

**A Multiscale Manufacturing Operations Management Study
from Discrete-event Dynamics to Machining Dynamics by
Machine Learning based Modeling and Optimization**

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

Meiheng Ren
December 2025

Copyright © 2025 by Meiheng Ren

All Rights Reserved.

ACKNOWLEDGEMENTS

First and foremost, I am sincerely thankful to my advisor, Dr. Tony Shi, for your guidance and continuous support throughout my research life. Your supervision shaped not only my research work but also influenced my outlook on life. Thank you to Dr. Tony Schmitz for your support and technical insights, which laid the solid foundation of machining dynamics component of my research. Next, thank you to all my committee members, Dr. Bing Yao, Dr. Hongyu Zheng, and Dr. Haochen Li, for all your time and constructive feedback, which significantly contributed to the improvement of my research work.

I would like to acknowledge the funding supports provided by DoD Southeastern Advanced Machine Tools Network (SEAMTN), NSF Engineering Research Center - Hybrid Autonomous Manufacturing, Moving from Evolution to Revolution (HAMMER), and University of Tennessee Knoxville (UTK) AI Tennessee Initiative for providing opportunities to make my research possible. Furthermore, I am grateful to the Machine Tool Research Center (MTRC) for the opportunity to gain more knowledge about machine tools and for sharing valuable experimental data.

Thank you to my colleagues, Mason Ma, Victor Wu, Dr. Christopher Tyler, and Dr. Jaydeep Karandikar. Your support was invaluable in helping me overcome challenges throughout my research studies. I would not have made it this far without your assistance and cooperation.

Finally, I wish to express my sincere thanks to my parents, my little sisters, and all my friends for your unconditional love, understanding, and companionship. Your support has been my greatest source of motivation throughout this journey.

ABSTRACT

This dissertation advances the field of manufacturing operations management through machine learning based modeling and optimization across multiple scales. Starting from the scale of a factory, the classical customer order scheduling problem with minimized completion time is investigated by automatically designing scheduling/dispatching rules via a novel genetic programming-based machine learning method. At this scale, discrete-event dynamics under consideration includes customer order compositions and sequences while treating the whole factory as a single machine. Furthermore, a zoom-in perspective on the factory leads to additional challenges at the scale of a single machine tool, where tooling and operations cost of the machine tools must be considered. By incorporating machining dynamics, including machining stability and tool life, into the discrete-event dynamics for customer order scheduling, a learning-based cost function and a set of new optimization models and algorithms are developed to minimize machine tool operations costs. Lastly, by deeply diving into the machining process itself, the physics of machining dynamics are investigated. The second-order, coupled, delayed differential equations of motion governing machining dynamics are discovered by a cutting mechanics-based machine learning modeling method, that integrates existing physics in cutting mechanics and unknown physics in data to achieve automated model discovery. Taking these studies as a combined agenda, this dissertation establishes a systematic investigation linking operations management at different scales, which will contribute to both the fundamental understanding and practical applications towards smart manufacturing.

TABLE OF CONTENTS

Chapter 1 GENERAL INTRODUCTION	1
Chapter 2 A Learning-based two-stage optimization method for customer order scheduling	5
2.1 Introduction.....	5
2.2 Problem Definition and Formulation	9
2.3 Proposed Learning Method: Feature-enhanced Genetic Programming.....	12
2.3.1 Main Idea of the Proposed FEGP Method.....	13
2.3.2 Feature and Function Design for FEGP to Construct Dispatching Rules.	17
2.3.3 Proposed Fitness Evaluation Function.....	25
2.3.4 Genetic Operators with Duplicate Removal	26
2.3.5 Algorithm Procedure of the Proposed FEGP Method	28
2.4 Two New Dispatching Rules Learned by FEGP	34
2.4.1 New Dispatching Rule: LDR-I	34
2.4.2 New Dispatching Rule: LDR-II.....	40
2.5 Development of Two-stage Method	45
2.6 Computational Experiments.....	52
2.6.1 Testing Instances.....	53
2.6.2 Comparison Results of LDRs and Well-known Dispatching Rules	56
2.6.3 Comparison Results of LDR-AS and State-of-the-art Methods	62
2.7 Conclusions.....	71

Chapter 3 Integration of discrete-event dynamics and machine dynamics for machine	
tool: modeling, analysis and algorithm	73
3.1 Introduction.....	73
3.2 Problem Setting.....	76
3.2.1 Discrete-event Dynamics: Order and Part Setting	76
3.2.2 Machining Dynamics: Spindle Speed and Tool Life Setting.....	78
3.2.3 Production and Cost Settings	79
3.3 Learning-based Cost Function for Machine Tool.....	80
3.3.1 A Machine Learning Model for Tool Life Prediction	80
3.3.2 Calculation of Learning-based Cost Function	83
3.3.3 Illustration of Cost Saving by the Proposed Integration	85
3.4 Integrated Optimization of Cost and Makespan	87
3.4.1 Model with Only Cost Consideration	87
3.4.2 Decomposition-based Greedy Algorithms.....	89
3.4.3 Integration with Makespan	90
3.5 Integrated Optimization of Cost and Due Dates	90
3.5.1 Modeling with Completion Time Variables	91
3.5.2 Modeling with Linear Ordering Variables.....	92
3.5.3 Modeling with Positional Variables.....	96
3.6 Computational Experiments.....	99
3.6.1 Data Generation	100
3.6.2 Results for Problem IDM-C.....	102

3.6.3	Results for Problem IDM-CM	105
3.6.4	Results for Problem IDM-CD	108
3.7	Conclusion, Discussion and Future Work	109
Chapter 4 A cutting mechanics-based machine learning modeling method discover		
	governing equations of machining.....	113
4.1	Introduction.....	113
4.2	Proposed Cutting Mechanics-based Machine Learning Modeling Method ...	116
4.2.1	Establish Cutting Mechanics-based Modeling Structure by First Principle	118
4.2.2	Discover DAEs from Data by Machine Learning.....	122
4.3	Simulation Experiments.....	127
4.3.1	Simulation Experiments Setting	128
4.3.2	Case I: Milling with Time Delayed Effect in Forces	130
4.3.3	Case II: Milling with Time Delayed Effect, Nonlinear Process Damping	
	and Edge Force	139
4.4	Conclusions.....	146
Chapter 5 CONCLUSIONS.....		148
LIST OF REFERENCES.....		150
VITA.....		168

LIST OF TABLES

Table 2.1. Notation for the proposed FEGP.....	14
Table 2.2. Designed features of problem COS for FEGP to construct dispatching rules	21
Table 2.3. Functions of the proposed FEGP for problem COS.....	24
Table 2.4. Parameter settings of the proposed FEGP.....	33
Table 2.5. Functions and features used in LDR-II	41
Table 2.6. Well-known dispatching rules for customer order scheduling	55
Table 2.7. ARPD comparison of LDR with well-known dispatching rules on Test-1. ...	57
Table 2.8. ARPD comparison of LDR with well-known dispatching rules on Test-2. ...	58
Table 2.9. ARPD comparison of LDR with well-known dispatching rules on Test-3. ...	59
Table 2.10. Number of best found solutions by LDR and the selected dispatching rules.	60
Table 2.11. Comparison of MILP, PSO, GA, PSA and LDR-AS on Test-1.	65
Table 2.12. Comparison of MILP, PSO, GA, PSA and LDR-AS on Test-2.	66
Table 2.13. Comparison of MILP, PSO, GA, PSA and LDR-AS on Test-3.	67
Table 2.14. Number of best found solutions by PSA and LDR-AS	69
Table 3.1. Numerical results of solving problem IDM-C by the expected cost method (ECM) [1], solving models M1 and M1-LP, and the exact algorithm DG.....	103
Table 3.2. Numerical results of solving problem IDM-CM by expected cost method (ECM) [1], and solving model M2	106
Table 3.3. Numerical results of problem IDM-CD by solving models M3-1, M3-2-L and M3-3-L.....	107

Table 4.1. Number of correct DAEs (A value) discovered by CMML for the milling dynamics in Case I (with only time delayed effect in force models).....	134
Table 4.2. Discovered DAEs for the milling dynamics in Case I (with only time delayed effect in force model for $\Omega = 6000$ rpm in Table 4.1).....	135
Table 4.3. MAPE of model coefficients of correctly discovered governing equations in Table 4.2 by CMML for Case I.	137
Table 4.4. Number of correct DAEs (A value) discovered by CMML for the milling dynamics in Case II with time delayed effect, nonlinear process damping, and edge effect in the force models.....	142
Table 4.5. MAPE of model coefficients of correctly discovered governing equations in Table 4.5 by CMML for Case II.	143
Table 4.6. MAPE of model coefficients of correctly discovered governing equations in Table 4.5 by CMML for Case II.	144

LIST OF FIGURES

Figure 2.1. Flow chart of the proposed FEGP.	16
Figure 2.2. An illustrative example on tree representation of dispatching rule ECT.	23
Figure 2.3. Illustration of subtree mutation and crossover operators.....	35
Figure 2.4. Tree representation of LDR-I.	36
Figure 2.5. Illustrative example of solutions by LDR-I and ECT.....	42
Figure 2.6. Tree representation of LDR-II.	43
Figure 2.7. Flow chart of the proposed two-stage optimization method LDR-AS.....	47
Figure 2.8. Comparison of LDRs and NEW.....	61
Figure 2.9. Comparison of PSA and LDR-AS on Test-1, Test-2, Test-3.	68
Figure 3.1. Stability lobe diagram (SLD) to determine the stable and unstable cutting (chatter) by choosing parameters spindle speed (Ω) and axial depth of cut (b).	77
Figure 3.2. Probability of tool worn as a function of cutting speed (logarithmic/original scale in left/right panel) from the trained logistic model; the train data points are shown in light blue, while the test data points in dark blue (tool not worn) and red cross (tool worn). The gray scale indicates the probability of tool worn with a cutting speed (horizontal axis) and cutting time (vertical axis). Good extrapolation outside of the train data space of cutting speed can be obtained, as shown by the dark blue dots and red cross. This extrapolation capability enables the accurate prediction of tool life for wide range of cutting speeds, which will be used in the experiments. Figures are reproduced from [1].....	82

Figure 3.3. Illustrative example for choosing different speeds to process multiple parts of an order by the expected cost method in [1] and solving our developed model M1 in Section4. By choosing slower speed, we process one more part per tool edge in (b) in comparison with (a). The last three parts on the 4th too; edge of (a) are thus moved back to the 3rd tool edge. This leads to saving the last tool edge and reducing the waste time for each tool edge in (a), and importantly, reducing the cost from \$244.18 to \$233.14. See details in Section 6.	86
Figure 3.4. The comparison of cost reduction ratio (CRR) for problem IDM-C. M1, M1-LP and DG are overlapped as they all obtain the optimal solutions.	104
Figure 4.1. Schematic illustration of two fundamental principles of cutting mechanics: 1) time delayed effect, and 2) dynamic cutting force generation and projection, The resultant dynamic cutting forces excite structural dynamics to vibrate on machine tool's coordinate.....	117
Figure 4.2. The stability lobe diagrams for $\Omega = 6000$ rpm in Table 4.2. The stability lobes under noises 0.01%, 0.1%, 1%, 10% almost perfectly overlap with stability lobes by exact governing equations.	136
Figure 4.3. Both (a) and (b) show the time domain displacement (on x direction) and the Poincaré map of discovered equations in Table 4.2 using parameters $\Omega = 6000$ rpm and $b = 2$ mm. The single group of red points from once-per-tooth sampling shows the cut is stable.....	138
Figure 4.4. Both (a) and (b) show the time domain displacement (for x direction) and the Poincaré map of discovered equations in Table 4.5 using parameters $\Omega = 6000$ rpm	

and $b = 2$ mm. The single group of red points from once-per-tooth sampling shows
the cut is stable..... 145

CHAPTER 1

GENERAL INTRODUCTION

In modern manufacturing enterprises, operations management involves planning, controlling, and optimizing production processes. The overarching goal is to ensure high product quality, high operational efficiency, and low operational cost. To achieve this goal, this dissertation investigates manufacturing operations management across multi scales, spanning from customer order scheduling in discrete-event dynamics at the scale of factory, to machine tool cost of multiple parts production at scale of machine tool, to machining modeling at scale of machining dynamics, thereby establishing a comprehensive study for improving operations performance of manufacturing enterprises.

Efficient completion of customer orders is essential for the profitability of any manufacturing enterprise. Customer order scheduling represents a discrete-event dynamic process and has long been an important topic in traditional Industrial Engineering research. In a competitive market, customer orders are the main profit source. Along with the increase in the volume and diversity of orders, manufacturing factories are faced with the complicated task of planning and scheduling the production processes to ensure timely delivery and efficient resource utilization. This challenge can be formally modeled as a sequencing problem with constraints of diverse practical production requirements such as due dates and order of priorities. From the perspective of manufacturing factory, this research treats the manufacturing factory as a single machine and focuses on automatically designing new dispatching rules by developing a customized genetic programming-based machine learning method, together with an adaptive local search, to generate better

sequence of customer orders for large-scale customer order scheduling problem. Such optimization is critical for improving overall operational efficiency, reducing orders completion time, and enhancing customer satisfaction within manufacturing systems.

Machine tool cost is a critical factor in manufacturing operations, comprising primarily machine tool operations costs and cutting tool costs. Both components are strongly influenced by process parameters, particularly spindle speed, which must be selected in consideration of machining dynamics to ensure stable cutting. Higher spindle speeds will shorten processing time and reduce the operating cost, whereas tool wear will be accelerated, and cutting tool cost is increased. Therefore, to optimize total machining operation cost, balancing operating cost and cutting tool cost is required. Existing research in machining dynamics has largely focused on single part production with a single machine, neglecting the complexities of multiple parts in multiple customer orders considering discrete-event dynamics. From the perspective of the machine tool, this dissertation addresses this gap by integrating machining dynamics with discrete-event dynamics to optimize total machine tool costs across multiple parts and orders. A learning-based cost function is developed, along with a set of effective mathematical models and algorithms tailored to different production requirements.

Machining dynamics and cutting mechanics are fundamental to understanding chatter, a self-excited vibration that limits productivity in metal cutting. In the machining process, the chatter phenomenon in the process of rotating and translating tool interacts with the workpiece through complex mechanics. Chatter was first formally defined as a scientific research question by Frederick Taylor in 1906 in his *The Art of Cutting Metals* as “Chatter

is the most obscure and delicate of all problems facing the machinist, and in the case of castings and forgings of miscellaneous shapes probably no rules or formulas can be devised which will accurately guide the machinist in taking the maximum cuts and speeds possible without producing chatter”. By the 1950s, researchers developed a classical model using delay differential equations, which couples machine tool structural dynamics with the cutting force at the tool tip and explains chatter through the regenerative time-delay effect. Although this model captures the fundamentals of machining dynamics, its accuracy is limited because it relies on a linear cutting force approximation, despite the inherently nonlinear nature of machining. Motivated by the limitation, this dissertation aims to discover nonlinear models of machining dynamics to advance machining dynamics modeling in practice. A cutting mechanics-based machine learning modeling method is proposed that integrates existing knowledge of machining physics and unknown physics in data to achieve nonlinear model discovery of machining dynamics.

The remainder of the dissertation is organized as follows. Chapter 2 introduces the customer order scheduling problem and presents a detailed discussion of the proposed learning-based two-stage optimization method. The effectiveness of the proposed approach is evaluated through extensive computational experiments. Chapter 3 addresses the integrated problem of discrete-event dynamics and machining dynamics to study machine tool cost and presents a set of effective optimization models and algorithms tailored to various production requirements. The proposed models are validated through comprehensive computational experiments conducted under diverse problem settings. Chapter 4 focuses on the model discovery of machining dynamics. The proposed cutting

mechanics-based machine learning modeling method is introduced and validated through high-fidelity time domain simulation experiments. Chapter 5 concludes the dissertation.

CHAPTER 2

A LEARNING-BASED TWO-STAGE OPTIMIZATION METHOD FOR CUSTOMER ORDER SCHEDULING

This paper addresses the customer order scheduling problem in parallel production environment commonly appearing in the pharmaceutical and paper industries. To achieve the research objective of minimizing the total completion time of orders, a learning-based two-stage optimization method is proposed, comprising a learned dispatching rule in the first stage and an adaptive local search in the second stage. Numerical results demonstrate the effectiveness of the learned dispatching rule and indicate the proposed two-stage optimization method significantly outperforms state-of-the-art methods in the literature.

2.1 Introduction

This paper studies the customer order scheduling (COS) problem in parallel machine environment to minimize the total completion time of all the customer orders. The manufacturing facility consists of multiple machines and each machine is dedicated to produce one particular type of product. Each customer order is composed of several individual products each with a demand that needs to be processed by a dedicated machine. Since the products of an order are independent with each other, they can be processed on different machines in parallel. An order is completed when all its included products are completed. The objective is to find a schedule to minimize the total completion time of the orders.

Problem COS has a large number of practical applications in manufacturing. Typical industrial applications include equipment maintenance and repair [2], finishing paper processing [3] and pharmaceutical packaging [4]. Moreover, problem COS can also be regarded as a degenerate case of two-stage assembly scheduling problem if ignoring the assembly processing time in the second stage, such as part kitting [5] and semi-product manufacturing [6]. With the increasingly diversified demands from the customers, modern manufacturing is developing rapidly towards mass customization under the make-to-order production manner. To avoid additional handling and shipment cost, most of the customers prefer to receive all kinds products included in the order in a single shipment. This shifts the focus of attention from the availability of individual products to that of the holistic order for both the producers and the customers.

In the literature, problem COS has attracted a lot of attentions since first studied in [7]. Many studies have considered this problem with configuration of parallel machines as that in this paper. Several studies focusing on the complexity analysis have shown that problem COS is strongly NP-hard even for two machines [8-10]. Blocher and Chhajed studied this problem to minimize the total completion time of orders with identical parallel machines [11]. Several heuristics were proposed by considering both the order sequence and job placement, and two lower bounds were also developed. With the same setting, Yang and Posner proposed a heuristic with tight worst-case bound $2 - 1/m$ where m is the number of machines [5]. Additionally, two heuristics that are examined with the tight worst-case bounds $6/5$ and $9/7$ were developed for two machines case. In [2], two heuristics, namely earliest completion time first (ECT) and shortest

processing time first applied to the machine with the largest load (SPTL), were proposed where the former was proved to have the worst-case bound m . The authors also analyzed the other three dispatching rules, namely shortest total processing time first (STPT), shortest maximum processing time first (SMPT) and smallest maximum completion time first (SMCT) proposed by Wang and Cheng for arbitrary m machines [12]. Along this line with problem COS, many studies focus on the different objective functions. In [13], a quadratic programming formulation was proposed to minimize the total weighted completion time. Some other objectives are minimizing the total weighted completion time [12, 14, 15], minimizing the maximum lateness and number of late orders [3], minimizing the total tardiness [16], and multi-objective of minimizing the linear sum of the total flowtime and the maximum tardiness [17]. In addition, many variants of problem COS have been studied. In [18], a branch-and-bound algorithm was proposed to solve the order scheduling problem with learning effect to minimize the total tardiness. Some studies also investigate the variant with order release times [19, 20]. A new order scheduling on a product-service system with time windows is studied in [21], where an order is composed of a product processed by a manufacturing plant and a service from the service center. In [22], the order scheduling problem with rejection was investigated to minimize a linear sum of the maximum completion time of the accepted orders and the total penalty of the rejected orders. De Athayde Prata et al. proposed two matheuristics to solve the order scheduling problem with sequence-dependent setup times [23]. Other variants in terms of different machine environments have also been addressed including single machine [24], two-stage assembly flowshop [25], parallel batch processing

machines [26] and permutation flowshop [27]. In addition, metaheuristics are commonly used to solve problem COS and its variants, including genetic algorithm (GA) [28-31], particle swarm optimization (PSO) [17, 32], simulated annealing [18, 21, 33], artificial bee colony algorithm [20, 27, 34], tabu search [21, 35] and differential evolution [36]. For more details about problem COS, the readers can refer to the survey paper [37].

Recently, Framinan and Perez-Gonzalez proposed an effective dispatching rule for problem COS by utilizing a look-ahead indicator to assess the impact of both the scheduled and unscheduled orders [38]. This dispatching rule denoted by NEW outperforms the existing dispatching rules in the literature. Based on this new dispatching rule, a greedy search algorithm was also developed to solve problem COS more efficiently. Riahi et al. proposed a new constructive heuristic by utilizing the processing times in multiple ways [39]. This heuristic was then integrated into a newly developed perturbative search algorithm (PSA). PSA achieves a powerful computational performance on the total of 720 benchmark instances of problem COS and has updated the best solutions for almost all those instances.

Although existing methods in the literature have been proposed for solving problem COS efficiently, there is still a lot of room for improvement especially for the large-scale problems. This paper aims to develop a learning-based optimization approach to solving problem COS. The main contributions are summarized as follows. We develop an efficient feature-enhanced genetic programming (FEGP) method for the automatic design of high-quality dispatching rules for problem COS in an off-line learning manner.

Two new dispatching rules learned by the proposed FEGP method outperform the existing well-known dispatching rules manually designed in the literature. Utilizing the high-quality initial solutions provided by the new dispatching rules, we further develop an adaptive local search algorithm to form a two-stage optimization method denoted by LDR-AS for problem COS. On solving a total of 1080 benchmark instances, LDR-AS achieves the best computational performance compared with the state-of-the-art methods and is capable of finding the best feasible solutions for 1016 instances.

The remainder of this paper is organized as follows. Section 2 presents the problem definition and a mixed-integer linear programming (MILP) model. In Section 3, we propose the FEGP method for the automatic design of dispatching rules for problem COS. Section 4 presents two new dispatching rules learned by the proposed EFGP method. Based on the high-quality initial solutions provided by LDR, we further design an adaptive local search algorithm as the two-stage method in Section 5. We conduct the computational experiments to test the performance of the proposed methods in Section 6. Section 7 concludes this paper.

2.2 Problem Definition and Formulation

Problem COS can be described as follows. There are a total of n available orders needed to be produced by a manufacturer for the customers. The manufacturing facility has m parallel machines with each machine dedicated to produce only one type of job. Each order from the customer is composed of at most m individual jobs, each of which should be manufactured exactly by a dedicated machine without preemption of interruption. Each machine can process at most one job at a time and the jobs in an order

can be processed by multiple machines simultaneously. The completion time of each order is defined by the time at which all jobs in the order are completed. The objective of problem COS is to find a schedule to minimize the total completion time of the orders.

An important optimality property of problem COS with minimization of total weighted completion time is that there exists an optimal schedule in which the permutation of orders on each machine is the same [11]. Based on this optimality property, Shi et al. first formulated problem COS as an mixed-integer quadratic programming (MIQP) model, and then an MILP formulation was derived through applying the reformulation technique (by introducing a new decision variable α_{jk}^t) on the proposed MIQP formulation [13]. This derived MILP formulation achieves the tightest LP-relaxation compared with the order models for this problem.

However, there is no physical meaning and illustration provided for the new introduced variable α_{jk}^t and the MILP formulation in [13]. In this paper, we introduce the same MILP formulation based on the following two purposes. First, we define the decision variable α_{jk}^t by giving its the physical meaning and show that we can formulate this problem directly as the same MILP model rather than the reformulation technique in [13]. Second, with the development of optimization technique, the state-of-the-art commercial optimization solver like CPLEX 12.8 integrates plenty of efficient algorithms and has a strong power to search the best feasible solutions for a given MILP model. Therefore, we want to compare the solution quality of the proposed method with the best feasible solutions of this MILP model solving by CPLEX 12.8. The notations and decision variables are given as follows.

Notations:

N : set of orders, $N = \{1, 2, \dots, n\}$;

M : set of machines, $M = \{1, 2, \dots, m\}$;

p_j^t : job process time in order j on machine t , where $j \in N, t \in M$.

Decision Variables:

x_{jk} : $x_{jk} = 1$, if order j is at the position k when the jobs are processed on machines.

Otherwise, $x_{jk} = 0$, where $j, k \in N$;

C_j : completion time of order j , where $j \in N$;

α_{jk}^t : $\alpha_{jk}^t = \sum_{k'=1}^k \sum_{j=1}^n x_{jk'} p_j^t - p_j^t$, i.e., the starting time of order j at position k on machine t if $x_{jk} = 1$, and otherwise, $\alpha_{jk}^t = 0$, where $j, k \in M, t \in M$.

Based on the above notations and variables, problem COS can be formulated as the following MILP model.

$$\text{(MILP)} \quad \min \sum_{j \in N} C_j \quad (1)$$

$$\text{s. t.} \quad \sum_{k \in N} x_{jk} = 1, \quad \forall j \in N, \quad (2)$$

$$\sum_{j \in N} x_{jk} = 1, \quad \forall k \in N, \quad (3)$$

$$0 \leq \alpha_{jk}^t \leq M_j^t x_{jk}, \quad \forall j, k \in N, t \in M, \quad (4)$$

$$\sum_{j=1}^n \alpha_{jk}^t = \sum_{k'=1}^k \sum_{j=1}^n x_{jk'} p_j^t - \sum_{j=1}^n x_{jk} p_j^t, \quad \forall k \in N, t \in M, \quad (5)$$

$$C_j \geq \sum_{k=1}^n \alpha_{jk}^t + p_j^t, \quad \forall j \in N, t \in M, \quad (6)$$

$$x_{jk} \in \{0, 1\}, \quad \forall j, k \in N. \quad (7)$$

The objective function (1) is to minimize the total completion time of the orders. Constraints (2), (3) ensure each order can be assigned to only one position and each position can be occupied by only one order, which define a permutation of the orders. Constraints (4) ensure that $\alpha_{jk}^t = 0$ if $x_{jk} = 0$ and M_j^t is a large positive number where we set $M_j^t = \sum_{i=1}^n p_i^t - p_j^t$. Constraints (5) define the starting time equation for any order assigned at the position k of machine t . Constraints (6) indicate the completion time of order j on the machine t is greater than or equal to the summation of its starting time and processing time. Based on the constraints (2)–(4), it is simple to check that the left-hand side $\sum_{j=1}^n \alpha_{jk}^t$ of constraints (5) denotes the starting time of the order assigned at the position k of machine t and $\alpha_{jk}^t = \sum_{k'=1}^k \sum_{j=1}^n x_{jk'} p_j^t - p_j^t$ if $x_{jk} = 1$. According to this, $\sum_{k=1}^n \alpha_{jk}^t$ indicates the starting time of the order j on the machine t and thus constraints (6) hold. Constraints (7) are the binary variables restriction. Though the problem size that can be solved to optimality by the MILP model is small for most of scheduling problems, we will solve this model to compare the best feasible solutions obtained by CPLEX with our proposal method in Section 6.

2.3 Proposed Learning Method: Feature-enhanced Genetic Programming

In this section, we propose the *feature-enhanced genetic programming*, denoted by FEGP, for the automatic design of dispatching rules for problem COS. Genetic programming (GP) is a powerful evolution-based artificial intelligence approach for automatic design of computer programs proposed by [40]. GP can be regarded as a generalization of genetic algorithm (GA). The key focus of GP, instead of searching good

solutions from the solution space conducted by GA, is to search high quality algorithms from the algorithm space. GP iteratively evolves a population of individuals using genetic operators such as mutation and crossover to find high-quality computer programs (see reviews [41, 42]). During the last thirty years, extensive studies have been carried out, such as genetic operators [43, 44], representations [45], surrogate models [46, 47] and evaluation efficiency [48], to enhance computational performance of GP.

In the proposed FEGP method, individuals in the population are dispatching rules for problem COS. By adequately utilizing the feature information of problem COS, the proposed FEGP is able to generate high-quality dispatching rules through an offline learning manner. Next, we first present the main idea of the proposed FEGP. Then we present the designs for the important components of the proposed FEGP in detail. At the end, we present the complete algorithm procedure of FEGP. For a better presentation, we summarize the notations used in FEGP in Table 2.1.

2.3.1 Main Idea of the Proposed FEGP Method

The main idea of the proposed FEGP method is to automatically generate effective dispatching rules for problem COS through an offline learning manner. The individuals in the population of the proposed FEGP are dispatching rules. FEGP iteratively evolves a population of dispatching rules, creates new dispatching rules through mutation and crossover, and selects the dispatching rules with superior performance to form new population. For ease of computer handling in FEGP, the dispatching rules of problem COS are transferred by tree-based representation. By adequately utilizing the feature information of problem COS to construct various tree-based dispatching rules, the proposed FEGP is

Table 2.1. Notation for the proposed FEGP.

$\mathcal{P}$	Population of the dispatching rules
$\mathcal{P}_1$	Population of the offspring dispatching rules
$N_{\mathcal{P}}$	Number of dispatching rules in population $\mathcal{P}$
g_{max}	Maximum generations of FEGP evolves
DR_i	i th dispatching rule in the population, $i \in \{1, \dots, N_{\mathcal{P}}\}$
DR^*	Best evolved dispatching rule
$f(DR_i)$	Fitness value of rule DR_i
p_e	Proportion of elite rules selected to form the next generation population
p_c	Probability of crossover operator is chosen to create offspring rules; $1 - p_c$ is the probability of mutation operator is chosen;
d_{max}	Maximum depth of a tree

able to generate a large number of dispatching rules and select high-quality dispatching rules based on effective fitness evaluations. Then the learned dispatching rules can be used to solve problem COS directly. According to this idea, Figure 2.1 shows the detailed flow chart of our proposed FEGP method.

As shown in Figure 2.1, the proposed FEGP generates a population $\mathcal{P}$ of dispatching rules and selects the current best dispatching rule iteratively. The initial population of dispatching rules is first constructed using the features of problem COS. Then all the dispatching rules are evaluated using a developed fitness evaluation function and the current best dispatching rule DR^* is updated. If the maximum generation is not reached, FEGP utilizes genetic operators such as mutation and crossover to generate the next population and the algorithm iterates. When the termination condition of maximum generation holds, the best dispatching rule DR^* is output and the FEGP method stops.

There are several important components in the proposed FEGP significantly affecting the quality of the learned dispatching rules. First, the features of problem COS must be well designed such that useful problem information can be utilized to construct high-quality dispatching rules. Second, a good fitness evaluation function should be developed such that superior dispatching rules can be accurately selected. Thirdly, to maintain the diversity of the population, specific mutation and crossover operators with duplicate removal strategy should be developed such that more powerful dispatching rules are possible to be discovered. Next, we present the designs of these important components in detail.

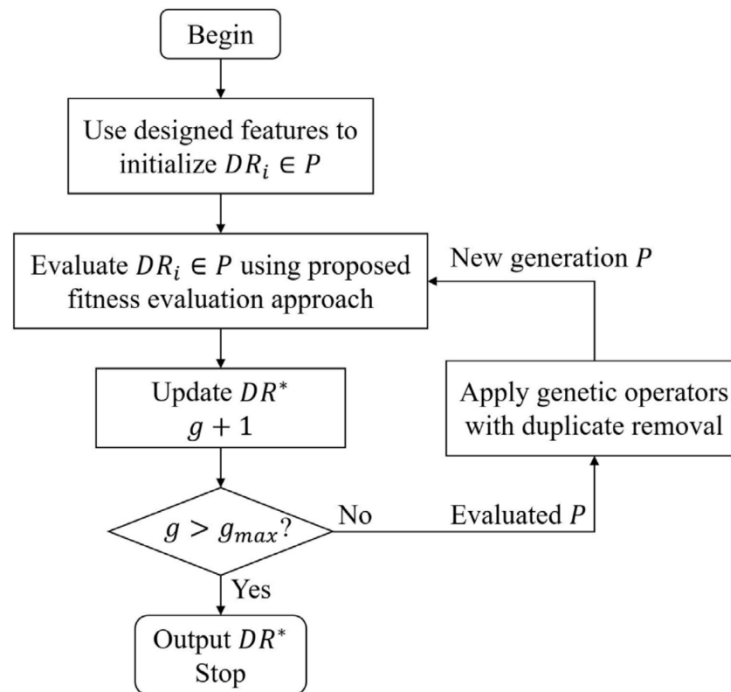


Figure 2.1. Flow chart of the proposed FEGP.

2.3.2 Feature and Function Design for FEGP to Construct Dispatching Rules

In the proposed FEGP, dispatching rules are constructed using features and functions in tree-based form. A *feature* is a problem related attribute, variable or constant that contains useful information to help solve the studied problems. A *function* can be any arithmetic operator, logic operator and mathematical function. We use tree representation to express dispatching rules for problem COS in this paper. Figure 2.2 shows an illustrative example how the well-known dispatching rule ECT proposed in [2] for problem COS can be expressed as a tree. In this example, C_k and C_j at the leaf nodes of the tree are features. Besides, operator “ $-$ ” at the internal node is the minus, which is an important part to construct the tree of ECT. Feature C_k denotes the completion time of the last order k in the partial schedule and feature C_j is the completion time of order j assuming it is selected to schedule immediately after k . By this means, $C_k - C_j$ denotes the priority of order j , and the one with the earliest completion time thus has the largest priority to be scheduled next. Obviously, the features C_j and C_k are the key factors to determine the effectiveness of the dispatching rule ECT.

