	10.55	2.822	14.92	8.298
4.777	3.087	2.720	7.216	4.926	3.224	2.832	7.395	4.785	3.109	2.746	7.225
10.09	10.06	6.883	6.867	10.29	10.23	7.034	7.052	10.11	10.06	6.907	6.870
4.895	8.343	3.140	5.339	5.058	8.529	3.250	5.509	4.909	8.372	3.157	5.350
6.092	2.989	5.273	8.375	6.269	3.094	5.423	8.509	6.119	3.007	5.281	8.390
6.197	5.036	5.262	3.379	6.375	5.170	5.436	3.494	6.224	5.062	5.263	3.391
10.92	3.316	5.254	6.657	11.09	3.463	5.410	6.805	10.94	3.352	5.266	6.659
14.32	3.965	3.523	12.99	14.51	4.132	3.639	13.14	14.33	4.003	3.536	13.00
11.31	6.278	12.50	4.094	11.49	6.464	12.66	4.233	11.31	6.279	12.50	4.119
4.628	6.696	3.882	5.818	4.787	6.851	4.040	6.006	4.647	6.703	3.900	5.842
5.590	5.101	6.886	9.397	5.761	5.276	7.032	9.554	5.613	5.109	6.902	9.402
4.091	11.07	10.87	5.916	4.240	11.25	11.02	6.054	4.107	11.07	10.89	5.930
4.295	8.219	5.533	6.004	4.443	8.400	5.678	6.175	4.331	8.230	5.550	6.006
3.954	5.107	13.35	4.320	4.118	5.276	13.52	4.432	3.969	5.136	13.35	4.342
2.977	4.239	5.478	11.12	3.091	4.386	5.639	11.27	2.996	4.253	5.507	11.13
6.418	9.460	2.695	4.216	6.597	9.637	2.812	4.391	6.422	9.466	2.714	4.235
7.163	5.297	3.309	4.218	7.335	5.432	3.420	4.344	7.178	5.307	3.324	4.237
12.06	5.948	5.872	3.939	12.25	6.152	6.017	4.061	12.06	5.951	5.890	3.966
$H = 1.821, P = 0.610$				$H = 1.839, P = 0.606$				$H = 1.767, P = 0.622$			

Table D.9: $avgCT$, H , and P values for $\{PT=W, AT=W, TT\}$ problem configuration

Variant 1				Variant 2				Variant 3			
W,W,NS	W,W,WS	W,W,ND	W,W,WD	W,W,NS	W,W,WS	W,W,ND	W,W,WD	W,W,NS	W,W,WS	W,W,ND	W,W,WD
3.410	4.669	3.956	5.851	3.635	4.930	4.155	6.098	3.468	4.676	3.975	5.862
3.278	3.339	6.846	10.92	3.504	3.537	7.093	11.18	3.307	3.387	6.873	10.95
3.558	4.210	3.737	3.840	3.778	4.436	3.940	4.054	3.585	4.243	3.776	3.879
2.734	4.302	6.403	3.412	2.919	4.516	6.652	3.628	2.766	4.346	6.449	3.447
4.382	7.487	3.244	6.844	4.599	7.748	3.453	7.109	4.417	7.496	3.287	6.883
3.267	3.226	13.62	4.804	3.461	3.445	13.89	4.999	3.309	3.255	13.63	4.827
4.767	9.254	5.410	9.797	5.037	9.532	5.601	10.00	4.786	9.270	5.433	9.817
5.003	3.581	6.802	3.562	5.261	3.795	6.995	3.767	5.039	3.596	6.809	3.598
4.102	6.721	7.541	5.271	4.352	6.960	7.794	5.492	4.131	6.747	7.564	5.286
12.09	4.906	8.112	5.175	12.35	5.160	8.372	5.409	12.10	4.947	8.119	5.229
9.110	5.392	9.219	4.133	9.386	5.648	9.460	4.325	9.110	5.426	9.245	4.164
6.037	2.956	7.672	3.002	6.298	3.143	7.902	3.183	6.057	2.980	7.701	3.019
12.80	5.996	4.252	7.100	13.08	6.214	4.462	7.346	12.82	6.029	4.287	7.123
10.51	5.175	3.690	12.04	10.78	5.401	3.914	12.28	10.52	5.212	3.729	12.05
8.135	6.306	7.086	4.681	8.430	6.569	7.316	4.926	8.163	6.330	7.092	4.709
7.049	3.724	3.258	3.908	7.303	3.983	3.432	4.124	7.058	3.755	3.288	3.915
5.415	8.972	4.133	3.910	5.662	9.247	4.357	4.059	5.444	8.989	4.165	3.924
5.829	3.470	17.63	6.164	6.057	3.707	17.87	6.393	5.858	3.516	17.63	6.175
6.839	4.809	4.037	9.055	7.093	5.051	4.274	9.307	6.862	4.843	4.076	9.055
4.566	3.301	5.013	13.63	4.813	3.510	5.223	13.88	4.617	3.339	5.045	13.63
$H = 2.312, P = 0.510$				$H = 2.130, P = 0.546$				$H = 2.339, P = 0.505$			

Table D.10: $avgCT$, H , and P values for $\{PT=N, AT=E, TT\}$ problem configuration

Variant 1				Variant 2				Variant 3			
N,E,NS	N,E,WS	N,E,ND	N,E,WD	N,E,NS	N,E,WS	N,E,ND	N,E,WD	N,E,NS	N,E,WS	N,E,ND	N,E,WD
7.878	9.999	27.626	17.208	7.918	10.043	27.676	17.265	7.880	10.001	27.631	17.210
15.544	17.742	25.160	13.582	15.605	17.794	25.203	13.622	15.547	17.743	25.161	13.583
10.285	20.201	12.170	9.983	10.327	20.253	12.218	10.019	10.288	20.202	12.176	9.988
17.681	10.624	8.958	27.279	17.738	10.675	8.995	27.337	17.687	10.625	8.963	27.279
41.706	12.138	6.464	12.596	41.771	12.190	6.508	12.639	41.722	12.141	6.467	12.599
10.778	7.398	6.565	19.910	10.829	7.444	6.605	19.963	10.781	7.404	6.565	19.910
18.895	8.207	7.013	17.362	18.945	8.254	7.048	17.410	18.898	8.209	7.014	17.362
37.806	28.354	7.418	6.280	37.870	28.404	7.462	6.323	37.807	28.354	7.422	6.283
15.207	17.726	16.175	18.505	15.256	17.781	16.229	18.546	15.211	17.727	16.180	18.504
24.055	9.242	15.694	6.733	24.110	9.291	15.749	6.775	24.055	9.246	15.700	6.739
5.668	28.673	5.730	12.846	5.706	28.724	5.765	12.886	5.670	28.673	5.734	12.847
11.823	18.971	9.467	32.054	11.874	19.023	9.510	32.099	11.824	18.974	9.469	32.054
15.299	4.234	11.037	20.304	15.341	4.270	11.088	20.356	15.301	4.237	11.040	20.304
18.216	15.610	11.906	37.418	18.265	15.655	11.951	37.457	18.218	15.611	11.905	37.420
6.318	21.798	9.004	22.381	6.360	21.842	9.044	22.433	6.320	21.798	9.006	22.381
21.366	10.281	8.872	11.267	21.427	10.322	8.915	11.305	21.367	10.283	8.875	11.269
24.997	10.101	11.980	22.371	25.051	10.145	12.030	22.419	24.997	10.105	11.979	22.372
9.680	5.660	6.817	25.614	9.729	5.702	6.860	25.663	9.681	5.668	6.821	25.613
4.656	10.510	31.785	8.234	4.698	10.563	31.836	8.273	4.660	10.512	31.786	8.236
27.761	7.537	8.494	7.363	27.812	7.587	8.534	7.405	27.762	7.546	8.494	7.367
$H = 5.353, P = 0.148$				$H = 5.353, P = 0.148$				$H = 5.353, P = 0.148$			

