University of Tennessee, Knoxville
TRACE: Tennessee Research and Creative Exchange

# The Integration of Maintenance Decisions and Flow Shop Scheduling 

Javad Seif<br>University of Tennessee, jseif@utk.edu

Follow this and additional works at: https://trace.tennessee.edu/utk_graddiss

## Recommended Citation

Seif, Javad, "The Integration of Maintenance Decisions and Flow Shop Scheduling. " PhD diss., University of Tennessee, 2018.
https://trace.tennessee.edu/utk_graddiss/4965

This Dissertation is brought to you for free and open access by the Graduate School at TRACE: Tennessee Research and Creative Exchange. It has been accepted for inclusion in Doctoral Dissertations by an authorized administrator of TRACE: Tennessee Research and Creative Exchange. For more information, please contact trace@utk.edu.

To the Graduate Council:
I am submitting herewith a dissertation written by Javad Seif entitled "The Integration of Maintenance Decisions and Flow Shop Scheduling." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Industrial Engineering.

Andrew J. Yu, Major Professor
We have read this dissertation and recommend its acceptance:
Reza Abedi, Oleg Shylo, James L. Simonton
Accepted for the Council:
Dixie L. Thompson
Vice Provost and Dean of the Graduate School
(Original signatures are on file with official student records.)

The Integration of Maintenance Decisions and Flow Shop Scheduling

A Dissertation Presented for the Doctor of Philosophy<br>Degree<br>The University of Tennessee, Knoxville

Copyright © 2018 by Javad Seif All rights reserved.

## DEDICATION

To my mother
Touran Nemati

## ACKNOWLEDGEMENTS

I would like to thank the members of my dissertation committee for taking time to review this work: Dr. Andrew Yu, Dr. James Simonton, Dr. Oleg Shylo, and Dr. Reza Abedi. I would like to especially thank my advisor, Dr. Yu for his support throughout my Ph.D. program. I would also like to thank Dr. Shylo for his instructions in Stochastic Programming and Heuristic Methods in Optimization, and his valuable comments and suggestions. Special thanks to Dr. Simonton for his kind support, guidance, and valuable comments. Thanks to Dr. Reza Abedi for his time, valuable comments, and friendship.

Many thanks to Dr. Mohammad Dehaghani for his contributions in the problem definition, coding, and "Simulation-Optimization" of Chapter III. Fahimeh Rahmanniya has also contributed in problem definition and "Bi-Objective Formulation" in Chapter II.

I would like to express my gratitude and appreciation to all the faculty members of the Tickle College of Engineering who instructed me during my Ph.D. program. Special thanks to Dr. Jamie Coble, Assistant Professor in the Department of Nuclear Engineering, as I learned a great deal about Reliability and Maintainability Engineering from her classes.

I am thankful to Sarah Stone in the Graduate School for her time reviewing this dissertation and improving its formatting. Also, thanks to the UT Writing Center for helping me improve my writing skills.

I appreciate the financial support I received from the University of Tennessee Knoxville (UTK) and Space Institute (UTSI) during my Ph.D. program. I am grateful to all the staff of UTSI for being supportive and friendly, especially: Charlotte Henley, Mark Cross, Clara Furgosen, Laura Horton, Kaycee Edwards, Kanawful Massingille, Paula Morris, Robin Nee, and Dee Merriman.

I appreciate the support, love and encouragements I have received over many years of my education from all of my kind family members, relatives, and great friends. I am forever thankful to my parents: Reza Seif and Touran Nemati, my brother: Mahdi Seif, my cousin: Mohsen Seif, my high school math teacher: Dr. Farrokh Karroubi, and all of the teachers who directly or indirectly instructed me.

Finally, I would like to thank anyone who made my Ph.D. life easy and smooth. I am thankful to all of my UTSI friends. Special thanks to Brett Shields for his great friendship, research collaborations, support, and encouragements. Many thanks to Tullahoma local soccer players; playing soccer was one of the highlights of my life here in Tennessee.


#### Abstract

In the conventional production and service scheduling problems, it is assumed that the machines can continuously process the jobs and the information is complete and certain. However, in practice the machines must stop for preventive or corrective maintenance, and the information available to the planners can be both incomplete and uncertain. In this dissertation, the integration of maintenance decisions and production scheduling is studied in a permutation flow shop setting. Several variations of the problem are modeled as (stochastic) mixed-integer programs. In these models, some technical nuances are considered that increase the practicality of the models: having various types of maintenance, combining maintenance activities, and the impact of maintenance on the processing times of the production jobs. The solution methodologies involve studying the solution space of the problems, genetic algorithms, stochastic optimization, multi-objective optimization, and extensive computational experiments. The application of the problems and managerial implications are demonstrated through a case study in the earthmoving operations in construction projects.


## PREFACE

My dissertation topic was motivated by a common problem I had encountered in industry prior to my graduate studies. I designed and implemented computerized maintenance information software (CMMS) for various companies. It has been my observation in many cases that regardless of the complexity and comprehensiveness of maintenance plans and maintenance management software systems, preventive maintenance activities are very likely to be deferred, or refrained from, due to production priorities. In my research, I strive to solve this problem by integrating production scheduling and maintenance decisions. The outcome is a schedule that simultaneously optimizes both production and maintenance objectives.

In this dissertation, the integration of maintenance decisions and production scheduling is studied in a permutation flow shop setting, where a number of jobs (orders) are to be processed consecutively on a number of machines in series. The machines should undergo various types of maintenance after operating for certain number of hours. The objective is to minimize the tardiness of the jobs with respect to their due times, and minimize the maintenance costs. In the mathematical models and solution algorithms that are presented, I consider the technical nuances that increase the practicality of these models: having various types of maintenance activities, combining these activities, and how maintenance affects the performance of the machines.

Through extensive computational experiments and case studies, it is shown that the proposed models and solution methodologies are reliable, robust, and independent from commercial solvers. This independence facilitates the incorporation and automation of these solutions in the existing information systems found in manufacturing and service industries. By implementing the proposed solutions, manufacturing and service industries can 1) resolve potential conflicts between production and maintenance, 2) minimize maintenance costs, 3) improve the health of their assets, 4) increase the readiness and performance of the production lines, and 5) increase customer satisfaction through optimal production scheduling and timely deliveries. All of these benefits can be attained without reliance on commercial solvers that can be financially and computationally expensive to use.

Javad Seif
Tullahoma, TN
April, 2018

## TABLE OF CONTENTS

INTRODUCTION ..... 1
Background ..... 1
Maintenance Planning ..... 1
Flow Shop Scheduling ..... 3
Integrating Maintenance Decisions into Flow Shop Scheduling ..... 4
CHAPTER I Integrating multiple age-based maintenance activities into flow shop scheduling ..... 7
Abstract ..... 8
1.1 Introduction ..... 8
1.2 Problem Definition ..... 13
1.3 Mathematical Formulation ..... 15
1.4 Solution Approach ..... 22
Complexity of the problem ..... 22
Lower bounds ..... 23
Lower-Bound-Based Genetic Algorithm (LBGA) ..... 27
1.5 Computational Results ..... 34
Test Problem Generation ..... 34
Performance ..... 38
Gap Analysis ..... 38
1.6 Case Study (Earthmoving Operations) ..... 44
1.7 Conclusion and Future Research ..... 46
CHAPTER II Bi-objective optimization of the integrated flow shop and maintenance scheduling problem ..... 48
Abstract ..... 49
2.1 Introduction ..... 49
2.2 Mathematical Formulation ..... 52
Bi-Objective Formulation ..... 58
2.3 Solution Methodology ..... 65
Confining the Solution Space ..... 66
Designing a search in the Solution Space ..... 72
Global search by crossover ..... 72
Local search by mutation ..... 72
Setting the parameters ..... 74
2.4 Computational Results ..... 74
Test problems generation ..... 75
Computational experiment ..... 77
2.5 Case Study ..... 81
2.6 Conclusion and Future Research ..... 86
CHAPTER III Combinied maintenance activities in integrated flow shop and maintenance scheduling under uncertainity ..... 87
Abstract ..... 88
3.1 Introduction ..... 88
3.2 Problem Definition and Mathematical Formulation ..... 91
Combined Maintenance Activities ..... 91
Prolonged Processing Times ..... 95
The Stochastic Mixed-Integer Program (SMIP) ..... 96
3.3 Simulation-Optimization ..... 102
3.4 Computational Experiments ..... 106
Test Problem Generation ..... 106
Computational Experiments for the Population Size ..... 107
Computational Experiment for Performance Evaluation ..... 108
3.5 Case Study ..... 118
3.6 Conclusion and Future Research ..... 127
CONCLUSION ..... 128
Unplanned Maintenance ..... 129
Considering Risk ..... 129
Fatigue and Degradation of Machines and Their Components ..... 129
REFERENCES ..... 130
APPENDICES ..... 138
Appendix A: Related Published Works ..... 139
Appendix B: OPL Models Used in CPLEX Optimization Studio ..... 140
B.1: The Mixed-Integer Program in Chapter I ..... 141
B.2: The Mixed-Integer Program in Chapter II ..... 143
B.3: The Stochastic Mixed-Integer Program in Chapter III ..... 146
Appendix C: MATLAB Codes Used in Chapter I ..... 149
C.1: Test Problem Generator (CPLEX Data File Generator) ..... 150
C.2: Objective Function (Fitness Function) ..... 154
C.3: Mutation Function ..... 156
C.4: Crossover Function ..... 157
C.5: The Roulette Wheel Selection Function ..... 158
C.6: The GA ..... 159
Appendix D: MATLAB Codes Used in Chapter II ..... 163
D.1: Test Problem Generator Function ..... 164
D.2: Updating the Processing Times ..... 165
D.3: Crossover Function ..... 166
D.4: Mutation Function ..... 167
D.5: Roulette Wheel Selection Function ..... 168
D.6: Fitness Function ..... 169
D.7: Function for Calculating the Coefficients of the iFMOLP ..... 173
D.8: Algorithm 2 ..... 174
D.9: The GAs ..... 175
D.10: A Function for Writing CPLEX Data Files ..... 191
D.11: Automation of the Computational Experiments ..... 195
Appendix E: MATLAB Codes Used in Chapter III ..... 197
E.1: Computational Experiments Main File ..... 198
E.2: Test Problem Generator ..... 200
E.3: CPLEX Data File Writer ..... 203
E.4: Crossover Function ..... 209
E.5: Mutation Function ..... 210
E.6: Roulette Wheel Selection Function ..... 211
E.7: The GA and Related Functions ..... 212
VITA ..... 221

## LIST OF TABLES

Table 1. Experimental Design ..... 32
Table 2. Results of the experiment. ..... 35
Table 3. Analysis of variance for the GA parameters ..... 36
Table 4. Generation method of test problems. ..... 37
Table 5. Comparison between CPLEX and LBGA. ..... 39
Table 6. Increasing both population size and problem size ( $m=5, l=3$ ) ..... 41
Table 7. Maintenance intervals (hours) recommended by the equipment manufacturer (Caterpillar Inc.). ..... 44
Table 8. Operation requirements (days) x deterioration rates (hours), due dates (days), and penalty for one day delay ..... 45
Table 9. The optimal solution for the case study ..... 46
Table 10. Membership function $f g z g, g=1,2$ ..... 60
Table 11. Piecewise linear membership functions for the numerical example. ..... 62
Table 12. Piecewise linear membership functions for the improved results. ..... 63
Table 13. Solution comparison. ..... 64
Table 14. Different membership values for (z1, f1 (z1)) and (z2,f2 (z2)). ..... 64
Table 15. Objective function values for the optimal solutions. ..... 65
Table 16. Generation method of test problems, adapted from Yu and Seif (2016).75
Table 17. Piecewise linear membership functions for test problems ..... 76
Table 18. Comparison of computation times (seconds) for problems with a realistic size. ..... 78
Table 19. Comparison of feasibility success (percentage) for problems with a realistic size. ..... 79
Table 20. Comparison of quality of objective function value (OFV, satisfaction degree) for problems with a realistic size ..... 80
Table 21. Maintenance Intervals (hours) Recommended by the equipment manufacturer (Caterpillar Inc.), reproduced from Yu and Seif (2016). ..... 82
Table 22. Operation requirements (days) $\times$ deterioration rates (hours) and due dates (days), reproduced from Yu and Seif (2016) ..... 83
Table 23. Piecewise linear membership functions for the case study. ..... 84
Table 24. Optimal solution for the case study. ..... 85
Table 25. All possible combinations for three types of maintenance activities. ..... 93
Table 26. The values for the GA parameters. ..... 104
Table 27. An example for the solution represtaion of the GA. ..... 104
Table 28. The impact of population size on the performance of the algorithm. . ..... 107
Table 29. Comparing the simulation-optimization method with CPLEX for $\mathrm{n}=4$. ..... 111
Table 30. Comparing the simulation-optimization method with CPLEX when problem size increases ..... 112
Table 31. Comparing the simulation-optimization method with CPLEX for $\mathrm{n}=10$.113
Table 32. Sensitivity of the gap to the input data, $n=4$ ..... 114
Table 33. Sensitivity of the solution time to the input data, $n=4$. ..... 115
Table 34. Analysis of variance for the gap sensitivity experiment ..... 116
Table 35. Analysis of variance for the experiment on solution time sensitivity. ..... 117
Table 36. Maintenance intervals (in hours) recommended by the equipment manufacturer (Caterpillar, 2010a, 2010b, 2010c), reproduced from (Yu \& Seif, 2016). ..... 119
Table 37. Processing times, due dates, and penalty costs for the jobs, adapted from Yu and Seif (2016) ..... 120
Table 38. Task list of each MA for the excavator (excerpts from Caterpillar (2010c)). ..... 121
Table 39. Two tasks with similar steps (excerpts from Caterpillar (2010c)) ..... 122
Table 40. The optimal solution for the case study. ..... 125
Table 41. The optimal solution, when the durations of the MAs do not change in combinations ..... 126

## LIST OF FIGURES

Figure 1. Age-based maintenance in taxonomy of maintenance philosophies
(Kothamasu et al., 2006). ..... 2
Figure 2. An example for a permutation flow shop scheduling problem, $n=m=3$... 3
Figure 3. Dissertation outline ..... 6
Figure 4. An illustrative example of the problem. ..... 14
Figure 5. Lower-bound-based GA (LBGA) ..... 33
Figure 6. The effect of increasing population size on the performance of the algorithm ..... 43
Figure 7. The value of the two objectives and $L$ against different membership values ..... 65
Figure 8. Maintainability block diagram for combined maintenance activities ..... 92
Figure 9. Overview of the simulation-optimization method ..... 103
Figure 10. Chromosome representation using real numbers ..... 105
Figure 11. The impact of population size on the objective function ..... 109
Figure 12. The impact of population time on solution time ..... 110

## INTRODUCTION

This dissertation is concerned with the integration of maintenance decisions in flow shop scheduling problems. In Chapter I, I introduce the problem of integrating multiple meter/age-based maintenance activities in flow shop scheduling, and present a deterministic version of the problem. In Chapter II, a fuzzy bi-objective version of the problem is modeled and solved. In Chapter III, uncertainty in processing and maintenance times will be considered. All three chapters show the application of the problem via a case study in operations and maintenance scheduling of construction machinery.

## Background

## Maintenance Planning

Previously considered more as a cost center, maintenance in recent years is being gradually understood as a profit generating function by industrial managers (Alsyouf, 2007). Since 1940, with the growing advances in science and technology, different maintenance techniques have been emerged as the true value of better maintenance has been appreciated by the industry (Garg \& Deshmukh, 2006). As illustrated in Figure 1, maintenance philosophies can be generally classified as reactive (or unplanned) maintenance and proactive (or planned) maintenance (Kothamasu, Huang, \& VerDuin, 2006).

When a failure occurs, unplanned maintenance types are conducted to either restore the failed item to its original condition, namely corrective maintenance, or to immediately perform a required action to avoid hazardous situations, i.e. emergency maintenance (Veldman, Wortmann, \& Klingenberg, 2011). The preventive types of planned maintenance are performed at a fixed and predetermined interval to decrease the likelihood of failure or performance degradation (Kothamasu et al., 2006). However, preventive maintenance (PM) does not give insight about real time condition of the system and its components. Reliability-centered maintenance (RCM) and condition-based maintenance (CBM) are predictive types of planned maintenance. RCM benefits from reliability estimates of a system to formulate its cost-effective maintenance schedule, but CBM is a decision making strategy for maintenance execution based on the condition of the system which is quantified by some parameters that are constantly monitored (Kothamasu et al., 2006).


Figure 1. Age-based maintenance in taxonomy of maintenance philosophies (Kothamasu et al., 2006).

## Flow Shop Scheduling

Flow shop scheduling has been studied by many researchers after Johnson (1954) introduced the problem for two machines. The main goal in flow shop scheduling is to find a sequence for $n$ jobs that are to be processed by $m$ machines to optimize an objective function. Minimizing the completion time of the very last job (the makespan), the overall completion time, and the tardiness of the jobs are some examples for such an objective.

Figure 2 shows a schematic of a flow shop scheduling problem. The sequence of jobs in the figure is $(1,2,3)$, and the objective is to minimize the makespan. Changing this sequence will yield another value for the objective function, and the goal is to find the optimal sequence for processing the jobs. A different sequence will change the waiting times of the jobs for machines, and the idle times of the machines. In this figure, Job 2 has to wait for Machine 2, and Machine 2 has a waiting time for Job 3. Machine 3 has waiting times for both Job 2 and Job 3. If the objective function changes to minimizing the tardiness of the jobs (in which case a due date is given for every job), two solutions with the same makespan may yield different tardiness values.


Figure 2. An example for a permutation flow shop scheduling problem, $n=m=3$.

When all the jobs are assumed to go through the same sequence of machines, the problem is called a permutation flow shop, and otherwise, nonpermutation (flexible) flow shop. After a job is processed on a machine, and before it proceeds with the next machine, if the next machine is busy with another job, the job can wait in the buffer between the consecutive machines. If the buffer has zero capacity the problem is called blocking flow shop in which case when the next machine is busy the job has to be blocked on the current machine (Abdollahpour \& Rezaeian, 2015).

Scheduling falls into the optimization class of problems where the objective function is to be minimized or maximized; for example, minimizing the total completion time of all the jobs (makespan). From a computational
complexity point of view, it is proved that, even with two machines, flow shop scheduling problem is NP-hard (Papadimitriou \& Kanellakis, 1980). That is, the growth of the time for solving the corresponding decision problem is not a polynomial function of the size of the problem. As a result, when the number of jobs is relatively high, the time for finding the exact optimal solution is not justifiable. Most of the literature related to flow shop scheduling deals with proposing new heuristic or meta-heuristic algorithms that can yield near-optimal solutions in a relatively short amount of time. See for examples in (Abdollahpour \& Rezaeian, 2015), (Ronconi, 2004), (Ying, 2008), (Bryan A Norman, 1999), (Smutnicki, 1998), (Nowicki, 1999), (Brucker, Heitmann, \& Hurink, 2003), and (Hsieh, You, \& Liou, 2009).

## Integrating Maintenance Decisions into Flow Shop Scheduling

In the conventional production scheduling problems, it is assumed that the machines can continuously process the jobs (M. Pinedo, 2012) and the information is complete and certain. However, in practice the machines must stop for preventive or corrective maintenance, and the information available to the planners can be both incomplete and uncertain in scheduling environments (Berry, 1993). In addition, Maintenance costs cover a big percentage of the total operating costs (Ángel-Bello, Álvarez, Pacheco, \& Martínez, 2011; Yip, Fan, \& Chiang, 2014). Therefore, it is reasonable to include minimizing the maintenance cost in the objective function.

The integration of maintenance and scheduling has appeared in the literature in the last two decades (Xu, Wan, Liu, \& Yang, 2015; Yu \& Seif, 2016). The goal of this integration is to mimic the manufacturing or service environments as closely as possible. The more the technical nuances of the maintenance management are considered, the higher the practicality of these models and solutions is going to be; however, incorporating maintenance decisions into the production scheduling problems, requires more sophisticated modeling approaches. This could also make the computational effort larger, especially for the large-scale problems. The issue becomes even more complex when uncertainty is taken into account. In this dissertation, I fully address the integration of age-based maintenance decisions in flow shop scheduling problems.

I provide three types of models: a mixed-integer program (Chapter I), a biobjective fuzzy mixed-integer program (Chapter II), and a stochastic mixedinteger program in which uncertainty of the input data is considered (Chapter III). As for solution methodologies, solution space of the problem is studied in Chapters I and II, and Genetic Algorithms are used as the solution method. Chapter III employs simulation optimization. Three variations of a case study in
construction projects is solved in each chapter. Figure 3 shows how the three chapters are connected.


Figure 3. Dissertation outline.

## CHAPTER I

INTEGRATING MULTIPLE AGE-BASED MAINTENANCE ACTIVITIES INTO FLOW SHOP SCHEDULING

A version of this chapter was originally published by Javad Seif and Andrew J. Yu:

Yu, A. J., \& Seif, J. (2016). Minimizing tardiness and maintenance costs in flow shop scheduling by a lower-bound-based GA. Computers \& Industrial Engineering, 97, 26-40. DOI: https://doi.org/10.1016/j.cie.2016.03.024

Based on my original idea of incorporating maintenance activities into production scheduling, I originated and completed this research project and Dr. Yu supervised my work.


#### Abstract

A permutation flow shop scheduling problem is reformulated as a mixed-integer linear program after incorporating flexible and diverse maintenance activities for minimizing total tardiness and maintenance costs. The terms "flexible" and "diverse" mean that the maintenance activities are not required to perform following fixed and predetermined time intervals, and there can be different types of maintenance activities for each machine. The problem is proved to be NP-hard and a lower bound for the problem is proposed. A lower-bound-based genetic algorithm (LBGA) is presented, in which the algorithm parameters are first tested through a factorial experiment to identify the statistically significant parameters. The LBGA algorithm self-tunes these parameters for its performance improvement based on the solution gap from the lower bound. While it is experienced that only the population size is statistically significant in improving the quality of solutions, through a computational experiment it is also shown that an optimal population size for one problem size yields the same quality of solutions for larger sizes of problems and increasing the population size beyond the optimal size for larger sizes of problems will only negatively affects the efficiency of the algorithm. Computational results that show efficiency and effectiveness of the algorithm are also provided.


### 1.1 Introduction

In conventional machine scheduling problems, it is assumed that the machines are continuously operating and available over the planning horizon (M. L. Pinedo, 2012) which cannot be the case in real world problems where equipment could be unavailable due to breakdown and/or maintenance activities. Although maintenance planning and production scheduling are often studied separately such as in semiconductor manufacturing (Xiaodong, Fernandez-Gaucherand, Fu, \& Marcus, 2004), integration of machine maintenance and scheduling has also appeared in many researches in the last two decades (Xu et al., 2015).

This integration has been proposed for different configurations of manufacturing environments such as single machine, flow shop, parallel machine, job shop, or flexible flow shop, and based on different objective functions such as minimizing makespan, total (expected) completion time, total workload of machines, total workload of critical machines, tardiness, or a combination of them (S. Wang \& Liu, 2014). In this paper, integration of maintenance and operations scheduling in flow shop is presented where the objective function is to minimize the total maintenance and tardiness costs. In some industries such as heavy construction projects, the maintenance costs form a significant portion of the overall costs (Yip et al., 2014). Therefore, it is important to consider the maintenance cost in the objective function along with conventional scheduling criteria such as tardiness.

Flow shop scheduling refers to the problem of determining the optimum permutation of a series of independent jobs which are to be processed by a set of machines. When all the jobs are assumed to go through the same sequence of machines, the problem is called permutation flow shop, and otherwise, nonpermutation flow shop. After a job is processed on a machine, and before it proceeds with the next machine, if the next machine is busy with another job, the job can wait in the buffer between the consecutive machines. If the buffer has zero capacity the problem is called blocking flow shop in which case when the next machine is busy the job has to be blocked on the current machine (Abdollahpour \& Rezaeian, 2015).

Scheduling falls into the optimization class of problems where the objective function is to be minimized or maximized; for example, minimizing the total completion time of all the jobs (makespan). From a computational complexity point of view, it is proved that, even with two machines, flow shop scheduling problem is NP-hard (Papadimitriou \& Kanellakis, 1980). That is, the growth of the time for solving the corresponding decision problem is not a polynomial function of the size of the problem. As a result, when the number of jobs is relatively high, the time for finding the exact optimal solution is not justifiable. Most of the literature related to flow shop scheduling deals with proposing new heuristic or meta-heuristic algorithms that can yield near-optimal solutions in a relatively short amount of time. See for examples in (Abdollahpour \& Rezaeian, 2015), (Ronconi, 2004), (Ying, 2008), (Bryan A Norman, 1999), (Smutnicki, 1998), (Nowicki, 1999), (Brucker et al., 2003), and (Hsieh et al., 2009).

The literature related to the integration of maintenance planning and scheduling was classified differently by (Xu et al., 2015) and (Aramon Bajestani \& Beck, 2015). (Xu et al., 2015) considered the literature to fall into two categories based on the maintenance duration. In the first category, the duration is prefixed. These research works consider the maintenance times as availability constraints
(times at which the machine is not available). In the surveys by (Sanlaville \& Schmidt, 1998), (Schmidt, 2000), (Ma, Chu, \& Zuo, 2010), and (Gordon, Strusevich, \& Dolgui, 2012), this kind of works are identified and further categorized. In the second category, maintenance duration may change based on some factors that are dependent on the scheduling. For example, if the production schedule forces a maintenance activity to be performed at a later time, it takes more time to perform. In short, the duration is a function of the start time of the activity. (Xu et al., 2015) also discussed the subtle differences between these functions as appeared in the works of (S. J. Yang \& Yang, 2010), (T. C. E. Cheng, Yang, \& Yang, 2012), (Mor \& Mosheiov, 2012), (Luo \& Ji, 2015), (Xu, Yin, \& Li, 2010), (S. J. Yang, 2012), and (S.-J. Yang, 2013). In this paper, we will consider prefixed duration for maintenance activities.

Aramon Bajestani and Beck (2015) also divided the literature in two categories. The first category was the same as the first category determined by (Xu et al., 2015). The second category, however, is different and addresses those research works which assume that the processing times of the jobs varies based on the maintenance. In the models presented in these literatures, a rate, which is dependent on maintenance activities, is applied to the processing times of the jobs (C. Y. Lee \& Leon, 2001). Since we do not have such assumption for processing times, we will not further discuss the related works in the second category.

In this paper we will model and optimize a flow shop scheduling problem integrated with diverse and flexible maintenance activities. Most of the related works consider a single machine. However, there are some works such as (Allaoui \& Artiba, 2004) in which the integration of maintenance planning and production scheduling has been extended to flow shop setting. They considered a hybrid (non-permutation) flow shop with different objective functions while also considering setup, cleaning and transportation times. They proposed a combination of simulation and one of the meta-heuristic algorithms (simulated annealing) as the solution approach. Other meta-heuristic solution approaches such as genetic algorithm and tabu search have been utilized by (Aggoune, 2004), (Ruiz, Carlos García-Díaz, \& Maroto, 2007), and a detailed review of all the approaches along with a variable neighborhood search was presented by (Naderi, Zandieh, \& Fatemi Ghomi, 2009).

What distinguishes this paper from the related works is flexibility and diversity of maintenance activities. Flexibility means that we are not limited to perform maintenance activities at fixed intervals. Diversity means that we have different set of maintenance activities for a machine. One downside of fixedinterval preventive maintenance (PM) activities is that we do not know if the oil or bearing which are to be replaced, for example, have been fully utilized. Condition based maintenance (CBM) involves monitoring equipment's health and
replacements or other maintenance actions that are performed only when they are necessary. The cost of conducting condition monitoring, however, is not always justifiable and there are researches dedicated solely to cost-wise justification of running a CBM program (Azadeh, Asadzadeh, \& Seif, 2014).

Flexible maintenance activities try to imitate CBM without monitoring, that is, by estimating the remaining useful life of a system based on the known deterioration rate that each job incurs in the system. Job-dependent deterioration of machine means that in environments analogous to manufacturing, when different jobs are processed by a machine, we can expect the health of a machine to be deteriorated with different rates when different jobs are processed. Having these deterioration rates available, a more economic maintenance plan can be achieved in which maintenance activities are not necessarily performed with fixed intervals (in the literature, general, flexible, or noncyclical PMs are also used with the same meaning).
S. Bock, D. Briskorn, and A. Horbach (2012) tried to extend classic machine scheduling problems by taking machine deterioration and maintenance activities (MAs) into account. They described health of a single machine by a bounded maintenance level (ML) which is deteriorated as jobs are processed. They assumed that the deterioration is a linear function of the processing time of the jobs and each job has its own coefficient (failure rate). They considered pure scheduling objective functions such as minimization of completion times, makespan, and tardiness. Majority of their work is dedicated to the determination of computational complexity of the problems introduced in their paper.

Diversity of maintenance activities has not been observed in flow shop literature. As for the objective function, the main focus of our model is on minimizing the maintenance cost (unlike most of the discussed research works) because in some flow shop settings such as in a petrochemical plant or a construction project, the maintenance cost forms the main portion of the expenses.

In many of the existing research works, the maintenance cost is usually considered as a whole along with other production costs (Allaoui, Lamouri, Artiba, \& Aghezzaf, 2008). In addition, some practical considerations have never been taken into account. One of such considerations is that a machine usually has more than one type of MA. Because terms like "multi-maintenance activities" and "multiple maintenance activities" appeared in the literature (Zarook, Rezaeian, Tavakkoli-Moghaddam, Mahdavi, \& Javadian, 2014), (Sun \& Li, 2010), and (Shi \& Xu, 2014) do not refer to different types of maintenance activities, we have adopted the term "diverse maintenance activities" in order to more distinctively represent the problem.

Note that some works that integrated preventive maintenance planning and production might not be comparable with this research as they are basically focusing on production planning, not jobs scheduling. For example, (Aghezzaf, Jamali, \& Ait-Kadi, 2007) integrated maintenance, repair, and inventory in their models. Their model was to find the best production quantity for different products along with the optimum PM interval that minimizes total cost. Aghezzaf et al. (2007) and a few other researchers have considered maintenance cost in their works but unlike the presented research, they did not incorporate the maintenance resource cost into the maintenance planning. Instead, they considered the maintenance and repair cost as a fixed value multiplied by the frequency of maintenance activities. In our proposed model, we break the maintenance cost into various costs of resources and optimize the jobs schedule in a way that minimum resource is used.

There are some researches that consider both corrective (and unplanned) maintenance (CM) and PM. (Allaoui et al., 2008), also, tried to find the optimum length for PM cycles with minimal repair at failure for different machines working in a parallel setting with almost the same objective function as their previous work. They also integrated maintenance with production planning and suggested an approximation Lagrangian decomposition to solve their problem for both cyclic and noncyclic (flexible) cases.
(Chen, 2008; Sun \& Li, 2010; Xu et al., 2010) reduced the rigidity of fixed interval PMs by assuming lower and upper bounds for the time between successive maintenance activities. There can be other not-so-common restrictions, too, such as limiting the number of times a specific maintenance activity can be performed in (Mosheiov \& Sarig, 2009). The proposed model in this paper with flow shop setting and flexible and multiple (diverse) maintenance activities can cover both single machine with multiple maintenance activities and, with simple adjustments in the input parameters, parallel machines with single maintenance activity for each machine.

The contribution of this paper is threefold. First, a practical problem is introduced that extends mathematical formulation of the conventional flow shop scheduling problem as a mixed integer linear program (MILP) by incorporating flexible and diverse maintenance activities into it. Second, the lower bounds of the problem are found using the proposed algorithms that convert the problem into several small and easy-to-solve Knapsack problems. Finally, a new genetic algorithm (GA) that can solve any realistic sizes of the problem effectively and efficiently is introduced. The algorithm uses the lower-bounds and factorial experiments to fine-tune its parameters, and is called lower-bound-based GA (LBGA).

The rest of this chapter is organized as follows. In Section 1.2, the problem is described along with a summary of assumptions, and in Section 1.3 the problem is mathematically formulated. Computational complexity discussion, lower bounds of the problem, and a genetic algorithm that has been developed based on the lower bounds are presented in Section 1.4. Computational results that validate effectiveness and efficiency of the algorithm along with other computational experiments are presented in Section 1.5. A case study in heavy construction projects is presented in Section 1.6 to show the application of the problem. Conclusions and possible future works as extensions of this paper are discussed in Section 1.7.

### 1.2 Problem Definition

We try to minimize the total maintenance and customer dissatisfaction costs in a flow shop setting. The jobs can have different processing times with respect to a certain machine and each job can have different processing times on different machines. A machine's health condition could be expressed by the machine's diverse maintenance levels (MLs). ML was suggested first by (S. Bock et al., 2012). Diversity means, for example, one ML may indicate the cleanliness of an air filter and another one for quality of the engine's oil. Each maintenance level will be depleted from its maximum value as the jobs are processed. If an ML value falls below zero, in theory it is equivalent to a failure, and in practice, it indicates a high failure probability.

After a certain job is processed on a certain machine, each ML of the machine is decreased by a certain amount because the job has a certain deterioration rate with respect to each ML for each machine. When the remaining useful life in terms of an ML is not enough for processing the next job, its respective maintenance activity (MA) will be performed in order to restore the ML to its maximum. We are looking for a sequence of jobs that requires minimum number of MAs. Figure 4 shows an illustrative example of two machines, three jobs, and two MLs for each machine. The example shows one feasible sequence of jobs in which one and three MAs are performed on the first and the second machine, respectively. Note that each MA has a different duration on each machine and it only affects the respective ML. If the initial value of the ML (the maximum) is set to infinity, it implies that this ML does not exist for the machine. So, we consider the same set of MLs for all the machines. Also, it is possible that a job does not affect a certain ML of a certain machine in which case the deterioration rate is equal to zero.

Customer dissatisfaction occurs when the completion time of a job is greater than its due date. However, the cost might be relatively lower than the


M: Machine - J: Job - L: Maintenance Level -MA: Maintenance Activity Figure 4. An illustrative example of the problem.
maintenance costs. For example, delay in production in a make-to-stock production setting is often insignificant comparing with the maintenance cost as long as the production efficiency is not affected and the delay is not prolonged. Our approach to the solution of the problem focuses more on the maintenance cost than the customer dissatisfaction cost. For modeling the customer satisfaction, we use the conventional tardiness objective function. The following is a summary of assumptions considered in the formulation of the problem.

1. By flow shop we mean permutation flow shop.
2. All the machines have the same set/types of MLs, and hence, the same set of Mas.
3. There is no buffer between machines.
4. Duration of a specific MA for a specific machine is known and invariable. The same MA can have a different duration on a different machine.
5. When a job is being processed, all the MLs are subject to deterioration according to a linear function by $\delta \cdot p$ where $\delta$ is the deterioration rate of ML caused by a job after it is processed and $p$ is processing time of the job.
6. Before processing the first job, all the MLs of all the machines are at their maximum.
7. Sufficient/unlimited resources (maintenance spare parts, materials, and workforces, operators, etc.) are available for processing the jobs and performing the Mas.
8. Pre-emption is not allowed.
9. All the MAs are performed to completion.
10. The quantity $\delta \cdot p$ is always less than the maximum of the corresponding maintenance level. Otherwise, the problem will be infeasible.
11. Random failures are not considered.

### 1.3 Mathematical Formulation

Following is a list of sets, parameters, and variables used throughout the mathematical formulation of the problem. Let $m, n$, and $/$ be the number of machines, jobs, and maintenance levels or their respective PMs (PM types), respectively. Then we have the following indices, parameters and variables.
$i \quad$ Represents machines where $i=1,2, \ldots, m$
$j \quad$ Represents production jobs where $j=1,2, \ldots, n$
$q \quad$ Represents sequence of jobs (jobs positions) where $q=1,2, \ldots, n$
$k \quad$ represents MLs or their respective MAs where $k=1,2, \ldots, l$
$p_{i j} \quad$ Processing time of job $j$ on machine $i$
$\delta_{i j k} \quad$ Deterioration rate of maintenance level $k$ (MA type $k$ ) of machine $i$ when job $j$ is processed
$e_{i k} \quad$ Duration of respective MA type $k$ on machine $i$
$M L_{\text {max }}^{k}$ Maximum of ML type $k$
$S P_{i k} \quad$ Cost of required spare parts and materials for MA type $k$ on machine $i$
$W F_{k} \quad$ Cost of skilled workforce per time unit for performing MA type $k$
$d_{j} \quad$ The time at which job $j$ is due
$\pi_{j} \quad$ Penalty cost associated with each time unit delay in completion of job $j$
$M, M^{\prime} \quad$ Sufficiently large numbers
$z \quad$ Total cost
$x_{j q} \quad$ Binary variable that takes the value 1 if job $j$ is assigned to position $q$ and 0 otherwise
$y_{i q k} \quad$ Binary variable that takes the value 1 when PM type $k$ is performed on machine $i$ before processing the $q$-th job and 0 otherwise
$M L_{i q}^{k} \quad$ Numerical representation of ML type k of machine $i$ before processing the $q$-th job
$c_{q} \quad$ Completion time of the job assigned to position $q$
$t_{q} \quad$ Tardiness of the job assigned to position $q$ (amount of lateness in completion of the job)
$\Pi_{j q} \quad$ Penalty associated with job $j$ assuming that it is in position $q$
$v_{i q} \quad$ Waiting time of the machine $i$ for the $q$-th job (idle time)
$w_{i q} \quad$ Waiting time of the $q$-th job for machine $i$
As it was discussed earlier, the objective function (OF) of the model is to minimize the total cost which comprises the penalty cost incurred because of lateness in completion of each job (tardiness) and cost of maintenance resources, namely cost of spare parts and required workforces. Total penalty costs can be expressed as in Equation (1).

Total penalty cost $=\sum_{j=1}^{n} \sum_{q=1}^{n} \pi_{j} x_{j q}\left(\max \left\{0, c_{q}-d_{j}\right\}\right)$
The term $\max \left\{0, c_{q}-d_{j}\right\}$ in Equation (1) will only be meaningful when $x_{j q}=1$, so the term is equivalent to $\max \left\{0, c_{q}-\sum_{j=1}^{n} x_{j q} d_{j}\right\}$. In order to linearize Equation (1), we take the following steps. Firstly, we replace the term $\max \left\{0, c_{q}-\sum_{j=1}^{n} x_{j q} d_{j}\right\}$ with a new variable $t_{q}$ which is subject to the following constraints

$$
\begin{equation*}
t_{q} \geq c_{q}-\sum_{j=1}^{n} x_{j q} d_{j} \quad q=1,2, \ldots, n \tag{1.2}
\end{equation*}
$$

$$
\begin{equation*}
t_{q} \geq 0 \quad q=1,2, \ldots, n \tag{1.3}
\end{equation*}
$$

If the completion time becomes greater than the due time, the minimum value for $t_{q}$ will be the difference between them, and zero otherwise. Since the OF seeks the minimum value, the algorithm always chooses the minimum value for $t_{q}$. Now we have $\sum_{j=1}^{n} \sum_{q=1}^{n} x_{j q} \pi_{j} t_{q}$ instead of Equation (1) which is still nonlinear (quadratic). In order to linearize it, we introduce a new variable $\Pi_{j q}$ which is the penalty associated with job j, if it is assigned to position $q$ in the sequence of the jobs. Namely,

$$
\Pi_{j q}=\left\{\begin{array}{rll}
\pi_{j} t_{q}, & x_{j q}=1 & j=1,2, \ldots, n  \tag{1.4}\\
0, & x_{j q}=0 & q=1,2, \ldots n
\end{array}\right.
$$

The following four inequalities are proposed in order to express Equation (4) in a linear manner.

$$
\begin{array}{cc}
\Pi_{j q}-\pi_{j} t_{q} \geq-M\left(1-x_{j q}\right) & j=1,2, \ldots, n \\
\Pi_{j q}-\pi_{j} t_{q} \leq M\left(1-x_{j q}\right) & q=1,2, \ldots n \\
\Pi_{j q} \geq-M\left(x_{j q}\right) & j=1,2, \ldots, n \\
\Pi_{j q} \leq M\left(x_{j q}\right) & q=1,2, \ldots n  \tag{1.8}\\
& j=1,2, \ldots, n \\
& q=1,2, \ldots n \\
& j=1,2, \ldots, n \\
& q=1,2, \ldots n
\end{array}
$$

Note that when $x_{j q}=0$, the first pair inequalities, Equations (1.5) and (1.6), are turned off (because of $M$ which is a sufficiently big number and makes the constraint feasible for any values of variables) and the second pair, Equations (1.7) and (1.8), are turned on. Each pair forms an equation when it is turned on and the right hand side of both inequalities becomes zero. The same reasoning can be applied for the case when $x_{j q}=1$ when Equations (1.5) and (1.6) are turned on and Equations (1.7) and (1.8) are turned off.

We can now write the full OF as

$$
\begin{equation*}
\min z=\sum_{j=1}^{n} \sum_{q=1}^{n} \Pi_{j q}+\sum_{i=1}^{m} \sum_{q=1}^{n} \sum_{k=1}^{l} y_{i q k}\left(S P_{i k}+e_{i k} W F_{k}\right) \tag{1.9}
\end{equation*}
$$

where $y_{i q k}$ is used in order to take into account only the cost of those MAs that are decided to be performed. This OF is subject to Constraint sets (1.2-1.3 and 1.5-1.8), and the following constraints.

To make sure that each job has one and only one position in the sequence of jobs, we use the following two sets of constraints

$$
\begin{array}{ll}
\sum_{\substack{q=1 \\
n}} x_{j q}=1 & j=1,2, \ldots, n \\
\sum_{j=1}^{n} x_{j q}=1 & q=1,2, \ldots, n
\end{array}
$$

According to the flow shop literature (Selen, 1986), and before incorporating flexible and diverse maintenance activities to the flow shop problem, the waiting times of machines and jobs can be calculated through the following set of equations:

$$
\left.\left.\begin{array}{rl}
{\left[\begin{array}{c}
\text { idle time of } \\
\text { machine } i \\
\text { for job } q+1
\end{array}\right]} & +\left[\begin{array}{c}
\text { processing time of } \\
\text { job } q+1 \\
\text { on machine } i
\end{array}\right]
\end{array}\right] \quad \begin{array}{c}
\text { waiting time of } \\
\text { job } q+1 \\
\text { for machine } i+1
\end{array}\right] \quad i=1,2, \ldots, m-1, ~ q=1,2, \ldots, n-1 .
$$

The MAs could be performed before or after any job for each of the machines in the shop. For the simplicity, we attach the maintenance time to the job processing time at its beginning. The actual MAs for each machine will only be scheduled when they are necessary which are determined by the model.
Thus, Equation (1.12a) can be extended to Equation (1.12b), after incorporating the potentially necessary MAs in the job scheduling.

$$
\begin{align*}
& {\left[\begin{array}{c}
\text { idle time of } \\
\text { machine } i \\
\text { for job } q+1
\end{array}\right]+\left(\left[\begin{array}{c}
\text { processing time of } \\
\text { job } q+1 \\
\text { on machine } i
\end{array}\right]+\right.} \\
& \left.\left[\begin{array}{c}
\text { duration of all necessary MAs } \\
\text { before processing job } q+1 \\
\text { on machine } i
\end{array}\right]\right)+ \\
& {\left[\begin{array}{c}
\text { waiting time of } \\
\text { job } q+1 \\
\text { for machine } i+1
\end{array}\right]=\left[\begin{array}{c}
\text { idle time of } \\
\text { machine } i+1 \\
\text { for job } q+1
\end{array}\right]+} \\
& \left(\left[\begin{array}{c}
\text { processing time of } \\
\text { job } q \\
\text { on machine } i+1
\end{array}\right]+\right. \\
& {\left[\begin{array}{c}
i=1,2, \ldots, m-1, \\
q=1,2, \ldots, n-1, \\
k=1,2, \ldots, l .
\end{array}\right.}  \tag{1.12b}\\
& {\left[\begin{array}{c}
\text { duration of } \left.\left.\begin{array}{l}
\text { before processing job } q \\
\text { on machine } i+1
\end{array}\right]\right)+ \\
{\left[\begin{array}{c}
\text { waiting time of } \\
\text { job } q \\
\text { for machine } i+1
\end{array}\right]}
\end{array}\right.}
\end{align*}
$$

Throughout the model we will schedule the necessary MAs prior to a job at an arbitrary position $q$. The final form of Equation (1.12b) can be expressed as follows.

$$
\begin{array}{rlr}
v_{i(q+1)}+\left(\sum_{k=1}^{l} y_{i(q+1) k} e_{i k}+\sum_{j=1}^{n} x_{j(q+1)} p_{i j}\right) & \\
& +w_{(i+1)(q+1)} & \\
& =v_{(i+1)(q+1)} & \\
& +\left(\sum_{k=1}^{l} y_{(i+1) q k} e_{(i+1) k}\right. & \begin{aligned}
& \\
& \\
& \\
& +\sum_{j=1}^{n} x_{j q} p_{(i+1) j}=1, \ldots, \ldots, n-1, \ldots, l .
\end{aligned} \\
& k=1, w_{(i+1) q} & \tag{1.12c}
\end{array}
$$

According to Assumption (6), all the maintenance levels prior to the first job are at their maximum and hence no MA is performed before processing the first job. This can be expressed either by Equation (1.13a) or (1.13b). We use Equation (1.13a).

$$
\begin{equation*}
M L_{i 1}^{k}=M L_{\max }^{k} \tag{1.13a}
\end{equation*}
$$

$$
\begin{aligned}
& i=1,2, \ldots, m, k= \\
& 1,2, \ldots, l
\end{aligned}
$$

or,

$$
\begin{array}{ll}
y_{i 1 k}=0 & i=1,2, \ldots, m, k=  \tag{1.13b}\\
1,2, \ldots, l .
\end{array}
$$

The job in the first position $(q=1)$ does not wait in buffer for any of the machines as it is processed first by all the machines. This can be expressed using the following constraints

$$
\begin{equation*}
w_{i 1}=0 \quad i=1,2, \ldots, m . \tag{1.14}
\end{equation*}
$$

The first machine in the flow shop also does not wait for any of the jobs,

$$
\begin{equation*}
v_{1 q}=0 \quad q=1,2, \ldots, n \tag{1.15}
\end{equation*}
$$

Idle times for machines 2 to $m$ with respect to the first job $(q=1)$ will be

$$
\begin{equation*}
v_{i 1}=\sum_{j=1}^{n} \sum_{f=1}^{i-1} x_{j 1} p_{f j} \quad i=2,3, \ldots, m \tag{1.16}
\end{equation*}
$$

The first two summations in Equation (1.17) can be interpreted as a search through all the jobs to see which one is assigned to the first position on the machines prior to the machine $i$ and then adding its processing times on the previous machines to the idle time of machine $i$. The interpretation of summations like these as a means for search can be used for the rest of the constraints with analogous summations. Buffer time of the jobs scheduled after the first job, before proceeding with the first machine, can be modeled as follows

$$
\begin{equation*}
w_{1 q}=\sum_{r=1}^{q-1} \sum_{j=1}^{n} x_{j r} p_{1 j}+\sum_{r=1}^{q} \sum_{k=1}^{l} y_{1 r k} e_{1 k} \quad q=2,3, \ldots, n \tag{1.17}
\end{equation*}
$$

In order to make sure that maintenance levels do not fall below zero during or after processing a job we use the following set of constraints

$$
M L_{i q}^{k} \geq \sum_{j=1}^{n} x_{j q} p_{i j} \delta_{i j k} \quad \begin{array}{ll}
i=1,2, \ldots, m  \tag{1.18}\\
& q=1,2, \ldots, n \\
k=1,2, \ldots, l
\end{array}
$$

Constraint (1.18) requires all maintenance levels of a machine to be equal or greater than the amount of linear deterioration by which the ML drops so that none of the levels fall below zero because when a maintenance level falls below zero it implies machine breakdown. After processing a job, a maintenance level
is equal to its level before processing the previous job minus the corresponding deterioration. If Constraint (1.18) does not hold, the respective MA will be performed in order to restore the level to its maximum. This is expressed as

$$
M L_{i q}^{k}=\left\{\begin{array}{rll}
M L_{i(q-1)}^{k}-\sum_{j=1}^{n} x_{j(q-1)} p_{i j} \delta_{i j k}, & y_{i q k}=0 & i=1,2, \ldots, m  \tag{1.19}\\
M L_{\max }^{k}, & y_{i q k}=1 & q=2,3, \ldots, n \\
k=1,2, \ldots, l
\end{array}\right.
$$

We reapply analogously the method that we used to convert Equation (1.4) to Equations (5-8) in order to linearize Equation (1.19).

$$
\begin{array}{cl}
M L_{i q}^{k}-\left(M L_{i(q-1)}^{k}-\sum_{j=1}^{n} x_{j(q-1)} p_{i j} \delta_{i j k}\right) \geq-M^{\prime}\left(y_{i q k}\right) & i=1,2, \ldots, m \\
& q=2,3, \ldots, n \\
M L_{i q}^{k}-\left(M L_{i(q-1)}^{k}-\sum_{j=1}^{n} x_{j(q-1)} p_{i j} \delta_{i j k}\right) \leq M^{\prime}\left(y_{i q k}\right) & i=1,2, \ldots, l \\
& q=2,3, \ldots, n \\
& k=1,2, \ldots, l \\
M L_{i q}^{k}-M L_{\max }^{k} \geq-M^{\prime}\left(1-y_{i q k}\right) & i=1,2, \ldots, m \\
& q=2,3, \ldots, n \\
& k=1,2, \ldots, l  \tag{1.23}\\
M L_{i j}^{k}-M L_{\max }^{k} \leq M^{\prime}\left(1-y_{i q k}\right) & i=1,2, \ldots, m \\
& q=2,3, \ldots, n
\end{array}
$$

Completion time of each job is equal to sum of its processing times and its waiting times in the buffer. This yields the following set of constraints. Note that, as stated earlier, we consider duration of required MAs, too, whenever we take into account processing times.

$$
\begin{equation*}
c_{q}=\sum_{i=1}^{m}\left(w_{i q}+\sum_{j=1}^{n} x_{j q} p_{i j}+\sum_{k=1}^{l} y_{i q k} e_{i k}\right) \quad q=1,2, \ldots, n . \tag{1.24}
\end{equation*}
$$

Finally,

$$
\begin{array}{rl}
x_{j q}, y_{i q k} \in\{0,1\}, & i=1,2, \ldots, m, \\
M L_{i q}^{k}, c_{q}, t_{q}, \Pi_{j q}, v_{i q}, w_{i q} \geq 0 & j=1,2, \ldots, n, \\
& q=1,2, \ldots, n,  \tag{1.25}\\
k=1,2, \ldots, l .
\end{array}
$$

As explained above, we added new variables in order to be able to linearize the model. Now we have a mixed integer linear model. The final model comprises of the objective function in Equation (1.9) and constraints in Equations
(1.2-1.3, 1.5,1.8, 1.12c, 1.13a, 1.14-1.25). In the next section we will present the algorithms developed and the methods for solving the model.