Motivated by the fast and robust computational performance of the well-known dispatching rules manually developed in the literature, we conduct the feature design for the proposed FEGP method by extracting the important features from those well-known dispatching rules. Some of the well-known dispatching rules like ECT are also approximation algorithms. In general, a dispatching rule for problem COS contains one or several features that are key factors to determine its effectiveness for solving the problem. By thoroughly exploring dozens of well-known dispatching rules, including

STPT, SMPT, SMCT, ECT, SPTL [2, 12] and eight dispatching rules proposed in [39], we design eighteen useful features of problem COS and summarize them in Table 2.2. These features cover most information of the unscheduled orders and the current partial schedule in solving problem COS. For example, the total job processing time of an unscheduled order denoted by OTPT, is the key feature of rule STPT. Furthermore, we include the constants 0 and 1 to increase varied structures of the dispatching rules. As discussed in Figure 2.2, the ECT rule uses $C_k - C_j$ to calculate the priority of order j . According to Table 2.2, feature $ODCT = C_j - C_k$, thus ECT can also be expressed as $0 - ODCT_j$ to calculate the priority equivalently.

In addition, we design the functions used to organize features into dispatching rules and present them in Table 2.3. The function set consists of basic arithmetic operators, the maximum function and the ternary version of the if-then-else function (if $a \geq 0$ then b else c ; a , b and c are the related sub-expressions evolved). Involving such logical operators would enrich the search power to find more combinations of features and evolve advanced scheduling logic.

Using the designed features and functions, population $\mathcal{P}$ can be initialized by integrating well-known dispatching rules as initial individuals. We first transfer these dispatching rules into trees, then insert them into the initial population $\mathcal{P}$ of FEGP. For example, rule STPT can be transferred into a tree ($0 - OTPT$) using the developed features, where smaller OTPT will have larger priority. Then, the rest rules are randomly generated by the well known *Ramped half-and-half* method [40]. *Ramped half-and-half* creates a half of population by *full* method and the other half by *grow* method. The

full method cultivates a tree in a way that the features could only occur at nodes in the maximum depth d_{max} , and all the internal nodes are assigned with functions. The *grow* method, on the contrary, allows to select both the features and functions to assign on the nodes not at the d_{max} depth.

With the designed features in Table 2.2 and functions in Table 2.3, dispatching rules of problem COS can be constructed in the tree-based form. Consider the LDR in which all the leaf nodes grow to the maximum depth d_{max} , it will give the worst-case time complexity for any LDR generated by FEGP. We present this complexity analysis in the following lemma.

Lemma 1. For a given maximum tree depth d_{max} , any learned dispatching rule by the proposed FEGP method is a polynomial-time algorithm with the worst-case time complexity of $O(mn^3/2 + n_f n^2/2 + m^2 n)$, where $n_f = (3^{d_{max}-1} - 1)/2$ is the maximum number of functions in the tree-based LDR, m is the number of machines and n is the number of orders.

Proof. Let $\pi = \{\pi(1), \pi(2), \dots, \pi(n)\}$ be an order sequence, where $\pi(k)$, $k = 1, \dots, n$ is the order at position k . LDR constructs a sequence by selecting one order a time from the unscheduled orders and appending at the end of current partial sequence, denoted by $\pi = \{\pi(1), \pi(2), \dots, \pi(k-1)\}$, $\pi = \phi$ when $k = 1$. This implies for each position $k = 1, \dots, n$, LDR needs to calculate the priority of $n - k + 1$ unscheduled orders, which gives $\sum_{k=1}^n (n - k + 1) = n(n + 1)/2$ times of priority calculation using LDR. Next, we analyze the worst-case time complexity for one-time priority calculation using LDR. Given a maximum tree depth d_{max} , the worst-case time complexity of LDR is obtained if all the

leaf nodes of LDR grow to d_{max} and all the internal nodes from depth 1 to $d_{max} - 1$ are ternary functions, i.e. if-then-else(a, b, c). This indicates the total number of functions in LDR is $\sum_{d=1}^{d_{max}-1} 3^{d-1} = (3^{d_{max}-1} - 1)/2$. Let $n_f = (3^{d_{max}-1} - 1)/2$, it follows n_f time is needed to calculate the priority of one order if the features are ready. Next, we consider the feature with the maximum time complexity in the designed features in Table 2.2. Feature WLCPT first finds machine $t' = \arg \min_t TCT^t$ where TCT^t is the total completion time if following the order sequence on machine t by sorting the jobs in non-decreasing order of p_j^t . This takes $O(m^2n + mn \log n)$ time. Then feature WLCPT calculates the order weight w_j by using $w_j = \sum_{t=1}^m p_j^t / \sum_{j=1}^n \sum_{t=1}^m p_j^t$, which takes time $O(mn)$. Overall, $WLCPT_j = w_j p_j^{t'}$ takes $O(m^2n + mn \log n)$ time. Since feature WLCPT is static constant, meaning they can be used repeatedly without recalculating. If all the nodes at depth d_{max} are WLCPT, it takes $O(m^2n + mn \log n + n_f n(n+1)/2) = O(m^2n + mn \log n + n_f n^2/2)$ time. However, feature ODCT needs to be recalculated every time before priority calculation. As $ODCT_j = C_j - C_k$ is obtained by applying partial sequence $\pi = \{\pi(1), \dots, \pi(k), j\}$, it takes $O(mn)$ time. Thus, if all the nodes at d_{max} are ODCT, it takes $O(n(n+1)(mn + n_f)/2) = O(mn^3/2 + n_f n^2/2)$ time. Therefore, a tree-based LDR will take worst-case time complexity if contains both WLCPT and ODCT. Overall, the worst-case time complexity for any LDR is $O(m^2n + mn \log n + n(n+1)(mn + n_f)/2) = O(mn^3/2 + n_f n^2/2 + m^2n)$ where $n_f = (3^{d_{max}-1} - 1)/2$. Obviously, it is polynomial for any given d_{max} . ■

Table 2.2. Designed features of problem COS for FEGP to construct dispatching rules

Feature	Description
MINPT	Minimum job processing time of order j , $MINPT_j = \min_t p_j^t$
MAXPT	Maximum job processing time of order j , $MAXPT_j = \max_t p_j^t$
OTPT	Total processing time of order j , $OTPT_j = \sum_{t=1}^m p_j^t$
OTWPT	Total weighted processing time of order j , $OTWPT_j = \sum_{t=1}^m w_t p_j^t$, herein $w_t = \sum_{j=1}^n p_j^t / (\sum_{t=1}^m \sum_{j=1}^n p_j^t)$
OTCWP T	Total completion time weighted processing time of order j , $OTCWP_j = \sum_{t=1}^m w_t p_j^t$, $w_t = TCT^t / \sum_{i=1}^m TCT^i$. TCT^i is the total completion time (TCT) if following the order sequence on machine t by sorting the jobs in non-decreasing order of p_j^t
HMPT	Job processing time on the machine with the highest total processing time, $HMPT_j = p_j^{t'}$, $t' = \arg \max_t \sum_{j=1}^n p_j^t$
LMPT	Job processing time on the machine with the lowest total processing time, $LMPT_j = p_j^{t'}$, $t' = \arg \min_t \sum_{j=1}^n p_j^t$
LCPT	Job processing time on the machine with the lowest TCT, $LCPT_j = p_j^{t'}$, $t' = \arg \min_t TCT_t$. TCT^t is the same with that in OTCWP
WLCPT	Weighted job processing time on the machine with the lowest TCT, $WLCPT_j = w_j p_j^{t'}$, t' is the same with that in LCPT, $w_j = \sum_{t=1}^m p_j^t / \sum_{j=1}^n \sum_{t=1}^m p_j^t$
MAXCT	Maximum completion time of jobs in order j assuming it is selected next, $MAXCT_j = \max_t C_j^t$
MINCT	Minimum completion time of jobs in order j assuming it is selected next, $MINCT_j = \min_t C_j^t$
ODCT	$ODCT_j = C_j - C_k$, difference of completion time of order j and that of order k immediately before j , assuming j is selected next

OMS	Makespan of order j assuming it is selected next, $OMS_j = C_j - C_k^{t'}$, where k is the order immediately before j and $C_k^{t'}$ denotes the earliest completion time of jobs in order k
EXPT	Extra processing time of order j assuming it is selected next, $EXPT_j = C_j - C_k^{t'} - \max_t p_j^t$ where k is the order immediately before j and $C_k^{t'}$ is the same with that in OMS
NUO	Number of unscheduled orders
EST	Earliest completion time of current partial schedule, $EST = \min_t C^t$
LST	Latest completion time of current partial schedule, $LST = \max_t C^t$
AST	Average completion time of current partial schedule, $AST = \sum_{t=1}^m C_t / m$
Constant	0,1

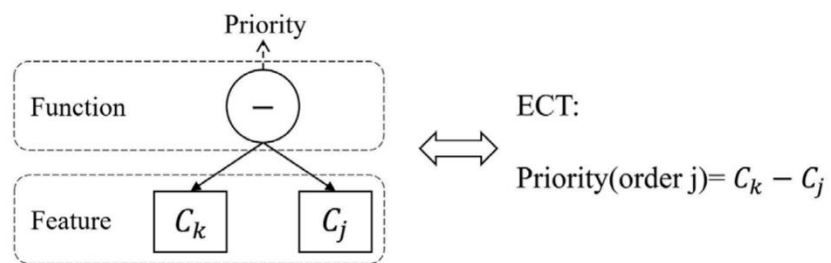


Figure 2.2. An illustrative example on tree representation of dispatching rule ECT.

Table 2.3. Functions of the proposed FEGP for problem COS

Function	Description
+	Addition, binary operator
−	Subtraction, binary operator
×	Multiplication, binary operator
÷ (or div)	Protected division, binary operator
If-then-else(a, b, c)	If $a \geq 0$ then b else c , ternary operator
max(a, b)	Maximum of a and b , binary operator

2.3.3 Proposed Fitness Evaluation Function

During the learning process of FEGP, fitness evaluation is a critical step as it determines the performance of a dispatching rule and guides the FEGP search on the space of dispatching rules. An individual with better fitness indicates it has larger probability to be selected to reproduce the next generation. We use the following equation to calculate the average relative deviation of a dispatching DR_i

$$dev_{avg}(DR_i) := \frac{1}{T} \sum_{I_t^s \in \{I_1^s, I_2^s, \dots, I_t^s\}} \frac{Obj(DR_i, I_t^s) - Obj(DR_{ref}, I_t^s)}{Obj(DR_{ref}, I_t^s)}, \quad (8)$$

where $\mathcal{J}^s = \{I_1^s, I_2^s, \dots, I_t^s\}$ is a set of data instances having the same problem size $s \in \mathcal{S}$ and T denoted the total number of instances in this problem size. $\mathcal{S}$ is a set of the total problem sizes. $Obj(DR_i, I_t^s)$ and $Obj(DR_{ref}, I_t^s)$ are the objective values by applying DR_i and the reference rule DR_{ref} on instance I_t^s , respectively. Thus, $dev_{avg}(DR_i)$ measures the average deviation between the objective values of DR_i and DR_{ref} . Herein we select ECT as DR_{ref} .

Based on the above deviation equation, we propose the ranking based fitness evaluation function of DR_i as follows

$$f(DR_i) := \sum_{s \in \mathcal{S}} rank(DR_i, \mathcal{J}^s), \quad (9)$$

where $rank(DR_i, \mathcal{J}^s)$ is the rank (ascending order) of $dev_{avg}(DR_i)$ among the individual population on solving a set of instances $\mathcal{J}^s$ that have the same problem size s . By summing up the ranks in all problem sizes $s \in \mathcal{S}$, $\sum_{s \in \mathcal{S}} rank(DR_i, \mathcal{J}^s)$ can represent the overall rank of DR_i on solving various sizes of problem instances. Thus, we define $\sum_{s \in \mathcal{S}} rank(DR_i, \mathcal{J}^s)$

as the fitness of the dispatching rule $f(DR_i)$. The pseudo-code of this proposed fitness evaluation function in Eq. (9), denoted by `FitnessEvaluate`, is given in Algorithm 2.1.

Algorithm 2.1 evaluates a DR_i using various problem instance sizes. Before evaluation, T instances for each $s \in \mathcal{S}$ are generated and used to evaluate all the dispatching rules in the population. By Hildebrandt et al., we use the same data set for each generation to reduce the impacts of data uncertainty while avoids the premature convergency of the proposed FEGP method [49]. After the ranking information $rank(DR_i, \mathcal{J}^s)$ for each problem size s is obtained, the ranking summation $\sum_{s \in \mathcal{S}} rank(DR_i, \mathcal{J}^s)$ is set to be $f(DR_i)$. The smaller $f(DR_i)$ indicates the better fitness DR_i can obtain. Such fitness evaluation function design is expected to reduce the impact of different value ranges of $dev_{avg}(DR_i)$ and focus on selecting individuals that can perform well and robustly for different problem sizes.

2.3.4 Genetic Operators with Duplicate Removal

After the fitness evaluation for each generation, genetic operators are used to select candidates from the current generation and produce new individuals for the next generation, such as selection, mutation and crossover operators [43]. In this study, we apply the tournament selection, subtree mutation and subtree crossover operators. In both the mutation and crossover, if the newly generated rules duplicate with existing rules on solving two instances with $n = 50$ and $m = 2$, i.e. having the same objective value, we abandon it and repeat applying the operators until different trees are obtained. In addition,

Algorithm 2.1 FitnessEvaluate ($\mathcal{P}$)

```
1:  Generate  $T$  data instances for each  $d \in \mathcal{S}$ 

2:  for  $s \in \mathcal{S}$  do

3:    for  $DR_i \in \mathcal{P}$  do

4:      for  $I_t^s \in \{I_1^s, I_2^s, \dots, I_T^s\}$  do

5:         $Obj(DR_i, I_t^s) \leftarrow$  apply  $DR_i$  on instance  $I_t^s$ 

6:      end for

7:      Calculate  $dev_{avg}(DR_i)$  by Eq. (8)

8:    end for

9:     $rank(DR_i, \mathcal{J}^s) \leftarrow$  sort  $DR_i$  among  $\mathcal{P}$  in ascending order of  $dev_{avg}(DR_i)$ 

10:  end for

11:  for  $DR_i \in \mathcal{P}$  do

12:     $f(DR_i) \leftarrow \sum_{s \in \mathcal{S}} rank(DR_i, \mathcal{J}^s)$ 

13:  end for

14:  return  $f(DR_i)$  for all  $DR_i \in \mathcal{P}$ 
```

depth of the new trees is still constrained by d_{max} ; otherwise repeat applying the operators. Details of the three operators are presented as following.

Tournament () : Randomly select a predefined tournament size n_t dispatching rules from $\mathcal{P}$, return the rule with the best fitness value.

Mutation (DR_1^p) : One parent tree DR_1^p is modified by randomly choosing one part of it and replacing with a newly generated subtree. Check for constraints of d_{max} and duplicates. For example, in Figure 2.3a, the left child node *LST* of the parent tree is picked and replaced with a newly generated subtree ($NUO \times HMPT$) to reproduce one offspring.

Crossover (DR_1^p, DR_2^p) : Two parent trees DR_1^p, DR_2^p are recombined by randomly picking a node in each tree and swapping over the subtree connected to the picked node. Check for constraints of d_{max} and duplicates. For example, in Figure 2.3b, the subtree *OTPT* of parent 1 and subtree ($EST \times OMS$) of parent 2 are chosen and swapped over to create two new offsprings.

2.3.5 Algorithm Procedure of the Proposed FEGP Method

Based on the above developed components of the proposed FEGP method, we present the detailed algorithm procedure of FEGP in Algorithm 2.2.

Algorithm 2.2 runs in the following procedure. Using the problem features and functions designed in Section 3.2, we first express several well-known dispatching rules into trees and insert these trees into the initial population. The rest individuals in population $\mathcal{P}$ are randomly are randomly initialized with the tree depth no exceeding the maximum

Algorithm 2.2 FEGP

```
1:  Set  $DR^* \leftarrow null$  and  $f(DR^*) \leftarrow +\infty$ 
2:  Invoke the designed features and functions
3:  Initialize  $\mathcal{P}$ 
4:   $g \leftarrow 0$ 
5:  while  $g < g_{max}$  do
6:     $\mathcal{P}_1 \leftarrow \emptyset$ 
7:    Apply FitnessEvaluate ( $\mathcal{P}$ ) to obtain  $f(DR_i)$  for each  $DR_i$ 
8:     $f(DR^*) \leftarrow \min_{i=1}^{N_{\mathcal{P}}} f(DR_i)$ 
9:     $DR^* \leftarrow \arg \min_{i=1}^{N_{\mathcal{P}}} f(DR_i)$ 
    Sort  $DR_i$  in  $\mathcal{P}$  in ascending order of  $f(DR_i)$ ; select the first  $p_e \times N_{\mathcal{P}}$  rules
10:    $DR_1, DR_2, \dots, DR_{p_e \times N_{\mathcal{P}}}$ 
11:    $\mathcal{P}_1 \leftarrow \mathcal{P}_1 \cup \{DR_1, DR_2, \dots, DR_{p_e \times N_{\mathcal{P}}}\}$ 
12:   while  $N_{\mathcal{P}_1} < N_{\mathcal{P}}$  do
13:     Generate a random number  $rand \in [0,1]$ 
14:     if  $rand < p_c$  then
15:        $DR_1^p$  and  $DR_2^p \leftarrow$  apply Tournament () twice to select two parents
        $DR_1^p$  and  $DR_2^p \leftarrow$  apply Crossover ( $DR_1^p, DR_2^p$ ) to create two
16:       offspring rules
17:        $\mathcal{P}_1 \leftarrow \mathcal{P}_1 \cup \{DR_1^c, DR_2^c\}$ 
18:     else
```

```

19:       $DR_1^p \leftarrow \text{apply Tournament} () \text{ to select one parent rule}$ 
20:       $DR_1^m \leftarrow \text{apply Mutation} (DR_1^p) \text{ to create one offspring}$ 
21:       $\mathcal{P}_1 \leftarrow \mathcal{P}_1 \cup \{DR_1^m\}$ 
22:      end if
23:  end while
24:   $\mathcal{P} \leftarrow \mathcal{P}_1$ 
25:   $g \leftarrow g + 1$ 
26: end while
27: return  $DR^*$ 

```

tree depth d_{max} . Then, we use the proposed fitness evaluation function `FitnessEvaluate()` developed in Section 3.3 to evaluate the performance of all dispatching rules based on various problem instance sizes and the performance ranking information of DR_i among population $\mathcal{P}$. After the fitness evaluation, the enhanced genetic operator `crossover()` and `mutation()` designed in Section 3.4, are applied to generate different offsprings with duplicate removal strategy for the next generation. The best rule DR^* is returned when the maximum generation is reached.

Table 2.4 summarizes the training parameters setting of the proposed FEGP method. The systematical parameter tuning of GP has been thoroughly conducted in the literatures [40, 49, 50], and those parameter settings are widely adopted by the GP-related studies. The population size is set to be 600 to ensure the diversity of the population. For the number of generations, we run the proposed FEGP with 300 generations for 3 times and the best dispatching rule at generation 50, 100, 150, 200, 250 and 300 are selected to test the performance, respectively. The testing results show the best dispatching rule at generation 200 obtains the best performance, while dispatching rules after generation 200 exhibit overfitting. The number of evaluations per individual is highly problem specific [51]. The choice of this parameter needs to be high enough to avoid over specialization within a single generation while it needs to be low enough to keep the experiments computationally feasible. Based on our proposed fitness evaluation function, we evaluate each dispatching rule using 120 randomly generated instances, which are further divided into 12 groups and each group is a combination from $n \in \{50, 100, 200\}$ orders and $m \in \{2, 5, 10, 20\}$ machines with 10 instances. Among the 10 instances of each

combination, 5 instances are generated following the processing time generation rule of Test-1 and the rest 5 following that of Test-2. It should be noted that the generations of Test-1, Test-2 and the following Test-3 are summarized in Section 6.1. Similar to [49], common random numbers are used for all dispatching rules in a certain generation while different random numbers are used for different generations to avoid overfitting and reduce the impact of data uncertainty simultaneously. As FEGP is trained using randomly generated data following generation rule of Test-1 and Test-2, the new dataset Test-3 randomly generated following different generation rule can be used to cross-validate the performance of LDRs. The crossover proportion, mutation proportion, selection method, initialization method, and the maximum tree depth 17 are following recommendations from [40] and are mostly the default settings of ECJ,¹ the evolutionary computation framework we use for our FEGP implementation. We also compared three sets of crossover and mutation proportions (85%, 15%)(95%, 5%) and (99%, 1%) with the default (90%, 10%). We found these settings do not lead to improved performance and thus retained using the default settings. While greater maximum tree depth with 17 can extend the search space of FEGP, we also choose this maximum tree depth to 5 to make the learned dispatching rules easier to analyze. Based on these settings and our computing environment, the proposed FEGP method takes the training time of 1.5 h to obtain the first new dispatching rule with the maximum tree depth of 5, of about 13 h to obtain the second new dispatching rule with the maximum tree depth of 17.

Table 2.4. Parameter settings of the proposed FEGP

Population size	600
Generation	200
Number of evaluations per rule	120
Crossover proportion	90%
Mutation proportion	10%
Elites	5% of population size
Selection method	Tournament selection (Size: 7)
Initialization	Ramped half-and-half (Depth range: [2,6])
Maximum tree depth	5 or 17

2.4 Two New Dispatching Rules Learned by FEGP

In this section, we present two new dispatching rules for problem COS. These two new dispatching rules are automatically generated by the proposed FEGP method and all the related experimental details are presented in Section 6.

2.4.1 New Dispatching Rule: LDR-I

The first new dispatching rule, denoted by LDR-I, is obtained by setting the maximum depth of the tree representation to be 5 during the training process of the proposed FEGP method. The best learned dispatching rule in the final generation is selected as LDR-I. Figure 2.4 shows the tree structure of LDR-I. The corresponding closed-form of LDR-I to calculate the priority of an order j is:

$$\begin{aligned}
 \text{Priority}(j) = & \text{MAXPT}_j \\
 & / \left((NUO + ODCT_j) \times NUO \times OTWPT_j \times (MINPT_j + OMS_j) \right. \\
 & \times \left((HMPT_j + NUO) \times ODCT_j^2 \right. \\
 & \left. \left. + NUO \times OTWPT_j \times (NUO + MAXPT_j) \right) \right) \quad (10)
 \end{aligned}$$

In Eq. (10), all the features of problem COS such as the number of unscheduled orders NUO, are given in Table 2.2. By plugging in the value of each feature into Eq. (10), the order with the highest priority from the unscheduled orders is selected to schedule next. Let $\pi = \{\pi(1), \pi(2), \dots, \pi(n)\}$ be an order sequence where $\pi(k)$ is the order at position k , C_j^t be the completion time of order j on machine t , and w_t be the weight of machine t . Then detailed pseudo-code of LDR-I is presented in Algorithm 2.3.

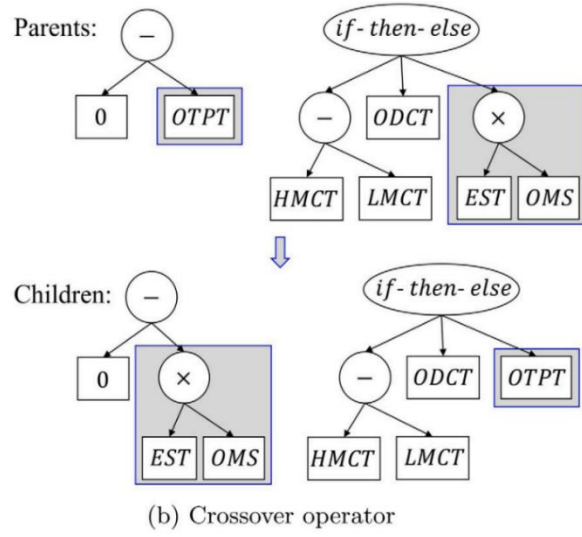
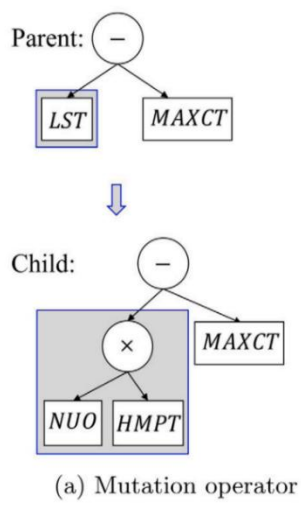


Figure 2.3. Illustration of subtree mutation and crossover operators.

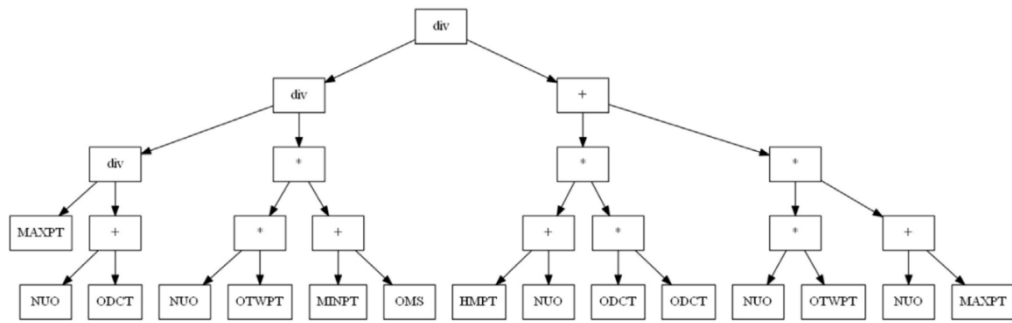


Figure 2.4. Tree representation of LDR-I.

Algorithm 2.3 LDR-I

```
1: Set  $DR^* \leftarrow null$  and  $f(DR^*) \leftarrow +\infty$ 
2: Invoke the designed features and functions
3: Initialize  $\mathcal{P}$ 
4:  $g \leftarrow 0$ 
5: while  $g < g_{max}$  do
6:    $\mathcal{P}_1 \leftarrow \emptyset$ 
7:   Apply FitnessEvaluate ( $\mathcal{P}$ ) to obtain  $f(DR_i)$  for each  $DR_i$ 
8:    $f(DR^*) \leftarrow \min_{i=1}^{N_{\mathcal{P}}} f(DR_i)$ 
9:    $DR^* \leftarrow \arg \min_{i=1}^{N_{\mathcal{P}}} f(DR_i)$ 
10:  Sort  $DR_i$  in  $\mathcal{P}$  in ascending order of  $f(DR_i)$ ; select the first  $p_e \times N_{\mathcal{P}}$  rules
     $DR_1, DR_2, \dots, DR_{p_e \times N_{\mathcal{P}}}$ 
11:   $\mathcal{P}_1 \leftarrow \mathcal{P}_1 \cup \{DR_1, DR_2, \dots, DR_{p_e \times N_{\mathcal{P}}}\}$ 
12:  while  $N_{\mathcal{P}_1} < N_{\mathcal{P}}$  do
13:    Generate a random number  $rand \in [0,1]$ 
14:    if  $rand < p_c$  then
15:       $DR_1^p$  and  $DR_2^p \leftarrow$  apply Tournament ( ) twice to select two parents
16:       $DR_1^p$  and  $DR_2^p \leftarrow$  apply Crossover (  $DR_1^p, DR_2^p$  ) to create two
        offspring rules
17:       $\mathcal{P}_1 \leftarrow \mathcal{P}_1 \cup \{DR_1^c, DR_2^c\}$ 
18:    else
19:       $DR_1^p \leftarrow$  apply Tournament ( ) to select one parent rule
20:       $DR_1^m \leftarrow$  apply Mutation ( $DR_1^p$ ) to create one offspring
21:       $\mathcal{P}_1 \leftarrow \mathcal{P}_1 \cup \{DR_1^m\}$ 
22:    end if
23:  end while
24:   $\mathcal{P} \leftarrow \mathcal{P}_1$ 
25:   $g \leftarrow g + 1$ 
26: end while
27: return  $DR^*$ 
```

Algorithm 2.3 constructs the order sequence by fixing one order a time at position k . Features related to the orders such as HMPT and MINPT are calculated before the scheduling process because they remain unchanged during the scheduling process. ODCPT and OMS are calculated whenever they are required to be used during the scheduling process as they are dependent on the partial schedule. With all the needed features ready, priority of an order is calculated using Eq. (10) and the one with largest priority is chosen to append to π to schedule next. Next, we give the analysis on the time complexity of LDR-I.

Corollary 1. *The time complexity of LDR-I in Algorithm 2.3 is $O(mn^3/2 + n_f n^2/2)$, where $n_f = 14$ is the number of the functions used in LDR-I, m is the number of machines and n is the number of orders.*

Based on Lemma 1, we first identify the features with the maximum time complexity in LDR-I that are static constants, namely features $HMPT_j, MINPT_j, MAXPT_j$ and $OTWPT_j$ for order $j = 1, \dots, n$. Feature $HMPT_j = p_j^{t'}$ where $t' = \arg \max_t \sum_{j=1}^n p_j^t$ takes $O(mn)$ time. Feature $MINPT_j = \min_t p_j^t$, $MAXPT_j = \max_t p_j^t$ and $OTWPT_j = \sum_{t=1}^m w_t p_j^t$ all take $O(mn)$ time for $j = 1, \dots, n$. Feature $ODCT_j$ and OMS_j are obtained by applying partial sequence $\pi = \{\pi(1), \dots, \pi(k), j\}$, thus both of them take $O(mn)$ time. Then using these features, calculating the order priority by Eq. (10) contains $n_f = 14$ arithmetic operations. It follows $O(n(n+1)(mn + n_f)/2)$ time is taken for the two nested loops. Overall, the complexity of LDR-I in Algorithm 2.3 is $O(mn + n(n+1)(mn + n_f)/2) = O(mn^3/2 + n_f n^2/2)$. It should be noted that the term $m^2 n$ in Lemma 1 is not appearing

in the time complexity of LDR-I. This is because the features like ODCT, OMS and EXPT with time complexity $O(m^2n + mn \log n)$ are not involved in LDR-I. With depth to be 5 and only 14 functions in the tree-based structure, LDR-I is very fast for solving problem COS. Among all the 1080 instances, LDR-I takes at most 0.07 s to solve each instance and achieves much better solutions than the existing dispatching rules (refer to Section 6.2 for more details).

To shed the light of the composition structure of LDR and its impacts, we can have some intuitive analysis to LDR-I and how it exhibits better performance compared with the existing dispatching rules. For example, the most commonly used features in LDR-I are: NUO (4 times), ODCT (3 times), OTWPT (2 times) and MAXPT (2 times). From the definition of ODCT and OTWPT in Table 2.2, $ODCT_j = C_j - C_k$ is exactly the feature of the ECT rule and $OTWPT_j = \sum_{t=1}^m w_t p_j^t$ is exactly the feature of TWPT rule. With several divisions, both ODCT and OTWPT will finally be located at the denominator of a complex fraction. This reflects the basic principles from the rules ECT and TWPT, namely, orders with smaller ODCT and OTWPT will have larger priority and thus be scheduled earlier. It is intuitive to connect the good performance of LDR-I with these well defined features in the well-known dispatching rules. Moreover, LDR-I enhances these features through various combinations. We give an illustrative example of the solutions using LDR-I and ECT on an instance of 10 orders and 2 machines as shown in Figure 2.5. It can be seen LDR-I basically follows the dispatching of ECT with the exception of orders 1, 5, 6 and 9. LDR-I performs a pair swap for order pairs (6,5) and (9,1) on the basis of ECT sequence, resulting in the total completion time decreasing from 2899 to 2881.

2.4.2 New Dispatching Rule: LDR-II

In addition, by setting the maximum tree depth as 17 to further exert the power of the proposed FEGP method for learning dispatching rules, we obtained the second new dispatching rule, denoted by LDR-II. Compared with LDR-I, LDR-II is more complex in size and shape. If we write the closed-form of LDR-II as that of LDR-I in Eq. (10), it will take more than one page. Due to the space limit and for a better representation, we do not present LDR-II in the closed-form. Alternatively, the visualization of the tree structure of LDR-II is given in Figure 2.6.