Table D.11: $avgCT$, H , and P values for $\{PT=M, AT=E, TT\}$ problem configuration

Variant 1				Variant 2				Variant 3			
M,E,NS	M,E,WS	M,E,ND	M,E,WD	M,E,NS	M,E,WS	M,E,ND	M,E,WD	M,E,NS	M,E,WS	M,E,ND	M,E,WD
15.860	21.583	7.138	6.354	16.053	21.767	7.298	6.484	15.863	21.603	7.164	6.357
33.186	4.708	7.971	22.677	33.387	4.850	8.143	22.877	33.186	4.736	7.979	22.679
10.660	20.351	10.157	6.855	10.841	20.543	10.319	6.987	10.684	20.357	10.169	6.869
8.997	15.066	8.476	11.495	9.187	15.223	8.647	11.658	9.005	15.066	8.481	11.499
10.998	26.269	16.577	17.788	11.172	26.476	16.722	17.976	11.006	26.268	16.581	17.786
7.549	7.308	10.581	11.216	7.711	7.469	10.739	11.394	7.561	7.315	10.585	11.228
19.960	7.449	6.879	10.699	20.164	7.619	7.042	10.850	19.962	7.466	6.896	10.705
7.083	22.820	7.342	8.049	7.261	22.988	7.478	8.200	7.098	22.821	7.356	8.058
12.985	10.925	20.602	13.329	13.205	11.119	20.761	13.505	12.993	10.927	20.604	13.330
5.610	16.219	20.358	9.403	5.764	16.411	20.540	9.565	5.633	16.220	20.360	9.429
10.610	8.835	17.547	7.919	10.798	9.025	17.693	8.049	10.614	8.839	17.548	7.927
17.555	8.137	11.999	26.684	17.757	8.319	12.176	26.870	17.570	8.144	12.011	26.688
5.824	8.842	18.533	15.792	5.989	9.028	18.714	15.970	5.829	8.858	18.574	15.795
8.747	8.486	4.129	14.110	8.911	8.633	4.244	14.291	8.757	8.498	4.138	14.123
10.658	5.775	11.579	6.080	10.836	5.941	11.751	6.225	10.674	5.786	11.585	6.089
4.871	9.916	11.942	20.907	5.037	10.113	12.121	21.091	4.882	9.924	11.945	20.921
10.211	6.744	14.635	23.858	10.385	6.914	14.812	24.003	10.216	6.756	14.638	23.859
6.270	12.363	28.304	14.367	6.435	12.559	28.478	14.555	6.285	12.378	28.305	14.364
18.990	23.315	13.147	8.016	19.206	23.493	13.329	8.199	19.009	23.316	13.147	8.022
5.524	6.241	7.831	17.944	5.695	6.387	7.993	18.118	5.540	6.260	7.852	17.948
$H = 2.066, P = 0.559$				$H = 2.066, P = 0.559$				$H = 2.066, P = 0.559$			

Table D.12: $avgCT$, H , and P values for $\{PT=W, AT=E, TT\}$ problem configuration

Variant 1				Variant 2				Variant 3			
W,E,NS	W,E,WS	W,E,ND	W,E,WD	W,E,NS	W,E,WS	W,E,ND	W,E,WD	W,E,NS	W,E,WS	W,E,ND	W,E,WD
20.573	6.074	5.398	12.090	20.879	6.328	5.623	12.327	20.573	6.100	5.411	12.101
12.859	8.090	10.250	14.011	13.112	8.378	10.514	14.291	12.872	8.110	10.255	14.018
14.861	13.360	19.446	7.704	15.130	13.652	19.737	7.988	14.873	13.367	19.451	7.721
10.534	10.398	9.953	6.369	10.815	10.674	10.211	6.593	10.544	10.398	9.961	6.391
5.606	15.347	13.147	4.992	5.845	15.612	13.383	5.219	5.623	15.353	13.150	5.020
7.005	8.244	7.637	6.469	7.267	8.504	7.883	6.658	7.020	8.271	7.651	6.483
11.562	14.291	18.782	7.291	11.869	14.586	19.007	7.530	11.592	14.299	18.782	7.316
7.699	7.353	8.346	11.042	7.962	7.627	8.592	11.335	7.722	7.368	8.345	11.061
6.114	5.418	18.414	7.083	6.395	5.635	18.685	7.307	6.154	5.438	18.414	7.111
5.610	14.783	13.299	4.668	5.859	15.056	13.564	4.908	5.655	14.789	13.299	4.703
6.279	8.602	11.257	8.169	6.595	8.897	11.515	8.393	6.335	8.637	11.266	8.191
11.663	10.180	13.683	6.934	11.953	10.461	13.968	7.199	11.670	10.191	13.705	6.938
14.324	18.644	12.666	23.013	14.623	18.932	12.901	23.284	14.325	18.653	12.669	23.013
14.589	5.876	6.278	7.663	14.905	6.096	6.517	7.916	14.595	5.896	6.290	7.679
8.860	12.177	7.242	26.836	9.185	12.435	7.524	27.073	8.869	12.194	7.264	26.838
15.224	9.488	7.037	11.874	15.565	9.780	7.292	12.166	15.224	9.511	7.059	11.878
9.353	16.406	13.259	6.739	9.610	16.659	13.545	7.021	9.359	16.410	13.263	6.764
5.469	5.069	6.622	13.013	5.735	5.311	6.919	13.309	5.495	5.082	6.659	13.020
10.736	17.126	6.053	5.006	11.027	17.404	6.284	5.242	10.739	17.132	6.086	5.039
12.983	8.609	7.545	11.375	13.293	8.930	7.838	11.602	13.007	8.662	7.573	11.369
$H = 1.507, P = 0.681$				$H = 1.532, P = 0.675$				$H = 1.532, P = 0.675$			

APPENDIX E

CYCLE TIME AND TARDINESS RELATED
PERFORMANCE OF DISPATCHING RULES

Table E.1: Mean and 95% confidence interval of average and standard deviation of cycle time and tardiness for FIFO rule under different problem configurations

Problem Configurations	FIFO			
	<i>avgCT</i>	<i>sdCT</i>	<i>avgTARD</i>	<i>sdTARD</i>
{P1,DL}	318.366 ± 4.801	132.900 ± 1.553	31.289 ± 0.982	81.403 ± 1.369
{P2,DL}	315.234 ± 3.838	136.134 ± 1.572	32.843 ± 0.877	83.347 ± 1.226
{P3,DL}	319.399 ± 4.631	133.644 ± 1.356	31.475 ± 0.873	82.213 ± 1.253
{P4,DL}	315.123 ± 4.318	138.254 ± 1.297	33.077 ± 0.897	84.445 ± 1.286
{P1,DT}	317.429 ± 4.399	133.078 ± 1.573	74.959 ± 1.822	112.237 ± 1.155
{P2,DT}	313.581 ± 4.683	136.686 ± 1.543	78.032 ± 2.025	114.053 ± 1.453
{P3,DT}	316.834 ± 3.072	134.654 ± 1.408	74.941 ± 1.348	112.717 ± 0.841
{P4,DT}	317.214 ± 4.105	136.898 ± 1.194	79.672 ± 1.936	115.507 ± 1.218

Table E.2: Mean and 95% confidence interval of average and standard deviation of cycle time and tardiness for SPT rule under different problem configurations