### 1.4 Solution Approach

In this section, first, we prove that the presented problem is NP-hard. Then, we find a lower bound for the problem. Based on this lower bound, we design a genetic algorithm (GA) whose parameters are set in such a way that the gap between its best solutions and the lower bound does not increase with the size of the problem. Computational results that verify efficiency and effectiveness of the GA will be presented in the next section.

## Complexity of the problem

Two variations of the modeled problem can be defined based on the number of machines and MLs: a problem that involves only a single machine with only a single maintenance level ( $m=n=1$ ), MAINTFLOW-SINGLE, and a problem in which the number of machines or maintenance levels is not limited, MAINTFLOW-FULL. In order to show that an optimization problem is NP-Hard, we will show that its corresponding decision problem is NP-Complete. In doing so, we will show that 1) a solution for an instance of the problem can be verified for feasibility in polynomial time, and 2) a problem which has already been proven to be NP-Complete can be reduced to it. Although the problem of minimizing total tardiness of a set of jobs which are to be scheduled on a single machine has already been proven to be NPhard (Du \& Leung, 1990) and it can be easily reduced to the presented problem, we present another proof as a contribution.

Theorem 1. MAINTFLOW-SINGLE is NP-hard.
Proof. Any sequence (permutation) of the jobs can be considered as a feasible solution. If a sequence is given as a solution for an instance the problem, it is only needed to check whether:

1. All the jobs exist in the sequence, and
2. No job has been repeated in the sequence.

Obviously, this can be done in polynomial time. Next, we reduce the KNAPSACK problem to MAINTFLOW-SINGLE. Let the following, $K S$, be an instance of KNAPSACK optimization problem.

$$
\begin{equation*}
\max z=\sum_{o=1}^{o} v_{o} U_{o} \tag{1.26}
\end{equation*}
$$

Subject to:

$$
\begin{equation*}
\sum_{o=1}^{o} w_{o} U_{o} \leq W \tag{1.27}
\end{equation*}
$$

where $U_{o} \in\{0,1\}, o=1, \ldots, O$, are the decision variables, and $W, w_{o}$ and $v_{o}$ are capacity of the knapsack, size and value of object $o$, respectively. The optimum solution for this problem (the most valuable subset of objects whose total size does not exceeds the capacity of the knapsack) can be obtained if we solve the following instance of the problem modeled in Section 1.3.

- $m=1, n=O, l=1$,
- $i=1, j=o, k=1$,
- $M L_{\max }^{k}=W, p_{i j} \delta_{i j k}=w_{o}, \pi_{j}=-v_{o}$,
- $e_{i k}=c_{1}, d_{j}=c_{2}, S P_{i k}=0, W F_{k}=0$.

In this reduction, $c_{1}$ and $c_{2}$ are arbitrarily-selected constants.
Theorem 2. MAINTFLOW-FULL is NP-hard.
Proof. MAINTFLOW-SINGLE can be reduced to MAINTFLOW-FULL by setting $m=l=1$ in MAINTFLOW-FULL.

## Lower bounds

The objective function of the presented problem consists of two major costs which were to be minimized: maintenance cost and tardiness cost. In order to obtain a lower bound for the problem, we can find the lower bound for each of the two costs and then sum them up. Because the use of this lower bound is to control its gap from the GA solution, it is not a major concern to find the tightest lower bound.

## Lower bound for MAINTFLOW-SINGLE

Maintenance Cost. The least number of maintenance activities that are required in order for the maintenance level not to fall below zero can be obtained by grouping the jobs based on their sum of deterioration rates, that is, we try to find groups of jobs that deteriorate the maintenance level (deplete the remaining useful life) as completely as possible. Following the same notations that we used for modeling the problem, the KNAPSACK problem can be remodeled as follows.

$$
\begin{equation*}
\max z=\sum_{j=1}^{n} U_{j} p_{j} \delta_{j} \tag{1.28}
\end{equation*}
$$

Subject to:

$$
\begin{equation*}
\sum_{j=1}^{n} U_{j} p_{j} \delta_{j} \leq M L_{\max } \tag{1.29}
\end{equation*}
$$

where $U_{j} \in\{0,1\}$ is the decision variable which determines whether we select job $j\left(U_{j}=1\right)$ or not $\left(U_{j}=0\right)$. Capacity of the knapsack is the maintenance level's maximum value, i.e. $M L_{\max }$, weight and also value of each job (object) is how much it can deteriorate the level, namely $p_{j} \delta_{j}$. The optimal solution determines which jobs consume the ML the most without MA between them. These jobs are crossed out from the original list of jobs. The problem is solved again for the remaining jobs and this process continues until no job is left. Number of required MAs will be equal to the number of groups minus one since we require MAs between the groups. Algorithm 1 was used for finding the maintenance cost lower bound.

Tardiness Cost. Although there are some papers that have found the lower bound for total tardiness in a single machine scheduling problem ((Tansel, Kara, \& Sabuncuoglu, 2001), (Della Croce, Grosso, \& Paschos, 2004)), their result cannot be directly used in our work as we are dealing with the cost of tardiness not the tardiness itself. The tightness of the bound is not a main concern. What follows is a proposed lower bound for total tardiness cost in MAINTFLOW-

Algorithm 1. Total Maintenance Cost Lower Bound for MAINTFLOW-SINGLE.

```
Input: \(J\), set of all jobs; \(M=S P_{11}+e_{11} \times W F_{k}\), cost of each maintenance
activity
Output: \(L\), total maintenance cost
\(G \leftarrow \emptyset\)
\(N \leftarrow 0\) //Number of required MAs
while \(J \neq \emptyset\) do
    \(G \leftarrow \operatorname{KNAPSACK}(J)\)
        \(J \leftarrow J \backslash G\)
        \(G \leftarrow \emptyset\)
        \(N \leftarrow N+1\)
\(L \leftarrow(N-1) \times M\)
```

SINGLE problem. Following the same notations that we used in modeling the problem, let $d_{0}=\max d_{j}, \pi_{0}=\min \pi_{j}$, and $p_{10}=\min p_{1 j}$ where $j=1, \ldots, n$. Algorithm 2 yields a possible lower bound for the problem.

Algorithm 2. Total Tardiness Cost Lower Bound for MAINTFLOW-SINGLE.

```
C\leftarrow0 //Completion Time
T\leftarrow0//Total Tardiness
for j=1 to }n\mathrm{ do
C}\leftarrowC+\mp@subsup{p}{10}{
if }\mp@subsup{d}{0}{}<C\mathrm{ then
    T\leftarrowT+C-d
L\leftarrowT\times\mp@subsup{\pi}{0}{}//Lower Bound
```


## Lower bound for MAINTFLOW-FULL

Maintenance Cost. When the jobs are to be processed by more than one machine and each machine has more than one ML, we can get the lower bound for each ML of each machine using Algorithm 1 and then adding them together. Algorithm 3 summarizes this.

## Algorithm 3. Total Maintenance Cost Lower Bound for MAINTFLOW-FULL.

```
Input: J, set of all jobs; \(M_{i k}=S P_{i k}+e_{i k} \times W F_{k}\), cost of each maintenance
activity
Output: \(L\), total maintenance cost
\(L \leftarrow 0\)
for \(i=1\) to \(m\) do
    for \(k=1\) to \(l\) do
        \(G \leftarrow \varnothing\)
        \(N \leftarrow 0 / /\) Number of required MAs
        while \(J \neq \varnothing\) do
            \(G \leftarrow \operatorname{KNAPSACK}(J)\)
                                    \(J \leftarrow J \backslash G\)
            \(G \leftarrow \varnothing\)
                        \(N \leftarrow N+1\)
\(L \leftarrow L+(N-1) \times M_{i k}\)
```

Tardiness Cost. With some changes, Algorithm 2 can be enhanced to calculate the lower bound for total tardiness cost where $p_{i 0}=\min p_{i j} \forall i=1, \ldots, m$ and $j=$ $1, \ldots, n . c_{i j}, w_{i j}$ and $e_{0}$ are completion time of the job in position $i$ with respect to machine $j$, waiting time of job $i$ for machine $j$, and minimum execution time of maintenance activities, respectively. Algorithm 4 shows this.

Algorithm 4. Total Tardiness Cost Lower Bound for MAINTFLOW-FULL.

```
C\leftarrow\mathbf{0}//Completion Time: A matrix of size m\timesn that shows completion time of
each job on each machine
W\leftarrow\mathbf{0}//Waiting Time: A matrix with size m\timesn that shows waiting time of each
job for each machine
T\leftarrow0 //Total Tardiness
for }i=1\mathrm{ to }m\mathrm{ do
    c
for j=2 to n do
    c
for i=2 to m do
    for j=2 to n do
        if }\mp@subsup{c}{i(j-1)}{}<\mp@subsup{c}{(i-1)j}{}\mathrm{ then
                            wij}\leftarrow\mp@subsup{c}{i(j-1)}{}-\mp@subsup{c}{(i-1)j}{
            c}\mp@subsup{c}{ij}{}=\mp@subsup{c}{(i-1)j}{}+\mp@subsup{w}{ij}{}+\mp@subsup{p}{i0}{
for j=1 to n do
    if d}\mp@subsup{|}{0}{< c}\mp@subsup{c}{mj}{}\mathrm{ then
                        T\leftarrowT+\mp@subsup{c}{mj}{}-\mp@subsup{d}{0}{}
T\leftarrowT+(N-1)\times 的
L\leftarrowT\times 
```


## Lower-Bound-Based Genetic Algorithm (LBGA)

We showed that the presented problem is NP-hard. As will be shown in the next section, only very small sizes of the problem can be solved in a reasonable and predictable time for exact optimal solution using commercial solvers such as IBM CPLEX. Complex structure of the problem also makes it difficult to come up with an exact heuristic algorithm. This leads to the call for designing and implementing a metaheuristic algorithm for the problem where it is ensured that the whole feasible solution space can be searched through randomly-generated solutions and each solution can be further improved by a local search. We propose a genetic algorithm for the presented problem by using an experimental design to identify the significant parameters of the algorithm and then tuning those parameters based on the identified lower bounds. Figure 5 illustrated the solution approach.

## Genetic Algorithms

Genetic algorithms (GAs) are among the most widely-used and known search heuristics. GAs have been applied to different research areas (Chambers, 1998) including several applications in machine scheduling (B. A. Norman \& Bean, 1999). In the works of (Sortrakul, Nachtmann, \& Cassady, 2005) and (Sortrakul \& Cassady, 2007), GA has been used to solve the integrated scheduling of production and maintenance for a single machine. A GA generally works by keeping a population of candidate solutions represented as chromosomes whose fitness is determined by their respective objective function value. The chromosome is composed of a sequence of elements (numbers) each of which is indicative of a feature of the solution. A fixed number of most fit chromosomes are selected as the population of the current generation, on a percentage of which the local search operators, crossover and mutation, are applied. Crossover produces new offspring chromosomes from certain chromosomes of the current generation which are selected as parents. Mutation is applied to certain chromosomes of either current population or the offspring chromosomes according to a specific mutation scheme in order to produce new chromosomes. This process is iterated until a stopping condition is satisfied.

Like other types of meta-heuristic algorithms, a GA has a set of parameters whose values affect performance and quality of the solutions of the algorithm. This set of parameters includes population size (number of competing chromosomes) in each iteration, percentage of solutions for crossover and mutation, parameters for the specific method of crossover and mutation that are used, and parameter of stopping condition (for example, the time at which the algorithm stops). The interactions between GA parameters and respective literature was reviewed and studied by (Deb \& Agrawal, 1999). According to that study, the most important parameters are population size $(N)$, crossover probability $\left(p_{c}\right)$, and mutation probability $\left(p_{m}\right)$. The optimal value for the mutation
probability and crossover probability are highly dependent on the chromosome representation (Tate \& Smith, 1993). As a result, for a specific chromosome representation of a certain problem, it must be tested whether these parameters affect solution quality or performance.

## Design of the GA

Chromosome representation. Each individual chromosome (solution) is a sequence of the jobs. Although in the MILP problem formulation we introduced other decision variables, such as the binary variable that determines whether a maintenance activity is placed before a job position ( $y_{i q k}$ ), the only independent variable is the position of each job in the permutation $\left(x_{i j}\right)$. When a sequence of jobs is represented as a chromosome, random permutations provide access to different areas of the solution space relatively fast and since the genes are jobs, neighborhood of a solution can be searched relatively fast by crossover and mutation operators which make the local search computationally simple, and hence, fast. In addition, because the values of dependent variables can be easily calculated for a given sequence, there is no need for feasibility check when producing random solutions.

Crossover. A single point crossover operator has been used in order to produce two offspring from two parent chromosomes. A sequence of jobs represents a feasible chromosome (solution) if it satisfies the two conditions of a valid solution discussed in proving Theorem 1. In a single point crossover, after the first left sections of the chromosomes are exchanged, it is possible that the right sections have duplicate genes. In that case, those genes are replaced by the genes of the other chromosome that are in the same position. Algorithm 5 shows this.

Mutation. The mutation has been used as a local search that can further improve the fitness of existing solutions. With probability $\mu$, two random genes are selected and swapped. Otherwise, no change occurs to the chromosome. After mutation, the chromosome will remain a feasible chromosome. Algorithm 6 shows the mutation scheme used in the proposed algorithm. We repeat the mutation $\mathrm{M}-1$ more times where M is $20 \%$ of the number of jobs. This is because swapping only two jobs decreases effectiveness of this search when the number of jobs increases in larger problems.

Selection. A Roulette Wheel Selection method (C. R. Reeves, 1995) has been adapted, as is shown in Algorithm 7, for selecting either parents for crossover, or

Algorithm 5. Crossover Operator.

```
Input: Two sequences of jobs (parents); \(x_{1}\) and \(x_{2}\).
Output: Two sequences of jobs (offspring chromosomes); \(x_{3}\) and \(x_{4}\).
Note: \(x(i)\) mean \(i\)-th element in \(x\) and \(x(i: j)\) means elements of \(x\) from \(i\) to \(j\).
\(c \leftarrow\) a random integer between 1 and \(n\) (number of jobs)
\(x_{3} \leftarrow x_{1}(1: c)+x_{2}(c+1: n)\)
\(x_{4} \leftarrow x_{2}(1: c)+x_{1}(c+1: n)\)
for \(i \leftarrow 1\) to \(n-c\) do
    if \(x_{3}(c+i)\) exists in \(x_{3}(1: c+i-1)\) do
        for \(j \leftarrow 1\) to \(n\) do
            if \(x_{4}(j)\) does not exist in \(x_{3}(1: c+i-1)\)
                \(x_{3}(c+i) \leftarrow x_{4}(j)\)
                break
for \(i \leftarrow 1\) to \(n-c\) do
    if \(x_{4}(c+i)\) exists in \(x_{4}(1: c+i-1)\) do
        for \(j \leftarrow 1\) to \(n\) do
            if \(x_{3}(j)\) does not exist in \(x_{4}(1: c+i-1)\)
                    \(x_{4}(c+i) \leftarrow x_{3}(j)\)
                break
```


## Algorithm 6. Mutation Operator.

```
Input: A single chromosome (a sequence of jobs), x, and mutation probability,
\mu.
Output: A new chromosome, y.
Note: }x(|)\mathrm{ means }i\mathrm{ -th element in }x\mathrm{ .
y\leftarrowx
r}\leftarrowa\mathrm{ random value between 0 and 1
if }\mu<r\mathrm{ do
    i\leftarrowa random integer between 1 and n (number of jobs)
    j}\mathrm{ a random integer between 1 and n (number of jobs)
    y(i)\leftrightarrowy(j)
```

Algorithm 7. Roulette Wheel Selection.

```
Input: Selection pressure, \(0 \leq s \leq 1\), and a population of sorted chromosomes,
\(P\).
Output: A selected chromosome, \(y\).
Note: P. Cost represents a vector that has the cost of each chromosome in it
\(w \leftarrow \operatorname{man}(P\). Cost \() / /\) cost of the worst solution
\(p \leftarrow e^{-\frac{s \cdot P \cdot \operatorname{Cos} t}{w}}\)
\(p \leftarrow \frac{p}{\sum p}\)
\(r \leftarrow\) a random value between 0 and 1
\(c \leftarrow\) cummulative summation of \(p\)
\(i \leftarrow\) index of the first element in \(c\) which is less than or equal to \(r\)
\(y \leftarrow P(i)\)
```

single chromosomes for mutation, from the current population. A selection pressure equal to 0.8 has been used.

Fitness function. The function used to evaluate fitness (objective function value) of a chromosome (solution) calculates minimum cost of maintenance activities for a given sequence by calculating the current maintenance level after processing each job in the sequence, and adding a maintenance activity and its respective cost only if the next job causes the maintenance level to fall below zero. It also follows Algorithm 4 for calculation of tardiness cost with the only exception that it considers actual processing times for jobs and actual durations for maintenance activities as the MILP model does.

Stopping condition. Convergence has been used as the stopping condition, which mean, when the respective cost of the best solution does not improve for a certain number of iterations, $I$, the algorithm stops and the chromosome that has the minimum cost in the last iteration (generation) is returned as the best solution.

## Setting the Parameters

In this study, a $3^{3}$ factorial design with confidence interval of 0.95 is used to test whether the quality of solutions significantly change for different settings of a parameter. If so, the parameter will be incorporated to the main algorithm that is shown in Figure 5. This algorithm re-adjusts the parameters and runs the GA until a desired gap percentage between the GA and the lower bound is reached. Table 1 shows what levels are used for each factor (parameter) of the GA. Based on a report by (Colin R Reeves, 1997), many authors suggest that a population size as small as 30 is sufficient for producing satisfactory results, we consider it as the starting level. For the other two factors, obviously, possible values are between zero and one.

Table 1. Experimental Design.

| Parameter | Level 1 | Level 2 | Level 3 |
| :--- | :--- | :--- | :--- |
| Population size | 30 | 100 | 200 |
| Crossover probability | 0.2 | 0.5 | 0.8 |
| Mutation probability | 0.2 | 0.5 | 0.8 |



Figure 5. Lower-bound-based GA (LBGA).

For the experiment, 27 combinations are possible and two replications (Montgomery, 2008) are required which implies a total number of 54 trials are run. The results are shown in Table 2 for a randomly-generated test problem of size ( $m=5, n=7, l=3$ ). Note that the response variable (cost) is the best cost produced by the GA minus the exact minimum cost for the same test problem from CPLEX. Analysis of variance for the results of the experiment is shown in Table 3. From the results we see that only the population size is statistically significant for the proposed problem. As a result, we will only incorporate the population size to the LBGA. Using Minitab's Factorial Optimization based on the experiment, a probability of 0.5 for both crossover and mutation, and the maximum possible value for population size minimizes the cost. Increasing these probabilities will negatively affect the time-wise efficiency of the algorithm as more operations are likely to be performed. Decreasing these probabilities, on the other hand, limits the ability of the algorithm in searching the neighborhood.

Time to convergence and the gap between the best cost obtained by the algorithm and the lower bound can be considered as efficiency and quality of the solutions of the algorithm. As shown in Figure 5, the LBGA adjusts statistically significant parameters of the designed GA (population size for this problem) in such a way that a desired level of both measures that can be set by the user are obtained.

### 1.5 Computational Results

Computational results of the proposed solution approach are presented in this section. The NP-hardness of the presented problem is numerically experienced. It is shown that solution times of the exact algorithms have an exponential increase in CPU time for a linear increase in size of the problem. The results of the proposed algorithm will also be compared with the exact solutions from IBM CPLEX in order to validate efficiency and quality of the solutions of the algorithm.

## Test Problem Generation

Table 4 shows how all the test problems used throughout this section are generated. First and second columns show the parameters whose values are to be randomly generated as input and the size of their respective matrices, respectively. Third column shows the ranges within which the random values (of a matrix) are generated and the last column shows considerations in generating the values.

Table 2. Results of the experiment.

| Trial \# | Population Size (A) | Crossover Probability (B) | Mutation Probability (C) | Response (Cost) |
| :---: | :---: | :---: | :---: | :---: |
| 1 | 30 | 0.2 | 0.2 | 3594 |
| 2 | 30 | 0.2 | 0.2 | 7825 |
| 3 | 30 | 0.2 | 0.5 | 17170 |
| 4 | 30 | 0.2 | 0.5 | 5440 |
| 5 | 30 | 0.2 | 0.8 | 3594 |
| 6 | 30 | 0.2 | 0.8 | 3594 |
| 7 | 30 | 0.5 | 0.2 | 3594 |
| 8 | 30 | 0.5 | 0.2 | 5480 |
| 9 | 30 | 0.5 | 0.5 | 3594 |
| 10 | 30 | 0.5 | 0.5 | 3594 |
| 11 | 30 | 0.5 | 0.8 | 3594 |
| 12 | 30 | 0.5 | 0.8 | 5480 |
| 13 | 30 | 0.8 | 0.2 | 3594 |
| 14 | 30 | 0.8 | 0.2 | 8028 |
| 15 | 30 | 0.8 | 0.5 | 5480 |
| 16 | 30 | 0.8 | 0.5 | 5440 |
| 17 | 30 | 0.8 | 0.8 | 3594 |
| 18 | 30 | 0.8 | 0.8 | 5480 |
| 19 | 100 | 0.2 | 0.2 | 0 |
| 20 | 100 | 0.2 | 0.2 | 5440 |
| 21 | 100 | 0.2 | 0.5 | 3594 |
| 22 | 100 | 0.2 | 0.5 | 3594 |
| 23 | 100 | 0.2 | 0.8 | 0 |
| 24 | 100 | 0.2 | 0.8 | 3594 |
| 25 | 100 | 0.5 | 0.2 | 11826 |
| 26 | 100 | 0.5 | 0.2 | 0 |
| 27 | 100 | 0.5 | 0.5 | 0 |
| 28 | 100 | 0.5 | 0.5 | 3594 |
| 29 | 100 | 0.5 | 0.8 | 0 |
| 30 | 100 | 0.5 | 0.8 | 3594 |
| 31 | 100 | 0.8 | 0.2 | 3594 |
| 32 | 100 | 0.8 | 0.2 | 3594 |
| 33 | 100 | 0.8 | 0.5 | 0 |
| 34 | 100 | 0.8 | 0.5 | 0 |
| 35 | 100 | 0.8 | 0.8 | 0 |
| 36 | 100 | 0.8 | 0.8 | 0 |
| 37 | 200 | 0.2 | 0.2 | 5480 |
| 38 | 200 | 0.2 | 0.2 | 0 |
| 39 | 200 | 0.2 | 0.5 | 3594 |
| 40 | 200 | 0.2 | 0.5 | 3594 |
| 41 | 200 | 0.2 | 0.8 | 3594 |
| 42 | 200 | 0.2 | 0.8 | 0 |
| 43 | 200 | 0.5 | 0.2 | 3594 |
| 44 | 200 | 0.5 | 0.2 | 0 |
| 45 | 200 | 0.5 | 0.5 | 0 |
| 46 | 200 | 0.5 | 0.5 | 0 |
| 47 | 200 | 0.5 | 0.8 | 0 |
| 48 | 200 | 0.5 | 0.8 | 0 |
| 49 | 200 | 0.8 | 0.2 | 0 |
| 50 | 200 | 0.8 | 0.2 | 0 |
| 51 | 200 | 0.8 | 0.5 | 0 |
| 52 | 200 | 0.8 | 0.5 | 0 |
| 53 | 200 | 0.8 | 0.8 | 0 |
| 54 | 200 | 0.8 | 0.8 | 0 |

Table 3. Analysis of variance for the GA parameters.

| Source of <br> Variation | Sum of <br> Squares | Degrees of <br> Freedom | Mean Square | $\mathrm{F}_{0}$ | $P$-Value |
| :--- | :--- | :--- | :--- | :--- | :--- |
| A | 180550845 | 2 | 90275422 | 10.84 | $<0.0001$ |
| B | 36384505 | 2 | 18192252 | 2.18 | 0.132 |
| C | 26472269 | 2 | 13236135 | 1.59 | 0.223 |
| $A B$ Interaction | 22000726 | 4 | 5500182 | 0.66 | 0.625 |
| $A C$ Interaction | 23585052 | 4 | 5896263 | 0.71 | 0.594 |
| $B C$ Interaction | 44696386 | 4 | 11174097 | 1.34 | 0.280 |
| $A B C$ Interaction | 22238943 | 8 | 2779868 | 0.33 | 0.945 |
| Error | 224944831 | 27 | 8331290 |  |  |
| Total | 580873557 | 53 |  |  |  |

Table 4. Generation method of test problems.

| Parameter | Size | Range | Generation Method |
| :--- | :---: | :--- | :--- |
| Processing times | $m \times n[1,10]$ | Random (integer, with Uniform distribution) |  |
| Duration of MAs | $1 \times l[1,4]$ | Random (integer, with Uniform distribution) |  |
| Deterioration rates | $m \times n$ | $\times l, 2]$ | Random (fractional, with Uniform distribution) |
| Penalty costs | $1 \times n$ | $[500,600]$ | Random (integer, with Uniform distribution) |
| Due dates | $1 \times n$ | $[10,30]$ | Random (integer, with Uniform distribution) |
| Spare parts costs | $m \times l$ | $[1000,20000]$ | Random (integer, with Uniform distribution) |
| Workforce costs | $1 \times l$ | $[500,2000]$ | Random (integer, with Uniform distribution) |
| Maximum of MLs | $1 \times l$ | NA | (Upper bound of processing times) $\times$ (Upper bound of |
|  |  |  | deterioration rates) |

## Performance

In order to evaluate the quality of the solutions of LBGA, we have solved several test problems of different sizes. We have increased the size of the problems up to a point when CPLEX could no longer solve the problem in a reasonable or predictable time (in the table, "P." in stands for Problem). As shown in Table 5, the objective function value (OFV, the minimum total cost) of the designed LBGA algorithm is either the same as the exact solution from CPLEX or considerably close to it. The time to reach the best solution (in seconds), also, shows that the algorithm is efficient. The CPLEX time, on the other hand, increases exponentially as the size of the problem increases linearly.

## Gap Analysis

In Section 1.4 we proposed a self-tuning lower-bound-based GA (LBGA) for finding the optimal population size that satisfies predetermined levels of both performance and solution quality. An optimal population size found by LBGA for a certain problem size may not be optimal for larger sizes of the problem. This seems to be obvious that the algorithm would consume much more time for finding the optimal set of parameters as the size of the problem increases. In this subsection, we introduce a computational experiment to see whether an optimal set of GA parameters (an optimal population size in this case) for a certain size of the problem can also be considered acceptable for all the remaining larger sizes of the problem.

We want to investigate how an optimal population size for a certain problem size will work for larger sizes of the problem. In Table 6, we generated a test problem for each certain problem size, then we solved it by the GA twice; first we solved it with a fixed population size of 200 which has been obtained by LBGA for $n=10$, then we solved it with a larger population size which linearly increases with respect to the size of the problem.

Figure 6 summarizes Table 6: increasing the population size does not significantly decrease the gap between the GA and the lower bound (quality of solutions was not improved significantly) or the number of iterations before convergence. However, it significantly increases the time to convergence which implies the performance degradation. As a result, we conclude that, for the problem discussed in this paper, if we find an optimal population size for a certain problem size, we can use that population size for any problem size. The LBGA does not need to go through excessive loops to experiment the population size for different sizes of the problem. This finding lets us utilize the GA more efficiently (in terms of time) and effectively (in terms of solution quality).

Table 5. Comparison between CPLEX and LBGA.

| P. | Size |  |  | LBGA |  | CPLEX |  | OFV <br> Gap |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | m | n | I | OFV | Time | OFV | TIME |  |
| 1 | 1 | 3 | 1 | 22212 | 7 | 22212 | 1 | 0.00\% |
| 2 | 1 | 5 | 1 | 28926 | 8 | 28926 | 1 | 0.00\% |
| 3 | 1 | 7 | 1 | 48757 | 9 | 48757 | 1 | 0.00\% |
| 4 | 1 | 9 | 1 | 75063 | 10 | 75063 | 3 | 0.00\% |
| 5 | 1 | 11 | 1 | 118438 | 12 | 118438 | 5 | 0.00\% |
| 6 | 1 | 12 | 1 | 119660 | 13 | 119660 | 14 | 0.00\% |
| 7 | 1 | 13 | 1 | 58674 | 12 | 58465 | 18 | 0.36\% |
| 8 | 1 | 14 | 1 | 302153 | 13 | 301221 | 17050 | 0.31\% |
| 9 | 2 | 6 | 1 | 53879 | 10 | 53879 | 1 | 0.00\% |
| 10 | 2 | 7 | 1 | 91500 | 11 | 91500 | 2 | 0.00\% |
| 11 | 2 | 8 | 1 | 83626 | 10 | 83626 | 2 | 0.00\% |
| 12 | 2 | 9 | 1 | 113672 | 12 | 113578 | 5 | 0.08\% |
| 13 | 2 | 10 | 1 | 163422 | 12 | 163302 | 271 | 0.07\% |
| 14 | 1 | 6 | 2 | 39158 | 9 | 39158 | 1 | 0.00\% |
| 15 | 1 | 7 | 2 | 54842 | 9 | 54842 | 1 | 0.00\% |
| 16 | 1 | 8 | 2 | 85318 | 10 | 85318 | 2 | 0.00\% |
| 17 | 1 | 9 | 2 | 105178 | 10 | 105106 | 10 | 0.07\% |
| 18 | 1 | 10 | 2 | 118578 | 11 | 118578 | 9 | 0.00\% |
| 19 | 1 | 11 | 2 | 195407 | 12 | 195407 | 60 | 0.00\% |
| 20 | 1 | 12 | 2 | 256657 | 13 | 256641 | 630 | 0.01\% |
| 21 | 3 | 6 | 1 | 91549 | 11 | 91549 | 1 | 0.00\% |
| 22 | 3 | 7 | 1 | 144724 | 12 | 142262 | 2 | 1.73\% |
| 23 | 3 | 8 | 1 | 106275 | 13 | 104559 | 6 | 1.64\% |
| 24 | 3 | 9 | 1 | 125309 | 15 | 124150 | 10 | 0.93\% |
| 25 | 3 | 10 | 1 | 222184 | 16 | 218316 | 2185 | 1.77\% |
| 26 | 3 | 4 | 2 | 92550 | 10 | 92550 | 1 | 0.00\% |
| 27 | 3 | 5 | 2 | 147751 | 11 | 147136 | 1 | 0.42\% |
| 28 | 3 | 6 | 2 | 113814 | 12 | 113113 | 1 | 0.62\% |
| 29 | 3 | 7 | 2 | 145661 | 13 | 145551 | 3 | 0.08\% |
| 30 | 3 | 8 | 2 | 273555 | 14 | 272648 | 27 | 0.33\% |
| 31 | 3 | 9 | 2 | 270579 | 15 | 266853 | 3490 | 1.40\% |
| 32 | 1 | 6 | 3 | 63887 | 10 | 63887 | 1 | 0.00\% |
| 33 | 1 | 7 | 3 | 80920 | 10 | 80920 | 2 | 0.00\% |
| 34 | 1 | 8 | 3 | 44858 | 10 | 44858 | 1 | 0.00\% |
| 35 | 1 | 9 | 3 | 197920 | 12 | 197920 | 351 | 0.00\% |
| 36 | 2 | 6 | 3 | 147016 | 11 | 147016 | 2 | 0.00\% |
| 37 | 2 | 7 | 3 | 131718 | 11 | 130270 | 2 | 1.11\% |
| 38 | 2 | 8 | 3 | 214500 | 13 | 214500 | 9 | 0.00\% |

Table 5. Continued.

| P. | Size |  |  | LBGA |  | CPLEX |  | $\begin{aligned} & \text { OFV } \\ & \text { Gap } \end{aligned}$ |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | m | n | I | OFV | Time | OFV | TIME |  |
| 39 | 2 | 9 | 3 | 235188 | 13 | 229921 | 332 | 2.29\% |
| 40 | 3 | 6 | 3 | 137579 | 13 | 137579 | 1 | 0.00\% |
| 41 | 3 | 7 | 3 | 264244 | 14 | 259636 | 23 | 1.77\% |
| 42 | 3 | 8 | 3 | 283053 | 16 | 283053 | 189 | 0.00\% |
| 43 | 7 | 10 | 5 | 1647512 | 38 | 1702766* | 172800 | NA |

* Best feasible solution found after 48 hours.

Table 6. Increasing both population size and problem size ( $m=5, l=3$ ).

| P. | Size <br> (n) | GA with a fixed population size (A) |  |  | Lower <br> Bound | GA with increasing population size (B) |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | Cost | Iterations | Time |  | Cost | Iterations | Time |
| 1 | 10 | 1357822 | 38 | 3.413495 | 1034990 | 1369003 | 6 | 0.776334 |
| 2 | 10 | 1289404 | 27 | 2.739804 | 993260 | 1293785 | 28 | 1.419067 |
| 3 | 10 | 1103666 | 41 | 3.595149 | 850148 | 1113149 | 20 | 1.159057 |
| 4 | 20 | 3514750 | 37 | 6.331777 | 2529097 | 3481338 | 56 | 8.517733 |
| 5 | 20 | 3111348 | 51 | 7.833044 | 1974912 | 3142235 | 62 | 9.067386 |
| 6 | 20 | 3239546 | 66 | 9.319994 | 2252677 | 3287542 | 49 | 7.620991 |
| 7 | 30 | 6008532 | 88 | 17.30267 | 3858826 | 5985798 | 84 | 25.67928 |
| 8 | 30 | 5847222 | 84 | 17.12846 | 3614259 | 5721203 | 104 | 30.90619 |
| 9 | 30 | 6630725 | 71 | 15.27042 | 4197579 | 6665582 | 57 | 19.51996 |
| 10 | 40 | 8775033 | 117 | 29.60771 | 5389715 | 8717491 | 102 | 53.00777 |
| 11 | 40 | 9091908 | 73 | 20.26242 | 5547657 | 9048168 | 142 | 69.59584 |
| 12 | 40 | 9207676 | 98 | 25.85184 | 5733537 | 8960551 | 152 | 75.47809 |
| 13 | 50 | 12698574 | 131 | 40.9435 | 7479909 | 12433575 | 189 | 139.2959 |
| 14 | 50 | 13883341 | 155 | 48.09594 | 8412250 | 13603060 | 178 | 133.5371 |
| 15 | 50 | 11506884 | 147 | 45.01884 | 6489562 | 11541169 | 192 | 141.3605 |
| 16 | 60 | 16180271 | 235 | 80.67539 | 9112971 | 16016216 | 176 | 186.2603 |
| 17 | 60 | 17826556 | 256 | 88.75478 | 10207561 | 17597984 | 183 | 196.9761 |
| 18 | 60 | 16787968 | 90 | 36.53791 | 9466694 | 16384531 | 150 | 167.5249 |
| 19 | 70 | 23500984 | 148 | 63.53675 | 12610303 | 22968423 | 262 | 369.9774 |
| 20 | 70 | 22933999 | 172 | 72.06714 | 12242464 | 22506265 | 270 | 377.1611 |
| 21 | 70 | 21797249 | 220 | 91.40516 | 11796697 | 21214008 | 272 | 392.317 |
| 22 | 80 | 29336445 | 207 | 98.66956 | 15283399 | 28921857 | 402 | 716.7778 |
| 23 | 80 | 26752887 | 178 | 86.65114 | 14281013 | 26108861 | 281 | 524.6216 |
| 24 | 80 | 28608588 | 106 | 55.67532 | 15641739 | 27940407 | 175 | 341.1104 |
| 25 | 90 | 34116742 | 221 | 120.7167 | 17032171 | 33513125 | 314 | 732.3489 |
| 26 | 90 | 35636830 | 172 | 95.79678 | 18580421 | 35017974 | 286 | 681.2377 |
| 27 | 90 | 36721795 | 203 | 112.335 | 18403696 | 36418390 | 191 | 480.5361 |
| 28 | 100 | 44671803 | 225 | 134.331 | 22625879 | 44337511 | 261 | 796.222 |
| 29 | 100 | 40722246 | 285 | 169.049 | 20866059 | 40238496 | 208 | 652.4085 |
| 30 | 100 | 40461783 | 295 | 172.9161 | 20382082 | 40420367 | 178 | 572.7291 |
| 31 | 110 | 48304135 | 179 | 126.0298 | 26468086 | 47139627 | 326 | 1171.559 |
| 32 | 110 | 49434030 | 281 | 181.994 | 24718995 | 48796784 | 216 | 823.8144 |
| 33 | 110 | 51096605 | 263 | 171.1317 | 26367819 | 50639136 | 269 | 990.5269 |
| 34 | 120 | 60261706 | 192 | 148.7522 | 30345652 | 58175023 | 548 | 2281.366 |
| 35 | 120 | 59760090 | 394 | 273.6253 | 30531018 | 59110255 | 412 | 1712.454 |
| 36 | 120 | 57392427 | 436 | 295.0103 | 27164218 | 56861165 | 302 | 1290.498 |
| 37 | 130 | 61052962 | 297 | 226.4157 | 31624493 | 60296896 | 342 | 1725.214 |

Table 6. Continued.

| P. | $\begin{aligned} & \text { Size } \\ & (n) \end{aligned}$ | GA with a fixed population size (A) |  |  | Lower Bound | GA with increasing population size <br> (B) |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | Cost | Iterations | Time |  | Cost | Iterations | Time |
| 38 | 130 | 64150827 | 181 | 149.4786 | 32151965 | 62769807 | 296 | 1520.949 |
| 39 | 130 | 68243101 | 261 | 204.4468 | 33991633 | 67814396 | 272 | 1434.061 |
| 40 | 140 | 69546643 | 445 | 355.842 | 35245794 | 68816417 | 253 | 1566.712 |
| 41 | 140 | 73025400 | 293 | 244.7684 | 35906152 | 70958910 | 318 | 1861.639 |
| 42 | 140 | 79112053 | 229 | 198.1027 | 37178822 | 78634672 | 277 | 1672.875 |
| 43 | 150 | 90118324 | 300 | 265.1772 | 42228222 | 88443419 | 404 | 2706.224 |
| 44 | 150 | 84518094 | 309 | 275.242 | 41435942 | 83861145 | 242 | 1730.565 |
| 45 | 150 | 84135462 | 280 | 254.4936 | 41988981 | 82189148 | 495 | 3275.692 |
| 46 | 160 | 95855660 | 298 | 287.9158 | 46359806 | 94701837 | 269 | 2196.804 |
| 47 | 160 | 99877898 | 174 | 177.5147 | 48788058 | 97021257 | 352 | 2714.185 |
| 48 | 160 | 97575368 | 362 | 339.2074 | 45673937 | 96227839 | 468 | 3519.544 |
| 49 | 170 | $1.16 \mathrm{E}+08$ | 254 | 260.4629 | 54219150 | $1.13 \mathrm{E}+08$ | 476 | 4076.364 |
| 50 | 170 | 97162793 | 195 | 207.859 | 49432663 | 93068701 | 529 | 4522.738 |
| 51 | 170 | $1.12 \mathrm{E}+08$ | 226 | 238.4961 | 52248987 | $1.08 \mathrm{E}+08$ | 374 | 3308.45 |
| 52 | 180 | $1.24 \mathrm{E}+08$ | 367 | 400.517 | 55355808 | $1.21 \mathrm{E}+08$ | 565 | 5411.51 |
| 53 | 180 | 1.17E+08 | 327 | 363.0825 | 56782729 | $1.16 \mathrm{E}+08$ | 283 | 2961.246 |
| 54 | 180 | $1.17 \mathrm{E}+08$ | 602 | 620.4908 | 56072705 | $1.15 \mathrm{E}+08$ | 427 | 4246.796 |



Figure 6. The effect of increasing population size on the performance of the algorithm.

### 1.6 Case Study (Earthmoving Operations)

In this section we will try to show the application of the presented model via a case study. One of the main activities in the early stages of a heavy construction project is earthmoving. This activity is highly dependent on earthmoving machinery. The most commonly used equipment for earthworks are (wheel) loaders, dozers, excavators, and haul trucks. A simplified version of the earthmoving process described by Fu (2013) is as follows. The first step is preparation which is done best by excavators which can dig natural form of material from the earth. Next, in loading step, wheel loaders can load the removed and prepared soil into haul trucks. Finally, in hauling step, haul trucks transport earth to a deposit point by travelling through routes with different slopes and ground conditions.

Typical (preventive) maintenance activities for construction machinery are usually based on the service hours of the machinery. In Table 7, maintenance intervals recommended by one of the manufacturers of heavy construction equipment is listed for the machinery that are required for the simplified earthmoving process (Caterpillar, 2010c) (Caterpillar, 2010b) (Caterpillar, 2010a). These intervals can be considered as $M L_{\max }^{k}$ according to the presented model. Different tasks are included in each maintenance activity. For example, the tasks included in the 50-hour maintenance activity of excavators shown in the table are lubrication of boom, stick and bucket linkage, drive shaft universal joint, etc.

Table 7. Maintenance intervals (hours) recommended by the equipment manufacturer (Caterpillar Inc.).

| Machine | 10 | 50 | 100 | 250 | 500 | 1000 | 2000 | 3000 | 4000 | 5000 | 6000 |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Excavators | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ |  | $\checkmark$ |
| Wheel | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ |  |  |  |  |
| Loaders | $\checkmark$ | $\checkmark$ |  |  | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ |
| (Haul) Trucks | $\checkmark$ |  |  |  |  |  |  |  |  |  |  |

In a project with four locations, in which earth moving operations need to be done, there are three machines (one excavator, one wheel loader, and one truck). Because of the significant distance between these locations, a machine needs to work in one location at a time. In Table 8, the amount of operations in each location is shown. Due dates are also shown along with the penalty for each day of delay (GDOT, 2013). Note that the amount of work that a machine can work in one location can be different from other locations due to the condition of the location. As a result, operation requirements in Table 8 are expressed as number of time periods (days) multiplied by the time a machine can work in each time period (in hours) which can be considered as deterioration rates of MLs because the MLs have been expressed in hours.

Table 8. Operation requirements (days) $\times$ deterioration rates (hours), due dates (days), and penalty for one day delay.

| Location (Jobs) | Excavator | Wheel Loader | Tuck | Due Date | Penalty |
| :---: | :---: | :---: | :---: | :---: | :---: |
| A | $20 \times 5$ | $20 \times 3$ | $40 \times 3$ | 90 | $\$ 211$ |
| B | $14 \times 8$ | $14 \times 6$ | $13 \times 8$ | 60 | $\$ 118$ |
| C | $20 \times 4$ | $21 \times 5$ | $20 \times 5$ | 90 | $\$ 118$ |
| D | $30 \times 3$ | $40 \times 2$ | $30 \times 5$ | 60 | $\$ 346$ |

Average cost of performing a preventive maintenance activity and a responsive maintenance activity (after major failure) for a wheel loader is approximately $\$ 234$ and $\$ 15,652$, respectively (Azadeh et al., 2014). We have used these values to approximate the overall cost of each maintenance activity for each machine, considering the risk of major failure due to missing an MA and relative price of the machines ( $S P_{i k}=\$ 234 \forall i, k, W F_{1}=\$ 800, W F_{2}=$ $\$ 1600, W F_{3}=\$ 3200$ ). Because the first three MAs ( $10,50,100$ hours) are usually done in a fraction of an operational day and where the machine is located, and because 2000 hours MAs and above are not going to be reached they are not considered as MAs (we only consider 250, 500, and 1000). Deterioration rate for ML 250 will be zero for the truck because it does not have the respective MA. We will also consider one day for performing all the maintenance activities. This case study has been solved by IBM ILOG CPLEX Optimization Studio and the results are shown in Table 9. Total cost for this solution is $\$ 38,030.00$, with three maintenance activities.

Table 9. The optimal solution for the case study.

| Location (Jobs) | Sequence | Completion Day | Tardiness (days) |
| :--- | :--- | :--- | :--- |
| A | 2 | 94 | 4 |
| B | 1 | 41 | 0 |
| C | 4 | 156 | 66 |
| D | 3 | 136 | 76 |

### 1.7 Conclusion and Future Research

In this paper, a new permutation flow shop scheduling problem was introduced in which maintenance was incorporated where jobs deteriorate maintenance levels of the machines. We assume that each machine can have different types of maintenance activities corresponding to different maintenance levels and each maintenance activity can be scheduled flexibly. The problem was formulated as a mixed-integer linear program with the objective of minimizing the total cost of tardiness and maintenance. Since the problem was proved to be NP-hard, a special genetic algorithm has been developed. Parameters of the algorithm were statistically tested through a factorial experiment and it was found that only the population size can affect the quality of solutions significantly. Lower bounds were found for two different variations of the problem. A self-tuning genetic algorithm based on the lower bounds was introduced (LBGA). The efficiency and effectiveness of the algorithm is due to its ability to find the best population size, a significant GA parameter for the underlining problem. Through the computational experiment and gap analysis it was found that the optimal population size could be uniquely identified for certain set of problems. A case study of construction machinery scheduling with maintenance considerations was also presented to show one possible application of the problem.

Several assumptions were considered in Section 1.2. Potential extensions of the presented problem, as future works, can be defined by relaxing or changing these assumptions. The followings are other possible considerations in future works:

- Changing the type of flow shop or incorporating the flexible and diverse maintenance activities to other production settings such as parallel machines scheduling.
- Using different meta-heuristics as solution approaches and comparing the results with the presented algorithm.
- Incorporating the random failures into the problem and modeling the problem with stochastic techniques.

The proposed LBGA and computational experiments in Sections 1.4 and 1.5 can be used for other different problems with analogous structures.

## CHAPTER II <br> BI-OBJECTIVE OPTIMIZATION OF THE INTEGRATED FLOW SHOP AND MAINTENANCE SCHEDULING PROBLEM

A version of this chapter was originally published by Javad Seif, Andrew J. Yu, and Fahimeh Rahmanniya:

Seif, J., Yu, A. J., \& Rahmanniyay, F. (2017). Modelling and optimization of a bi-objective flow shop scheduling with diverse maintenance requirements. International Journal of Production Research, DOI:10.1080/00207543.2017.1403660

In this work, I extend my original work that was presented in Chapter I. Mrs. Rahmanniyay has had contributions in problem definition and in the subsection "Bi-Objective Formulation," and Dr. Yu has supported this research.


#### Abstract

In real-world problems machines cannot continuously operate and have to stop for maintenance before they fail. Lack of maintenance can also affect the performance of machines in processing jobs. In this paper, a permutation flow shop scheduling problem with multiple age-based maintenance requirements is modeled as a novel mixed-integer linear program in which the objectives are conflicting. In modeling the problem, we assume that infrequent maintenance can prolong job processing times. One of the objectives is to minimize the total maintenance cost by planning as few maintenance activities as possible to only meet the minimum requirements, and the other objective tries to minimize the total tardiness by sequencing the jobs and planning the maintenance activities in such a way that the processing times are not prolonged and unnecessary maintenance times are avoided. Because of this conflict, an interactive fuzzy-biobjective model is introduced. Application of the model is illustrated through a case study for operations and maintenance scheduling of heavy construction machinery. An effective and efficient solution methodology is developed based on the structure of the problem and tested against commercial solvers and a standard GA. Computational results have verified the efficiency of the proposed solution methodology and show that unlike the proposed method, a generic meta-heuristic that does not consider the unique structure of the problem can become ineffective for real world problem sizes.


### 2.1 Introduction

Although maintenance planning and production scheduling are often studied separately such as in semiconductor manufacturing (Xiaodong et al., 2004), integration of machine maintenance and scheduling has appeared in many researches in the last two decades (Xu et al., 2015). This integration has been proposed for different configurations of manufacturing environments such as
single machine, flow shop, parallel machine, job shop, or flexible flow shop, and for different objective functions such as minimizing makespan, total (expected) completion time, total workload of machines, total workload of critical machines, tardiness, or a combination of them (S. Wang \& Liu, 2014). In this paper, we model and optimize a permutation flow shop scheduling problem with maintenance constraints for two conflicting objective functions, namely maintenance cost and tardiness. This conflict stems from how production jobs deteriorate machines and how deterioration of machines can affect the processing times of the jobs. Then we introduce a solution methodology that effectively and efficiently solves realistic instances of the problem by considering the unique structure of the problem.

The literature related to the integration of maintenance planning and scheduling has been classified by Xu et al. (2015) and Aramon Bajestani and Beck (2015). Xu et al. (2015) classified the literature into two categories based on the duration of maintenance activities. Aramon Bajestani and Beck (2015) divided the literature into two categories. The first category is the same as the first category determined by Xu et al. (2015). The second category, however, is different and addresses those research works which assume that the processing times of the jobs varies based on the maintenance. The interaction between maintenance and production, i.e. how they affect each other, is an interesting and important subject. For example, when considering nurses and doctors as processors and patients as jobs, the time to perform a surgery increases if they have not had a rest between consecutive surgeries. As another example, performance of construction machinery can also degrade leading to longer processing times in earthmoving operations (due to unplanned failures or decreasing performance) if the machinery has not been serviced according to their maintenance requirements. Xiang, Cassady, Jin, and Zhang (2014) modeled deterioration of a manufacturing unit due to production with Markov chains. S. Bock et al. (2012) studied the computational complexity of single machine scheduling problems when there exists a maintenance level for the machine and processing of the jobs deteriorates the maintenance level and a maintenance activity increases it. Yu and Seif (2016) used the same concept and proposed a mixed-integer programming model for flow shop scheduling problems with diverse maintenance activities. C. Y. Lee and Leon (2001) applied a rate which is dependent on maintenance activities to the processing times of the jobs.

In some industries such as heavy construction projects, maintenance costs form a significant portion of the overall costs (Yip et al., 2014). Therefore, it is important to consider the maintenance cost in the objective function along with conventional scheduling criteria such as tardiness. Yu and Seif (2016) considered minimizing maintenance cost as part of the objective function. Ideally, an optimization problem has only one objective that is to be minimized or maximized. However, in most of the real world problems, there more than one
objective is required to be optimized and the objectives are usually conflicting. In order to find an optimal decision, the trade-offs between two or more conflicting objectives should be considered via multi-objective optimization techniques. Also, because some information is incomplete and the environmental coefficients are typically uncertain, the objectives are fuzzy with imprecise aspiration levels. Fuzzy set theory introduced by Zadeh (1965) has been found with extensive applications in various fields, particularly with applications of linear programming (Rommelfanger, 1996). Zimmermann (1978) for the first time proposed application of fuzzy linear programming into conventional multi-objective linear programming (MOLP) problems. Fuzzy multi-objective linear programming (FMOLP) technique deals with problems that include multiple conflicting and fuzzy objectives. As some examples, see Stanciulescu, Fortemps, Installé, and Wertz (2003), R.-C. Wang and Liang (2004) and Liang (2006). Liang (2006) proposed an interactive fuzzy multi-objective linear programming (i-FMOLP) for a supply chain problem that provides a systematic framework for facilitating the fuzzy decision-making process, enabling a decision maker (DM) to interactively modify the fuzzy data and related parameters until a set of satisfactory solutions is obtained. Liang (2009) applied i-FMOLP to project management.