Figure 2.6 shows the overall tree structure of LDR-II without showing the explicit features composition. To further shed the light of LDR-II, Table 2.5 summarizes the statistics of the features and functions used in LDR-II, where the number in the parenthesis indicates the frequency of the feature or function being used. In LDR-II, 18 features among the total 20 are used except for feature MINPT and MINCT, while all the functions are used. The most commonly used features are 0(73 times), ODCT(68 times) and WLCPT(41 times), where feature ODCT is the exactly the key feature of rule ECT. It is thus intuitive to connect the performance of LDR-II with the frequently used features to some extent. In addition, LDR-II contains 354 functions, where function “+” is used for 111 times and the ternary function $\text{if-then-else}(a, b, c)$ is used for 66 times. The limited total number of functions is important to the time efficiency of LDR-II. Next, we give the following corollary for the time complexity of LDR-II.

Table 2.5. Functions and features used in LDR-II

Features (421)	0(73), ODCT(68), WLCPT(41), EST(38), 1(37), LMPT(36), EXTIME(35), HMPT(17), OTCWPT(13), AST(13), NUO(12), MAXPT(11), OTPT(11),OMS(9), OTWPT(3), MAXCT(2), LCPT(1), LST(1), MINPT(0), MINCT(0)
Functions (354)	+ (111), $\times$ (78), if-then-else(a, b, c)(66), $\div$ (46), max(4), $-$ (13)

Number in the parenthesis is the frequency of the feature or function being used in LDR-II.

LDR-I: Obj = 2881

10	3	6	5	9	1	8	4	2	7
----	---	---	---	---	---	---	---	---	---

ECT: Obj = 2899

10	3	9	1	6	5	8	4	2	7
----	---	---	---	---	---	---	---	---	---

Figure 2.5. Illustrative example of solutions by LDR-I and ECT.

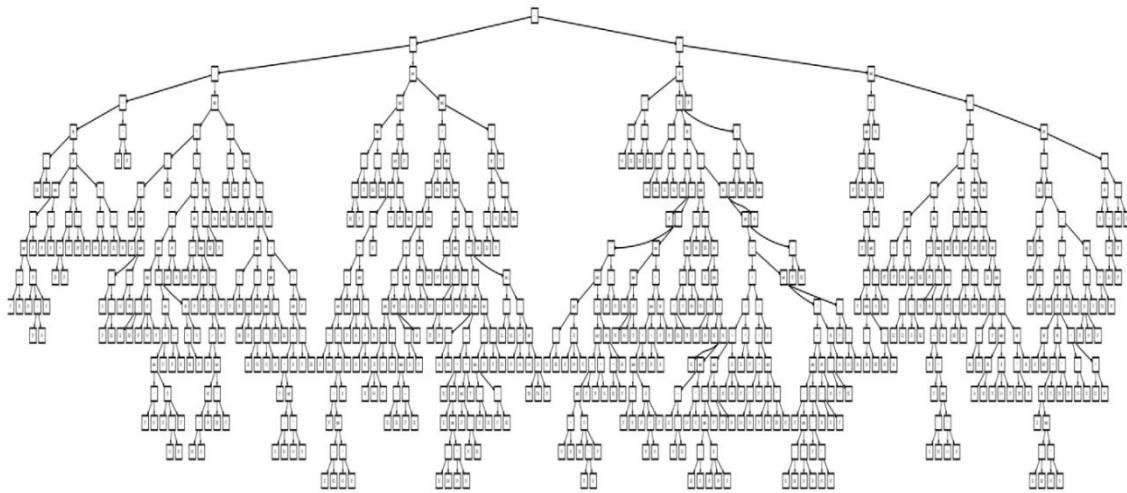


Figure 2.6. Tree representation of LDR-II.

Corollary 2. *The time complexity of LDR-II is $O(mn^3/2 + n_f n^2/2 + m^2 n)$ where $n_f = 354$ is the number of functions used in LDR-II, m is the number of machines and n is the number of orders.*

Based on Lemma 1, Corollary 2 is immediate as it contains both features ODCCT and WLCPT. Even setting d_{max} to 17, LDR-II has only 354 functions, much smaller than the theoretical maximum number of functions $n_f = (3^{17-1} - 1)/2$. Thus, the time complexity of LDR-II is far smaller than the theoretical time complexity in Lemma 1. In fact, LDR-II takes less than 0.1 s for solving each of the 1080 instances and significantly outperforms all the well-known dispatching rules. LDR-II also obtains even much better performance than LDR-I (refer to Section 6.2 for more details). Comparing LDR-II with LDR-I, it can be seen the parameter maximum tree depth (d_{max}) plays an important role for the performance of LDR. While smaller d_{max} produces small size trees for ease of analysis and implementation, it also suffers a little loss in performance compared with larger d_{max} . Meanwhile, larger d_{max} results in increasing of training time due to the large size of trees for fitness evaluation. From the experiment, it takes 1.5 h to train FEGP to obtain LDR-I when d_{max} is 5, and less than 13 h to obtain LDR-II when d_{max} is 17. We also provide the source code and executable program file of both LDR-I and LDR-II online. We will compare LDR-I and LDR-II with existing well-known dispatching rules in Section 6. In the next section, we will develop a two-stage optimization method which uses LDR to provide initial solutions in the first stage.

2.5 Development of Two-stage Method

To further enhance the quality of the solutions generated by LDR (LDR-I or LDR-II), we develop a two-stage optimization method denoted by *LDR-based adaptive search* (LDR-AS) in this section. Figure 2.7 shows the outline of the proposed LDR-AS method, i.e., LDR generates the initial solution in the first stage and then feeds the initial solution to an adaptive local search algorithm in the second stage.

The detailed procedure of LDR-AS is given in Algorithm 2.4. The LDR-AS method consists of LDR initialization stage and adaptive search stage. In the LDR initialization stage, initial solution is obtained by applying LDR to problem COS. Then the adaptive search conducts an iterative search scheme to explore the solution space based on this initial solution. In every iteration, the newly discovered best solution is appended into a collection of schedules *List*, which ensures every solution in the historical iteration does contribute to the final best solution.

Inspired by the PSA proposed in [39], the developed adaptive search algorithm also consists of two main operators: *Intensification* and *Diversification*. In the *Intensification* operator, the current best solution undergoes a bunch of local search operators to find if any potentially better solutions exist and can be accepted under certain acceptance criteria. Next, the *Diversification* operator introduces random disturbance to the solutions that have been trapped in the local optima. The iteration of adaptive search continues until the termination conditions are satisfied. $Obj(\pi)$ is the objective value of using sequence π to solve problem COS. We use i to count the number of iterations. Parameter s is used to count the times of reproducible results appear

consecutively. For instance, if the result after intensification of the 10th (current) iteration is the same with that of the 9th and 8th but differs from that of the 7th iteration, then $s = 3$ and otherwise $s = 1$. Three stopping criteria are included: (1) maximum running time (*maxTime*), (2) maximum number of total iterations (*Num*), and (3) the maximum times of the reproducible results appear consecutively (*sNum*).

We first discuss the *Intensification* as described in Algorithm 2.5. The *Intensification* starts with an initial solution π and then it proceeds with applying uniformly and randomly chosen local search operators from set *insert*, *swap*, *insert-pair*, *swap-pair* to π until the termination criteria satisfy. The newly generated schedule π' is further examined in correspondence of its performance $Obj(\pi')$. Performance of a new schedule after applying local search operators is classified into three categories: good, moderate and poor. We define the good category to be the solution whose improvement is larger than a predefined threshold δ , the moderate category to be the schedule whose improvement is smaller than δ but larger than 0, and the others with negative improvement as the poor category. Only good category schedules are accepted in the algorithm. Parameters used in the algorithm are given as follows: δ is the threshold to decide whether a new schedule can be accepted, *nNum* is the maximum times of moderate improvement, *pNum* is the maximum times of poor improvement and r is the range within which the local operators work. Let π^* be the best schedule and π_0 record the best schedule that is not accepted so far in current iteration. p and m are the counter of the moderate and poor improvement, respectively.

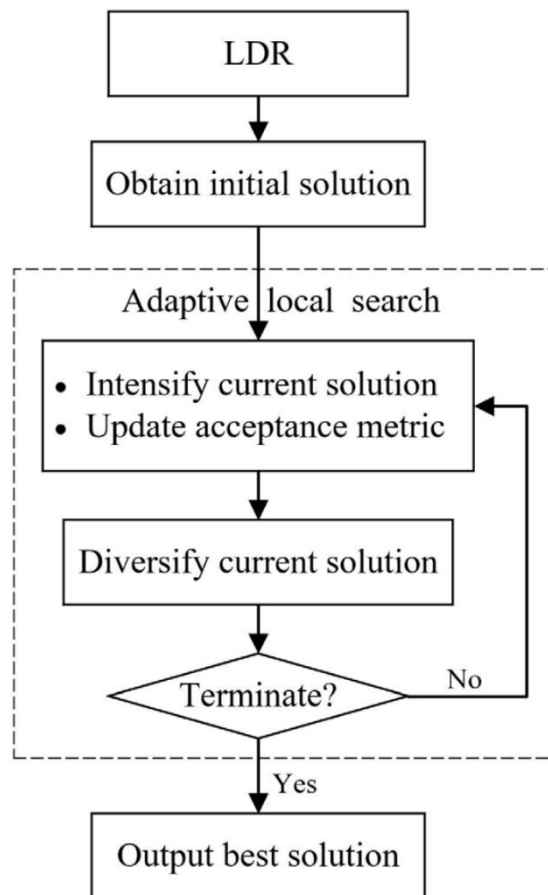


Figure 2.7. Flow chart of the proposed two-stage optimization method LDR-AS.

Algorithm 2.4 LDR-AS

```
1:   $i \leftarrow 0; t \leftarrow 0$ ; start counting running time
2:   $\pi_0 \leftarrow$  Use LDR to obtain the initial solution
   while iteration number  $i < Num$  and running time  $< maxTime$  and  $s < sNum$  do
3:
4:     $\pi_1 \leftarrow$  Intensification ( $\pi_0$ )
5:    Add  $\pi_1$  to List; update  $s$ 
6:     $\pi_0 \leftarrow$  Diversification ( $\pi_1$ )
7:     $i \leftarrow i + 1$ 
8:  end while
9:  Return  $\pi^* = \arg \min_{\pi' \in List} Obj(\pi')$ 
```

The key factor of the *Intensification* to categorize the schedule performance lies on the determination of the acceptance threshold δ . In the LDR-AS method, we propose an adaptive update mechanism to refresh δ to appropriate value timely. We use π_0 to store the best moderate (not accepted) schedule when the cumulative times of moderate improvements m reaches $nNum$. Then δ is set to be the difference between the current best accepted schedule π^* and the best moderate schedule π_0 , i.e., $\delta = \max\{1, Obj(\pi^*) - Obj(\pi_0)\}$. This ensures even if the search continues to find moderate improvement for accumulative $mNum$ times, the current best moderate schedule π_0 will be used as π^* . Note that the minimum value of δ is 1. Intuitively, if the search algorithm continues achieving unacceptable moderate improvement repeatedly, this implicates the threshold δ is too large and needs to be decreased down. In contrast, when the number of poor improvement p exceeds its maximum $pNum$, it basically means there is no further potential improvement. In this case, it is not necessary to update the value of δ and the algorithm should be terminated. We tailor the local search operators by limiting its operators to work within range $[\max\{k - r, 0\}, \min\{k + r, n\}]$ where k is the position in the sequence. The new local search operators in LDR-AS are given as follows.

insert (π, r) : randomly and uniformly select a customer order at position k_1 in π , and an insertion position $k_2 \in [\max\{k_1 - r, 0\}, \min\{k_1 + r, n\}]$ where $k_2 \neq k_1$. If $k_1 > k_2$, move the orders at positions $k_2 \leq k' < k_1$ to positions $k' + 1$. Otherwise, move the orders at position $k_1 < k' \leq k_2$ to positions $k' - 1$. Insert order at position k_1 to position k_2 to obtain π' .

Algorithm 2.5 Intensification(π)

```
1: Initialize  $\pi^* \leftarrow \pi; \pi_0 \leftarrow \pi; p = 0; m = 0$ 
2: while  $p < pNum$  do
3:   operator  $\leftarrow$  select an operator uniformly and randomly from operator set
   {insert, swap, insert-part, swap-pair}
4:    $\pi' \leftarrow$  apply operator ( $\pi^*, r$ ) to  $\pi^*$ 
5:   if  $Obj(\pi^*) - Obj(\pi') \geq \delta$  then
6:      $\pi^* \leftarrow \pi'$ 
7:      $m \leftarrow 0; p \leftarrow 0$ 
8:   else if  $0 \leq Obj(\pi^*) - Obj(\pi') < \delta$  then
9:      $m \leftarrow m + 1; p \leftarrow 0$ 
10:    if  $Obj(\pi') < Obj(\pi_0)$  then
11:       $\pi_0 \leftarrow \pi'$ 
12:    end if
13:  else
14:     $p \leftarrow p + 1$ 
15:  end if
16:  if  $m \geq mNum$  then
17:     $\delta = \max\{1, Obj(\pi^*) - Obj(\pi_0)\}$ 
18:     $\pi^* \leftarrow \pi_0; m \leftarrow 0$ 
19:  end if
23: end while
24: return  $\pi^*$ 
```

Algorithm 2.6 Diversification (π)

1: $i \leftarrow 0; i \leftarrow 0$; start counting running time

2: operator $\leftarrow$ select an operator uniformly and randomly from set
 {insert, swap, insert-part, swap-pair}

3: $\pi' \leftarrow$ apply operator (π, r) to schedule π

4: $\pi \leftarrow \pi'$

5: **end for**

6: **Return** π

$\text{swap}(\pi, r)$: randomly and uniformly select a customer order at position k_1 in π , and a

swap position $k_2 \in [\max\{k_1 - r, 0\}, \min\{k_1 + r, n\}]$ where $k_2 \neq k_1$. Swap the orders at position k_1 and k_2 to obtain π' .

$\text{insert-pair}(\pi, r)$: randomly and uniformly select a customer order at position k_1 in

π , and an insertion position $k_2 \in [\max\{0, k_1 - r\}, \min\{k_1 + r, n\}]$ where $|k_2 - k_1| \geq 2$. If $k_1 > k_2$, move the orders at positions $k_2 \leq k' < k_1$ to positions $k' + 2$. Otherwise, move the orders at positions $k_1 + 1 < k' < k_2 + 1$ to positions $k' - 2$. Move orders at $k_1, k_1 + 1$ to $k_2, k_2 + 1$ to obtain π' .

$\text{swap-pair}(\pi, r)$: randomly and uniformly select a customer order at position k_1 in π ,

and an insertion position $k_2 \in [\max\{0, k_1 - r\}, \min\{k_1 + r, n\}]$ where $|k_2 - k_1| \geq 2$. Swap orders at $k_1, k_1 + 1$ with those at positions $k_2, k_2 + 1$ to obtain π' .

Schedule obtained from obtained from the Intensification is highly possible to be trapped within local optima, we thus force to involve random disturbance to get rid of the local optima basin through the Diversification as presented in Algorithm 2.6. Local search operators are randomly and uniformly selected to diversify the schedule for $dNum$ times, and the newly generated schedule π' is always accepted regardless of its performance.

2.6 Computational Experiments

We conduct computational experiments to test the performance of the proposed algorithms and state-of-the-art methods for problem COS in the literature. A total of seven well-known dispatching rules, the MILP model, the efficient PSA method [39], a PSO [17]

and a GA [28] are used for the comparison with the proposed LDR and LDR-AS. Table 2.6 briefly summarizes the selected seven well-known dispatching rules.

All runs are made on a 64-bit Windows 10 platform with Intel Core i7-8750H 2.20 GHz CPU and 16 GB RAM. The proposed FEGP is implemented based on the ECJ library, a Java-based evolutionary computation system. The PSO, LDR and LDR-AS algorithms are coded in Java as well. The executable program file of PSA algorithm is obtained from the authors of [39]. The MILP model is solved using CPLEX 12.8 with default configurations except that the time limit is set to be 3600 s and the “nodefile” configuration is turned on to avoid running out of memory. The source code and executable programs of LDR-I, LDR-II, the dataset and the best found solutions by our proposed methods are available online.

2.6.1 Testing Instances

- Test-1: a total of 360 instances are divided into 12 groups with 30 instances in each. Each group is a combination of $n \in \{50, 100, 200\}$ and $m \in \{2, 5, 10, 20\}$. For the job in order j on machine t , its processing time p_j^t is randomly generated from a uniform distribution $[1, 100)$

We use three classes of testing instances to validate the effectiveness of the proposed methods in this paper. Among them, Test-1 and Test-2 are the same with that in [39] and originally generated by Framinan and Perez-Gonzalez [38]. Test-3 is a new dataset randomly generated following the processing generation rule in [22]. In each class, a total of 360 instances are generated and the generation rules are given in details below.

- Test-2: a total of 360 instances remain the same configurations with Test-1 except for the generation of job processing time. For each customer, the number of jobs included in this order, l , is first randomly selected from a uniform distribution $[1, m]$. l jobs are then randomly selected from the job set and then the processing time p_j^t is generated from uniform distribution $[1, 100]$
- Test-3: (newly generated): a total of 360 instances remain the same configurations with Test-1 except for the generation of job processing time. For a specific $n - m$, 30 instances are generated with job processing time s randomly generated from uniform distribution $[1, l_{max}]$, where $l_{max} \in \{20, 50, 100\}$. For each $l_{max} \in \{20, 50, 100\}$, 10 instances are generated

We use the metric, average relative percentage deviation (ARPD), to compare the computational performance of different methods on the given instances in each group. The ARPD is defined as follows.

$$ARPD(Alg) := \frac{1}{T} \sum_{I_t \in \{I_1, I_2, \dots, I_T\}} \frac{Obj(Alg, I_t) - Obj_{ref}(I_t)}{Obj_{ref}(I_t)} \times 100,$$

where $Obj(Alg, I_t)$ denotes the objective value of algorithm Alg for solving instance I_t , and $Obj_{ref}(I_t)$ denotes the best objective value among all the algorithms for solving instance I_t . $T = 30$ is the total number of instances in each group. It is simple to check that smaller ARPD indicates better performance for a given algorithm. Particularly, if the ARPD of an algorithm is equal to zero, that means this algorithm outperforms all the others for each instance in the given group.

Table 2.6. Well-known dispatching rules for customer order scheduling

STPT	Arrange orders by non-decreasing sequence of the total job process time, i.e., $\sum_{t=1}^m p_j^t$.
SMPT	Arrange the orders by the non-decreasing sequence of its maximum job processing time, i.e., $\max_t p_j^t$.
SMCT	First arrange jobs on each machine by non-decreasing order of p_j^t and compute the completion time C_j' for each order j . Then schedule the orders in non-decreasing sequence of C_j' .
SPTL	First find the machine t' with the smallest workload in the current partial schedule, then select the order with smallest $p_j^{t'}$ to schedule
ECT	Select the order with the earliest completion time to schedule, i.e., $j^* = \arg \min_j (C_j - C_k)$ wherein k is the order immediately before j .
TWPT	Arrange orders by non-decreasing sequence of total weighted job processing time of the orders $\sum_{t=1}^m w_t p_j^t$, herein $w_t = \sum_{j=1}^n p_j^t / (\sum_{t=1}^m \sum_{j=1}^n p_j^t)$.
NEW	Design a look-ahead indicator $\psi_1 = C(S_l) + C^*(S_l, R_l) / (n - k + 1)$ including the contribution of both the scheduled orders S_l and unscheduled orders R_l . The order with the smallest ψ_1 is selected to schedule.

2.6.2 Comparison Results of LDRs and Well-known Dispatching Rules

Table 2.7 -Table 2.9 present the ARPD comparison of LDR-I, LDR-II and seven well-known dispatching rules on Test-1, Test-2 and Test-3, respectively. In these tables, the entries of each column shows the ARPD value of the corresponding method over 30 instances for the combination $n - m$.

In Table 2.7, LDR-II outperforms all the other rules in all the combinations of $n - m$. In particular, LDR-II obtains an ARPD of zero in rows 100-2, 200-2, 200-5 and 200-10, and generates the best solution for the 30 instances in combinations of 200-2 and 200-10. At row 50-20, LDR-II obtains the largest ARPD 0.23, but is still significantly smaller than that of the others. This indicates LDR-II is able to better handle more complex and large-scale problem instances than the other rules. In terms of LDR-I, it performs slightly worse than LDR-II but still significantly better than all the well-known dispatching rules for all $n - m$ combinations. In Table 2.8, LDR-II obtains the best ARPD on 9 rows while LDR-I obtains the best ARPD on 5 rows. LDR-I generates the best solution for all the 30 instances in combination 200-5, having an ARPD of zero. This shows LDR-I can perform very well even with much simpler algorithm structure comparing with LDR-II. The superiority of both LDR-I and LDR-II is more obvious when comparing with well-known dispatching rules. For instance, SPTL obtains the smallest ARPD 12.71 at row 50-2 but the largest ARPD of 30.50 at row 200-20 while LDR-II obtains 0.06 and 0.03 on corresponding rows. Notably, NEW performs the best among the seven selected well-known dispatching rules, and followed by ECT. In terms of the solution quality, LDRs outperform the existing dispatching rules for all the combinations in Test-1 and Test-2. It should be noted that the

Table 2.7. ARPD comparison of LDR with well-known dispatching rules on Test-1.

n-m	STPT	SMPT	SMCT	SPTL	ECT	TWPT	NEW	LDR-I	LDR-II
50-2	3.60	6.03	5.08	4.10	1.48	2.83	0.28	0.09	0.01
50-5	6.50	10.97	8.93	5.39	1.53	5.52	0.70	0.21	0.04
50-10	8.04	12.71	9.11	5.22	1.26	6.99	0.85	0.21	0.12
50-20	7.82	12.17	9.18	4.97	1.03	7.25	0.58	0.28	0.23
100-2	3.27	6.40	5.59	4.35	2.42	2.71	0.29	0.05	0.00
100-5	6.31	11.21	8.94	5.64	1.53	5.60	0.80	0.14	0.01
100-10	7.03	11.33	9.36	5.62	1.36	6.28	0.87	0.17	0.01
100-20	7.42	11.21	8.78	4.98	0.95	6.93	0.62	0.10	0.05
200-2	2.70	6.26	5.32	4.18	2.90	2.26	0.28	0.05	0.00
200-5	4.92	10.64	9.08	6.07	1.91	4.44	1.04	0.15	0.00
200-10	5.35	10.96	9.20	6.06	1.51	4.95	0.93	0.20	0.00
200-20	5.77	10.98	8.77	5.21	1.19	5.43	0.75	0.12	0.02

Table 2.8. ARPD comparison of LDR with well-known dispatching rules on Test-2.

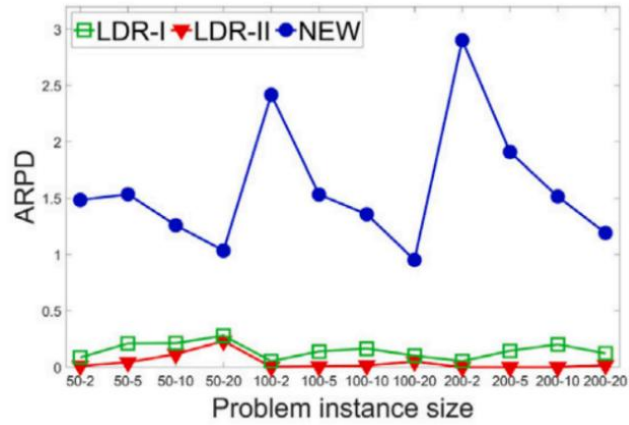
n-m	STPT	SMPT	SMCT	SPTL	ECT	TWPT	NEW	LDR-I	LDR-II
50-2	7.02	9.90	6.47	12.71	2.20	5.04	0.54	0.06	0.06
50-5	12.10	20.96	14.88	19.84	2.78	9.69	1.16	0.20	0.10
50-10	15.15	25.89	19.03	23.19	2.09	13.42	1.13	0.31	0.16
50-20	14.70	28.25	19.81	22.67	1.72	13.22	1.36	0.29	0.18
100-2	5.96	9.71	6.81	13.29	2.80	4.76	0.53	0.05	0.05
100-5	10.55	20.99	15.60	23.16	3.43	8.58	1.52	0.07	0.08
100-10	13.23	26.29	20.21	23.54	2.54	11.94	1.44	0.13	0.12
100-20	13.54	28.61	21.83	27.22	1.73	12.67	1.26	0.28	0.09
200-2	3.79	8.61	6.76	13.67	3.73	2.74	0.40	0.03	0.06
200-5	9.91	20.50	16.18	23.32	4.19	8.19	2.05	0.00	0.04
200-10	11.20	25.82	22.04	29.09	2.77	10.24	1.58	0.24	0.03
200-20	11.48	29.85	24.22	30.50	2.02	10.74	1.24	0.35	0.03

Table 2.9. ARPD comparison of LDR with well-known dispatching rules on Test-3.

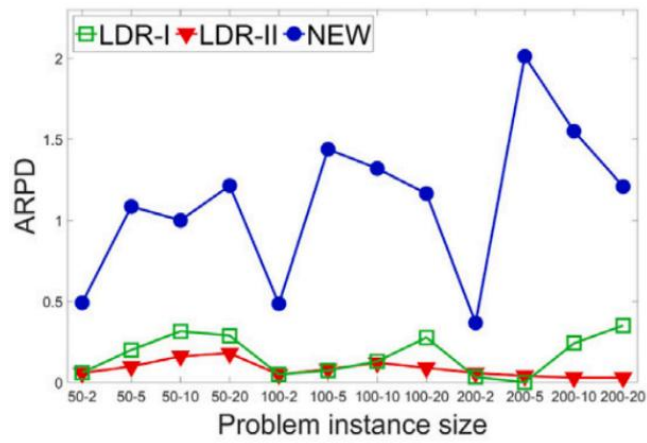
n-m	STPT	SMPT	SMCT	SPTL	ECT	TWPT	NEW	LDR-I	LDR-II
50-2	3.39	6.30	5.35	4.02	1.75	2.91	0.29	0.05	0.12
50-5	6.40	11.39	8.87	5.84	1.50	5.55	0.64	0.19	0.10
50-10	7.29	11.09	8.87	5.37	1.22	6.70	0.51	0.15	0.17
50-20	7.46	10.15	8.08	5.10	1.02	6.99	0.69	0.29	0.15
100-2	2.64	5.96	5.04	4.15	2.39	2.37	0.24	0.07	0.02
100-5	5.59	10.86	8.47	5.52	1.50	4.93	0.75	0.14	0.04
100-10	6.88	11.32	9.00	5.41	1.23	6.28	0.74	0.16	0.03
100-20	6.56	11.23	8.68	5.02	0.98	6.14	0.49	0.18	0.09
200-2	2.89	5.79	-5.56	4.08	2.69	2.38	0.19	0.13	0.01
200-5	5.11	10.31	8.67	5.90	1.83	4.62	0.96	0.22	0.00
200-10	5.40	10.72	8.99	5.81	1.36	4.99	0.86	0.25	0.00
200-20	5.74	10.27	8.23	4.97	1.07	5.38	0.62	0.23	0.01

Table 2.10. Number of best found solutions by LDR and the selected dispatching rules.

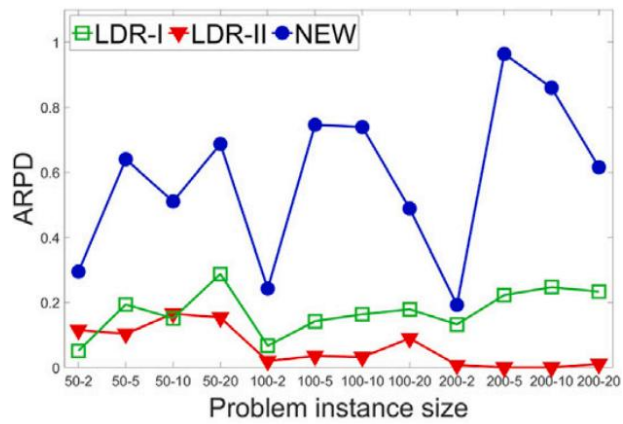
n-m	STPT	SMPT	SMCT	SPTL	ECT	TWPT	NEW	LDR-I	LDR-II	LDR
Test-1	0	0	0	0	0	0	9	66	287	351
Test-2	0	0	0	0	0	0	7	132	223	353
Test-3	0	0	0	0	2	0	25	77	257	333
Total	0	0	0	0	2	0	41	275	767	1037



(a) ARPD comparison on Test-1



(b) ARPD comparison on Test-2



(c) ARPD comparison on Test-3

Figure 2.8. Comparison of LDRs and NEW.

maximum solution times for each instance by LDR-I and LDR-II are 0.07 and 0.1 s respectively, much faster than 0.6 s by NEW. All the other testing dispatching rules take less than 0.01 s to solve each instance.

In Table 2.9, LDR-II achieves the best ARPD on 10 rows while LDR-I achieves the best ARPD on 2 rows for Test-3. As LDR-I and LDR-II are learned by FEGP using randomly generated data following generation rule of Test-1 and Test-2, this indicates LDRs have robust performance regardless of the data structure information. Figure 2.8 graphically shows the ARPD comparison of LDR-I, LDR-II and NEW according to the results in Table 2.7 -Table 2.9. Obviously, LDR achieves much better solutions than the dispatching rule NEW in all combinations of problem instance sizes.

Table 2.10 summarizes the number of best found solutions by LDR and the existing well-known dispatching rules. As shown in this table, LDR-I and LDR-II obtain the best solution for 275 and 767 instances, respectively. In total, LDR-I and LDR-II altogether obtain the best solution for 1037 instances compared with 41 by NEW among the total 1080 instances. There are only two instances that ECT achieves the best solution. All the other dispatching rules are not capable of finding the best solution for any of the instances.

2.6.3 Comparison Results of LDR-AS and State-of-the-art Methods

We further compare the proposed two-stage method LDR-AS with PSA, MILP (solving the MILP formulation using CPLEX) and two representative metaheuristics (PSO and GA) on Test-1, Test-2 and Test-3. For PSO and GA, LDR-I, LDR-II and NEW are used to create initial solutions, respectively.

We perform the executable program of PSA 5 times on each instance. The maximum iteration number $IterMax$ in the PSA method is set to be 2000. The smallest objective value of these 5 runs together with the published records in [39] is selected as the best performance of PSA method. Accordingly, we run LDR-AS method 5 times on each instance and the smallest among the 5 runs is chosen as the best performance of the LDR-AS method. Based on a number of preliminary testing with consideration of convergence speed and solution quality, we set the parameters of the LDR-AS method as follows: $Num = 400$, $sNum = 5$, initial $\delta = 20$, $nNum = 1500$, $pNum = 5000$, $dNum = 2$, $r = 0.1 \times n$ where n is the number of the orders in the problem instance. In addition, we adopt the well-tuned PSO method developed in [17] wherein pairwise swap operator is used to improve the solution in every PSO iteration. The same parameters in that paper are also adopted as following: $B_1 = 0.5$, $B_2 = 0.5$ and $w = 0.5$. We set the number of particles $N = 2 \times n$, number of PSO iterations $ITRN = 300$ and number of local search improvement $NN = 20$ after a number of testing experiments considering both the problem size and algorithm convergency. Regarding to the GA, we adopt the hybrid GA with a variable neighborhood local search in [28]. We use the well tuned parameters in the original paper as following: local search probability $PM = 0.1$, mutation probability $P_m = 0.1$ and crossover probability $P_c = 0.8$. Considering both the problem size the comparable time with other algorithms, we set the population size $PS = 2 \times n$ and the iteration for local search $Inloop_{max} = 500$. PSA, LDR-AS, PSO and GA are executed with a time limit $maxTime = 600$ s. It should be noted that the PSA executable file does not support

setting the running time limit with the input parameters, we realize this through controlling the running time of the executable file using Bash script.

Table 2.11 - Table 2.13 present the comparison results of MILP, PSA, PSO, GA and LDR-AS on Test-1, Test-2 and Test-3, respectively. In addition, we use LDR-I, LDR-II and NEW to create initial solutions for PSO and GA. Accordingly, we have methods NEW-PSO, LDR-I-PSO, LDR-II-PSO, NEW-GA, LDR-I-GA, LDR-II-GA, LDR-I-AS and LDR-II-AS. For the corresponding method in those tables, column “ARPD” indicates the ARPD value and column “T (s)” is the average solution time (in seconds) over the whole 30 instances in each combination.