Problem Configurations	SPT			
	<i>avgCT</i>	<i>sdCT</i>	<i>avgTARD</i>	<i>sdTARD</i>
{P1,DL}	306.162 ± 4.867	356.887 ± 5.948	59.859 ± 1.894	190.550 ± 4.511
{P2,DL}	312.469 ± 4.551	366.914 ± 5.184	53.171 ± 2.091	178.211 ± 4.559
{P3,DL}	307.288 ± 5.026	356.195 ± 5.730	59.544 ± 2.147	188.995 ± 4.327
{P4,DL}	310.353 ± 4.877	363.467 ± 5.221	52.469 ± 1.543	176.269 ± 3.291
{P1,DT}	305.818 ± 4.952	357.150 ± 5.685	126.686 ± 3.990	268.185 ± 5.421
{P2,DT}	309.137 ± 5.731	363.539 ± 6.309	116.239 ± 3.897	252.621 ± 5.051
{P3,DT}	304.603 ± 3.300	354.267 ± 4.738	124.949 ± 2.647	265.311 ± 4.495
{P4,DT}	311.987 ± 4.735	364.413 ± 4.896	117.948 ± 3.381	253.860 ± 4.790

Table E.3: Mean and 95% confidence interval of average and standard deviation of cycle time and tardiness for SCT rule under different problem configurations

Problem Configurations	SCT			
	<i>avgCT</i>	<i>sdCT</i>	<i>avgTARD</i>	<i>sdTARD</i>
{P1,DL}	318.469 ± 4.714	132.816 ± 1.572	31.374 ± 0.975	81.506 ± 1.337
{P2,DL}	315.480 ± 3.702	135.537 ± 1.566	33.423 ± 0.878	84.173 ± 1.249
{P3,DL}	319.384 ± 4.699	133.647 ± 1.314	31.544 ± 0.894	82.365 ± 1.265
{P4,DL}	315.549 ± 4.442	137.472 ± 1.269	33.738 ± 0.907	85.511 ± 1.296
{P1,DT}	317.556 ± 4.482	132.951 ± 1.599	75.087 ± 1.864	112.327 ± 1.156
{P2,DT}	313.742 ± 4.568	136.116 ± 1.449	79.428 ± 1.994	115.231 ± 1.450
{P3,DT}	316.613 ± 3.059	134.600 ± 1.419	74.980 ± 1.394	112.763 ± 0.864
{P4,DT}	317.440 ± 4.087	136.273 ± 1.168	81.135 ± 1.944	116.679 ± 1.228

Table E.4: Mean and 95% confidence interval of average and standard deviation of cycle time and tardiness for EDD rule under different problem configurations

Problem Configurations	EDD			
	<i>avgCT</i>	<i>sdCT</i>	<i>avgTARD</i>	<i>sdTARD</i>
{P1,DL}	318.22 ± 4.811	292.708 ± 2.854	0.608 ± 0.030	4.629 ± 0.151
{P2,DL}	315.311 ± 4.225	325.208 ± 2.999	0.719 ± 0.036	5.326 ± 0.248
{P3,DL}	319.076 ± 4.74	292.326 ± 2.690	0.727 ± 0.035	5.322 ± 0.225
{P4,DL}	315.22 ± 4.621	324.004 ± 3.361	0.839 ± 0.045	5.837 ± 0.232
{P1,DT}	317.439 ± 4.383	198.910 ± 0.823	42.321 ± 1.644	62.306 ± 1.121
{P2,DT}	312.599 ± 4.664	215.349 ± 1.459	41.730 ± 1.742	63.722 ± 1.353
{P3,DT}	316.369 ± 3.246	199.747 ± 0.814	42.574 ± 1.508	63.660 ± 1.125
{P4,DT}	316.158 ± 4.089	216.547 ± 1.078	43.537 ± 1.445	65.773 ± 1.064

Table E.5: Mean and 95% confidence interval of average and standard deviation of cycle time and tardiness for CR rule under different problem configurations

Problem Configurations	CR			
	<i>avgCT</i>	<i>sdCT</i>	<i>avgTARD</i>	<i>sdTARD</i>
{P1,DL}	318.740 ± 4.896	289.258 ± 2.659	0.619 ± 0.032	4.678 ± 0.163
{P2,DL}	315.934 ± 3.989	297.685 ± 2.607	0.743 ± 0.027	5.423 ± 0.184
{P3,DL}	318.674 ± 4.679	288.359 ± 2.734	0.731 ± 0.039	5.326 ± 0.253
{P4,DL}	316.788 ± 4.488	297.689 ± 2.680	0.836 ± 0.049	5.803 ± 0.249
{P1,DT}	317.563 ± 4.157	198.428 ± 0.837	42.263 ± 1.598	62.208 ± 1.130
{P2,DT}	315.153 ± 4.748	210.211 ± 1.486	41.993 ± 1.757	63.043 ± 1.406
{P3,DT}	316.443 ± 3.309	199.014 ± 0.792	42.539 ± 1.560	63.459 ± 1.133
{P4,DT}	318.847 ± 4.112	211.527 ± 1.259	43.782 ± 1.471	65.099 ± 1.058

Table E.6: Mean and 95% confidence interval of average and standard deviation of cycle time and tardiness for LS rule under different problem configurations

Problem Configurations	LS			
	<i>avgCT</i>	<i>sdCT</i>	<i>avgTARD</i>	<i>sdTARD</i>
{P1,DL}	318.301 ± 4.709	292.414 ± 2.77	0.614 ± 0.029	4.676 ± 0.171
{P2,DL}	314.998 ± 4.252	322.698 ± 2.95	0.730 ± 0.039	5.346 ± 0.262
{P3,DL}	318.907 ± 4.730	291.958 ± 2.64	0.732 ± 0.035	5.361 ± 0.246
{P4,DL}	315.456 ± 4.620	321.968 ± 3.169	0.819 ± 0.039	5.749 ± 0.196
{P1,DT}	317.523 ± 4.353	198.518 ± 0.773	42.351 ± 1.639	62.291 ± 1.099
{P2,DT}	312.611 ± 4.612	212.374 ± 1.297	41.785 ± 1.736	63.707 ± 1.330
{P3,DT}	316.311 ± 3.191	199.323 ± 0.823	42.584 ± 1.470	63.656 ± 1.102
{P4,DT}	315.989 ± 4.240	213.286 ± 1.089	43.518 ± 1.525	65.645 ± 1.072

Table E.7: Mean and 95% confidence interval of average and standard deviation of cycle time and tardiness for SSU rule under different problem configurations

Problem Configurations	SSU			
	<i>avgCT</i>	<i>sdCT</i>	<i>avgTARD</i>	<i>sdTARD</i>
{P1,DL}	252.946 ± 3.945	171.935 ± 1.797	24.702 ± 0.939	82.614 ± 2.078
{P2,DL}	249.602 ± 2.772	174.738 ± 1.416	27.619 ± 0.653	90.108 ± 1.568
{P3,DL}	253.586 ± 3.494	172.790 ± 1.333	25.019 ± 0.828	83.848 ± 1.548
{P4,DL}	251.233 ± 3.731	176.448 ± 1.919	28.210 ± 0.936	91.359 ± 1.986
{P1,DT}	250.719 ± 4.070	171.362 ± 1.537	58.424 ± 1.755	119.664 ± 1.894
{P2,DT}	248.333 ± 4.251	174.235 ± 2.000	62.658 ± 2.059	127.872 ± 2.460
{P3,DT}	250.039 ± 2.868	172.176 ± 2.333	58.260 ± 1.630	119.540 ± 2.348
{P4,DT}	250.846 ± 3.088	174.785 ± 1.840	63.472 ± 1.515	128.723 ± 1.828

Table E.8: Mean and 95% confidence interval of average and standard deviation of cycle time and tardiness for SSU+ rule under different problem configurations