In this paper, a complete interaction where maintenance and production affect each other, is modeled. Jobs can deteriorate the maintenance levels with different deterioration rates and when the average of the maintenance levels is low, the processing times of the jobs can be prolonged. More maintenance activities prevent the increase in processing times but is costly and adds to the total completion times of the jobs and could lead to a greater tardiness. Less maintenance activities can also increase tardiness by prolonging the processing times. This leads to a very complex trade-off between maintenance cost and tardiness. Adapting the thought process of R.-C. Wang and Liang (2004), the solution of fuzzy multi-objective optimization problems benefit from considering DM's imprecise judgments such as, 'the objective function of total maintenance costs should be substantially less than or equal to 100 thousands', or 'total tardiness should be substantially less than or equal to 100 hours or days'. These conflicting objectives are required to be optimized simultaneously by the DM in the framework of fuzzy aspiration levels. An interactive fuzzy multi-objective linear programming (i-FMOLP) method for solving the fuzzy multi objective problems with piecewise linear membership function (PMLF) has been found to be effective for the problem discussed in this paper and is used in solving the problem.

The contributions and significance of this paper are as follows. A practical problem introduced by (Yu \& Seif, 2016) that extends mathematical formulation of the conventional flow shop scheduling problem as a mixed integer linear program (MILP) by incorporating age-based and diverse maintenance activities, is further extended. The impact of maintenance levels (health) of a machine on
processing times, along with deterioration of maintenance levels by processing the jobs, is considered. With an i-FMOLP both production and maintenance objectives are considered simultaneously and optimized in a practical fashion. A solution method is introduced that uses special properties of the presented model and outperforms a standard GA in terms of effectiveness (quality of solutions) and a commercial solver in terms of efficiency (solution time).

The rest of the paper is organized as follows. In Section 2.2, the problem is formulated modeled as a mixed-integer linear program and a summary of assumptions is presented. The interactive fuzzy multi-objective linear programming (iFMOLP) technique used for this problem is introduced in Section 2.3 along with a numerical example. In Section 2.4, a solution methodology is proposed that increases efficiency and effectiveness of a standard GA and outperforms it by confining the solution space and intelligently searching through the solution space. Section 2.5 shows the results of a computational experiment for evaluation of the proposed solution method. A case study in construction projects is introduced and solved in Section 2.6 that shows one of the applications of this work. Conclusions and possible future works as extensions of this paper are discussed in Section 2.7.

### 2.2 Mathematical Formulation

Yu and Seif (2016) incorporated diverse maintenance activities in permutation flow shop scheduling (Chapter 1). An extension of their MIP model is introduced in this section where two conflicting objectives are considered and the processing time of the jobs can be prolonged. The following is a list of assumptions considered in the formulation of the problem.

1. By flow shop we refer to the permutation flow shop without any buffers (blocking flow shop).
2. All the machines have the same set/types of maintenance levels (MLs), and hence, the same set of maintenance activities (MAs).
3. Duration of a specific MA on a specific machine is known and invariable. However, these durations can vary on different machines.
4. When a job is being processed, all the MLs are subject to deterioration according to a linear function by $\delta \times p$ where $\delta$ is the deterioration rate of ML caused by a job after it is processed and $p$ is processing time of the job.
5. Before processing the first job, all the MLs of all the machines are at their maximum.
6. Sufficient/unlimited resources (maintenance spare parts, materials, and workforces, operators, etc.) are available for processing the jobs and performing the MAs.
7. Pre-emption is not allowed.
8. All the MAs are performed to completion.
9. The quantity $\delta \times p$ should always be less than the maximum of the respective maintenance level. Otherwise, the problem is infeasible.
10. Random failures are not considered.
11. As will be explained in Section 2.3, the parameters in the model are considered as crisp while the objectives are fuzzy. Because the desired value of maintenance cost and tardiness are vague and imprecise, the objective functions are fuzzy with imprecise aspiration levels (Paksoy, Pehlivan, \& Özceylan, 2012).
12. Let $\epsilon_{k}$ be the current value of the $k$-th ML of a machine as a fraction of its maximum value before a job is processed by the machine, $0 \leq \epsilon_{k} \leq 1, \forall k$, and $0 \leq \mathrm{E} \leq 1$ be the average of all $\epsilon_{k}$ representing the health state of the machine. Prolonged processing time of the job, $\rho$, is assumed to be defined as

$$
\rho=\left\{\begin{array}{l}
\alpha p, a<\mathrm{E} \leq 1 \\
\beta p, b<\mathrm{E} \leq a \\
\gamma p, 0<\mathrm{E} \leq b
\end{array}\right.
$$

where $p$ is the nominal processing time of the job on that machine, $0<$ $b<a<1$ and $1 \leq \alpha \leq \beta \leq \gamma$, for the special case when the health of a machine has only three states. A generalized form will be considered in the mathematical model in the following section.

These assumptions are the basis for the following model. Assumptions (1) and (7) state the scope of the problem regarding the literature of flow shop scheduling. Assumptions (2-5), (9) and (11-12) describe the properties of the problem that is to be modeled. See the case study in Section 2.6 that shows these properties in an application in construction projects. Finally, Assumptions (6), (8), and (10) highlight the limitations of the model. As is discussed in Section 2.7, the latter assumptions can be dealt with by adjusting the input parameters and they can be addressed in further research works. Let $m, n$, and / be the number of machines, jobs, and maintenance levels (activities), respectively. Following is a list of indices, parameters, and variables used throughout the mathematical formulation of the problem.
$i \quad$ Represents machines where $i=1,2, \ldots, m$
$j \quad$ Represents production jobs where $j=1,2, \ldots, n$
$q \quad$ Represents sequence of jobs (jobs positions) where $q=1,2, \ldots, n$
$k \quad$ represents MLs or their respective MAs where $k=1,2, \ldots, l$
$h \quad$ represents health state of a machine $h=1,2, \ldots, s$
$p_{i j} \quad$ Processing time of job $j$ on machine $i$
$\rho_{i q} \quad$ Processing time of the $q$-th job on machine $i$ (decision variable)
$\delta_{i j k} \quad$ Deterioration rate of maintenance level $k$ (MA type $k$ ) of machine $i$ when job $j$ is processed
$e_{i k} \quad$ Duration of MA type $k$ on machine $i$
$M L_{\text {max }}^{k}$ Maximum of ML type $k$
$S P_{i k} \quad$ Cost of required spare parts and materials for MA type $k$ on machine $i$
$W F_{k} \quad$ Cost of workforce per time unit for performing MA type $k$
$d_{j} \quad$ The time at which job $j$ is due
$\lambda^{h} \quad$ Coefficient with respect to state $h$ that is multiplied by nominal processing times of the jobs to prolong them based on the health state of a machine before the job is processed
$x_{j q} \quad$ Binary variable that takes the value 1 if job $j$ is assigned to position $q$ in the sequence of the jobs, and 0 otherwise
$y_{i q k} \quad$ Binary variable that takes the value 1 when PM type $k$ is performed on machine $i$ before processing the $q$-th job, and 0 otherwise
$M L_{i q}^{k} \quad$ Quantitative value of ML type $k$ of machine $i$ before processing the $q$-th job
$c_{q} \quad$ Completion time of the job assigned to position $q$ (the $q$-th job)
$t_{q} \quad$ Tardiness of the job assigned to position $q$ (amount of lateness in completion of the job)
$v_{i q} \quad$ Waiting time of the machine $i$ for the $q$-th job (idle time)
$w_{i q} \quad$ Waiting time of the $q$-th job for machine $i$ while the machine is busy processing another job
$\Lambda_{i q}^{h} \quad$ Binary variables that takes the value 1 if machine $i$ after processing the $q$ th job is in health state $h$, and 0 otherwise
$z_{i j q}^{h} \quad$ Auxiliary binary variables
$\gamma_{i j q} \quad$ Auxiliary variables
$K, K^{\prime} \quad$ Sufficiently large numbers
The mathematical model presented below has two objective functions ( $z_{1}$ and $z_{2}$ ) subject to the constraints that follow them.

$$
\begin{align*}
& \operatorname{Min} z_{1} \cong \sum_{q=1}^{n} t_{q}  \tag{2.1}\\
& \operatorname{Min} z_{2} \cong \sum_{i=1}^{m} \sum_{q=1}^{n} \sum_{k=1}^{l} y_{i q k}\left(S P_{i k}+e_{i k} W F_{k}\right) \tag{2.2}
\end{align*}
$$

subject to:

$$
\begin{align*}
& t_{q} \geq c_{q}-\sum_{j=1}^{n} x_{j q} d_{j},  \tag{2.3}\\
& q=1,2, \ldots, n, \\
& t_{q} \geq 0,  \tag{2.4}\\
& q=1,2, \ldots, n, \\
& \sum_{q=1}^{n} x_{j q}=1  \tag{2.5}\\
& \sum_{j=1}^{n} x_{j q}=1  \tag{2.6}\\
& v_{i(q+1)}+\sum_{k=1}^{l} y_{i(q+1) k} e_{i k}+\rho_{i(q+1)}+w_{(i+1)(q+1)} \\
& =v_{(i+1)(q+1)}+\sum_{k=1}^{l} y_{(i+1) q k} e_{(i+1) k}+\rho_{(i+1) q}  \tag{2.7}\\
& +w_{(i+1) q}, \\
& M L_{i 1}^{k}=M L_{\text {max }}^{k},  \tag{2.8}\\
& w_{i 1}=0 \text {, }  \tag{2.9}\\
& v_{1 q}=0 \text {, }  \tag{2.10}\\
& v_{i 1}=\sum_{f=1}^{i-1} \rho_{f 1},  \tag{2.11}\\
& w_{1 q}=\sum_{r=1}^{q-1} \rho_{1 r}+\sum_{r=1}^{q} \sum_{k=1}^{l} y_{1 r k} e_{1 k},  \tag{2.12}\\
& M L_{i q}^{k}-\sum_{j=1}^{n} \gamma_{i j q} \delta_{i j k} \geq 0 \text {, }  \tag{2.13}\\
& M L_{i q}^{k} \geq M L_{i(q-1)}^{k}-\sum_{j=1}^{n} \gamma_{i j(q-1)} \delta_{i j k}-y_{i q k} K,  \tag{2.14}\\
& M L_{i q}^{k} \leq M L_{i(q-1)}^{k}-\sum_{j=1}^{n} \gamma_{i j(q-1)} \delta_{i j k}+y_{i q k} K,  \tag{2.15}\\
& M L_{i q}^{k} \geq M L_{\max }^{k}-K\left(1-y_{i q k}\right),  \tag{2.16}\\
& M L_{i q}^{k} \leq M L_{\max }^{k}+K\left(1-y_{i q k}\right), \tag{2.17}
\end{align*}
$$

$$
\begin{align*}
& c_{q}=\sum_{i=1}^{m}\left(w_{i q}+\rho_{i q}+\sum_{k=1}^{l} y_{i q k} e_{i k}\right),  \tag{2.18}\\
& q=1,2, \ldots, n, \\
& j=1,2, \ldots, n \text {, } \\
& q=1,2, \ldots, n \text {, }  \tag{2.19}\\
& h=1,2,3 \text {, } \\
& i=1,2, \ldots, m \text {, } \\
& j=1,2, \ldots, n \text {, }  \tag{2.20}\\
& q=1,2, \ldots, n \text {, } \\
& h=1,2,3 \text {, } \\
& i=1,2, \ldots, m \text {, } \\
& j=1,2, \ldots, n \text {, } \\
& q=1,2, \ldots, n \text {, }  \tag{2.21}\\
& h=1,2,3 \text {, } \\
& i=1,2, \ldots, m \text {, } \\
& u_{i j q}^{h} \geq x_{j q}+\Lambda_{i q}^{h}-1,  \tag{2.22}\\
& \sum_{h=1}^{3} \Lambda_{i q}^{h}=1,  \tag{2.23}\\
& \sum_{k=1}^{l} \frac{M L_{i q}^{k}}{l \cdot M L_{\max }^{k}} \geq 0.66 \Lambda_{i q}^{1}-K^{\prime}\left(\Lambda_{i q}^{2}+\Lambda_{i q}^{3}\right) \text {, }  \tag{2.24}\\
& \sum_{k=1}^{l} \frac{M L_{i q}^{k}}{l \cdot M L_{\max }^{k}} \leq 0.66 \Lambda_{i q}^{2}+K^{\prime}\left(\Lambda_{i q}^{1}+\Lambda_{i q}^{3}\right),  \tag{2.25}\\
& \sum_{k=1}^{l} \frac{M L_{i q}^{k}}{l \cdot M L_{\max }^{k}} \geq 0.33 \Lambda_{i q}^{2}-K^{\prime}\left(\Lambda_{i q}^{1}+\Lambda_{i q}^{3}\right) \text {, }  \tag{2.26}\\
& \sum_{k=1}^{l} \frac{M L_{i q}^{k}}{l \cdot M L_{\max }^{k}} \leq 0.33 \Lambda_{i q}^{3}+K^{\prime}\left(\Lambda_{i q}^{1}+\Lambda_{i q}^{2}\right),  \tag{2.27}\\
& \gamma_{i j q} \leq x_{j q} K^{\prime},  \tag{2.28}\\
& \gamma_{i j q} \leq \rho_{i q},  \tag{2.29}\\
& \gamma_{i j q} \geq \rho_{i q}+\left(x_{j q}-1\right) K^{\prime},  \tag{2.30}\\
& x_{j q}, y_{i q k}, \Lambda_{i q}^{h}, u_{i j q}^{h} \in\{0,1\}, \quad i=1,2, \ldots, m, j=1,2, \ldots, n, q=1,2, \ldots, n \text {, } \\
& M L_{i q}^{k}, c_{q}, t_{q}, v_{i q}, w_{i q} \geq 0, \quad k=1,2, \ldots, l, h=1,2,3 . \tag{2.31}
\end{align*}
$$

Equations (2.1) and (2.2) show the formulation of each objective function. The symbol ' $\cong$ ' is the fuzzy version of ' $=$ ' and refers to fuzzy aspiration levels. For each objective function in the original fuzzy problem, it is assumed that the DM has a fuzzy objective such as, "the objective function should be essentially equal to some value" (Liang 2008). The first objective function $z_{1}$ is equal to the sum of the tardiness of all jobs. The tardiness is calculated for each job position not for the actual jobs. Tardiness of each job is obtained from constraint in Equation (2.3) and lower bound constraint on $t_{q}$ in Equation (2.31); these two constraints together are the linearization form of $t_{q}=\max \left\{0, c_{q}-d_{j}\right\}$. Total maintenance cost $\left(z_{2}\right)$ is obtained by multiplying the workforce and spare parts cost of a potential maintenance activity before processing a job on a machine by the binary variable $y_{i q k}$ that determines whether that potential maintenance activity is realized by the solution. Equations (2.5) and (2.6) together ensure that each job has one and only one position in the sequence of jobs.

Equation (2.7) maintains feasibility between machine and job idle-times. If machine $i$ finishes processing the $(q+1)$-th job before the next machine $(i+1)$ finishes the previous job $(q)$, the job $(q+1)$ has to wait for the machine $(i+1)$. Therefore, $w_{(i+1)(q+1)}$ is positive and $v_{(i+1)(q+1)}$ must be zero. On the other hand, if machine $i$ finishes job $(q+1)$ after the next machine $(i+1)$ finishes job $(q)$, the machine $(i+1)$ has to wait for the job $(q+1)$. Therefore, $v_{(i+1)(q+1)}$ is positive and $w_{(i+1)(q+1)}$ is zero.

According to Assumption (2.6), all the maintenance levels prior to the first job are at their maximum and hence no MA is performed before processing the first job. This is expressed in Equation (2.8). The first job ( $q=1$ ) does not wait in buffer for any of the machines because it is processed first by all the machines, as expressed in Equation (2.9). The first machine in the flow shop also does not wait for any of the jobs as expressed in Equation (2.10). In Equation (2.11), the idle times for machines 2 to $m$ with respect to the first job $(q=1)$ is equal to the sum of the processing times of the job on the previous machines. Equation (2.12) calculates the waiting time of the jobs for the first machine. Equation (2.13) ensures that the maintenance levels do not fall below zero during or after processing of the next job. The auxiliary variable $\gamma_{i j q}$ substitutes the nonlinear term $\rho_{i q} \cdot x_{j q}$. To linearize these nonlinear terms, constraints in Equations (2.28$2.30)$ are added. Equations (2.14-2.17) are the linearized form of the following equation that calculates the values of the maintenance levels after processing the jobs ( $\forall i, k, q=2 \ldots n$ ):

$$
M L_{i q}^{k}=\left\{\begin{aligned}
M L_{i(q-1)}^{k}-\sum_{j=1}^{n} x_{j(q-1)} \rho_{i j} \delta_{i j k}, & y_{i q k}=0 \\
M L_{\max }^{k}, & y_{i q k}=1
\end{aligned}\right.
$$

Completion time of each sequenced job is equal to sum of its processing times and its waiting times in the buffer. This is expressed in Equation (2.18). In Equation (2.19), prolonged processing times are calculated. The original form of this equation, when there are only three states for the health of a machine, is:

$$
\rho_{i q}=\sum_{j=1}^{n} p_{i j} x_{j q}\left(\alpha \Lambda_{i q}^{1}+\beta \Lambda_{i q}^{2}+\gamma \Lambda_{i q}^{3}\right), \forall i, q
$$

where $\Lambda_{i q}^{h}, h=1,2,3$ is a binary variable that shows whether machine $i$ is in the health state $h$ before processing the $q$-th job. The value within the parentheses will be either $\alpha, \beta$, or $\gamma$. In Equation (2.19), $u_{i j q}^{h}$ substitutes the nonlinear term $x_{j q} \Lambda_{i q}^{h}$. Constraints in Equations (2.20-2.22) are added for linearization of this term. For simplicity, in this paper we have considered only three states for the health of the machines. Constraints in Equation (2.23) ensure that only one of the states of a machine is realized before processing a job. Constraints in Equations (2.24-2.27) determine which state is realized based on the health of a machine before a job is processed, in linear forms. Constraints in Equations (2.28-2.30) linearize the term $\gamma_{i j q}=\rho_{i q} x_{j q}$ that is used to get the correct processing time for the job that occupies a certain position in the sequence; then, $\gamma_{i j q}$ is used throughout the model as the processing time of job $j$ on machine $i$ if it is assigned to position $q$ in the sequence of the jobs. Equation (2.31) ensures that all the variables can take only the values that are within their boundaries.

## Bi-Objective Formulation

In the proposed problem, a DM may have a vague and fuzzy idea about the desired values of the objective functions based on the current maintenance budget and criticality or priority of the current jobs (hence, the level of tardiness). For example, the DM may specify that the tardiness should be "somewhat less than" 100 hours and the maintenance cost "substantially less than" 100 thousand dollars. A multi-objective technique that is suitable for the proposed problem should have the following characteristics: it should be easy to understand and interact with; it should capture the fuzzy nature of the DM's ideas about the values of the objective functions; and it should allow the DMs change these values based on the changes that happen at operational levels.

Liang (2006) proposes an interactive fuzzy multi-objective linear program (i-FMOLP) for a supply chain problem that provides a systematic framework for facilitating the fuzzy decision-making process in a problem where the aspiration levels of the DMs with respect to the objectives are fuzzy. Interactive techniques are more desirable in some applications because they yield a single preferred solution (Hannan, 1981; Liang, 2006). i-FMOLP uses a membership function that
helps DMs quantify the degree of their fuzzy satisfaction for each objective function. A membership function quantifies the vague and fuzzy statements of the DMs. A scale between 0.0 and 1.0 is stablished with 0.0 representing the worst value for the objective function with no degree of satisfaction and 1.0 representing the best value for the objective function with full satisfaction. In order to form a membership function, the DM is asked to specify the degree of satisfaction for several values of each objective based on current operational limitations and requirements, or his or her preference, knowledge and experience. The solution of i-FMOLP optimization problems benefit from considering DM's imprecise judgements (R.-C. Wang \& Liang, 2004).

Among different multi-objective optimization methods, the i-FMOLP with piecewise linear membership function (PLMF) has been found to be suitable for the problem discussed in this paper. The main advantage of using PLMF is that, by eliciting only a small finite number of values for the membership function from the DM, we can approximate the intermediate points between the elicited points in the membership function (Hannan, 1981). In this section, the i-FMOLP method is introduced for formulation of the bi-objective problem incorporating both objective functions. Next, a simple numerical example along with analysis of the solution is presented.

## i-FMOLP

The outline of the interactive solution procedure of the proposed i-FMOLP method for solving fuzzy multi-objective problems is as follows (R.-C. Wang \& Liang, 2004):

Step 1: Formulate the MOLP (which is a MILP, to be precise)
Step 2: Solve the problem for each objective function and obtain the best possible value for each OF.

Step 3: Specify the value of the membership functions $f_{g}\left(z_{g}\right), g=1,2$ for several values of each objective function $Z_{g}$, to determine the satisfaction levels for each objective function based on the DM's preference, experience and knowledge. As it is shown in Table 10, $X_{g, i}, i=0, \ldots, p+1$ show different values of each objective function in order to cover the full range of the DM's aspiration levels. $P$ is the number of points between the best values ( $X_{1, P+1}, X_{2, P+1}$ ) and the worse values ( $X_{1,0}, X_{2,0}$ ) of the objective functions. The best values (lower bounds for

Table 10. Membership function $f_{g}\left(z_{g}\right), g=1,2$.

| Parameters | $>X_{10}$ | $X_{10}$ | $X_{11}$ | $X_{12}$ | $\ldots$ | $X_{1 P}$ | $X_{1, P+1}$ | $<X_{1, P+1}$ |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $z_{1}$ | 0 | 0 | $q_{11}$ | $q_{12}$ | $\ldots$ | $q_{1 P}$ | 1.0 | 1.0 |
| $f_{1}\left(z_{1}\right)$ | $>X_{20}$ | $X_{20}$ | $X_{21}$ | $X_{22}$ | $\ldots$ | $X_{2 P}$ | $X_{2, P+1}$ | $<X_{2, P+1}$ |
| $z_{2}$ | 0 | 0 | $q_{21}$ | $q_{22}$ | $\ldots$ | $q_{2 P}$ | 1.0 | 1.0 |
| $f_{2}\left(z_{2}\right)$ | Note: $0 \leq q_{i j} \leq 1, q_{i j} \leq q_{i, j+1}, i=1,2, j=1,2, \ldots, P$. |  |  |  |  |  |  |  |

the objective function values) are obtained from Step 2. However, the worse values (upper bounds) are determined by the DM; the range that is defined by the best and the worse values should cover realistic values that the objective functions can take. Also, it should be carefully taken into account by the DM that any value greater than or equal to the worse value has a satisfaction degree of 0 .

Step 4: For each pair $\left(z_{g}, f_{g}\left(z_{g}\right)\right), g=1,2$ derive the formulation according to (R.C. Wang \& Liang, 2004) and (Liang, 2008). Piecewise linear membership functions are specified to represent the fuzzy sets involved. By introducing the auxiliary variable $L$, the original fuzzy bi-objective problem can be converted into an equivalent ordinary MILP model that can be solved efficiently using the standard exact methods. The auxiliary variable $L(0<L<1)$ represents overall DM's satisfaction with the determined objective values (Liang, 2008). If the solution is $L=1$, then each objective is fully satisfied; if $0<L<1$, then all of the objectives are satisfied at the level of $L$, and if $L=0$, then none of the objectives are satisfied. A detailed explanation of this method can be found in Liang (2006).

Step 5: Solve the following model:
$\operatorname{Max} L$

Subject to:

$$
\begin{align*}
& L \leq-\left(\frac{t_{g 2}+t_{g 1}}{2}\right)\left(d_{g 1}^{-}-d_{g 1}^{+}\right)-\left(\frac{t_{g 3}+t_{g 2}}{2}\right)\left(d_{g 2}^{-}-d_{g 2}^{+}\right)-\cdots \\
& -\left(\frac{t_{g(P+1)}+t_{g P}}{2}\right)\left(d_{g P}^{-}-d_{g P}^{+}\right)+\left(\frac{t_{g(P+1)}+t_{g 1}}{2}\right) z_{g} \\
& +\frac{S_{g(P+1)}+S_{g 1}}{2}, \quad g=1,2  \tag{2.36}\\
& z_{g}+d_{g e}^{-}-d_{g e}^{+}=X_{g e}, \quad g=1,2, e=1,2, \ldots, P  \tag{2.37}\\
& d_{g e}^{+}, d_{g e}^{-} \geq 0, g=1,2, e=1,2, \ldots, P \tag{2.38}
\end{align*}
$$

Equations (2.3) to (2.31).
where $t_{g e}$ and $S_{g e}$ are calculated according to Liang (2008).
Step 6: If the user is not satisfied with the results, and the objective function values are not acceptable, go to Step 3.

## Numerical Example

A small size of the problem, where number of machines $m=3$, number of jobs $n=6$, and number of maintenance types $l=2$, is presented here as a numerical example. The data used for this example are obtained from random test problem generator as explained in Section 2.5.

The interactive solution procedure uses the proposed i-FMOLP method for this problem. First, the initial solution for each of the objective functions using the ordinary crisp MILP model is determined. The optimal value for tardiness $\left(z_{1}\right)$ is 83 and optimal value for total maintenance cost $\left(z_{2}\right)$ is 105 . Then, we specify the degree of membership $f_{g}\left(z_{g}\right)$ for $g=1,2$ regarding several values for each of the objective functions. Table 11 shows the piecewise linear membership functions for initial solutions.

Table 11. Piecewise linear membership functions for the numerical example.

| Parameters | Values |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $z_{1}$ | $<83$ | 83 | 88 | 98 | 110 | $>110$ |
| $f_{1}\left(z_{1}\right)$ | 1 | 1 | 0.8 | 0.5 | 0 | 0 |
| $z_{2}$ | $<105$ | 105 | 108 | 115 | 120 | $>120$ |
| $f_{2}\left(z_{2}\right)$ | 1 | 1 | 0.8 | 0.5 | 0 | 0 |

In functional situations, the ordinary single-objective MILP solution for each of the fuzzy objective functions often produce a starting point for specifying the piecewise linear membership function, and both intervals must cover the MILP solution (Liang, 2006). Finally, we formulate the FMOLP model using the initial solutions and the presented bi-objective MILP:

Max L
Subject to:
$L \leq-0.0036\left(d_{11}^{-}-d_{11}^{+}\right)-0.0114\left(d_{12}^{-}-d_{12}^{+}\right)-0.035\left(\sum_{q=1}^{n} t_{q}\right)+4.08$
$L \leq-0.1063\left(d_{21}^{-}-d_{21}^{+}\right)+0.0012\left(d_{21}^{-}-d_{21}^{+}\right)$

$$
-0.1450 \sum_{i=1}^{m} \sum_{q=1}^{n} \sum_{k=1}^{l} y_{i q k}\left(S P_{i k}+e_{i k} W F_{k}\right)+17
$$

$\sum_{q=1}^{n} t_{q}+\left(d_{11}^{-}-d_{11}^{+}\right)=88$

$$
\begin{aligned}
& \sum_{q=1}^{n} t_{q}+\left(d_{12}^{-}-d_{12}^{+}\right)=98 \\
& \sum_{i=1}^{m} \sum_{q=1}^{n} \sum_{k=1}^{l} y_{i q k}\left(S P_{i k}+e_{i k} W F_{k}\right)+\left(d_{21}^{-}-d_{21}^{+}\right)=108 \\
& \sum_{i=1}^{m} \sum_{q=1}^{n} \sum_{k=1}^{l} y_{i q k}\left(S P_{i k}+e_{i k} W F_{k}\right)+\left(d_{22}^{-}-d_{22}^{+}\right)=115
\end{aligned}
$$

Equations (2.3-2.31).
IBM ILOG CPLEX Optimization Studio has been used for solving the initial models and the i-FMOLP. The optimal values are $z_{1}=87$ and $z_{2}=105$, and the optimal DM's satisfaction value, when the i-FMOLP is solved, is $L=0.843$. If the solution is not satisfactory, the DM may attempt to improve the results interactively by adjusting the related parameters to obtain a satisfactory solution.

Furthermore, the DM can change the membership functions for both fuzzy objectives. Table 12 shows the new values for the membership functions. Consequently, supposedly improved solutions are again $z_{1}=87$ and $z_{2}=105$, but the overall degree of DM's satisfaction increases sharply to $L=0.92$. Therefore, changing the values of the membership functions does not necessarily change the optimal value of the individual objectives, but it changes the satisfaction degree based on the DM's preference.

Table 12. Piecewise linear membership functions for the improved results.

| Parameters | Values |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| $z_{1}$ | $<83$ | 83 | 93 | 100 | 110 | $>110$ |
| $f_{1}\left(z_{1}\right)$ | 1 | 1 | 0.8 | 0.5 | 0 | 0 |
| $z_{2}$ | $<105$ | 105 | 110 | 118 | 120 | $>120$ |
| $f_{2}\left(z_{2}\right)$ | 1 | 1 | 0.8 | 0.5 | 0 | 0 |

There are several significant management implications regarding the application of the i-FMOLP method. The fuzzy goal programming method of (Hannan, 1981) adopted in this work, which uses the piecewise linear membership function and the minimum operator, yields efficient solutions to fuzzy multi-objective programming problems. It follows that maximization of two or more membership functions is best achieved by maximizing the minimum membership grade. In addition, the coefficients and related parameters of this problem such as processing time, deterioration rate and cost of spare parts which affect the value of objectives are normally fuzzy or imprecise because of some information being incomplete or unobtainable. This model gives a tool to
the DM typically to solve the problem by optimizing simultaneously two conflicting objectives in the framework of imprecise aspiration levels (Liang, 2008). Table 13 compares the results obtained by the single objective MILPs and the i-FMOLP model. The results show that by solving the i-FMOLP, instead of each one of the single objective MILPs, we can simultaneously optimize both objectives with an acceptable trade-off.

Table 13. Solution comparison.

|  | Problem 1 | Problem 2 | i-FMOLP |
| :--- | :---: | :---: | :---: |
| Objective Function | Min. $z_{1}$ | Min. $z_{2}$ | Max. $L$ |
| Value of $L$ | $100 \%$ | $100 \%$ | $92 \%$ |
| Value of $z_{1}$ | 83 | 145 | 87 |
| Value of $z_{2}$ | 110 | 105 | 105 |

Different values for the degree of membership for each one of the objectives $\left(z_{1}, f_{1}\left(z_{1}\right)\right)$ and $\left(z_{2}, f_{2}\left(z_{2}\right)\right)$ for the numerical example is shown in Table 5 and the results are shown in Table 14. As the results show, the value of memberships for each objective function affects the overall level of satisfaction and the decision variables. This has significant implications. First, the most important task of the DM is to carefully specify the degree of membership for each objective function; second, the DM may flexibly revise the range of value of the degree of membership to yield satisfactory solutions (R.-C. Wang \& Liang, 2004).

Table 14. Different membership values for $\left(\boldsymbol{z}_{1}, \boldsymbol{f}_{\mathbf{1}}\left(\mathbf{z}_{1}\right)\right)$ and $\left(\boldsymbol{z}_{2}, \boldsymbol{f}_{2}\left(\mathbf{z}_{\mathbf{2}}\right)\right)$.

| Run | Parameters | Values |  |  |  |  |  |
| :--- | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 1 | $z_{1}$ | $<83$ | 83 | 93 | 100 | 110 | $>110$ |
| 1 | $f_{1}\left(z_{1}\right)$ | 1 | 1 | 0.8 | 0.5 | 0 | 0 |
| 1 | $z_{2}$ | $<105$ | 105 | 110 | 118 | 120 | $>120$ |
| 1 | $f_{2}\left(z_{2}\right)$ | 1 | 1 | 0.8 | 0.5 | 0 | 0 |
| 2 | $z_{1}$ | $<83$ | 83 | 85 | 88 | 90 | $>90$ |
| 2 | $f_{1}\left(z_{1}\right)$ | 1 | 1 | 0.8 | 0.5 | 0 | 0 |
| 2 | $z_{2}$ | $<105$ | 105 | 110 | 113 | 115 | $>115$ |
| 2 | $f_{2}\left(z_{2}\right)$ | 1 | 1 | 0.8 | 0.5 | 0 | 0 |
| 3 | $z_{1}$ | $<83$ | 83 | 85 | 90 | 92 | $>92$ |
| 3 | $f_{1}\left(z_{1}\right)$ | 1 | 1 | 0.8 | 0.5 | 0 | 0 |
| 3 | $z_{2}$ | $<105$ | 105 | 108 | 110 | 115 | $>115$ |
| 3 | $f_{2}\left(z_{2}\right)$ | 1 | 1 | 0.8 | 0.5 | 0 | 0 |
| 4 | $z_{1}$ | $<83$ | 83 | 88 | 98 | 110 | $>110$ |
| 4 | $f_{1}\left(z_{1}\right)$ | 1 | 1 | 0.8 | 0.5 | 0 | 0 |
| 4 | $z_{2}$ | $<105$ | 105 | 108 | 115 | 120 | $>120$ |
| 4 | $f_{2}\left(z_{2}\right)$ | 1 | 1 | 0.8 | 0.5 | 0 | 0 |

The change in the objective function values and satisfaction degrees are depicted in Figure 7. Please note that in this numerical example, the values of the second objective (maintenance) have been scaled down.

Table 15. Objective function values for the optimal solutions.

| Objective | Run1 | Run2 | Run3 | Run4 |
| ---: | :--- | :--- | :--- | :--- |
| $L$ | 0.92 | 0.8 | 0.67 | 0.84 |
| $z_{1}$ | 87 | 85 | 88.296 | 87 |
| $z_{2}$ | 105 | 110 | 110 | 105 |



Figure 7. The value of the two objectives and $L$ against different membership values.

### 2.3 Solution Methodology

As will be shown and discussed in the next section, commercial solvers, like CPLEX, do not perform satisfactorily in terms of the computation time in finding the optimal solution, specifically for realistic problems with relatively large number of variables and constraints. On the other hand, meta-heuristic algorithms that are usually used as an alternative, suffer from not guaranteeing optimality. In both cases, the problem stems from not having insight to the structure of the problem and the solution space. In this section, a solution algorithm is introduced with the help of a few developed theorems that give insight into the unique properties of the problem. This insight helps make the solution space significantly smaller, and hence the search faster. A Genetic Algorithm is then used as a tool for searching through the confined solution space to find optimal or near-optimal solutions, efficiently.

## Confining the Solution Space

The solution space of the presented problem in Section 2.2 is very large due to having many variables. We will show that, by knowing the values of the variables $x_{j q}$ and $y_{i q k}, \forall i, j, q, k$, the values for other variables can be derived. In other words, all the other variables depend on $x_{j q}$ and $y_{i q k}$. The variables that can determine the values of the objective functions are $t_{q}$ and $y_{i q k}$. Variable $t_{q}$ itself can be determined if $x_{j q}$ and $y_{i q k}$ are known. $x_{j q}, \forall j, q$ represents the sequence of the jobs and $y_{i q k}$ shows the position of maintenance activities (MAs). A pair ( $\boldsymbol{S}, \boldsymbol{y}$ ) represents a solution in the confined solution space, where $S$ is a vector for the sequence and y is a matrix whose elements are $y_{i q k}$. Algorithm 1 shows a procedure that takes such a solution as input and calculates the optimal values of other variables for the given solution, and then calculates the values of the two objective functions that can be directly used for calculation of the satisfaction degree of the solution. The outputs of the algorithm are the values of the objective functions and feasibility status of the solution.

Feasibility-check of a solution as an output of the algorithm is trivial; if the value of any of the variables $M L_{i q}^{k}, \forall k, i, q$ calculated by Algorithm 1 turns out to be negative, the solution is infeasible. Regarding the constrains of the model in Section 2.3, the algorithm ensures that the values taken by the variables do not violate any of the constraints. An overview of Algorithm 1 is as follows. First, all the maintenance levels are set equal to their maximum before processing the first job for each machine. Then, the levels are decreased by the deterioration caused by the jobs that have been processed. If a maintenance activity takes place, the level is reset to its maximum. Processing times of the jobs are adjusted based on the average maintenance level of the machine, according to Assumption 12 and the mathematical formulation of the problem in Section 2.3.

Next, the completion time of each job on each machine $\left(C_{i q}, \forall i, q\right)$ is calculated. This is done by first, calculating the completion time of the first job in each machine ( $C_{i 1}, \forall i$ ), then, calculating the completion time of all the jobs on the first machine ( $C_{1 q}, \forall q=2, \ldots, n$ ), and finally, calculating the completion times of all the jobs except the first job on all the machines except the first machine $\left(C_{1 q}, \forall q=2, \ldots, n, i=2, \ldots, m\right)$. The waiting time of the jobs and the maintenance times are considered in calculation of the completion times. Algorithm 1 ends by calculating the value of each objective function.

By using Algorithm 1, we reduced a solution space defined by all the variables to a solution space defined only by $x_{j q}$ and $y_{i q k}, \forall i, j, q, k$. Next we show that the confined solution space can be further confined in a special case where the processing times are not affected by the maintenance levels. The following propositions are the basis in designing Algorithm 2. Algorithm 2 derives the optimal values of $y_{i q k}, \forall i, q, k$ for a given sequence $\boldsymbol{S}\left(x_{j q}, \forall i, q\right)$. Because

Algorithm 1. Obtaining the value of objective functions from only a sequence $(S)$ and positions of maintenance activities $(y)$.
Inputs: $\boldsymbol{S}$ and $\boldsymbol{y}$
Outputs: $z_{1}, z_{2}$ and feasibility
for $i=1, \ldots, m$ do
$M L_{i 1}^{k}=M L_{\text {max }}^{k}, \forall k$
end for
for $i=1, \ldots, m$ do
for $q=2, \ldots, n$ do
$j=S_{q}$
$j^{\prime}=S_{q-1}$
for $k=1, \ldots, l$ do
if $y_{i q k}=1 \mathrm{do}$

$$
M L_{i q}^{k}=M L_{\max }^{k}
$$

else do

$$
M L_{i q}^{k}=M L_{i(q-1)}^{k}-p_{i j^{\prime}} \delta_{i j^{\prime} k}
$$

end if
end for
$\mu=\sum_{k=1}^{l} \frac{M L_{i q}^{k}}{M L_{\text {max }}^{k}} / l$
if $\mu<0.33$ do
$p_{i j}=2 \times p_{i j}$
else if $0.33 \leq \mu<0.66$ do

$$
p_{i j}=1.5 \times p_{i j}
$$

end if
end for
end for
for $i=1, \ldots, m$ do
$C_{i 1}=\sum_{i^{\prime}=1}^{i} p_{i^{\prime} s_{1}}$
end for
for $q=2, \ldots, n$ do
$C_{1 q}=C_{1(q-1)}+p_{1 S_{q}}+\sum_{k=1}^{l} e_{1 k} y_{1 q k}$
end for
for $i=2, \ldots, m$ do
for $q=2, \ldots, n$ do
if $C_{i(q-1)}>C_{(i-1) q}$ do
$w_{i q}=C_{i(q-1)}-C_{(i-1) q}$

```
Algorithm 1. Continued.
        end if
            \(C_{i q}=C_{(i-1) q}+w_{i q}+p_{i s_{q}}+\sum_{k=1}^{l} e_{i k} y_{i q k}\)
        end for
end for
for \(q=1, \ldots, n\) do
    \(t_{q}=\max \left\{0, C_{m q}-d_{S_{q}}\right\}\)
\(z_{1}=\sum_{q=1}^{n} t_{q}\)
\(z_{2}=\sum_{i=1}^{m} \sum_{q=1}^{n} \sum_{k=1}^{l} y_{i q k}\left(S P_{i k}+e_{i k} W F_{k}\right)\)
if \(M L_{i q}^{k} \geq 0, \forall i, k, q\) do
    feasibility \(=\) True
else do
    feasibility \(=\) False
```

maintenance levels do affect the processing times in the presented problem, the algorithm updates the processing times based on the maintenance levels. Also, the local search discussed in the next subsection will improve the output of Algorithm 2.

Proposition 1. For a fixed sequence, $\boldsymbol{S}^{0}$, and when the processing times of the jobs are not dependent on the average of maintenance levels, by setting all the positions in $\boldsymbol{y}$ equal to 0 and resetting a specific maintenance position $y_{i_{0} q_{0} k_{0}}$ to 1 only if otherwise $M L_{i_{0}\left(q_{0}+1\right)}^{k_{0}}$ becomes negative, the first objective function $\left(z_{1}\right)$ will be minimized.

Proof. Let sol ${ }^{0}=\left(\boldsymbol{y}^{0}, \boldsymbol{S}^{0}\right)$ be a feasible solution, using Algorithm 1. If there exists another feasible solution sol ${ }^{1}=\left(\boldsymbol{y}^{1}, \boldsymbol{S}^{0}\right)$ in which $\boldsymbol{y}^{1}$ differs from $\boldsymbol{y}^{0}$ only in one position, i.e. $y_{i_{0} q_{0} k_{0}}^{1}=1$ and $y_{i_{0} q_{0} k_{0}}^{0}=0$, because for a fixed sequence, the maintenance activities that are placed between the jobs are the only variables that can change the value of the first objective function (tardiness), $\therefore z_{1}^{\text {sol }}{ }^{1}<z_{1}^{\text {sol }}{ }^{0}$. Infeasibility happens only when $M L_{i q}^{k}<0, \forall i, k, q$. Therefore, the optimal solution is the one in which $y_{i_{0} q_{0} k_{0}}^{0}=1$ if and only if otherwise $M L_{i_{0}\left(q_{0}+1\right)}^{k_{0}}<0$.

Proposition 2. For a fixed sequence, $\boldsymbol{S}^{0}$, and when the processing times of the jobs are not dependent on the average of maintenance levels, by setting all the positions in $\boldsymbol{y}$ equal to 0 and resetting a specific maintenance position $y_{i_{0} q_{0} k_{0}}$ to 1 only if otherwise $M L_{i_{0}\left(q_{0}+1\right)}^{k_{0}}$ becomes negative, the second objective function ( $z_{2}$ ) will be minimized.

Proof. Let sol ${ }^{0}=\left(\boldsymbol{y}^{0}, \boldsymbol{S}^{0}\right)$ be a feasible solution, using Algorithm 1. If there exists another feasible solution sol ${ }^{1}=\left(\boldsymbol{y}^{1}, \boldsymbol{S}^{0}\right)$ in which $\boldsymbol{y}^{1}$ differs from $\boldsymbol{y}^{0}$ only in one position, i.e. $y_{i_{0} q_{0} k_{0}}^{1}=1$ and $y_{i_{0} q_{0} k_{0}}^{0}=0$, because in the formulation of the second objective function, $z_{2}=\sum_{i=1}^{m} \sum_{q=1}^{n} \sum_{k=1}^{l} y_{i q k}\left(S P_{i k}+e_{i k} W F_{k}\right)$, the only variable is the position of the maintenance activities, $y_{i q k}$ which is a binary variable, $\therefore z_{2}^{\text {sol } l^{1}}<$ $z_{2}^{\text {sol }}{ }^{0}$. Infeasibility happens only when $M L_{i q}^{k}<0, \forall i, k, q$. Therefore, the optimal solution is the one in which $y_{i_{0} q_{0} k_{0}}^{0}=1$ if and only if otherwise $M L_{i_{0}\left(q_{0}+1\right)}^{k_{0}}<0$.

When the processing times are not prolonged (due to a low average maintenance level), the total deterioration of the maintenance levels of a machines caused by processing all the jobs is constant. Total deterioration for maintenance type $k_{0}$ of machine $i_{0}$ is $T D_{i_{0} k_{0}}=\sum_{j=1}^{n} \delta_{i_{0} j k_{0}} p_{i_{0} j}$. The theoretical value for the minimum number of maintenance activities of type $k_{0}$ required on machine $i_{0}$ can
be defined as $M A_{i_{0} k_{0}}^{\min }=\left\lfloor\frac{T D_{i_{0} k_{0}}}{M L_{\text {max }}^{k_{0}}}\right\rfloor$; any number less that $M A_{i_{0} k_{0}}^{\min }$ leads to a negative value for the maintenance level while processing the jobs.

For a set of jobs, when trying to perform as few maintenance activities as possible according to the above prepositions, different sequences can lead to different number of MAs. For example, consider four jobs with the following deteriorations (processing time multiplied by deterioration rate), a given machine $i_{0}$ and a given maintenance level $k_{0}$; deteriorations for the four jobs are $D_{1}=$ $0.65, D_{2}=0.40, D_{3}=0.50, D_{4}=0.35$, and $M L_{\max }^{k_{0}}=1.00$. The theoretical minimum number of MAs is $M A_{i_{0} k_{0}}^{\min }=\left[\frac{0.65+0.40+0.50+0.35}{1.00}\right]=\lfloor 1.9\rfloor=1$, and the minimum number of MAs with respect to sequences $\boldsymbol{S}^{1}=[1,4,3,2]$ and $\boldsymbol{S}^{2}=[1,2,3,4]$ are 1 and 2, respectively. The positions of MAs for $\boldsymbol{S}^{1}$ and $\boldsymbol{S}^{2}$ are [1,4, MA, 3,2] and [ $1, M A, 2,3, M A, 4]$, respectively. Removing any of these MAs from these sequences leads to a negative value for the maintenance level while processing the job that is placed after the removed MA. For a set of jobs that is to be processed by machine $i_{0}$, and when only considering maintenance level $k_{0}$, there exist a set of sequence $S^{*}$ whose minimum number of MAs is the closest to $M A_{i_{0} k_{0}}^{\min }$. In this example, $M A_{i_{0} k_{0}}^{S^{1}}=M A_{i_{0} k_{0}}^{\min }$.

For a given sequence, we can use a procedure to find the minimum number of MAs. Algorithm 2 finds the minimum number of MAs while considering all the machines and all the maintenance levels/activities. It also updates the processing times for a case when the jobs can be prolonged. When the jobs can be prolonged, there is no guarantee that this procedure gives the minimum number of MAs. This is because when the processing times of the jobs are prolonged due to low average maintenance levels, the amount of deterioration also increases since the amount of deterioration is equal to deterioration rate multiplied by the processing time. This can lead to more MAs.

Unnecessary maintenance activities can be added before processing jobs in order to prevent the increase in processing times, and hence prevent the increase in deteriorations. For the first objective function (tardiness), this does not guarantee the improvement in the objective; although the processing times will not be prolonged, the duration of maintenance activities adds to the total tardiness. The only way to guarantee that processing times will not be prolonged is by adding a MA before each job for each ML of each machine. This will significantly increase the number of MAs and completion times, and worsen the value of both objective functions. For the second objective function (maintenance cost), unnecessary MAs should not be added in the first place because adding them contradicts the goal of doing so; the goal of adding unnecessary MAs is to make sure that the processing times are not prolonged, and hence deterioration is not increased and eventually less MAs are expected to be needed.

Algorithm 2. Finding the minimum number of maintenance activities for a given sequence.
Input: A sequence, $\boldsymbol{S}$, and other input parameters.
Output: Positions of maintenance activities, $\boldsymbol{y}, \boldsymbol{M A}$, and $\boldsymbol{M L}$.
for $i=1, \ldots, m$ do
$M A_{i k}=0, \forall k$
$M L_{i 1}^{k}=M L_{\text {max }}^{k}, \forall k$
for $q=2, \ldots, n$ do
for $k=1, \ldots, l$ do
$M L_{i q}^{k}=M L_{i(q-1)}^{k}-p_{i S_{q-1}} \delta_{i S_{q-1} k}$
end for
$\mu=\sum_{k=1}^{l} \frac{M L_{L(q-1)}^{k}}{M L_{\max }^{k}} / l$
update processing time of job $S_{q}$ on machine $i\left(p_{i S_{q}}\right)$ according to
Assumption 12, using $\mu$
for $k=1, \ldots, l$ do
if $M L_{i q}^{k}-p_{i S_{q}} \delta_{i S_{q} k}<0$ do
$M L_{\text {iq }}^{k}=M L_{\text {max }}^{k}$

$$
M A_{i k}=M A_{i k}+1
$$

$$
y_{i q k}=1
$$

else do
$y_{i q k}=0$
end if
end for
end for
end for

The presented algorithms and the discussion that followed them give an insight into the confined solution space that helps making the search more efficient and effective.

## Designing a search in the Solution Space

It was shown that the pair ( $\boldsymbol{S}, \boldsymbol{y}$ ) can represent the solution space. $\boldsymbol{S}$ is the sequence of $n$ jobs and there are $n$ ! different realizations for it. For a given sequence, the value of optimal $\boldsymbol{y}$ with respect to the second objective can be found using Algorithm 2. Let ( $\boldsymbol{S}^{0}, \boldsymbol{y}^{0}$ ) be a solution for which Algorithm 2 has determined $\boldsymbol{y}^{0}$ based on $\boldsymbol{S}^{0}$. Therefore, the number of MAs is minimal. In order to search the neighborhood of this solution for finding solutions that can improve tardiness (the first objective), random MAs can be added to positions where the sum of maintenance duration and nominal processing time is less than the prolonged processing time. Based on the structure of the solution and the type of search that was described, a Genetic Algorithm is designed for efficiently searching through the solution space and converging to a near-optimal solution.

The crossover operator is applied to a sequence of jobs, in order to escape from local optimality and search everywhere in the solution space. The mutation operator serves as a local search for a given sequence and is applied to the $\boldsymbol{y}$ section of the chromosome. In the following subsections, these operators and other settings of the GA are introduced in detail.

## Global search by crossover

When using a binary representation for jobs and their sequence, the crossover operator can produce "illegal" or "bad" solutions that mean infeasible solutions and there are a few methods for handling this issue (Bierwirth, 1995; Yamada \& Nakano, 1997). These methods make the algorithms computationally more expensive. Because the goal here is only to search different sequences, a singlepoint crossover operator is used to produce two offspring from two parent chromosomes. In a single point crossover applied to a sequence $\boldsymbol{S}$, after the first left sections of the chromosomes are exchanged, it is possible that the right sections have duplicate genes (a job can be seen more than once in the sequence). In that case, those genes are replaced by the genes of the other chromosome that are in the same position. This method is implemented as shown in Algorithm 3.