As shown in Table 2.11, the proposed LDR-AS method obtains the best solutions among all the methods. Specifically, LDR-II-AS obtains the best ARPD on 11 rows, both LDR-I-AS and PSA obtain the best on 2 rows whereas LDR-II-PSO obtains the best ARPD on only 1 row. LDR-II-AS outperforms PSA in both solution quality and solution time. Comparing with LDR-II-AS, LDR-I-AS performs slightly worse. Only at combination 200-5, LDR-I-AS outperforms LDR-II-AS. This indicates with the same adaptive search method in the second stage, the good quality of initial solution is critical to improve the overall performance. Nevertheless, LDR-I-AS still outperforms PSA, achieving the best ARPD on 9 rows comparing with PSA. The performance improvement brought by LDR can also be seen by comparing NEW-PSO with LDR-I-PSO and LDR-II-PSO, and also comparing NEW-GA with LDR-I-GA and LDR-II-GA. Specifically, LDR-I-PSO and LDR-II-PSO obtain the best ARPD on 12 and 11 rows respectively comparing with NEW-PSO. Both the LDR-I-GA and LDR-II-GA outperform the NEW-GA on 10 rows. The GA

Table 2.11. Comparison of MILP, PSO, GA, PSA and LDR-AS on Test-1.

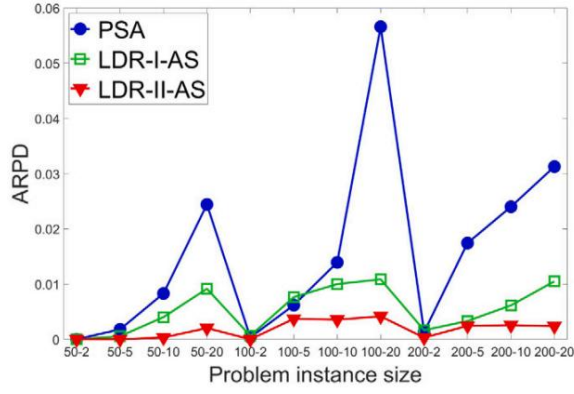
n-m	MILP		NEW-PSO		LDR-I-PSO		LDR-II-PSO		NEW-GA		LDR-I-GA		LDR-II-GA		PSA		LDR-I-AS		LDR-II-AS	
	ARPD	T (s)	ARPD	T (s)	ARPD	T (s)	ARPD	T (s)	ARPD	T (s)	ARPD	T (s)	ARPD	T (s)	ARPD	T (s)	ARPD	T (s)	ARPD	T (s)
50-2	0.045	3600	0.001	2.06	0.001	2.18	0.000	2.30	0.001	6.99	0.001	4.73	0.001	4.63	0.000	3.18	0.000	1.07	0.000	1.00
50-5	1.169	3600	0.054	5.97	0.031	7.34	0.029	6.45	0.033	13.67	0.027	9.98	0.025	9.78	0.002	6.47	0.001	5.72	0.000	5.87
50-10	3.675	3600	0.157	10.54	0.081	12.33	0.127	13.02	0.071	22.29	0.067	18.24	0.070	18.05	0.007	11.61	0.002	9.38	0.001	9.80
50-20	3.529	3600	0.263	20.66	0.152	26.14	0.156	26.87	0.111	46.84	0.101	33.32	0.129	32.79	0.024	21.26	0.008	17.31	0.004	17.20
100-2	0.294	3600	0.005	17.89	0.005	15.46	0.006	16.05	0.006	30.97	0.006	24.46	0.005	23.94	0.000	20.92	0.001	15.30	0.000	15.46
100-5	3.145	3600	0.135	50.32	0.098	36.25	0.094	32.60	0.153	55.58	0.112	46.38	0.100	45.35	0.006	44.02	0.007	25.62	0.005	25.49
100-10	7.446	3600	0.419	98.53	0.226	74.41	0.222	67.87	0.342	105.73	0.224	89.93	0.215	89.01	0.012	78.78	0.008	45.96	0.005	46.95
100-20	8.391	3600	0.775	154.38	0.351	149.00	0.384	151.85	0.553	204.44	0.320	171.31	0.335	169.47	0.057	145.91	0.011	90.77	0.005	95.65
200-2	1.353	3600	0.010	173.89	0.008	107.47	0.005	108.94	0.014	122.05	0.010	113.65	0.007	111.53	0.001	149.87	0.002	47.76	0.000	46.73
200-5	11.172	3600	0.212	283.25	0.112	298.67	0.086	276.13	0.370	253.17	0.135	234.80	0.100	231.92	0.016	322.95	0.002	93.51	0.004	92.77
200-10	17.355	3600	0.752	574.41	0.226	544.06	0.195	523.64	0.849	476.93	0.241	415.03	0.211	410.24	0.023	576.25	0.005	162.17	0.004	168.36
200-20	14.564	3600	1.110	603.20	0.367	605.54	0.415	605.27	1.139	605.10	0.403	606.37	0.429	603.76	0.029	600.00	0.009	324.03	0.003	332.94

Table 2.12. Comparison of MILP, PSO, GA, PSA and LDR-AS on Test-2.

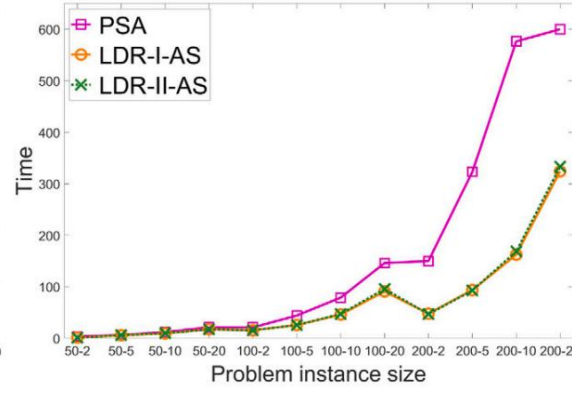
n-m	MILP		NEW-PSO		LDR-I-PSO		LDR-II-PSO		NEW-GA		LDR-I-GA		LDR-II-GA		PSA		LDR-I-AS		LDR-II-AS	
	ARPD	T (s)	ARPD	T (s)	ARPD	T (s)	ARPD	T (s)	ARPD	T (s)	ARPD	T (s)	ARPD	T (s)	ARPD	T (s)	ARPD	T (s)	ARPD	T (s)
50-2	0.039	3600	0.004	1.83	0.000	2.00	0.000	2.29	0.000	7.40	0.000	5.58	0.000	5.44	0.000	3.17	0.000	0.64	0.000	0.52
50-5	2.062	3600	0.046	5.72	0.029	5.58	0.033	6.27	0.026	14.03	0.023	10.39	0.018	9.95	0.001	6.44	0.000	5.43	0.000	5.43
50-10	4.333	3600	0.119	8.74	0.057	10.54	0.072	10.92	0.052	23.30	0.043	17.67	0.045	17.08	0.003	11.42	0.001	8.17	0.000	8.31
50-20	5.970	3600	0.173	15.52	0.054	20.74	0.070	21.35	0.052	47.99	0.033	34.69	0.040	33.47	0.009	21.23	0.001	16.13	0.000	14.85
100-2	0.495	3600	0.004	13.80	0.004	18.54	0.003	19.42	0.004	32.21	0.004	24.47	0.004	23.97	0.000	20.60	0.000	14.33	0.000	14.18
100-5	3.848	3600	0.177	40.78	0.132	31.04	0.175	35.46	0.196	57.49	0.135	45.77	0.161	44.22	0.005	43.55	0.012	26.98	0.004	25.91
100-10	8.879	3600	0.441	71.09	0.256	59.95	0.311	65.62	0.356	109.08	0.230	87.75	0.277	86.52	0.023	77.61	0.016	51.60	0.001	47.31
100-20	29.278	3600	0.658	139.23	0.373	120.71	0.395	130.28	0.454	209.34	0.296	215.83	0.346	211.97	0.041	144.23	0.017	93.57	0.002	94.62
200-2	1.921	3600	0.007	128.91	0.006	89.63	0.006	109.19	0.008	127.97	0.007	128.76	0.007	123.70	0.001	150.33	0.000	43.87	0.000	46.15
200-5	40.524	3600	0.282	348.49	0.161	313.87	0.191	342.70	0.418	267.03	0.191	272.12	0.227	264.55	0.016	323.39	0.009	93.60	0.002	94.66
200-10	61.882	3600	0.830	578.14	0.286	547.18	0.433	515.68	0.940	488.49	0.339	478.73	0.494	469.05	0.031	561.37	0.011	160.69	0.002	166.66
200-20	58.994	3600	1.290	605.55	0.516	605.70	0.622	604.66	1.415	608.27	0.530	606.39	0.672	604.47	0.053	600.00	0.017	318.79	0.004	329.86

Table 2.13. Comparison of MILP, PSO, GA, PSA and LDR-AS on Test-3.

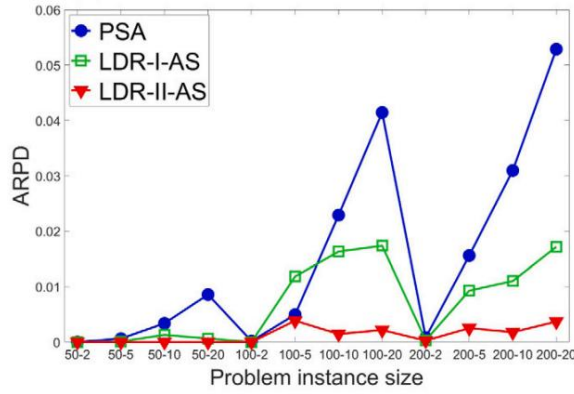
n-m	NEW-PSO		LDR-I-PSO		LDR-II-PSO		NEW-GA		LDR-I-GA		LDR-II-GA		PSA		LDR-I-AS		LDR-II-AS	
	ARPD	T (s)	ARPD	T (s)	ARPD	T (s)	ARPD	T (s)	ARPD	T (s)	ARPD	T (s)	ARPD	T (s)	ARPD	T (s)	ARPD	T (s)
50-2	0.006	2.02	0.002	2.12	0.001	2.20	0.000	6.41	0.000	4.73	0.000	4.64	0.000	2.57	0.001	0.82	0.000	0.90
50-5	0.054	5.67	0.026	6.58	0.029	6.90	0.037	11.79	0.030	9.89	0.033	9.74	0.025	5.21	0.002	5.90	0.001	5.77
50-10	0.207	10.13	0.104	12.32	0.097	12.19	0.073	20.49	0.059	21.23	0.066	20.82	0.055	9.64	0.008	9.12	0.004	8.75
50-20	0.319	21.95	0.153	26.40	0.150	26.48	0.126	38.94	0.109	40.65	0.094	40.10	0.065	17.10	0.010	15.87	0.005	15.98
100-2	0.007	14.97	0.007	15.36	0.006	15.39	0.008	28.20	0.006	26.10	0.005	25.68	0.003	16.58	0.001	14.07	0.000	14.35
100-5	0.140	45.96	0.081	33.31	0.084	30.75	0.147	51.90	0.089	50.93	0.084	50.19	0.045	34.66	0.007	26.20	0.001	26.67
100-10	0.427	87.65	0.192	66.76	0.186	67.93	0.350	96.03	0.187	99.77	0.181	99.25	0.059	62.25	0.012	47.92	0.003	48.55
100-20	0.734	149.80	0.345	138.85	0.340	144.64	0.570	174.92	0.297	189.15	0.298	187.92	0.087	115.27	0.015	94.80	0.004	96.73
200-2	0.011	138.30	0.009	103.88	0.009	99.78	0.015	120.31	0.011	121.37	0.011	119.31	0.006	118.72	0.002	41.50	0.001	41.10
200-5	0.244	257.25	0.093	277.64	0.074	273.31	0.379	240.68	0.113	252.82	0.089	250.22	0.052	256.07	0.003	92.91	0.002	92.93
200-10	0.764	569.67	0.205	523.36	0.178	532.88	0.860	418.42	0.224	456.47	0.208	450.03	0.065	453.91	0.006	169.82	0.003	170.53
200-20	1.053	604.34	0.335	605.68	0.364	605.16	1.063	607.70	0.359	607.00	0.388	604.62	0.055	600.00	0.007	339.38	0.004	339.07



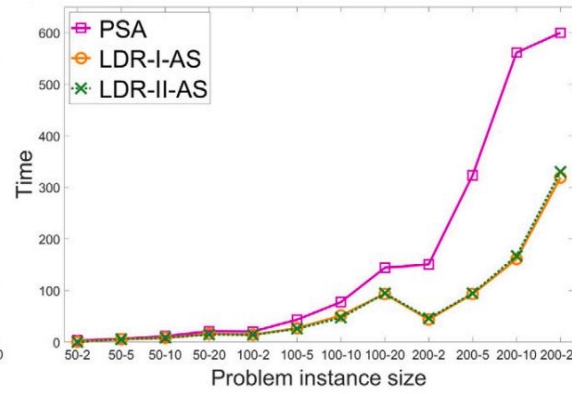
(a) ARPD comparison on Test-1



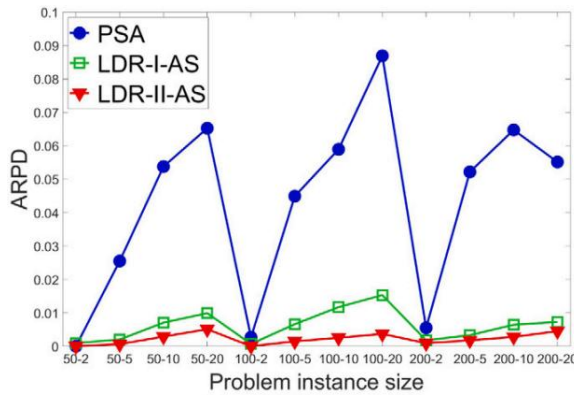
(b) ARPD comparison on Test-1



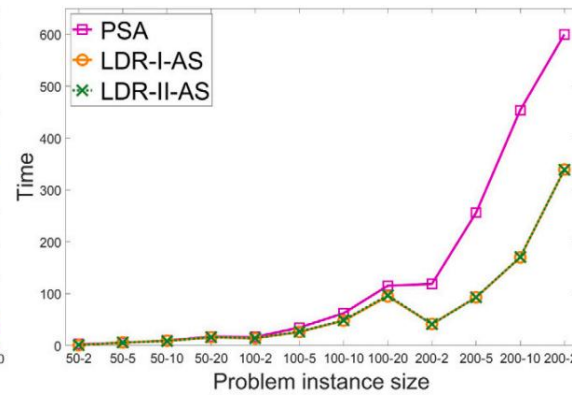
(c) ARPD comparison on Test-2



(d) ARPD comparison on Test-2



(e) ARPD comparison on Test-1



(f) ARPD comparison on Test-1

Figure 2.9. Comparison of PSA and LDR-AS on Test-1, Test-2, Test-3.

Table 2.14. Number of best found solutions by PSA and LDR-AS

Instances	PSA	LDR-I-AS	LDR-II-AS	LDR-AS
Test-1	122	191	245	327
Test-2	169	194	297	333
Test-3	50	187	285	356
Total	341	572	827	1016

performs quite similarly with PSO on ARPD. On rows 50-2, LDR-I-AS, LDR-II-AS, PSA and LDR-II-PSO obtain the same ARPD value of zero. This is reasonable for less machines result in less computational challenge, and thus all the four methods are capable of finding similar near-optimal solutions. While for the instances with more orders and machines, the ARPD values of PSA are much larger than that of LDR-AS. For instance, the ARPD of PSA at 100-20 is 0.057 while that of LDR-I-AS and LDR-II-AS are 0.011 and 0.005, respectively. As almost all the best solutions are generated by LDR-AS, this indicates LDR-AS exhibits better computational performance on solving large-scale instances.

In terms of the solution time, either LDR-I-AS or LDR-II-AS is the fastest comparing with other methods in Table 2.11. Also, LDR-I-AS and LDR-II-AS take almost the same time for solving each $n - m$ combination. The time superiority of LDR-AS is more obvious at large instance cases. For example, PSA takes 576.25 s on row 200-10 while LDR-II-AS takes 168.36 s. Also, for row 200-20, the solution time for PSA on all instances exceeds time limit 600 s. There is no obvious relationship of the times of GA and PSO. In general their computational times are comparable. It should be noted that CPLEX cannot find good enough solutions for all the instances by solving the MILP model within one hour.

The comparison results on Test-2 and Test-3 are presented in Table 2.12 and Table 2.13. Since solving MILP models is not able to obtain good results in Test-1 and Test-2, we do not present the MILP results in Table 2.13. The conclusions of comparing LDR-AS with state-of-the-art methods in both Table 2.12 and Table 2.13 are consistent with that

in Table 2.11. We thus conclude that the proposed two-stage optimization method achieves the better computational performance using less solution time on all the instances.

Figure 2.9 shows the comparison of PSA and LDR-AS based on the ARPD value and solution time in Table 2.11 - Table 2.13, respectively. As shown in the figure, the LDR-AS method obtains the smallest ARPD over all combinations of the problem instances, which shows that LDR-AS outperforms PSA in terms of the solution quality. The superiority of LDR-AS is more significant when the number of machines increases. This indicates that the proposed LDR-AS is more efficient for solving large-scale instances. Also, LDR-AS uses less solution time compared with PSA.

Table 2.14 summarizes the number of best solutions obtained by PSA, LDR-I-AS and LDR-II-AS on Test-1, Test-2 and Test-3. As shown in this table, LDR-I-AS has found the best solutions for 572 instances, LDR-II-AS obtains the best solutions for 827 instances compared with 341 by PSA. In total, LDR-I-AS and LDR-II-AS altogether has found the best solutions for 1016 instances among the total of 1080 instances.

2.7 Conclusions

In this paper, we address the customer order scheduling problem in parallel production environment with multiple products in each order. Each product needs to be produced on a dedicated machine in parallel and the objective is to minimize the total completion time of the customer orders. By exploring the problem structure and extracting the important features from the well-known dispatching rules manually designed in the literature, we propose an artificial intelligence method, denoted by the feature-enhanced genetic programming (FEGP), to achieve the automatic design of dispatching rules for this

problem. Through an off-line learning manner, two new dispatching rules are automatically generated by the proposed FEGP method. Utilizing the high-quality initial solutions provided by the new dispatching rules as the first stage, we develop an adaptive local search algorithm for the further solution quality improvement to form the two-stage optimization method for this problem. Based on the large-scale testing on a total of 1080 instances, two learned dispatching rules outperforms all the efficient well-known dispatching rules manually designed for this problem previously. The proposed two-stage optimization method also achieves the best computational performance using less solution time compared with the state-of-the-art methods in the literature and is capable of finding the best solutions for almost all the benchmark instances. Future work should be focused on improving the search efficiency of the proposed FEGP method to obtain better dispatching rules and extending the proposed method to other complex production systems with uncertainty. In addition, we will investigate to apply the proposed FEGP method into practical case studies to further validate its effectiveness on handling practical production problems.

CHAPTER 3

INTEGRATION OF DISCRETE-EVENT DYNAMICS AND MACHINE DYNAMICS FOR MACHINE TOOL: MODELING, ANALYSIS AND ALGORITHM

This study integrates machining dynamics with discrete-event dynamics in machine shop operations to reduce machining costs. A learning-based cost function is developed, leveraging predicted tool life across stable cutting speeds to optimize speed selection. Mathematical models are then formulated to address production objectives including cost, makespan, and due date. Numerical results validate the effectiveness and practical applicability of the proposed approach.

3.1 Introduction

The daily operations of machine tools aim to produce products to fulfill orders with various production requirements at the minimum cost. This involves both the machining dynamics that govern the material cutting by the machine tool, and the discrete-event dynamics that govern the production of a series of parts queued at the machine tool. Studies in the two fields have independently made important contributions to the operational excellence of machine tools in production environments.

The research of machining dynamics has laid a solid foundation for the operational excellence of machine tool by providing a set of stable cutting parameters, including spindle speed. The related studies focus on the physical constraints and modeling of machining processes [52, 53], the stable and chatter machining parameters identification

[54, 55], tool wear and tool life prediction [1, 56], cutting force modeling and measurement [57, 58] and etc. The machining of a single part is typically considered in these research studies. However, few research efforts have considered the practical production requirements for hundreds or even thousands of parts from multiple accepted orders.

In the meantime, the discrete-event dynamics research for manufacturing typically considers the production scheduling of a series of parts that are in the queue of the machine tool. The operational excellence is usually achieved at the machine shop level by considering various production requirements, such as order scheduling on parallel machines [13, 15, 59], production scheduling on batch machines [26, 60], buffers for machines [61, 62], due date requirement [16] and etc. However, these studies do not consider the physical constraints and practical operating of the machine tools, such as the machining stability and tool life, which can significantly impact the accuracy of processing times. Most existing studies assume that the processing times are fixed and known. As such, the solutions from these studies are often impractical.

Therefore, the integration of machining dynamics and discrete-event dynamics is critical and promising. In the literature, a few studies have considered the production scheduling of machine tool, such as machine tool feed rate scheduling for processing parts with various geometry [63, 64], computer numerical control (CNC) machine scheduling with controllable processing times constrained by tool life [65, 66], tool replacement scheduling to balance cost of machining conditions, product quality and tool life [67, 68]. However, to our knowledge, no research to date addresses how to integrate machining dynamics, including the machining stability, various cutting speeds and tool life, into

discrete-event dynamics for the operational excellence of machine tool. We seek to fill this gap.

In this paper, we address the problem of integration of discrete-event dynamics and machining dynamics, denoted by IDM, to increase the profit of machine shop. In industrial practice, cutting speed, feed, depth of cut along with cutting geometry and tool material system (substrate, coating, and type of coating method) make a difference in the cutting time, tool life, makespan and eventually the cost. As the first attempt for such integration, the scope of this paper is to select an appropriate cutting speed for each order to reduce the total cost. We analyze the problem under two scenarios of the practical production constraints with increasing complexities: 1) The integrated optimization of cost and makespan; 2) The integrated optimization of cost and due date. While only optimizing cost is the foundation, the above two scenarios increase the fidelity of our studied problem to practical production. We show that we can take advantage of high speed machining and proper production schedule to reduce the overall cost.

We organize the rest of the paper as follows. In Section 2, we present the detailed problem settings for both the discrete-event dynamics and the machining dynamics. A learning-based cost function for machine tool is given in Section 3. Different production scenarios are studied in Sections 4 and 5. The mathematical optimization models, analysis and algorithms are provided. In Section 6, we present numerical results for the proposed methods. Section 7 concludes this paper.

3.2 Problem Setting

In this section, we describe our problem settings of IDM in terms of orders and parts, stable spindle speed, processing time and tool life, and machine shop production and the cost.

3.2.1 Discrete-event Dynamics: Order and Part Setting

We specify the production for n accumulated orders, each consisting of a number of parts. For example, a machine shop may receive more than 10 orders from the master production plan for daily operations. We address how to schedule the production of the machine tool by identifying the best speeds for each of these orders and scheduling their sequences in the queue of the machine tool to save total production cost.

Mathematically, we define the set of n orders as $\mathcal{J} = \{1, 2, \dots, n\}$. Each order consists of a specific quantity of parts, denoted by l_j , $j \in \mathcal{J}$. In this paper, we assume all the parts of an order are the same type, but parts for different orders can be different. The different part types can be resulted by different situations, such as material type, part geometry, and etc. For the single type of part in an order j , let the volume to be remove for each part be v_j , such that the total volume to be removed (i.e., workload of order j) can be calculated by $v_j l_j$. All the orders are released at time zero. A due data d_j is associated for order j before which all parts of the order must be produced.

The challenges for such discrete-event dynamics research are that the processing times of orders are various due to different cutting speeds for different part types. To obtain the operational excellence for machine tool, the critical issue is to obtain the accurate estimation of processing time for orders.

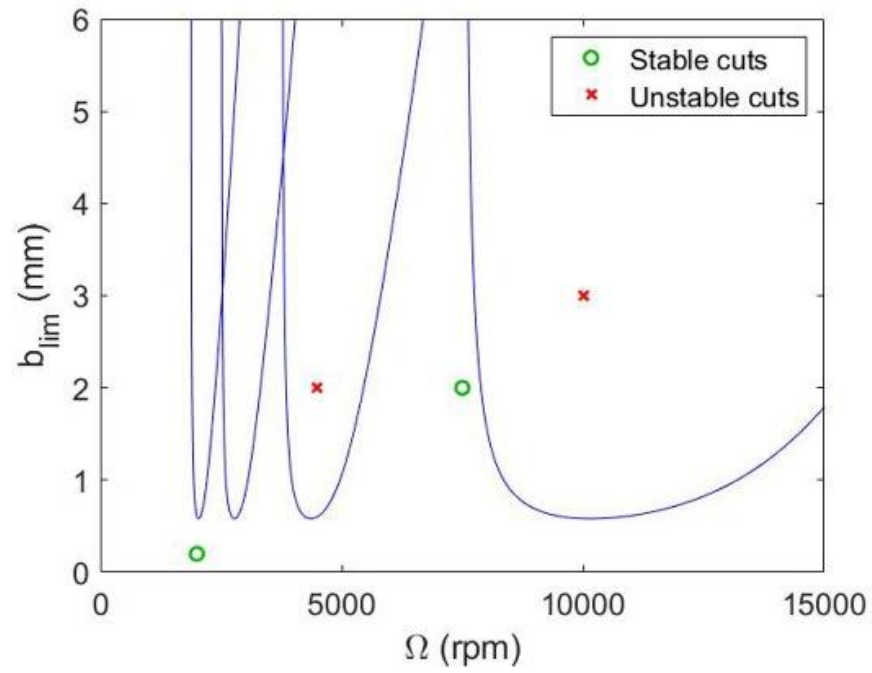


Figure 3.1. Stability lobe diagram (SLD) to determine the stable and unstable cutting (chatter) by choosing parameters spindle speed (Ω) and axial depth of cut (b).

3.2.2 Machining Dynamics: Spindle Speed and Tool Life Setting

Machining dynamics provides the foundation for accurate processing time estimation via the machining stability and tool life estimation under a wide variety of spindle speeds. Without loss of generality, we adopt the milling settings throughout this paper. The stability lobe diagram (SLD) enables to select the best stable spindle speeds at increased axial depth of cut, as shown in Figure 3.1. The machining process can either be stable, that are the green circles below the stability boundary (blue curve), and chatter (red cross). In practice, for a given tool-material combination, the SLD can be obtained by conducting impact test. For more details, see the SLD and its governing relationship in [69]. In this paper, we assume the stable spindle speeds are selected from SLD generated by impact test.

Given a stable spindle speed Ω by SLD and a part with the volume to be removed as v , the machining time then can be calculated as

$$t = \frac{v}{MRR} = \frac{v}{abf_t N_t \Omega} \quad (1)$$

where MRR is the material removal rate defined as the average volume of material removed per unit time with a chosen stable spindle speed Ω from the SLD, a and b the radial and axial depth of cut, and f_t is the feed per tooth. In this paper, we calculate t by considering various Ω with fixed a , b and f_t as the same setting in [1].

Notably, the tool life is decided directly given by the machining parameters, like the spindle speed Ω . In general, this is given by the Taylor's tool life equation [70]:

$$VT^N = C, \quad (2)$$

Where the cutting speed V is the peripheral velocity by $V = \pi D\Omega$, D is the diameter of the tool, and T is the tool life. N and F are constants. Whilst higher spindle speed can increase the machining efficiency of jobs with less machining time t , it leads to reduction of tool life. Thus, there exists a balance of tool life and the machining time which will eventually impact the cost. To obtain the tool life model is usually by machine learning methods. In this paper, we use a state-of-the-art machine learning model in [1] to generate the tool life as input for our method, as discussed in Section 3.

3.2.3 Production and Cost Settings

For production setting, we specify there is a single machine tool. There are sufficient tool edges of the same type for processing all the n orders. In practice, it is often infeasible to complete a fractional number of a part for a tool edge as it requires to change the tool edge in the middle of the cut. As a result, when the rest of tool life is not long enough for machining a single part, the tool edge needs to be changed with a new one. The orders are produced sequentially in the queue of the machine tool. No preemption and interruption of orders are allowed. For all parts in one order, the chosen cutting speed is the same. For different orders, the cutting speeds can be different.

The objective is to minimize the overall cost introduced by machine tool operating cost for machining parts plus the tool edges. In [1], a cost model per part was provided as

$$C_p = r_m t + \frac{(r_m t_{ch} + C_{te})t}{T}, \quad (3)$$

where we denote r_m , t_{ch} , C_{te} and C_p as the machine tool operation cost in \$ per unit time, the time required for changing a tool edge, the cost in \$ per tool edge and the total cost per

part, respectively. This model, however, considers the cost for a single part, rather than the overall cost of orders for hundreds or even thousands of parts made by different materials.

In this study, we extend the above model to represent the overall cost for producing multiple parts in multiple orders as

$$C_0 = \sum_{j \in J} [r_m p_j + (r_m t_{ch} + C_{te}) q_j], \quad (4)$$

where the first term $r_m p_j$ calculates the machine tool operating cost for processing order j and p_j refers to the total machining time with a chosen speed. The second term $(r_m t_{ch} + C_{te}) q_j$ calculates the cost induced by changing tool edges, and q_j is the number of tool edges needed for processing order j .

Eq. (4) is the generalized form of Eq. (3) to include all orders in the practical machine shop production. In next section, we will present a new cost function through the calculation of p_j and q_j under the context of multiple orders in machine shop.

3.3 Learning-based Cost Function for Machine Tool

The determination of cost of the orders depends on the machining time and number of tool edges used, which is directly decided by the prediction of tool life. In this section, we present a new cost function based on the tool life prediction in the sense of expectation with a state-of-the-art machine learning model.

3.3.1 A Machine Learning Model for Tool Life Prediction

Karandikar et al. (2021) [1] developed a physics-guided logistic regression model to predict tool worn probability with cutting speed V and cutting time T as inputs. Through

introducing into machine learning Taylor's power law-based tool life description (see Eq. (2)), the model has a powerful performance in prediction accuracy. The logistic decision boundary of the model, that is, when the probability of tool worn is 0.5, provides an accurate estimation for tool life in the sense of expectation as

$$\log(T) = -\frac{\theta_1}{\theta_2}\log(V) - \frac{\theta_0}{\theta_2}, \quad (5)$$

where the coefficients $\theta_0 = -89.57$, $\theta_1 = 13.57$ and $\theta_2 = 5.26$ are estimated through experimental data for a single-insert end-mill (Kennametal KICR073SD30333C) with $D = 19.05$ mm diameter and a square uncoated carbide insert (Kennametal 107,888,126 C9 JC). Equivalently, this model can be transformed into the below Taylor's tool life model after using logarithmic transformation,

$$T = \left(\frac{C}{V}\right)^{\frac{1}{N}} = \left(\frac{C}{\pi D \Omega}\right)^{\frac{1}{N}}, \quad (6)$$

where $N = 0.388$ and $C = 735.2$ m/min using the above values of θ_0 , θ_1 and θ_2 . The logistic model in the logarithmic and original space of V and T is shown in the two panels of Figure 3.2. The gray scale indicates the probability of tool worn with a cutting speed V (horizontal axis) and cutting time T (vertical axis). We will use Eq. (6) throughout this paper for tool life prediction.

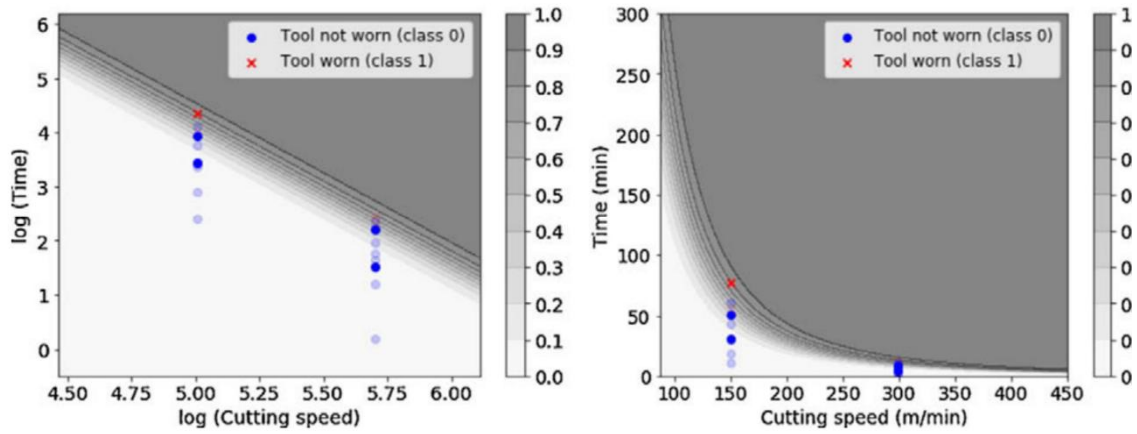


Figure 3.2. Probability of tool worn as a function of cutting speed (logarithmic/original scale in left/right panel) from the trained logistic model; the train data points are shown in light blue, while the test data points in dark blue (tool not worn) and red cross (tool worn). The gray scale indicates the probability of tool worn with a cutting speed (horizontal axis) and cutting time (vertical axis). Good extrapolation outside of the train data space of cutting speed can be obtained, as shown by the dark blue dots and red cross. This extrapolation capability enables the accurate prediction of tool life for wide range of cutting speeds, which will be used in the experiments. Figures are reproduced from [1].