Problem Configurations	SSU+			
	<i>avgCT</i>	<i>sdCT</i>	<i>avgTARD</i>	<i>sdTARD</i>
{P1,DL}	252.114 ± 3.848	186.789 ± 1.988	26.585 ± 0.965	93.080 ± 2.321
{P2,DL}	248.792 ± 2.785	194.354 ± 2.286	31.635 ± 0.847	109.595 ± 2.829
{P3,DL}	252.196 ± 3.438	186.754 ± 1.624	26.962 ± 0.928	93.850 ± 2.045
{P4,DL}	249.911 ± 3.707	195.800 ± 2.536	32.734 ± 1.085	111.041 ± 2.623
{P1,DT}	249.777 ± 4.043	185.748 ± 1.712	61.607 ± 1.767	134.234 ± 2.138
{P2,DT}	247.440 ± 4.268	195.444 ± 3.160	68.016 ± 2.261	153.344 ± 3.770
{P3,DT}	249.266 ± 2.860	185.164 ± 2.487	60.836 ± 1.711	132.219 ± 2.686
{P4,DT}	250.364 ± 3.221	194.216 ± 2.619	69.037 ± 1.522	152.121 ± 2.628

APPENDIX F

SIMULATION RESULTS FOR MOFCS AND MOFCS+ RULES

Table F.1: Mean and 95% confidence interval of $avgCT$, $sdCT$, $avgTARD$, and $sdTARD$ of the weighted MOFCS+ rule for the $\{PI, DL\}$ problem configuration

ID of Weights	Weights			MOFCS+ {P1,DL}			
	W_{FIFO}	W_{CR}	W_{SSU+}	$avgCT$	$sdCT$	$avgTARD$	$sdTARD$
1	0.000	0.333	0.667	257.607 $\pm$ 3.833	218.869 $\pm$ 2.204	16.650 $\pm$ 0.618	72.376 $\pm$ 1.984
2	0.000	0.500	0.500	272.230 $\pm$ 3.825	238.380 $\pm$ 2.234	8.579 $\pm$ 0.347	46.018 $\pm$ 1.460
3	0.000	0.667	0.333	293.271 $\pm$ 4.461	265.557 $\pm$ 2.254	1.673 $\pm$ 0.091	12.645 $\pm$ 0.603
4	0.167	0.417	0.417	271.866 $\pm$ 3.924	204.866 $\pm$ 1.484	8.358 $\pm$ 0.376	41.030 $\pm$ 1.262
5	0.167	0.667	0.167	303.053 $\pm$ 4.597	252.568 $\pm$ 2.077	1.137 $\pm$ 0.061	8.452 $\pm$ 0.456
6	0.167	0.167	0.667	255.849 $\pm$ 3.621	190.725 $\pm$ 1.712	20.591 $\pm$ 0.716	79.822 $\pm$ 2.171
7	0.333	0.333	0.333	277.919 $\pm$ 4.067	170.268 $\pm$ 1.090	10.336 $\pm$ 0.444	41.411 $\pm$ 1.142
8	0.333	0.667	0.000	318.731 $\pm$ 4.886	233.339 $\pm$ 1.416	2.333 $\pm$ 0.122	14.287 $\pm$ 0.522
9	0.333	0.000	0.667	257.665 $\pm$ 3.804	180.613 $\pm$ 1.501	26.890 $\pm$ 0.972	91.916 $\pm$ 2.189
10	0.417	0.167	0.417	274.509 $\pm$ 3.811	156.580 $\pm$ 0.855	17.946 $\pm$ 0.582	61.299 $\pm$ 1.417
11	0.417	0.417	0.167	295.013 $\pm$ 4.606	165.273 $\pm$ 1.043	10.146 $\pm$ 0.437	38.043 $\pm$ 0.981
12	0.500	0.000	0.500	271.839 $\pm$ 3.779	161.953 $\pm$ 0.967	26.510 $\pm$ 0.834	84.149 $\pm$ 1.625
13	0.500	0.500	0.000	319.368 $\pm$ 4.740	169.931 $\pm$ 0.933	12.206 $\pm$ 0.519	41.786 $\pm$ 1.020
14	0.667	0.167	0.167	305.466 $\pm$ 4.550	131.455 $\pm$ 1.471	24.318 $\pm$ 0.825	68.198 $\pm$ 1.316
15	0.667	0.000	0.333	294.434 $\pm$ 4.254	132.124 $\pm$ 1.165	26.997 $\pm$ 0.875	75.404 $\pm$ 1.397
16	0.667	0.333	0.000	319.245 $\pm$ 4.883	139.558 $\pm$ 1.394	22.194 $\pm$ 0.775	63.074 $\pm$ 1.302
17	1.000	0.000	0.000	318.366 $\pm$ 4.801	132.900 $\pm$ 1.553	31.289 $\pm$ 0.982	81.403 $\pm$ 1.369
18	0.000	1.000	0.000	318.740 $\pm$ 4.896	289.258 $\pm$ 2.659	0.619 $\pm$ 0.032	4.678 $\pm$ 0.163
19	0.000	0.000	1.000	252.114 $\pm$ 3.848	186.789 $\pm$ 1.988	26.585 $\pm$ 0.965	93.080 $\pm$ 2.321

Table F.2: Mean and 95% confidence interval of $avgCT$, $sdCT$, $avgTARD$, and $sdTARD$ of the weighted MOFCS+ rule for the $\{PI,DT\}$ problem configuration

ID of Weights	Weights			MOFCS+ {P1,DT}			
	W_{FIFO}	W_{CR}	W_{SSU+}	avgCT	sdCT	avgTARD	sdTARD
1	0.000	0.333	0.667	251.525 $\pm$ 4.050	194.222 $\pm$ 2.012	50.470 $\pm$ 1.268	116.647 $\pm$ 1.701
2	0.000	0.500	0.500	266.024 $\pm$ 4.582	193.778 $\pm$ 1.955	41.397 $\pm$ 1.355	93.130 $\pm$ 1.660
3	0.000	0.667	0.333	284.600 $\pm$ 4.887	191.241 $\pm$ 0.789	32.787 $\pm$ 1.346	63.936 $\pm$ 1.258
4	0.167	0.417	0.417	270.545 $\pm$ 4.789	171.365 $\pm$ 1.119	38.904 $\pm$ 1.341	80.046 $\pm$ 1.398
5	0.167	0.667	0.167	298.290 $\pm$ 5.063	176.683 $\pm$ 0.707	35.833 $\pm$ 1.470	60.371 $\pm$ 1.178
6	0.167	0.167	0.667	250.638 $\pm$ 4.074	181.209 $\pm$ 1.712	53.515 $\pm$ 1.526	118.704 $\pm$ 1.914
7	0.333	0.333	0.333	279.445 $\pm$ 5.122	150.956 $\pm$ 0.721	43.784 $\pm$ 1.564	79.757 $\pm$ 1.323
8	0.333	0.667	0.000	316.489 $\pm$ 5.536	166.639 $\pm$ 1.025	47.196 $\pm$ 1.865	72.137 $\pm$ 1.203
9	0.333	0.000	0.667	251.767 $\pm$ 4.146	177.036 $\pm$ 1.624	59.906 $\pm$ 1.651	127.032 $\pm$ 2.085
10	0.417	0.167	0.417	273.715 $\pm$ 4.806	148.613 $\pm$ 0.895	51.394 $\pm$ 1.657	95.597 $\pm$ 1.565
11	0.417	0.417	0.167	296.373 $\pm$ 5.242	146.329 $\pm$ 0.986	47.897 $\pm$ 1.630	79.226 $\pm$ 1.172
12	0.500	0.000	0.500	266.515 $\pm$ 4.585	161.095 $\pm$ 1.241	60.936 $\pm$ 1.783	118.107 $\pm$ 1.916
13	0.500	0.500	0.000	316.337 $\pm$ 5.409	147.935 $\pm$ 1.319	55.857 $\pm$ 1.884	85.113 $\pm$ 1.135
14	0.667	0.167	0.167	302.423 $\pm$ 5.152	132.320 $\pm$ 1.363	62.980 $\pm$ 1.822	99.306 $\pm$ 1.336
15	0.667	0.000	0.333	289.571 $\pm$ 5.079	133.273 $\pm$ 1.119	63.233 $\pm$ 1.894	104.931 $\pm$ 1.559
16	0.667	0.333	0.000	316.350 $\pm$ 5.558	137.223 $\pm$ 1.603	64.599 $\pm$ 2.061	97.458 $\pm$ 1.307
17	1.000	0.000	0.000	317.429 $\pm$ 4.399	133.078 $\pm$ 1.573	74.959 $\pm$ 1.822	112.237 $\pm$ 1.155
18	0.000	1.000	0.000	317.563 $\pm$ 4.157	198.428 $\pm$ 0.837	42.263 $\pm$ 1.598	62.208 $\pm$ 1.130
19	0.000	0.000	1.000	249.777 $\pm$ 4.043	185.748 $\pm$ 1.712	61.607 $\pm$ 1.767	134.234 $\pm$ 2.138