## Local search by mutation

The mutation operator has been used as a local search on $\boldsymbol{y}$ for a given sequence $\boldsymbol{S}^{0}$ that can further improve the fithess of existing solutions. As was explained in the previous subsection, Algorithm 2 determines the positions of

Algorithm 3. Finding new sequences from existing sequences using a
crossover operator.

```
Input: Two sequences of jobs (parents), \(\boldsymbol{S}^{1}\) and \(\boldsymbol{S}^{2}\).
Output: Two new sequences of jobs (offspring chromosomes), \(\boldsymbol{S}^{3}\) and \(\boldsymbol{S}^{4}\).
Note: \(S_{i}^{a}\) means \(i\)-th element in \(\boldsymbol{S}^{a}\) and \(S_{i: j}^{a}\) means elements of \(\boldsymbol{S}^{a}\) from \(i\) to \(j\).
\(c \leftarrow\) a random integer between 1 and \(n\) (number of jobs)
\(\boldsymbol{S}^{3} \leftarrow S_{1: c}^{1}+S_{c+1: n}^{2}\)
\(\boldsymbol{S}^{4} \leftarrow S_{1: c}^{2}+S_{c+1: n}^{1}\)
for \(i=1, \ldots, n-c\) do
    if \(S_{c+i}^{3}\) exists in \(S_{1: c+i-1}^{3}\) do
        for \(j=1, \ldots, n\) do
            if \(S_{j}^{4}\) does not exist in \(S_{1: c+i-1}^{3}\)
                    \(S_{c+i}^{3} \leftarrow S_{j}^{4}\)
                    break
            end if
        end for
    end if
end for
for \(i=1, \ldots, n-c\) do
    if \(S_{c+i}^{4}\) exists in \(S_{1: c+i-1}^{4}\) do
        for \(j=1, \ldots, n\) do
            if \(S_{j}^{3}\) does not exist in \(S_{1: c+i-1}^{4}\)
                \(S_{c+i}^{4} \leftarrow S_{j}^{3}\)
                break
            end if
        end for
    end if
end for
```

maintenance activities for a given sequence in such a way that it minimizes the second objective function. Here, a position that is a Possible Improvement Point (PIP) is located and with a probability $\mu_{1}$, a MA is placed in that position. A PIP is one in which the nominal processing time of the job that is placed after the point ( $i, q, k$ ) plus duration of the respective maintenance activity is less than the prolonged processing time of the job, if the job has been prolonged due to low average maintenance level of the machine.

## Setting the parameters

The Roulette Wheel Selection method by C. R. Reeves (1995) has been adapted for selecting parents for crossover, or a single chromosome for mutation, from the current population. The output of Algorithm 1 is used for evaluating fitness (satisfaction degree) of a chromosome and for checking the feasibility of a chromosome. Convergence has been used as the stopping condition; when the satisfaction degree of the fittest chromosome does not improve for a certain number of iterations, $I$, the GA stops and the chromosome that has the maximum satisfaction degree in the last iteration (generation) is returned as the best solution. The values used for the parameters of a GA that works best for this problem, are chosen as follows: 20 for the maximum number of iterations; 500 for population size; 30\% of chromosomes were chosen for crossover; 30\% of the chromosomes were chosen for mutation; and each the maintenance positions in each chromosome were mutated with the probability of 0.05 . The methodology introduced by Yu and Seif (2016) was used in determining the values of these parameters. In the next section a computational experiment for performance evaluation of the presented solution methodology is presented.

### 2.4 Computational Results

In this section, performance of the proposed solution methodology is evaluated through analyzing the results of a set of computational experiments. A series of random test problems with different sizes are generated which are used for evaluation of the efficiency and effectiveness of the presented solution methodology (which will be referred to as ALG). Time-to-convergence and quality of solutions of ALG will be tested against those from the commercial solver, IBM CPLEX. In addition, performance of ALG will be compared with a standard GA. In the standard GA (denoted by GA) the algorithms introduced in Section 2.3 will not be used. The premise of the comparison between ALG and GA is to show how the solution methodology that was introduced in Section 2.3 can improve a meta-heuristic that blindly searches through the solution space. In ALG, only the sequence of the jobs is generated randomly. The positions of the MAs are derived from a given sequence using Algorithms 1 and 2. However, in the GA, not only the sequence is generated randomly, the positions of maintenance
activities are also generated randomly. In short, ALG is an improved GA that considers the unique features of the solution space and benefits from the algorithms that were introduced in Section 2.3. The same values for the parameters of GA and ALG were used in the computational experiments.

## Test problems generation

Table 16 and Table 17 explain how the input data for each problem in the computational experiments is generated. Table 16, adapted from Yu and Seif (2016) similar to Table 4, shows the dimensions (size) of each parameter, the range within which the values of the elements of each parameter are generated, and the type of random distribution used in generating these values. As the last four rows of Table 16 show, another input for each test problem is the table of membership function. Each problem is solved twice with a standard GA; once with $z_{1}$ as the objective function and once $z_{2}$ as the objective function. Because both objectives are in the form of minimization and GA's solution is not guaranteed to be optimal, a large fraction of the output of GA is considered as the first point in forming the table.

Table 17 shows how the membership function table is formed. The values $[0.00,0.50,0.75,1.00]$ were chosen arbitrarily for $f_{g}\left(z_{g}\right), g=1,2$. As was explained in Section 2.2, in practice the values of the membership functions are determined by decision makers and there is no "best" value for these values in the membership function tables. What matters in the computational experiment is to have an estimate of the range of values that each objective take, and then using the same table in a test problem that is to be solved by the all three solvers. The method shown in Table 7 provides us with such an estimate. How many points are considered for $f_{g}\left(z_{g}\right), g=1,2$ and how good are the values of $z_{g}, g=1,2$, also should not matter in the computational experiment as long as the same values are used as a basis for comparison. It should be re-emphasized that the same membership function table were used in each test problem by all three solution methods, namely CPLEX, GA and ALG.

Table 16. Generation method of test problems, adapted from Yu and Seif (2016).

| Parameter | Size | Range | Generation Method |
| :--- | :---: | :--- | :--- |
| Processing times | $m \times n$ | $[1,10]$ | Random (integer, Uniform distribution) |
| Duration of MAs | $1 \times l$ | $[1,4]$ | Random (integer, Uniform distribution) |
| Deterioration rates | $m \times n \times l$ | $[0,2]$ | Random (fractional, Uniform distribution) |
| Penalty costs | $1 \times n$ | $[500,600]$ | Random (integer, Uniform distribution) |
| Due dates | $1 \times n$ | $[10,30]$ | Random (integer, Uniform distribution) |
| Spare parts costs | $m \times l$ | $[1000,20000]$ | Random (integer, Uniform distribution) |
| Workforce costs | $1 \times l$ | $[500,2000]$ | Random (integer, Uniform distribution) |
| Maximum of MLs | $1 \times l$ |  | (U.B. of Processing times) $\times$ (U.B. of Deterioration rates) |
| $z_{1}^{G A}$ | $1 \times 4$ |  | Solving a GA with $z_{1}$ as the only objective (fitness) function |
| $z_{2}^{G A}$ | $1 \times 4$ |  | Solving a GA with $z_{2}$ as the only objective (fitness) function |

Table 17. Piecewise linear membership functions for test problems.

| Parameters | Values |  |  |  |  |  |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| $\mathrm{z}_{1}$ | $<0.75 \times z_{1}^{G A}$ | $0.75 \times z_{1}^{G A}$ | $1.25 \times z_{1}^{G A}$ | $1.75 \times z_{1}^{G A}$ | $2.00 \times z_{1}^{G A}$ | $>2.00 \times z_{1}^{G A}$ |
| $\mathrm{f}_{1}\left(\mathrm{z}_{1}\right)$ | 1.00 | 1.00 | 0.75 | 0.50 | 0.00 | 0.00 |
| $\mathrm{z}_{2}$ | $<0.75 \times z_{2}^{G A}$ | $0.75 \times z_{2}^{G A}$ | $\mathbf{1 . 2 5} \times z_{2}^{G A}$ | $\mathbf{1 . 7 5 \times z _ { 2 } ^ { G A }}$ | $\mathbf{2 . 0 0} \times z_{2}^{G A}$ | $>2.00 \times z_{2}^{G A}$ |
| $\mathrm{f}_{2}\left(\mathrm{z}_{2}\right)$ | 1.00 | 1.00 | 0.75 | 0.50 | 0.00 | 0.00 |

## Computational experiment

The proposed solution algorithm and a standard GA were implemented in MATLAB and the performance was compared against that of CPLEX 12.5 (2012). Our computational experiments were performed on an i7-3770 @ 3.40 gigahertz Intel processor with 8.00 gigabytes system memory. In order to evaluate the quality of the solutions obtained from ALG, we have solved numerous test problems with different sizes. The size of the problem is defined based on the number of machines $(m)$, number of jobs $(n)$, and number of maintenance levels ( $l$ ). 27 different problem sizes can be defined by all possible combinations of $m=\{1,2,5\}, n=\{6,10,15\}$, and $l=\{1,2,3\}$. These ranges for the size of the problem were considered mainly based on the values that ( $m, n, l$ ) can take in real world problems, and also in such a way that a portion of them can be solved by CPLEX so the performance of the proposed solution method can be evaluated. For each of these sizes, 30 random problem instances were solved.

Table 18 compares the computational times of three solution methods, in seconds, for realistic problem sizes. Both GA and ALG were set to stop if the solution does not improve for 20 iterations. Table 9 also shows the number of iterations of GA and ALG before the 20 iterations (convergence). The number of iterations reported in Table 18 should be added to 20 to get the actual number of iterations. For example, when the number of iterations before the final 20 iterations is 158 , it means that the actual number of iterations were $158+20=178$, but the objective function value had not improved in iterations 159 to 178 and hence the algorithm stopped. The blank entries for CPLEX columns show that CPLEX could not find the optimal solution in 300 seconds. This time limit was chosen because it was observed that when a problem cannot be solved within 300 seconds, solution time can vary from a few minutes to several hours. Usually, if it takes several hours to solve the problem, the software stops due to memory related issues, and can disrupt the computational experiment. As the results suggest, ALG is faster than a standard GA. Also, unlike CPLEX, the average time of ALG does not grow exponentially when the problem size increases. The results also imply that the main contributor to the computational complexity of the problem is the number of jobs.

Although the computation time of the standard GA was reported in Table 18 for comparison purposes, as shown in Table 19, the standard GA is not always able to find a feasible solution. This is mainly because the probability of generating a feasible solution for a very large solution space decreases when the size of the problem increases. On the other hand, according to the results, ALG always finds a feasible solution. Table 20 compares the results of ALG with those of the standard GA and CPLEX. For CPLEX, the solver has been run for a few hours for test problems for which the value of OFV could be converged. As can be inferred from Table 20, while the gap between the value of other objective functions $\left(z_{1}, z_{2}\right)$ can be very high or low, ALG has an average of 7 percent gap

Table 18. Comparison of computation times (seconds) for problems with a realistic size.

| Problem$\begin{aligned} & \text { Class } \\ & (m, n, l) \end{aligned}$ | CPLEX <br> Time |  | Standard GA |  |  |  | Proposed Solution Method (ALG) |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  | Time |  | Iterations |  | Time |  | Iterations |  |
|  | Avg. | Max./Min | Avg. | Max. | Avg. | Max. | Avg. | Max. | Avg. | Max. |
| $(1,6,1)$ | 0.98 | 1.52/0.00 | 16.08 | 25.31 | 8.00 | 20 | 14.39 | 29.28 | 4.37 | 21 |
| $(1,6,2)$ | 1.18 | 1.58/0.00 | 23.75 | 42.60 | 19.43 | 48 | 19.32 | 38.91 | 11.57 | 36 |
| $(1,6,3)$ | 1.42 | 2.09/1.01 | 30.05 | 56.31 | 26.47 | 55 | 22.02 | 41.66 | 15.07 | 50 |
| $(1,10,1)$ | 8.69 | 21.39/2.09 | 33.56 | 61.33 | 38.70 | 71 | 28.93 | 46.81 | 30.07 | 63 |
| $(1,10,2)$ |  | /6.83 | 49.01 | 77.98 | 55.53 | 99 | 37.99 | 81.67 | 41.10 | 118 |
| $(1,10,3)$ |  | /24.95 | 64.93 | 113.7 6 | 77.00 | 139 | 41.63 | 87.79 | 41.30 | 108 |
| $(1,15,1)$ |  | /16.22 | 55.56 | 89.49 | 67.60 | 125 | 44.12 | 91.07 | 55.37 | 137 |
| $(1,15,2)$ |  |  | 77.89 | $\begin{gathered} 157.4 \\ 8 \end{gathered}$ | 87.03 | 186 | 43.56 | 134.70 | 41.00 | 158 |
| $(1,15,3)$ |  |  | 87.93 | $\begin{gathered} 158.1 \\ 7 \end{gathered}$ | 86.43 | 175 | 40.62 | 140.00 | 32.30 | 160 |
| $(2,6,1)$ | 1.17 | 1.93/0.00 | 23.00 | 41.03 | 17.03 | 50 | 17.23 | 34.58 | 8.80 | 28 |
| $(2,6,2)$ | 1.82 | 4.47/1.00 | 42.99 | 85.99 | 45.53 | 101 | 31.32 | 61.63 | 29.30 | 74 |
| $(2,6,3)$ | 2.75 | 9.21/1.01 | 59.08 | 95.87 | 61.70 | 116 | 26.08 | 60.88 | 16.93 | 61 |
| $(2,10,1)$ |  | /23.77 | 49.54 | 90.02 | 55.30 | 122 | 34.09 | 94.18 | 35.83 | 129 |
| $(2,10,2)$ |  | 176.45 | 72.86 | 138.7 9 | 78.17 | 170 | 26.77 | 85.09 | 17.57 | 101 |
| $(2,10,3)$ |  | /244.42 | 92.35 | $\begin{gathered} 181.2 \\ 7 \end{gathered}$ | 83.60 | 191 | 21.35 | 50.29 | 6.10 | 44 |
| $(2,15,1)$ |  |  | 87.09 | $\begin{gathered} 142.8 \\ 6 \end{gathered}$ | 92.20 | 168 | 37.96 | 132.74 | 34.67 | 174 |
| $(2,15,2)$ |  |  | 99.61 | $\begin{gathered} 258.3 \\ 4 \end{gathered}$ | 86.10 | 251 | 26.20 | 64.81 | 9.77 | 59 |
| $(2,15,3)$ |  | /299.97 | 47.84 | $\begin{gathered} 283.3 \\ 3 \end{gathered}$ | 27.27 | 248 | 23.73 | 39.40 | 3.43 | 20 |
| $(5,6,1)$ | 4.19 | 15.42/1.38 | 54.74 | $\begin{gathered} 106.3 \\ 6 \\ 134.2 \end{gathered}$ | 54.47 | 123 | 18.01 | 37.87 | 5.63 | 26 |
| $(5,6,2)$ | 21.24 | 66.82/3.44 | 71.61 | $\begin{gathered} 104.4 \\ 8 \\ 209.4 \end{gathered}$ | 57.47 | 113 | 23.34 | 50.18 | 5.70 | 35 |
| $(5,6,3)$ |  | 14.18 | 73.00 | 8 8 | 52.90 | 195 | 22.51 | 25.72 | 2.00 | 2 |
| $(5,10,1)$ |  |  | 74.10 | $\begin{gathered} 157.4 \\ 9 \end{gathered}$ | 65.87 | 161 | 19.95 | 34.03 | 4.33 | 22 |
| $(5,10,2)$ |  |  | 31.66 | 250.4 9 | 12.20 | 201 | 23.32 | 26.60 | 2.00 | 2 |
| $(5,10,3)$ |  |  | 25.73 | 29.13 | 2.00 | 2 | 26.91 | 31.05 | 2.00 | 2 |
| $(5,15,1)$ |  |  | 37.32 | 216.6 4 | 18.97 | 187 | 21.37 | 23.99 | 2.00 | 2 |
| $(5,15,2)$ |  |  | 26.33 | 30.08 | 2.00 | 2 | 28.18 | 31.41 | 2.00 | 2 |
| $(5,15,3)$ |  |  | 32.91 | 36.89 | 2.00 | 2 | 34.72 | 38.05 | 2.00 | 2 |
| Average | - | - | 53.35 | $\begin{gathered} 121.1 \\ 4 \end{gathered}$ | 47.44 | 123.00 | 27.98 | 59.79 | 17.12 | 60.59 |

Table 19. Comparison of feasibility success (percentage) for problems with a realistic size.

| Problem Class <br> $(m, n, l)$ | Standard GA | Proposed Solution Method (ALG) |
| :---: | :---: | :---: |
| $(1,6,1)$ | $100 \%$ | $100 \%$ |
| $(1,6,2)$ | $100 \%$ | $100 \%$ |
| $(1,6,3)$ | $100 \%$ | $100 \%$ |
| $(1,10,1)$ | $100 \%$ | $100 \%$ |
| $(1,10,2)$ | $100 \%$ | $100 \%$ |
| $(1,10,3)$ | $100 \%$ | $100 \%$ |
| $(1,15,1)$ | $100 \%$ | $100 \%$ |
| $(1,15,2)$ | $97 \%$ | $100 \%$ |
| $(1,15,3)$ | $97 \%$ | $100 \%$ |
| $(2,6,1)$ | $100 \%$ | $100 \%$ |
| $(2,6,2)$ | $100 \%$ | $100 \%$ |
| $(2,6,3)$ | $100 \%$ | $100 \%$ |
| $(2,10,1)$ | $100 \%$ | $100 \%$ |
| $(2,10,2)$ | $100 \%$ | $100 \%$ |
| $(2,10,3)$ | $83 \%$ | $100 \%$ |
| $(2,15,1)$ | $100 \%$ | $100 \%$ |
| $(2,15,2)$ | $80 \%$ | $100 \%$ |
| $(2,15,3)$ | $33 \%$ | $100 \%$ |
| $(5,6,1)$ | $100 \%$ | $100 \%$ |
| $(5,6,2)$ | $97 \%$ | $100 \%$ |
| $(5,6,3)$ | $87 \%$ | $100 \%$ |
| $(5,10,1)$ | $87 \%$ | $100 \%$ |
| $(5,10,2)$ | $27 \%$ | $100 \%$ |
| $(5,10,3)$ | $0 \%$ | $100 \%$ |
| $(5,15,1)$ | $23 \%$ | $100 \%$ |
| $(5,15,2)$ | $0 \%$ | $100 \%$ |
| $(5,15,3)$ | $0 \%$ | $100 \%$ |

Table 20. Comparison of quality of objective function value (OFV, satisfaction degree) for problems with a realistic size.

| Proble m Class ( $m, n, l$ ) | $\begin{aligned} & \text { CPLEX } \\ & \text { OFV } \end{aligned}$ |  |  | Standard GA |  |  |  |  |  | Proposed Solution Method (ALG) |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  | OFV |  |  | Gap |  |  | OFV |  |  | Gap |  |  |
|  | Avg. | Min. | Max. | Avg. | Min. | Max. | OFV | $z_{1}$ | $z_{2}$ | Avg. | Min. | Max. | OFV | $z_{1}$ | $z_{2}$ |
| $(1,6,1)$ | 0.90 | 0.82 | 0.97 | 0.88 | 0.69 | 0.97 | 2\% | -6\% | 0\% | 0.89 | 0.82 | 0.97 | 0\% | -8\% | 0\% |
| $(1,6,2)$ | 0.94 | 0.89 | 1.09 | 0.93 | 0.86 | 1.09 | 1\% | -18\% | 0\% | 0.93 | 0.89 | 1.09 | 1\% | -4\% | 0\% |
| $(1,6,3)$ | 0.95 | 0.89 | 1.03 | 0.94 | 0.87 | 1.02 | 1\% | -4\% | 0\% | 0.94 | 0.89 | 1.03 | 1\% | 3\% | 0\% |
| $(1,10,1)$ | 0.90 | 0.88 | 0.93 | 0.87 | 0.63 | 0.92 | 3\% | 2\% | 50\% | 0.87 | 0.39 | 0.93 | 3\% | 4\% | 50\% |
| $(1,10,2)$ | 0.93 | 0.89 | 0.97 | 0.89 | 0.79 | 0.96 | 4\% | 11\% | 0\% | 0.89 | 0.79 | 0.95 | 4\% | 15\% | 15\% |
| $(1,10,3)$ | 0.95 | 0.89 | 1.00 | 0.88 | 0.77 | 0.97 | 7\% | 16\% | 10\% | 0.88 | 0.78 | 0.96 | 7\% | 29\% | 21\% |
| $(1,15,1)$ | 0.90 | 0.89 | 0.93 | 0.87 | 0.79 | 0.92 | 4\% | 9\% | 20\% | 0.86 | 0.75 | 0.92 | 5\% | 8\% | -20\% |
| $(1,15,2)$ | 0.94 | 0.89 | 0.97 | 0.83 | 0.00 | 0.93 | 12\% | 11\% | 26\% | 0.86 | 0.79 | 0.93 | 9\% | 35\% | 55\% |
| $(1,15,3)$ | 0.95 | 0.94 | 0.97 | 0.77 | 0.00 | 0.94 | 20\% | 57\% | 64\% | 0.83 | 0.76 | 0.92 | 13\% | 38\% | 48\% |
| $(2,6,1)$ | 0.89 | 0.76 | 0.95 | 0.89 | 0.76 | 0.95 | 0\% | 0\% | 0\% | 0.89 | 0.76 | 0.95 | 0\% | 0\% | 0\% |
| $(2,6,2)$ | 0.94 | 0.89 | 0.98 | 0.91 | 0.84 | 0.98 | 4\% | 12\% | 0\% | 0.91 | 0.84 | 0.96 | 3\% | -1\% | 0\% |
| $(2,6,3)$ | 0.96 | 0.89 | 1.08 | 0.90 | 0.73 | 1.00 | 6\% | 8\% | 10\% | 0.90 | 0.81 | 0.99 | 7\% | 15\% | 12\% |
| $(2,10,1)$ | 0.90 | 0.87 | 0.93 | 0.84 | 0.71 | 0.90 | 6\% | 4\% | 0\% | 0.84 | 0.74 | 0.92 | 6\% | 12\% | 16\% |
| $(2,10,2)$ | 0.94 | 0.90 | 0.97 | 0.84 | 0.75 | 0.91 | 11\% | 30\% | 28\% | 0.83 | 0.78 | 0.89 | 12\% | 35\% | 40\% |
| $(2,10,3)$ | 0.96 | 0.93 | 1.00 | 0.70 | 0.00 | 0.94 | 27\% | 19\% | 17\% | 0.82 | 0.78 | 0.89 | 15\% | 43\% | 36\% |
| $(2,15,1)$ | 0.90 | 0.84 | 0.93 | 0.83 | 0.78 | 0.88 | 8\% | 21\% | 37\% | 0.82 | 0.72 | 0.90 | 9\% | 19\% | 41\% |
| $(2,15,2)$ | 0.93 | 0.90 | 0.95 | 0.59 | 0.00 | 0.87 | 37\% | 8534\% | -77\% | 0.80 | 0.76 | 0.84 | 14\% | 30\% | 41\% |
| $(2,15,3)$ | 0.95 | 0.93 | 0.98 | 0.18 | 0.00 | 0.84 | 81\% | 10487\% | -80\% | 0.81 | 0.78 | 0.84 | 15\% | 48\% | 43\% |
| $(5,6,1)$ | 0.90 | 0.85 | 0.94 | 0.85 | 0.77 | 0.94 | 5\% | 10\% | 8\% | 0.86 | 0.81 | 0.90 | 4\% | 6\% | 8\% |
| $(5,6,2)$ | 0.94 | 0.91 | 0.97 | 0.78 | 0.00 | 0.91 | 17\% | 28\% | 34\% | 0.87 | 0.84 | 0.89 | 8\% | 14\% | 23\% |
| $(5,6,3)$ | 0.96 | 0.92 | 1.00 | 0.60 | 0.00 | 0.88 | 37\% | 69\% | 59\% | 0.86 | 0.83 | 0.88 | 10\% | 30\% | 32\% |
| $(5,10,1)$ | 0.89 | 0.85 | 0.91 | 0.65 | 0.00 | 0.86 | 27\% | 48\% | 49\% | 0.82 | 0.79 | 0.86 | 7\% | 13\% | 12\% |
| $(5,10,2)$ | 0.93 | 0.91 | 0.96 | 0.10 | 0.00 | 0.80 | 89\% | 13440\% | -84\% | 0.83 | 0.80 | 0.86 | 11\% | 20\% | 30\% |
| $(5,10,3)$ | 0.96 | 0.94 | 0.98 |  |  |  |  |  |  | 0.84 | 0.80 | 0.86 | 12\% | 27\% | 34\% |
| $(5,15,1)$ | 0.86 | 0.79 | 0.89 |  |  |  |  |  |  | 0.81 | 0.77 | 0.84 | 6\% | 4\% | 11\% |
| $(5,15,2)$ | 0.91 | 0.88 | 0.93 |  |  |  |  |  |  | 0.82 | 0.79 | 0.84 | 9\% | 17\% | 29\% |
| $(5,15,3)$ | 0.93 | 0.90 | 0.97 |  |  |  |  |  |  | 0.83 | 0.81 | 0.86 | 11\% | 26\% | 31\% |
| Avg. | 0.93 | 0.88 | 0.97 | 0.65 | 0.40 | 0.82 | 29\% | 2284\% | -7\% | 0.86 | 0.78 | 0.91 | 7\% | 18\% | 23\% |

with CPLEX's converged solution which seems acceptable. This gap can be further improved if the number of iterations before convergence is set to a higher number. ALG shows that it can provide a satisfactory solution for any problem size within a minute.

### 2.5 Case Study

In this section we present a case study to show an application for the presented problem. The input data and description of this case study are reproduced (with slight changes) from the case study by Yu and Seif (2016) (as presented in Chapter I) who solve the problem for a single objective function. After presentation of the data and describing the case study, we will present the solution and managerial implications for the bi-objective problem discussed in this paper.

One of the main activities in the early stages of a heavy construction project is earthmoving. This activity is highly dependent on earthmoving machinery. The most commonly used equipment for earthworks are (wheel) loaders, dozers, excavators, and haul trucks. A simplified version of the earthmoving process described by Fu (2013) is as follows. The first step is preparation which is done best by excavators which can dig natural form of material from the earth. Next, in loading step, wheel loaders can load the removed and prepared soil into haul trucks. Finally, in hauling step, haul trucks transport earth to a deposit point by travelling through routes with different slopes and ground conditions.

Typical (preventive) maintenance activities for construction machinery are usually based on the service hours of the machinery. In Table 21, maintenance intervals recommended by one of the manufacturers of heavy construction equipment is listed for the machinery that are required for the simplified earthmoving process (Caterpillar, 2010c); (Caterpillar, 2010b) (Caterpillar, 2010a). These intervals can be considered as $M L_{\max }^{k}$ according to the presented model. Different tasks are included in each maintenance activity. For example, the tasks included in the 50-hour maintenance activity of excavators shown in the table are lubrication of boom, stick and bucket linkage, drive shaft universal joint, etc.

In a project with four locations, in which earth moving operations need to be done, there are three machines (one excavator, one-wheel loader, and one truck). Because of the significant distance between these locations, a machine needs to work in one location at a time. In Table 22, the operation requirements in each location are shown. Due dates are also shown along with the penalty for each day of delay (GDOT, 2013). Note that the amount of work that a machine

Table 21. Maintenance Intervals (hours) Recommended by the equipment manufacturer (Caterpillar Inc.), reproduced from Yu and Seif (2016).

| Machine | 10 | 50 | 100 | 250 | 500 | 1000 |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Excavators | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ |
| Wheel Loaders | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ |
| (Haul) Trucks | $\checkmark$ | $\checkmark$ |  |  | $\checkmark$ | $\checkmark$ |

can work in one location can be different from other locations due to the condition of the location. As a result, operation requirements in Table 22 are expressed as number of time periods (days) multiplied by the time a machine can work in each time period (in hours) which can be considered as deterioration rates of MLs because the MLs have been expressed in hours.

Table 22. Operation requirements (days) $\times$ deterioration rates (hours) and due dates (days), reproduced from Yu and Seif (2016).

| Location/Work <br> zone (Jobs) | Excavator | Wheel Loader | Tuck | Due Date |
| :---: | :---: | :---: | :---: | :---: |
| A | $20 \times 5$ | $20 \times 3$ | $40 \times 3$ | 90 |
| B | $14 \times 8$ | $14 \times 6$ | $13 \times 8$ | 60 |
| C | $20 \times 4$ | $21 \times 5$ | $20 \times 5$ | 90 |
| D | $30 \times 3$ | $40 \times 2$ | $30 \times 5$ | 60 |

Average cost of performing a preventive maintenance activity and a responsive maintenance activity (after a major failure) for a wheel loader is approximately $\$ 234$ and $\$ 15,652$, respectively (Azadeh et al., 2014). We have used these values to approximate the overall cost of each maintenance activity for each machine, while also considering the risk of major failure due to missing an MA and relative price of the machines. Because the first three MAs (10, 50, 100 hours) are usually done in a fraction of an operational day and usually by the operators, where the machine is located, and because 2000 hours MAs and above are not going to be reached they are not considered as MAs. Deterioration rate for ML 100000 will be zero for the truck because it does not have the respective MA. We will also consider one day for performing all the maintenance activities which is usually the case.

This case study was solved by the IBM ILOG CPLEX. The initial solution of $z_{1}$ and $z_{2}$ are 125 and 4136 respectively. The piecewise linear membership function and optimum solution for the case study are shown in Table 23 and Table 24, respectively. It should be noted that there is no relationship between the two objectives and they are conflicting. Therefore, a solution that decreases tardiness $\left(z_{1}\right)$ at the expense of slightly increasing the maintenance cost $\left(z_{2}\right)$ seems intuitive, yet such a solution might not exist. There might be a solution that decreases the tardiness yet increases the maintenance cost so much (higher than 5000) that the total satisfaction becomes lower than the optimal satisfaction degree (0.675). In fact, there is no solution in this case that decreases the tardiness without lowering the total satisfaction degree. If desired, changing the values for $z_{2}$ and $f_{2}\left(z_{2}\right)$ in Table 23 such that the range of the thresholds for the maintenance cost is higher can yield a better value for tardiness. Therefore, it is up to the DM to update the values of the table based on the maintenance budget and the criticality of the deadlines for each location.

Table 23. Piecewise linear membership functions for the case study.

| Parameters |  | Values |  |  |  |  |
| :---: | :--- | :--- | :--- | :--- | :--- | :--- |
| $z_{1}$ | $<125$ | 125 | 133 | 140 | 145 | $>145$ |
| $f_{1}\left(z_{1}\right)$ | 1 | 1 | 0.8 | 0.5 | 0 | 0 |
| $z_{2}$ | $<4100$ | 4100 | 4500 | 4800 | 5000 | $>5000$ |
| $f_{2}\left(z_{2}\right)$ | 1 | 1 | 0.8 | 0.5 | 0 | 0 |

Table 24. Optimal solution for the case study.

| Variable | Value in the Optimal Solution |
| :--- | :--- |
| $1^{\text {st }}$ location to process | B |
| $2^{\text {nd }}$ location to process | C |
| $3^{\text {rd }}$ location to process | A |
| $4^{\text {th }}$ location to process | D |
| Tardiness $\left(z_{1}\right)$ | 138 |
| Maintenance cost $\left(z_{2}\right)$ | 4136 |
| Satisfaction degree $(L)$ | 0.675 |

### 2.6 Conclusion and Future Research

In this paper we introduced a new extension for the classic formulation of flow shop scheduling by incorporating the machines' requirements for age-based and diverse maintenance activities. The model is general enough to cover a wide range of applications with any number of machines, jobs, or maintenance activities. In modeling the problem, we considered the effect of maintenance and health of the machine on the processing times of the production jobs. This led to a conflict between the two objectives of the problem, namely minimizing total maintenance cost and minimizing total tardiness of the production. We used iFMOLP to capture the fuzzy aspiration level of the decision maker and simultaneously optimize the two objectives. A solution methodology was developed based on the unique structure of the solution space. The results showed that because the solution space can become extremely large for realistic instances of the problem, a metaheuristic algorithm such as a standard genetic algorithm that randomly produces solutions can get stuck in infeasibility.

Commercial solvers also can be very inefficient from a computational time standpoint for larger sizes of the problem where many machines, jobs or maintenance activities are involved. The proposed solution methodology, however, demonstrated satisfactory performance. As was shown in the case study, some real world problems are small enough to be solved by commercial solvers such as CPLEX. Because the presented problem deals with decisions at an operational level, the solution time is very important. The use of the presented solution methodology is recommended only when the commercial solvers are too expensive to obtain or their solution time is not satisfactory.

Although it was assumed in the model that sufficient resources (workforce, spare parts, etc.) are available for the maintenance activities, the two parameters used in the second objective function capture the cost of these resources and the users can adjust these parameters to incorporate the resource limitations into the model. The presented model does not consider random failures. Therefore, some applications in which random failures are common call for a stochastic extension of this model, where the unplanned failures are incorporated into the model. However, random failures can be taken into account by adjusting the duration of maintenance activities in the current model so they cover the average time of unplanned maintenance activities. Also, unplanned failures that disrupt the implementation of a solution obtained from the model can be dealt with just like any other disruptive and unplanned event. In this case the users can reset the parameters and solve the problem again after a disruption. Another opportunity for future research is modeling the same problem under different production settings such as parallel machine scheduling, instead of flow shop scheduling.

## CHAPTER III <br> COMBINIED MAINTENANCE ACTIVITIES IN INTEGRATED FLOW SHOP AND MAINTENANCE SCHEDULING UNDER UNCERTAINITY

A version of this chapter is submitted to the European Journal of Operational Research by Javad Seif, Mohammad Dehghanimohammadabadi, and Andrew J . Yu , and is currently under the second round of review.

In this chapter, I extended my original works that were presented in Chapters I and II. Dr. Dehghani has had contributions in problem definition and simulation-optimization, and Dr. Yu has supported this research. In this chapter, a fixed deterioration rate of 1 is considered and maintenance levels are reworded as residual operating times.


#### Abstract

This article is concerned with incorporating the concept of combined maintenance activities in modeling and optimization of a stochastic permutation flow shop scheduling problem. The objective is to minimize the total expected cost of performing maintenance activities (MAs), and lateness penalties. The processing times of production jobs, as well as the duration of MAs are uncertain and follow certain probability distributions. We formulate the problem as a twostage stochastic mixed-integer program and develop a simulation-optimization solution approach for large-scale instances of the problem. We present extensive computational experiments for performance measurement of the solution and managerial implications. In addition, we demonstrated the application of the problem through a case study in the construction industry.


### 3.1 Introduction

In the conventional scheduling problems, it is assumed that the machines can continuously process the jobs (M. Pinedo, 2012) and the information is complete and certain. However, in practice the machines must stop for preventive or corrective maintenance, and the information available to the planners can be both incomplete and uncertain in scheduling environments (Berry, 1993). The integration of maintenance and scheduling has appeared in the literature in the last two decades (Xu et al., 2015; Yu \& Seif, 2016). The goal of this integration is to mimic the manufacturing or service environments as closely as possible. The more the technical nuances of the maintenance management are considered, the higher the practicality of these models and solutions is going to be; however, incorporating maintenance decisions into the production scheduling problems, requires more sophisticated modeling approaches. This could also make the computational effort larger, especially for the large-scale problems. The issue becomes even more complex when uncertainty is taken into account. This paper provides a stochastic mixed integer program to properly include maintenance decisions into the production scheduling when uncertainty exists. Using two
solution approaches, namely stochastic programming and SimulationOptimization (SO), this paper presents practical insights to dealing with computational limitations.

Flow shop scheduling has been studied by many researchers after Johnson introduced the problem for two machines in 1954 (Johnson, 1954). The main goal in flow shop scheduling is to find a sequence for $n$ jobs that are to be processed by $m$ machines to optimize an objective function. Minimizing the completion time of the very last job (the makespan), the overall completion time, the tardiness of the jobs are some examples of such an objective. Maintenance costs cover a big percentage of the total operating costs (Ángel-Bello et al., 2011; Yip et al., 2014). Therefore, it is reasonable to include minimizing the maintenance cost in the objective function. In this paper, we integrate tardiness cost of the jobs and maintenance cost to define the objective function.

Yoo and Lee (2016) classify scheduling problems with MAs incorporated, as fixed and coordinated. The first class of problems, scheduling with machine availability constraints, considers maintenance as a constraint not a decision. For instance, Choi, Lee, Leung, and Pinedo (2010) consider a number of maintenance periods in ordered and proportionate flow shop. They assume that these maintenance periods have been scheduled in advance with known start and finish times. Therefore, the maintenance schedule is incorporated to their model as a constraint not a decision. Other researchers use the same approach to model machine unavailability in a two-machine flow shop scheduling (T. C. Edwin Cheng \& Wang, 2000; T. E. Cheng \& Wang, 1999; Kubiak, Błażewicz, Formanowicz, Breit, \& Schmidt, 2002; Kubzin, Potts, \& Strusevich, 2009; C.-Y. Lee, 1997, 1999).

In the second class of problems, scheduling of maintenance and job processing are considered simultaneously. Aggoune (2004) is one of the few papers that studies the coordinated variant of flow shop scheduling, while allowing a decision for performing maintenance within a time window. Performing maintenance in a time window has been modeled in different types of scheduling problems, yet they are not precise in timing of MAs. Stefan Bock, Dirk Briskorn, and Andrei Horbach (2012) study the computational complexity of single machine scheduling problems where each machine has a maintenance level and processing of the jobs deteriorates it. Maintenance needs to be performed in order to restore or increase the maintenance level before the level becomes negative. Seif et al. (2017) and Yu and Seif (2016) adapt the concept of maintenance levels in flow shop scheduling when multiple types of maintenance levels are involved (Chapters I and II). In this paper, we formulate and solve a permutation flow shop scheduling problem under uncertainty, and incorporate the concept of combining different types of MAs in the problem.

Knezevic (1997) classifies maintenance tasks as simultaneous, sequential, and combined. A simultaneous task is composed of activities that are mutually independent, yet can be performed concurrently. A sequential task includes mutually independent activities that are performed in a predetermined order. A combined task includes some activities that can be performed simultaneously, and some activities that are sequential. Therefore, a combined maintenance task is a generalization of the other two types. To prevent any confusion, by MA, we refer to a set of tasks that have a common usage-based periodic interval. MAs may have some similar tasks, so the combination of two or more MAs could prevent the repetition of the common tasks. We model the concept of combined maintenance activities in scheduling problems for the first time. In doing so, we cover all of the possible scenarios. The case study in Section 3.5 shows a practical example for better understanding of the problem.

Over the last few years, the importance of considering uncertainty in the scheduling problems has been highlighted by researchers and industrial practitioners; however, the methods used to deal with uncertainty do not seem to be very effective (Zheng, Lian, Fu, \& Mesghouni, 2015). Gourgand, Grangeon, and Norre (2000) and González-Neira, Montoya-Torres, and Barrera (2017) conduct comprehensive reviews of research papers that involve flow shop scheduling under uncertainty. The latter review found that most of the papers in the literature consider processing times as a stochastic parameter, but maintenance has never been included as a stochastic process; while Mean Time to Repair (MTTR) is a well-known term in the maintenance and reliability literature (Ben-Daya, Ait-Kadi, Duffuaa, Knezevic, \& Raouf, 2009; Hastings, 2009) and is based on the assumption of the uncertain durations of MAs.

González-Neira et al. (2017) report the superiority of the stochastic optimization approach in modeling the uncertainty. They also mention stochastic programming and simulation-optimization as the most promising methods in stochastic flow shop scheduling. In this paper for the first time we consider two stochastic parameters to capture the uncertainties: (i) the processing times of production jobs, and (ii) the durations of MAs. To cross-validate the performance of solution approaches, we apply both stochastic programming and simulationoptimization to model and solve the problem. The use of both methods facilitates their performance evaluation in computational experiments.

In this paper, we deal with the uncertainties of the problem via stochastic programming and simulation optimization, and we use Monte-Carlo simulation for scenario generation. This simulation allows us to generate a number of scenarios that sufficiently represent all the possibilities. The problem is then handled as deterministic via these two solution approaches. The contributions of this paper are:

- the concept of combined MAs is introduced and formulated,
- the conventional permutation flow shop scheduling problems are extended by incorporating maintenance decisions,
- uncertainty is considered for both processing times and the durations of MAs,
- two approaches, namely stochastic programming and simulationoptimization, are employed for modeling, cross-validation, and solving the problem,
- extensive computational experiments are conducted to evaluate both approaches, and drive conclusions for industrial practitioners, and
- the application of this research is demonstrated through a case study in earthmoving operations.

The rest of this chapter is laid out as follows. First, we formulate the problem as a two-stage Stochastic Mixed-Integer Program (SMIP) in Section 3.2. In Section 3.3, we present a SO algorithm as an alternative approach for modeling and solving the problem, which allows the validation of the SMIP. In Section 3.4, we will evaluate and report in details the performance of the SO in comparison with one of the commercial solvers through extensive computational experiments. In Section 3.5, the case study is presented. Conclusions and remarks along with directions for future research are discussed in Section 3.6.

### 3.2 Problem Definition and Mathematical Formulation

In this section, we define mixed-integer stochastic program to define the problem. To do this, first, we define and formulate the concepts of combining maintenance activities, and discuss the prolonged processing times. The SMIP model is presented at the end of this section.

## Combined Maintenance Activities

When two or more maintenance activities are scheduled consecutively (in a row), it is likely that the total duration of these combined activities becomes less than the sum of the durations of the individual activities when they are performed separately. This is due to the fact that, the activities can share one or more tasks. Figure 8 illustrates an example in which Activities 1 and 2 share Tasks A and C. When these two activities are combined, each of the shared tasks is performed only once, which shortens the total duration of performing maintenance.

The individual maintenance activities are considered independently in the scheduling process. The MAs are combined only when they are scheduled back-


Figure 8. Maintainability block diagram for combined maintenance activities.
to-back. For $l$ maintenance activities, there exists $o=\sum_{k=1}^{l}\binom{l}{k}=2^{l}$ combinations of maintenance activities. The binary variables $y_{k}, k=1, \ldots, l$ determines whether a maintenance activity is scheduled at a certain position on the timeline between production jobs. The binary variable $\phi_{r}, r=1, \ldots, o$ determines whether combination $r$ is forming in that position, and is a function of $y_{k}, k=1, \ldots, l$. Table 25 lists all of the possible combinations for three types of maintenance activities $(l=3)$. The parameter $e_{k}$ denotes the duration of maintenance type $k$, while $e_{r}^{\prime}$ denotes the duration of maintenance combination $r$.

Table 25. All possible combinations for three types of maintenance activities.

| Combination | $y_{1}$ | $y_{2}$ | $y_{3}$ | Number of MAs | Nominal Duration | Actual Duration |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 1 | 0 | 0 | 0 | 0 | 0 | $e_{1}^{\prime}=0$ |
| 2 | 1 | 0 | 0 | 1 | $e_{1}$ | $e_{2}^{\prime}=e_{1}$ |
| 3 | 0 | 1 | 0 | 1 | $e_{2}$ | $e_{3}^{\prime}=e_{2}$ |
| 4 | 0 | 0 | 1 | 1 | $e_{3}$ | $e_{4}^{\prime}=e_{3}$ |
| 5 | 1 | 1 | 0 | 2 | $e_{1}+e_{2}$ | $e_{5}^{\prime}=0.85\left(e_{1}+e_{2}\right)$ |
| 6 | 1 | 0 | 1 | 2 | $e_{1}+e_{3}$ | $e_{6}^{\prime}=0.85\left(e_{1}+e_{3}\right)$ |
| 7 | 0 | 1 | 1 | 2 | $e_{2}+e_{3}$ | $e_{7}^{\prime}=0.85\left(e_{2}+e_{3}\right)$ |
| 8 | 1 | 1 | 1 | 3 | $e_{1}+e_{2}+e_{3}$ | $e_{8}^{\prime}=0.75\left(e_{1}+e_{2}+e_{3}\right)$ |

In this example we assumed when two or more maintenance activities are combined, the total duration of combined MAs is less than the sum of the individual durations. For instance, as provided in Table 25, in combination type 8 where three MAs are combined ( $y_{1}=y_{2}=y_{3}=1$ ), the total duration of performing all three MAs is $25 \%$ less than the sum of the individual durations. In practice, the duration of these combinations can be different based on the application and the actual conditions. One might break down the MAs into the tasks that comprise them, and then take into account the common tasks in a combination only once, similar to the example in Table 25. Regardless of the method for calculating the duration of maintenance activities in a combined form, new durations $\left(e_{r}^{\prime}, r=1, \ldots, 2^{l}\right)$ will be used as input parameters in the SMIP.

Equation (3.1) maps the decisions for performing the maintenance activities (on the right-hand side) to the decision for choosing/performing one of the combinations. The values of the coefficients $a_{k}, k=1, \ldots, l$ and $b_{r}, r=$ $1, \ldots, 2^{l}-1$ must be chosen such that only one of the combinations in the lefthand side gets chosen. This means that none, one, or more than one of the variables on the right-hand side can take the value 1, but at most only one of the variables on the left-hand side must take the value 1.

$$
\begin{equation*}
\sum_{r=1}^{o=2^{l}-1} b_{r} \phi_{r}=\sum_{k=1}^{l} a_{k} y_{k}, \quad \phi_{r}, y_{k} \in\{0,1\}, \forall k=1, \ldots, l \tag{3.1}
\end{equation*}
$$

Next, we introduce a method for choosing the coefficients such that, at most only one combinations is selected. In the example provided in Table 1, for instance, the selection of only Maintenance Activities 1 and 2 is equal to the selection of only Combination 5 . Note that the number of combinations is $2^{l}-1$; the first combination that corresponds to the case where none of the maintenance activities is performed has been removed from the set of combinations because that combination is realized when none of the other combinations is selected. Next we introduce a method for choosing the coefficients in (1) such that at most only one combination, the right one, is selected.

Consider the set $\boldsymbol{a}=\left\{a_{1}, a_{2}, \ldots, a_{l}\right\}, \forall k=1, \ldots, l, a_{k} \in \mathbb{R} . \boldsymbol{S}=$ $\left\{\boldsymbol{s}_{1}, \boldsymbol{s}_{2}, \ldots, \boldsymbol{s}_{\left(2^{\boldsymbol{l}}-\mathbf{1}\right)}\right\}$ is the set of all the possible subsets of $\boldsymbol{a}$ that are not null, and $\boldsymbol{b}=\left\{b_{1}, b_{2}, \ldots, b_{\left(2^{l}-1\right)}\right\} \ni b_{r}=\sum_{a^{\prime} \in s_{r}} a^{\prime}, \forall r=1, \ldots, 2^{l}-1$. We want to find the elements of $\boldsymbol{a}$ such that the elements of $\boldsymbol{b}$ are unique. For example, for $\boldsymbol{a}_{\boldsymbol{1}}=$ $\{1,2,3\}, l=3, \boldsymbol{S}_{\mathbf{1}}=\{\{1\},\{2\},\{3\},\{1,2\},\{1,3\},\{2,3\},\{1,2,3\}\} \therefore \boldsymbol{b}_{\mathbf{1}}=\{1,2,3,3,4,5,6\}$, where the members of $\boldsymbol{b}_{\mathbf{1}}$ are not unique. But for $\boldsymbol{a}_{2}=\{1.1,1.2,1.3\}, l=3, \boldsymbol{S}_{2}=$ $\{\{1.1\},\{1.2\},\{1.3\},\{1.1,1.2\},\{1.1,1.3\},\{1.2,1.3\},\{1.1,1.2,1.3\}\} \quad \therefore \boldsymbol{b}_{2}=$ $\{1.1,1.2,1.3,2.3,2.4,2.5,3.6\}$, where the members of $\boldsymbol{b}_{2}$ are unique.

Proposition. If $\boldsymbol{a}=\{1.1,1.01, \ldots, 1 . \underbrace{00 \ldots 0}_{l-1} 1\},|\boldsymbol{a}|=l$, the uniqueness of the elements in $\boldsymbol{b}$ is guaranteed.
Proof. For $\boldsymbol{a}=\{1.1,1.01, \ldots, 1 \underbrace{00 \ldots 0}_{l-1} 1\},|\boldsymbol{a}|=l$ :

$$
\begin{aligned}
& \boldsymbol{S}=\{\{1.1\},\{1.01\}, \ldots,\{1 . \underbrace{00 \ldots 0}_{l-1} 1\},\{1.1,1.2\}, \ldots,\{1.1,1.01, \ldots, 1 . \underbrace{00 \ldots 0}_{l-1} 1\}\},|\boldsymbol{S}| \\
& =2^{l}-1 \therefore \boldsymbol{b}=\{1.1,1.01, \ldots, 1 . \underbrace{00 \ldots 0}_{l-1} 1,2.11,2.101, \ldots, l . \underbrace{11 \ldots 1}_{l}\},|\boldsymbol{b}| \\
& =2^{l}-1 .
\end{aligned}
$$

Assume $\exists b_{i}, b_{j} \in \boldsymbol{b}$ and $\boldsymbol{s}_{\boldsymbol{i}}, \boldsymbol{s}_{\boldsymbol{j}} \in \boldsymbol{S} \ni b_{i}=b_{j}$ and $\boldsymbol{s}_{\boldsymbol{i}} \neq \boldsymbol{s}_{\boldsymbol{j}}$. Because $b_{i}=b_{j}$, the integer and decimal parts of the two numbers are equal. Because the integer parts of all of the elements in $\boldsymbol{a}$ are the same (1), the integer part of $b_{i}$ and $b_{j}$ indicate the number of elements in their respective subsets, i.e. $\boldsymbol{s}_{\boldsymbol{i}}$ and $\boldsymbol{s}_{\boldsymbol{j}}$. Because all elements in each subset have 1 as their integer part and each element has a unique number of zeros in the decimal part, the elements have to be equal for $b_{i}$ and $b_{j}$ to be equal. If the elements of $\boldsymbol{s}_{\boldsymbol{i}}$ and $\boldsymbol{s}_{\boldsymbol{j}}$ are identical, $\boldsymbol{s}_{\boldsymbol{i}}=$
$\boldsymbol{s}_{\boldsymbol{j}}$ which contradicts the assumption. Therefore, for $\boldsymbol{a}=$ $\{1.1,1.01, \ldots, 1 \underbrace{00 \ldots}_{l-1} 1\},|\boldsymbol{a}|=l$ the elements of $\boldsymbol{b}$ are unique.

## Prolonged Processing Times

An age-based maintenance activity allows a machine to operate only for certain number of hours, called maintenance interval. As soon as the machine's cumulative operating times equals the maintenance interval, it must be stopped for performing the maintenance activity. Let $r_{k}$ and $R_{k}$ be the residual operating time and the age-based interval of the $k$-th maintenance activity of a machine, respectively. The variable $r_{k}$ equals $R_{k}$ after the maintenance activity type $k$ is performed on the machine and it approaches 0 as the machine processes production jobs. We expect that in practice the processing time of a job prolongs as the residual operating times approach 0 . This is because: 1) the performance of the machine might degrade as the machine gets closer to its maintenance requirements, which can lead to a slower processing, and 2) when both tardiness and maintenance costs exist in a minimization objective function, it motivates the solution algorithms to schedule maintenance activities as early as possible to reduce the risk of failures, but not too early that causes excessive tardiness and maintenance costs. Next, we try to adapt the model proposed in [5] for prolonged processing times.