3.3.2 Calculation of Learning-based Cost Function

Based on the learning-based tool life model in [1], a cost function is developed for multiple orders. As the parts for different orders can be made by different materials, the SLD of the same tool edge for different orders (parts) are different. As a result, the candidate stable spindle speeds for orders are also different. Thus, we define the matrix of spindle speed as

$$\mathbf{\Omega} = \begin{bmatrix} \Omega_{11} & \Omega_{12} & \cdots & \Omega_{1n} \\ \Omega_{21} & \Omega_{22} & \cdots & \Omega_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \Omega_{m1} & \Omega_{m2} & \cdots & \Omega_{mn} \end{bmatrix} \quad (7)$$

where the entry Ω_{ij} refers to the spindle speed for processing order j with spindle speed i , $i \in \mathcal{I}$, and $\mathcal{J} = \{1, 2, \dots, m\}$ is defined as the index set of available stable spindle speed.

For a given spindle speed Ω_{ij} to process order j , the tool life T_{ij} is calculated by Eq. (6). Also, the total machining time p_{ij} for processing order j with spindle speed Ω_{ij} calculated as

$$p_{ij} = \frac{v_j l_j}{ab f_t N_t \Omega_{ij}}, \quad (8)$$

where $v_j l_j$ indicates total volume of order j by multiplying its part quantity l_j with the part volume v_j . When arranged T_{ij} and p_{ij} in matrix form, we can obtain the below matrices for tool life and machining time:

$$\mathbf{T} = \begin{bmatrix} T_{11} & T_{12} & \cdots & T_{1n} \\ T_{21} & T_{22} & \cdots & T_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ T_{m1} & T_{m2} & \cdots & T_{mn} \end{bmatrix} \text{ and } \mathbf{P} = \begin{bmatrix} P_{11} & P_{12} & \cdots & P_{1n} \\ P_{21} & P_{22} & \cdots & P_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ P_{m1} & P_{m2} & \cdots & P_{mn} \end{bmatrix}, \quad (9)$$

Using $\mathbf{T}$, we can now compute the number of tool edges needed for each order with different spindle speeds as matrix

$$\mathbf{Q} = \begin{bmatrix} q_{11} & q_{12} & \cdots & q_{1n} \\ q_{21} & q_{22} & \cdots & q_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ q_{m1} & q_{m2} & \cdots & q_{mn} \end{bmatrix}, \quad (10)$$

where each entry is calculated by

$$q_{ij} = \left\lceil \frac{l_j}{\lfloor T_{ij}/t_{ij} \rfloor} \right\rceil, \text{ and } t_{ij} = \frac{v_j}{abf_t N_t \Omega_{ij}}. \quad (11)$$

The term t_{ij} is the machining time for a single part of order j with spindle speed Ω_{ij} . Thus, with floor function, $\lfloor T_{ij}/t_{ij} \rfloor$ refers to the number of parts of order j that can be finished by a single tool edge. Finally, the number of tool edges for processing l_j parts can be obtained by the ceiling function.

We can further obtain the summation of total machining time and tool edge change time for each order under all its candidate spindle speeds. This summation, denoted by h_{ij} , represents the total processing time occupied by order j if with speed i :

$$h_{ij} = p_{ij} + t_{ch} q_{ij}. \quad (12)$$

The associated total time matrix is denoted by

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1n} \\ h_{21} & h_{22} & \cdots & h_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ h_{m1} & h_{m2} & \cdots & h_{mn} \end{bmatrix}.$$

Finally, with matrices $\mathbf{P}$, $\mathbf{Q}$ and $\mathbf{H}$, we can obtain the cost function for processing all orders as

$$f_{cost}(x) = \sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{I}} (r_m h_{ij} + C_{te} q_{ij}) x_{ij}. \quad (14)$$

Note here we introduce a binary variable x_{ij} defined as

$$x_{ij} := \begin{cases} 1, & \text{if speed } i \text{ is chosen for processing order } j, \\ 0, & \text{otherwise.} \end{cases} \quad (15)$$

Eq. (14) integrates both the machining dynamics, that is, term $r_m h_{ij} + C_{te} q_{ij}$ based on cost function for parts of orders, and the discrete-event dynamics, that is, discrete variable x_{ij} in Eq. (15) for choosing different speeds for different orders to achieve the reduction of production cost.

In next section, we will optimize the speed selection through controlling x_{ij} under the discrete-event dynamics environment of machine shop such that $f_{cost}(x)$ can be minimized for processing all orders. Before that, we show an illustrative example.

3.3.3 Illustration of Cost Saving by the Proposed Integration

As an illustrative example of our proposed integration, we show two scenarios in Figure 3.3 for processing an order: with faster speed in (a) obtained by the expected cost method in [1], and slower speed in (b) by solving our developed M1 model in Section 4. The light grey period is for tool machining, the light green period is for changing the tool edge. The red dashed sections refer to the waste time that is not enough for machining a single part before its tool life comes to end, while the one in the last tool edge is wasted because all parts are done.

By choosing slower speed in (b), we process one more part per tool edge in (b) in comparison with (a), namely from 9 part per tool edge to 10. The last three parts on the 4th

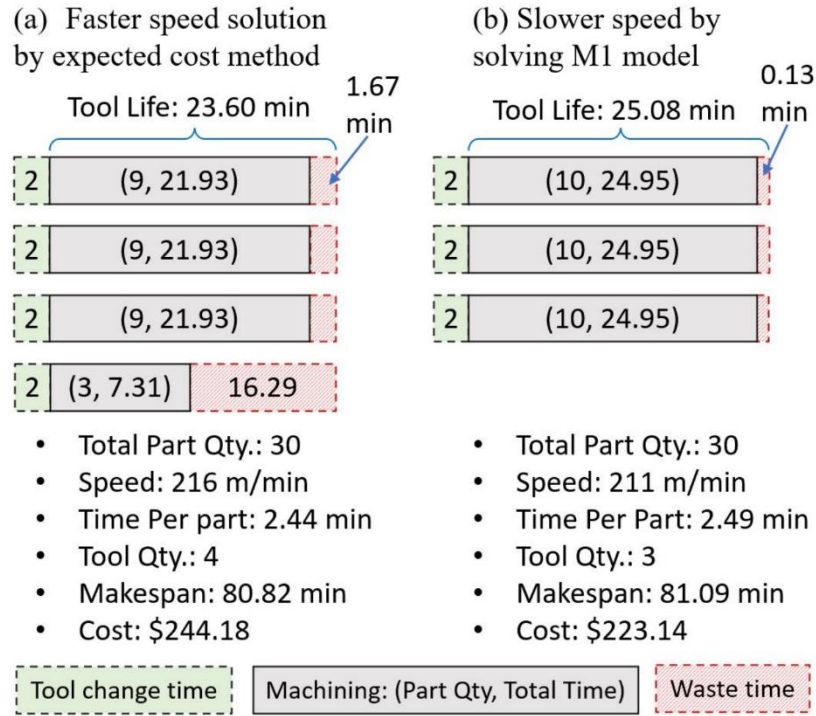


Figure 3.3. Illustrative example for choosing different speeds to process multiple parts of an order by the expected cost method in [1] and solving our developed model M1 in Section 4. By choosing slower speed, we process one more part per tool edge in (b) in comparison with (a). The last three parts on the 4th tool edge of (a) are thus moved back to the 3rd tool edge. This leads to saving the last tool edge and reducing the waste time for each tool edge in (a), and importantly, reducing the cost from \$244.18 to \$223.14. See details in Section 6.

tool edge of (a) are thus moved back to be processed with the 3rd tool edge, saving the last tool edge and reducing the waste time for each tool edge in (a). Notably, the production cost is reduced from \$244.18 to \$223.14. The makespan is increased from 80.82 min to 81.09 min as a sacrifice of the reduced cost. Such trade-off between cost and makespan is meaningful in practical production. Note Figure 3.3 only gives the case when smaller speed will lead to cost reduction, it is more often and realistic to reduce cost by increasing the speed for high speed machining, and this exactly shows our motivation for this study.

3.4 Integrated Optimization of Cost and Makespan

We denote our problem when only cost is considered as IDM-C, which is studied in Sections 4.1–4.2. In Section 4.3, we study the IDM problem when integrated optimization of cost and makespan is considered, denoted by IDM-CM. Integer optimization (IO) models are presented for both problems.

3.4.1 Model with Only Cost Consideration

When only cost is under consideration, the below IO model M1 aims to find the optimal spindle speed for each order, such that the cost function can be minimized.

$$\min f_{cost}(x) \quad (M1)$$

$$s. t. \sum_{i \in \mathcal{I}} x_{ij} = 1, \quad \forall j \in \mathcal{J}, \quad (16)$$

$$x_{ij} \in \{0,1\}, \quad \forall i \in \mathcal{I}, j \in \mathcal{J} \quad (17)$$

The objective is to minimize the overall cost $f_{cost}(x)$ of producing n orders. Constraints (16) ensure that each order must be processed by only one spindle speed. Constraints (17) provide the binary variable restriction on x_{ij} .

We show the constraint matrix of formulation M1 is totally unimodular. Thus, the IO model can be solved as a linear programming (LP) model. To do this, we arrange the variables in the sequence of $x_{11}, x_{21}, \dots, x_{m1}; x_{12}, \dots, x_{m2}; x_{1n}, \dots, x_{mn}$ as columns of the constraint matrix. The constraint matrix $\mathbf{A} \in \mathbb{Z}^{n \times mn}$ for constraints (16) is as below:

$$\mathbf{H} = \begin{bmatrix} 1^{th} \dots m^{th} (m+1)^{th} \dots 2m^{th} \dots ((n-1)m+1)^{th} \dots nm^{th} \\ 1 \dots 1 & 0 & \dots & 0 & \dots & 0 & \dots & 0 \\ 0 \dots 0 & 1 & \dots & 1 & \dots & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 \dots 0 & 0 & \dots & 0 & \dots & \dots & \dots & 1 \end{bmatrix}. \quad (18)$$

The above 0, 1 matrix $\mathbf{A}$ is totally unimodular because it satisfies the following two conditions: (1) there are at most two nonzero elements in each column of $\mathbf{A}$, which is number 1 and there is only one entry for each column to be 1; (2) $\mathbf{A}$ admits an equitable row-bicoloring. In particular, the rows of $\mathbf{A}$ can be partitioned into two sets, say “red set” as row 1 and “blue set” as rows 2 to n . It’s easy to check that the sum of the red rows minus the sum of the blue rows is a vector with entries ± 1 .

As a result, the polyhedron for formulation M1 is integral, which has integral extreme points. Thus, it always admits an integral solution and is equivalent to solving the LP relaxation:

$$\min f_{cost}(x) \quad (M1 - LP)$$

$$s. t. \sum_{i \in \mathcal{I}} x_{ij} = 1, \quad \forall j \in \mathcal{J}, \quad (19)$$

$$x_{ij} \in [0,1], \quad \forall i \in \mathcal{I}, j \in \mathcal{J} \quad (20)$$

where we relax the binary variable constraints $x_{ij} \in \{0,1\}$ into the interval $x_{ij} \in [0,1]$.

Next, we develop an exact algorithm.

3.4.2 Decomposition-based Greedy Algorithms

Formulation M1 naturally adopts a decomposition for both the objective and the constraints for each order. While every order must be assigned with a spindle speed, there are no additional constraints that couples the assignment of spindle speed for different orders. Therefore, the assignment of spindle speed for each order is independent. That is, for each order j we can select the optimal speed from m candidates that minimize its cost $r_m h_{ij} + C_{te} q_{ij}$ in the objective. In general, we denote the described procedure as Decomposition-based Greedy (DG) and summarize as the below Algorithm 3.1.

For Algorithm 3.1, we have the below lemma to show the optimality of its solution.

Lemma 4.1. Given n orders and m candidate speeds, Algorithm 3.1 solves problem IDM-C optimally in $O(18mn + n)$ times.

Proof. To prove DG algorithm is optimal, we do it by contradiction. Suppose the solution of DG is not optimal, there must exist at least one order, denoted by $\hat{j}$, that selects a spindle speed with smaller cost $r_m h_{ij} + C_{te} q_{ij}$ than $\hat{j}$. This contradicts with steps 7–9 of DG, which selects the spindle speed with the smallest cost. Thus, Algorithm 3.1 is an exact algorithm.

Next, we consider the solution time of Algorithm 3.1. For preparing **T**, **Q**, **P** and **H**, it takes 2 operations to calculate T_{ij} , 6 operations to calculate q_{ij} , 3 operations to calculate p_{ij} and 2 operations to calculate h_{ij} . Together with the coupled loops, this takes $14mn$ times. For each j , it takes $3m$ operations to compute Obj_{ij} for all $i \in \mathcal{I}$, m operations to find the speed i^* with minimum Obj_{ij} , and 1 operation to update the overall

cost $f_{cost}(x)$, respectively. To sum up, we can get the total time complexity of Algorithm 3.1 is $O(18mn + n)$, which complete the proof.

3.4.3 Integration with Makespan

Makespan constraint refers to the hard constraint that all the orders must be completed within a given time period. For example, this can describe the fixed time length for the worker shift. To model problem IDM-CM, we need an additional parameter:

c : a given constant of the upper bound for the makespan.

With this additional parameter, we can model problem IDM-CM by modifying formulation M1 with makespan constraint as below IO model, denoted by M2:

$$\begin{aligned} \min f_{cost}(x) & \quad (M2) \\ s. t. \sum_{i \in \mathcal{I}} x_{ij} &= 1, \quad \forall j \in \mathcal{J} & (21) \\ \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} h_{ij} x_{ij} &\leq c, & (22) \\ x_{ij} &\in \{0,1\}, \quad \forall i \in \mathcal{I}, j \in \mathcal{J} & (23) \end{aligned}$$

Constraint (22) ensures the makespan of producing all the orders not exceeding the given time length c . The makespan constraint adds additional complexity for the problem. We remark the complexity of problem IDM-CM is still open.

3.5 Integrated Optimization of Cost and Due Dates

We consider problem IDM when the integrated optimization of cost and due date is to be achieved, denoted by IDM-CD. Due to the due date constraints, the sequences of order processing need to be considered to determine the cost. We propose three categories of

model based on the choices of decision variables which determine the order sequences. That are, completion time variables (5.1), linear order variables (5.2) and positional variables (5.3) of orders, respectively. In the first category, we propose a mixed-integer optimization (MIO) model that can be efficiently solved by off-the-shelf optimization solvers. The latter two categories of model involve quadratic constraints. Considering that off-the-shelf optimization solvers cannot efficiently deal with this kind of nonlinear problem, we adopt linearization strategies for the proposed formulations.

3.5.1 Modeling with Completion Time Variables

In the first model, we use completion time variables to indicate the order sequence for problem IDM-CD. The needed additional variables are defined as follows:

y_{jk} : $y_{ij} = 1$ if order j is processed before order k , $\forall j, k \in \mathcal{J}$ and $j \neq k$. Otherwise

$$y_{ij} = 0;$$

C_j : completion time of order j , $\forall j \in \mathcal{J}$.

The problem IDM-CD can be formulated as below.

$$\min f_{cost}(x) \tag{M3-1}$$

$$\text{s. t. } \sum_{i \in \mathcal{J}} x_{ij} = 1, \quad \forall j \in \mathcal{J}, \tag{24}$$

$$C_j \geq \sum_{i \in \mathcal{J}} h_{ij} x_{ij}, \quad \forall j \in \mathcal{J}, \tag{25}$$

$$C_j + \sum_{i \in \mathcal{J}} h_{ik} x_{ik} \leq C_k + M(1 - y_{jk}), \quad \forall j < k \in \mathcal{J}, \tag{26}$$

$$C_k + \sum_{i \in \mathcal{J}} h_{ij} x_{ij} \leq C_j + M y_{jk}, \quad \forall j < k \in \mathcal{J}, \tag{27}$$

$$C_j \in [0, d_j], \quad \forall j \in \mathcal{J}, \tag{28}$$

$$x_{ij}, y_{jk} \in \{0, 1\}, \quad \forall i \in \mathcal{I}, j, k \in \mathcal{J}. \tag{29}$$

Constraint (25) ensure that the completion time of each order is no later than its processing time. Constraints (26), (27) are disjunctive constraints which ensure that either order j is processed before order k , i.e. $y_{jk} = 1$ in constraints (26), or order k is before order j , that is $y_{jk} = 0$ in constraints (27). M is a sufficiently large number. Constraints (28) enforce the completion time is non-negative and no larger than the due date. In this formulation, the M is a natural upper bound for C_j for arbitrary $j \in \mathcal{J}$. Thus, we can set $M = \sum_{j \in \mathcal{J}} \max_{i \in \mathcal{J}} \{h_{ij}\}$. The total number of variables for formulation M3-1 is $n^2 + mn + n$, which includes $n^2 + mn$ binary variables and $n^2 + 3n$ constraints.

Remark 1. *If $d_j = c$ for $j \in \mathcal{J}$, problem IDM-CD is reduced to problem IDM-CM.*

To check the above remark is valid, it's obvious to see that when $d_j = c$ for $j \in \mathcal{J}$, then the makespan of any feasible solutions to problem IDM-CD also satisfy the makespan limit c . Thus, problem IDM-CM is a special case of problem IDM-CD. It is more general for us to study problem IDM-CD.

3.5.2 Modeling with Linear Ordering Variables

3.5.2.1 Proposed Integer Optimization Model

This formulation is based on the linear order variables y_{jk} and y_{kj} that either of them should be equal to 1 for arbitrary $j < k \in \mathcal{J}$. The following formulation M3-2 is considered.

$$\min f_{cost}(x) \quad (M3-2)$$

$$s. t. \sum_{i \in \mathcal{J}} x_{ij} = 1, \quad \forall j \in \mathcal{J}, \quad (30)$$

$$y_{jk} + y_{kj} = 1 \quad \forall j, k \in \mathcal{J}, j < k, \quad (31)$$

$$y_{jk} + y_{kl} + y_{lj} \leq 2, \quad \forall j, k, l \in \mathcal{J}, j \neq k \neq l, \quad (32)$$

$$\sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{J}} h_{ij} x_{ij} y_{jk} + \sum_{i \in \mathcal{J}} h_{ik} x_{ik} \leq d_k \quad \forall k \in \mathcal{J}, \quad (33)$$

$$x_{ij}, y_{jk} \in \{0, 1\}, \quad \forall i \in \mathcal{J}, j, k \in \mathcal{J} \quad (34)$$

Constraints (31) are a set of conflict constraints, which ensure that either order j is processed before order k or vice versa. Note here $j < k$ because of symmetry. Constraints (32) indicate the transitivity constraints that ensure a linear order of three arbitrary orders. Constraints (33) indicate the starting time of order k plus its processing time must be no larger than its due date. The total number of variables of formulation M3-2 with m speeds and n orders is $n^2 + mn$, which are all binary variables and involve $n^3 - 2.5n^2 + 3.5n$ constraints.

Note that M3-2 is a quadratically constrained IO formulation due to constraints (33). Considering that the current optimization solvers like CPLEX cannot efficiently deal with this kind of nonlinear problem, we adopt linearization strategies for this formulation in the following section.

3.5.2.2 Reformulation with Linearization Technique

To enhance the solvability of formulation M3-2, we first apply the linearization technique proposed in [71] to derive an MIO model, which is equivalent to M3-2. For arbitrary $j \in \mathcal{J}$, $k \in \mathcal{J}$, let $\alpha_{jk} = (\sum_{i \in \mathcal{J}} h_{ij} x_{ij}) y_{jk}$ and substitute the cross-product $\sum_{i \in \mathcal{J}} h_{ij} x_{ij} y_{jk}$ in M3-2 using the following inequalities:

$$\begin{cases} \sum_{i \in \mathcal{J}} h_{ij} x_{ij} - M(1 - y_{jk}) \leq \alpha_{jk} \leq \sum_{i \in \mathcal{J}} h_{ij} x_{ij} + M(1 - y_{jk}), \\ -My_{jk} \leq \alpha_{jk} \leq My_{jk}, \end{cases} \quad (35)$$

where M is a sufficiently large number. For arbitrary $j, k \in \mathcal{J}$, if $y_{jk} = 1$, then $\alpha_{jk} = \sum_{i \in \mathcal{J}} h_{ij} x_{ij}$, and the inequalities become $\sum_{i \in \mathcal{J}} h_{ij} x_{ij} \leq \alpha_{jk} \leq \sum_{i \in \mathcal{J}} h_{ij} x_{ij}$ and $-M \leq \alpha_{jk} \leq M$. Clearly, the equivalence holds. Otherwise, if $y_{jk} = 0$, the inequalities are $\sum_{i \in \mathcal{J}} h_{ij} x_{ij} - M \leq \alpha_{jk} \leq \sum_{i \in \mathcal{J}} h_{ij} x_{ij} + M$ and $0 \leq \alpha_{jk} \leq 0$, which indicates $\alpha_{jk} = 0$. Thus, we get an equivalent model to M3-2.

We further reduce the above inequality constraints (35) as

$$\begin{cases} \sum_{i \in \mathcal{J}} h_{ij} x_{ij} - M(1 - y_{jk}) \leq \alpha_{jk}, \\ 0 \leq \alpha_{jk} \leq My_{jk}. \end{cases} \quad (36)$$

Thus, we obtain a new linearized formulation M3-2-L as below:

$$\min f_{cost}(x) \quad (M3 - 2 - L)$$

$$s. t. \text{ Constraints (30) - (32)} \quad (37)$$

$$\sum_{j \in \mathcal{J}} \alpha_{jk} + \sum_{i \in \mathcal{J}} h_{ik} x_{ik} \leq d_k, \quad \forall k \in \mathcal{J}, \quad (38)$$

$$\sum_{i \in \mathcal{J}} h_{ij} x_{ij} - M(1 - y_{jk}) \leq \alpha_{jk} \leq My_{jk}, \quad \forall j, k \in \mathcal{J}, \quad (39)$$

$$\alpha_{jk} \geq 0, \quad \forall j, k \in \mathcal{J}, \quad (40)$$

$$x_{ij}, y_{jk} \in \{0, 1\}, \quad \forall i \in \mathcal{I}, j, k \in \mathcal{J}. \quad (41)$$

This formulation is obtained by eliminating the right-hand-side of the first inequality and the left-hand-side of the second inequality in (35). The value of M is set as the maximum total processing time of the orders, that is $M = \max_{i,j} h_{ij}$. The total number of variables

is $2n^2 + mn$, which involves $n^2 + mn$ binary variables and $n^3 + 0.5n^2 + 3.5n$ constraints. For formulations M3-2L and M3-2, we have the below Lemma 5.1.

Lemma 5.1. *For arbitrary optimal solution to M3-2-L, there exists an optimal solution to M3-2 with the same objective value.*

Proof. For arbitrary optimal solution (x^*, y^*, α^*) to M3-2-L, we can show (x^*, y^*) as an optimal solution to M3-2 with the same objective value.

We first show (x^*, y^*) is a feasible solution to M3-2. For arbitrary $j, k \in \mathcal{J}$, if $y_{jk}^* = 0$, then $\alpha_{jk}^* = 0$ based on constraints (39) and (40). Then we have $\sum_{i \in \mathcal{J}} h_{ik} x_{ik}^* \leq d_k$ in constraints (38). This indicates (x^*, y^*) satisfies constraints (33) in formulation M3-2 when $y_{jk}^* = 0$. If $y_{jk}^* = 1$, then by substituting into constraints (39) first and then (38), we have

$$\sum_{i \in \mathcal{J}} h_{ij} x_{ij}^* \leq \alpha_{jk}^* \leq M, \quad (42)$$

$$\sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{J}} h_{ij} x_{ij}^* + \sum_{i \in \mathcal{J}} h_{ik} x_{ik}^* \leq \sum_{j \in \mathcal{J}} \alpha_{jk}^* + \sum_{i \in \mathcal{J}} h_{ik} x_{ik}^* \leq d_k \quad (43)$$

Inequalities (43) indicate (x^*, y^*) satisfies constraints (33) when $y_{jk}^* = 1$. To sum up, (x^*, y^*) is a feasible solution to M3-2.

Next, we prove the objective value of (x^*, y^*) is less than or equal to that of any feasible solutions to M3-2. Let $\mathcal{S}_2$ be the solution space of formulation M3-2, i.e.,

$$\mathcal{S}_2 := \{(x, y): \text{Constraints (30) -- (34)}\}. \quad (44)$$

As M3-2-L is a relaxation of M3-2, the objective value of the optimal solution (x^*, y^*, α^*) will be less than or equal to that of any feasible solutions to M3-2. That is

$$f_{cost}(x^*) = \sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{J}} (r_m h_{ij} + C_{te} q_{ij}) x_{ij}^* \leq f_{cost}(x), \quad \forall (x, y) \in \mathcal{S}_2 \quad (45)$$

Note $f_{cost}(x^*)$ is also the objective value of feasible solution (x^*, y^*) to M3-2. This indicates the objective value of (x^*, y^*) is no larger than any feasible solutions to M3-2.

Therefore, (x^*, y^*) is an optimal solution to M3-2 with the same objective value $f_{cost}(x^*)$, which completes the proof. ■

3.5.3 Modeling with Positional Variables

3.5.3.1 Proposed Integer Optimization Model

This model is based on the positional variables that describe the relationship between arbitrary order with the position it is assigned in the schedule. The additional decision variables are defined as follows.

z_{jk} : $z_{jk} = 1$ if order j is assigned on position k , $\forall j, k \in \mathcal{J}$. Otherwise $z_{jk} = 0$;

C_k : completion time of the order at position k , $\forall k \in \mathcal{J}$.

A new IO model can be formulated as below.

$$\min f_{cost}(x) \quad (M3 - 3)$$

$$s. t. \sum_{i \in \mathcal{J}} x_{ij} = 1, \quad \forall j \in \mathcal{J}, \quad (46)$$

$$\sum_{k \in \mathcal{J}} z_{jk} = 1, \quad \forall j \in \mathcal{J}, \quad (47)$$

$$\sum_{j \in \mathcal{J}} z_{jk} = 1, \quad \forall k \in \mathcal{J}, \quad (48)$$

$$0 \leq C_k \leq \sum_{j \in \mathcal{J}} d_j z_{jk}, \quad \forall k \in \mathcal{J}, \quad (49)$$

$$C_1 \geq \sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{J}} h_{ij} x_{ij} y_{j1} \quad (50)$$

$$C_k \geq C_{k-1} + \sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{J}} h_{ij} x_{ij} z_{jk} \quad \forall k \in \mathcal{J}/\{1\}, \quad (51)$$

$$x_{ij}, y_{jk} \in \{0,1\}, \quad \forall i \in \mathcal{I}, j, k \in \mathcal{J} \quad (52)$$

Constraints (47), (48) ensure that a particular order can only be assigned to exactly one position and each position can be assigned with exactly one order. Constraints (49) ensure the completion time of order at position k is non-negative and no later than its due time. Constraints (50), (51) describe the completion time of the order at position k . The total number of variables in this formulation is $n^2 + mn + n$, which involves $n^2 + mn$ binary variables and $6n$ constraints. Note that M3-3 is a quadratically constrained IO formulation. We will also apply linearization technique to reformulate M3-3.

3.5.3.2 Reformulation with Linearization Technique

For arbitrary $j, k \in \mathcal{J}$, define a new variable $\beta_{jk} = (\sum_{i \in \mathcal{J}} h_{ij} x_{ij}) z_{jk}$. An equivalent MIO model can be obtained by substituting $\sum_{i \in \mathcal{J}} h_{ij} x_{ij} z_{jk}$ in M3-3 and with the below additional constraints

$$\begin{cases} \sum_{i \in \mathcal{I}} h_{ij}x_{ij} - M(1 - z_{jk}) \leq \beta_{jk} \leq \sum_{i \in \mathcal{I}} h_{ij}x_{ij} + M(1 - z_{jk}), \\ -Mz_{jk} \leq \beta_{jk} \leq My_{jk}, \end{cases} \quad (53)$$

Note the equivalence validation of the above inequalities is the same with (35). For concise presentation we skip it.

We further reduce the above inequality constraints (53) as

$$\begin{cases} \sum_{i \in \mathcal{I}} h_{ij}x_{ij} - M(1 - z_{jk}) \leq \beta_{jk}, \\ 0 \leq \beta_{jk} \leq Mz_{jk}. \end{cases} \quad (54)$$

Thus, we obtain a new linearized formulation M3-2-L as below:

$$\begin{aligned} \min f_{cost}(x) & \quad (M3 - 3 - L) \\ \text{s. t. Constraints (46) - (49)} & \quad (55) \end{aligned}$$

$$C_1 \geq \sum_{j \in \mathcal{J}} \beta_{j1}, \quad (56)$$

$$C_k \geq C_{k-1} \geq \sum_{j \in \mathcal{J}} \beta_{jk}, \quad \forall k \in \mathcal{J}/\{1\}, \quad (57)$$

$$\sum_{i \in \mathcal{I}} h_{ij}x_{ij} - M(1 - z_{jk}) \leq \beta_{jk} \leq Mz_{jk}, \quad \forall j, k \in \mathcal{J}, \quad (58)$$

$$\beta_{jk} \geq 0, \quad \forall j, k \in \mathcal{J}, \quad (59)$$

$$x_{ij}, z_{jk} \in \{0,1\}, \quad \forall i \in \mathcal{I}, j, k \in \mathcal{J}. \quad (60)$$

The value of M is set as the maximum total processing time among all the orders, that is

$M = \max_{i,j} \{h_{ij}\}$. The total number of variables is $2n^2 + mn + n$, including $n^2 + mn$

binary variables and $2n^2 + 6n$ constraints. For formulations M3-3 and M3-3-L, we have

the below Lemma 5.2.

Lemma 5.2 *For arbitrary optimal solution to M3-3-L, there exists an optimal solution to M3-3 with the same objective value.*

Proof. The proof odea is similar to Lemma 5.1. For arbitrary optimal solution (x^*, z^*, C^*, β^*) to M3-3-L, we will construct (x^*, z^*, C^*) as an optimal solution to M3-3 with the same objective value.

We first show (x^*, z^*, C^*) is a feasible solution to M3-3. It is simple to check (x^*, z^*, C^*) is feasible when $z_{jk}^* = 0$ by substituting $z_{jk}^* = 0$ into (58) and then (56) and (57).

Otherwise, when substituting $z_{jk}^* = 1$ we have

$$\sum_{i \in \mathcal{J}} h_{ij} x_{ij}^* \leq \beta_{jk}^* \leq M, \quad \forall j, k \in \mathcal{J} \quad (61)$$

$$C_1^* \geq \sum_{j \in \mathcal{J}} \beta_{j1}^* \geq \sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{J}} h_{ij} x_{ij}^*, \quad (62)$$

$$C_k^* \geq C_{k-1}^* + \sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{J}} h_{ij} x_{ij}^*, \quad \forall k \in \mathcal{J}/\{1\}. \quad (63)$$

Inequalities (62) and (63) indicate (x^*, z^*, C^*) is a feasible solution when $z_{jk}^* = 1$.

Next, similar to Lemma 5.1, we define the solution space of formulation M3-3 as $\mathcal{S}_3 := \{(x, z, C) : \text{Constraints (46) – (52)}\}$, and we can get that $f_{cost}(x^*) \leq f_{cost}(x)$, $\forall (x, z, C) \in \mathcal{S}_3$ as M3-3-L is a relaxation of M3-3. Thus, we conclude (x^*, z^*, C^*) is an optimal solution to M3-3. ■

Based on Lemmas 5.1 and 5.2, we will directly solve models M3-2-L and M3-3-L using MIO solver instead of M3-2 and M3-3, respectively. The comparison of the three MIO formulations is discussed in next section.

3.6 Computational Experiments

Comprehensive computational experiments are conducted for different problem settings. For problem IDM-C, we compare the performance of methods by solving models

M1 and M1-LP with modern optimization solver CPLEX, DG algorithm and the expected cost method (ECM henceforth) from [1]. For problem IDM-CM, we compare methods by solving M2 using CPLEX with ECM. The three MIO models for problem IDM-CD, i.e., M3-1, M3-2L and M3-3-L, are solved by CPLEX and compared in terms of their computational efficiency.

All the methods are coded in Python. The optimization solver CPLEX 20.1.0 is used for solving all the IO and MIO models with the Python interface docplex. All the settings for CPLEX remain default except for time limit as 600 s. The experiments are deployed on a mobile workstation with Intel(R) Xeon(R) W-10885 M CPU @ 2.40 GHz, 128 GB memory, 64 bit Windows 10 Pro operating system for workstations.