Table F.3: Mean and 95% confidence interval of $avgCT$, $sdCT$, $avgTARD$, and $sdTARD$ of the weighted MOFCS rule for the $\{PI,DL\}$ problem configuration

ID of Weights	Weights			MOFCS {P1,DL}			
	W_{FIFO}	W_{CR}	W_{SSU}	avgCT	sdCT	avgTARD	sdTARD
1	0.000	0.333	0.667	254.900 $\pm$ 3.677	206.617 $\pm$ 2.173	14.683 $\pm$ 0.592	59.919 $\pm$ 1.592
2	0.000	0.500	0.500	260.300 $\pm$ 3.861	213.674 $\pm$ 2.208	11.994 $\pm$ 0.534	51.922 $\pm$ 1.489
3	0.000	0.667	0.333	285.252 $\pm$ 4.048	255.700 $\pm$ 2.378	1.834 $\pm$ 0.121	13.030 $\pm$ 0.715
4	0.167	0.417	0.417	263.416 $\pm$ 3.928	193.092 $\pm$ 1.783	9.904 $\pm$ 0.479	44.015 $\pm$ 1.437
5	0.167	0.667	0.167	296.291 $\pm$ 4.613	246.997 $\pm$ 2.105	1.176 $\pm$ 0.058	8.627 $\pm$ 0.377
6	0.167	0.167	0.667	254.106 $\pm$ 3.745	180.192 $\pm$ 1.451	19.474 $\pm$ 0.722	71.509 $\pm$ 1.760
7	0.333	0.333	0.333	268.742 $\pm$ 3.900	165.279 $\pm$ 1.257	11.303 $\pm$ 0.465	44.162 $\pm$ 1.241
8	0.333	0.667	0.000	318.731 $\pm$ 4.886	233.339 $\pm$ 1.416	2.333 $\pm$ 0.122	14.287 $\pm$ 0.522
9	0.333	0.000	0.667	254.318 $\pm$ 3.677	172.208 $\pm$ 1.476	25.077 $\pm$ 0.846	83.484 $\pm$ 1.979
10	0.417	0.167	0.417	264.688 $\pm$ 3.856	155.829 $\pm$ 0.870	17.591 $\pm$ 0.666	61.156 $\pm$ 1.342
11	0.417	0.417	0.167	286.101 $\pm$ 4.190	161.881 $\pm$ 1.075	9.553 $\pm$ 0.399	36.551 $\pm$ 0.988
12	0.500	0.000	0.500	260.225 $\pm$ 3.727	164.227 $\pm$ 1.406	24.903 $\pm$ 0.840	81.207 $\pm$ 2.001
13	0.500	0.500	0.000	319.368 $\pm$ 4.740	169.931 $\pm$ 0.933	12.206 $\pm$ 0.519	41.786 $\pm$ 1.020
14	0.667	0.167	0.167	298.826 $\pm$ 4.367	129.384 $\pm$ 1.378	23.089 $\pm$ 0.761	65.958 $\pm$ 1.245
15	0.667	0.000	0.333	284.384 $\pm$ 4.038	129.996 $\pm$ 0.970	25.040 $\pm$ 0.792	72.186 $\pm$ 1.380
16	0.667	0.333	0.000	319.245 $\pm$ 4.883	139.558 $\pm$ 1.394	22.194 $\pm$ 0.775	63.074 $\pm$ 1.302
17	1.000	0.000	0.000	318.366 $\pm$ 4.801	132.900 $\pm$ 1.553	31.289 $\pm$ 0.982	81.403 $\pm$ 1.369
18	0.000	1.000	0.000	318.740 $\pm$ 4.896	289.258 $\pm$ 2.659	0.619 $\pm$ 0.032	4.678 $\pm$ 0.163
19	0.000	0.000	1.000	252.946 $\pm$ 3.945	171.935 $\pm$ 1.797	24.702 $\pm$ 0.939	82.614 $\pm$ 2.078

Table F.4: Mean and 95% confidence interval of $avgCT$, $sdCT$, $avgTARD$, and $sdTARD$ of the weighted MOFCS rule for the $\{P1,DT\}$ problem configuration

ID of Weights	Weights			MOFCS $\{P1,DT\}$			
	w_{FIFO}	w_{CR}	w_{SSU}	$avgCT$	$sdCT$	$avgTARD$	$sdTARD$
1	0.000	0.333	0.667	248.613 $\pm$ 4.031	185.357 $\pm$ 1.626	49.306 $\pm$ 1.528	106.389 $\pm$ 1.656
2	0.000	0.500	0.500	253.707 $\pm$ 3.921	184.816 $\pm$ 1.710	44.836 $\pm$ 1.489	97.614 $\pm$ 2.006
3	0.000	0.667	0.333	275.283 $\pm$ 4.698	186.389 $\pm$ 1.116	30.447 $\pm$ 1.205	61.298 $\pm$ 1.205
4	0.167	0.417	0.417	260.009 $\pm$ 4.371	167.035 $\pm$ 1.183	37.824 $\pm$ 1.377	79.067 $\pm$ 1.481
5	0.167	0.667	0.167	290.435 $\pm$ 5.021	174.210 $\pm$ 0.691	32.526 $\pm$ 1.379	57.497 $\pm$ 1.121
6	0.167	0.167	0.667	248.248 $\pm$ 4.016	173.879 $\pm$ 1.524	52.400 $\pm$ 1.633	110.733 $\pm$ 1.791
7	0.333	0.333	0.333	268.267 $\pm$ 4.612	148.580 $\pm$ 0.752	40.011 $\pm$ 1.347	76.522 $\pm$ 1.201
8	0.333	0.667	0.000	316.489 $\pm$ 5.536	166.639 $\pm$ 1.025	47.196 $\pm$ 1.865	72.137 $\pm$ 1.203
9	0.333	0.000	0.667	248.454 $\pm$ 4.025	170.697 $\pm$ 1.793	57.432 $\pm$ 1.754	118.191 $\pm$ 1.964
10	0.417	0.167	0.417	262.050 $\pm$ 4.482	146.607 $\pm$ 0.985	47.428 $\pm$ 1.550	91.805 $\pm$ 1.421
11	0.417	0.417	0.167	287.687 $\pm$ 5.222	144.228 $\pm$ 0.832	44.279 $\pm$ 1.574	75.988 $\pm$ 1.232
12	0.500	0.000	0.500	253.990 $\pm$ 4.044	163.095 $\pm$ 1.249	57.419 $\pm$ 1.639	115.138 $\pm$ 1.455
13	0.500	0.500	0.000	316.337 $\pm$ 5.409	147.935 $\pm$ 1.319	55.857 $\pm$ 1.884	85.113 $\pm$ 1.135
14	0.667	0.167	0.167	294.632 $\pm$ 5.108	130.486 $\pm$ 1.212	59.276 $\pm$ 1.777	95.974 $\pm$ 1.318
15	0.667	0.000	0.333	279.216 $\pm$ 4.780	130.876 $\pm$ 1.003	58.459 $\pm$ 1.712	100.462 $\pm$ 1.436
16	0.667	0.333	0.000	316.350 $\pm$ 5.558	137.223 $\pm$ 1.603	64.599 $\pm$ 2.061	97.458 $\pm$ 1.307
17	1.000	0.000	0.000	317.429 $\pm$ 4.399	133.078 $\pm$ 1.573	74.959 $\pm$ 1.822	112.237 $\pm$ 1.155
18	0.000	1.000	0.000	317.563 $\pm$ 4.157	198.428 $\pm$ 0.837	42.263 $\pm$ 1.598	62.208 $\pm$ 1.130
19	0.000	0.000	1.000	250.719 $\pm$ 4.070	171.362 $\pm$ 1.537	58.424 $\pm$ 1.755	119.664 $\pm$ 1.894