The value $f_{k}=r_{k} / R_{k}$ represents the remaining/residual operating time as a fraction of the respective maintenance interval. Obviously, $0 \leq f_{k} \leq 1, \forall k=$ $1, \ldots, l$, and $0 \leq F \leq 1$ where $F=\sum_{k=1}^{l} f_{k} / l$ is the average of fractional residual operating times and represents the machine's health. The prolonged processing time of a job, $\rho$, is defined as

$$
\rho= \begin{cases}\lambda^{1} p, & A<\mathrm{F} \leq 1  \tag{3.2}\\ \lambda^{2} p, & B<\mathrm{F} \leq A, \quad 0 \leq B \leq A \leq 1 \leq \lambda^{1} \leq \lambda^{2} \leq \lambda^{3} \\ \lambda^{3} p, & 0<\mathrm{F} \leq B\end{cases}
$$

where $p$ is the nominal processing time of the job. If the machine's health, $F$, falls between the constant $A$ and 1 , the nominal processing time is multiplied by the coefficient $\lambda^{1}$ which can be greater than or equal to 1 with the potential to prolong the processing time. If the machine's health is between $A$ and $B$, processing time is multiplied by the coefficient $\lambda^{2}$ which can be greater than $\lambda^{1}$, and prolong the processing time, and if it is between 0 and $B$, multiplied by $\lambda^{3}$ which can be greater than $\lambda^{2}$ prolonging the processing time even more. Here we are considering a special case in which the health of a machine has only three states. The generalized form will be considered in the SMIP formulation.

## The Stochastic Mixed-Integer Program (SMIP)

We model a permutation flow shop scheduling problem in which a number of jobs will be processed by a series of machines in the same order. All machines have a number of age-based maintenance activities. The MAs take place only after a machine completes processing of a job and before starts processing the next job (preemption is not allowed). The residual operating time of a machine with respect to any of the MAs linearly decreases as the machine processes the jobs. The residual operating times cannot be negative. Therefore, an MA must be performed before processing a job, if the respective residual operating time is going to become negative while processing the job. The durations of MAs and the processing times of jobs are all uncertain and follow certain probability distributions. There are several scenarios in each of which the MA durations and processing times are sampled from respective distributions. Under each scenario, the maintenance durations can change if the MAs are combined.

## Notations

Let $m, n, I$, and $S$ be the number of machines, jobs, maintenance activities, and scenarios, respectively. The following indices, parameters, and variables are used in the formulation of the problem.

| Indices |  |
| :---: | :---: |
|  | Represents machines where $i=1, \ldots, m$. |
| j | Represents jobs where $j=1, \ldots, n$. |
| q | Represents job positions in a sequence where $q=1, \ldots, n$. |
| $k$ | Represents maintenance activities where $k=1, \ldots, l$. |
| $h$ | Represents the health state of a machine $h=1, \ldots, H$. |
| $r$ | Represents maintenance combinations where $r=1,2, \ldots, 0$. |
| $s$ | Represents a specific scenario where $s=1, \ldots, S$. |
| Parameters (Input Data) |  |
| $p_{i j}^{s}$ | Nominal processing time of job $j$ on machine $i$ under scenario s. |
| $e_{i k}^{s}$ | Nominal duration of MA type $k$ on machine $i$ under scenario s. |
| $e_{i r}^{\prime \prime}$ | Duration of MA combination type $r$ on machine $i$ under scenario $s$. |
| $R_{i, k}$ | Time interval for maintenance activity type $k$ for machine $i$. |
| $S P_{i k}$ | Cost of required spare parts and materials for MA type $k$ on machine $i$. |
| $S P_{i k}^{\prime}$ | Cost of required spare parts and materials for MA combination type $r$ on machine $i$. |
| WF | Cost of skilled workforce per time unit for performing maintenance activities. |
| $d_{j}$ | The due date of job $j$. |
| $\pi_{j}$ | Penalty cost associated with each time unit delay in completion of job $j$. |
| $\lambda^{h}$ | The coefficient which is multiplied by the nominal processing times of the jobs to prolong them, when the machine is in the health state $h$. |

$\operatorname{Pr}(s) \quad$ Probability of scenario $s$ being realized.
$K \quad$ A sufficiently large number.
Decision Variables
$Z_{s} \quad$ Total cost, the value of the objective function under scenario $s$.
$x_{j q} \quad$ First-stage decision variable that takes the value 1 if job $j$ is processed as the $q$-th job, and 0 otherwise.
$y_{i q k} \quad$ First-stage binary decision variable that takes the value 1 if MA type $k$ is performed on machine $i$ before processing the $q$-th job, and 0 otherwise.
$\phi_{i q r} \quad$ First-stage binary decision variable that takes the value 1 if MA combination type $r$ is performed on machine $i$ before processing the $q$-th job, and 0 otherwise.
Residual operating time with respect to the MA type $k$ of machine $i$ before processing the $q$-th job, under scenario $s$.
$S T_{i q}^{s} \quad$ Start time of the $q$-th job on machine $i$, under scenario $s$.
$F T_{i q}^{s} \quad$ Finish time of the $q$-th job on machine $i$, under scenario $s$.
$t_{q}^{s} \quad$ Tardiness of the $q$-th job, under scenario $s$.
$\rho_{i q}^{s} \quad$ Processing time of the $q$-th job on machine $i$, under scenario $s$.
$\gamma_{i j q}^{s} \quad$ Processing time of job $j$ on machine $i$ when it is processed as the $q$-th job, under scenario s.
$\Pi_{j q}^{s} \quad$ Penalty cost associated with job $j$ if it is processed as the $q$-th job, under scenario s
$\Lambda_{i q}^{h s} \quad 1$ if machine $i$ is in the health state $h$ before processing the $q$-th job, under scenario $s$, and 0 otherwise.
$u_{i j q}^{h s} \quad 1$ if machine $i$ is in the health state $h$ before processing job $j$ when it is processed as the $q$-th job, under scenario $s$, and 0 otherwise.

## The Model

The objective function (OF) of the model is to minimize the total expected cost, which comprises the penalty cost incurred because of lateness in completion of the jobs (tardiness), and the maintenance cost (spare parts and required workforces).

$$
\operatorname{minimize} E\left[Z_{s}\right]
$$

$$
\begin{equation*}
=\sum_{s=1}^{s} \operatorname{Pr}(s)\left[\sum_{j=1}^{n} \sum_{q=1}^{n} \Pi_{j q}^{s}+\sum_{i=1}^{m} \sum_{q=1}^{n} \sum_{r=1}^{o=2^{l}-1} \phi_{i q r}\left(S P_{i r}^{\prime}+e_{i r}^{s} W F\right)\right] \tag{3.3}
\end{equation*}
$$

Subject to:

$$
\begin{array}{ll}
\sum_{q=1}^{n} x_{j q}=1, & j=1, \ldots, n \\
\sum_{j=1}^{n} x_{j q}=1, & q=1, \ldots, n \\
S T_{11}^{s}=0, & s=1, \ldots, S \tag{3.6}
\end{array}
$$

$$
\begin{gather*}
s=1, \ldots, S, i=1, \ldots, m, q=1, \ldots, n, k  \tag{3.13}\\
=1, \ldots, l
\end{gather*}
$$

$$
\begin{equation*}
S T_{i 1}^{S}=\sum_{i^{\prime}=1}^{i-1} \rho_{i^{\prime} 1}^{S} \tag{3.7}
\end{equation*}
$$

$$
i=2, \ldots, m, s=1, \ldots, S
$$

$$
S T_{1 q}^{S}=F T_{1(q-1)}^{s}
$$

$$
\begin{equation*}
+\sum_{r=1}^{o} \phi_{1 q r} e_{1 r}^{\prime s} \tag{3.8}
\end{equation*}
$$

$$
q=2, \ldots, n, s=1, \ldots, S
$$

$$
S T_{i q}^{S} \geq F T_{i(q-1)}^{S}
$$

$$
\begin{equation*}
+\sum_{r=1}^{o} \phi_{i q r} e_{1 r}^{s s} \tag{3.9}
\end{equation*}
$$

$$
i=2, \ldots, m, q=2, \ldots, n, s=1, \ldots, S
$$

$$
\begin{equation*}
S T_{i q}^{S} \geq F T_{(i-1) q}^{S} \tag{3.10}
\end{equation*}
$$

$$
i=2, \ldots, m, q=2, \ldots, n, s=1, \ldots, S
$$

$$
\begin{equation*}
F T_{i q}^{S}=S T_{i q}^{s}+\rho_{i q}^{s}, \tag{3.11}
\end{equation*}
$$

$$
i=1, \ldots, m, q=1, \ldots, n, s=1, \ldots, S
$$

$$
\begin{equation*}
r_{i 1}^{k s}=R_{i, k} \tag{3.12}
\end{equation*}
$$

$$
i=1, \ldots, m, s=1, \ldots, S, k=1, \ldots, l
$$

$$
r_{i q}^{k s} \geq \sum_{j=1}^{n} \gamma_{i j q}^{s}
$$

$$
\begin{equation*}
r_{i q}^{k s} \geq r_{i(q-1)}^{k s}-\sum_{j=1}^{n} \gamma_{i j(q-1)}^{s} \tag{3.14}
\end{equation*}
$$

$$
\begin{gathered}
s=1, \ldots, S, i=1, \ldots, m, q=2, \ldots, n, k \\
=1, \ldots, l
\end{gathered}
$$

$$
-y_{i q k} K
$$

$$
\begin{equation*}
r_{i q}^{k s} \leq r_{i(q-1)}^{k s}-\sum_{j=1}^{n} \gamma_{i j(q-1)}^{s} \tag{3.15}
\end{equation*}
$$

$$
\begin{gathered}
s=1, \ldots, S, i=1, \ldots, m, q=2, \ldots, n, k \\
=1, \ldots, l
\end{gathered}
$$

$$
+y_{i q k} K
$$

$$
\begin{equation*}
r_{i q}^{k s} \geq R_{i, k}-K\left(1-y_{i q k}\right) \tag{3.16}
\end{equation*}
$$

$$
\begin{align*}
& s=1, \ldots, S, i=1, \ldots, m, q=2, \ldots, n, k \\
& \quad=1, \ldots, l \\
& s=1, \ldots, S, i=1, \ldots, m, q=2, \ldots, n, k  \tag{3.17}\\
& \quad=1, \ldots, l
\end{align*}
$$

$$
r_{i q}^{k s} \leq R_{i, k}+K\left(1-y_{i q k}\right)
$$

$$
\begin{equation*}
t_{q}^{s} \geq F T_{m q}^{s}-\sum_{j=1}^{n} x_{j q} d_{j} \tag{3.18}
\end{equation*}
$$

$$
\begin{align*}
& \Pi_{j q}^{S}-\pi_{j} t_{q}^{s} \geq-K(1- \\
& x_{j q} \text { ), } \\
& \Pi_{j q}^{\mathrm{s}}-\pi_{j} t_{q}^{\mathrm{s}} \leq K(1 \\
& \left.-x_{j q}\right), \\
& \Pi_{j q}^{\mathrm{S}} \geq-K x_{j q}, \quad s=1, \ldots, S, j=1, \ldots, n, q=1, \ldots, n \\
& \Pi_{j q}^{\mathrm{S}} \leq K x_{j q}, \quad s=1, \ldots, S, j=1, \ldots, n, q=1, \ldots, n \\
& \rho_{i q}^{S}=\sum_{j=1}^{n} \sum_{h=1}^{H} u_{i j q}^{h s} \lambda^{h} p_{i j}, \quad s=1, \ldots, S, i=1, \ldots, m, q=1, \ldots, n \\
& u_{i j q}^{h s} \leq \Lambda_{i q}^{h s}, \\
& s=1, \ldots, S, j=1, \ldots, n, q=1, \ldots, n \\
& s=1, \ldots, S, j=1, \ldots, n, q=1, \ldots, n \\
& \begin{array}{r}
s=1, \ldots, S, i=1, \ldots, m, j=1, \ldots, n, q=1, \ldots, n, \\
h=1, \ldots, H
\end{array} \\
& s=1, \ldots, S, i=1, \ldots, m, j=1, \ldots, n, q=1, \ldots, n \text {, } \\
& h=1, \ldots, H \\
& u_{i j q}^{h s} \leq x_{j q}, \\
& \begin{array}{r}
s=1, \ldots, S, i=1, \ldots, m, j=1, \ldots, n, q=1, \ldots, n, \\
h=1, \ldots, H
\end{array} \\
& u_{i j q}^{h s} \geq x_{j q}+\Lambda_{i q}^{h s}-1, \\
& s=1, \ldots, S, i=1, \ldots, m, q=1, \ldots, n \\
& \sum_{h=1}^{H} \Lambda_{i q}^{h s}=1, \\
& \sum_{k=1}^{l} \frac{r_{i q}^{k s}}{l \cdot R_{i, k}}>\frac{H-1}{l} \Lambda_{i q}^{1 s}-K \sum_{h=2}^{H} \Lambda_{i q}^{h s} \quad s=1, \ldots, S, i=1, \ldots, m, q \\
& \sum_{k=1}^{l} \frac{r_{i q}^{k s}}{l \cdot R_{i, k}} \leq \frac{h}{l} \Lambda_{i q}^{h s}+K\left(\sum_{h^{\prime}=1}^{H} \Lambda_{i q}^{h^{\prime} s}-\Lambda_{i q}^{h s}\right), \quad h=2, \ldots, H-1, s=1, \ldots, S, \\
& \sum_{k=1}^{l} \frac{r_{i q}^{k s}}{l \cdot R_{i, k}} \geq \frac{h-1}{l} \Lambda_{i q}^{h s} \\
& -K\left(\sum_{h^{\prime}=1}^{H} \Lambda_{i q}^{h^{\prime} s}-\Lambda_{i q}^{h s}\right), \\
& \sum_{k=1}^{l} \frac{r_{i q}^{k s}}{l \cdot R_{i, k}}<\frac{1}{l} \Lambda_{i q}^{H s}+K \sum_{h=1}^{H-1} \Lambda_{i q}^{h s},  \tag{3.31}\\
& \begin{array}{c}
s=1, \ldots, S, i=1, \ldots, m, q \\
=1, \ldots, n
\end{array} \\
& h=2, \ldots, H-1, s=1, \ldots, S \text {, }  \tag{3.30}\\
& i=1, \ldots, m, q=1, \ldots, n
\end{align*}
$$

$$
\begin{align*}
& \sum_{r=1}^{o} b_{r} \phi_{i q r}=\sum_{k=1}^{l} a_{k} y_{i q k},  \tag{3.35}\\
& i=1, \ldots, m, q=1, \ldots, n \\
& s=1, \ldots, S, j=1, \ldots, n, q=1, \ldots, n, i \\
& x_{j q}, y_{i q k}, \Lambda_{i q}^{h}, u_{i j q}^{h s}, \phi_{i q r} \in\{0,1\},  \tag{3.36}\\
& =1, \ldots, m \text {, } \\
& h=1, \ldots, H \\
& s=1, \ldots, S, j=1, \ldots, n, q=1, \ldots, n, i \\
& =1, \ldots, m \text {, }  \tag{3.37}\\
& h=1, \ldots, H
\end{align*}
$$

The OF in Equation (3.3) is comprised of two parts; the penalty cost, and the maintenance cost. The penalty cost for each job is presented as a variable that has two indices, $j$ and $q$. For each job $j$, the variable is set equal to 0 in Constraints (3.21) and (3.22) if it does not occupy the position $q$ in the sequence of the jobs. Otherwise, it is set equal to the penalty cost for job $j$ times the tardiness value, as expressed in Constraints (3.19) and (3.20). The maintenance cost itself is comprised of two parts; the cost of spare parts and the cost of the cost of workforce. The binary variable $\phi_{i q r}$ determines whether maintenance combination $r$ is scheduled before processing the $q$-th job on machine $i$. It is multiplied by the cost of spare part for that combination plus the unit cost of workforce times the duration of that combination. The MAs are considered individually in the scheduling process. Constraint (3.35), which is a generalization of Equation (3.1), chooses the combination that correctly represents the scheduled MAs.

Constraints (3.5) and (3.6) ensure that each job is assigned to only one position in the sequence of the jobs, and each position is filled by only one job. These two constraint sets are only concerned with the first stage variables. However, the rest of the constraints involve at least one second-stage variable with $s$ in their superscripts. Therefore, the rest of the constraints must be feasible for every scenario, otherwise the solution is infeasible.

Constraints (3.6-3.11) together determine the start and finish time of the jobs on every machine. The Start Time (ST) and Finish Time (FT) variables are first calculated for the first job in the sequence and the first machine in the flow shop, and then they will be calculated for the rest of the jobs and machines. Constraint (3.6) sets 0 as the ST of the first job on the first machine. Constraint (3.7) sets the ST of the first job on each machine equal to the sum of its processing times on the previous machines. Constraint (3.8) sets the ST of each job on the first machine equal to the finish time of the previous job on the first machine, plus the duration of MAs. As was already explained, instead of nominal durations of the individual MAs, the duration of maintenance combinations is
considered in timings. Constraints (3.9) and (3.10) are the linear form of $S T_{i q}^{S}=$ $\max \left(F T_{i(q-1)}^{S}+\sum_{r=1}^{o} \phi_{i q r} e_{1 r}^{s}, F T_{(i-1) q}^{S}\right)$; the ST of every job on a machine is equal to the maximum of its FT on the previous machine plus the maintenance time, and the FT of the previous job on the machine. Constraint (3.11) sets the FT of all jobs on every machine equal to the respective ST plus the processing time of the job.

Constraint (3.12) sets the residual operating times of all machines equal to the maintenance interval, before processing the first job. Constraint (3.13) sets the residual operating time of machine $i$ before processing the $q$-th job to be greater than or equal to the time it takes to process the job. This constraint ensures that the machines do not operate while their maintenance requirements are overdue. Constraints (3.14-3.17) are the linearizes form of the following equation.

$$
r_{i q}^{k s}=\left\{\begin{array}{rl}
r_{i(q-1)}^{k s}-\sum_{j=1}^{n} \gamma_{i j(q-1)}^{s}, & y_{i q k}=0  \tag{3.38}\\
R_{i, k}, & y_{i q k}=1
\end{array} \quad \forall s, i, q, k\right.
$$

The residual operating time of a machine before processing a job with respect to MA $k$ is equal to the respective maintenance interval, if the MA is performed, and otherwise it is equal to its value before processing the previous job minus the processing time of the previous job. Tardiness for each job in the sequence is calculated in Constraint (3.18). The prolonged processing times of the jobs are calculated as expressed in Constraint (3.23); the nominal processing time of job $j$ on machine $i$ times the coefficient of state $h$, times the binary variable $u_{i j q}^{h s}$ that takes the value 1 if machine $i$ is in state $h$ before processing the $q$-th job and if the $q$-th job is job $j$, and 0 oherwise. Constraints (3.24-3.26) are the linearization form of $u_{i j q}^{h s}=\Lambda_{i q}^{h s} x_{j q}$.

Constraint (3.27) ensures that machine $i$ is in only in one of the predefined health states before processing the $q$-th job. Constraints (3.28-3.31) are the generalized and linearized form of the following equations when the machines have only three states $(H=3)$.

$$
\Lambda_{i q}^{1 s}=\left\{\begin{array}{cc}
1, & \sum_{k=1}^{l} \frac{r_{i q}^{k s}}{l \cdot R_{i, k}}>0.66  \tag{3.39}\\
0, & \text { otherwise }
\end{array} \quad \forall s, i, q\right.
$$

$$
\begin{align*}
& \Lambda_{i q}^{2 s}= \begin{cases}1, & 0.33 \leq \sum_{k=1}^{l} \frac{r_{i q}^{k s}}{l \cdot R_{i, k}} \leq 0.66, \\
0, & \text { otherwise }\end{cases}  \tag{3.40}\\
& \Lambda_{i q}^{3 s}= \begin{cases}1, & \sum_{k=1}^{l} \frac{r_{i q}^{k s}}{l \cdot R_{i, k}}<0.33, \\
0, & \text { otherwise }\end{cases} \tag{3.41}
\end{align*}
$$

Constraints (3.32-3.34) are the linearized form of $\gamma_{i j q}^{S}=x_{j q} \rho_{i q}^{S}$ for obtaining the actual processing time of job $j$ on machine $i$, if it is scheduled as the $q$-th job. Constraint (3.35) ensures that the correct maintenance combination is chosen based on the scheduling of MAs. The one that corresponds to the set of maintenance activities that are decided to be performed. This is a generalization of Equation (3.1) for flow shop scheduling with multiple types of MAs. Constraints (3.36-3.37) ensure that all the decision variables are within their bounds.

### 3.3 Simulation-Optimization

Simulation-Optimization (SO) is a promising avenue of research to tackle stochastic problems with uncertain parameters (Dehghanimohammadabadi, 2016; Dehghanimohammadabadi, Keyser, \& Cheraghi, 2017). We applied Simulation-Based Optimization (SBO), in which an optimization module explores the solution space to obtain the best configuration for the stochastic problem created by the simulation module. In this approach, the Monte-Carlo simulation is used to generate a number of possible scenarios based on the probability distributions of the stochastic parameters. The scenarios are inputs to the optimization module. Although each one of these scenarios represents a deterministic instance of the problem, in which the value of the stochastic parameters is certain, they collectively represent the stochastic nature of the problem.

As depicted in Figure 9, after the scenarios are generated via the MonteCarlo simulation, the optimization module generates a solution $(X)$, recursively, and the objective function value is calculated for each scenario $\left(Y_{s}\right)$. The expected value (average) will determine the ultimate value of the solution. The new solution is generated based on the internal operators and search methods of the specific meta-heuristic method that is being used. This cycle repeats until the optimization module satisfies some stopping criteria. Depending on the user's preferences, these criteria could be running model for a certain number of iterations or achieving a desirable performance measure (such as time).


Figure 9. Overview of the simulation-optimization method.

We use the Genetic Algorithm (GA), as the meta-heuristic technique. GA is an algorithm inspired by the basic mechanism of natural evolution, introduced by Holland (1975). The GA procedure initializes from a randomly generated population of solutions, and evolves good local solutions by mimicking the process of natural selection using mechanisms such as mutation to generate variants and crossover to improve combinations (Trevino \& Falciani, 2006). GA is a population-based algorithm and employs random choices to have a highly exploitative search, keeping a balance between exploration of the feasible domain and exploitation of good solutions. In this works, the parameters of GA are tuned properly by running several experiments with different values for those parameters to ensure the quality of solutions. The ultimate values of the GA parameters are listed in Table 26.

Table 26. The values for the GA parameters.

| Parameters | Value |
| :--- | :--- |
| Initial population size | 200 |
| Crossover percentage | 0.8 |
| Mutation percentage | 0.8 |
| Mutation rate | 0.03 |

In this study, a new strategy is used to represent solutions generated by GA. This solution representation determines (i) the sequence of jobs in the flow shop system, and (ii) the number of MAs needed to be performed on each machine prior to each job by choosing one of combination. Table 27 shows the solution representation for a flow shop system with 4 jobs, 3 machines, and 3 MAs. The second row in the solution matrix indicates the sequence of jobs that go through processes in all machines (permutation flow shop). The numbers provided in the $3^{\text {rd }}$ to the $5^{\text {th }}$ rows indicate the combination of MAs that needs to be performed on each machine before processing each job. Table 25 showed what MAs are included in each combination. For instance, Combination 4, which includes MAs 1 and 2 , should be performed on Machine 1 before processing Job 3.

Table 27. An example for the solution represtaion of the GA.

| Parameters | Values |  |  |  |
| :---: | :---: | :---: | :---: | :---: |
| Jobs | 1 | 2 | 3 | 4 |
| Jobs Sequence | 4 | 1 | 3 | 2 |
| The Required <br> Combination of <br> MAs on: | Machine 1 | 7 | $0^{*}$ | 4 |

* It is assumed that no MA is needed before the first job in the sequence.

The number of variables used to represent a solution for a given problem with $m$ machines, $n$ jobs, and $l$ MAs is $(n+n m)$, which is independent of $l$. This is another advantage of using the concept of combined MAs. We use real numbers for chromosome representation, as shown in Figure 10. The chromosomes are then parsed to obtain the sequence of the jobs, and the required combination of MAs before processing each job on each machine. In Figure 10, for 4 jobs, 3 machines, and 3 MAs , each chromosome is coded as an array of $(4+(4-1) \times 3=13)$ real numbers. The rank position of the first four numbers indicate the sequence of the jobs. The rest of the numbers are converted to an integer value ( $1,2, \ldots, 8$ ) to determine the combination of MAs that are required before processing each job on each machines, according to Table 25. For instance, the generated real number for the MA policy of the $3^{\text {rd }}$ job in the sequence (Job 3) on Machine 2 is 0.701 . By mapping this number to the range of $1-8$, combination 6 is obtained.


Figure 10. Chromosome representation using real numbers.

This solution representation, compared to the ones presented in Chapters I and II, considerably improves the performance of the GA for the presented problem. In the chromosome representations proposed by Seif et al. (2017) and Yu and Seif (2016) (Chapters I and II), the chromosomes are presented via $n+$ nml integer variables. This representation, unlike the ones proposed in this paper, grows as the number of maintenance activities increases. Because the variables are integer, their representation also requires additional computational efforts for feasibility checks within/after crossover and mutation operations.

### 3.4 Computational Experiments

In this section, we present the computational experiments that we designed to tune the SO algorithm, and to evaluate the performances of the algorithm in comparison with a commercial solver. First, we introduce the methods used to generate test problems for the experiments. Then, we discuss tuning of the algorithm. Finally, we present the main experiments that evaluate the SO algorithm by comparing its performance against a commercial solver. We coded the algorithm in MATLAB and used IBM ILOG CPLEX Optimization Studio (Version: 12.5.1.0) to solve the problem formulated in Equations (3.3-3.37) in Section 3.2. All algorithms and the CPLEX solver were run on an i7-3770 @ 3.40 gigahertz Intel processor with 8.00 gigabytes of system memory. Throughout the experiments, we solve 30 test problems with various settings. Solving 30 test problems allows us to draw reliable inferences about the performance of the solution methods. Each test problem includes 30 scenarios. These scenarios are generated via Monte-Carlo simulation. The comparison is between two different solution approaches, namely the exact solution methods built in the commercial solver and a metaheuristic method, which is the proposed GA. Both the solver and the GA check the feasibility of a solution against all of the constraints defined in the stochastic program that was presented in Section 3.2.

## Test Problem Generation

A test problem must contain the values of all the parameters introduced in Section 3.2. We used a test problem generator, which is a function with the following arguments: number of jobs ( $n$ ), Due Date Tightness Factor (DDTF), and Maintenance Interval Factor (MIF). The values of the parameter in each test problem are generated as follows. Number of machines, $m=3$, number of MAs, $l=3$, the health states of the machines, $H=3$, number of scenarios, $S=30$, maintenance intervals, $R_{1}=4 \times M I F, R_{2}=5 \times M I F, R_{3}=6 \times M I F$, spare part costs, $S P_{i k} \sim U(\$ 150, \$ 450)$, the workforce cost, $W F=\$ 20 /$ hour, the due dates, $d_{j} \sim U\left(240,\left\lfloor\frac{240 n}{D D F T}\right\rfloor\right)$, penalty costs, $\pi_{j} \sim U(10,20)$, the probability of scenarios, $\operatorname{Pr}(s)=1 / S$, the coefficients for prolonging the processing times, $\lambda^{1}=1.0, \lambda^{1}=$ $1.5, \lambda^{1}=2.0$, nominal processing times, $p_{i j}^{S} \sim T R I(20,35,70)$, and the nominal duration of MAs, $e_{i k}^{S} \sim \operatorname{TRI}(5,15,25)$, where $\sim U(a, b)$ denotes a random value that follows the Uniform distribution in the range [ $a, b$ ], and $\sim \operatorname{TRI}(a, b, c)$ denotes a random value that follows the Triangular distribution with $a, b$, and $c$ as the minimum, most likely, and the maximum values that the random variable can take, respectively. All the test problems (as CPLEX files) can be retrieved online at this link:
https://www.dropbox.com/sh/wn7u776g4fwzren/AADC0LdeITjWuF9WWTZDUmd Ma ? $\mathrm{dl}=0$.

Although these might be close to the values used in some of applications, the sole purpose of generating them within the presented bounds is to form test problems that are both feasible and challenging to solve. We will present a case study in Section 3.5 in which the values are chosen such that they represent an application that is similar to one of the real world problems. We keep the number of machines, types of MAs, and the health state of the machines constant in all of the test problems, but we will increase the number of jobs in order to test the performance of the algorithm in dealing with large-scale instances of the problem. In practice and for a particular application, the values that are fixed do not usually change significantly yet the number of jobs is usually subject to change and will increase. Therefore, we decided to only increase the number of jobs in the forthcoming experiments.

## Computational Experiments for the Population Size

Yu and Seif (2016) (Chapter I) use a GA for solving a flow shop scheduling problem with diverse maintenance activities, and showed that only the population size is statistically significant in improving the quality of solutions. They also showed that increasing the population size up to a certain point increases the quality of solutions. After that point, the quality does not improve significantly, yet the solution time keeps increasing. Table 28 shows the results of the experiment we performed in order to find an appropriate population size for the problem presented in this paper. First, we generated a test problem, solved it with the GA, and then recorded the objective function value (OFV) and the solution while the population size was 25 . We used these values as the baseline. Then, we solved the same problem with the same settings, yet with a larger population size, and recorded the improvement in the OFV and increase in the solution time. We repeated this experiment three times for three different problem sizes. The average improvements and increases are reported in Table 28.

Table 28. The impact of population size on the performance of the algorithm.

| Number of <br> Jobs $(n)$ | Population Size | Average Improvement in the <br> OFV (Cost) | Average Increase in the <br> Solution Time |
| :---: | :---: | :---: | :---: |
| 4 | 25 | $0 \%$ | $0 \%$ |
| 4 | 50 | $7 \%$ | $77 \%$ |
| 4 | 100 | $15 \%$ | $255 \%$ |
| 4 | 200 | $22 \%$ | $546 \%$ |
| 4 | 400 | $27 \%$ | $1215 \%$ |
| 6 | 25 | $0 \%$ | $0 \%$ |
| 6 | 50 | $6 \%$ | $104 \%$ |
| 6 | 100 | $18 \%$ | $399 \%$ |
| 6 | 200 | $22 \%$ | $756 \%$ |
| 6 | 400 | $17 \%$ | $1394 \%$ |
| 8 | 25 | $0 \%$ | $0 \%$ |
| 8 | 50 | $10 \%$ | $131 \%$ |
| 8 | 100 | $17 \%$ | $359 \%$ |
| 8 | 200 | $22 \%$ | $764 \%$ |
| 8 | 400 | $23 \%$ | $1720 \%$ |

Figure 11 summarizes the results of Table 28, by plotting the total average of improvements and increases against the population size. Increasing the population size beyond 200 does not lead to any improvement in the OFV, yet the solution time keeps increasing, as Figure 12 shows. This result agrees with the findings of Yu and Seif (2016), which is the published verion of Chapter I.

## Computational Experiment for Performance Evaluation

In this section, we present computational experiments that evaluate the performance of the presented Simulation-Optimization (SO) method compared with CPLEX as a commercial solver that uses exact solution algorithms for the presented SMIP formulation. In all of the experiments, we use the following stopping conditions for the SO algorithm and CPLEX. The algorithm will stop and return the best solution after I number of iterations, or after the OFV does not improve for $[0.2 \times I]$ iterations. We chose $I=100$, but it can take any positive integer value. Obviously, a higher value of $I$ is more likely to result in a lower OFV and higher solution time, and depends on user's preference.. For CPLEX, we observed that when the optimal solution cannot be find within an hour, there is a possibility that it cannot be found even within several hours. Therefore, we set a time limit of 3000 seconds for CPLEX. However, if the solution time of the algorithm becomes greater than 3000 seconds, we use a time limit greater than the solution time of the algorithm, as the CPLEX time limit. We generated a randomized test problem and solved it with both algorithms, SO and CPLEX solver, to ensure these algorithms are consistent and accurate.

Table 29 shows the results of solving 30 test problems solved once via CPLEX and once via SO. The number of jobs $(n)$ is 4 in all of these test problem and the number of scenarios is 30 in each problem. These problem are considered as small-scale problems. In some of the test problems (bold-faced and underlined) the SO method finds the global optimal solution, and in some problems (bold-faced) the gap between the two methods is less than $1.00 \%$ for the OFV. However, the solution time of the SO is considerably larger than that of CPLEX. Next, we want to see how the results of the comparison changes when the problem size (number of jobs) increases.

At the bottom of Table 29, the results are summarized by reporting the average, minimum, and maximum values of each column. Table 30 summarizes the results for the average of 30 test problems for different number of jobs. A negative gap means that the SO algorithm has performed better than CPLEX. One observation is that, as the number of jobs (problem size) increases, the quality of CPLEX solutions decreases under a limited solution time. Also, the gap between the SO algorithm and CPLEX decreases with respect to both OFV and solution time. The maximum gaps in the $7^{\text {th }}$ and the $10^{\text {th }}$ columns show the worse


Figure 11. The impact of population size on the objective function.


Figure 12. The impact of population time on solution time.

Table 29. Comparing the simulation-optimization method with CPLEX for $n=4$.

| Test Problem | CPLEX (SMIP) |  | Simulation-Optimization (SO) |  |  | Gap |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | OFV | $\begin{gathered} \hline \text { Time } \\ \text { (sec.) } \end{gathered}$ | OFV | Time (sec.) | Iterations | OFV | Time |
| 1 | 6210.44 | 96 | 6263.10 | 414 | 45 | 0.85\% | 332.04\% |
| 2 | 6534.53 | 111 | 6938.77 | 443 | 47 | 6.19\% | 298.53\% |
| 3 | 5556.52 | 121 | 5577.18 | 504 | 55 | 0.37\% | 315.17\% |
| 4 | 6598.37 | 98 | 7166.55 | 450 | 48 | 8.61\% | 357.97\% |
| 5 | 5710.22 | 113 | 5710.27 | 455 | 49 | 0.00\% | 302.32\% |
| 6 | 5211.41 | 89 | 5360.73 | 550 | 60 | 2.87\% | 521.18\% |
| 7 | 5604.63 | 136 | 5604.68 | 332 | 36 | 0.00\% | 144.24\% |
| 8 | 7235.49 | 114 | 7288.27 | 751 | 84 | 0.73\% | 559.98\% |
| 9 | 6993.4 | 85 | 7932.77 | 417 | 45 | 13.43\% | 388.73\% |
| 10 | 5470.29 | 86 | 5728.43 | 389 | 42 | 4.72\% | 353.86\% |
| 11 | 6325.52 | 138 | 6740.75 | 451 | 48 | 6.56\% | 226.21\% |
| 12 | 5700.31 | 113 | 6338.80 | 450 | 49 | 11.20\% | 298.14\% |
| 13 | 7250.16 | 86 | 7616.83 | 429 | 46 | 5.06\% | 396.56\% |
| 14 | 5425.85 | 171 | 5427.85 | 525 | 58 | 0.04\% | 207.04\% |
| 15 | 5291.15 | 179 | 6153.85 | 596 | 65 | 16.30\% | 232.12\% |
| 16 | 6798.6 | 143 | 6814.17 | 561 | 60 | 0.23\% | 291.54\% |
| 17 | 5243.63 | 103 | 6486.17 | 390 | 41 | 23.70\% | 279.57\% |
| 18 | 7035.76 | 114 | 7074.84 | 416 | 45 | 0.56\% | 265.91\% |
| 19 | 6048.12 | 112 | 6092.67 | 426 | 46 | 0.74\% | 279.50\% |
| 20 | 7329.23 | 144 | 7771.63 | 555 | 60 | 6.04\% | 286.41\% |
| 21 | 6832.7 | 132 | 7229.67 | 541 | 58 | 5.81\% | 308.26\% |
| 22 | 5176.95 | 114 | 5177.00 | 531 | 58 | 0.00\% | 365.59\% |
| 23 | 5388.92 | 96 | 6005.27 | 526 | 56 | 11.44\% | 446.69\% |
| 24 | 5831.19 | 153 | 6687.63 | 389 | 41 | 14.69\% | 153.71\% |
| 25 | 7286.2 | 94 | 7527.65 | 582 | 64 | 3.31\% | 519.06\% |
| 26 | 5079.85 | 180 | 5079.90 | 430 | 47 | 0.00\% | 138.78\% |
| 27 | 6122.64 | 103 | 6664.98 | 411 | 44 | 8.86\% | 300.46\% |
| 28 | 5452.83 | 124 | 5452.88 | 586 | 64 | 0.00\% | 374.55\% |
| 29 | 7393.03 | 145 | 7734.09 | 441 | 48 | 4.61\% | 203.53\% |
| 30 | 6329.74 | 117 | 6329.80 | 601 | 67 | 0.00\% | 412.40\% |
| Average | 6148.92 | 120 | 6465.91 | 485 | 52 | 5.23\% | 318.67\% |
| Minimum | 5079.85 | 85 | 5079.90 | 332 | 36 | 0.00\% | 138.78\% |
| Maximum | 7393.03 | 180 | 7932.77 | 751 | 84 | 23.70\% | 559.98\% |

performance of the SO algorithm compared to CPLEX. When the gap for the worst case scenario is negative, it means that the SO algorithm has consistently performed better than CPLEX within a time limit. These values are indicated in boldfaced. As a result, it can be concluded that under a limited solution time, the proposed SO algorithm outperforms a commercial solver as the problem size increases. In Table 30 we increased the number of jobs up to a point where all 30 test problems can be solved with CPLEX. Table 31 shows the results for $n=10$. We increased the solution time limit to 5000 seconds which is considerably higher than the average solution time of the SO algorithm. For the bold-faced test problems, CPLEX was unable to find any feasible solution for the problem. Again, even in the worst-case scenario, the SO algorithm finds a better solution with a lower OFV than CPLEX.

Table 30. Comparing the simulation-optimization method with CPLEX when problem size increases.

| N. of Jobs (n) | CPLEX Avg. Time |  | Avg. N of Iter. | Gap in the Solution Time |  |  | Gap in the OFV |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  | Avg. | Min. | Max. | Avg. | Min. | Max. |
| 4 | 120 | 485 | 52 | 318.67\% | 138.78\% | 559.98\% | 5.23\% | 0.00\% | 23.70\% |
| 5 | 2383 | 777 | 68 | -63.78\% | -79.81\% | -21.89\% | 9.43\% | 0.00\% | 32.46\% |
| 6 | 3002 | 1091 | 79 | -63.66\% | -78.43\% | -53.80\% | 0.99\% | -15.18\% | 21.19\% |
| 7 | 3001 | 1430 | 88 | -52.34\% | -73.58\% | -44.63\% | -10.57\% | -34.84\% | 17.08\% |
| 8 | 3001 | 1828 | 98 | -39.09\% | -52.89\% | -35.11\% | -23.38\% | -43.39\% | -6.48\% |
| 9 | 3000 | 2081 | 98 | -30.63\% | -44.06\% | -25.03\% | -26.64\% | -42.29\% | -8.22\% |

As the last part of the experiments, we want to make sure that the quality of solutions of the algorithm (as measured by the gap in OFV), as well as the time to find a solution, are not dependent on how we generate the input data of the test problems. In other words, we want to examine the impact of the input data on the performance of the proposed algorithm. We changed the arguments (DDTF and MIF) of the test problem generator function introduced in Section 3.4 and solved 30 test problems for each setting. Table 32 shows a summary of the results. Table 33 provides the average, minimum, and maximum solution times in CPLEX and the SO algorithm, for each setting. Table 33 provides the average, minimum, and maximum solution times in CPLEX and the SO algorithm, for each setting. In order to examine whether the gap or solution time are significantly affected by the input data we performed analysis of variance (ANOVA) on samples drawn from the data used for Table 32 and Table 33. Table 34 and Table 35 are the ANOVA tables in which five treatments ( $a=5$, the way test problems are generated, the settings), a sample size of seven ( $n=7$ ), and a confidence interval of $\alpha=0.01$ is used.

As the results suggest, the input data has no statistical significance in the solution time or the quality of the solutions (the gaps). This is intuitive when

Table 31. Comparing the simulation-optimization method with CPLEX for $\mathrm{n}=10$.

| Test <br> Problem | CPLEX (SMIP) |  | Simulation Optimization |  |  | Gap |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | OFV | Time <br> (sec.) | OFV | Time <br> (sec.) | Iterations | OFV | Time |
| 1 | 46517.1 | 5001 | 4934 | 31969.8 | 100 | $-31.27 \%$ | $-1.33 \%$ |
| 2 | 46070.3 | 5000 | 4862 | 34726.5 | 100 | $-24.62 \%$ | $-2.76 \%$ |
| 3 | 48594 | 5001 | 4840 | 36507.0 | 100 | $-24.87 \%$ | $-3.21 \%$ |
| 4 |  | 4999 | 4778 | 32680.8 | 100 |  |  |
| 5 | 53894.7 | 5000 | 4878 | 35050.7 | 100 | $-34.96 \%$ | $-2.44 \%$ |
| 6 | 42035.5 | 5000 | 4796 | 32312.5 | 100 | $-23.13 \%$ | $-4.09 \%$ |
| 7 | 46760 | 5000 | 4852 | 32919.7 | 100 | $-29.60 \%$ | $-2.95 \%$ |
| 8 | 48287.1 | 5000 | 5011 | 35255.7 | 100 | $-26.99 \%$ | $0.22 \%$ |
| 9 |  | 5000 | 4882 | 30531.8 | 100 |  |  |
| 10 | 37772.4 | 5000 | 4907 | 25120.3 | 100 | $-33.50 \%$ | $-1.86 \%$ |
| 11 | 49068.5 | 5002 | 4998 | 35048.1 | 100 | $-28.57 \%$ | $-0.08 \%$ |
| 12 |  | 5000 | 4996 | 40068.8 | 100 |  |  |
| 13 | 47049.4 | 5001 | 4988 | 32131.5 | 100 | $-31.71 \%$ | $-0.26 \%$ |
| 14 | 41246.2 | 5000 | 4864 | 23863.0 | 100 | $-42.15 \%$ | $-2.72 \%$ |
| 15 | 43165.8 | 5001 | 4842 | 34974.3 | 100 | $-18.98 \%$ | $-3.18 \%$ |
| 16 | 40944.7 | 5001 | 4946 | 27929.2 | 100 | $-31.79 \%$ | $-1.10 \%$ |
| 17 | 47965.9 | 4999 | 4819 | 35923.4 | 100 | $-25.11 \%$ | $-3.59 \%$ |
| 18 | 46645 | 5001 | 4857 | 39589.5 | 100 | $-15.13 \%$ | $-2.88 \%$ |
| 19 | 42654.6 | 5000 | 4900 | 23057.4 | 100 | $-45.94 \%$ | $-2.00 \%$ |
| 20 | 47003.6 | 5000 | 4928 | 39221.8 | 100 | $-16.56 \%$ | $-1.46 \%$ |
| 21 | 41515.4 | 5000 | 4917 | 34909.3 | 100 | $-15.91 \%$ | $-1.66 \%$ |
| 22 | 44415.9 | 5000 | 4878 | 32601.5 | 100 | $-26.60 \%$ | $-2.45 \%$ |
| 23 | 50445 | 5000 | 4866 | 31690.7 | 100 | $-37.18 \%$ | $-2.70 \%$ |
| 24 | 37282.9 | 4999 | 4825 | 31712.7 | 100 | $-14.94 \%$ | $-3.47 \%$ |
| 25 | 33822.2 | 5000 | 4838 | 30608.1 | 100 | $-9.50 \%$ | $-3.25 \%$ |
| 26 |  | 5000 | 4827 | 34578.4 | 100 |  |  |
| 27 | 36036.3 | 5000 | 4824 | 33809.9 | 100 | $-6.18 \%$ | $-3.52 \%$ |
| 28 | 51582.1 | 5000 | 4877 | 32966.8 | 100 | $-36.09 \%$ | $-2.47 \%$ |
| 29 | 44263.5 | 5000 | 4764 | 32498.6 | 100 | $-26.58 \%$ | $-4.72 \%$ |
| 30 | 49271 | 5002 | 4812 | 35931.3 | 100 | $-27.07 \%$ | $-3.79 \%$ |
| Average | 44781.1 | 5000 | 32781.9 | 4878 | 100 | $-26.34 \%$ | $-2.45 \%$ |
| Minimum | 33822.2 | 4999 | 23057.4 | 4764 | 100 | $-45.94 \%$ | $-4.72 \%$ |
| Maximum | 53894.7 | 5002 | 39589.5 | 5011 | 100 | $-6.18 \%$ | $0.22 \%$ |
|  |  |  |  |  |  |  |  |

Table 32. Sensitivity of the gap to the input data, $n=4$.

| Setting | DDTF | MIF | No. of Problems Solved | Gap in the OFV |  |  |
| :--- | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  |  | Avg. | Min. | Max. |
| 1 | 3 | 50 | 30 | $10.42 \%$ | $0.00 \%$ | $26.63 \%$ |
| 2 | 5 | 50 | 30 | $9.41 \%$ | $0.00 \%$ | $32.29 \%$ |
| 3 | 4 | 50 | 30 | $11.82 \%$ | $0.29 \%$ | $30.77 \%$ |
| 4 | 4 | 60 | 30 | $15.39 \%$ | $0.49 \%$ | $32.40 \%$ |
| 5 | 30 | $11.92 \%$ | $0.00 \%$ | $37.32 \%$ |  |  |

Table 33. Sensitivity of the solution time to the input data, $n=4$.

| Setting | DDTF | MIF | The solution time of CPLEX |  |  |  | The solution time of the SO algorithm |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  | Avg. (sec.) | Min. (sec.) | Max. (sec.) | Variance | Avg. (sec.) | Min. (sec.) | Max. (sec.) | Variance |
| 1 | 4 | 40 | 74.20 | 5.58 | 259.03 | 4,358.92 | 576.47 | 316.50 | 1,013.11 | 26,187.84 |
| 2 | 4 | 50 | 198.48 | 42.80 | 418.69 | 8,263.82 | 551.35 | 323.34 | 759.26 | 11,789.68 |
| 3 | 4 | 60 | 203.72 | 96.48 | 426.07 | 5,990.88 | 510.35 | 228.56 | 670.08 | 7,919.88 |
| 4 | 3 | 40 | 300.70 | 42.40 | 1,179.03 | 54,610.11 | 565.92 | 361.50 | 831.70 | 12,494.47 |
| 5 | 5 | 40 | 199.73 | 48.13 | 464.04 | 9,319.93 | 524.53 | 352.58 | 722.45 | 8,576.37 |

Table 34. Analysis of variance for the gap sensitivity experiment.

| Source of | Sum of <br> Squares | Degree of <br> Freedom | Mean <br> Square | $\mathrm{F}_{0}$ | P-Value |
| :--- | :---: | :---: | :---: | :---: | :---: |
| Setting | 0.0558 | 4 | 0.0140 | 2.6731 | 0.0510 |
| Error | 0.1567 | 30 | 0.0052 |  |  |
| Total | 0.2126 | 34 |  |  |  |

Table 35. Analysis of variance for the experiment on solution time sensitivity.

| Source of | Sum of | Degree of | Mean | $F_{0}$ | P-Value |
| :--- | :--- | :--- | :--- | :--- | :--- |
| Variation | Squares | Freedom | Square |  |  |
| Setting | 19196.26013 | 4 | 4799.0650 |  |  |
| Error | 349524.7108 | 30 | 11650.8237 |  |  |
| Total | 368720.9709 | 34 |  |  |  |

looking at the results in Table 32 and Table 33. This finding implies that the results that were shown previously are independent of the problem instances and the conclusions about the performance of the SO method are robust.

### 3.5 Case Study

In this section we show one of the applications of the presented problem and solution method. The input data and description of this case study are adapted from the case study by Yu and Seif (2016) (a published version of Chapter I) that is designed for a deterministic flow shop scheduling with multiple MAs in which combining the MAs was not considered and processing times were fixed regardless of the machines' health state. After presentation of the data and describing the case study, we will discuss the solution and draw managerial implications.

One of the main activities in the early stages of a heavy construction project is earthmoving. A simplified version of the earthmoving process described by Fu (2013) is as follows. The first step is called preparation. Excavators are used in this step; they dig natural form of material from the earth. Next, in the loading step, wheel loaders can load the removed and prepared soil into haul trucks. Finally, in the hauling step, haul trucks transport earth to a deposit point by travelling through routes.

Typical (preventive) maintenance activities for construction machinery are usually based on the operating hours of the machinery. In Table 36, maintenance intervals ( $R_{k}, k=1, \ldots, 6$ ) recommended by one of the manufacturers of heavy construction equipment is listed for the machinery that are required for the simplified earthmoving process (Caterpillar, 2010a, 2010b, 2010c). Different tasks are included in each MA. For example, the tasks included in the 50-hour MA of excavators shown are lubrication of boom, stick and bucket linkage, drive shaft universal joint, etc.