3.6.1 Data Generation

Machine tool. The settings for machine tool including the tool geometry, cutting parameters (except for spindle speed set) and tool life model are the same with that in [1]. In particular, the tool is a single-insert endmill with diameter $D = 19.05$ mm and $N_t = 1$. The radial depth of cut is $a = 4.7$ mm, the axial depth of cut is $b = 3$ mm and the feed per tooth is $f_t = 0.06$ mm, respectively. The constants in the Taylor tool life equation are $N = 0.388$ and $C = 735.2$ m/min.

Order and part. The number of orders n is from a set $\{2,4,6,8,10,12,14,16,18,20\}$, with one single part type for each order. Then quantity l_j of the part for one order is an integer number randomly sampled from uniform distribution $U[10, 200]$. The volume v_j to be removed per part is an integer number randomly sampled from uniform distribution $U[5000, 15000]$ with unit mm^3 . The makespan limitation c under each case of n is decided

by considering two values: the makespan of the solution by solving problem IDM-C which only considers the cost, and the minimum makespan when adopting the minimum h_{ij} for each order j . We set c as the integer number by rounding the average of these two values.

Due date. We specify the due date by generating the data without regarding for feasibility and then discarding instances that have no feasible schedule. Specifically, we first obtain the expected cutting speed for each order using the ECM in [1]. Henceforth, the makespan $\bar{C}_{max}$ and the mean value of total processing time $\bar{h}$ among the n orders can be obtained. The initial due date d_j is generated by randomly sampling from uniform distribution $U[0.4\bar{C}_{max}, 0.5\bar{C}_{max}]$. A schedule is then generated using earliest due date rule based on d_j , and the completion time C_j for each order is thus obtained. We then check the feasibility of d_j , i.e., d_j is accepted if $d_j \geq C_j$; otherwise $d_j = C_j + r$, where r is a randomly sampled integer number from uniform distribution $U[0, \bar{h}]$. By this way of setting d_j , we ensure at least the expected speeds are feasible solution to problem IDM-CD.

Production and cost. The candidate cutting speed V is evenly spaced in $[220, 420]$ m/min with the interval 2 m/min. Thus, in total $m = 100$ speeds are used for one order. Note there without loss of generality, we use the same set of cutting speed for all the n orders. The spindle speed matrix Ω is thus constructed, with the equivalent minimum spindle speed 3676.02 rpm and the maximum spindle speed 6984.44 rpm. Lastly, the coefficients of the cost model are set as $r_m = 1$ \$/min, $t_{ch} = 2$ min and $C_{te} = \$12.5$ per tool edge.

For each order number, 10 independent data instances using different random seeds are generated with the above data generation procedure. Therefore, in total 100 instances are used. All the reported results are the average values obtained by the 10 independent instances for each order number.

3.6.2 Results for Problem IDM-C

Table 3.1 presents the results for solving problem IDM-C by the ECM in [1], solving models M1 and M1-LP, and the DG Algorithm 3.1. The columns show the objectives (minimum cost in \$ by each method) and the solution time in second. In the column “CRR”, we calculate the cost reduction ratio (CRR) as

$$CRR = \frac{Obj_{ECM} - Obj}{Obj_{ECM}} \times 100\% \quad (64)$$

Where Obj_{ECM} and Obj , respectively, refer to the objective value from ECM and other methods under comparison.

As shown in Table 3.1, solving M1 and M1-LP by CPLEX, and the DG algorithm all find the optimal solution for all the 100 instances within 0.1 second, shown as the bold numbers. Remarkably, the DG algorithm preforms as the fastest among the three, within 0.005 s for all the 100 instances. In comparison with the ECM, formulations M1 and M1-LP and DG algorithm show that, by selecting different cutting speeds, the overall production cost can be reduced by 3% – 5% on average, as shown by the highlighted columns in gray in Table 3.1. Figure 3.4 shows results of CRR in Table 3.1. Note the CRR for formulations M1, M1-LP and DG are the same and thus are overlapped.

Table 3.1. Numerical results of solving problem IDM-C by the expected cost method (ECM) [1], solving models M1 and M1-LP, and the exact algorithm DG.

Order #	ECM		M1			M1-LP			DG		
	Obj: Cost	Time (s)	Obj: Cost	CRR (%)	Time (s)	Obj: Cost	CRR (%)	Time (s)	Obj: Cost	CRR (%)	Time (s)
2	2082.55	2.81	1977.23	5.06	0.01	1977.23	5.06	0.01	1977.23	5.06	0.000
4	3301.91	5.58	3174.74	3.85	0.01	3174.74	3.85	0.02	3174.74	3.85	0.000
6	5418.85	8.28	5184.44	4.33	0.01	5184.44	4.33	0.03	5184.44	4.33	0.000
8	6670.49	11.01	6406.96	3.95	0.01	6406.96	3.95	0.03	6406.96	3.95	0.003
10	9688.88	13.87	9314.33	3.87	0.01	9314.33	3.87	0.03	9314.33	3.87	0.003
12	11872.06	16.66	11363.36	4.28	0.01	11363.36	4.28	0.05	11363.36	4.28	0.003
14	13597.67	19.27	13061.32	3.94	0.02	13061.32	3.94	0.05	13061.32	3.94	0.003
16	15417.88	21.77	14768.81	4.21	0.02	14768.81	4.21	0.06	14768.81	4.21	0.005
18	16849.63	24.57	16127.64	4.28	0.02	16127.64	4.28	0.06	16127.64	4.28	0.003
20	20654.23	27.29	19883.29	3.73	0.02	19883.29	3.73	0.08	19883.29	3.73	0.003

* All values in all columns are the average results of 10 independent runs for each order number.

* All the 10 independent instances for all the order numbers are solved to optimality with Gap 0.00% by M1, M1-LP and DG (shown by bold numbers).

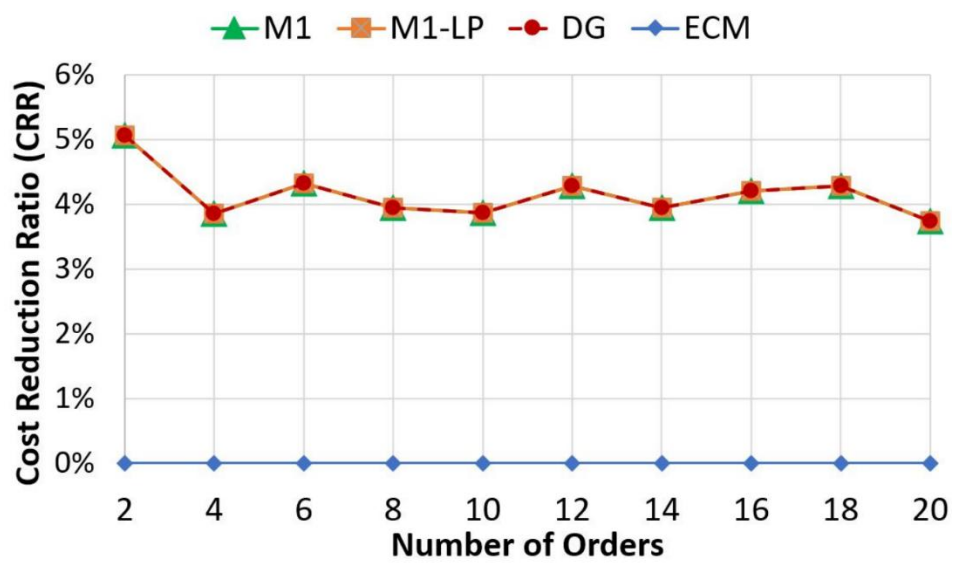


Figure 3.4. The comparison of cost reduction ratio (CRR) for problem IDM-C. M1, M1-LP and DG are overlapped as they all obtain the optimal solutions.

Essentially, the cause of 3% – 5% CRR comes from the rationale of our integration idea. That is, the ECM from machining dynamics view considers only one part to choose the best speed. Thus, when multiple parts of an order are considered from the discrete event dynamics view, this speed might not be optimal since it might lead to larger waste at the rear of the tool life, then larger tool edge quantities and higher cost. Instead, our methods choose the best speed with a balance of the number of tool edges and the tool life, thus could avoid such waste of tool life and save the cost (see illustrative example in Figure 3.3).

3.6.3 Results for Problem IDM-CM

Table 3.2 shows the results for solving problem IDM-CM. In the column “Makespan Limit: c ”, we show the average makespan limit within which all parts must be finished. Formulation M2 is solved to optimality (with the gap 0.00) for all the 10 instances under each order number within 0.11 second, shown as the bold numbers. The LP relaxation results show M2 is tight, which can explain for its good computational efficiency.

Notably, due to the makespan limit, the minimum cost obtained in solving problem IDM-C in Table 3.1 cannot be obtained in solving problem IDM-CM. Nevertheless, all these optimal solutions satisfy the makespan limit. While the ECM does not support to set makespan limit, the results here show that our integration provides the capability to set many practical production requirements. Furthermore, for all the 10 order numbers, both the cost and makespan of ECM solutions are larger than those of the M2 solutions. In summary, by using our method, it is practical for the machine shop to achieve solutions that lead to both smaller cost and shorter makespan.

Table 3.2. Numerical results of solving problem IDM-CM by expected cost method (ECM) [1], and solving model M2.

Order #	Makespan Limit: c	ECM			M2					
		Obj: Cost	Time (s)	Makespan	Obj: Cost	Num	Gap (%)	LP Relaxation	Time (s)	Makespan
2	633	2082.55	2.81	763.15	2052.42	10	0.00	2034.58	0.00	624.96
4	1027	3301.91	5.58	1209.71	3245.44	10	0.00	3233.46	0.05	1021.47
6	1678	5418.85	8.28	2000.68	5281.53	10	0.00	5271.44	0.08	1672.02
8	2070	6670.49	11.01	2449.62	6532.52	10	0.00	6526.56	0.08	2067.51
10	3007	9688.88	13.87	3580.07	9512.87	10	0.00	9502.75	0.11	3003.93
12	3669	11872.06	16.66	4362.28	11571.73	10	0.00	11566.20	0.08	3667.11
14	4198	13597.67	19.27	5013.83	13364.26	10	0.00	13354.87	0.05	4195.26
16	4775	15417.88	21.77	5687.06	15047.90	10	0.00	15042.45	0.08	4772.70
18	5202	16849.63	24.57	6221.69	16426.20	10	0.00	16421.71	0.05	5199.98
20	6413	20654.23	27.29	7656.49	20286.36	10	0.00	20281.15	0.08	6411.31

* All values in all columns except for "Num" are the average results of 10 independent runs for each order number.

* Column "Num" indicates the number of instances solved to optimality among the 10 instances.

Table 3.3. Numerical results of problem IDM-CD by solving models M3-1, M3-2-L and M3-3-L.

Order #	M3-1				M3-2-L				M3-3-L			
	Obj: Cost	Num	Gap (%)	Time (s)	Obj: Cost	Num	Gap (%)	Time (s)	Obj: Cost	Num	Gap (%)	Time (s)
2	1920.68	10	0.00	0.02	1920.68	10	0.00	0.02	1920.68	10	0.00	0.03
4	4463.55	10	0.00	0.08	4463.55	10	0.00	0.05	4463.55	10	0.00	0.07
6	5412.14	10	0.00	0.25	5412.14	10	0.00	0.07	5412.14	10	0.00	0.14
8	7293.43	10	0.00	0.27	7293.43	10	0.00	0.10	7293.43	10	0.00	1.83
10	11129.40	10	0.00	1.02	11129.40	10	0.00	0.39	11129.40	10	0.01	25.43
12	12700.74	10	0.01	14.17	12700.74	10	0.00	2.66	12700.74	5	3.14	508.27
14	12615.67	8	0.44	163.11	12615.67	10	0.01	10.80	12615.67	1	6.05	601.27
16	13232.52	4	2.52	485.71	13232.52	10	0.01	45.09	13232.52	1	6.06	602.53
18	18001.38	0	6.47	600.30	18001.38	10	0.01	184.39	18001.38	0	8.06	602.33
20	21394.91	0	8.76	600.60	21394.91	8	0.76	380.35	21394.91	0	8.76	602.19

* All values in all columns except for "Num" are the average results of 10 independent runs for each order number.

* Column "Num" indicates the number of instances solved to optimality among the 10 instances.

3.6.4 Results for Problem IDM-CD

In general, the problem with due date consideration (IDM-CD) is harder to solve. The proposed three formulations (M3-1, M3-2L and M3-3-L) can perform differently in terms of their solution quality (measured by Gap, the smaller the better; the Number of instances solved to optimality, the larger the better) and time efficiency (measured by Time, the smaller the better).

Table 3.3 presents such numerical results. Obviously, the solution quality of M3-2L is the best among the three. This can be seen from the number of instances solved to optimality. Among all the 100 instances, M3-2L solves 98 to optimality except for 2 instances at $n = 20$ orders, while that of M3-1 and M3-3-L are 72 and 66, respectively. The same conclusions can be drawn from the gap and the solution time. For example, at $n = 18$, M3-2L takes 184.39 seconds to solve all 10 instances to optimality, while M3-1 takes 600.30 seconds yet still has gap 6.47%, and M3-3-L takes 602.33 seconds with gap 8.06%. Note that time above 600 seconds means the solving procedure is terminated due to time limit of CPLEX. It is interesting to see although with different gaps for the three models, the objectives are the same. In our experience, it can often be the case that CPLEX has found the optimal solution in 5 minutes, though it takes much longer time to certify its optimality. Thus, formulations with fast increasing lower bounds can certify the optimality in shorter time.

In summary, these models consider the cost, makespan and due date, and could have large potentials to be used for industrial practice. Among them, model M3-2L is the best one that gives the best solution within the smallest time. This validates our idea of

integrating discrete event dynamics with machining dynamics for the operational excellence of machine tool.

3.7 Conclusion, Discussion and Future Work

We have identified and studied a significant problem of the integration of discrete-event dynamics and machining dynamics for the operational excellence of machine tool for the first time. The machining stability, various cutting speeds and tool life were incorporated into the production of a series of parts from multiple orders in the queue of machine tool. Based on a state-of-the-art logistic regression model for tool life prediction, a new cost function was developed for orders with cutting speed choices as the variables. Then the cost was minimized via choosing different stable cutting speeds.

We have demonstrated a set of models and algorithms by considering various practical production scenarios, including cost, makespan and due date. When considering the integrated optimization of cost and makespan, we developed integer optimization models and exact algorithm. Then, we considered the integrated optimization of cost and due date. Three categories of mixed-integer optimization models were presented based on the choices of decision variables. The linearization technique was applied to obtain the models. The results showed while satisfying the makespan and due date requirements, our methods achieved on average 3% – 5% reduction of the cost, which has the potential for cost saving in practice.

Future work includes the deeper integration from both the discrete-event dynamics and the machining dynamics. Besides cutting speed, other machining parameters can be as decision variables, including axial depth of cut and feed rate. Thus, we may have nonlinear

and nonconvex constraints from the stability lobe diagram. Also, the stochastic tool life prediction can be considered. In the meantime, the complexities of the problems for makespan and due date constraints are one future direction. The practical extensions of the problems are to consider different candidate speeds for orders, various part types of an order, and multiple machine tools in the machine shop.

Nomenclature	
$\mathcal{J}$	$\mathcal{J} = \{1, \dots, n\}$ is the set of n orders, indexed by j, k or l
$\mathcal{I}$	$\mathcal{I} = \{1, \dots, m\}$ is the index set of m candidate speeds, indexed by i
l_j	the quantity of parts for order j
v_j	volume to be removed for each part in order j , mm^3
d_j	the due date for order j , min
a	radial depth of cut, mm
b	axial depth of cut, mm
f_t	feed per tooth, mm
N_t	number of teeth
Ω	spindle speed, rpm
V	cutting speed, mm/min
D	diameter of the tool edge, mm
t	machining time for one part, min
T	tool life, min
MRR	material removal rate, mm^3/min
r_m	machine tool operating cost, \$/min
t_{ch}	time required for changing a tool edge, min
C_{te}	cost per tool edge, \$
C_p	total cost per part, \$
C_0	total cost per order, \$
Ω_{ij}	i -th spindle speed for processing order j , rpm
T_{ij}	tool life when using speed i to process order j , min
p_{ij}	total machining time of order j with speed i , min
q_{ij}	number of tool edges needed for order j by speed i
h_{ij}	total processing time of order j with speed i , min
$\mathbf{\Omega}$	spindle speed assignment matrix, $\mathbf{\Omega} = \{\Omega_{ij}\}_{m \times n}$

T	tool life time matrix, $\mathbf{T} = \{T_{ij}\}_{m \times n}$
P	total machining time matrix, $\mathbf{P} = \{p_{ij}\}_{m \times n}$
Q	matrix for number of tool edges, $\mathbf{Q} = \{q_{ij}\}_{m \times n}$
H	total processing time matrix, $\mathbf{H} = \{h_{ij}\}_{m \times n}$
x_{ij}	binary variable, $x_{ij} = 1$ if speed i is chosen for processing order j ; otherwise $x_{ij} = 0$
$f_{cost}(x)$	cost of all orders as a function of x , \$
c	a given constant of the upper bound for the makespan, min
M	M is a sufficiently large number
y_{jk}	binary variable, $y_{jk} = 1$ if order j is processed before order k ; otherwise $y_{jk} = 0$
C_j	completion time of order j , min
z_{jk}	binary variable, $z_{jk} = 1$ if order j is assigned on position k ; otherwise $z_{jk} = 0$
C_k	completion time of the order at position k , min

CHAPTER 4

A CUTTING MECHANICS-BASED MACHINE LEARNING MODELING

METHOD DISCOVER GOVERNING EQUATIONS OF MACHINING

This paper presents a cutting mechanics-based machine learning method for automated discovery of governing equations in machining dynamics. By integrating known cutting mechanics with data-driven approaches, the proposed method establishes a general model structure using differential-algebraic equations. Then, a nonlinear learning function space and discrete optimization algorithm enable the identification of unknown dynamics. Time domain milling simulations experiments validate that the proposed method can accurately discover governing equations of motion for machining dynamics systems with considering of nonlinear damping and edge forces, from noisy data. Results also indicate the potential of proposed method to be used for advancing machining modeling in practice with the development of effective metrology systems.

4.1 Introduction

Machining dynamics studies the physical process of metal cutting due to the interaction of computer numerical control (CNC) machine, tool, and workpiece. In the machining process, dynamic tool-workpiece engagement induces dynamic cutting forces, that can further excite the structural dynamics of machine tool to vibrate at the tool tip point, and eventually form the material surface of workpiece [69]. As such, modeling machining dynamics generally needs to integrate two parts: the structural dynamics of machine tool and cutting forces of dynamic cutting process.

Significant modeling efforts on machining dynamics have been made in the past decades. A consensus in the research community is that machining dynamics can be mathematically described by second order, coupled, time delayed differential equations (DDEs) of motion. The first model of machining dynamics was developed in [72] as a single degree of freedom DDE for turning dynamics. It described that chatter, or self-excited vibration, is primarily caused by regenerative chip thickness due to time delayed effect. The pioneer research findings of stability laws for turning then were concurrently and independently presented in [73, 74]. This single degree of freedom DDE for orthogonal cutting, however, cannot describe multiple degrees of freedom cutting like milling. For milling, the first model was then developed in [75] where time periodicity and time delay were formulated to characterize the milling dynamics. In addition to the time delayed effect, it was observed that the stability increases at low spindle speeds. This is attributed to the process damping effect caused by the penetration of cutting edge into wavy cut surface [76]. The process damping for turning and milling was studied in [52, 53, 77] by adding a process damping force, that is dependent on the surface normal velocity, chip width, cutting speed, and empirical coefficients. Other force models like edge force and runout were also studied based on different cutting mechanics [78]. What's more, the receptance coupling substructure analysis approach, analytically joining substructure receptance of tool, holder, and spindle-machine, was proposed in [79, 80] to advance machining modeling for better structural dynamics prediction. These models are experimentally verified and widely used in academia and industry.

However, due to the complexity of stability solutions of DDEs, most of the existing physical models are developed in linear form. Nonlinear models are seldom explored. One representative nonlinear method is time domain simulation that can well capture the nonlinearities caused by tool jumping out of the cut, runout, and varying tool geometry [81-83]. A few other studies of nonlinearities on structural dynamics and cutting forces are summarized in [84]. On the other hand, for data-driven methods, machine learning (ML) models (generally nonlinear) have been applied to machining dynamics. One research focus is developing mechanics-based ML methods to improve the accuracy of coefficient estimation for existing mechanistic force models [85-87]. Another research focus is developing ML methods to improve the accuracy of stability maps, including neural networks [88], support vector machine [89], k-nearest neighbors [90], transfer learning [91], and physics-based Bayesian learning [55, 92-94]. However, those ML models are mainly designed to generate more accurate stability solutions based on existing DDEs and experimental data, rather than discovering governing equations of machining dynamics. These limitations of current physical and ML models motivate this study to design an integrated physics-based and data-driven modeling method, that can leverage existing physics in cutting mechanics to explore unknown physics in data and therefore discover nonlinear governing equations of machining dynamics.

This paper presents the first attempt to develop a cutting mechanics-based machine learning (CMML) modeling method to discover governing equations for machining dynamics. Based on existing physics in cutting mechanics, CMML first establishes a general modeling structure through representing the traditional DDEs of machining

dynamics by a set of unknown differential algebraic equations. Then, to discover these unknown differential algebraic equations, effective nonlinear learning function space design and discrete optimization-based learning algorithm are developed as the ML component of CMML. Experimentally verified time domain simulations of milling with considerations of nonlinear process damping and edge force are used to validate the proposed CMML modeling method. Numerical results demonstrate CMML can successfully discover the exact milling dynamics models from noisy data. This indicates the proposed CMML modeling method has the potential for advancing machining modeling in practice with the development of effective metrology systems.

This paper is organized as follows. Section 2 presents the proposed CMML modeling method. The simulation experiments, numerical results and analyses are given in Section 3. Section 4 concludes this paper.

4.2 Proposed Cutting Mechanics-based Machine Learning Modeling Method

The proposed CMML modeling method is technically presented based on two sequential components: Establish a general cutting mechanics-based modeling structure by first principles (Section 2.1); and discover governing equations of machining dynamics from data by ML algorithm design (Section 2.2).

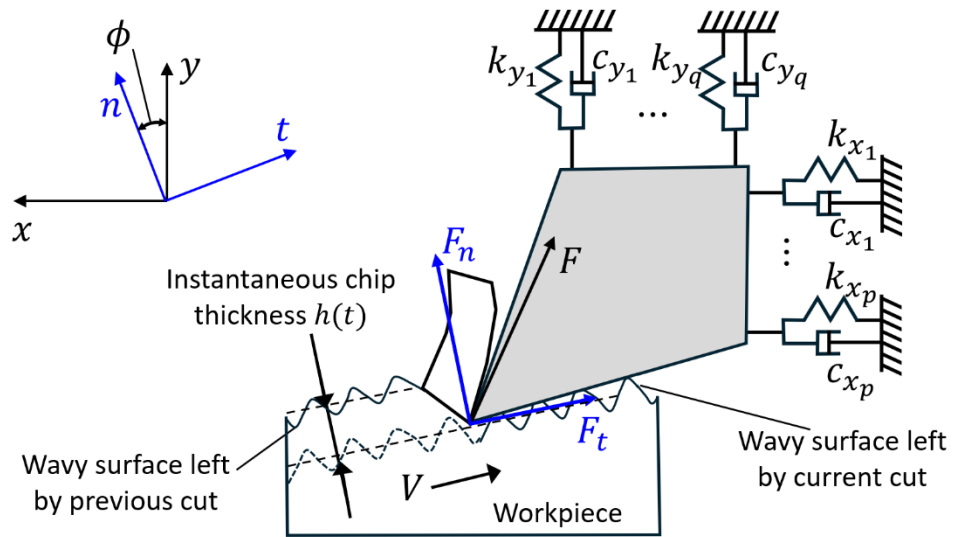


Figure 4.1. Schematic illustration of two fundamental principles of cutting mechanics: 1) time delayed effect, and 2) dynamic cutting force generation and projection, The resultant dynamic cutting forces excite structural dynamics to vibrate on machine tool's coordinate.

4.2.1 Establish Cutting Mechanics-based Modeling Structure by First Principle

4.2.1.1 Time Delayed Effect and Dynamics Cutting Force

Metal cutting consists of various processes, including the most common turning, milling, and drilling, and followed by special operations such as boring, broaching, and shaping. Although the tool geometry and tool kinematics of these cutting operations may differ from each other, they share the same principles of cutting mechanics [95]. Two fundamental principles of cutting mechanics are introduced below: 1) time-delayed effect, and 2) dynamic cutting force generation and projection. They lay a solid foundation to establish a general modeling structure for the proposed CMML modeling method.

Without loss of generality, a simple case of two-dimensional orthogonal cutting is used to explain these two fundamental principles, as shown in Figure 4.1. The tool edge is orthogonal to the cutting speed V . A metal chip with an instantaneous chip thickness $h(t)$ is sheared away from the workpiece. Let the chip thickness direction be normal or radial direction, and the direction of cutting speed V be tangential direction.

Time delayed effect determines the instantaneous chip thickness $h(t)$, as shown in Eq. (1).

$$h(t) = h + n(t - \tau) - n(t). \quad (1)$$

h is the static chip thickness attributed to rigid body motion of the cutting tool, and $n(t - \tau) - n(t)$ is the dynamic chip thickness caused by time-delayed effect. $n(t - \tau)$ and $n(t)$ are the vibrations of tool along chip thickness direction at previous and current cuts. The time delay τ is dependent on the spindle speed and can be different for different cutting processes. For example, $\tau = 1/\Omega$ for turning where Ω (rps) is the spindle speed. The time

delay can be explained as follows. As soon as the cutting tool encounters the workpiece, it starts to vibrate because of the mass and stiffness. This leaves a wavy surface on the workpiece, as shown in Figure 4.1. The vibrations contribute to generating variable cutting force, and the cutting force will again generate vibrations of the cutting tool. Due to periodic movement of the spindle, when the next vibrating cutting edge encounters the wavy surface left by the previous cut, the dynamic chip thickness presents.

The resultant dynamic cutting forces are exerted on the tangential and normal directions, denoted by F_t and F_n . In general, the cutting forces are functions that can be described as in Eq. (2):

$$F_t(t) = f_t(n(t - \tau) - n(t), \dot{n}(t), \mathbf{u}), \quad (2a)$$

$$F_n(t) = f_n(n(t - \tau) - n(t), \dot{n}(t), \mathbf{u}). \quad (2b)$$

$\dot{n}$ denotes the velocity of tool vibration on the normal direction, and vector $\mathbf{u} \in \mathbb{R}^{1 \times r}$ consists of r cutting process parameters (e.g., spindle speed Ω , cutting depth b , and feed rate f_t). The cutting forces F_n and F_t on the cutter's coordinate are projected onto the machine tool's coordinate as F_x and F_y to excite structural dynamics to vibrate in both directions. The projection relationship in Figure 4.1 can be resolved as

$$F_x = -F_t \cos(\phi) + F_n \sin(\phi), \quad (3a)$$

$$F_y = F_t \sin(\phi) + F_n \cos(\phi). \quad (3b)$$

ϕ is the angle between the chip thickness and machine tool's y (normal) directions. In general, the projection angle ϕ can be a constant or time dependent variable. For example, in turning, this angle is a constant, while in milling this angle is a time varying rotational angle of cutting edge. The projection relationship in Eq. (3) is critical for studying

structural dynamics as it shifts the dynamic cutting forces on the varying coordinate of tool to the fixed Cartesian coordinate of machine.

4.2.1.2 Cutting Mechanics-based Modeling Structure Design

The dynamic cutting forces on x and y directions of machine tools excite the structural dynamics on both directions to vibrate at the tooltip during machining processes. Thus, to model machining dynamics is to understand the sources of force generation and the resultant machine tool's structural dynamics excited by these forces. In the following, the structural dynamics of machine tool is assumed to have p and q modes on x and y directions, as shown in Figure 4.1.

In prior studies, the governing equations of motion for machining dynamics are commonly modeled by second order, coupled DDEs as shown in Eq. (4):

$$\mathbf{m}_x \ddot{\mathbf{x}}^T + \mathbf{c}_x \dot{\mathbf{x}}^T + \mathbf{k}_x \mathbf{x}^T = F_x(t, \mathbf{x}(t), \mathbf{x}(t - \tau), \dot{\mathbf{x}}), \quad (4a)$$

$$\mathbf{m}_y \ddot{\mathbf{y}}^T + \mathbf{c}_y \dot{\mathbf{y}}^T + \mathbf{k}_y \mathbf{y}^T = F_y(t, \mathbf{y}(t), \mathbf{y}(t - \tau), \dot{\mathbf{y}}), \quad (4b)$$

where $\mathbf{x}$, $\dot{\mathbf{x}}$, $\ddot{\mathbf{x}}$, $\mathbf{m}_x$, $\mathbf{c}_x$, and $\mathbf{k}_x$ are the displacement, velocity, acceleration, modal mass, viscous damping, and stiffness on x direction and all vectors in $\mathbb{R}^{1 \times p}$; and $\mathbf{y}$, $\dot{\mathbf{y}}$, $\ddot{\mathbf{y}}$, $\mathbf{m}_y$, $\mathbf{c}_y$, and $\mathbf{k}_y$ are the displacement, velocity, acceleration, modal mass, viscous damping, and stiffness on y direction and all vectors are in $\mathbb{R}^{1 \times q}$. The left-hand-side of Eq. (4) represents the structural dynamics of machine tool with respect to displacement, velocity, and acceleration. The right-hand-side of Eq. (4) is the cutting force model considering the time delayed term and projection. Common assumptions are imposed like orthogonality of x and y directions, linear model structure, and viscous damping [96]. These models are well

studied and experimentally verified through decades of research efforts in the machining community.

Based on the time delayed effect and dynamic cutting force, a general cutting mechanics-based modeling structure is proposed in state-space form through representing the governing DDEs in Eq. (4) by a set of differential algebraic equations (DAEs) in Eq. (5) as follows.

First, the governing differential equations for displacement and velocity that describe the structural dynamics of machine tools can be formulated as the unknown equations in Eqs. (5a) - (5d):

$$\dot{\mathbf{x}} = \mathbf{f}_1(\mathbf{s}(t), \mathbf{u}, \mathbf{F}(t)), \quad (5a)$$

$$\dot{\mathbf{v}}_x = \mathbf{f}_2(\mathbf{s}(t), \mathbf{u}, \mathbf{F}(t)), \quad (5b)$$

$$\dot{\mathbf{y}} = \mathbf{f}_3(\mathbf{s}(t), \mathbf{u}, \mathbf{F}(t)), \quad (5c)$$

$$\dot{\mathbf{v}}_y = \mathbf{f}_4(\mathbf{s}(t), \mathbf{u}, \mathbf{F}(t)). \quad (5d)$$

Notations $\mathbf{v}_x$ and $\mathbf{v}_y$ are introduced by letting $\mathbf{v}_x = \dot{\mathbf{x}}$ and $\mathbf{v}_y = \dot{\mathbf{y}}$. Note that $\mathbf{s}(t) := [\mathbf{x}(t) \ \mathbf{v}_x(t) \ \mathbf{y}(t) \ \mathbf{v}_y(t)]$ in $\mathbb{R}^{1 \times (2p+2q)}$ consists of all the state variables and $\mathbf{F}(t) := [F_x(t) \ F_y(t)]$ in $\mathbb{R}^{1 \times 2}$ is the vector of cutting forces.

Then, by integrating the projection relationship from F_n and F_t to F_x and F_y in Eq. (3), the governing equations for cutting forces can be modeled as the following algebraic equations in Eqs. (5e) - (5h):

$$F_n = g_1(\mathbf{x}(t), \mathbf{x}(t - \tau), \mathbf{y}(t), \mathbf{y}(t - \tau), \mathbf{u}), \quad (5e)$$

$$F_t = g_2(\mathbf{x}(t), \mathbf{x}(t - \tau), \mathbf{y}(t), \mathbf{y}(t - \tau), \mathbf{u}), \quad (5f)$$

$$F_x = -F_t(t) \cos(\phi) + F_n(t) \sin(\phi), \quad (5g)$$

$$F_y = F_t(t) \sin(\phi) + F_n(t) \cos(\phi). \quad (5h)$$

It should be noted that Eqs. (5e) and (5f) are known and once they are determined, the analytical forms of F_x and F_y in Eqs. (5g) and (5h) are determined. The general coupled dynamics on x and y directions are considered for both differential and algebraic equations. The coupled dynamics can be decoupled by assuming that zero cross talk is between the two orthogonal directions of machine tools.

Therefore, a general cutting mechanics-based modeling structure is obtained through representing the traditional governing equations of DDEs by a set of unknown DAEs, i.e., Eq. (5) including Eqs. (5a) – (5h). This general modeling structure for machining dynamics enables the possibility that utilizing existing physics in cutting mechanics to explore the unknown physics in data by developing effective machine learning algorithms to discover those unknown DAEs.