Table F.5: Comparing MOFCS+ and MOFCS rules for the $\{P1,DL\}$ and $\{P1,DT\}$ problem configurations using the concept of domination across three objectives ($avgCT$, $sdCT$, and $avgTARD$)

Set ID of Weights	Weights			$\{P1,DL\}$		$\{P1,DT\}$	
	w_{FIFO}	w_{CR}	w_{SSU+}	MOFCS+	MOFCS	MOFCS+	MOFCS
1	0.000	0.333	0.667	* [1]	✓	* [1]	✓
2	0.000	0.500	0.500	* (4)	✓	* [4]	✓
3	0.000	0.667	0.333	✓	✓	* [3]	✓
4	0.167	0.417	0.417	✓	✓	* [4]	✓
5	0.167	0.667	0.167	✓	✓	* [5]	✓
6	0.167	0.167	0.667	* [6]	✓	* [6]	✓
7	0.333	0.333	0.333	✓	✓	* [7]	✓
8	0.333	0.667	0.000	✓	✓	* (7) & [7]	* (7,8) & [7,11]
9	0.333	0.000	0.667	* [6,9,19]	* [19]	* [6,9,19]	✓
10	0.417	0.167	0.417	* [10]	✓	* [7,10]	✓
11	0.417	0.417	0.167	* [11]	✓	* [11]	✓
12	0.500	0.000	0.500	* [10]	✓	* [10]	✓
13	0.500	0.500	0.000	* (11) & [7]	* (11,13) & [7,11]	* (11) & [10]	* (11,13) & [10,11]
14	0.667	0.167	0.167	* [14]	✓	* [14]	✓
15	0.667	0.000	0.333	* [15]	✓	* [15]	✓
16	0.667	0.333	0.000	✓	✓	* (14,15) & [14]	* (14,15,16) & [14,15]
17	1.000	0.000	0.000	* (14,15) & [14]	* (14,15,17) & [14,15]	* (14) & [14]	* (14,17) & [14,15]
18	0.000	1.000	0.000	✓	✓	* (2,3,4,5) & [3]	* (2,3,4,5,18) & [3,4,5,7]
19	0.000	0.000	1.000	✓	✓	* [1,6,9]	* [9]

Figure F.1: Almost no correlation between the average cycle time and standard deviation of cycle time objectives for $\{P1, DL\}$ problem configuration

Figure F.2: Almost no correlation between the average cycle time and average tardiness objectives for $\{P1, DL\}$ problem configuration

Figure F.3: Negative correlation between the average cycle time and standard deviation of tardiness objectives for $\{P1, DL\}$ problem configuration

Figure F.4: Negative correlation between the standard deviation of cycle time and average tardiness objectives for $\{PI, DL\}$ problem configuration

Figure F.5: Negative correlation between the standard deviation of cycle time and standard deviation of tardiness objectives for $\{PI, DL\}$ problem configuration

Figure F.6: Almost no correlation between the average cycle time and standard deviation of cycle time objectives for $\{PI, DT\}$ problem configuration

Figure F.7: Almost no correlation between the average cycle time and average tardiness objectives for $\{PI, DT\}$ problem configuration

Figure F.8: Negative correlation between the average cycle time and standard deviation of tardiness objectives for $\{PI, DT\}$ problem configuration

Figure F.9: Negative correlation between the standard deviation of cycle time and average tardiness objectives for $\{PI, DT\}$ problem configuration

Figure F.10: Negative correlation between the standard deviation of cycle time and standard deviation of tardiness objectives for $\{PI, DT\}$ problem configuration

APPENDIX G

SIMULATION RESULTS FOR COMPARISON OF RSM & LASGAD

G.1 FITTING META-MODEL FOR THE AVERAGE CYCLE TIME RESPONSE

Table G.1: Sequential model sum of squares for the average cycle time (*avgCT*) response

Source	Sum of Squares	DF	Mean Square	F Value	Prob > F
Mean	1522792.961	1	1522792.961		
Linear	10433.220	2	5216.609864	47.47747	< 0.0001
Quadratic	1451.460	3	483.8198574	20.51768	< 0.0001
Special Cubic	101.152	1	101.1524255	5.909707	0.0317 Suggested
Cubic	45.651	3	15.21714311	0.857334	0.4974
Residual	159.744	9	17.74937568		
Total	1534984.189	19	80788.64152		

Table G.2: ANOVA for mixture special cubic model for the average cycle time (*avgCT*) response

Source	Sum of Squares	DF	Mean Square	F Value	Prob > F
Model	11985.832	6	1997.639	116.71	< 0.0001 significant
Linear Mixture	10941.349	2	5470.674	319.617	< 0.0001
AB	2.696	1	2.696	0.157	0.6984
AC	476.082	1	476.082	27.815	0.0002
BC	464.553	1	464.553	27.141	0.0002
ABC	101.152	1	101.152	5.91	0.0317
Residual	205.396	12	17.116		
Cor Total	12191.228	18			
Std. Dev.	4.137		R-Squared	0.983	
Mean	283.102		Adj R-Squared	0.975	
C.V.	1.461		Pred R-Squared	0.921	
PRESS	965.370		Adeq Precision	27.757	(>4, therefore desirable)

The Model F-value of 116.71 implies the model is significant.