Table 38 shows the task lists of the 250 -hour, 500 -hour, and 1000 -hour MAs for the excavator. Tasks Numbers 1-4 for the 500-hour MA are shared in the 1000-hour MA, as shown in boldfaced. When combined, the total duration of these two MAs should be approximately $75 \%$ of the sum of the durations of the two MAs because $25 \%$ of the tasks listed under the two MAs will be redundant when they are combined (assuming that the tasks have the same duration). Although the 250-hour MA does not share any tasks with the other two MAs, after checking the details of some of the tasks we noticed that their share certain steps within their tasks. Table 39 shows the steps for performing Task Number 11 of the 250 -hour MA and Task Number 4 of the 500 -hour MA. Steps 1,

Table 36. Maintenance intervals (in hours) recommended by the equipment manufacturer (Caterpillar, 2010a, 2010b, 2010c), reproduced from (Yu \& Seif, 2016).

| Machine | 10 | 50 | 100 | 250 | 500 | 1000 |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Excavators | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ |
| Wheel Loaders | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ |
| (Haul) Trucks | $\checkmark$ | $\checkmark$ |  |  | $\checkmark$ |  |

Table 37. Processing times, due dates, and penalty costs for the jobs, adapted from Yu and Seif (2016).

| Location <br> $($ Jobs $)$ | Processing Times (no. of days $\times$ hours $/$ day $)$ |  |  | Due <br> Date | Penalt <br> y/ |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | Excavator | Wheel Loader | Truck |  | days) |
| $L_{1}$ | $\sim \operatorname{TRI}(20,22,25) \times 8$ | $\sim \operatorname{TRI}(25,28,30) \times 8$ | $\sim \operatorname{TRI}(5,7,9) \times 16$ | 90 | $\$ 211$ |
| $L_{2}$ | $\sim \operatorname{TRI}(15,20,25) \times 8$ | $\sim \operatorname{TRI}(20,25,30) \times 8$ | $\sim \operatorname{TRI}(7,10,14) \times 16$ | 100 | $\$ 118$ |
| $L_{3}$ | $\sim \operatorname{TRI}(10,15,20) \times 8$ | $\sim \operatorname{TRI}(15,20,25) \times 8$ | $\sim \operatorname{TRI}(3,9,14) \times 16$ | 80 | $\$ 118$ |
| $L_{4}$ | $\sim \operatorname{TRI}(18,23,28) \times 8$ | $\sim \operatorname{TRI}(15,20,25) \times 8$ | $\sim \operatorname{TRI}(9,11,13) \times 16$ | 70 | $\$ 346$ |

Table 38. Task list of each MA for the excavator (excerpts from Caterpillar (2010c)).

| Task No. | Maintenance Activity Interval (hours) |  |  |
| :--- | :--- | :--- | :--- |
|  | 250-hour maintenance | 500-hour maintenance | 1000-hour maintenance |
| 1 | Air Conditioner - Test | Axle Oil (Front) - Change | Axle Oil (Front) - Change |
| 2 | Axle Bearings (Front) - Lubricate | Axle Oil (Rear) - Change | Axle Oil (Rear) - Change |
| 3 | Axle Oil Level (Front) - Check | Final Drive Oil - Change | Battery Hold-Down - Tighten |
| 4 | Axle Oil Level (Rear) - Check | Transmission Oil - Change | Drum Brakes - Inspect |
| 5 | Braking System - Test | Drive Shaft Support Bearing Lubricant - Check | Final Drive Oil - Change |
| 6 | Condenser (Refrigerant) - Clean | Fuel System Priming Pump - Operate | Overhead Guard - Inspect |
| 7 | Cooling System Hoses - Inspect | Fuel System Secondary Filter - Replace | Transmission Oil - Change |
| 8 | Engine Oil and Filter - Change | Fuel Tank Cap and Strainer - Clean |  |
| 9 | Final Drive Oil Level - Check | Fuel System Primary Filter/Water Separator- |  |
| 10 | Swing Bearing - Lubricate | Element - Replace |  |
| 11 | Transmission Oil Level - Check |  |  |
| 12 | V-Belts - Inspect/Adjust/Replace |  |  |

Table 39. Two tasks with similar steps (excerpts from Caterpillar (2010c)).

| Task | Steps |
| :---: | :---: |
| Transmission Oil Level - Check | 1. Remove filler plug (1). |
|  | 2. Check the lubricant level. The lubricant level should be at the bottom of the opening for filler plug (1). |
|  | 3. If necessary, fill the gearbox with lubricant to the bottom of the opening for filler plug (1). |
|  | 4. Clean filler plug (1). |
|  | 5. Inspect the O-ring seal. If damage or wear is noticed on the O-ring seal, replace the seal. |
|  | 6. Install filler plug (1). |
|  | 1. Remove the dirt that is around filler plug (1) and around drain plug (2). |
|  | 2. Remove drain plug (2). Drain the lubricant into a suitable container. |
|  | 3. Clean drain plug (2). |
| Transmission Oil - Change | 4. Inspect the O-ring seal. If damage or wear is noticed on the O-ring seal, replace the seal. |
|  | 5. Install drain plug (2). |
|  | 6 . Remove filler plug (1). |
|  | 7. Fill the gearbox with lubricant to the bottom of the filler plug opening. |
|  | 8. Clean filler plug (1). |
|  | 9. Inspect the O-ring seal. If damage or wear is noticed on the O-ring seal, replace the seal. 10. Install filler plug (1). |

4,5 , and 6 of the first task are the same as Steps $6,8,9$, and 10 of the second task. We assumed the duration of the combined tasks to be $60 \%$ of the sum of the three durations. In practice, these values can be calculated precisely after a time study is conducted on the MAs.

We consider a project with four locations, in which earth moving operations need to be done. There are three machines allocated for earthmoving operations of these locations; one excavator, one-wheel loader, and one truck. The locations are too far from each other for the machines to be able to simultaneously work in more than one location. In Table 37, the operation requirements in each location are shown. Table 37 shows the processing times as the number of days a machine is expected to work in a location multiplied by the number of hours worked per day. The due date and penalty costs for completing a job after the due date are also presented in this table. We assumed a 10-hours shift for the working days in the last two columns.

Average cost of performing a preventive maintenance activity on a wheel loader is approximately $\$ 234$ (Azadeh et al., 2014). We have used this value to approximate the overall cost of each MA's spare part cost, i.e. $\sim U(\$ 200, \$ 300)$. We consider $\$ 25 /$ hour as the workforce cost. Because the first three MAs (10, 50,100 hours) are usually done in a fraction of an operational day, and usually by the operators, where the machine is operating, and because 2000 hours MAs and above are not going to be reached within the scheduling process for this case study, we have considered only the 250 -hour, 500 -hour, and 1000 -hour MAs. Because the trucks do not have the 250-hour MA, we set its maintenance interval equal to infinity, $R_{3,1}=+\infty$, in order to nullify it. Although these MAs can be performed ideally in one day, we consider the triangular distribution $\sim T R I(1,2,5)$ for the maintenance durations because in practice the machines might wait in the maintenance station for a few days due to spare part unavailability, no empty spot being available in the maintenance stations, etc.

The optimal solution is presented in Table 40. This solution provides a schedule for routing of the machines between the construction locations and a maintenance station, as well as the maintenance plan. For example, the truck goes to $L_{1}$ first, then goes to $L_{4}$, then to the maintenance station because Maintenance Combination 2 is scheduled before processing the third job $\left(L_{1}\right)$. Maintenance Combination 2 means performing only the 500-hour MA. Note that the truck does not need the 250-hour MA. After maintenance, it goes to $L_{1}$, and then to $L_{2}$. The excavator and loader need to go to stop for maintenance before operating in any of the locations (except for the first location). Maintenance Combinations 1, 4, and 7 correspond to performing only the 250-hour MA, the 250 -hour and the 500-hour MAs in a row, and all three MAs in a row, respectively.

We solved the problem again after changing the input data for the durations of the MAs. This time we used the sum of MAs for combinations, instead of a portion ( $75 \%$ or $60 \%$ ) of the sum. Table 41 shows the solution for the new problem. The only changes in the optimal schedule are the MAs of the loader before going to $L_{4}$ and $L_{1}$. With the new data, Combination 5 which includes the 500 -hour and the 1000 -hour MAs is prescribed before $L_{4}$, and Combination 4 which includes the 250 -hour and the 500 -hour MAs is prescribed before operating in $L_{1}$. This means performing an excessive 500-hour MA compared to the original solution. The reason is that in the new data performing the MAs takes longer which leads to an increase in tardiness. The solver tries to compensate for this increase in the duration of the MAs by performing more MAs so that the processing times of the jobs do not get prolonged due to the poor health of the machine. However, the value of the objective function is still worse than the original problem.

Table 40. The optimal solution for the case study.

| Variable | Optimal Value |
| :--- | :--- |
|  |  |
| $2^{\text {nd }}$ location to process | $L_{4}$ |
| $3^{\text {rd }}$ location to process | $L_{1}$ |
| $4^{\text {th }}$ location to process | $L_{2}$ |
| Maintenance combination for the exacavator before processing the $1^{\text {st }}$ job | 0 |
| Maintenance combination for the exacavator before processing the $2^{\text {nd }}$ job | 1 |
| Maintenance combination for the exacavator before processing the $3^{\text {rd }}$ job | 4 |
| Maintenance combination for the exacavator before processing the $4^{\text {th }}$ job | 1 |
| Maintenance combination for the loader before processing the $1^{\text {st }}$ job | 0 |
| Maintenance combination for the loader before processing the $2^{\text {nd }}$ job | 1 |
| Maintenance combination for the loader before processing the $3^{\text {rd }}$ job | 7 |
| Maintenance combination for the loader before processing the $4^{\text {th }}$ job | 1 |
| Maintenance combination for the truck before processing the $1^{\text {st }}$ job | 0 |
| Maintenance combination for the truck before processing the $2^{\text {nd }}$ job | 0 |
| Maintenance combination for the truck before processing the $3^{\text {rd }}$ job | 2 |
| Maintenance combination for the truck before processing the $4^{\text {th }}$ job | 0 |
| Total Expected Cost | $\$ 4,078$ |
| Expected Maintenance Cost | $\$ 3,076$ |
| Expected Penalty Cost | $\$ 1,002$ |

Table 41. The optimal solution, when the durations of the MAs do not change in combinations.

|  |  |
| :--- | :--- |
| Variable | Optimal Value |
|  |  |
| $1^{\text {st }}$ location to process | $L_{3}$ |
| $2^{\text {nd }}$ location to process | $L_{4}$ |
| $3^{\text {rd }}$ location to process | $L_{1}$ |
| $4^{\text {th }}$ location to process | $L_{2}$ |
| Maintenance combination for the exacavator before processing the $1^{\text {st }}$ job | 0 |
| Maintenance combination for the exacavator before processing the 2 ${ }^{\text {nd }}$ job | 1 |
| Maintenance combination for the exacavator before processing the 3 3rd job | 4 |
| Maintenance combination for the exacavator before processing the $4^{\text {th }}$ job | 1 |
| Maintenance combination for the loader before processing the $1^{\text {st }}$ job | 0 |
| Maintenance combination for the loader before processing the $2^{\text {nd }}$ job | 5 |
| Maintenance combination for the loader before processing the $3^{\text {rd }}$ job | 4 |
| Maintenance combination for the loader before processing the $4^{\text {th }}$ job | 1 |
| Maintenance combination for the truck before processing the $1^{\text {st }}$ job | 0 |
| Maintenance combination for the truck before processing the $2^{\text {nd }}$ job | 0 |
| Maintenance combination for the truck before processing the $3^{\text {rd }}$ job | 2 |
| Maintenance combination for the truck before processing the 4 $4^{\text {th }}$ job | 0 |
| Total Expected Cost | $\$ 4,188$ |
| Expected Maintenance Cost | $\$ 3,170$ |
| Expected Penalty Cost | $\$ 1,018$ |

### 3.6 Conclusion and Future Research

In this chapter, a new extension of the flow shop scheduling problem was introduced. We incorporated the concept of combined maintenance activities in the permutation flow shop, and considered the impact of the health of machines on the processing times of jobs. The objective was to minimize the total cost of maintenance activities and lateness penalties. We formulated the problem as a two-stage stochastic mixed-integer program in which the first-stage decision variables determined both the sequence of the jobs and a combination of maintenance activities. Because the commercial solvers were not able to solve large-scale instances of the problem in a reasonable time, we developed a simulation-optimization solution method that can efficiently solve these instances. We designed a series of computational experiments in order to tune the algorithm, evaluate its performance in comparison with CPLEX, and assess the sensitivity of its performance to the input data. We concluded that:

- an increase in the population size in the algorithm improves the quality of the solutions only up to a certain point, after which only the solution time increases,
- although for small-sized instances of the problem we recommend the use of commercial/exact solvers, for medium to large-scale instances and under a limited time frame, the presented solution method outperforms these computationally and financially expensive solvers, and
- the quality of the solutions and solution time of the presented simulationoptimization method is not sensitive to the input data under a limited solution time, which alludes to the robustness of the method.

We demonstrated an application of the presented problem through a case study in construction projects. The results of the case study showed that considering the decrease maintenance time, when the activities are combined, leads to savings and improvement in the objective function value. Taking random failures into the consideration is highly desirable and can be studied as an extension of this paper. Also, the concept of combined maintenance activities can be applied to other production settings such as flexible flow shop and job shop scheduling.

## CONCLUSION

This dissertation was an attempt to integrate maintenance decisions into production scheduling. Permutation flow shop scheduling was considered as the production environment, and preventive maintenance activities were incorporated with the scheduling process. In Chapter I, I introduced a new mixed-integer program for flow shop scheduling that could handle scheduling of multiple agebased maintenance activities. The objective was to minimize the overall cost of tardiness (penalty costs) and maintenance. In Chapter II, tardiness and maintenance cost were divided into two separate objectives and the problem was reformulated as a bi-objective optimization problem. The effect of machine's health on processing times was also modeled in Chapter II. In Chapter III, the unified objective in Chapter I was used again, but the processing times, as well as maintenance time were treated as random variables. In addition, the possibility of combining maintenance activities was incorporated into the model. The problem was modeled as a stochastic program in Chapter III, and MonteCarlo simulation was used for scenario generation. The model in Chapter III was slightly different from the ones presented in the earlier chapters; instead of using waiting and buffer times, start/finish times were used for modeling the timeline of each job with respect to various machines.

The problem was shown to be NP-hard in Chapter I. A Genetic Algorithm (GA) was designed and presented as the solution method because the existing generic exact solution methods would be inefficient in solving large sizes of the problem. After performing an ANOVA experiment, the population size of the GA was determined to be the only significant factor in improving the quality of the solutions of the algorithm. Then, a lower bound was formulated for the problem, and the GA would set the population size automatically based on the lower bound and desired performance defined by the user. The GA was improved in Chapter II by confining the solution space. A new design for the GA was presented in Chapter III. The new design improved and algorithm by simplifying its searching procedure.

Extensive computational experiments were conducted in Chapters I-III through which the reliability, efficiency, and effectiveness of the solution methods were demonstrated. In Chapter I, a case study was presented that showed the application of the problem in construction projects. The same case study was used in Chapters II and III. Although the main application of flow shop scheduling is in manufacturing industries, this case study showed how broad the applications of this work can be. The jobs were earthmoving locations, and the machines were construction machinery (loaders, trucks, and excavators).

## Unplanned Maintenance

Unplanned maintenance activities (caused by emergencies, random failures, etc.) were not directly addressed in this dissertation. However, there are two ways to manage them. Because the solution method is relatively fast, after each interruption in the schedule, a new problem defined by the new data can be solved. In applications where unplanned maintenance activities are common, the model that was presented in Chapter III can be used to handle those cases. The duration of maintenance activities were modeled as random variables (input parameters), and the model can handle multiple types of maintenance activities. Therefore, various failure modes can be treated as maintenance activities.

## Considering Risk

An alternative objective function for maintenance is minimizing the risk associated with delaying the maintenance activities. In some organizations, reducing risk is more important than reducing maintenance costs. The main constraint in such cases is on the maintenance budget and maintenance workforce. This dissertation can be extended by considering a risk factor for each type of maintenance, and multiplying this factor by the amount of tardiness in completing the maintenance activities in the objective function (similar to the penalty and tardiness for processing the jobs, in Chapters I and III).

## Fatigue and Degradation of Machines and Their Components

The combination of maximum maintenance levels and deterioration rates that were introduced in Chapter I allows the user of these models to take the degradation of machines and remaining useful life of their components into account. Because the processing times of the jobs are relatively small (compared to the remaining useful life of components) deterioration rates can be used to model the non-linear degradation functions as piece-wise linear. In addition, becase we allow multiple ML's, the degradation values of of multiple components (or multiple degaradation parameters) of a machine, or critical component, can be taken into account. This allows using the output of prognostic models as inputs for the models presented in this dissertation. This work can be extended by changing the maximum maintenance levels to the current values of degradation parameters. The stochastic models also allow taking into account the uncertainity associated with the outputs of prognostic models.

## REFERENCES

Abdollahpour, S., \& Rezaeian, J. (2015). Minimizing makespan for flow shop scheduling problem with intermediate buffers by using hybrid approach of artificial immune system. Applied Soft Computing, 28(0), 44-56. doi:http://dx.doi.org/10.1016/i.asoc.2014.11.022
Aggoune, R. (2004). Minimizing the makespan for the flow shop scheduling problem with availability constraints. European Journal of Operational Research, 153(3), 534-543. doi:http://dx.doi.org/10.1016/S0377-2217(03)00261-3
Aghezzaf, E. H., Jamali, M. A., \& Ait-Kadi, D. (2007). An integrated production and preventive maintenance planning model. European Journal of Operational Research, 181(2), 679-685. doi:http://dx.doi.org/10.1016/j.ejor.2006.06.032
Allaoui, H., \& Artiba, A. (2004). Integrating simulation and optimization to schedule a hybrid flow shop with maintenance constraints. Computers \& Industrial Engineering, 47(4), 431-450. doi:http://dx.doi.org/10.1016/i.cie.2004.09.002
Allaoui, H., Lamouri, S., Artiba, A., \& Aghezzaf, E. (2008). Simultaneously scheduling n jobs and the preventive maintenance on the two-machine flow shop to minimize the makespan. International Journal of Production Economics, 112(1), 161-167.
Alsyouf, I. (2007). The role of maintenance in improving companies' productivity and profitability. International Journal of Production Economics, 105(1), 70-78.
Ángel-Bello, F., Álvarez, A., Pacheco, J., \& Martínez, I. (2011). A single machine scheduling problem with availability constraints and sequence-dependent setup costs. Applied Mathematical Modelling, 35(4), 2041-2050. doi:https://doi.org/10.1016/j.apm.2010.11.017
Aramon Bajestani, M., \& Beck, J. C. (2015). A two-stage coupled algorithm for an integrated maintenance planning and flowshop scheduling problem with deteriorating machines. Journal of Scheduling, 1-16. doi:10.1007/s10951-015-0416-2
Azadeh, A., Asadzadeh, S. M., \& Seif, J. (2014). An integrated simulationanalysis of variance methodology for effective analysis of CBM alternatives. International Journal of Computer Integrated Manufacturing, 27(7), 624-637.
Ben-Daya, M., Ait-Kadi, D., Duffuaa, S. O., Knezevic, J., \& Raouf, A. (2009). Handbook of maintenance management and engineering (Vol. 7): Springer.
Berry, P. M. (1993). Uncertainty in scheduling: probability, problem reduction, abstractions and the user. Paper presented at the Advanced Software Technologies for Scheduling, IEE Colloquium on.
Bierwirth, C. (1995). A generalized permutation approach to job shop scheduling with genetic algorithms. Operations-Research-Spektrum, 17(2-3), 87-92.

Bock, S., Briskorn, D., \& Horbach, A. (2012). Scheduling flexible maintenance activities subject to job-dependent machine deterioration. Journal of Scheduling, 1-14.
Bock, S., Briskorn, D., \& Horbach, A. (2012). Scheduling flexible maintenance activities subject to job-dependent machine deterioration. Journal of Scheduling, 15(5), 565-578.
Brucker, P., Heitmann, S., \& Hurink, J. (2003). Flow-shop problems with intermediate buffers. OR Spectrum, 25(4), 549-574.
Caterpillar. (2010a). 725 and 730 OEM Articulated Trucks - Maintenance Intervals Operation \& Maintenance Manual Excerpt Retrieved from https://safety.cat.com
Caterpillar. (2010b). 988 Wheel Loader - Maintenance Intervals Operation \& Maintenance Manual Excerpt Retrieved from https://safety.cat.com
Caterpillar. (2010c). M312 and M315 Excavators - Maintenance Intervals Operation \& Maintenance Manual Excerpt Retrieved from https://safety.cat.com
Chambers, L. D. (1998). Practical handbook of genetic algorithms: complex coding systems (Vol. 3): CRC press.
Chen, J. S. (2008). Scheduling of nonresumable jobs and flexible maintenance activities on a single machine to minimize makespan. European Journal of Operational Research, 190(1), 90-102.
Cheng, T. C. E., \& Wang, G. (2000). An improved heuristic for two-machine flowshop scheduling with an availability constraint. Operations Research Letters, 26(5), 223-229. doi:https://doi.org/10.1016/S0167-6377(00)00033X
Cheng, T. C. E., Yang, S. J., \& Yang, D. L. (2012). Common due-window assignment and scheduling of linear time-dependent deteriorating jobs and a deteriorating maintenance activity. International Journal of Production Economics, 135(1), 154-161. doi:10.1016/j.ijpe.2010.10.005
Cheng, T. E., \& Wang, G. (1999). Two-machine flowshop scheduling with consecutive availability constraints. Information Processing Letters, 71(2), 49-54.
Choi, B.-C., Lee, K., Leung, J. Y. T., \& Pinedo, M. L. (2010). Flow shops with machine maintenance: Ordered and proportionate cases. European Journal of Operational Research, 207(1), 97-104. doi:https://doi.org/10.1016/i.ejor.2010.04.018
Deb, K., \& Agrawal, S. (1999). Understanding interactions among genetic algorithm parameters. Foundations of Genetic Algorithms, 265-286.
Dehghanimohammadabadi, M. (2016). Iterative Optimization-based Simulation (IOS) with Predictable and Unpredictable Trigger Events in Simulated Time. Western New England University.
Dehghanimohammadabadi, M., Keyser, T. K., \& Cheraghi, S. H. (2017). A novel Iterative Optimization-based Simulation (IOS) framework: An effective tool
to optimize system's performance. Computers \& Industrial Engineering, 111(Supplement C), 1-17. doi:https://doi.org/10.1016/j.cie.2017.06.037
Della Croce, F., Grosso, A., \& Paschos, V. T. (2004). Lower bounds on the approximation ratios of leading heuristics for the single-machine total tardiness problem. Journal of Scheduling, 7(1), 85-91. doi:10.1023/B:JOSH.0000013056.09936.fd
Du, J., \& Leung, J. Y.-T. (1990). Minimizing total tardiness on one machine is NP-hard. Mathematics of operations research, 15(3), 483-495.
Fu, J. (2013). Logistics of Earthmoving Operations: Simulation and Optimization. KTH Royal Institute of Technology, Stockholm, Sweden.
Garg, A., \& Deshmukh, S. (2006). Maintenance management: literature review and directions. Journal of Quality in Maintenance Engineering, 12(3), 205238.

GDOT. (2013). Standard Specifications Construction of Transportation Systems Retrieved from http://www.dot.ga.gov/PartnerSmart/Business/Source/specs/DOT2013.pdf
González-Neira, E., Montoya-Torres, J., \& Barrera, D. (2017). Flow-shop scheduling problem under uncertainties: Review and trends. International Journal of Industrial Engineering Computations, 8(4), 399-426.
Gordon, V., Strusevich, V., \& Dolgui, A. (2012). Scheduling with due date assignment under special conditions on job processing. Journal of Scheduling, 15(4), 447-456. doi:10.1007/s10951-011-0240-2
Gourgand, M., Grangeon, N., \& Norre, S. (2000). A review of the static stochastic flow-shop scheduling problem. Journal of decision systems, 9(2), 1-31.
Hannan, E. L. (1981). Linear programming with multiple fuzzy goals. Fuzzy sets and systems, 6(3), 235-248.
Hastings, N. A. J. (2009). Physical asset management: Springer Science \& Business Media.
Holland, J. (1975). Adaption in natural and artificial systems. Ann Arbor MI: The University of Michigan Press.
Hsieh, Y. C., You, P. S., \& Liou, C. D. (2009). A note of using effective immune based approach for the flow shop scheduling with buffers. Applied Mathematics and Computation, 215(5), 1984-1989. doi:http://dx.doi.org/10.1016/j.amc.2009.07.033
Johnson, S. M. (1954). Optimal two-and three-stage production schedules with setup times included. Naval Research Logistics (NRL), 1(1), 61-68.
Knezevic, J. (1997). Systems Maintainability (Vol. 1): Springer Science \& Business Media.
Kothamasu, R., Huang, S. H., \& VerDuin, W. H. (2006). System health monitoring and prognostics-a review of current paradigms and practices. The International Journal of Advanced Manufacturing Technology, 28(910), 1012-1024.

Kubiak, W., Błażewicz, J., Formanowicz, P., Breit, J., \& Schmidt, G. (2002). Twomachine flow shops with limited machine availability. European Journal of

Operational Research, 136(3), 528-540.
doi:https://doi.org/10.1016/S0377-2217(01)00083-2
Kubzin, M. A., Potts, C. N., \& Strusevich, V. A. (2009). Approximation results for flow shop scheduling problems with machine availability constraints. Computers \& Operations Research, 36(2), 379-390. doi:https://doi.org/10.1016/j.cor.2007.10.013
Lee, C.-Y. (1997). Minimizing the makespan in the two-machine flowshop scheduling problem with an availability constraint. Operations Research Letters, 20(3), 129-139. doi:https://doi.org/10.1016/S0167-6377(96)000417
Lee, C.-Y. (1999). Two-machine flowshop scheduling with availability constraints. European Journal of Operational Research, 114(2), 420-429. doi:https://doi.org/10.1016/S0377-2217(97)00452-9
Lee, C. Y., \& Leon, V. J. (2001). Machine scheduling with a rate-modifying activity1. European Journal of Operational Research, 128(1), 119-128. doi:http://dx.doi.org/10.1016/S0377-2217(99)00066-1
Liang, T.-F. (2006). Distribution planning decisions using interactive fuzzy multiobjective linear programming. Fuzzy sets and systems, 157(10), 13031316.

Liang, T.-F. (2008). Fuzzy multi-objective production/distribution planning decisions with multi-product and multi-time period in a supply chain. Computers \& Industrial Engineering, 55(3), 676-694. doi:http://dx.doi.org/10.1016/j.cie.2008.02.008
Liang, T.-F. (2009). Fuzzy multi-objective project management decisions using two-phase fuzzy goal programming approach. Computers \& Industrial Engineering, 57(4), 1407-1416.
Luo, W., \& Ji, M. (2015). Scheduling a variable maintenance and linear deteriorating jobs on a single machine. Information Processing Letters, 115(1), 33-39. doi:10.1016/j.ipl.2014.08.011
Ma, Y., Chu, C., \& Zuo, C. (2010). A survey of scheduling with deterministic machine availability constraints. Computers and Industrial Engineering, 58(2), 199-211. doi:10.1016/j.cie.2009.04.014
Montgomery, D. C. (2008). Design and analysis of experiments: John Wiley \& Sons.
Mor, B., \& Mosheiov, G. (2012). Scheduling a maintenance activity and duewindow assignment based on common flow allowance. International Journal of Production Economics, 135(1), 222-230. doi:10.1016/j.ijpe.2011.07.013
Mosheiov, G., \& Sarig, A. (2009). Scheduling a maintenance activity and duewindow assignment on a single machine. Computers and Operations Research, 36(9), 2541-2545.
Naderi, B., Zandieh, M., \& Fatemi Ghomi, S. M. T. (2009). A study on integrating sequence dependent setup time flexible flow lines and preventive
maintenance scheduling. Journal of Intelligent Manufacturing, 20(6), 683694. doi:10.1007/s10845-008-0157-6

Norman, B. A. (1999). Scheduling flowshops with finite buffers and sequencedependent setup times. Computers \& Industrial Engineering, 36(1), 163177.

Norman, B. A., \& Bean, J. C. (1999). A Genetic Algorithm Methodology for Complex Scheduling Problems. Naval Research Logistics, 46(2), 199-211.
Nowicki, E. (1999). The permutation flow shop with buffers: A tabu search approach. European Journal of Operational Research, 116(1), 205-219. doi:http://dx.doi.org/10.1016/S0377-2217(98)00017-4
Paksoy, T., Pehlivan, N. Y., \& Ozceylan, E. (2012). Application of fuzzy optimization to a supply chain network design: A case study of an edible vegetable oils manufacturer. Applied Mathematical Modelling, 36(6), 27622776. doi:http://dx.doi.org/10.1016/j. apm.2011.09.060

Papadimitriou, C. H., \& Kanellakis, P. C. (1980). Flowshop scheduling with limited temporary storage. Journal of the ACM (JACM), 27(3), 533-549.
Pinedo, M. (2012). Scheduling: Springer.
Pinedo, M. L. (2012). Scheduling: theory, algorithms, and systems: Springer.
Reeves, C. R. (1995). A genetic algorithm for flowshop sequencing. Computers and Operations Research, 22(1), 5-13. doi:10.1016/0305-0548(93)E0014K

Reeves, C. R. (1997). Feature Article-Genetic Algorithms for the Operations Researcher. INFORMS journal on computing, 9(3), 231-250.
Rommelfanger, H. (1996). Fuzzy linear programming and applications. European journal of operational research, 92(3), 512-527.
Ronconi, D. P. (2004). A note on constructive heuristics for the flowshop problem with blocking. International Journal of Production Economics, 87(1), 39-48. doi:http://dx.doi.org/10.1016/S0925-5273(03)00065-3
Ruiz, R., Carlos García-Díaz, J., \& Maroto, C. (2007). Considering scheduling and preventive maintenance in the flowshop sequencing problem. Computers \& Operations Research, 34(11), 3314-3330. doi:http://dx.doi.org/10.1016/j.cor.2005.12.007
Sanlaville, E., \& Schmidt, G. (1998). Machine scheduling with availability constraints. Acta Informatica, 35(9), 795-811.
Schmidt, G. (2000). Scheduling with limited machine availability. European Journal of Operational Research, 121(1), 1-15. doi:10.1016/S0377-2217(98)00367-1
Seif, J., Yu, A. J., \& Rahmanniyay, F. (2017). Modelling and optimization of a biobjective flow shop scheduling with diverse maintenance requirements. International Journal of Production Research, 1-22. doi:10.1080/00207543.2017.1403660
Shi, X., \& Xu, D. (2014). Best possible approximation algorithms for single machine scheduling with increasing linear maintenance durations. The Scientific World Journal, 2014.

Smutnicki, C. (1998). A two-machine permutation flow shop scheduling problem with buffers. Operations-Research-Spektrum, 20(4), 229-235.
Sortrakul, N., \& Cassady, C. R. (2007). Genetic algorithms for total weighted expected tardiness integrated preventive maintenance planning and production scheduling for a single machine. Journal of Quality in Maintenance Engineering, 13(1), 49-61. doi:10.1108/13552510710735113
Sortrakul, N., Nachtmann, H. L., \& Cassady, C. R. (2005). Genetic algorithms for integrated preventive maintenance planning and production scheduling for a single machine. Computers in Industry, 56(2), 161-168. doi:10.1016/j.compind.2004.06.005
Stanciulescu, C. v., Fortemps, P., Installé, M., \& Wertz, V. (2003). Multiobjective fuzzy linear programming problems with fuzzy decision variables. European journal of operational research, 149(3), 654-675.
Sun, K., \& Li, H. (2010). Scheduling problems with multiple maintenance activities and non-preemptive jobs on two identical parallel machines. International Journal of Production Economics, 124(1), 151-158. doi:http://dx.doi.org/10.1016/ji.ipe.2009.10.018
Tansel, B. Ç., Kara, B. Y., \& Sabuncuoglu, I. (2001). An efficient algorithm for the single machine total tardiness problem. IIE Transactions, 33(8), 661-674. doi:10.1080/07408170108936862
Tate, D. M., \& Smith, A. E. (1993). Expected Allele Coverage and the Role of Mutation in Genetic Algorithms. Paper presented at the ICGA.
Trevino, V., \& Falciani, F. (2006). GALGO: an R package for multivariate variable selection using genetic algorithms. Bioinformatics, 22(9), 1154-1156. doi:10.1093/bioinformatics/btl074
Veldman, J., Wortmann, H., \& Klingenberg, W. (2011). Typology of condition based maintenance. Journal of Quality in Maintenance Engineering, 17(2), 183-202.
Wang, R.-C., \& Liang, T.-F. (2004). Application of fuzzy multi-objective linear programming to aggregate production planning. Computers \& Industrial Engineering, 46(1), 17-41. doi:http://dx.doi.org/10.1016/j.cie.2003.09.009
Wang, S., \& Liu, M. (2014). Two-machine flow shop scheduling integrated with preventive maintenance planning. International Journal of Systems Science, 1-19. doi:10.1080/00207721.2014.900137
Xiang, Y., Cassady, C. R., Jin, T., \& Zhang, C. W. (2014). Joint production and maintenance planning with machine deterioration and random yield. International Journal of Production Research, 52(6), 1644-1657.
Xiaodong, Y., Fernandez-Gaucherand, E., Fu, M. C., \& Marcus, S. I. (2004). Optimal preventive maintenance scheduling in semiconductor manufacturing. Semiconductor Manufacturing, IEEE Transactions on, 17(3), 345-356. doi:10.1109/TSM.2004.831948
Xu, D., Wan, L., Liu, A., \& Yang, D. L. (2015). Single machine total completion time scheduling problem with workload-dependent maintenance duration. Omega (United Kingdom), 52, 101-106. doi:10.1016/j.omega.2014.11.002

Xu, D., Yin, Y., \& Li, H. (2010). Scheduling jobs under increasing linear machine maintenance time. Journal of Scheduling, 13(4), 443-449. doi:10.1007/s10951-010-0182-0
Yamada, T., \& Nakano, R. (1997). Job shop scheduling. IEE control Engineering series, 134-134.
Yang, S.-J. (2013). Unrelated parallel-machine scheduling with deterioration effects and deteriorating multi-maintenance activities for minimizing the total completion time. Applied Mathematical Modelling, 37(5), 2995-3005. doi:http://dx.doi.org/10.1016/j.apm.2012.07.029
Yang, S. J. (2012). Single-machine scheduling problems simultaneously with deterioration and learning effects under deteriorating multi-maintenance activities consideration. Computers and Industrial Engineering, 62(1), 271275. doi:10.1016/j.cie.2011.09.014

Yang, S. J., \& Yang, D. L. (2010). Minimizing the makespan on single-machine scheduling with aging effect and variable maintenance activities. Omega, 38(6), 528-533. doi:10.1016/j.omega.2010.01.003
Ying, K. C. (2008). Solving non-permutation flowshop scheduling problems by an effective iterated greedy heuristic. International Journal of Advanced Manufacturing Technology, 38(3-4), 348-354. doi:10.1007/s00170-007-1104-y
Yip, H.-I., Fan, H., \& Chiang, Y.-h. (2014). Predicting the maintenance cost of construction equipment: Comparison between general regression neural network and Box-Jenkins time series models. Automation in Construction, 38(0), 30-38. doi:http://dx.doi.org/10.1016/j.autcon.2013.10.024
Yoo, J., \& Lee, I. S. (2016). Parallel machine scheduling with maintenance activities. Computers \& Industrial Engineering, 101(Supplement C), 361371. doi:https://doi.org/10.1016/j.cie.2016.09.020

Yu, A. J., \& Seif, J. (2016). Minimizing tardiness and maintenance costs in flow shop scheduling by a lower-bound-based GA. Computers \& Industrial Engineering, 97, 26-40.
Zadeh, L. A. (1965). Fuzzy sets. Information and control, 8(3), 338-353.
Zarook, Y., Rezaeian, J., Tavakkoli-Moghaddam, R., Mahdavi, I., \& Javadian, N. (2014). Minimization of makespan for the single batch-processing machine scheduling problem with considering aging effect and multi-maintenance activities. The International Journal of Advanced Manufacturing Technology, 1-14. doi:10.1007/s00170-014-6342-1
Zheng, Y., Lian, L., Fu, Z., \& Mesghouni, K. (2015). Evolutional algorithm in solving flexible job shop scheduling problem with uncertainties LISS 2013 (pp. 1009-1015): Springer.
Zimmermann, H.-J. (1978). Fuzzy programming and linear programming with several objective functions. Fuzzy sets and systems, 1(1), 45-55.

## APPENDICES

## Appendix A: Related Published Works

Seif, J., Dehghanimohammadabadi, M., \& Yu, A. J. (2018). Incorporating combined maintenance activities in flow shop scheduling: Applying stochastic programming and simulation optimization. Submitted to European Journal of Operational Research for publication.

Seif, J., \& Yu, A. J. (2018). The Integration of Operations, Maintenance and Workforce Planning. INFORMS International Conference, Taipei, Taiwan.

Seif, J., \& Yu, A. J. (2018). An extensive operations and maintenance planning problem with an efficient solution method. Computers and Operations Research.
DOI: 10.1016/j.cor.2018.03.010
Seif, J., Yu, A. J., \& Rahmanniyay, F. (2017). Modelling and optimization of a biobjective flow shop scheduling with diverse maintenance requirements. International Journal of Production Research.
DOI: 10.1080/00207543.2017.1403660
Yu, A. J., \& Seif, J. (2016). Minimizing tardiness and maintenance costs in flow shop scheduling by a lower-bound-based GA. Computers \& Industrial Engineering.
DOI: 10.1016/j.cie.2016.03.024

Appendix B: OPL Models Used in CPLEX Optimization Studio

## B.1: The Mixed-Integer Program in Chapter I

```
/*sets*/
int m=...; /*number of machines*/
int n=...; /*number of jobs*/
int l=...; /*number of maintenance levels or activities*/
float M1=...; /*the big M*/
float M2=...;
/*input parameters*/
float prcsTime[1..m][1..n]=...;
float dtrRate[1..m][1..n][1..l]=...;
float execTime[1..m][1..l]=...;
float MLmax[1..l]=...;
float spCost[1..m][1..l]=...;
float wfCost[1..l]=...;
float dueDate[1..n]=...;
float pnltCost[1..n]=...;
/*variables*/
dvar boolean x[1..n][1..n];
dvar boolean y[1..m][1..n][1..l];
dvar float ML[1..m][1..n][1..l];
dvar float+ completionTotal[1..n];
dvar float+ tardiness[1..n];
dvar float+ waitMachine[1..m][1..n];
dvar float+ waitJob[1..m][1..n];
dvar float+ PLT[1..n][1..n];
dvar int+ nbMA;
dvar int+ seq[1..n];
/*OF*/
minimize sum(i,j in 1..n) PLT[i][j] + sum(i in 1..m,j in 1..n, k in
1..l) y[i][j][k]*(spCost[i][k]+execTime[i][k]*wfCost[k]);
/*constraints*/
subject to {
    forall(i in 1..n) sum(j in 1..n) x[i][j]==1;
    forall(j in 1..n) sum(i in 1..n) x[i][j]==1;
    forall(i in 1..m) waitJob[i][1]==0;
    forall(j in 1..n) waitMachine[1][j] == 0;
    forall(i in 1..m,k in 1..l) ML[i][1][k]==MLmax[k];
    forall(i in 2..m) waitMachine[i][1]==sum(j in 1..n,k in 1..i-1)
x[j][1]*prcsTime[k][j]+sum(g in 1..l,k in 1..i-1)
y[k][1][g]*execTime[k][g];
    forall(i in 1..m-1,j in 1..n-1) waitMachine[i][j+1]+sum(k in
1..l) y[i][j+1][k]*execTime[i][k]+sum(jj in 1..n)
x[jj][j+1]*prcsTime[i][jj]+waitJob[i+1][j+1]
```

```
        == waitJob[i+1][j]+sum(k in 1..l)
y[i+1][j][k]*execTime[i+1][k]+sum(jj in 1..n)
x[jj][j]*prcsTime[i+1][jj]+waitMachine[i+1][j+1];
    forall(i in 1..m,j in 2..n,k in 1..l) ML[i][j][k]-MLmax[k] >= -
M1*(1-y[i][j][k]);
    forall(i in 1..m,j in 2..n,k in 1..l) ML[i][j][k]-MLmax[k] <=
M1*(1-y[i][j][k]);
    forall(i in 1..m,j in 2..n,k in 1..l) ML[i][j][k]-(ML[i][j-1][k]-
sum(jj in 1..n)x[jj][j-1]*prcsTime[i][jj]*dtrRate[i][jj][k]) >= -
M1*(y[i][j][k]);
    forall(i in 1..m,j in 2..n,k in 1..l) ML[i][j][k]-(ML[i][j-1][k]-
sum(jj in 1..n)x[jj][j-1]*prcsTime[i][jj]*dtrRate[i][jj][k]) <=
M1*(y[i][j][k]);
    forall(i in 1..m,j in 1..n,k in 1..l) ML[i][j][k] >= sum(jj in
1..n) x[jj][j]*prcsTime[i][jj]*dtrRate[i][jj][k];
    forall(j in 2..n) waitJob[1][j]==sum(r in 1..j-1,jj in
1..n)x[jj][r]*prcsTime[1][jj]+sum(r in 1..j-1,k in
1..l)y[1][r][k]*execTime[1][k];
    forall(j,k in 1..n) PLT[j][k]-pnltCost[j]*tardiness[k] >= -M2*(1-
x[j][k]);
    forall(j,k in 1..n) PLT[j][k]-pnltCost[j]*tardiness[k] <= M2*(1-
x[j][k]);
    forall(j,k in 1..n) PLT[j][k] >= -M2*x[j][k];
    forall(j,k in 1..n) PLT[j][k] <= M2*x[j][k];
    forall(j in 1..n) completionTotal[j]==sum(i in 1..m)
(waitJob[i][j]+sum(jj in 1..n)x[jj][j]*prcsTime[i][jj]+sum(k in 1..l)
y[i][j][k]*execTime[i][k]);
    forall(j in 1..n) tardiness[j] >= completionTotal[j]-sum(jj in
1..n) dueDate[jj]*x[jj][j];
    n.bMA==sum(i in 1..m,j in 1..n,k in 1..l) y[i][j][k];
    forall (i in 1..n) seq[i]==sum(j in 1..n) x[j,i]*j;
}
```


## B.2: The Mixed-Integer Program in Chapter II

```
/*sets*/
int m=...; /*number of machines*/
int n=...; /*number of jobs*/
int l=...; /*number of maintenance levels or activities*/
/*input parameters*/
float prcsTime[1..m][1..n]=...;
float dtrRate[1..m][1..n][1..l]=...;
float execTime[1..m][1..l]=...;
float MLmax[1..l]=...;
float spCost[1..m][1..l]=...;
float wfCost[1..l]=...;
float dueDate[1..n]=...;
float alpha11=...;
float alpha12=...;
float betta1=...;
float gamma1=...;
float alpha21=...;
float alpha22=...;
float betta2=...;
float gamma2=...;
float mem12=...;
float mem13=...;
float mem22=...;
float mem23=...;
float temp;
float M1=...; /*the big M*/
float M2=...;
/*variables* /
dvar boolean x[1..n][1..n];
dvar boolean y[1..m][1..n][1..l];
dvar float ML[1..m][1..n][1..l];
dvar float+ completionTotal[1..n];
dvar float+ tardiness[1..n];
dvar float+ waitMachine[1..m][1..n];
dvar float+ waitJob[1..m][1..n];
dvar float+ Rho[1..m][1..n];
dvar boolean L[1..m][1..n][1..3];
dvar boolean Z[1..m][1..n][1..n][1..3];
dvar float+ Gamma[1..m][1..n][1..n];
dvar int+ nbMA;
dvar float+ Tardiness;
dvar float+ MaintCost;
dvar int+ seq[1..n];
dvar float+ dl1;
dvar float+ d12;
dvar float+ d21;
dvar float+ d22;
dvar float+ e11;
dvar float+ e12;
```

```
dvar float+ e21;
dvar float+ e22;
dvar float+ S;
/*OF*/
maximize S;
/*Constraints*/
subject to {
    S<=-(alpha11)*(e11-d11)-(alpha12)*(e12-d12)+(betta1)*(sum( q in
1..n)tardiness[q])+(gamma1);
    S<=-(alpha21)*(e21-d21)-(alpha22)*(e22-d22)+(betta2)*(sum( i in
1..m,q in 1..n,k in
1..l)y[i][q][k]*(spCost[i][k]+execTime[i][k]*WfCost[k]))/1000+(gamma2);
    sum( q in 1..n) tardiness[q]+e11-d11==mem12;
    sum( q in 1..n) tardiness[q]+e12-d12==mem13;
    (sum( i in 1..m,q in 1..n,k in 1..l)
y[i][q][k]*(spCost[i][k]+execTime[i][k]*wfCost[k]))/1000+e21-
d21==mem22;
    (sum( i in 1..m,q in 1..n,k in 1..l)
y[i][q][k]*(spCost[i][k]+execTime[i][k]*wfCost[k]))/1000+e22-
d22==mem23;
    ct03: forall(q in 1..n) tardiness[q] >= completionTotal[q]-sum(j
in 1..n) dueDate[j]*x[j][q];
    ct05: forall(i in 1..n) sum(j in 1..n) x[i][j]==1;
    ct06: forall(j in 1..n) sum(i in 1..n) x[i][j]==1;
    ct07: forall(i in 1..m-1,q in 1..n-1) waitMachine[i][q+1]+sum(k
in 1..l) y[i][q+1][k]*execTime[i][k]+Rho[i][q+1]+waitJob[i+1][q+1]
                    == waitJob[i+1][q]+sum(k in 1..l)
y[i+1][q][k]*execTime[i+1][k]+Rho[i+1][q]+waitMachine[i+1][q+1];
    ct08: forall(i in 1..m,k in 1..l) ML[i][1][k]==MLmax[k];
    ct09: forall(i in 1..m) waitJob[i][1]==0;
    ct10: forall(j in 1..n) waitMachine[1][j] == 0;
    ct11: forall(i in 2..m) waitMachine[i][1]==sum(k in 1..i-1)
Rho[k][1];
    ct12: forall(q in 2..n) waitJob[1][q]==sum(r in 1..q-
1) Rho[1][r]+sum(r in 1..q,k in 1..l)y[1][r][k]*execTime[1][k];
    ct13: forall(i in 1..m,q in 1..n,k in 1..l) ML[i][q][k] >= sum(j
in 1..n) Gamma[i][j][q]*dtrRate[i][j][k];
    ct14: forall(i in 1..m,q in 2..n,k in 1..l) ML[i][q][k]-ML[i][q-
1][k]+sum(j in 1..n)Gamma[i][j][q-1]*dtrRate[i][j][k] >= -
M1*(y[i][q][k]);
    ct15: forall(i in 1..m,q in 2..n,k in 1..l) ML[i][q][k]-ML[i][q-
1][k]+sum(j in 1..n)Gamma[i][j][q-1]*dtrRate[i][j][k] <=
M1*(y[i][q][k]);
    ct16: forall(i in 1..m,q in 2..n,k in 1..l) ML[i][q][k]-MLmax[k]
>= -M1*(1-y[i][q][k]);
    ct17: forall(i in 1..m,q in 2..n,k in 1..l) ML[i][q][k]-MLmax[k]
<= M1*(1-y[i][q][k]);
    ct18: forall(i in 1..m,j,q in 1..n) Gamma[i][j][q]<=x[j][q]*M2;
    ct19: forall(i in 1..m,j,q in 1..n) Gamma[i][j][q]<=Rho[i][q];
```

```
    ct20: forall(i in 1..m,j,q in 1..n)
Gamma[i][j][q]>=Rho[i][q]+(x[j][q]-1)*M2;
    ct21: forall(q in 1..n) completionTotal[q]==sum(i in 1..m)
(waitJob[i][q]+Rho[i][q]+sum(k in 1..l) y[i][q][k]*execTime[i][k]);
        ct22: forall (i in 1..m,q in 1..n) Rho[i][q]==sum(j in
1..n)prcsTime[i][j]*((1.0*Z[i][j][q][1])+(1.5*Z[i][j][q][2])+(2.0*Z[i][
j][q][3]));
    ct23: forall (i in 1..m, j in 1..n, q in 1..n, h in 1..3)
Z[i][j][q][h]<=x[j][q];
    ct24: forall (i in 1..m, j in 1..n, q in 1..n, h in 1..3)
Z[i][j][q][h]<=L[i][q][h];
    ct25: forall (i in 1..m, j in 1..n, q in 1..n, h in 1..3)
Z[i][j][q][h]>=x[j][q]+L[i][q][h]-1;
    ct26: forall(i in 1..m, q in 1..n)
L[i][q][1]+L[i][q][2]+L[i][q][3]==1;
    ct27: forall(i in 1..m, q in 1..n) sum(k in
1..l)(ML[i][q][k]/(l*MLmax[k]))>=0.66*L[i][q][1]-
M2*(L[i][q][2]+L[i][q][3]);
    ct28: forall(i in 1..m, q in 1..n) sum(k in
1..l)(ML[i][q][k]/(l*MLmax[k]))<=0.66*L[i][q][2]+M2*(L[i][q][1]+L[i][q]
[3]);
    ct29: forall(i in 1..m, q in 1..n) sum(k in
1..l)(ML[i][q][k]/(l*MLmax[k]))}>=0.33*L[i][q][2]--
M2*(L[i][q][1]+L[i][q][3]);
    ct30: forall(i in 1..m, q in 1..n) sum(k in
1..l)(ML[i][q][k]/(l*MLmax[k]))<=0.33*L[i][q][3]+M2*(L[i][q][1]+L[i][q]
[2]);
    n.bMA==sum(i in 1..m,j in 1..n,k in 1..l) y[i][j][k];
    forall (i in 1..n) seq[i]==sum(j in 1..n) x[j,i]*j;
    Tardiness==sum(j in 1..n)tardiness[j];
    MaintCost==sum( i in 1..m,q in 1..n,k in 1..l)
y[i][q][k]*(spCost[i][k]+execTime[i][k]*wfCost[k]);
}
```