4.2.2 Discover DAEs from Data by Machine Learning

The discrete-optimization based machine learning method for discovering differential equations of nonlinear dynamics in [97] is extended as the ML component of the proposed CMML modeling method, to discover all the unknown DAEs.

4.2.2.1 Cutting Mechanics-based Nonlinear Learning Function Space Design

Throughout this paper, let $[P] = \{1, 2, \dots, P\}$ for any given positive integer P . In addition, for given process parameters $\mathbf{u}$, the state variables $\mathbf{s}(t)$, time derivatives $\dot{\mathbf{s}}(t)$, and cutting forces $\mathbf{F}(t)$ are assumed to be measurable in time series form. The resulting data matrices are denoted by $\mathbf{S} \in \mathbb{R}^{N \times (2p+2q)}$, $\dot{\mathbf{S}} \in \mathbb{R}^{N \times (2p+2q)}$, and $\mathbf{F} \in \mathbb{R}^{N \times 2}$, respectively. This assumption is reasonable. For example, the dynamometers, capacitance

probes, laser vibrometers, and accelerometers were used to concurrently measure cutting force, displacement, velocity and acceleration of cutting tools [98]. For ease of presentation, the process parameter vector $\mathbf{u}$ is rewritten in a matrix form $\mathbf{U} = \mathbf{I}_{N \times r} \text{diag}\{\mathbf{u}\} \in \mathbb{R}^{N \times r}$ with the same row number as $\mathbf{S}$. Herein, $\mathbf{I}_{N \times r} \in \mathbb{R}^{N \times r}$ is a matrix with all entries set to 1, and $\text{diag}\{\mathbf{u}\} \in \mathbb{R}^{r \times r}$ is a diagonal matrix with diagonal entries set to $\mathbf{u}$. In this way, $\mathbf{u}$ is constructed as paired data of $\mathbf{s}(t)$, $\dot{\mathbf{s}}(t)$, and $\mathbf{F}(t)$. The data in $\mathbf{S}$, $\dot{\mathbf{S}}$, and $\mathbf{F}$ can be injected with noises.

Note that the developed machine learning algorithm can be applied to all unknown DAEs in Eq. (5). Therefore, without loss of generality, the discovery of governing equation of displacement x_j of the j -th mode is used throughout, to illustrate the developed machine learning algorithm. The learning function space design for unknown DAEs is to create an augmented function space by expanding the function space that current DDEs-based governing equations live. Existing physical knowledge in cutting mechanics can be incorporated for the design of this expanded learning function space.

The governing equation $\dot{x}_j = f_j(\mathbf{s}, \mathbf{u}, \mathbf{F})$ is to be discovered using data matrices $\mathbf{S}$, $\mathbf{U}$, $\mathbf{F}$, the process parameters $\mathbf{u}$, and the j -th column $\dot{\mathbf{x}}_j$ of matrix $\dot{\mathbf{S}}$ for method illustration. The subscript j is further omitted for notational simplicity. First, a cutting mechanics-based candidate set $\boldsymbol{\theta}(\mathbf{s}, \mathbf{u}, \mathbf{F})$ consisting of a total of P candidate physical terms for $f(\mathbf{s}, \mathbf{u}, \mathbf{F})$ should be constructed as follows:

$$\boldsymbol{\theta}(\mathbf{s}, \mathbf{u}, \mathbf{F}) := [\theta_1(\mathbf{s}, \mathbf{u}, \mathbf{F}) \ \theta_2(\mathbf{s}, \mathbf{u}, \mathbf{F}) \ \cdots \ \theta_P(\mathbf{s}, \mathbf{u}, \mathbf{F})], \quad (6)$$

where $\theta_1(\mathbf{s}, \mathbf{u}, \mathbf{F}), \dots, \theta_P(\mathbf{s}, \mathbf{u}, \mathbf{F})$ represent the candidate physical terms that can be contained in the unknown function $f(\mathbf{s}, \mathbf{u}, \mathbf{F})$. Existing physics in cutting mechanics needs

to be incorporated into the design of candidate set. For example, in milling dynamics, besides the time delayed term and cutting force projection, the periodic immersion angle ϕ caused by the rotary milling cutter should also be taken into consideration. The corresponding cutting mechanics-based candidate set can be constructed as

$$\boldsymbol{\theta}(\mathbf{s}, \mathbf{u}, \mathbf{F}) = [1 \ \mathbf{s} \ \mathbf{u} \ \mathbf{F} \ \mathbf{s}^2 \ \mathbf{s} \otimes \mathbf{u} \ n(t - \tau) - n(t) \sin \phi \ \dots]. \quad (7)$$

The operator $\otimes$ denotes the element-wise products of vectors. The basic physical terms in $\boldsymbol{\theta}(\mathbf{s}, \mathbf{u}, \mathbf{F})$ can include constants, polynomial terms of $\mathbf{s}$, $\mathbf{u}$, and $\mathbf{F}$, etc. In addition, the time delayed term and trigonometric terms of ϕ are also included based on cutting force projection. By constructing the candidate set $\boldsymbol{\theta}(\mathbf{s}, \mathbf{u}, \mathbf{F})$, nonlinearities of machining dynamics can be introduced through the nonlinear combinations of state variables, process parameters, and cutting forces.

Then, the constructed candidate set $\boldsymbol{\theta}(\mathbf{s}, \mathbf{u}, \mathbf{F})$ is used to expand the function space for the unknown functions in Eq. (5). In this paper, two important assumptions are made. The first assumption is that all functions $f(\mathbf{s}, \mathbf{u}, \mathbf{F})$ in Eq. (5) live in the affine functional space expanded by $\boldsymbol{\theta}(\mathbf{s}, \mathbf{u}, \mathbf{F})$. That is, a cutting mechanics-based learning function space can be established as

$$f(\mathbf{s}, \mathbf{u}, \mathbf{F}) = \boldsymbol{\theta}(\mathbf{s}, \mathbf{u}, \mathbf{F}) \cdot \boldsymbol{\xi}, \quad (8)$$

where $\boldsymbol{\xi} = [\xi_1 \ \xi_2 \ \dots \ \xi_P]^T \in \mathbb{R}^{P \times 1}$ represents the vector of the coefficients. The second assumption is that physical terms in the right-hand side of Eq. (8) are sparse in predefined function space, namely, the number of active terms (terms with nonzero coefficient in $\boldsymbol{\xi}$) in $f(\mathbf{s}, \mathbf{u}, \mathbf{F})$ is much smaller than that in the candidate set $\boldsymbol{\theta}(\mathbf{s}, \mathbf{u}, \mathbf{F})$.

4.2.2.2 Discover DAEs by Discrete Optimization-based Learning Algorithm

A discrete optimization-based learning algorithm is presented to estimate the coefficients ξ in Eq. (8) for data-driven discovery of unknown DAEs. A physical term in $\theta(\mathbf{s}, \mathbf{u}, \mathbf{F})$ can be either present or not in $f(\mathbf{s}, \mathbf{u}, \mathbf{F})$. A new binary variable vector γ , where $\gamma = [\gamma_1 \gamma_2 \cdots \gamma_P]^T \in \mathbb{B}^{P \times 1}$, is introduced to indicate the presence of each physical term θ_p in $\theta(\mathbf{s}, \mathbf{u}, \mathbf{F})$. Each binary variable γ_p in the Boolean domain $\mathbb{B}$ is

$$\gamma_p = \begin{cases} 1, & \text{if } f(\mathbf{s}, \mathbf{u}, \mathbf{F}) \text{ includes } \theta_p, \\ 0, & \text{otherwise} \end{cases}, \quad \forall p \in [P]. \quad (9)$$

With the indicator vector γ , a sparse linear regression model can be established as in Eq. (10) to minimize the sum of squared errors between the measurements $\dot{\mathbf{x}}$ and model predictions of $f(\mathbf{s}, \mathbf{u}, \mathbf{F})$ with regularizations.

$$\min_{\gamma, \xi} \|\dot{\mathbf{x}} - \Theta(\mathbf{S}, \mathbf{U}, \mathbf{F}) \cdot (\gamma \circ \xi)\|_2^2 + \lambda_0 \|\xi\|_0 + \lambda_2 \|\xi\|_2^2. \quad (10)$$

$\Theta(\mathbf{S}, \mathbf{U}, \mathbf{F}) \in \mathbb{R}^{N \times P}$ denotes the augmented matrix consisting of values of candidate terms that are calculated by evaluating the candidate set $\theta(\mathbf{s}, \mathbf{u}, \mathbf{F})$ on datasets $\mathbf{S}$, $\mathbf{U}$, and $\mathbf{F}$. By introducing the element-wise product $\gamma \circ \xi$, Eq. (10) enables to control the presence of active physical terms in $\theta(\mathbf{s}, \mathbf{u}, \mathbf{F})$ for the purpose of discovering parsimonious function $f(\mathbf{s}, \mathbf{u}, \mathbf{F})$. Two regularization terms are introduced. The number of active physical terms is implicitly controlled by the l_0 -norm in the penalty $\lambda_0 \|\xi\|_0$, which constrains the number of nonzero entries of ξ . Another l_2 -norm of ξ , $\lambda_2 \|\xi\|_2^2$, is introduced to reduce the effects of noisy process measurements. The penalty weights λ_2 and λ_0 are hyperparameters that control the strength of regularizations.

The sparse regression problem in Eq. (10) can be equivalently reformulated as a discrete optimization formulation as follows:

$$\min_{\boldsymbol{\gamma}, \boldsymbol{\xi}} \|\dot{\mathbf{x}} - \boldsymbol{\Theta}(\mathbf{S}, \mathbf{U}, \mathbf{F}) \cdot \boldsymbol{\xi}\|_2^2 + \lambda_2 \|\boldsymbol{\xi}\|_2^2 \quad (11a)$$

$$s. t. \quad -M\boldsymbol{\gamma} \leq \boldsymbol{\xi} \leq M\boldsymbol{\gamma}, \quad (11b)$$

$$\langle \boldsymbol{\gamma}, \mathbf{e} \rangle = k, \quad (11c)$$

$$\boldsymbol{\gamma} \in \mathbb{B}^{P \times 1}, \boldsymbol{\xi} \in \mathbb{R}^{P \times 1}. \quad (11d)$$

The ℓ_0 -norm and the relationship between $\boldsymbol{\xi}$ and $\boldsymbol{\gamma}$ in the objective function are converted to the constraints as shown in Eqs. (11b) - (11c), and the inequalities in Eq. (11b) indicate the element-wise relationship. When $\gamma_p = 0$, the p -th term in $\boldsymbol{\theta}(\mathbf{s}, \mathbf{u}, \mathbf{F})$ is not in $f(\mathbf{s}, \mathbf{u}, \mathbf{F})$. As such, both sides of Eq. (11b) are equal to zero, leading to $\xi_p = 0$. When $\gamma_p = 1$, ξ_p is bounded by a range $[-M, M]$, where M is a constant number. $[-M, M]$ is the predefined lower and upper bound interval for coefficient $\boldsymbol{\xi}$. The number of active physical terms is controlled by the inner product of $\boldsymbol{\gamma}$ and $\mathbf{e}$ in Eq. (11c), where $\mathbf{e} \in \mathbb{R}^{P \times 1}$ is a vector with all entries set to 1.

To solve the discrete optimization problem in Eq. (11), a two-stage optimization algorithm presented in [97] is applied to determine $\boldsymbol{\gamma}$ and $\boldsymbol{\xi}$ separately. First, the column-wise normalization of data matrices $\boldsymbol{\Theta}(\mathbf{S}, \mathbf{U}, \mathbf{F})$ and vector $\dot{\mathbf{x}}$ is conducted to decrease the impacts of noise and physical scale of the data. The optimal term combination, $\boldsymbol{\gamma}^*$, will be determined by solving Eq. (11) using column-wise normalized data of $\boldsymbol{\Theta}(\mathbf{S}, \mathbf{U}, \mathbf{F})$ and $\dot{\mathbf{x}}$. Then, the coefficients $\boldsymbol{\xi}^*$ on the original data scale can be identified by using the least square algorithm on the original data matrices $\boldsymbol{\Theta}(\mathbf{S}, \mathbf{U}, \mathbf{F})$ and $\dot{\mathbf{x}}$. The column corresponding to nonzero value in $\boldsymbol{\gamma}^*$ is used for $\boldsymbol{\xi}^*$ estimation.

Thus, the discovered governing DAE for displacement $\dot{\mathbf{x}}$ can be given as follows:

$$\dot{x} = \boldsymbol{\theta}(\mathbf{s}, \mathbf{u}, \mathbf{F}) \cdot (\boldsymbol{\gamma}^* \circ \boldsymbol{\xi}^*). \quad (12)$$

The above algorithmic procedure is repeated to discover all the unknown DAEs in Eq. (5). In addition, chatter stability analysis can be conducted from both frequency and/or time domains by using the discovered governing DAEs of machining dynamics.

As a summary, the proposed CMML modeling method consists of two sequential parts: a mechanics-based DAEs model structure as the physical foundation and a data-driven machine learning model as the governing equation discovery. The DAEs model structure constrains the learning space for the exploration of machine learning modeling to fully respect existing cutting mechanics. Then, the discovery of governing equations can be achieved by solving the sparse optimization problem in Eq. (11) to optimality by using off-the-shelf discrete optimization method, i.e., the branch-and-bound algorithm adopted in [97]. Theoretically, the time complexity of branch-and-bound algorithm is exponential, and its scalability is not good for very large size of $\boldsymbol{\theta}(\mathbf{s}, \mathbf{u}, \mathbf{F})$. However, modern optimization techniques can solve the problem in Eq. (11) to optimality for a relatively large problem size. In addition, the size of candidate terms in $\boldsymbol{\theta}(\mathbf{s}, \mathbf{u}, \mathbf{F})$ can be easily manageable based on existing physical knowledge of machining. That means it is unwise to put any unnecessary candidate terms violating the physics of cutting mechanics in $\boldsymbol{\theta}(\mathbf{s}, \mathbf{u}, \mathbf{F})$. Consequently, the proposed CMML modeling method is sufficient for the use of model discovery of machining dynamics.

4.3 Simulation Experiments

Experimentally verified high-fidelity time domain simulation of milling is used to generate dynamic process data to validate the proposed CMML modeling method, in terms

of its accuracy, efficiency, and robustness. All simulation experiments are performed on a mobile workstation with Intel(R) Xeon(R) W-10885M CPU @ 2.40GHz, 128 GB memory, and 64-bit Windows 11 Pro operating system for workstations.

4.3.1 Simulation Experiments Setting

Time Domain Simulation of Milling. Experimentally verified time domain simulator of milling in [96] is used for dynamic process data generation. This sim-to-real simulator enables the high accuracy numerical solution of the second order and coupled DDEs for milling dynamics by forward Euler method and has shown good agreement with physical machine tools [81-83]. It is well suited to incorporate the inherent complexities of milling dynamics, including complicated tool geometries (e.g., runout, different radii, number of the cutter teeth, non-uniform teeth spacing, and variable helix angles) and the nonlinearity that occurs due to interrupted cutting. Two case studies of governing equations discovery for milling dynamics are presented with increasing complexities:

The above two milling cases refer to the most representative milling dynamics models that are dominantly used for decades of machining research. The use of these cases can demonstrate the general applicability of CMML method. It is noted that since turning dynamics can be regarded as a special case of milling dynamics by assuming constant immersion angle, the case study of turning is not presented in this paper. In both cases of milling, time series data is generated by time domain simulation. Gaussian noise is added to each measurable variable, namely the displacement, velocity, acceleration, and cutting forces as shown in Eq. (13):

$$\mathbf{S}^{noise} = \mathbf{S} + r\sigma_S\boldsymbol{\epsilon}. \quad (13)$$

$\mathbf{S} \in \mathbb{R}^{N \times 1}$ is an arbitrary column of data matrices $\mathbf{S}$, $\hat{\mathbf{S}}$, and $\mathbf{F}$. σ_S is the standard deviation of the data column. $\boldsymbol{\epsilon} \in \mathbb{R}^{N \times 1}$ is the added noise vector with each entry drawn from a standard normal distribution $\mathcal{N}(0,1)$. r is the noise ratio, where $r \in \{0, 0.01\%, 0.1\%, 1\%, 10\%, 50\%, 100\%\}$. Larger noise ratios indicate larger noises in data. A total of seven noise ratios are used to validate the proposed CMML modeling method.

- *Case I:* Milling dynamics with only time delayed effect in the cutting force model.
- *Case II:* Milling dynamics with time delayed effect, process damping, and edge force in the force model.

Parameter Settings of CMML. The setting of three parameters of CMML is given as follows. The number of active terms k is set to be the exact number of physical terms in the milling dynamics model, to test the capability of exact model discovery by CMML. The ℓ_2 -norm weight λ_2 is set to be 100 for dealing with data under large noise. The M value for the bounds is set to be 1000. The hyperparameter tuning procedure for parameters λ_2 and k by cross validation in [97] can also be applied. The discrete optimization problem in Eq. (11) is solved by branch-and-cut algorithm in modern discrete optimization solver CPLEX with version 20.1.

Performance Metric. To show the capability of CMML for exact governing equations discovery, the metric on the number of exactly discovered DAEs in Eq. (5) is defined as:

$$A := \sum_{i=1}^{2p+2q+2} 1_{\mathbf{r}_i^* = \mathbf{r}_i^\dagger} = \begin{cases} 1, & \text{if } \mathbf{r}_i^* = \mathbf{r}_i^\dagger \\ 0, & \text{if } \mathbf{r}_i^* \neq \mathbf{r}_i^\dagger \end{cases} \quad (14)$$

$\mathbf{r}_i^*$ indicates the active terms of the i -th DAE discovered by CMML, and $\mathbf{r}_i^\dagger$ indicates the ground truth terms in the given governing equations of the studied milling system. Note

here once the two algebraic equations for F_t and F_n are discovered, F_x and F_y will be determined. As such, in total $2p + 2q + 2$ DAEs need to be discovered by CMML.

In addition to the metric in Eq. (14), the accuracy of coefficients of discovered governing equations is characterized by the mean absolute percentage error (MAPE):

$$MAPE := \frac{1}{\sum_{i=1}^{2p+2q+2} \|\mathbf{r}_i^\dagger\|_0} \sum_{i=1}^{2p+2q+2} \sum_{p=1} \|\mathbf{r}_i^\dagger\|_0 \left| \frac{\xi_{ip}^* - \xi_{ip}^\dagger}{\xi_{ip}^\dagger} \right| \times 100\%. \quad (15)$$

In Eq. (15), ξ_{ip}^* and $\xi_{ip}^\dagger$ denote the discovered coefficient and ground truth coefficient for the p -th physical term of the i -th equation, $\|\mathbf{r}_i^\dagger\|_0$ is the number of physical terms in the i -th ground truth governing equation. For correctly discovered governing equations, MAPE can quantify the deviation of model coefficients from the ground truth coefficients.

4.3.2 Case I: Milling with Time Delayed Effect in Forces

4.3.2.1 Exact Milling Governing Equations of Case I

In this case study, the milling system with two orthogonal directions and a single mode on each direction is considered, namely $p = 1$ and $q = 1$. The force models only include regenerative time-delayed effect as shown in Eq. (16).

$$m_x \ddot{x}(t) + c_x \dot{x}(t) + k_x x(t) = F_x(t), \quad (16a)$$

$$m_y \ddot{y}(t) + c_y \dot{y}(t) + k_y y(t) = F_y(t). \quad (16b)$$

Cutting forces on the x and y directions are

$$F_x = -F_t \cos(\phi) + F_n \sin(\phi), \quad (16c)$$

$$F_y = F_t \sin(\phi) + F_n \cos(\phi). \quad (16d)$$

The tangential and normal force models are

$$F_t(t) = k_{tc}b(f_t \sin \phi + n(t - \tau) - n(t)), \quad (16e)$$

$$F_n(t) = k_{nc}b(f_t \sin \phi + n(t - \tau) - n(t)). \quad (16f)$$

k_{tc} and k_{nc} are cutting force coefficients for tangential and normal directions, f_t is feed per tooth and $f_t \sin(\phi) + n(t - \tau) - n(t)$ is instantaneous chip thickness.

To validate the proposed CMML modeling method, the DDEs in Eq. (16) are further equivalently represented as a set of DAEs for the studied milling system in Eq. (17).

$$\dot{x} = v_x, \quad (17a)$$

$$\dot{v}_x = \frac{-c_x v_x - k_x x + F_x}{m_x}, \quad (17b)$$

$$\dot{y} = v_y, \quad (17c)$$

$$\dot{v}_y = \frac{-c_y v_y - k_y y + F_y}{m_y}, \quad (17d)$$

$$F_t = k_{tc}b(f_t \sin \phi + n(t - \tau) - n(t)), \quad (17e)$$

$$F_n = k_{nc}b(f_t \sin \phi + n(t - \tau) - n(t)), \quad (17f)$$

$$F_x = -F_t \cos(\phi) + F_n \sin(\phi), \quad (17g)$$

$$F_y = F_t \sin(\phi) + F_n \cos(\phi). \quad (17h)$$

Note that only Eqs. (17a) - (17f) are the six DAEs that need to be discovered by CMML.

Eqs. (17g) - (17h) will be determined once Eqs. (17e) - (17f) are determined.

To discover the DAEs in Eqs. (17a) - (17f), the second-order polynomials are used to construct candidate terms in candidate set $\boldsymbol{\theta}(\mathbf{s}, \mathbf{u}, \mathbf{F})$ of CMML. By assuming zero cross talk between x and y directions, $\mathbf{s} = \{x, v_x\}$, $\mathbf{u} = \{b\}$, and $\mathbf{F} = \{F_x\}$ are used for discovering Eqs. (17a) - (17b); similarly, $\mathbf{s} = \{y, v_y\}$, $\mathbf{u} = \{b\}$, and $\mathbf{F} = \{F_y\}$ are used for discovering Eqs. (17c) - (17d). For discovering Eqs. (17e) - (17f), let $\Delta n = n(t - \tau) - n(t)$ denote the time delayed term. Thus, $\mathbf{s} = \{\Delta n\}$, $\mathbf{u} = \{b, \sin \phi\}$ and $\mathbf{F} = \emptyset$ are used to construct $\boldsymbol{\theta}(\mathbf{s}, \mathbf{u}, \mathbf{F})$. Δn can be calculated with x and y based on the time delayed effect and projection relationship of cutting mechanics. Note that the spindle speed is not

necessary to be included into $\mathbf{u}$, for it is used to calculate the time delay τ as $\tau = 1/(\Omega N_t)$ with a given Ω in rps and the number of teeth N_t .

4.3.2.2 Time Domain Simulation Setting for Case I

The setting of time domain simulation for Case I is given as follows. The modal parameters of the symmetric structural dynamics used are: $m_x = m_y = 0.198$ kg, $k_x = k_y = 5 \times 10^6$ N/m, and $c_x = c_y = 19.91$ N s/m. The cutting force coefficient and angle used are: $K_s = 750 \times 10^6$ N/m² and $\beta = 68$ degree. Thus, the tangential and normal cutting force parameters used are: $k_{tc} = K_s \sin(\beta) = 695,387,669.36$ N/m² and $k_{nc} = K_s \cos(\beta) = 280,954,790.02$ N/m². The cutter has a diameter of 20 mm and four equally spaced straight teeth. Up milling with 25% radial immersion and feed rate $f_t = 0.1$ mm/tooth is used. The time domain simulation is executed for 40 revolutions and each revolution is discretized into 1000 steps. A set of process parameters $\langle \Omega, b \rangle$ is sampled to obtain dynamic process data from stable and unstable cuts, where $\Omega \in \{4000, 6000, 8000, 10000, 12000\}$ rpm and $b \in \{2, 4, 6, 8, 10, 12\}$ mm. Notably, the transient state data, which contains rich information of milling dynamics, is used for training of CMML. Thus, the first 2000 data points are used for each combination of $\langle \Omega, b \rangle$. As the cutting depth b is treated as a variable, all the six datasets from six cutting depth b under one spindle speed are stacked as a single dataset. Thus, in total 12000 data points for each spindle speed are used for training of CMML to discover the set of DAEs in Eq. (17).

4.3.2.3 Numerical Results and Analyses of Case I

The numerical results of CMML for Case I of milling dynamics under different noise ratios are displayed in Table 4.1, Table 4.2, and Table 4.3. Table 4.1 shows the number of exactly discovered DAEs, based on the A value, defined in Eq. (14). The number 6 indicates that all 4 differential equations to describe structural dynamics and 2 algebraic equations to represent F_n and F_t are exactly discovered. Under $\Omega = 6000$ rpm, CMML modeling method can exactly discover all the six DAEs from noisy data with noise ratio up to 50% (grey cells). Correspondingly, the discovered DAEs are presented in Table 4.2. The MAPE of coefficients of correctly discovered governing equations by CMML are shown in Table 4.3. For the dynamic data with noise levels under 1%, the MAPE metric in Eq. (15) is almost zero. For the extremely high noise level of 50%, the MAPE is still less than 8%. This shows the high model accuracy of CMML in terms of both the discovery of the physical terms and determination of model coefficients. Moreover, the computing time of CMML is 7.67 seconds on average for all noise levels for discovering the set of DAEs in Eq. (17). In general, this shows the feasibility of the proposed CMML modeling method for use to discover governing equations of milling dynamics from highly noisy experimental data.

Using the discovered DAEs with noise ratio from 0.00% (exact governing equations) to 50%, stability lobe diagrams are plotted in Figure 4.2 by using zero order method [99]. It is seen the stability lobes with noise ratio $r = 0.01\%$, 0.1% , 1% , and 10% almost overlap

Table 4.1. Number of correct DAEs (A value) discovered by CMML for the milling dynamics in Case I (with only time delayed effect in force models).

Noise ratio r	Spindle Speed Ω (rpm)				
	4000	6000	8000	10000	12000
Noise Free	6	6	6	6	6
0.01%	6	6	6	6	6
0.1%	6	6	6	6	6
1%	6	6	6	6	6
10%	6	6	6	6	6
50%	5	6	5	4	4
100%	4	4	4	4	4
500%	4	4	3	4	3
1000%	2	2	2	2	2

Table 4.2. Discovered DAEs for the milling dynamics in Case I (with only time delayed effect in force model for $\Omega = 6000$ rpm in Table 4.1).

Noise ratio r	Discovered DAEs
Noise Free (exact governing equations)	$\dot{x} = v_x$ $\dot{v}_x = -25266187.27x - 100.53v_x + 5.05F_x$ $\dot{y} = v_y$ $\dot{v}_y = -25266187.27y - 100.53v_y + 5.05F_y$ $F_t = -695387890.93\Delta nb + 69538.79 \sin \phi b$ $F_n = 280954945.06\Delta nb - 28095.50 \sin \phi b$
0.01%	$\dot{x} = v_x$ $\dot{v}_x = -25266192.30x - 100.53v_x + 5.05F_x$ $\dot{y} = v_y$ $\dot{v}_y = -25266191.61y - 100.53v_y + 5.05F_y$ $F_t = -695387669.36\Delta nb + 69538.81 \sin \phi b$ $F_n = 280954790.02\Delta nb - 28095.48 \sin \phi b$
0.1%	$\dot{x} = v_x$ $\dot{v}_x = -25266237.52x - 100.53v_x + 5.05F_x$ $\dot{y} = v_y$ $\dot{v}_y = -25266230.73y - 100.53v_y + 5.05F_y$ $F_t = -695385675.23\Delta nb + 69539.00 \sin \phi b$ $F_n = 280953394.63\Delta nb - 28095.36 \sin \phi b$
1%	$\dot{x} = v_x$ $\dot{v}_x = -25266679.36x - 100.55v_x + 5.05F_x$ $\dot{y} = v_y$ $\dot{v}_y = -25266620.77y - 100.50v_y + 5.05F_y$ $F_t = -695365734.00\Delta nb + 69540.87 \sin \phi b$ $F_n = 280939440.76\Delta nb - 28094.17 \sin \phi b$
10%	$\dot{x} = v_x$ $\dot{v}_x = -25270056.27x - 99.72v_x + 5.04F_x$ $\dot{y} = v_y$ $\dot{v}_y = -25270392.27y - 98.84v_y + 5.03F_y$ $F_t = -695166321.72\Delta nb + 69559.61 \sin \phi b$ $F_n = 280799902.02\Delta nb - 28082.23 \sin \phi b$
50%	$\dot{x} = v_x$ $\dot{v}_x = -25262721.75x - 75.51v_x + 4.70F_x$ $\dot{y} = v_y$ $\dot{v}_y = -25282589.36y - 66.14v_y + 4.43F_y$ $F_t = -694280044.90\Delta nb + 69642.87 \sin \phi b$ $F_n = 280179729.87\Delta nb - 28029.19 \sin \phi b$

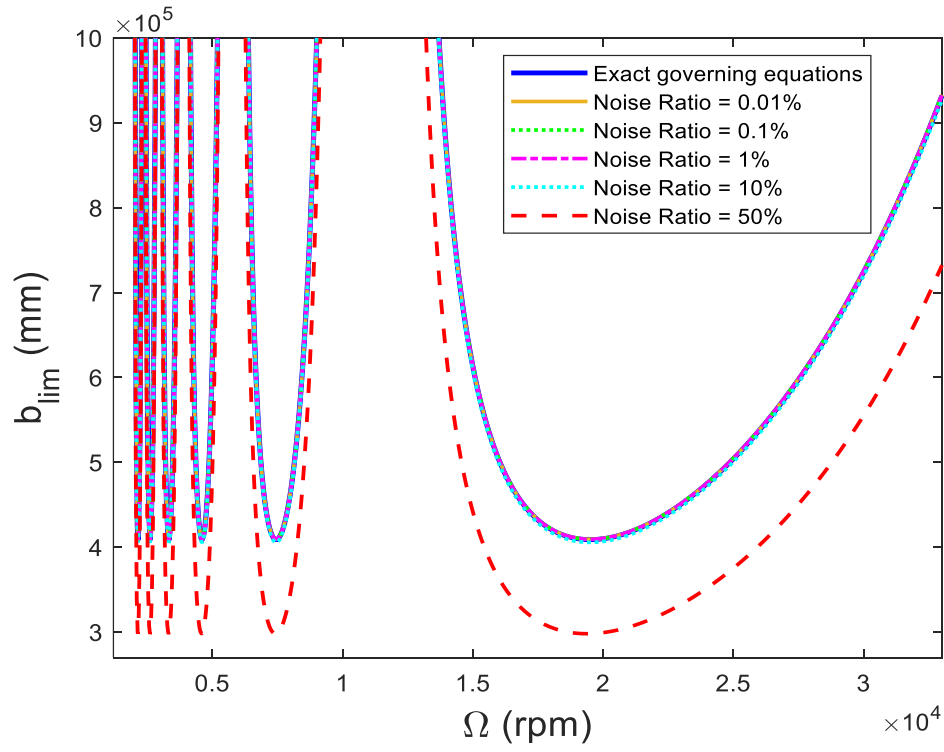
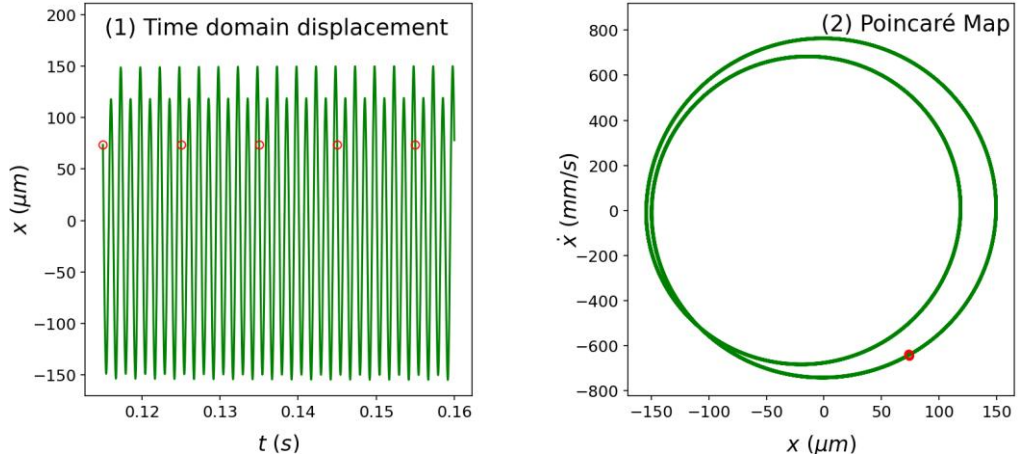


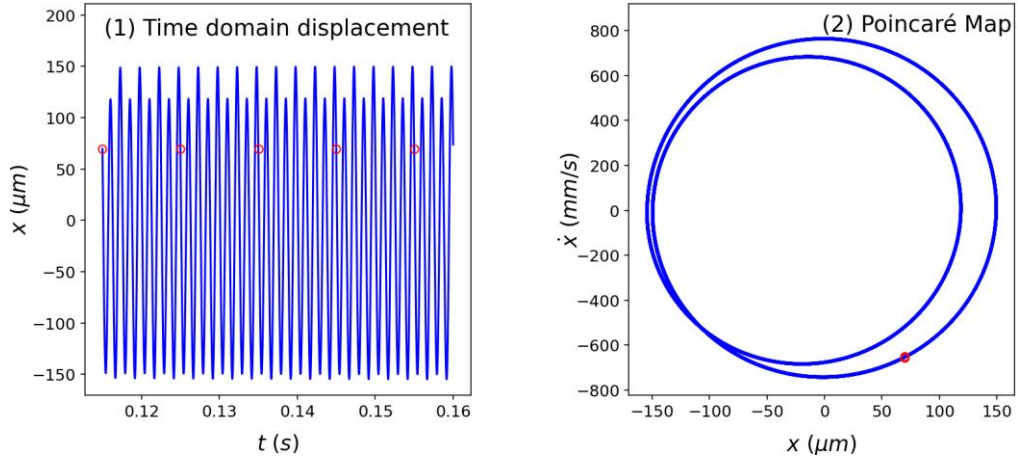
Figure 4.2. The stability lobe diagrams for $\Omega = 6000$ rpm in Table 4.2. The stability lobes under noises 0.01%, 0.1%, 1%, 10% almost perfectly overlap with stability lobes by exact governing equations.