1. A: w_{FIFO} , B: w_{CR} , C: w_{SSU}

2. Values of "Prob > F" less than 0.0500 indicate model terms are significant.

3. Values greater than 0.1000 indicate the model terms are not significant.

4. C.V. – Coefficient of Variation & PRESS – Predicted Residual Error Sum of Squares

The fitted meta-model for the average cycle time (*avgCT*) response is:

$$\begin{aligned} avgCT = & 320.994w_{FIFO} + 320.548w_{CR} + 251.292w_{SSU} \\ & -6.297w_{FIFO}w_{CR} - 83.682w_{FIFO}w_{SSU} - 82.663w_{CR}w_{SSU} \\ & -220.095w_{FIFO}w_{CR}w_{SSU} \end{aligned} \quad (G.1)$$

G.2 FITTING META-MODEL FOR THE STANDARD DEVIATION OF CYCLE TIME RESPONSE

Table G.3: Sequential model sum of squares for the standard deviation of cycle time (*sdCT*) response

Source	Sum of Squares	DF	Mean Square	F Value	Prob > F	
Mean	649164.4234	1	649164.4234			
Linear	31892.61769	2	15946.30885	49.07795	< 0.0001	Suggested
Quadratic	2577.50242	3	859.1674734	4.261116	0.0266	
Special Cubic	297.3674367	1	297.3674367	1.53558	0.2390	
Cubic	1655.238535	3	551.7461783	7.42726	0.0083	Suggested
Residual	668.5797611	9	74.28664012			
Total	686255.7293	19	36118.72259			

Table G.4: ANOVA for mixture cubic model [Not Reduced] for the standard deviation of cycle time (*sdCT*) response

Source	Sum of Squares	DF	Mean Square	F Value	Prob > F	
Model	36422.726	9	4046.970	54.478	< 0.0001	significant
Linear Mixture	31892.618	2	15946.309	214.659	< 0.0001	
AB	1251.881	1	1251.881	16.852	0.0027	
AC	21.506	1	21.506	0.289	0.6036	
BC	71.562	1	71.562	0.963	0.3520	
ABC	297.367	1	297.367	4.003	0.0765	
AB(A-B)	952.072	1	952.072	12.816	0.0059	
AC(A-C)	390.561	1	390.561	5.257	0.0475	
BC(B-C)	82.911	1	82.911	1.116	0.3183	
Residual	668.580	9	74.287			
Cor Total	37091.306	18				
Std. Dev.	8.619		R-Squared	0.982		
Mean	184.842		Adj R-Squared	0.964		
C.V.	4.663		Pred R-Squared	0.039		
PRESS	35660.378		Adeq Precision	27.240	(>4, therefore desirable)	

The Model F-value of 54.48 implies the model is significant.

The fitted meta-model for the standard deviation of cycle time ($sdCT$) response based on the mixture cubic model (not reduced) (with reference to Table G.4) is:

$$\begin{aligned}
 sdCT = & 133.349w_{FIFO} + 291.826w_{CR} + 172.742w_{SSU} \\
 & -136.629w_{FIFO}w_{CR} + 17.908w_{FIFO}w_{SSU} - 32.666w_{CR}w_{SSU} \\
 & -377.372w_{FIFO}w_{CR}w_{SSU} - 260.543w_{FIFO}w_{CR}(w_{FIFO} - w_{CR}) \\
 & -166.874w_{FIFO}w_{SSU}(w_{FIFO} - w_{SSU}) + 76.886w_{CR}w_{SSU}(w_{CR} - w_{SSU})
 \end{aligned} \quad (G.2)$$

Table G.5: ANOVA for mixture [Reduced] cubic model for the standard deviation of cycle time ($sdCT$) response

Source	Sum of Squares	DF	Mean Square	F Value	Prob > F	
Model	35313.101	4	8828.275	69.506	< 0.0001	significant
Linear Mixture	31892.618	2	15946.309	125.547	< 0.0001	
AB	2025.4	1	2025.4	15.946	0.0013	
AB(A-B)	1395.084	1	1395.084	10.984	0.0051	
Residual	1778.205	14	127.015			
Cor Total	37091.306	18				
Std. Dev.	11.270		R-Squared	0.952		
Mean	184.842		Adj R-Squared	0.938		
C.V.	6.097		Pred R-Squared	0.911		
PRESS	3306.312		Adeq Precision	27.364	(>4, therefore desirable)	
The Model F-value of 69.51 implies the model is significant.						

The fitted meta-model for the standard deviation of cycle time ($sdCT$) response based on the reduced mixture cubic model (with reference to Table G.5) is:

$$\begin{aligned}
 sdCT = & 132.728w_{FIFO} + 285.662w_{CR} + 168.747w_{SSU} \\
 & -150.968w_{FIFO}w_{CR} - 306.463w_{FIFO}w_{CR}(w_{FIFO} - w_{CR})
 \end{aligned} \quad (G.3)$$

‘Not Reduced’ implies that the fitted meta-model is based on the highest order polynomial (where the additional terms are significant and the model is not aliased) while ‘Reduced’ implies that the fitted meta-model is a reduced model in which insignificant model terms (not counting those required to support hierarchy) are removed to potentially improve the model (resulting in a larger F Value).

G.3 FITTING META-MODEL FOR THE AVERAGE TARDINESS RESPONSE

Table G.6: Sequential model sum of squares for the average tardiness (*avgTARD*) response

Source	Sum of Squares	DF	Mean Square	F Value	Prob > F	
Mean	4394.788246	1	4394.788246			
Linear	1455.758025	2	727.8790123	70.51017	< 0.0001	Suggested
Quadratic	71.55634961	3	23.85211654	3.312361	0.0540	
Special Cubic	10.61503409	1	10.61503409	1.534755	0.2391	
Cubic	66.88415986	3	22.29471995	12.4528	0.0015	Suggested
Residual	16.11304127	9	1.790337919			
Total	6015.714855	19	316.6165713			

Table G.7: ANOVA for mixture cubic model (not reduced) for the average tardiness (*avgTARD*) response

Source	Sum of Squares	DF	Mean Square	F Value	Prob > F	
Model	1604.814	9	178.313	99.597	< 0.0001	significant
Linear Mixture	1455.758	2	727.879	406.560	< 0.0001	
AB	16.445	1	16.445	9.186	0.0142	
AC	12.392	1	12.392	6.922	0.0273	
BC	8.821	1	8.821	4.927	0.0536	
ABC	10.615	1	10.615	5.929	0.0377	
AB(A-B)	47.759	1	47.759	26.676	0.0006	
AC(A-C)	3.454	1	3.454	1.929	0.1983	
BC(B-C)	9.826	1	9.826	5.488	0.0438	
Residual	16.113	9	1.790			
Cor Total	1620.927	18				
Std. Dev.	1.338		R-Squared	0.990		
Mean	15.209		Adj R-Squared	0.980		
C.V.	8.798		Pred R-Squared	0.728		
PRESS	441.446		Adeq Precision	32.205	(>4, therefore desirable)	

The Model F-value of 99.6 implies the model is significant.

The fitted meta-model for the average tardiness (*avgTARD*) response based on the mixture cubic model (not reduced) (with reference to Table G.7) is:

$$\begin{aligned}
avgTARD = & 31.447w_{FIFO} + 0.405w_{CR} + 24.590w_{SSU} \\
& -15.660w_{FIFO}w_{CR} - 13.594w_{FIFO}w_{SSU} - 11.469w_{CR}w_{SSU} \\
& -71.299w_{FIFO}w_{CR}w_{SSU} + 58.354w_{FIFO}w_{CR}(w_{FIFO} - w_{CR}) \\
& -15.693w_{FIFO}w_{SSU}(w_{FIFO} - w_{SSU}) - 26.469w_{CR}w_{SSU}(w_{CR} - w_{SSU})
\end{aligned} \tag{G.4}$$

Table G.8: ANOVA for mixture reduced cubic model for the average tardiness (*avgTARD*) response

Source	Sum of Squares	DF	Mean Square	F Value	Prob > F	
Model	1601.36	8	200.17	102.3	< 0.0001	significant
Linear Mixture	1455.758	2	727.879	371.995	< 0.0001	
AB	15.403	1	15.403	7.872	0.0186	
AC	12.392	1	12.392	6.333	0.0306	
BC	9.742	1	9.742	4.979	0.0497	
ABC	10.615	1	10.615	5.425	0.0421	
AB(A-B)	45.127	1	45.127	23.063	0.0007	
BC(B-C)	11.74	1	11.74	6	0.0343	
Residual	19.567	10	1.957			
Cor Total	1620.927	18				
Std. Dev.	1.399		R-Squared	0.988		
Mean	15.209		Adj R-Squared	0.978		
C.V.	9.197		Pred R-Squared	0.803		
PRESS	318.514		Adeq Precision	32.008	(>4, therefore desirable)	
The Model F-value of 102.30 implies the model is significant.						