## B.3: The Stochastic Mixed-Integer Program in Chapter III

```
/*sets*/
int m=...; /*number of machines*/
int n=...; /*number of jobs*/
int l=...; /*number of maintenance levels or activities*/
int S=...; /*number of scenarios*/
int o=...; /*number of maintenance combinations*/
/*input parameters*/
float K=...; /*the big M*/
float probability[1..S]=...;
float p[1..m][1..n][1..S]=...;
float epsilon[1..m][1..o][1..S]=...; /*total executionTime for
each maintenance combination*/
float MLmax[1..m][1..l]=...;
float SPprime[1..m][1..o]=...; /*total of SpCost for each
maintenance combination*/
float WF=...;
float d[1..n]=...;
float pi[1..n]=...;
float a[1..l]=...;
float b[1..o]=...;
/*variables*/
dvar boolean x[1..n][1..n];
dvar boolean y[1..m][1..n][1..l];
dvar boolean PHI[1..m][1..n][1..o];
dvar float ML[1..m][1..n][1..l][1..S];
dvar float+ t[1..n][1..S];
dvar float+ Rho[1..m][1..n][1..S];
dvar boolean L[1..m][1..n][1..3][1..S];
dvar boolean u[1..m][1..n][1..n][1..3][1..S];
dvar float+ Gamma[1..m][1..n][1..n][1..S];
dvar float+ PI[1..n][1..n][1..S];
dvar int+ seq[1..n];
dvar int+ maintPos[1..m][1..n];
dvar float+ expectedTardiness;
dvar float+ expectedTardinessCost;
dvar float+ expectedMaintCost;
dvar float+ expectedTotalCost;
dvar float+ ST[1..m][1..n][1..S];
dvar float+ FT[1..m][1..n][1..S];
/*OF* /
minimize sum(s in 1..S)probability[s]*(sum(i,j in 1..n) PI[i][j][s] +
sum(i in 1..m,q in 1..n, r in 1..o)
PHI[i][q][r]*(SPprime[i][r]+epsilon[i][r][s]*WF));
/*constraints*/
subject to {
    ct04: forall(j in 1..n) sum(q in 1..n) x[j][q] == 1;
    ct05: forall(q in 1..n) sum(j in 1..n) x[j][q] == 1;
```

```
    /*start times of the first jobs*/
    ct06: forall(s in 1..S)
    ST[1][1][s] == 0;
    ct07: forall(i in 2..m,s in 1..S) ST[i][1][s]
== sum(ii in 1..i-1)Rho[ii][1][s];
    /*start times of the other jobs on the first machine*/
    ct08: forall(q in 2..n,s in 1..S)
                            ST[1][q][s]
== FT[1][q-1][s]+sum(r in 1..o)PHI[1][q][r]*epsilon[1][r][s];
    /*start time of the other jobs on the other machines*/
    ct09: forall(i in 2..m,q in 2..n,s in 1..S)
        ST[i][q][s]
>= FT[i][q-1][s] + sum(r in 1..o) PHI[i][q][r]*epsilon[i][r][s];
    ct10: forall(i in 2..m,q in 2..n,s in 1..S) ST[i][q][s]
>= FT[i-1][q][s];
    /*finish times*/
    ct11: forall (s in 1..S,q in 1..n,i in 1..m) FT[i][q][s] ==
ST[i][q][s] + Rho[i][q][s];
    ct12: forall(i in 1..m,k in 1..l,s in 1..S) ML[i][1][k][s] ==
MLmax[i][k];
    ct13: forall(i in 1..m,q in 1..n,k in 1..l,s in 1..S)
ML[i][q][k][s] >= sum(j in 1..n) Gamma[i][j][q][s];
    ct14: forall(i in 1..m,q in 2..n,k in 1..l,s in 1..S)
ML[i][q][k][s]-ML[i][q-1][k][s]+sum(j in 1..n)Gamma[i][j][q-1][s] >= -
K*(y[i][q][k]);
    ct15: forall(i in 1..m,q in 2..n,k in 1..l,s in 1..S)
ML[i][q][k][s]-ML[i][q-1][k][s]+sum(j in 1..n)Gamma[i][j][q-1][s] <=
K*(y[i][q][k]);
    ct16: forall(i in 1..m,q in 2..n,k in 1..l,s in 1..S)
ML[i][q][k][s]-MLmax[i][k] >= - K*(1-y[i][q][k]);
    ct17: forall(i in 1..m,q in 2..n,k in 1..l,s in 1..S)
ML[i][q][k][s]-MLmax[i][k] <= K*(1-y[i][q][k]);
    ct18: forall(q in 1..n,s in 1..S) t[q][s] >= FT[m][q][s]-sum(j in
1..n) d[j]*x[j][q];
    ct19: forall(j,q in 1..n,s in 1..S) PI[j][q][s]-pi[j]*t[q][s] >=
-K*(1-x[j][q]);
    ct20: forall(j,q in 1..n,s in 1..S) PI[j][q][s]-pi[j]*t[q][s] <=
K*(1-x[j][q]);
    ct21: forall(j,q in 1..n,s in 1..S) PI[j][q][s] >= -K*x[j][q];
    ct22: forall(j,q in 1..n,s in 1..S) PI[j][q][s] <= K*x[j][q];
    ct23: forall(i in 1..m,q in 1..n,s in 1..S) Rho[i][q][s]==sum(j
in 1..n)
p[i][j][s]*(1.0*u[i][j][q][1][s]+1.5*u[i][j][q][2][s]+2.0*u[i][j][q][3]
[s]);
    ct24: forall(i in 1..m, j in 1..n, q in 1..n, h in 1..3,s in
1..S) u[i][j][q][h][s]<=L[i][q][h][s];
    ct25: forall (i in 1..m, j in 1..n, q in 1..n, h in 1..3,s in
1..S) u[i][j][q][h][s]<=x[j][q];
    ct26: forall (i in 1..m, j in 1..n, q in 1..n, h in 1..3,s in
1..S) u[i][j][q][h][s]>=x[j][q]+L[i][q][h][s]-1;
    ct27: forall(i in 1..m, q in 1..n,s in 1..S)
L[i][q][1][s]+L[i][q][2][s]+L[i][q][3][s]==1;
    ct28: forall(i in 1..m, q in 1..n,s in 1..S) sum(k in
1..l)(ML[i][q][k][s]/(l*MLmax[i][k]))>=0.66*L[i][q][1][s]-
K*(L[i][q][2][s]+L[i][q][3][s]);
```

```
    ct29: forall(i in 1..m, q in 1..n,s in 1..S) sum(k in
1..l)(ML[i][q][k][s]/(l*MLmax[i][k]))<=0.66*L[i][q][2][s]+K*(L[i][q][1]
[s]+L[i][q][3][s]);
    ct30: forall(i in 1..m, q in 1..n,s in 1..S) sum(k in
1..l)(ML[i][q][k][s]/(l*MLmax[i][k]))>=0.33*L[i][q][2][s]-
K*(L[i][q][1][s]+L[i][q][3][s]);
    ct31: forall(i in 1..m, q in 1..n,s in 1..S) sum(k in
1..l)(ML[i][q][k][s]/(l*MLmax[i][k]))<=0.33*L[i][q][3][s]+K*(L[i][q][1]
[s]+L[i][q][2][s]);
    ct32: forall(i in 1..m,j,q in 1..n,s in 1..S)
Gamma[i][j][q][s]<=x[j][q]*K;
    ct33: forall(i in 1..m,j,q in 1..n,s in 1..S)
Gamma[i][j][q][s]<=Rho[i][q][s];
    ct34: forall(i in 1..m,j,q in 1..n,s in 1..S)
Gamma[i][j][q][s]>=Rho[i][q][s]+(x[j][q]-1)*K;
    ct35: forall(i in 1..m,q in 1..n,s in 1..S) sum(r in 1..o)
PHI[i][q][r] <= 1;
    ct36: forall(i in 1..m,q in 1..n,s in 1..S) sum(r in 1..o)
b[r]*PHI[i][q][r] == sum(k in 1..l) a[k]*y[i][q][k];
    /*sequence*/
    ct37: forall (q in 1..n) seq[q]==sum(j in 1..n) x[j,q]*j;
    /*maintenance positions*/
    ct38: forall (i in 1..m,q in 1..n) maintPos[i][q]==sum(r in 1..o)
PHI[i][q][r]*r;
    /*tardiness*/
    ct39: expectedTardiness == sum(j in 1..n,s in
1..S)t[j][s]*probability[s];
    /*costs*/
    ct40: expectedTardinessCost == sum(i,j in 1..n,s in 1..S)
PI[i][j][s]*probability[s];
    ct41: expectedMaintCost == sum(i in 1..m,q in 1..n,r in 1..o,s in
1..S)PHI[i][q][r]*(SPprime[i][r]+epsilon[i][r][s]*WF)*probability[s];
    ct42: expectedTotalCost == expectedTardinessCost +
expectedMaintCost;
}
```


## Appendix C: MATLAB Codes Used in Chapter I

## C.1: Test Problem Generator (CPLEX Data File Generator)

```
function [
MLmax,M2, prcsTime,dtrRate,execTime,spCost,wfCost,dueDate,pnltCost ] =
TPG( m,n,l )
close all;
clc;
%% Settings for Random Values
% processing times
PRmin=1;
PRmax=10;
% degradation rates
DGmin=0;
DGmax=2;
% spare parts costs
SPmin=1000;
SPmax=20000;
% workforce hojrly costs
WFmin=500;
WFmax=2000;
% penalty costs
PNmin=500;
PNmax=600;
% maintenance activity execution times
EXmin=1;
EXmax=4;
%% Matrices and Scalars Definition
MLmax=zeros(1,l);
M1=100000;
M2=100000;
prcsTime=zeros (m,n);
dtrRate=zeros(m,n,l);
execTime=zeros (m,l);
spCost=zeros (m,l);
wfCost=zeros (1, l);
dueDate=zeros (1,n);
pnltCost=zeros(1,n);
%% Generation of Matrices and Scalars
for i=1:m
    for j=1:n
        prcsTime(i,j)=randi([PRmin, PRmax],1,1);
    end
end
```

```
for j=1:n
    dueDate(1,j)=randi([10,30],1,1);
    pnltCost(1,j)=randi([PNmin,PNmax],1,1);
end
for i=1:m
    for j=1:n
        for k=1:l
            dtrRate(i,j,k)= DGmin+(DGmax-DGmin)*rand;
        end
    end
end
% dtrRate=randi([0,3],m,n,l);
%[DTR,R]=max(prcsTime);[DTR2,C]=max(DTR);%find maximum processing time
prcMax=max(max(prcsTime));
for k=1:l
    MLmax(1,k)= DGmax*prcMax;%1000;
    wfCost(1,k)=randi([WFmin,WFmax],1,1);
    for i=1:m
            execTime(i,k)=randi([EXmin,EXmax],1,1);
            spCost(i,k)=randi([SPmin,SPmax],1,1);
    end
end
M1=M1*max (max (MLmax));
M2 =M2*PRmax+EXmax*l*m*n;
%% Write into data file (CPLEX OPL .dat file)
fid=fopen(['CPLEX_DataFile_' num2str(m) '-' num2str(n) '-' num2str(l)
'___ ' datestr(now,'yyyy-mm-dd_HH-MM-SS') '.dat'],'w' );
    fprintf(fid,['m=' num2str(m) ';' '\r\n']);
    fprintf(fid,['n=' num2str(n) ';' '\r\n']);
    fprintf(fid,['l=' num2str(l) ';' '\r\n']);
    fprintf(fid,['M1=' num2str(M1) ';' '\r\n']);
    fprintf(fid,['M2=' num2str(M2) ';' '\r\n']);
    STR='MLmax=';
    STR=[STR '[' num2str(MLmax(1,1))];
    for k=2:l
        STR=[STR ',' num2str(MLmax(1,k))];
    end
    STR=[STR '];' '\r\n'];
    fprintf(fid,STR);
    STR='wfCost=';
    STR=[STR '[' num2str(wfCost(1,1))];
    for k=2:l
        STR=[STR ',' num2str(wfCost(1,k))];
    end
    STR=[STR '];' '\r\n'];
    fprintf(fid,STR);
```

```
STR='dueDate=';
STR=[STR '[' num2str(dueDate(1,1))];
for j=2:n
    STR=[STR ',' num2str(dueDate(1,j))];
end
STR=[STR '];' '\r\n'];
fprintf(fid,STR);
STR='pnltCost=';
STR=[STR '[' num2str(pnltCost(1,1))];
for j=2:n
    STR=[STR ',' num2str(pnltCost(1,j))];
end
STR=[STR '];' '\r\n'];
fprintf(fid,STR);
% prcsTime
STR='prcsTime=[';
for i=1:m
STR=[STR '[' num2str(prcsTime(i,1))];
for j=2:n
        STR=[STR ',' num2str(prcsTime(i,j))];
end
STR=[STR '],' '\r\n'];
end
STR=[STR '];' '\r\n'];
fprintf(fid,STR);
% execTime
STR='execTime=[';
for i=1:m
STR=[STR '[' num2str(execTime(i,1))];
for k=2:l
    STR=[STR ',' num2str(execTime(i,k))];
end
STR=[STR '],' '\r\n'];
end
STR=[STR '];' '\r\n'];
fprintf(fid,STR);
% spCost
STR='spCost=[';
for i=1:m
STR=[STR '[' num2str(spCost(i,1))];
for k=2:l
    STR=[STR ',' num2str(spCost(i,k))];
end
STR=[STR '],' '\r\n'];
end
STR=[STR '];' '\r\n'];
fprintf(fid,STR);
```

```
    % dtrRate
    STR='dtrRate=[';
    for i=1:m
    STR=[STR '['];
    for j=1:n
    STR=[STR '[' num2str(dtrRate(i,j,1))];
    for k=2:l
        STR=[STR ',' num2str(dtrRate(i,j,k))];
    end
    STR=[STR '],' '\r\n'];
    end
    STR=[STR '],' '\r\n'];
    end
    STR=[STR '];' '\r\n'];
    fprintf(fid,STR);
fprintf(fid,' \r\n');
fclose('all');
```


## C.2: Objective Function (Fitness Function)

```
function [ y,z ] = SolnCalc( n,m,l,mlx,prc,dtr,exc,spc,wfc,due,plt,seq )
completion=zeros(m,n); % completion time after processing by a machine
tardiness=zeros(1,n);
y=zeros(m,n,l); % MA Positions
ML=zeros(m,n,l);
waitJob=zeros(m,n);
% start calculations.
% IMPORTANT NOTE: I do not set zero values because I already set them
through "zeros" method!
% MLs are at their maximum level at the beginning
for k=1:l
    ML(:,1,k)=mlx(1,k);
end
% ML after first job
for i=1:m
    for j=2:n
        for k=1:l
                ML(i,j,k)=ML(i,j-1,k)-prc(i,seq(1,j-1))*dtr(i,seq(1,j-1),k);
                if ML(i,j,k)<prc(i,seq(1,j))*dtr(i,seq(1,j),k)
                    ML(i,j,k)=mlx(1,k);
                    y(i,j,k)=1;
                end
            end
    end
end
% display(ML);
% completion times of first job
for i=1:m
    completion(i,1)=sum(prc(1:i,seq(1,1)));
end
for j=2:n
    completion(1,j)=completion(1,j-1)+prc(1,seq(1,j));
    for k=1:l
        completion(1,j)=completion(1,j)+exc(1,k)*y(1,j,k);
    end
end
% all completion times for n>2, m>2
for i=2:m
    for j=2:n
        if completion(i,j-1)>completion(i-1,j)
            waitJob(i,j)=completion(i,j-1)-completion(i-1,j);
        end
        completion(i,j)=completion(i-1,j)+waitJob(i,j)+prc(i,seq(1,j));
        for k=1:l
```

```
                completion(i,j)=completion(i,j)+exc(i,k)*y(i,j,k);
                end
    end
end
% tardiness calculation
for j=1:n
    if completion(m,j)>due(1,seq(1,j))
                tardiness (1,j)=completion(m,j)-due (1,seq(1,j));
    end
end
% calculation of total cost
z=0;
    % penalty
    for j=1:n
        z=z+tardiness(1,j)*plt(1,seq(1,j));
    end
    % maintenance costs
    for i=1:m
        for j=1:n
            for k=1:l
                        z=z+y(i,j,k)*(wfc(i,k)*exc(i,k)+spc(i,k));
                end
            end
        end
N=sum(y);
end
```


## C.3: Mutation Function

```
function z=GAMutate(x,mu,n)
p=ceil(n/5);
y=x;
for k=1:p
    r=rand;
    h=y;
    if mu<r
                j=randsample(n,1); % Selected Points to be Mutated
                i=randsample(n,1);
                h(1,i)=y(1,j);
                h(1,j)=y(1,i);
    end
    y=h;
end
z=h;
end
```


## C.4: Crossover Function

```
function [y1,y2]=GACrossover(x1,x2)
% Single-Point Crossover
n=numel(x1);
c=randi([1 n-1]);
y1=[x1(1:c) x2(c+1:end)];
y2=[x2(1:c) x1(c+1:end)];
% corrections
for i=1:n-c
    if isempty(find(y1(1:c+i-1)==y1(c+i)))==0
        for j=1:n
            if isempty(find(y1(1:c+i-1)==x2(j)))==1
                y1(c+i)=x2(j);
                break
                    end
        end
    end
end
for i=1:n-c
    if isempty(find(y2(1:c+i-1)==y2(c+i)))==0
        for j=1:n
            if isempty(find(y2(1:c+i-1)==x1(j)))==1
                y2(c+i)=x1(j);
                break
            end
        end
    end
end
end
```


## C.5: The Roulette Wheel Selection Function

```
function i=GARouletteWheelSelction(p)
    r=rand;
    c=cumsum(p);
    i=find(r<=c,1,'first');
end
```


## C.6: The GA

```
clc;
clear;
close all;
%% Problem Definition
m=7; % no. of machines
n=10; % no. of jobs
l=5; % no. of MLs
ctrl=200; % interations before convergence
VarSize=n; % Decision Variables Matrix Size
ML=zeros(m,n,l);
[ MLmax,M2,prcsTime,dtrRate,execTime,spCost,wfCost,dueDate,pnltCost ] =
TPG( m,n,l );
PreviousBest=-inf;
Counter=0;
%% GA Parameters
MaxIt=100; % Maximum Number of Iterations
nPop=200; % Population Size
pCrossover=0.8; % Crossover Percentage
nCrossover=round(pCrossover*nPop/2)*2; % Number of Parents (Offsprings)
pMutation=0.8; % Mutation Percentage
nMutation=round(pMutation*nPop); % Number of Mutants
mu=0.8; % Mutation rate/prob
SelectionPressure=8; % Selection Pressure
pause(0.01);
%% Initialization
tic
empty_individual.Sequence=[];
empty_individual.Cost=[];
empty_individual.MA=[];
```

```
pop=repmat(empty_individual,nPop,1);
% First Generation
for i=1:nPop
    % Create Random Solution
    pop(i).Sequence=randperm(n);
    % Evalute Newly Created Solution
    [pop(i).MA,pop(i).Cost]=SolnCalc(
n,m,l,MLmax, prcsTime, dtrRate, execTime, spCost,wfCost,dueDate, pnltCost, pop
(i).Sequence );
end
% Sort Population
Costs=[pop.Cost];
[Costs, SortOrder]=sort(Costs);
pop=pop(SortOrder);
% Store Best Solution
BestSol=pop(1);
% Update Worst Cost
WorstCost=max(Costs);
% Array To Hold Best Cost Values
BestCost=zeros(MaxIt,1);
%% GA Main Loop
it=1;
while Counter<ctrl
    % Calculate Selection Probabilities
    p=exp(-SelectionPressure*Costs/WorstCost);
    p=p/sum(p);
    % Crossover
    popc=repmat(empty_individual,nCrossover/2,2);
    for k=1:nCrossover/2
        i1=GARouletteWheelSelction(p);
        i2=GARouletteWheelSelction(p);
        p1=pop(i1);
        p2=pop(i2);
        [popc(k,1).Sequence,
popc(k,2).Sequence]=GACrossover(p1.Sequence, p2. Sequence);
```

```
        % Evaluatioan of children
        [popc(k,1).MA,popc(k,1).Cost]=SolnCalc(
n,m,l,MLmax,prcsTime,dtrRate,execTime,spCost,wfCost,dueDate,pnltCost,pop
c(k,1).Sequence);
    [popc(k,2).MA,popc(k,2).Cost]=SolnCalc(
n,m,l,MLmax,prcsTime,dtrRate,execTime,spCost,wfCost,dueDate,pnltCost,pop
c(k,2).Sequence);
    end
    popc=popc(:);
    % Mutation
    popm=repmat(empty_individual,nMutation,1);
    for k=1: nMutation
        i=GARouletteWheelSelction(p);
        pp=pop(i);
        popm(k).Sequence=GAMutate(pp.Sequence,mu,n);
        % Pst-mutation evaluation
        [popc(k).MA,popm(k).Cost]=SolnCalc(
n,m,l,MLmax,prcsTime,dtrRate, execTime, spCost,wfCost,dueDate,pnltCost,pop
c(k).Sequence);
end
% Merge
pop= [pop
            popc
            popm]; %#ok
% Sort Population
Costs=[pop.Cost];
[Costs,SortOrder]=sort(Costs);
pop=pop(SortOrder);
% Delete Extra Individuals
pop=pop(1:nPop);
Costs=Costs(1:nPop);
% Store Best Solution
BestSol=pop(1);
display(BestSol.Sequence); display(BestSol.Cost); display(BestSol.MA);
% Update Worst Cost
WorstCost=max(WorstCost,max(Costs));
% Store Best Cost
BestCost(it)=BestSol.Cost;
% Display Iteration Information
```

```
    disp(['Iteration ' num2str(it) ': Best Cost = '
num2str(BestCost(it))]);
    % Stoping Condition
    CurrentBest=BestCost(it);
    if CurrentBest==PreviousBest
    Counter=Counter+1;
    end
    if CurrentBest<PreviousBest
    Counter=0;
    end
    if Counter==ctrl
    disp(['number of interations before convergence is: ' num2str(it-
ctrl) ' iteraion(s) = ']);
        end
    PreviousBest=CurrentBest;
    it=it+1;
end
%% Results
figure;
plot(BestCost,'LineWidth',2);
toc
iteration=it-ctrl;
cost=CurrentBest;
% end
```


## Appendix D: MATLAB Codes Used in Chapter II

## D.1: Test Problem Generator Function

```
function [
MLmax,M1,M2,prcsTime,dtrRate,execTime,spCost,wfCost,dueDate,pnltCost ] =
TPG( m,n,l,tp )
close all;
clc;
%% Settings for Random Values
% processing times
PRmin=1;
PRmax=10;
% degradation rates
DGmin=0;
DGmax=2;
% spare parts costs
SPmin=1000;
SPmax=20000;
% workforce hojrly costs
WFmin=500;
WFmax=2000;
% penalty costs
PNmin=500;
PNmax=600;
% maintenance activity execution times
EXmin=1;
EXmax=4;
%% Matrices and Scalars Definition
MLmax=zeros(1,l);
M1=100000;
M2=100000;
prcsTime=zeros(m,n);
dtrRate=zeros(m,n,l);
execTime=zeros(m,l);
spCost=zeros(m,l);
wfCost=zeros(1,l);
dueDate=zeros(1,n);
pnltCost=zeros(1,n);
%% Generation of Matrices and Scalars
for i=1:m
    for j=1:n
        prcsTime(i,j)=randi([PRmin,PRmax],1,1);
    end
end
```

```
    for j=1:n
    dueDate(1,j)=randi([10, 30],1,1);
    pnltCost(1,j)=randi([PNmin, PNmax],1,1);
end
for i=1:m
    for j=1:n
            for k=1:l
                dtrRate(i,j,k)= DGmin+(DGmax-DGmin)*rand;
            end
    end
end
prcMax=max(max(prcsTime));
for k=1:l
    MLmax(1,k)= DGmax*prcMax;%1000;
    wfCost(1,k)=randi([WFmin,WFmax],1,1);
    for i=1:m
        execTime(i,k)=randi([EXmin,EXmax],1,1);
        spCost(i,k)=randi([SPmin,SPmax],1,1);
    end
end
M1=M1*max(max(MLmax));
M2 =M2 * PRmax +EXmax* l *m*n;
```


## D.2: Updating the Processing Times

```
function [ prc ] = updateProcTime( prc,ml )
%
    if ml>=0.66
        prc=prc*1.0;
    end
    if ml<0.66 && ml>=0.33
        prc=prc*1.5;
    end
    if ml<0.33
        prc=prc*2.0;
    end
end
```


## D.3: Crossover Function

```
function [y1,y2]=GACrossover(x1,x2)
% Single-Point Crossover
display(x1);display(x2);
n=numel(x1);
c=randi([1 n-1]);
y1=[x1(1:c) x2(c+1:end)];
y2=[x2(1:c) x1(c+1:end)];
% corrections
for i=1:n-c
    if isempty(find(y1(1:c+i-1)==y1(c+i)))==0
            for j=1:n
                if isempty(find(y1(1:c+i-1)==x2(j)))==1
                y1(c+i)=x2(j);
                break
                end
            end
    end
end
for i=1:n-c
    if isempty(find(y2(1:c+i-1)==y2(c+i)))==0
        for j=1:n
                    if isempty(find(y2(1:c+i-1)==x1(j)))==1
                y2(c+i)=x1(j);
                break
            end
        end
    end
end
end
```


## D.4: Mutation Function

```
function x=GAMutate2(x,mu,n)
p=ceil(n/5);
for k=1:p
    r=rand;
    el=randi(n);
    if mu<r
        x(el)=1-x(el);
    end
end
end
```

Mutation Function of ALG

```
function x=ALGMutate(x,ML,mu,n,i,kk,l,mlx,prc,seq,exc)
for q=1:n-1
    r=rand;
    if mu<r && x(q)==0
        if 1>0
            x(q)=1;
            end
        end
end
end
```


## D.5: Roulette Wheel Selection Function

```
function i=GARouletteWheelSelction(p)
    display(p);
    r=rand;
    c=cumsum(p);
    i=find(r<=c,1,'first');
end
```


## D.6: Fitness Function

```
function [ minL,Tardiness,MaintCost,isFeas,ML] = evalSol(
n,m,l,mlx,prc,dtr,exc,spc,wfc,due,seq,MA,Mem )
% Note: Here we use MA instead of y
display('=====');
display(MA);
completion=zeros(m,n); % completion time after processing by a machine
tardiness=zeros(1,n);
ML=zeros(m,n,l);
waitJob=zeros(m,n);
waitMac=zeros(m,n);
javad=prc;
% start calculations.
% IMPORTANT NOTE: I do not set zero values because I already set them
through "zeros" method!
% MLs are at their maximum level at the beginning
for k=1:l
    ML (:, 1,k)=mlx(1,k);
end
% ML after first job
ml=0;
isFeas=1;
for i=1:m
    if isFeas==1
        for q=2:n
            j=seq(1,q);
            jj=seq(1,q-1);
            for k=1:l
                if MA(i,q,k)==1
                    ML(i,q,k)=mlx(1,k);
                else
                    ML(i,q,k)=ML(i,q-1,k)-prc(i,jj)*dtr(i,jj,k);
                end
            end
            %display(q);display(ML);
            % update MLs
            if nnz (ML<0)==0
                for k=1:l
                    ml=ml+(ML(i,q,k)/mlx(1,k))/l;
                    end
                % update processing times
                prc(i,j)=updateProcTime(javad(i,j),ml);
                ml=0;
                if nnz(ML(i,q,:)<prc(i,j)*dtr(i,j,:))>0
                        isFeas=0;
                break
                end
            else
```

```
                    isFeas=0;
                    break
                end
        end
    end
end
% display(ML);
if isFeas==1
    % completion times of first job
    for i=1:m
        completion(i,1)=sum(prc(1:i,seq(1,1)));
    end
    % waitJob for the first machine
    for q=2:n
        for r=2:q
            for k=1:l
                    waitJob(1,q)=waitJob(1,q) +exc(1,k) *MA(1,r,k);
                end
            end
            for r=1:q-1
                waitJob(1,q)=waitJob(1,q)+prc(1,seq(1,r));
            end
    end
    % waitMac
    for i=2:m
        waitMac(i,1)=sum(prc(1:i-1,seq(1,1)));
    end
    % completion times of the other jobs after the first machine
    for q=2:n
        completion(1,q)=waitJob(1,q)+prc(1,seq(1,q));
    end
    % all completion times for n>2, m>2
    for i=2:m
        for q=2:n
            if completion(i,q-1)>completion(i-1,q)
                waitJob(i,q)=completion(i,q-1)-completion(i-1,q);
            elseif completion(i-1,q)>completion(i,q-1)
                waitMac(i,q)=completion(i-1,q) -completion(i,q-1);
            end
                completion(i,q) =completion(i-
1,q) +waitJob(i,q) +prc(i,seq(1,q));
                for k=1:l
                    completion(i,q)=completion(i,q) +exc(i,k) *MA(i,q,k);
                end
            end
    end
    % new completion time
    CompTime=zeros(1,n);
    for q=1:n
        CompTime(q) =sum(waitJob(:,q)) +sum(prc(:,seq(1,q)));
        for k=1:l
            for i=1:m
                    CompTime(q) =CompTime(q) +MA(i,q,k)*exc(i,k);
            end
```

```
        end
    end
        % tardiness calculation
    for q=1:n
        if CompTime(q)>due(1,seq(1,q))
                tardiness(1,q)=CompTime(q) -due(1,seq(1,q));
        end
    end
    % total tardiness (OFV1)
    Tardiness=sum(tardiness);
    %total cost of maintenance (OFV2)
    MaintCost=0;
    for i=1:m
        for q=1:n
                for k=1:l
MaintCost=MaintCost+MA(i,q,k)*(wfc(1,k)*exc(i,k)+spc(i,k));
                end
        end
    end
    %% Satisfaction
    n=size(Mem(1,:,1),2);
    if n>1
        [alpha11,alpha12,betta1,gamma1,alpha21,alpha22,betta2,gamma2] =
getCoef( Mem );
        D=zeros(2,2);
        OFV=[Tardiness MaintCost/1000];
        for g=1:2
            for e=1:2
                    D (g,e)=abs(Mem(g,e+1,1) -OFV(g));
                end
        end
        L=zeros(1,2);
        L(1)=-alpha11*D(1,1) -alpha12*D(1, 2) +betta1*OFV (1) +gamma1;
        L(2)=-alpha21*D(2,1)-alpha22*D (2,2) +betta2*OFV (2) +gamma2;
        %display(L);
        if nnz(L<0)>0
            minL=-0.0001/sum(OFV);
        else
            minL=-min(L);
        end
    end
else
    minL=0;
    Tardiness=inf;
```

MaintCost=inf;
end
end

## D.7: Function for Calculating the Coefficients of the iFMOLP

```
function [Q11,Q12,l1,s,Q21,Q22,u2,s2]=getCoef(M)
n=size(M(1,:,1),2);
% input: membership matrix
q=zeros(1,n-2);
Z1=M(1,:,1);%example: [90,85,80,76]
g1=M(1,:,2);%example: [0,0.5,0.8,1]
t=zeros(1,n-1);
for i=1:n-1
    t(i)=(g1(i+1)-g1(i))/(Z1(i+1)-Z1(i));
end
for j=1:n-2
    q(j)=(t(j+1)-t(j))/2;
end
l1=(t(3)+t(1))/2;
%%%for bn##
m=(1-g1(n-1))/(Z1(n)-Z1(n-1));
bn=g1(n)-Z1(n)*m;
%%%for b1%%%
m2=(g1(2)-0)/(Z1(2)-Z1(1));
b1=-m2*Z1(1);
s=(b1+bn)/2;
Q11=q(1);
Q12=q(2);
%%%%For the 2nd 0.bj%%%%%%%%%%%%%%%%%%%%%%
Z2=M(2,:,1);%[120,118,110,105]
g2=M(2,:,2);%[0,0.5,0.8,1]
t2=zeros(1,n-1);
for i=1:n-1
    t2(i)=(g2(i+1)-g2(i))/(Z2(i+1)-Z2(i));
end
for i=1:n-2
    q2(i)=(t2(i+1)-t2(i))/2;
end
u2=(t2(3)+t2(1))/2;
%%%for bn##
L=(1-g2(n-1))/(Z2(n)-Z2(n-1));
pn=g2(n)-Z2(n)*L;
%%%for b1%%%
L2=(g2(2)-0)/(Z2(2)-Z2(1));
p1=-L2*Z2(1);
s2=(p1+pn)/2;
Q21=q2(1);
Q22=q2(2);
end
```


## D.8: Algorithm 2

```
function [ MA ] = getMinMA( n,m,l,mlx,prc,dtr,exc,spc,wfc,due,seq )
MA = zeros(m,n,l);
ML = zeros(m,n,l);
javad = prc;
for k=1:l
        ML (:,1,k)=mlx(1,k);
end
ml=0;
for i=1:m
    for q=2:n
        j = seq(1,q);
        jj = seq(1,q-1);
            for k=1:l
                ML(i,q,k) = ML(i,q-1,k) - prc(i,jj)*dtr(i,jj,k);
            end
            for k=1:1
                ml=ml+(ML(i,q,k)/mlx(1,k))/l;
            end
            % update processing times
            prc(i,j)=updateProcTime(javad(i,j),ml);
            ml=0;
            for k=1:l
                if ML(i,q,k) - prc(i,j)*dtr(i,j,k)<0
                    ML (i,q,k)=mlx(1,k);
                    MA(i,q,k)=1;
                else
                        MA(i,q,k)=0;
                end
            end
        end
end
end
```


## D.9: The GAs

## Standard GA

function [ GA_OFV,GA_Time,GA_Iter,bestTard,bestMaint ] = solveGA( $m, n, l, M L m a x, M 2, p r c s T i m e, d t r R a t e, e x e c T i m e, s p C o s t, w f C o s t, d u e D a t e, p n l t C o s t$, Mem )
\% Initiate
VarSize=n; \% Decision Variables Matrix Size
ctrl=20; \% interations for convergence
PreviousBest=-inf;
Counter=0;
\% \% GA Parameters

MaxIt=100; \% Maximum Number of Iterations
nPop=500; \% Population Size
pCrossover=0.3; Crossover Percentage
nCrossover=round (pCrossover*nPop/2)*2; \% Number of Parents (Offsprings)
pMutation=0.3; \% Mutation Percentage
nMutation=round(pMutation*nPop); \% Number of Mutants

```
mu=0.5; % Mutation rate/prob
mu2=0.5; % Mutation rate/prob for MAs
```

SelectionPressure=8; \% Selection Pressure
pause(0.01);
\%\% Initialization
tic
empty_individual.Sequence=[];
empty_individual.Cost=[];
empty_individual. $\mathrm{MA}=\mathrm{zeros}(\mathrm{m}, \mathrm{n}, \mathrm{l})$;
empty_individual. $\mathrm{ML}=\operatorname{zeros}(\mathrm{m}, \mathrm{n}, \mathrm{l})$;
empty_individual.Tardiness=[];
empty_individual.MaintCost=[];
empty_individual.isFeas=0;
pop=repmat(empty_individual,nPop,1);
\% First Generation

```
% the randomly generated sequences should be unique!
poppy = randperm(factorial(n),nPop);
for i=1:nPop
    % Create Random Solution
    perm = perms_m(n,poppy(i)-1);
    pop(i).Sequence = perm';
% pop(i).Sequence=randperm(n);
    pop(i).MA(:, 2:n,:)=randi([0,1],m,n-1,l);
    % Evalute Newly Created Solution
[pop(i).Cost,pop(i).Tardiness,pop(i).MaintCost,pop(i).isFeas,pop(i).ML]=
evalSol(
n,m,l,MLmax,prcsTime,dtrRate, execTime, spCost,wfCost,dueDate,pop(i).Seque
nce,pop(i).MA,Mem );
    display(i);
end
% Sort Population
Costs=[pop.Cost];
[Costs, SortOrder]=sort(Costs);
pop=pop(SortOrder);
% Store Best Solution
BestSol=pop(1);
% Update Worst Cost
WorstCost=max(Costs);
% Array To Hold Best Cost Values
BestCost=zeros(MaxIt,1);
%% GA Main Loop
it=1;
while Counter<ctrl
    % Calculate Selection Probabilities
    p=-Costs+1;%exp(-SelectionPressure*Costs/WorstCost) ;
    p=p/sum (p) ;
    % Crossover
    popc=repmat(empty_individual,nCrossover/2,2);
    for k=1:nCrossover/2
        i1=GARouletteWheelSelction(p);
        i2=GARouletteWheelSelction(p);
% i2=randi(nPop);
        p1=pop(i1);
        p2=pop(i2);
```

```
        [popc(k,1).Sequence,
popc(k,2).Sequence]=GACrossover(p1.Sequence, p2.Sequence);
        for i=1:m
            for j=1:l
                [popc(k,1).MA(i,:,j),
popc(k,2).MA(i,:,j)]=GACrossover(p1.MA(i,:,j),p2.MA(i,:,j));
            end
        end
        % Evaluatioan of children
[popc(k,1).Cost,popc(k,1).Tardiness,popc(k,1).MaintCost,popc(k,1).isFeas
,popc(k,1).ML]=evalSol(
n,m,l,MLmax,prcsTime,dtrRate,execTime,spCost,wfCost,dueDate,popc(k,1).Se
quence,popc(k,1).MA,Mem );
[popc(k,2).Cost,popc(k,2).Tardiness,popc(k,2).MaintCost,popc(k,1).isFeas
,popc(k,1).ML]=evalSol(
n,m,l,MLmax,prcsTime,dtrRate,execTime,spCost,wfCost,dueDate,popc(k,2).Se
quence,popc(k,2).MA,Mem );
    end
    popc=popc(:);
    % Mutation
    popm=repmat(empty_individual,nMutation,1);
    for k=1: nMutation
        i=GARouletteWheelSelction(p);
        pp=pop(i);
        popm(k).Sequence=pp.Sequence;
        for i=1:m
            for j=1:l
                popm(k).MA(i, 2:n,j)=GAMutate2(pp.MA(i,2:n,j),mu2,n-1);
            end
        end
        % Pst-mutation evaluation
[popm(k).Cost,popm(k).Tardiness,popm(k).MaintCost,pop(i).isFeas,popc(k,1
).ML]=evalSol(
n,m,l,MLmax,prcsTime,dtrRate,execTime,spCost,wfCost,dueDate,popm(k).Sequ
ence,popm(k).MA,Mem );
    end
    % Merge
    pop=[pop
        popc
        popm];
```

```
    % Sort Population
    Costs=[pop.Cost];
    [Costs,SortOrder]=sort(Costs);
    pop=pop(SortOrder);
    % Delete Extra Individuals
    pop=pop(1:nPop);
    Costs=Costs (1:nPop);
    % Store Best Solution
    BestSol=pop(1);
display(BestSol.Sequence);display(BestSol.Cost);display(BestSol.MA);disp
lay(BestSol.Tardiness);display(BestSol.MaintCost);
    % Update Worst Cost
    WorstCost=max(WorstCost,max(Costs));
    % Store Best Cost
    BestCost(it)=BestSol.Cost;
    bestTard = BestSol.Tardiness;
    bestMaint = BestSol.MaintCost;
    % Display Iteration Information
    disp(['Iteration ' num2str(it) ': Best Cost = '
num2str(BestCost(it))]);
    % Stoping Condition
    CurrentBest=BestCost(it);
    if CurrentBest==PreviousBest
    Counter=Counter+1;
    end
    if CurrentBest<PreviousBest
    Counter=0;
    end
    if Counter==ctrl
        disp(['number of interations before convergence is: ' num2str(it-
ctrl) ' iteraion(s) = ']);
            end
        PreviousBest=CurrentBest;
        it=it+1;
end
%% Results
% figure;
% plot(BestCost,'LineWidth',2);
GA_Time = toc;
GA_Iter = it-ctrl;
GA_OFV = CurrentBest;
en\overline{d}
```


## $\underline{\text { Lower Bound for } Z_{1}}$

```
function [ GA OFV,GA Time,GA Iter ] = solveLBZ1(
m,n,l,MLmax,M\overline{2},prcsTime, dtrRäte, execTime, spCost,wfCost,dueDate,pnltCost,
Mem )
% this GA favors Z1; has ALG structure
% are blindly applied to S and y.
%% Initiate
VarSize=n; % Decision Variables Matrix Size
ctrl=20; % interations for convergence
PreviousBest=Inf;
Counter=0;
%% GA Parameters
MaxIt=100; % Maximum Number of Iterations
nPop=500; % Population Size
pCrossover=0.3; % Crossover Percentage
nCrossover=round(pCrossover*nPop/2)*2; % Number of Parents (Offsprings)
pMutation=0.3; % Mutation Percentage
nMutation=round(pMutation*nPop); % Number of Mutants
mu=0.5; % Mutation rate/prob
mu2=0.1; % Mutation rate/prob for MAs
SelectionPressure=8; % Selection Pressure
pause(0.01);
%% Initialization
tic
empty individual.Sequence=[];
empty_individual.Cost=[];
empty_individual.MA=zeros(m,n,l);
empty_individual.ML=zeros(m,n,l);
empty_individual.Tardiness=[];
empty_individual.MaintCost=[];
empty_individual.isFeas=0;
pop=repmat(empty_individual,nPop,1);
% First Generation
for i=1:nPop
    % Create Random Solution
```

```
    pop(i).Sequence=randperm(n);
    display('------');
    display(pop(i).Sequence);
    pop(i).MA=getMinMA(
n,m,l,MLmax,prcsTime,dtrRate,execTime,spCost,wfCost,dueDate,pop(i).Seque
nce );
    display(pop(i).MA);
    % Evalute Newly Created Solution
[pop(i).Cost,pop(i).Tardiness,pop(i).MaintCost,pop(i).isFeas,pop(i).ML]=
evalSol(
n,m,l,MLmax,prcsTime,dtrRate,execTime,spCost,wfCost,dueDate,pop(i).Seque
nce,pop(i).MA,Mem );
    display(pop(i).isFeas);
    display(pop(i).ML);
    display(i);
end
% Sort Population
Costs=[pop.Tardiness];
[Costs, SortOrder]=sort(Costs);
pop=pop(SortOrder);
display(Costs);
% Store Best Solution
BestSol=pop(1);
% Update Worst Cost
WorstCost=max(Costs);
if WorstCost==Inf
    WorstCost = Costs(find(Costs==Inf,1)-1);
end
% Array To Hold Best Cost Values
BestCost=zeros(MaxIt,1);
%% GA Main Loop
it=1;
while Counter<ctrl
    % Calculate Selection Probabilities
    p=exp(-SelectionPressure*Costs/WorstCost);%/WorstCost);
    display(p);
    p=p/sum(p);
    % Crossover
    popc=repmat(empty_individual,nCrossover/2,2);
    for k=1:nCrossover/2
        i1=GARouletteWheelSelction(p);
        i2=GARouletteWheelSelction(p);
% i2=randi(nPop);
        display(['i1 is ' num2str(i1) 'i2 is ' num2str(i2)]);
```

```
    p1=pop(i1);
    p2=pop(i2);
    [popc(k,1).Sequence,
popc(k,2).Sequence]=GACrossover(p1.Sequence, p2.Sequence);
    popc(k,1).MA=getMinMA(
n,m,l,MLmax,prcsTime,dtrRate, execTime,spCost,wfCost,dueDate,popc(k,1).Se
quence );
    popc(k,2).MA=getMinMA(
n,m,l,MLmax,prcsTime,dtrRate,execTime,spCost,wfCost,dueDate,popc(k,2).Se
quence );
    % Evaluatioan of children
[popc(k,1).Cost,popc(k,1).Tardiness,popc(k,1).MaintCost,popc(k,1).isFeas
,popc(k,1).ML]=evalSol(
n,m,l,MLmax,prcsTime,dtrRate,execTime,spCost,wfCost,dueDate,popc(k,1).Se
quence,popc(k,1).MA,Mem );
[popc(k,2).Cost,popc(k,2).Tardiness,popc(k,2).MaintCost,popc(k,2).isFeas
,popc(k,2).ML]=evalSol(
n,m,l,MLmax,prcsTime,dtrRate,execTime,spCost,wfCost,dueDate,popc(k,2).Se
quence,popc(k,2).MA,Mem );
    end
    popc=popc(:);
    % Mutation
    popm=repmat(empty_individual,nMutation,1);
    for k=1: nMutation
        i=GARouletteWheelSelction(p);
        pp=pop(i);
        popm(k).Sequence=pp.Sequence;
        for i=1:m
            for j=1:l
popm(k).MA(i, 2:n,j)=ALGMutate(pp.MA(i, 2:n,j),pp.ML,mu2,n,i,j,l,MLmax,prc
sTime,pp.Sequence,execTime);
            end
        end
        % Pst-mutation evaluation
[popm(k).Cost,popm(k).Tardiness,popm(k).MaintCost,popm(k).isFeas,popm(k)
.ML]=evalSol(
n,m,l,MLmax,prcsTime,dtrRate,execTime,spCost,wfCost,dueDate,popm(k).Sequ
ence,popm(k).MA,Mem );
    end
    % Merge
    pop= [pop
```

```
        popc
        popm]; %#ok
        % Sort Population
        Costs=[pop.Tardiness];
        [Costs,SortOrder]=sort(Costs);
        pop=pop(SortOrder);
    % Delete Extra Individuals
    pop=pop (1:nPop);
    Costs=Costs (1:nPop);
    % Store Best Solution
    BestSol=pop(1);
display(BestSol.Sequence); display(BestSol.Cost);display(BestSol.MA);disp
lay(BestSol.Tardiness);display(BestSol.MaintCost);
    % Update Worst Cost
    WorstCost=max(WorstCost,max(Costs)) ;
    % Store Best Cost
    BestCost(it)=BestSol.Tardiness;
    % Display Iteration Information
    disp(['Iteration ' num2str(it) ': Best Cost = '
num2str(BestCost(it))]);
    % Stoping Condition
    CurrentBest=BestCost(it);
    if CurrentBest==PreviousBest
        Counter=Counter+1;
    end
    if CurrentBest<PreviousBest
        Counter=0;
    end
    if Counter==ctrl
        disp(['number of interations before convergence is: ' num2str(it-
ctrl) ' iteraion(s) = ']);
            end
    PreviousBest=CurrentBest;
    it=it+1;
end
%% Results
    % figure;
% plot(BestCost,'LineWidth',2);
GA_Time = toc;
GA_Iter = it-ctrl;
GA_OFV = CurrentBest;
en\overline{d}
```


## Lower Bound for $Z_{2}$

```
function [ GA OFV,GA Time,GA Iter ] = solveLBZ2(
m,n,l,MLmax,M\overline{2},prcsTime, dtrRäte, execTime, spCost,wfCost,dueDate,pnltCost,
Mem )
%% Initiate
VarSize=n; % Decision Variables Matrix Size
ctrl=20; % interations for convergence
PreviousBest=inf;
Counter=0;
%% GA Parameters
MaxIt=100; % Maximum Number of Iterations
nPop=500; % Population Size
pCrossover=0.3; % Crossover Percentage
nCrossover=round(pCrossover*nPop/2)*2; % Number of Parents (Offsprings)
pMutation=0.3; % Mutation Percentage
nMutation=round(pMutation*nPop); % Number of Mutants
mu=0.5; % Mutation rate/prob
mu2=0.1; % Mutation rate/prob for MAs
SelectionPressure=8; % Selection Pressure
pause(0.01);
%% Initialization
tic
empty_individual.Sequence=[];
empty_individual.Cost=[];
empty_individual.MA=zeros(m,n,l);
empty_individual.ML=zeros(m,n,l);
empty_individual.Tardiness=[];
empty_individual.MaintCost=[];
empty_individual.isFeas=0;
pop=repmat(empty_individual,nPop,1);
% First Generation
for i=1:nPop
    % Create Random Solution
    pop(i).Sequence=randperm(n);
```

pop (i). MA=getMinMA(
n, m, l, MLmax, prcsTime, dtrRate, execTime, spCost, wfCost, dueDate, pop (i). Seque nce ) ;
\% Evalute Newly Created Solution
[pop(i).Cost, pop(i).Tardiness,pop(i).MaintCost,pop(i).isFeas,pop(i).ML]= evalsol(
n, m, l, MLmax, prcsTime, dtrRate, execTime, spCost,wfCost, dueDate, pop (i). Seque nce, pop (i).MA, Mem ) ;
display(i);
end
\% Sort Population
Costs=[pop.MaintCost];
[Costs, SortOrder]=sort(Costs);
pop=pop (SortOrder);
\% Store Best Solution
BestSol=pop (1);
\% Update Worst Cost
WorstCost=max (Costs) ;
if WorstCost==Inf
WorstCost $=\operatorname{Costs}(f i n d(\operatorname{Costs}==\operatorname{Inf}, 1)-1)$;
end
\% Array To Hold Best Cost Values
BestCost=zeros (MaxIt, 1) ;
\%\% GA Main Loop
it $=1$;
while Counter<ctrl
\% Calculate Selection Probabilities
p=exp (-SelectionPressure*Costs/WorstCost) ; \%/WorstCost) ;
$p=p / \operatorname{sum}(p)$;
\% Crossover
popc=repmat (empty_individual, nCrossover/2,2);
for $k=1$ :nCrossover/2
i1=GARouletteWheelSelction (p) ;
i2=GARouletteWheelSelction (p) ;
\% i2=randi(nPop);
p1=pop(i1);
p2=pop(i2);
[popc (k,1). Sequence,
popc (k, 2). Sequence]=GACrossover (p1. Sequence, p2. Sequence);

```
    popc(k,1).MA=getMinMA(
n,m,l,MLmax, prcsTime,dtrRate, execTime, spCost,wfCost,dueDate,popc(k,1).Se
quence );
    popc(k,2).MA=getMinMA(
n,m,l,MLmax, prcsTime, dtrRate, execTime, spCost,wfCost,dueDate, popc(k, 2).Se
quence );
    % Evaluatioan of children
[popc(k,1).Cost,popc(k,1).Tardiness,popc(k,1).MaintCost, popc(k,1).isFeas
, popc(k,1).ML]=evalSol(
n,m,l,MLmax, prcsTime,dtrRate, execTime, spCost,wfCost,dueDate, popc(k,1).Se
quence,popc(k,1).MA,Mem );
[popc(k, 2).Cost,popc(k, 2).Tardiness,popc(k, 2).MaintCost, popc(k, 2).isFeas
, popc(k,2).ML]=evalSol(
n,m,l,MLmax, prcsTime,dtrRate, execTime,spCost,wfCost,dueDate, popc(k, 2).Se
quence,popc(k, 2).MA,Mem );
    end
    popc=popc(:);
    % Mutation
    popm=repmat(empty_individual,nMutation,1);
    for k=1: nMutation
        i=GARouletteWheelSelction(p);
        pp=pop(i);
        popm(k).Sequence=pp.Sequence;
        for i=1:m
            for j=1:l
popm(k).MA(i, 2:n,j)=ALGMutate(pp.MA(i, 2:n,j),pp.ML,mu2,n,i,j,l,MLmax,prc
sTime,pp.Sequence, execTime);
            end
        end
        % Pst-mutation evaluation
[popm(k).Cost,popm(k).Tardiness,popm(k).MaintCost,popm(k).isFeas,popm(k)
.ML]=evalSol(
n,m,l,MLmax, prcsTime,dtrRate, execTime, spCost,wfCost,dueDate, popm(k).Sequ
ence,popm(k).MA,Mem );
    end
    % Merge
    pop=[pop
        popc
        popm]; %#ok
```

```
    % Sort Population
    Costs=[pop.MaintCost];
    [Costs,SortOrder]=sort(Costs);
    pop=pop(SortOrder);
    % Delete Extra Individuals
    pop=pop(1:nPop);
    Costs=Costs(1:nPop);
    % Store Best Solution
    BestSol=pop(1);
display(BestSol.Sequence);display(BestSol.Cost);display(BestSol.MA);disp
lay(BestSol.Tardiness);display(BestSol.MaintCost);
    % Update Worst Cost
    WorstCost=max(WorstCost,max(Costs));
    % Store Best Cost
    BestCost(it)=BestSol.MaintCost;
    % Display Iteration Information
    disp(['Iteration ' num2str(it) ': Best Cost = '
num2str(BestCost(it))]);
    % Stoping Condition
    CurrentBest=BestCost(it);
    if CurrentBest==PreviousBest
    Counter=Counter+1;
    end
    if CurrentBest<PreviousBest
    Counter=0;
    end
    if Counter==ctrl
            disp(['number of interations before convergence is: ' num2str(it-
ctrl) ' iteraion(s) = ']);
            end
        PreviousBest=CurrentBest;
    it=it+1;
end
%% Results
% figure;
% plot(BestCost,'LineWidth',2);
GA Time = toc;
GA_Iter = it-ctrl;
GA_OFV = CurrentBest;
en\overline{d}
```


## The ALG

```
function [ GA_OFV,GA_Time,GA_Iter,bestTard,bestMaint ] = solveGA(
m,n,l,MLmax,M2, prcsTime, dtrRate, execTime, spCost,wfCost,dueDate, pnltCost,
Mem )
%% Initiate
VarSize=n; % Decision Variables Matrix Size
ctrl=20; % interations for convergence
PreviousBest=-inf;
Counter=0;
%% GA Parameters
MaxIt=100; % Maximum Number of Iterations
nPop=500; % Population Size
pCrossover=0.3; % Crossover Percentage
nCrossover=round(pCrossover*nPop/2)*2; % Number of Parents (Offsprings)
pMutation=0.3; % Mutation Percentage
nMutation=round(pMutation*nPop); % Number of Mutants
mu=0.3; % Mutation rate/prob
mu2=0.3; % Mutation rate/prob for MAs
SelectionPressure=8; % Selection Pressure
pause(0.01);
%% Initialization
tic
empty individual.Sequence=[];
empty individual.Cost=[];
empty_individual.MA=zeros(m,n,l);
empty_individual.ML=zeros(m,n,l);
empty_individual.Tardiness=[];
empty_individual.MaintCost=[];
empty_individual.isFeas=0;
pop=repmat(empty_individual,nPop,1);
poppy = randperm(factorial(n),nPop);
for i=1:nPop
    % Create Random Solution
    perm = perms_m(n,poppy(i)-1);
    pop(i).Sequence = perm';
```

```
%
        pop(i).Sequence=randperm(n);
    pop(i).MA=getMinMA(
n,m,l,MLmax, prcsTime, dtrRate, execTime, spCost,wfCost,dueDate, pop(i).Seque
nce );
    % Evalute Newly Created Solution
[pop(i).Cost, pop(i).Tardiness,pop(i).MaintCost,pop(i).isFeas,pop(i).ML]=
evalSol(
n,m,l,MLmax,prcsTime,dtrRate, execTime, spCost,wfCost,dueDate,pop (i).Seque
nce,pop(i).MA,Mem );
    display(i);
end
% Sort Population
Costs=[pop.Cost];
[Costs, SortOrder]=sort(Costs);
pop=pop(SortOrder);
% Store Best Solution
BestSol=pop(1);
% Update Worst Cost
WorstCost=max(Costs);
if WorstCost==0
    WorstCost = Costs(find(Costs==0,1)-1);
end
% Array To Hold Best Cost Values
BestCost=zeros(MaxIt,1);
%% GA Main Loop
it=1;
while Counter<ctrl
    % Calculate Selection Probabilities
    p=-Costs+1;
%
    p=p/sum(p);
    % Crossover
    popc=repmat(empty_individual,nCrossover/2,2);
    for k=1:nCrossover/2
        i1=GARouletteWheelSelction(p);
        i2=GARouletteWheelSelction(p);
%
                i2=randi (nPop);
        p1=pop(i1);
        p2=pop(i2);
```

```
        [popc(k,1).Sequence,
popc(k,2).Sequence]=GACrossover(p1.Sequence,p2.Sequence);
        for i=1:m
            for j=1:l
                [popc(k,1).MA(i,:,j),
popc(k, 2).MA(i,:,j)]=GACrossover(p1.MA(i, :,j),p2.MA(i, :,j));
            end
        end
        % Evaluatioan of children
[popc(k,1).Cost,popc(k,1).Tardiness,popc(k,1).MaintCost,popc(k,1).isFeas
, popc(k,1).ML]=evalSol(
n,m,l,MLmax, prcsTime,dtrRate, execTime,spCost,wfCost,dueDate,popc(k,1).Se
quence, popc (k, 1).MA, Mem ) ;
[popc(k, 2).Cost,popc(k, 2).Tardiness,popc(k, 2).MaintCost,popc(k, 2).isFeas
, popc(k,2).ML]=evalSol(
n,m,l,MLmax, prcsTime,dtrRate, execTime,spCost,wfCost,dueDate, popc(k, 2).Se
quence,popc(k, 2).MA,Mem );
    end
    popc=popc(:);
    % Mutation
    popm=repmat(empty_individual, nMutation,1);
    for k=1: nMutation
        i=GARouletteWheelSelction(p);
        pp=pop(i);
        popm(k).Sequence=pp.Sequence;
        for i=1:m
            for j=1:l
                popm(k).MA(i, 2:n,j)=GAMutate2(pp.MA(i, 2:n,j),mu2,n-1);
            end
        end
        % Pst-mutation evaluation
[popm(k).Cost,popm(k).Tardiness,popm(k).MaintCost,popm(k).isFeas,popm(k)
.ML]=evalSol(
n,m,l,MLmax, prcsTime,dtrRate, execTime,spCost,wfCost,dueDate, popm(k).Sequ
ence,popm(k).MA,Mem );
    end
    % Merge
    pop=[pop
        popc
        popm];
```

```
    % Sort Population
    Costs=[pop.Cost];
    [Costs,SortOrder]=sort(Costs);
    pop=pop(SortOrder);
    % Delete Extra Individuals
    pop=pop(1:nPop);
    Costs=Costs(1:nPop);
    % Store Best Solution
    BestSol=pop(1);
display(BestSol.Sequence); display(BestSol.Cost); display(BestSol.MA);disp
lay(BestSol.Tardiness);display(BestSol.MaintCost);
    % Update Worst Cost
    WorstCost=max(WorstCost,max(Costs));
    % Store Best Cost
    BestCost(it)=BestSol.Cost;
    bestTard = BestSol.Tardiness;
    bestMaint = BestSol.MaintCost;
    % Display Iteration Information
    disp(['Iteration ' num2str(it) ': Best Cost = '
num2str(BestCost(it))]);
    % Stoping Condition
    CurrentBest=BestCost(it);
    if CurrentBest==PreviousBest
    Counter=Counter+1;
    end
    if CurrentBest<PreviousBest
    Counter=0;
    end
    if Counter==ctrl
        disp(['number of interations before convergence is: ' num2str(it-
ctrl) ' iteraion(s) = ']);
            end
        PreviousBest=CurrentBest;
        it=it+1;
end
%% Results
%
% figure;
% plot(BestCost,'LineWidth',2);
GA_Time = toc;
GA_Iter = it-ctrl;
GA_OFV = CurrentBest;
end
```


## D.10: A Function for Writing CPLEX Data Files

```
function writeCPLEX(
m,n,l,tp,MLmax,M1,M2,prcsTime,dtrRate,execTime,spCost,wfCost,dueDate,pnl
tCost,Mem )
fid=fopen(['CPLEX_16A_DataFile_' num2str(m) '-' num2str(n) '-'
num2str(l) '_tp' num2str(tp) '.dat'],'w' );
fprintf(fid,['m=' num2str(m) ';' '\r\n']);
fprintf(fid,['n=' num2str(n) ';' '\r\n']);
fprintf(fid,['l=' num2str(l) ';' '\r\n']);
fprintf(fid,['M1=' num2str(M1) ';' '\r\n']);
fprintf(fid,['M2=' num2str(M2) ';' '\r\n']);
STR='MLmax=';
STR=[STR '[' num2str(MLmax(1,1))];
for k=2:l
    STR=[STR ',' num2str(MLmax(1,k))];
end
STR=[STR '];' '\r\n'];
fprintf(fid,STR);
STR='wfCost=';
STR=[STR '[' num2str(wfCost(1,1))];
for k=2:l
    STR=[STR ',' num2str(wfCost(1,k))];
end
STR=[STR '];' '\r\n'];
fprintf(fid,STR);
STR='dueDate=';
STR=[STR '[' num2str(dueDate(1,1))];
for j=2:n
    STR=[STR ',' num2str(dueDate(1,j))];
end
STR=[STR '];' '\r\n'];
fprintf(fid,STR);
% prcsTime
STR='prcsTime=[';
for i=1:m
    STR=[STR '[' num2str(prcsTime(i,1))];
    for j=2:n
            STR=[STR ',' num2str(prcsTime(i,j))];
        end
        STR=[STR '],' '\r\n'];
end
trimPoint = numel(STR)-5;
STR=[STR(1:trimPoint) '\r\n' '];' '\r\n'];
fprintf(fid,STR);
% execTime
STR='execTime=[';
for i=1:m
```

```
    STR=[STR '[' num2str(execTime(i,1))];
    for k=2:l
        STR=[STR ',' num2str(execTime(i,k))];
    end
    STR=[STR '],' '\r\n'];
end
trimPoint = numel(STR)-5;
STR=[STR(1:trimPoint) '\r\n' '];' '\r\n'];
fprintf(fid,STR);
% spCost
STR='spCost=[';
for i=1:m
    STR=[STR '[' num2str(spCost(i,1))];
    for k=2:l
        STR=[STR ',' num2str(spCost(i,k))];
    end
    STR=[STR '],' '\r\n'];
end
trimPoint = numel(STR)-5;
STR=[STR(1:trimPoint) '\r\n' '];''\\r\n'];
fprintf(fid,STR);
% dtrRate
STR='dtrRate= [';
for i=1:m
    STR=[STR '['];
    for j=1:n
                STR=[STR '[' num2str(dtrRate(i,j,1))];
                for k=2:l
                STR=[STR ',' num2str(dtrRate(i,j,k))];
                end
                STR=[STR '],' '\r\n'];
    end
    trimPoint = numel(STR)-5;
    STR=[STR(1:trimPoint) '],' '\r\n'];
end
trimPoint = numel(STR)-5;
STR=[STR(1:trimPoint) '];' '\r\n'];
fprintf(fid,STR);
fprintf(fid,' \r\n');
%% iFMOLP parameters
% get the coefficients
[alpha11,alpha12,betta1,gamma1,alpha21,alpha22,betta2,gamma2] = getCoef(
Mem );
%write
STR = ['alpha11=' num2str(alpha11) ';'];
fprintf(fid,STR);
fprintf(fid,' \r\n');
STR = ['alpha12=' num2str(alpha12) ';'];
```

```
fprintf(fid,STR);
fprintf(fid,' \r\n');
STR = ['betta1=' num2str(betta1) ';'];
fprintf(fid,STR);
fprintf(fid,' \r\n');
STR = ['gamma1=' num2str(gamma1) ';'];
fprintf(fid,STR);
fprintf(fid,' \r\n');
STR = ['alpha21=' num2str(alpha21) ';'];
fprintf(fid,STR);
fprintf(fid,' \r\n');
STR = ['alpha22=' num2str(alpha22) ';'];
fprintf(fid,STR);
fprintf(fid,' \r\n');
STR = ['betta2=' num2str(betta2) ';'];
fprintf(fid,STR);
fprintf(fid,' \r\n');
STR = ['gamma2=' num2str(gamma2) ';'];
fprintf(fid,STR);
fprintf(fid,' \r\n');
STR = ['mem12=' num2str(Mem(1,2,1)) ';'];
fprintf(fid,STR);
fprintf(fid,' \r\n');
STR = ['mem13=' num2str(Mem(1,3,1)) ';'];
fprintf(fid,STR);
fprintf(fid,' \r\n');
STR = ['mem22=' num2str(Mem(2,2,1)) ';'];
fprintf(fid,STR);
fprintf(fid,' \r\n');
STR = ['mem23=' num2str(Mem(2,3,1)) ';'];
fprintf(fid,STR);
fprintf(fid,' \r\n');
%% iFMOLP Model
fprintf(fid,'/************i-FMOLP model');
fprintf(fid,' \r\n');
% write
STR = ['S<=-(' num2str(alpha11) ')*(e11-d11)-(' num2str(alpha12)
')*(e12-d12)+(' num2str(betta1) ')*(sum( q in 1..n)tardiness[q])+('
num2str(gamma1) ');'];
fprintf(fid,STR);
```

```
fprintf(fid,' \r\n');
STR = ['S<=-(' num2str(alpha21) ')*(e21-d21)-(' num2str(alpha22)
')*(e22-d22)+(' num2str(betta2) ')*(sum( i in 1..m,q in 1..n,k in
1..l)y[i][q][k]*(spCost[i][k]+execTime[i][k]*WfCost[k]))/1000+('
num2str(gamma2) ');'];
fprintf(fid,STR);
fprintf(fid,' \r\n');
STR = ['sum( q in 1..n) tardiness[q]+e11-d11==' num2str(Mem(1,2,1))
';'];
fprintf(fid,STR);
fprintf(fid,' \r\n');
STR = ['sum( q in 1..n) tardiness[q]+e12-d12==' num2str(Mem(1, 3,1))
';'];
fprintf(fid,STR);
fprintf(fid,' \r\n');
STR = ['(sum( i in 1..m,q in 1..n,k in 1..l)
y[i][q][k]*(spCost[i][k]+execTime[i][k]*wfCost[k]))/1000+e21-d21=='
num2str(Mem(2,2,1)) ';'];
fprintf(fid,STR);
fprintf(fid,' \r\n');
STR = ['(sum( i in 1..m,q in 1..n,k in 1..l)
y[i][q][k]*(spCost[i][k]+execTime[i][k]*wfCost[k]))/1000+e22-d22=='
num2str(Mem(2,3,1)) ';'];
fprintf(fid,STR);
fprintf(fid,' \r\n');
fprintf(fid,' \r\n');
fprintf(fid,'*************/');
fclose('all');
end
```


## D.11: Automation of the Computational Experiments

```
clc;
clear;
close all;
%determining the size
for m=2:2
    for n=15:15%[6,10,15]
        for l=3:3%[1,2,3]
                filename = ['16A-Results-' num2str(m) '-' num2str(n) '-'
num2str(l) '.xlsx'];
                A = [];
                sheet = 'Sheet1';
        row = 2;
        xlRange = ['A' num2str(row)];
        for tp=1:30
                try
                    % generate test problem
                    [
MLmax,M1,M2,prcsTime,dtrRate,execTime,spCost,wfCost,dueDate,pnltCost ] =
TPG(m,n,l,tp);
            % calculate the membership function (Mem)
                    Mem = zeros(2,4,2);
                            % calculate the lower bound of z1; a GA just for
tardiness
                            [ LBZ1_OFV,LBZ1_Time,LBZ1_Iter ] = solveLBZ1(
m,n,l,MLmax,M2,prcsTime,dtrRate,execTime,spCost,wfCost,dueDate,pnltCost,
Mem );
                    % calculate the lower bound of z2; a GA just for
MaintCost
                            [ LBZ2_OFV,LBZ2_Time,LBZ2_Iter ] = solveLBZ2(
m,n,l,MLmax,M2,prcsTime,dtrRate,execTime,spCost,wfCost,dueDate,pnltCost,
Mem );
    LBZ2_OFV = LBZ2_OFV/1000;
    % form the table
    Mem(1,4,1)=3*LBZ1_OFV/4;
    Mem(2,4,1)=3*LBZ2_OFV/4;
    for me=1:3
        me2=4-me;
    Mem(1,me2,1)=LBZ1_OFV+me*(1*LBZ1_OFV/3);
    Mem(2,me2,1)=LBZ2_OFV+me*(1*LBZ2_OFV/3);
    end
    Mem(1,:,2)=[0.0 0.5 0.75 1.0];
    Mem(2,:,2)=[l0.0 0.5 0.75 1.0];
    % CPLEX: write data file and iFMOLP-part of the
model
writeCPLEX(m,n,l,tp,MLmax,M1,M2,prcsTime,dtrRate, execTime,spCost,wfCost,
dueDate,pnltCost,Mem);
```

    \% solve with GA
    [GA_OFV, GA_Time, GA_Iter, GA_z1,GA_z2] = solveGA(
$m, n, l, M L m a x, M 2, p r c s T i m e, d t r R a t \bar{e}, ~ e x e c T i m e, ~ s p C o s t, w f C o \bar{s} t, d u e D a t e, ~ p n l t C o s t, ~$ Mem ) ;
\% solve with ALG
[ALG_OFV,ALG_Time,ALG_Iter,ALG_z1,ALG_z2] =
solveALG (
m, $n, l, M L m a x, M 2, p r c s T i m e, d t r R a t e, ~ e x e c T i m e, ~ s p C o s t, w f C o s t, d u e D a t e, ~ p n l t c o s t, ~$ Mem ) ;
\% save the results
$t t=\left[\begin{array}{lll}\mathrm{m} & \mathrm{n} & \mathrm{t} p \\ \mathrm{LBZ1} & \text { OFV LBZ2_OFV GA_OFV GA_Time }\end{array}\right.$
GA_Iter GA_z1 GA_z2 ALG_OFV ALG_Time ALG_Iter ALG_z1 ALG_z2 0 A 0 0 0 ;
$A=[A ; t t] ;$ catch
$\mathrm{tt}=\left[\begin{array}{llllllllllllllllllll}\mathrm{m} & \mathrm{n} & 1 & \mathrm{tp} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\end{array}\right] ;$ $A=[A ; t t]$; end
end
\% print the results in Excel
xlswrite(filename, A, sheet, xlRange);
end
end
end

## Appendix E: MATLAB Codes Used in Chapter III

## E.1: Computational Experiments Main File

```
clc;
clear;
%% Experiment Iniital setting
filename='Experiments.xlsx';
phase=4;
nTP=30; % Number of Test Problems
switch phase
        case 0
            N=[4,6,8];
            DDTF=4;
            maxML=50;
            POP=[25,50,100,200,400];
        case 1
            N=4;
            DDTF=4;
            maxML=50;
            POP=400;
        case 2
            N=5:9;
            DDTF=4;
            maxML=50;
            POP=200;
    case 3
            N=4;
            DDTF=3:4;
            maxML=50;
            POP=200;
    case 4
        N=4;
        DDTF=4;
        maxML=[40 50 60];
        POP=200;
end
gaParameter.MaxIt=100; % Max Iteration of
GA
gaParameter.MaxStall=floor(0.20*gaParameter.MaxIt); % Stall Counter
ROWs=size(POP,1)*size(N,1)*size(maxML,1)*size(DDTF,1)*nTP;
data=zeros(ROWs,10); % matrix to store
resutls
xlswrite(filename,data(:, 1:2),sheet,'A3')
```

```
xlswrite(filename, data(:, 3:end), sheet,'H3')
%% Run Experiment
row=0;
for n=N
    for k=maxML
            experiment.maxML=k;
            experiment. N=n;
            experiment.phase=phase;
            for q=DDTF
                experiment.DDTF=q;
                for TP=1:nTP
                    experiment.TP=TP;
                        model=CreateModel(experiment); % Create
Model
                        for popSize=POP
                disp(['Test Problem ', num2str(TP), ' for
N=', num2str(n),', maxML=', num2str(k),', DDTF=', num2str(q),', and
PopSize=',num2str(popSize),' initilized ...'])
                            gaParameter.nPop=popSize; %
Population size of GA
                                    row=row+1;
                                    tstart = tic; % model
start time
    BestSol = ga(model,gaParameter); % Call
GA
    tElapsed = toc(tStart);
%Calculating Computation time
    % Store data
    z=BestSol.Sol.TotalCost;
    z1=BestSol.Sol.AvgMAcost;
    z2=BestSol.Sol.AvgTardinessCost;
    it=BestSol.it;
    data(row,:)=[popSize, n, q, k, TP, tElapsed, z, z1,
z2, it];
num2str(2+row)])
    xlswrite(filename, data(row, 6:end), sheet,['J'
num2str(2+row) ])
disp('*** Test problem completed.')
    end
        end
        end
        end
end
save('data');
```


## E.2: Test Problem Generator

```
function model=CreateModel(experiment)
% clear;
%% Initial Parameters
N=experiment.N;
TP=experiment.TP;
DDTF=experiment.DDTF; % Due Date Tightness Factor vals=(3 4 5)
maxML=experiment.maxML; % Max Maintanace Level for each MA
M=3; % Number of Machines
L=3; % Number of Maintanance Activities
R=30; % Number of simulation replications
nVar=N+(N-1)*M; % Number of Decision Variables
%% Stochastic Data Distributions
% 1- Jobs TRI parameters
    JobsTRI_Parameters=zeros(N, 3); % Triangular Dist
Matrix (Holds min,mLikely, max for each job processing time
    for j=1:N
        parameters=randsample(20:120,3); % Picking 3 numbers
b/w 20 and 120 for TRI parameters
        JobsTRI Parameters(j,:)=sort(parameters);
    end
% 2- MAs TRI parameters
    MAsTRI_Parameters=zeros(L, 3); % Triangular Dist
Matrix (Holds min,mLikely, max for each MAs Duration
        for k=1:L
            parameters=randsample(8:30,3); % Picking 3 numbers
b/w 8 and 30 for TRI parameters
            MAsTRI_Parameters(k,:)=sort(parameters);
    end
    % Jobs Due Date Info
    eCmax=N* (max(JobsTRI_Parameters(:)) +max(JobsTRI_Parameters(:)) );
% Estimated Cmax
jobsDD=[max(JobsTRI_Parameters(:)) +max(JobsTRI_Parameters(:)),round(2*eC
max/DDTF)]; % UNIF(a b)
    jobsPC=[10 20]; % Jobs
Penalty Cost: UNIF(a b)
    MAspc=[150 450]; % MA
spare part cost: UNIF(a b)
    MAwfC=20; % MA
work force cost
    maxLevels=[4*maxML 5*maxML 6*maxML]; % MA Max
Levels: Fixed
```



```
    MADDUDR=[1, 0.85, 0.75];
%Maitanace Activity Duration Discount Rate
```

PTIR=[1 1; 0.66 1.5; 0.33 2;-5.00 2];
\%Maitanace Activity Duration Discount Rate
\% P Population based data
\%1-Jobs
empty_job.ST=zeros(1,M); \% Start Time of job on machine m
empty_job.FT=zeros(1,M); $\quad$ Finish Time of job on machine m
empty_job.Dd=0; $\quad$ \% Due Date of job j
empty_job.LateP=0; $\quad$ L Lateliness Penalty cost of job j
per time unit
empty_job.TDcost=0; \% Tardiness Cost of Job j
empty_job.Du=zeros(R,M); \% Duration of job on machine m
empty_job.MADu=zeros(R,M); \% Maintanance Activity Duration of
before starting job
empty_job.MACost=zeros(R,M); \% Maintanance Activity Duration of
before starting job empty_job.MA=zeros(L,M); \% Maitanance Activity type (0-7)/
Check the excel table
empty_job.TRI=zeros $(1,3) ; \quad$ Triangular Dist. Parameters used to genereate the processing time for the job

Job=repmat (empty_job,1,N); $\quad$ Create the population of jobs
\% \% Generating Jobs Info
for $j=1: N$
for $i=1: M$
for rep=1:R
Job (j). Du (rep,i)=round (randDist('TRIA',
JobsTRI_Parameters(j,1),
JobsTRI_Parameters(j,2), JobsTRI_Parameters(j, 3))); end
end
Job(j).Dd=round(randDist('UNI', jobsDD(1), jobsDD(2)));
Job(j).LateP=round(randDist('UNI', jobsPC(1), jobsPC(2))); Job(j).TRI=JobsTRI_Parameters(j,:);
end
\%\% Generating MAs Info
empty_MA.MaxLevel=0; $\quad$ Max level of MAs
empty_MA.wf=0; $\quad$ Whorkforce Cost of MA l per
time unit
empty MA.Level=zeros(1,M); $\quad$ Initial Maitanace Level of
part 1 on machine $m$
empty_MA.count=zeros (1,M); \% Number of replacement of
part l
empty_MA.sp=zeros $(1, M) ; \quad$ Spare Part cost of MA 1 on
machine m
empty_MA.TRI=zeros $(1,3) ; \quad$ O TRI dist parameters to
generate MAs Duration
empty_MA.Du=zeros (R,M); \% Duration time to replace
part $l$ on machine $m$
empty_MA.timeLevel=zeros(R,M,2,N*2); \% Status of Maitanance Level
of each part on each machine before and after each job

```
    MA=repmat(empty_MA,1,L);
    % Define Duration of maintanace on each machine
    for k=1:L
        for i=1:M
            for rep=1:R
            MA(k).Du(rep,i)=round(randDist('TRIA',
MAsTRI_Parameters(k,1),MAsTRI_Parameters(k,2),MAsTRI_Parameters(k,2)));
                end
                    MA(k).sp(1,i)=round(randDist('UNI',MAspc(1),MAspc(2)));
            end
        MA(k).MaxLevel=maxLevels(k);
        MA(k).wf=MAwfc;
        MA(k).Level(1,:)=maxLevels(k);
        MA(k).TRI=MAsTRI_Parameters(k,:);
    end
%% Capsulate Model
    model.N=N;
    model.M=M;
    model.L=L;
    model.R=R;
    model.Job=Job;
    model.MA=MA;
    model.nVar=nVar;
    model.MA_Types=MA_Types;
    model.MADUDR=MADUDR;
    model.PTIR=PTIR;
    model.TP=TP;
    model.phase=experiment.phase;
    model.DDTF=DDTF;
    model.maxML=experiment.maxML;
    save('model');
%% Create CPLEX File
    CreateCPLEXFile(model);
```


## E.3: CPLEX Data File Writer

```
function CreateCPLEXFile(model)
%% Extract Info from model
TP=model.TP;
N=model.N;
M=model.M;
L=model.L;
R=model.R;
Job=model.Job;
MA=model.MA;
DDTF=model.DDTF;
maxML=model.maxML;
phase = model.phase;
%% Initiliziation
    O=2^L-1; %number of combinations
    fid=fopen(['17D_CPLEX_N' num2str(N) '-DDTE' num2str(DDTF) '-maxML'
num2str(maxML) '-TP' num2\overline{str}(TP) '-Phase' num2str(phase) '.dat'],'w' );
    fprintf(fid,['/* Project 17D */' '\r\n' '/*Date: '
datestr(now,'yyyy-mm-dd_HH-MM-SS') '*/' '\r\n']);
    fprintf(fid,['/*==========================================*/' '\r\n']);
    fprintf(fid,['m=' num2str(M) ';' '\r\n']);
    fprintf(fid,['n=' num2str(N) ';' '\r\n']);
    fprintf(fid,['l=' num2str(L) ';' '\r\n']);
    fprintf(fid,['S=' num2str(R) ';' '\r\n']);
    fprintf(fid,['o=' num2str(O) ';' '\r\n']);
    fprintf(fid,['K=' num2str(100000) ';' '\r\n']);
    %probabilities
    str = 'probability =';
    prob = 1/R;
    str = [str '[' num2str(prob)];
    for s=2:R
        str = [str ',' num2str(prob)];
    end
    str = [str '];' '\r\n'];
    fprintf(fid,str);
    %processing times
    str = 'p = [';
    for i=1:M-1
    str=[str '['];
        for j=1:N-1
            str=[str '[' num2str(Job(j).Du(1,i))];
            for s=2:R
                str=[str ',' num2str(Job(j).Du(s,i))];
            end
            str=[str '],'];
        end
        str=[str '[' num2str(Job(N).Du(1,i))];
```

```
        for s=2:R
            str=[str ',' num2str(Job(N).Du(s,i))];
        end
        str=[str ']'];
        str=[str '],'];
    end
    str=[str '['];
    for j=1:N-1
        str=[str '[' num2str(Job(j).Du(1,M))];
        for s=2:R
            str=[str ',' num2str(Job(j).Du(s,M))];
        end
        str=[str '],'];
    end
    str=[str '[' num2str(Job(N).Du(1,M))];
    for s=2:R
    str=[str ',' num2str(Job(N).Du(s,M))];
    end
    str=[str ']]];' '\r\n'];
    fprintf(fid,str);
    %maintenance durations (combinations)
    str = ['epsilon = [' '\r\n'];
    for i=1:M-1
    str=[str '['];
        for o=1:0-1
            if 0<4
            str = [str '[' num2str(MA(o).Du(1,i))];
            elseif o==4
                str = [str '['
num2str(0.75*(MA(1).Du(1,i) +MA(2).Du(1,i)))];
            elseif o==5
                str = [str '['
num2str(0.75*(MA(1).Du(1,i)+MA(3).Du(1,i)))];
    elseif o==6
                            str = [str '['
num2str(0.75*(MA(2).Du(1,i) +MA(3).Du(1,i)))];
    elseif o==7
                            str = [str '['
num2str(0.60*(MA(1).Du(1,i)+MA(2).Du(1,i)+MA(3).Du(1,i))) ];
    end
    for s=2:R
                            if 0<4
                            str = [str ',' num2str(MA(o).Du(s,i))];
            elseif o==4
                            str = [str ','
num2str(0.75*(MA(1).Du(s,i) +MA(2).Du(s,i)))];
    elseif o==5
                                    str = [str ','
num2str(0.75*(MA(1).Du(s,i)+MA(3).Du(s,i)))];
    elseif o==6
                str = [str ','
num2str(0.75*(MA(2).Du(s,i)+MA(3).Du(s,i)))];
    elseif o==7
```

```
        str = [str ','
num2str(0.60*(MA(1).Du(s,i) +MA(2).Du(s,i) +MA(3).Du(s,i)))];
                        end
        end
        str=[str '],' '\r\n'];
    end
    str = [str '['
num2str(0.60*(MA(1).Du(1,i)+MA(2).Du(1,i)+MA(3).Du(1,i))) ];
    for s=2:R
        str = [str ','
num2str(0.60*(MA(1).Du(s,i)+MA(2).Du(s,i)+MA(3).Du(s,i)))];
        end
        str = [str ']],' '\r\n'];
    end
    str=[str '['];
    for o=1:0-1
        if o<4
        str = [str '[' num2str(MA(O).Du(1,M))];
        elseif o==4
            str = [str '[' num2str(0.75*(MA(1).Du(1,M) +MA(2).Du(1,M)))];
        elseif o==5
            str = [str '[' num2str(0.75*(MA(1).Du(1,M) +MA(3).Du(1,M)))];
        elseif o==6
            str = [str '[' num2str(0.75*(MA(2).Du(1,M) +MA(3).Du(1,M)))];
        elseif o==7
            str = [str '['
num2str(0.60*(MA(1).Du(1,M)+MA(2).Du(1,M)+MA(3).Du(1,i))) ];
        end
        for s=2:R
            if 0<4
            str = [str ',' num2str(MA(O).Du(s,M))];
            elseif o==4
                str = [str ','
num2str(0.75*(MA(1).Du(s,M) +MA(2).Du(s,M)))];
            elseif o==5
                str = [str ','
num2str(0.75*(MA(1).Du(s,M) +MA(3).Du(s,M)))];
            elseif o==6
                str = [str ','
num2str(0.75*(MA(2).Du(s,M) +MA (3).Du(s,M)))];
            elseif O==7
                str = [str ','
num2str(0.60*(MA(1).Du(s,M)+MA(2).Du(s,M)+MA(3).Du(s,M)))];
            end
        end
        str=[str '],' '\r\n'];
    end
    str = [str '['
num2str(0.60*(MA(1).Du(1,M)+MA(2).Du(1,M)+MA(3).Du(1,M))) ];
    for s=2:R
        str = [str ' ','
num2str(0.60*(MA(1).Du(s,M) +MA (2).Du(s,M) +MA(3).Du(s,M)))];
    end
    str = [str ']]];' '\r\n'];
```

```
fprintf(fid,str);
```

\%maintenance durations (original)
str $=$ ['epsilon2 $=$ [' '\r\n'];
for $i=1: M-1$
str=[str ' ['];
for $0=1: 0-1$
if $0<4$
str $=$ [str ' [' num2str(MA(o).Du(1,i))];
elseif $O==4$
str $=$ [str ' ['
num2str (1.0* (MA (1). Du (1, i) +MA (2). Du (1, i)))];
elseif $O==5$
str $=$ [str ' ['
num2str (1.0* (MA (1).Du(1, i) +MA (3).Du(1, i)))];
elseif $0==6$
str $=$ [str ' ['
num2str(1.0* (MA (2). Du(1,i) +MA (3). Du(1,i)))];
elseif $0==7$
str $=$ [str ' ['
num2str (1.0* (MA (1). Du (1, i) +MA (2). Du (1, i) +MA (3). Du (1, i)) ) ;
end
for $s=2: R$
if $0<4$
str $=$ [str ',' num2str(MA(o).Du(s,i))];
elseif $0==4$
str $=$ [str ','
num2str (1.0* (MA (1). Du (s,i) +MA (2). Du (s,i)))];
elseif $O==5$
str $=$ [str ','
num2str(1.0*(MA(1).Du(s,i)+MA(3).Du(s,i)))];
elseif $0==6$
str $=$ [str ','
num2str (1.0* (MA (2). Du (s,i) +MA (3). Du (s,i)))];
elseif $0==7$
str $=$ [str ','
num2str(1.0*(MA(1).Du(s,i) +MA(2).Du(s,i)+MA(3).Du(s,i)))];
end
end
str=[str '],' '\r\n'];
end
str $=$ [str ' ['
num2str (1.0* (MA (1). Du (1, i) +MA (2). Du (1, i) +MA (3). Du (1, i))) ];
for $s=2: R$
str $=$ [str ','
num2str (1.0* (MA (1).Du(s,i) +MA (2).Du(s,i) +MA (3).Du(s,i)))];
end
str $=[$ str '] $\left.], ' \backslash r \backslash n^{\prime}\right] ;$
end
str=[str ' ['];
for $O=1: 0-1$
if $0<4$

```
    str = [str '[' num2str(MA(O).Du(1,M))];
    elseif o==4
    str = [str '[' num2str(1.0*(MA(1).Du(1,M)+MA(2).Du(1,M)))];
    elseif o==5
    str = [str '[' num2str(1.0*(MA(1).Du(1,M)+MA(3).Du(1,M)))];
    elseif o==6
    str = [str '[' num2str(1.0*(MA (2).Du(1,M) +MA(3).Du(1,M)))];
    elseif o==7
    str = [str '['
num2str(1.0*(MA(1).Du(1,M) +MA(2).Du(1,M)+MA(3).Du(1,i))) ];
    end
    for s=2:R
        if 0<4
        str = [str ',' num2str(MA(O).Du(s,M))];
        elseif o==4
                str = [str ','
num2str(1.0*(MA(1).Du(s,M)+MA(2).Du(s,M)))];
        elseif o==5
                str = [str ','
num2str(1.0*(MA(1).Du(s,M)+MA(3).Du(s,M)))];
    elseif o==6
                str = [str ','
num2str(1.0*(MA (2).Du(s,M) +MA(3).Du(s,M)))];
            elseif o==7
                str = [str ','
num2str(1.0*(MA(1).Du(s,M) +MA(2).Du(s,M) +MA(3).Du(s,M)))];
            end
        end
        str=[str '],' '\r\n'];
    end
    str = [str '['
num2str(1.0*(MA(1).Du(1,M)+MA(2).Du(1,M)+MA(3).Du(1,M))) ];
    for s=2:R
        str = [str ','
num2str(1.0*(MA (1).Du(s,M) +MA (2).Du(s,M) +MA(3).Du(s,M)))];
    end
    str = [str ']]];' '\r\n'];
    fprintf(fid,str);
    %ML_max
    str = ['MLmax = [' num2str(MA(1).MaxLevel) ','
num2str(MA(2).MaxLevel) ',' num2str(MA(3).MaxLevel) '];' '\r\n'];
    fprintf(fid,str);
    %Spare part cost for MA combinations
    str =['SPprime = [' '\r\n'];
    for i=1:M-1
        str = [str '[' num2str(MA(1).sp(i))];
        str = [str ',' num2str(MA(2).sp(i))];
        str = [str ',' num2str(MA(3).sp(i))];
        str = [str ',' num2str(MA(1).sp(i) + MA(2).sp(i))];
        str = [str ',' num2str(MA(1).sp(i) + MA(3).sp(i))];
        str = [str ',' num2str(MA(2).sp(i) + MA(3).sp(i))];
```

```
        str = [str ',' num2str(MA(1).sp(i) + MA(2).sp(i) +
MA(3).sp(i))];
            str = [str '],' '\r\n'];
    end
    str = [str '[' num2str(MA(1).sp(M))];
    str = [str ',' num2str(MA(2).sp(M))];
    str = [str ',' num2str(MA(3).sp(M))];
    str = [str ',' num2str(MA(1).sp(M) + MA(2).sp(M))];
    str = [str ',' num2str(MA(1).sp(M) + MA(3).sp(M))];
    str = [str ',' num2str(MA(2).sp(M) + MA(3).sp(M))];
    str = [str ',' num2str(MA(1).sp(M) + MA(2).sp(M) + MA(3).sp(M))
']];' '\r\n'];
    fprintf(fid,str);
    %Work Force Unit Cost
    str = ['WF = 20;' '\r\n'];
    fprintf(fid,str);
    %due dates
    str = ['d = [' num2str(Job(1).Dd)];
    for j=2:N
        str = [str ',' num2str(Job(j).Dd)];
    end
    str = [str '];' '\r\n'];
    fprintf(fid,str);
    %penaltis
    str = ['pi = [' num2str(Job(1).LateP)];
    for j=2:N
        str = [str ',' num2str(Job(j).LateP)];
end
str = [str '];' '\r\n'];
fprintf(fid,str);
fprintf(fid,['a = [1.1,1.2,1.3];' '\r\n']);
fprintf(fid,['b = [1.1,1.2,1.3,2.3,2.4,2.5,3.6];' '\r\n']);
%the end of the CPLEX data file
end
```


## E.4: Crossover Function

```
function [y1 y2]=Crossover(x1,x2,gamma,VarMin,VarMax)
    alpha=unifrnd(-gamma,1+gamma,size(x1));
    y1=alpha.*x1+(1-alpha).*x2;
    y2=alpha.*x2+(1-alpha).*x1;
    y1=max(y1,VarMin);
    y1=min(y1,VarMax);
    y2=max(y2,VarMin);
    y2=min(y2,VarMax);
end
```


## E.5: Mutation Function

```
function y=Mutate(x,mu,VarMin,VarMax)
    nVar=numel(x);
    nmu=ceil(mu*nVar);
    j=randsample(nVar,nmu);
    sigma=0.1*(VarMax-VarMin);
    y=x;
    y(j)=x(j)+sigma*randn(size(j))';
    y=max (y,VarMin);
    y=min(y,VarMax);
end
```


## E.6: Roulette Wheel Selection Function

```
function i=RouletteWheelSelection(P)
    r=rand;
    C=cumsum(P);
    i=find(r<=c,1,'first');
end
```


## E.7: The GA and Related Functions

```
function BestSol = ga(model,gaParameter)
clearvars -except model gaParameter
    global NFE;
    NFE=0;
    CostFunction=@(q) MyCost(q,model); % Cost Function
    nVar=model.nVar; % Number of Decision
Variables
    VarSize=[1 nVar]; % Size of Decision Variables
Matrix
    VarMin=0; % Lower Bound of Variables
    VarMax=1;
    % Upper Bound of Variables
%% GA Parameters
    MaxIt=gaParameter.MaxIt; % Maximum Number of
Iterations
    nPop=gaParameter.nPop; % Population Size
    pc=0.8; % Crossover Percentage
    nc=2*round(pc*nPop/2); % Number of Offsprings
(Parnets)
    pm=0.8; % Mutation Percentage
    nm=round (pm*nPop); % Number of Mutants
    gamma=0.05; % Normal Dist Mutation
Factor
    mu=0.03; % Mutation Rate
    % Roulette Wheel Selection Parameters
    beta=8; % Selection Pressure
    % Stall Parameter
    StallCounter=0; % Stall Counter
    MaxStall=gaParameter.MaxStall; % Max number of Stalls
    BestSol.Cost=10^6;
% big M
%% Initialization
    empty_individual.Position=[];
    empty_individual.Cost=[];
    pop=repmat(empty_individual,nPop,1);
    for i=1:nPop
            % Initialize Position
                pop(i).Position=unifrnd(VarMin,VarMax,VarSize);
```

```
        % Evaluation
            [pop(i).Cost, pop(i).Sol]=CostFunction(pop(i).Position);
    end
    % Sort Population
    Costs=[pop.Cost];
    [Costs, SortOrder]=sort(Costs);
    pop=pop(SortOrder);
    % Array to Hold Best Cost Values
    BestCost=zeros(MaxIt,1);
    % Store Cost
    WorstCost=pop (end).Cost;
    % Array to Hold Number of Function Evaluations
    nfe=zeros(MaxIt,1);
%% Main Loop
    for it=1:MaxIt
        % Calculate Selection Probabilities
        P=exp(-beta*Costs/WorstCost);
        P=P/sum(P);
        % Crossover
        popc=repmat(empty_individual,nc/2,2);
        for k=1:nc/2
            % Select Parents Indices
            i1=RouletteWheelSelection(P);
            i2=RouletteWheelSelection(P);
            % Select Parents
            p1=pop(i1);
            p2=pop(i2);
            % Apply Crossover
            [popc(k,1).Position , popc(k,2).Position]=...
            Crossover(p1.Position,p2.Position,gamma,VarMin,VarMax);
            % Evaluate Offsprings
            [popc(k,1).Cost,
popc(k,1).Sol]=CostFunction(popc(k,1).Position);
            [popc(k, 2).Cost,
popc(k,2).Sol]=CostFunction(popc(k,2).Position);
        end
        popc=popc(:);
```

```
        % Mutation
    popm=repmat(empty_individual,nm,1);
    for k=1:nm
        % Select Parent
        i=randi([1 nPop]);
        p=pop(i);
        % Apply Mutation
        popm(k).Position=Mutate(p.Position,mu,VarMin,VarMax);
        % Evaluate Mutant
        [popm(k).Cost, popm(k).Sol]=CostFunction(popm(k).Position);
    end
    % Create Merged Population
    pop=[pop
        popc
        popm];
    % Sort Population
    Costs=[pop.Cost];
    [Costs, SortOrder]=sort(Costs);
    pop=pop(SortOrder);
    % Update Worst Cost
    WorstCost=max(WorstCost,pop(end).Cost);
    % Truncation
    pop=pop(1:nPop);
    Costs=Costs(1:nPop);
    % Store Best Solution
    lastBestCost=BestSol.Cost; % Storing last Besto
sol for checking stall condition
    BestSol=pop(1);
    BestSol.it=MaxIt;
    BestSol.Stall='False';
    % Checking model Stallation
        if lastBestCost==BestSol.Cost
                StallCounter=StallCounter+1;
            else
            StallCounter=0;
        end
    % Store Best Cost Ever Found
    BestCost(it)=BestSol.Cost;
    % Store NFE
    nfe(it)=NFE;
    if StallCounter>MaxStall
```

```
        BestSol.Stall='true';
        BestSol.it=it;
        break;
    end
end
save('BestSol');
```


## CreateNeighbor()

```
function qnew=CreateNeighbor(q)
    m=randi([1 3]);
    switch m
        case 1
        % Do Swap
            qnew=Swap(q);
        case 2
            % Do Reversion
            qnew=Reversion(q);
        case 3
            % Do Insertion
            qnew=Insertion(q);
    end
end
```


## CreateRandomSolution()

```
function q=CreateRandomSolution(model)
    nVar=model.nVar;
    q=randperm(nVar);
end
```

MyCost()
function [z , SimSol]=MyCost(q,model)

```
    global NFE;
    NFE=NFE+1;
    R=model.R;
    w1=1;
    w2=1;
    w3=10^6;
    Z=zeros(1,R);
    z1=zeros(1,R);
    z2=zeros(1,R);
    z3=zeros(1,R);
    sol=[];
    for rep=1:R
        sol=ParseSolution(q,model,rep);
        z1(rep)=sol.TotalMAcost;
        z2(rep)=sol.TotalTardinessCost;
        z3(rep)=sol.InfeasibilityCounter;
    end
    % Objective Function
    z=mean(w1*z1+w2*z2+w3*z3);
    % Capsulate important info in SimSol
    SimSol.AvgMAcost=mean(z1);
    SimSol.AvgTardinessCost=mean(z2);
    SimSol.TotalCost=z;
    SimSol.newQ=sol.newQ;
    SimSol.model=sol.model;
end
```


## ParseSolution()

```
function sol=ParseSolution(q,model,rep)
    InfeasibilityCounter=0;
    %% Convert q to newQ by adding 0 for the first job in each
machine(no Maintanance is required)
    N=model.N;
    M=model.M;
    L=model.L;
    % Create newQ matrix
    [~ ,newQ]=sort(q(1:N));
    for k=1:L
        newQ=[newQ 0 q(N+(k-1)*(N-1)+1:N+k* (N-1))];
    end
    nmodel=UpadeModel(newQ,model,rep); %Update model by finding MAs
duration and cost
Job=nmodel. Job;
MA=nmodel.MA;
```

```
    PTIR=nmodel.PTIR;
    % Retreiving jobs sequence
    JobSequence=newQ(1:N);
%% Parse Solution
    for i=1:M
        jobcounter=0;
        for j=JobSequence
            jobcounter=jobcounter+1;
                % calacualte the avg health level of machine parts
                AvgLevel=0;
                for k=1:L
                if Job(j).MA(k,i)==1
                    MA(k).Level(1, i)=MA(k).MaxLevel;
                    end
                    AvgLevel=AvgLevel+MA(k).Level(1, i)/MA(k).MaxLevel;
                end
                AvgLevel=AvgLevel/L;
                rate=PTIR(find(PTIR(:,1)<AvgLevel,1,'first')-1,2);
                JobDu=Job(j).Du(rep,i)*rate;
                if find(JobSequence==j)==1
                if i==1 % The first jobs start at time 0 on first
Machine
                            Job(j).ST(1, i)=0;
                            Job(j).FT(1,i)=Job(j).ST(1,i)+Job(j).Du(rep,i);
                else
                            Job(j).ST(1,i)=Job(j).FT(1,i-1);
                            Job(j).FT(1,i)=Job(j).ST(1,i) +Job(j).Du(rep,i);
                end
                else
                        previous_job=JobSequence(jobcounter-1);
                if i==1
Job(j).ST(1,i)=Job(previous_job).FT(1,i)+Job(j).MADu(rep,i);
                            Job(j).FT(1, i)=Job(j).ST(1,i) +JobDu;
        else
                            Job(j).ST(1, i)=max(Job(j).FT(1, i-
1), Job(previous_job).FT(1,i) +Job(j).MADu(rep,i));
                Job(j).FT(1, i)=Job(j).ST(1,i) +JobDu;
            end
        end
            % Update the level of MAs to Max level in case of
Maintanance
        for k=1:L
            % Deteriorate MAs
            MA(k).Level(1, i)=MA(k).Level(1, i) - JobDu;
            if MA(k).Level(1, i)<=0
                InfeasibilityCounter=InfeasibilityCounter+1;
```

```
                end
            end
        end
    end
    TotalMAcost=0;
    TotalTardinessCost=0;
    for j=1:N
        TotalMAcost=TotalMAcost+sum(Job(j).MACost(rep,:));
        Job(j).TDcost=max(Job (j).FT(1,M)-Job (j).Dd, 0) *Job (j).LateP;
        TotalTardinessCost=TotalTardinessCost+Job(j).TDcost;
    end
    %Update Job and MAs
    nmodel.Job=Job;
    nmodel.MA=MA;
    % Capsulate Sol
    sol.InfeasibilityCounter=InfeasibilityCounter;
    sol.TotalMAcost=TotalMAcost;
    sol.TotalTardinessCost=TotalTardinessCost;
    sol.newQ=newQ;
    sol.model=nmodel;
end
```


## RandDist()

```
function rnd=randDist(Dist, p1, p2,p3)
switch Dist
    case 'NORM'
        pd=makedist('Normal', p1,p2);
        rnd=random(pd);
    case 'TRIA'
        pd=makedist('Triangular', p1,p2,p3);
        rnd=random(pd);
    case 'UNI'
        rnd=p1+rand*abs((p2-p1));
end
end
```


## UpdateModel()

```
function nmodel=UpadeModel(q,model,rep)
    nmodel=model;
    M=nmodel.M; % Number of Machines
    N=nmodel.N; % Numbner of Jobs
    L=nmodel.L; % Number of MAs
```

```
    Job=nmodel.Job; % Jobs
    MA=nmodel.MA; % Maintanance Activities
    MA_Types=nmodel.MA_Types; % Table of Maintanance combination
(0-7)
    JobsSequence=q(1:N); % Get the sequence of jobs
    MADUDR=nmodel.MADUDR; %Maitanace Activity Duration
Discount Rate
    for i=1:M
        for j=JobsSequence
        JobOrder=find(JobsSequence==j,1,'first'); % obtain the
order of job
    typenumber=q(N+(i-1)*N+JobOrder); % this is a rand
number that indicates MAs combination type
    MAtype=min(floor(typenumber*8),7); %There are 7
different MA combination type
    row=MAtype+1; % row number
related to the type of Maintanance combination
    %% Set Job Maintanance Duration and Cost
    du=0;
    sp=0;
    wf=0;
    flag=0;
    rate=1;
    NumberOfCombinesMAs=sum(MA_Types(row,:)); % Total number
of MAs concurretly implemented
    if NumberOfCombinesMAs>0
            rate=MADUDR (NumberOfCombinesMAs); Obtain the
discount rate of MAs duration
            end
    Job(j).MA(:,i)=MA_Types(row,:); % Assign
MA type to its struct
    % Retreive Data from MAs
        for k=1:L
                        flag=MA_Types(row,k);
                        if flag==1
                    du=MA(k).Du(rep,i);
                    sp=MA(k).sp(1,i);
                    wf=MA(k).wf;
                    MA (k).Level(1,i)=MA(k).MaxLevel;
                    MA(k).count(1,i)=MA(k).count(1,i)+1;
                    Job(j).MADu(rep,i)=Job(j).MADu(rep,i)+du*rate;
Job(j).MACost(rep,i)=Job(j).MACost(rep,i)+du*rate*wf+sp;
            end
            end
        end
    end
```

nmodel. Job=Job;
nmodel.MA=MA;
end

## VITA

Javad Seif was born on July 29, 1984 in Tehran, Iran. He graduated from the University of Tabriz in 2008 with a Bachelor of Science degree in Industrial Engineering (Manufacturing Concentration). Then he worked in industry for a few years. In his industry experiences he developed software engineering skills and designed and implemented industrial information systems, such as maintenance management and material requirements planning systems. In 2011, he attended the University of Tehran and started working on various problems in maintenance optimization, while he continued working in industry. His research work in the University of Tehran led to three journal publications, and in August 2013, he received a Master of Science degree in Industrial Engineering.

In August 2014 he moved to the U.S. and started his Ph.D. program in Industrial Engineering in the University of Tennessee, Knoxville (his third UT school!). At the same time, he started working as a Graduate Research Assistant in the University of Tennessee Space Institute (UTSI) in Tullahoma where he spent most of his time on research. During his work at UTSI, he led and completed eight research projects independently, and collaborated in three research projects led by his colleagues. These research projects are currently published or under-review in the conference proceedings and scientific journals of industrial engineering. He has received a number of awards, including the Best Student Paper award in Construction Division at 2017 IISE Annual Conference. He has also obtained a Graduate Certificate in Reliability and Maintainability Engineering for which he has had additional training in data-centric analytics.