The fitted meta-model for the average tardiness (*avgTARD*) response based on the reduced mixture cubic model (with reference to Table G.8) is:

$$\begin{aligned}
avgTARD = & 31.000w_{FIFO} + 0.405w_{CR} + 25.037w_{SSU} \\
& -15.111w_{FIFO}w_{CR} - 13.594w_{FIFO}w_{SSU} - 12.017w_{CR}w_{SSU} \\
& -71.299w_{FIFO}w_{CR}w_{SSU} + 56.172w_{FIFO}w_{CR}(w_{FIFO} - w_{CR}) \\
& -28.651w_{CR}w_{SSU}(w_{CR} - w_{SSU})
\end{aligned} \tag{G.5}$$

G.4 NUMERICAL OPTIMIZATION RESULTS

G.1.1 OPTIMIZATION FOR CRITERIA WITH SPECIFIED TARGETS

Table G.9: Values used in the desirability function for criterion with specified target

Variables	Criterion x_i		
	avgCT	sdCT	avgTARD
Lower bound, $L(x_i)$	30	0	0
Upper bound, $U(x_i)$	350	300	70
Target value, $T(x_i)$	296.291	246.997	1.176
ω_1	0.1	0.1	0.1
ω_2	1	1	1

Table G.10: Numerical optimization solutions based on meta-models that are not reduced and values from Table G.9

Solution Number	Weights			Predicted Performance			Predicted Total Desirability	
	w_{FIFO}	w_{CR}	w_{SSU}	avgCT	sdCT	avgTARD		
1	0.08	0.68	0.25	284.981	246.997	1.17602	0.999	Selected
2	0.23	0.63	0.14	296.292	227.014	1.17601	0.997	
3	0.18	0.63	0.19	288.801	230.366	1.17634	0.997	
4	0.20	0.63	0.18	290.494	228.851	1.17628	0.997	

Table G.11: Numerical optimization solutions based on meta-models that are reduced (where possible) and values from Table G.9

Solution Number	Weights			Predicted Performance			Predicted Total Desirability	
	w_{FIFO}	w_{CR}	w_{SSU}	avgCT	sdCT	avgTARD		
1	0.07	0.68	0.25	284.596	246.997	1.176	0.99850	Selected
2	0.10	0.66	0.25	284.352	243.248	1.1765	0.99796	
3	0.23	0.63	0.14	296.291	230.595	1.176	0.99771	
4	0.16	0.63	0.21	286.739	235.907	1.17756	0.99725	
5	0.20	0.63	0.18	290.447	232.525	1.17697	0.99725	
6	0.00	0.98	0.02	317.315	283.145	0.104455	0.53357	

G.1.2 OPTIMIZATION FOR CRITERIA TO BE MINIMIZED

Table G.12: Values used in the desirability function for criterion to be minimized

Variables	Criterion x_i		
	avgCT	sdCT	avgTARD
Lower bound, $L(x_i)$	30	0	0
Upper bound, $U(x_i)$	350	300	70
ω	1	1	1

Table G.13: Numerical optimization solutions based on meta-models that are not reduced and values from Table G.12

Solution Number	Weights			Predicted Performance			Predicted Total Desirability	
	w_{FIFO}	w_{CR}	w_{SSU}	avgCT	sdCT	avgTARD		
1	0.37	0.20	0.43	263.19	156.943	16.935	0.46115	Selected
2	0.00	0.09	0.91	250.75	175.377	23.3452	0.44118	

Table G.14: Numerical optimization solutions based on meta-models that are reduced (where possible) and values from Table G.12

Solution Number	Weights			Predicted Performance			Predicted Total Desirability	
	w_{FIFO}	w_{CR}	w_{SSU}	avgCT	sdCT	avgTARD		
1	0.31	0.12	0.57	256.009	163.868	19.8204	0.45716	Selected

APPENDIX H

SIMULATION RESULTS FOR THE INDUSTRIAL CASE STUDY

Table H.1: Mean and 95% confidence interval for different combination of the decision policies with the dispatching time policy

Methodology	avgCT	sdCT	avgTARD	D*	#Dominated by Dispatching-
Dispatching - FIFO	281.949	81.876	181.912	0.096	EDD, CR, LS, MOFCS+CR
Dispatching - SPT	98.084	93.651	26.855	-0.575	Not dominated
Dispatching - EDD	199.222	36.423	99.568	0.902	MOFCS+CR
Dispatching - CR	199.299	50.781	112.464	0.772	EDD, MOFCS+CR
Dispatching - LS	199.505	37.056	99.794	0.896	EDD, MOFCS+CR
Dispatching - SSU	298.201	68.976	198.395	0.134	EDD, CR, LS, MOFCS+CR
Dispatching - WINQ	214.844	133.099	126.143	-0.701	EDD, CR, LS, MOFCS+CR
Dispatching – MOFCS**+CR	198.505	36.382	98.446	0.908	Not dominated

* D – Total Desirability (the higher the value, the more desirable the methodology; negative value implies methodology is performing worst than the specified bounds – highly undesirable and should not be used).

** Weights {wFIFO = 0.10, wCR = 0.86, wSSU = 0.04} are applied in the MOFCS rule

“Not dominated” implies the methodology is not dominated by any methodology listed in the table.

Table H.2: Mean and 95% confidence interval for different combination of the decision policies with the NEXTS time policy

Methodology	avgCT	sdCT	avgTARD	D*	#Dominated by NEXTS -
NEXTS - FIFO	65.766	9.240	1.542	0.613	MOFCS+CR (1/2/3)
NEXTS - SPT	59.031	9.711	0.870	0.678	Not dominated
NEXTS - EDD	71.614	13.698	0.172	0.372	CR, LS
NEXTS - CR	70.302	13.601	0.022	0.394	Not dominated
NEXTS - LS	66.946	11.831	0.107	0.542	Not dominated
NEXTS - SSU	68.727	11.173	2.053	0.492	FIFO, SPT, MOFCS+CR (1/2/3)
NEXTS - WINQ	63.990	13.461	1.203	0.414	SPT, MOFCS+CR(2)
NEXTS - MOFCS+CR (1) ¹	64.315	7.159	0.534	0.742	Not dominated
NEXTS - MOFCS+CR (2) ²	63.890	6.802	0.654	0.751	Not dominated
NEXTS - MOFCS+CR (3) ³	64.336	7.158	0.653	0.736	Not dominated

* D – Total Desirability (the higher the value, the more desirable the methodology).

¹ MOFCS+CR (1) shows the performance of the entire fab when weights {wFIFO = 0.64, wCR = 0.36, wSSU = 0.0}, optimized in Quarter 1 (Q1), are applied in the MOFCS rule throughout the four quarters

² MOFCS+CR (2) shows the performance of the entire fab when weights {wFIFO = 0.64, wCR = 0.36, wSSU = 0.0}, optimized in Q1, are applied in the MOFCS rule for Q1 and Q2; and weights {wFIFO = 0.0, wCR = 0.0, wSSU = 1.0}, optimized in Q3, are applied in the MOFCS rule for Q3 and Q4.

³ MOFCS+CR (3) shows the performance of the entire fab when weights {wFIFO = 0.64, wCR = 0.36, wSSU = 0.0}, optimized in Q1, are applied in the MOFCS rule for Q1 and Q2; and weights {wFIFO = 0.0, wCR = 0.0, wSSU = 1.0}, optimized in Q3, are applied in the MOFCS rule for Q3; and weights {wFIFO = 0.56, wCR = 0.27, wSSU = 0.17}, optimized in Q4, are applied in the MOFCS rule for Q4.

“Not dominated” implies the methodology is not dominated by any methodology listed in the table.