Table 4.3. MAPE of model coefficients of correctly discovered governing equations in Table 4.2 by CMML for Case I.

Noise ratio r	0.01%	0.1%	1%	10%	50%
MAPE	0.00%	0.00%	0.01%	0.33%	7.92%



(a) Trajectory of exact governing equations



- (b) Trajectory of discovered DAEs under noise $r = 1\%$ in Table 4.2.

Figure 4.3. Both (a) and (b) show the time domain displacement (on x direction) and the Poincaré map of discovered equations in Table 4.2 using parameters $\Omega = 6000$ rpm and $b = 2$ mm. The single group of red points from once-per-tooth sampling shows the cut is stable.

with that by the exact governing equations. This shows that the discovered DAEs are sufficiently accurate and robust for supporting stability analysis. As the noise ratio increases to 50%, the stability lobes (red virtual lines) deviate from the ground truth lobes (blue solid line).

In addition, the time domain simulation results of the discovered DAEs are also presented. The time domain displacement and Poincaré maps obtained from the cut of $\Omega = 6000$ rpm and $b = 2$ mm are shown in Figure 4.3. The results for exact governing equations and equations discovered under noise ratio $r = 1\%$ are presented in Figure 4.3(a) and Figure 4.3 (b), respectively. The Poincaré map plots the velocity against the displacement. The single group of red points in Poincaré maps from once-per-tooth sampling indicate that the sampled points repeat, and the milling operation is stable. In general, both the time domain displacement and Poincaré map by the discovered DAEs align well with those by the exact governing equations.

4.3.3 *Case II: Milling with Time Delayed Effect, Nonlinear Process Damping and Edge Force*

This section presents the numerical results and analyses for milling dynamics in Case II with time delayed effect, process damping, and edge force. Nonlinear process damping is used to validate CMML for nonlinear DAEs discovery.

4.3.3.1 Exact Milling Governing Equations and Simulation Data Generation of Case II

The exact governing equations used for Case II are the same as those in Case I for Eqs. (16a) - (16d), except for the following cutting force models:

$$F_t = k_{tc}b(f_t \sin \phi + n(t - \tau) - n(t)) + k_{te}b - C_t \frac{b}{V} \dot{n}^2, \quad (18a)$$

$$F_n = k_{nc}b(f_t \sin \phi + n(t - \tau) - n(t)) + k_{ne}b - C_n \frac{b}{V} \dot{n}^2. \quad (18b)$$

Instead of using the linear process damping force in [77], nonlinear process damping terms $-C_t \frac{b}{V} \dot{n}^2$ and $-C_n \frac{b}{V} \dot{n}^2$ are informally introduced in the cutting force model in Eq. (18) for demonstrating the capability of CMML for nonlinear model discovery. For the introduced nonlinear process damping term $-C \frac{b}{V} \dot{n}^2$, $\dot{n}$, b , V , and C are the normal velocity, chip width, cutting speed, and constant coefficient. For the cutting force coefficients k_{tc} , k_{nc} , k_{te} , and k_{ne} , subscripts t and n refer to the tangential and normal directions, c and e refer to the cutting and edge effects.

The simulation data is obtained from time domain simulation using the same parameter settings in Case I except for the following: process damping coefficients $C_t = C_n = 1.4 \times 10^3 \text{ N s/m}^2$, and edge force coefficients $k_{te} = k_{ne} = 2.5 \times 10^4 \text{ N/m}$. Because of the introduced nonlinear process damping term, third-order polynomials of $\mathbf{s} = \{\Delta n, \dot{n}\}$ and $\mathbf{u} = \{b, \sin \phi\}$ are used by CMML for discovering the algebraic equations of cutting forces. Δn and $\dot{n}$ can be calculated with x, v_x, y , and v_y based on the time delayed effect and projection relationship of cutting mechanics. Other settings of CMML modeling method remain unchanged as in Case I.

4.3.3.2 Numerical Results and Analyses of Case II

The numerical results of CMML to discover the DAEs of milling dynamics in Case II under different noise ratios are summarized in Table 4.4, Table 4.5 and Table 4.6, and Figure 4.4. The number of exactly discovered DAEs for milling dynamics in Case II is

presented in Table 4.4,. It is seen that CMML can exactly discover all the six DAEs from noisy data with noise ratio up to 50%. The computing time of CMML is 9.23 seconds on average for discovering a set of DAEs of milling dynamics in Case II. In addition, it is seen a lower value of Ω can lead to better performance of CMML for exact DAEs discovery. This can be used for guiding experiment design as lower value of spindle speed is good for tool life and can reduce potential damage to machine tools.

In addition, Table 4.5 shows the discovered DAEs of $\Omega = 6000$ rpm under different noise ratios (see grey cells of Table 4.4,). All physical terms including the nonlinear process damping are discovered correctly. Table 4.6 shows the MAPE of coefficients of all correctly discovered governing equations by CMML in Table 4.5. Clearly, CMML obtains high model accuracy in terms of coefficients determination, with 0.68% of MAPE for noise level of 10%, and 8.03% for the extremely high noise level of 50%. In Figure 4.4, the time domain displacement and Poincaré maps are plotted for the exact governing equations and the DAEs obtained under noise ratio 1% in Table 4.5 with parameters $\Omega = 6000$ rpm and $b = 2$ mm. Similar conclusions can be made as in Case I. The stability lobe diagram is not presented due to no available methods to obtain the stability solutions of nonlinear governing equations. In general, the numerical results of Case II show that CMML can discover accurate nonlinear governing equations of machining dynamics from highly noisy data.

Table 4.4. Number of correct DAEs (A value) discovered by CMML for the milling dynamics in Case II with time delayed effect, nonlinear process damping, and edge effect in the force models.

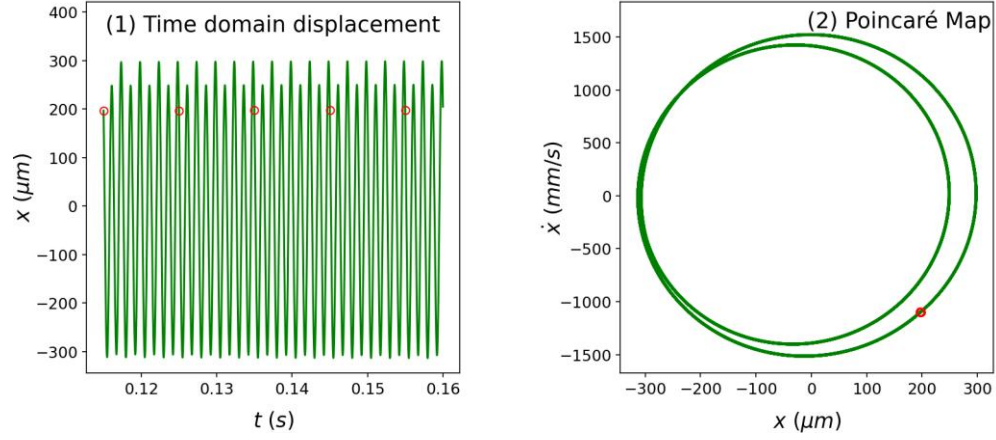
	Spindle Speed Ω (rpm)				
Noise ratio r	4000	6000	8000	10000	12000
Noise Free	6	6	6	6	6
0.01%	6	6	6	6	6
0.1%	6	6	6	6	6
1%	6	6	6	6	6
10%	6	6	4	5	6
50%	6	6	2	2	4
100%	4	4	3	3	3
500%	2	3	2	2	2
1000%	2	2	2	2	2

Table 4.5. MAPE of model coefficients of correctly discovered governing equations in Table 4.5 by CMML for Case II.

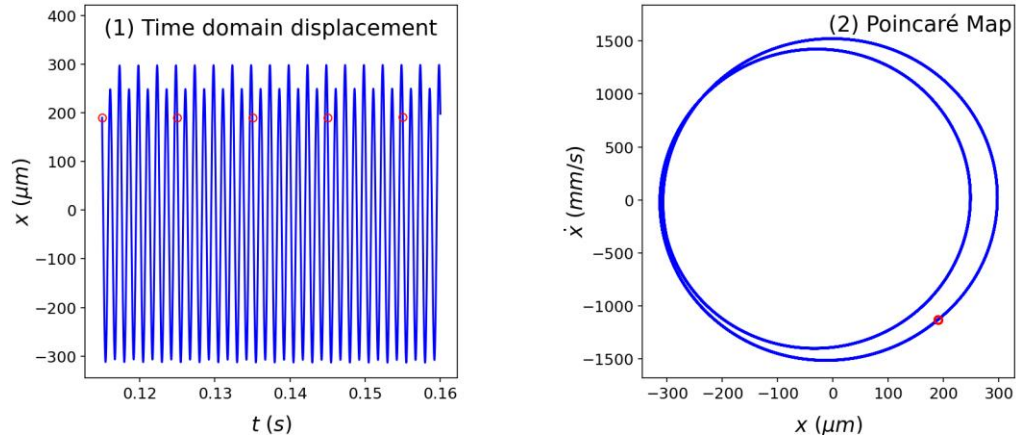
Noise ratio r	Discovered DAEs
Noise Free (exact governing equations)	$\dot{x} = v_x$ $\dot{v}_x = -25266187.27x - 100.53v_x + 5.05F_x$ $\dot{y} = v_y$ $\dot{v}_y = -25266187.27y - 100.53v_y + 5.05F_y$ $F_t = 25000.00b - 695387890.92\Delta nb + 69538.79 \sin \phi b - 222.82\dot{n}^2b$ $F_n = 25000.00b + 280954945.06\Delta nb - 28095.49 \sin \phi b + 222.82\dot{n}^2b$
0.01%	$\dot{x} = v_x$ $\dot{v}_x = -25266185.24x - 100.53v_x + 5.05F_x$ $\dot{y} = v_y$ $\dot{v}_y = -25266192.35y - 100.53v_y + 5.05F_y$ $F_t = 25000.44b - 695388128.39\Delta nb + 69538.33 \sin \phi b - 222.81\dot{n}^2b$ $F_n = 24999.86b + 280954746.63\Delta nb - 28095.88 \sin \phi b + 222.82\dot{n}^2b$
0.1%	$\dot{x} = v_x$ $\dot{v}_x = -25266166.88x - 100.54v_x + 5.05F_x$ $\dot{y} = v_y$ $\dot{v}_y = -25266237.93y - 100.53v_y + 5.05F_y$ $F_t = 25004.41b - 695390265.56\Delta nb + 69534.22 \sin \phi b - 222.80\dot{n}^2b$ $F_n = 24998.61b + 280952960.82\Delta nb - 28099.37 \sin \phi b + 222.83\dot{n}^2b$
1%	$\dot{x} = v_x$ $\dot{v}_x = -25265972.09x - 100.59v_x + 5.05F_x$ $\dot{y} = v_y$ $\dot{v}_y = -25266678.29y - 100.46v_y + 5.05F_y$ $F_t = 25044.14b - 695411637.29\Delta nb + 69493.07 \sin \phi b - 222.61\dot{n}^2b$ $F_n = 24986.10b + 280935102.62\Delta nb - 28134.27 \sin \phi b + 222.96\dot{n}^2b$
10%	$\dot{x} = v_x$ $\dot{v}_x = -25262914.86x - 99.77v_x + 5.04F_x$ $\dot{y} = v_y$ $\dot{v}_y = -25269532.49y - 98.42v_y + 5.03F_y$ $F_t = 25441.38b - 695625354.64\Delta nb + 69081.56 \sin \phi b - 220.80\dot{n}^2b$ $F_n = 24861.03b + 280756520.68\Delta nb - 28483.26 \sin \phi b + 224.26\dot{n}^2b$
50%	$\dot{x} = v_x$ $\dot{v}_x = -25227362.02x - 69.90v_x + 4.73F_x$ $\dot{y} = v_y$ $\dot{v}_y = -25249866.63y - 62.19v_y + 4.69F_y$ $F_t = 27206.92b - 696575209.50\Delta nb + 67252.64 \sin \phi b - 212.71\dot{n}^2b$ $F_n = 24305.18b + 279962823.18\Delta nb - 30034.33 \sin \phi b + 230.01\dot{n}^2b$

Table 4.6. MAPE of model coefficients of correctly discovered governing equations in Table 4.5 by CMML for Case II.

Noise ratio r	0.01%	0.1%	1%	10%	50%
MAPE	0.00%	0.00%	0.05%	0.68%	8.03%



(a) Trajectory of exact governing equations



(b) Trajectory of discovered DAEs under noise $r = 1\%$ in Table 4.5.

Figure 4.4. Both (a) and (b) show the time domain displacement (for x direction) and the Poincaré map of discovered equations in Table 4.5 using parameters $\Omega = 6000$ rpm and $b = 2$ mm. The single group of red points from once-per-tooth sampling shows the cut is stable.

In summary, numerical results in Section 3.2 and 3.3 demonstrate the effectiveness, accuracy, and robustness of the proposed CMML modeling method for discovering governing equations of machining dynamics. Under high noise levels, CMML can successfully discover the exact DAEs for machining dynamics in both Cases I and II with time delayed effect, process damping, and edge effect. This indicates that CMML has the potential to be used for advancing machining modeling in practice with the development of effective metrology systems.

4.4 Conclusions

This paper proposes a cutting mechanics-based machine learning (CMML) modeling method, that provides a new pathway to integrate existing physics in cutting mechanics and unknown physics in data, to discover governing equations of machining dynamics. In CMML, existing well-established physical principles of cutting mechanics are utilized at first to establish a general modeling structure to guide data-driven algorithm design. This is done through representing the second order, coupled, time delayed differential equations (DDEs) by a set of unknown differential algebraic equations. This representation strictly follows the known physics in both structural dynamics and cutting forces. Then, CMML can achieve data-driven discovery of these unknown differential algebraic equations by applying customized machine learning method, consisting of nonlinear learning function space design, loss function setting and discrete optimization-based training algorithm. Experimentally verified time domain simulation of milling is used to validate the proposed modeling method. Numerical results show CMML can discover the exact milling dynamics

governing equations with time delayed effect, process damping, and edge force from noisy data.

Future work will focus on developing effective metrology systems to simultaneously measure the cutting force and at least one among the displacement, velocity and acceleration. For example, a force dynamometer and two accelerometers can be set up for simultaneously measuring the cutting forces and accelerations on x and y directions of the machine's Cartesian coordinate. The proposed CMML modeling method can therefore be used for discovery of nonlinear governing equations of machining dynamics from experimental data to advance machining modeling, especially for nonlinear process damping models. In addition, for dealing with the potential nonlinear DDE models discovered by CMML combined with various tool geometries, efficient time domain simulation-based stability methods will be considered to accurately and fast generate stability lobe diagrams.

CHAPTER 5

CONCLUSIONS

In conclusion, this dissertation contributes to advancing manufacturing systems by developing machine learning based modeling and optimization methods for problems across multiple scales of manufacturing factories, from factory-scale order scheduling to machine tool-scale cost optimization and machining dynamics-scale machining modeling. The main contributions of this dissertation are summarized as follows:

- To address large-scale customer order scheduling problem in discrete-event dynamics, a learning-based two-stage optimization method is developed, which combines a feature-enhanced genetic programming algorithm to automatically design high-quality dispatching rules and further enhance the solution quality with an adaptive local search algorithm. Experimental results demonstrate that the learned dispatching rules outperform state-of-the-art dispatching rules in literature, and the proposed two-stage optimization method outperforms existing approaches in terms of both solution quality and computational efficiency.
- To address the machine tool operations cost optimization problem, this study proposes a cost function that integrates discrete-event dynamics with machining dynamics. A set of mathematical models and optimization algorithms are developed to tackle the integrated optimization problems with considerations of different practical production requirements. Numerical experiments validate the effectiveness of the proposed approach and highlight its potential use for cost saving in practice.

- To automate the discovery of governing equations for machining dynamics, a cutting mechanics-based machine learning modeling method is presented. Based on existing machining knowledge of cutting mechanics, a general modeling structure represented by differential algebraic equations is established, and the unknown governing equations will be further discovered by interpretable machine learning algorithm. By enabling accurate identification of governing equations, the proposed CMML method advances the understanding of machining dynamics, therefore can be used for nonlinear governing equations discovery of machining dynamics from experimental data to advance machining dynamics modeling.

Overall, these systematically structured studies and their corresponding contributions uniquely contribute to the field of manufacturing operations management. By developing machine learning based modeling and optimization techniques, this dissertation demonstrates their effectiveness to improve operational efficiency, reduce production costs, and enhance overall manufacturing system performance, thereby establishing a strong foundation for future research and practical implementation in more productive and competitive manufacturing enterprises.

LIST OF REFERENCES

- [1] Karandikar, J., T. Schmitz, and S. Smith, *Physics-guided logistic classification for tool life modeling and process parameter optimization in machining*. Journal of Manufacturing Systems, 2021. **59**: p. 522-534.
- [2] Leung, J.Y.-T., H. Li, and M. Pinedo, *Order scheduling in an environment with dedicated resources in parallel*. Journal of Scheduling, 2005. **8**: p. 355-386.
- [3] Leung, J.Y.-T., H. Li, and M. Pinedo, *Scheduling orders for multiple product types with due date related objectives*. European Journal of Operational Research, 2006. **168**(2): p. 370-389.
- [4] Venditti, L., D. Pacciarelli, and C. Meloni, *A tabu search algorithm for scheduling pharmaceutical packaging operations*. European Journal of Operational Research, 2010. **202**(2): p. 538-546.
- [5] Yang, J. and M.E. Posner, *Scheduling parallel machines for the customer order problem*. Journal of Scheduling, 2005. **8**: p. 49-74.
- [6] Lu, S., J. Pei, X. Liu, X. Qian, N. Mladenovic, and P.M. Pardalos, *Less is more: variable neighborhood search for integrated production and assembly in smart manufacturing*. Journal of Scheduling, 2020. **23**: p. 649-664.
- [7] Julien, F. and M. Magazine, *Scheduling customer orders: An alternative production scheduling approach*. Journal of Manufacturing and Operations Management, 1990. **3**(3): p. 177-199.
- [8] Wagneur, E. and C. Sriskandarajah, *Openshops with jobs overlap*. European Journal of Operational Research, 1993. **71**(3): p. 366-378.

- [9] Leung, J.Y., H. Li, M. Pinedo, and C. Sriskandarajah, *Open shops with jobs overlap—revisited*. European Journal of Operational Research, 2005. **163**(2): p. 569-571.
- [10] Roemer, T.A., *A note on the complexity of the concurrent open shop problem*. Journal of scheduling, 2006. **9**: p. 389-396.
- [11] Blocher, J.D. and D. Chhajed, *The customer order lead - time problem on parallel machines*. Naval Research Logistics (NRL), 1996. **43**(5): p. 629-654.
- [12] Wang, G. and T.E. Cheng, *Customer order scheduling to minimize total weighted completion time*. Omega, 2007. **35**(5): p. 623-626.
- [13] Shi, Z., L. Wang, P. Liu, and L. Shi, *Minimizing completion time for order scheduling: Formulation and heuristic algorithm*. IEEE Transactions on Automation Science and Engineering, 2015. **14**(4): p. 1558-1569.
- [14] Leung, J.Y.-T., H. Li, and M. Pinedo, *Scheduling orders for multiple product types to minimize total weighted completion time*. Discrete Applied Mathematics, 2007. **155**(8): p. 945-970.
- [15] Wang, L., Z. Shi, and L. Shi. *A novel quadratic formulation for customer order scheduling problem*. in *2013 IEEE International Conference on Automation Science and Engineering (CASE)*. 2013. IEEE.
- [16] Framinan, J.M. and P. Perez-Gonzalez, *Order scheduling with tardiness objective: Improved approximate solutions*. European Journal of Operational Research, 2018. **266**(3): p. 840-850.

- [17] Wu, C.-C., S.-C. Liu, T.-Y. Lin, T.-H. Yang, I.-H. Chung, and W.-C. Lin, *Bicriterion total flowtime and maximum tardiness minimization for an order scheduling problem*. Computers & Industrial Engineering, 2018. **117**: p. 152-163.
- [18] Xu, J., C.-C. Wu, Y. Yin, C. Zhao, Y.-T. Chiou, and W.-C. Lin, *An order scheduling problem with position-based learning effect*. Computers & Operations Research, 2016. **74**: p. 175-186.
- [19] Wu, C.-C., T.-H. Yang, X. Zhang, C.-C. Kang, I.-H. Chung, and W.-C. Lin, *Using heuristic and iterative greedy algorithms for the total weighted completion time order scheduling with release times*. Swarm and evolutionary computation, 2019. **44**: p. 913-926.
- [20] Lin, W.-C., J. Xu, D. Bai, I.-H. Chung, S.-C. Liu, and C.-C. Wu, *Artificial bee colony algorithms for the order scheduling with release dates*. Soft Computing, 2019. **23**: p. 8677-8688.
- [21] Zhang, Y., Y. Dan, B. Dan, and H. Gao, *The order scheduling problem of product-service system with time windows*. Computers & Industrial Engineering, 2019. **133**: p. 253-266.
- [22] Chen, R.-X. and S.-S. Li, *Minimizing maximum delivery completion time for order scheduling with rejection*. Journal of Combinatorial Optimization, 2020. **40**(4): p. 1044-1064.
- [23] de Athayde Prata, B., C.D. Rodrigues, and J.M. Framinan, *Customer order scheduling problem to minimize makespan with sequence-dependent setup times*. Computers & Industrial Engineering, 2021. **151**: p. 106962.

- [24] Gupta, J.N., J.C. Ho, and J.A. van der Veen, *Single machine hierarchical scheduling with customer orders and multiple job classes*. Annals of Operations Research, 1997. **70**(0): p. 127-143.
- [25] Komaki, G. and V. Kayvanfar, *Grey Wolf Optimizer algorithm for the two-stage assembly flow shop scheduling problem with release time*. Journal of computational science, 2015. **8**: p. 109-120.
- [26] Shi, Z., Z. Huang, and L. Shi, *Customer order scheduling on batch processing machines with incompatible job families*. International Journal of Production Research, 2018. **56**(1-2): p. 795-808.
- [27] Meng, T., Q.-K. Pan, and L. Wang, *A distributed permutation flowshop scheduling problem with the customer order constraint*. Knowledge-Based Systems, 2019. **184**: p. 104894.
- [28] Xiong, F., K. Xing, F. Wang, H. Lei, and L. Han, *Minimizing the total completion time in a distributed two stage assembly system with setup times*. Computers & Operations Research, 2014. **47**: p. 92-105.
- [29] Cunha Campos, S. and J.E. Claudio Arroyo. *NSGA-II with iterated greedy for a bi-objective three-stage assembly flowshop scheduling problem*. in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*. 2014.
- [30] Jung, S., Y.-B. Woo, and B.S. Kim, *Two-stage assembly scheduling problem for processing products with dynamic component-sizes and a setup time*. Computers & industrial engineering, 2017. **104**: p. 98-113.

- [31] Dauod, H., D. Li, S.W. Yoon, and K. Srihari, *Multi-objective optimization of the order scheduling problem in mail-order pharmacy automation systems*. The International Journal of Advanced Manufacturing Technology, 2018. **99**: p. 73-83.
- [32] Lin, W.-C., Y. Yin, S.-R. Cheng, T.E. Cheng, C.-H. Wu, and C.-C. Wu, *Particle swarm optimization and opposite-based particle swarm optimization for two-agent multi-facility customer order scheduling with ready times*. Applied Soft Computing, 2017. **52**: p. 877-884.
- [33] Kung, J.-Y., J. Duan, J. Xu, I.-H. Chung, S.-R. Cheng, C.-C. Wu, and W.-C. Lin, *Metaheuristics for order scheduling problem with unequal ready times*. Discrete Dynamics in Nature and Society, 2018. **2018**(1): p. 4657368.
- [34] Wang, B., Z. Guan, S. Ullah, X. Xu, and Z. He, *Simultaneous order scheduling and mixed-model sequencing in assemble-to-order production environment: a multi-objective hybrid artificial bee colony algorithm*. Journal of Intelligent manufacturing, 2017. **28**: p. 419-436.
- [35] Hazır, Ö., Y. Günalay, and E. Erel, *Customer order scheduling problem: a comparative metaheuristics study*. The International Journal of Advanced Manufacturing Technology, 2008. **37**: p. 589-598.
- [36] Du, W., W. Zhong, Y. Tang, W. Du, and Y. Jin, *High-dimensional robust multi-objective optimization for order scheduling: A decision variable classification approach*. IEEE transactions on industrial informatics, 2018. **15**(1): p. 293-304.

- [37] Framinan, J.M., P. Perez-Gonzalez, and V. Fernandez-Viagas, *Deterministic assembly scheduling problems: A review and classification of concurrent-type scheduling models and solution procedures*. European Journal of Operational Research, 2019. **273**(2): p. 401-417.
- [38] Framinan, J.M. and P. Perez-Gonzalez, *New approximate algorithms for the customer order scheduling problem with total completion time objective*. Computers & Operations Research, 2017. **78**: p. 181-192.
- [39] Riahi, V., M.H. Newton, M.M.A. Polash, and A. Sattar, *Tailoring customer order scheduling search algorithms*. Computers & Operations Research, 2019. **108**: p. 155-165.
- [40] Koza, J.R., *Genetic programming: On the programming of computers by means of natural selection (complex adaptive systems)*. A Bradford Book, 1993. **1**: p. 18.
- [41] Burke, E.K., M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, *Hyper-heuristics: A survey of the state of the art*. Journal of the Operational Research Society, 2013. **64**(12): p. 1695-1724.
- [42] Nguyen, S., Y. Mei, and M. Zhang, *Genetic programming for production scheduling: a survey with a unified framework*. Complex & Intelligent Systems, 2017. **3**: p. 41-66.
- [43] Koza, J.R., *Genetic programming as a means for programming computers by natural selection*. Statistics and computing, 1994. **4**: p. 87-112.
- [44] O'Reilly, U.-M., *An analysis of genetic programming*. 1995, Carleton University.

- [45] Ferreira, C., *Gene expression programming: a new adaptive algorithm for solving problems*. arXiv preprint cs/0102027, 2001.
- [46] Hildebrandt, T. and J. Branke, *On using surrogates with genetic programming*. Evolutionary computation, 2015. **23**(3): p. 343-367.
- [47] Nguyen, S., M. Zhang, and K.C. Tan, *Surrogate-assisted genetic programming with simplified models for automated design of dispatching rules*. IEEE transactions on cybernetics, 2016. **47**(9): p. 2951-2965.
- [48] Shi, Z., S. Gao, J. Du, H. Ma, and L. Shi. *Automatic design of dispatching rules for real-time optimization of complex production systems*. in *2019 IEEE/SICE international symposium on system integration (SII)*. 2019. IEEE.
- [49] Hildebrandt, T., J. Heger, and B. Scholz-Reiter. *Towards improved dispatching rules for complex shop floor scenarios: a genetic programming approach*. in *Proceedings of the 12th annual conference on Genetic and evolutionary computation*. 2010.
- [50] Branke, J., T. Hildebrandt, and B. Scholz-Reiter, *Hyper-heuristic evolution of dispatching rules: A comparison of rule representations*. Evolutionary computation, 2015. **23**(2): p. 249-277.
- [51] Branke, J., S. Nguyen, C.W. Pickardt, and M. Zhang, *Automated design of production scheduling heuristics: A review*. IEEE Transactions on Evolutionary Computation, 2015. **20**(1): p. 110-124.
- [52] Tyler, C.T. and T.L. Schmitz, *Analytical process damping stability prediction*. Journal of Manufacturing Processes, 2013. **15**(1): p. 69-76.

- [53] Tyler, C.T., J.R. Troutman, and T.L. Schmitz, *A coupled dynamics, multiple degree of freedom process damping model, Part 1: Turning*. Precision Engineering, 2016. **46**: p. 65-72.
- [54] Altintas, Y., G. Stepan, E. Budak, T. Schmitz, and Z.M. Kilic, *Chatter stability of machining operations*. Journal of Manufacturing Science and Engineering, 2020. **142**(11): p. 110801.
- [55] Karandikar, J., A. Honeycutt, T. Schmitz, and S. Smith, *Stability boundary and optimal operating parameter identification in milling using Bayesian learning*. Journal of Manufacturing Processes, 2020. **56**: p. 1252-1262.
- [56] Karandikar, J., T. McLeay, S. Turner, and T. Schmitz, *Tool wear monitoring using naive Bayes classifiers*. The International Journal of Advanced Manufacturing Technology, 2015. **77**: p. 1613-1626.
- [57] No, T., M. Gomez, R. Copenhaver, J.U. Perez, C. Tyler, and T.L. Schmitz, *Force and stability modeling for non-standard edge geometry endmills*. Journal of Manufacturing Science and Engineering, 2019. **141**(12): p. 121002.
- [58] Gomez, M.F. and T.L. Schmitz, *Displacement-based dynamometer for milling force measurement*. Procedia Manufacturing, 2019. **34**: p. 867-875.
- [59] Shi, Z., H. Ma, M. Ren, T. Wu, and A.J. Yu, *A learning-based two-stage optimization method for customer order scheduling*. Computers & Operations Research, 2021. **136**: p. 105488.
- [60] Qin, M., R. Wang, Z. Shi, L. Liu, and L. Shi, *A genetic programming-based scheduling approach for hybrid flow shop with a batch processor and waiting*

- time constraint*. IEEE Transactions on Automation Science and Engineering, 2019. **18**(1): p. 94-105.
- [61] Zhang, C., Z. Shi, Z. Huang, Y. Wu, and L. Shi, *Flow shop scheduling with a batch processor and limited buffer*. International Journal of Production Research, 2017. **55**(11): p. 3217-3233.
- [62] Ma, H., H.K. Lee, Z. Shi, and J. Li, *Workforce allocation in motorcycle transmission assembly lines: A case study on modeling, analysis, and improvement*. IEEE Robotics and Automation Letters, 2020. **5**(3): p. 4164-4171.
- [63] Ferry, W. and Y. Altintas, *Virtual five-axis flank milling of jet engine impellers—part II: feed rate optimization of five-axis flank milling*. 2008.
- [64] Beudaert, X., P.-Y. Pechard, and C. Tournier, *5-Axis tool path smoothing based on drive constraints*. International Journal of Machine Tools and Manufacture, 2011. **51**(12): p. 958-965.
- [65] Kayan, R.K. and M.S. Akturk, *A new bounding mechanism for the CNC machine scheduling problems with controllable processing times*. European journal of operational research, 2005. **167**(3): p. 624-643.
- [66] Gurel, S. and M.S. Akturk, *Considering manufacturing cost and scheduling performance on a CNC turning machine*. European journal of operational research, 2007. **177**(1): p. 325-343.
- [67] Vagnorius, Z., M. Rausand, and K. Sørby, *Determining optimal replacement time for metal cutting tools*. European Journal of Operational Research, 2010. **206**(2): p. 407-416.

- [68] Zaretalab, A., H.S. Haghighi, S. Mansour, and M.S. Sajadieh, *A mathematical model for the joint optimization of machining conditions and tool replacement policy with stochastic tool life in the milling process*. The International Journal of Advanced Manufacturing Technology, 2018. **96**: p. 2319-2339.
- [69] Schmitz, T.L. and K.S. Smith, *Machining dynamics*. Springer, 2009: p. 303.
- [70] Taylor, F.W., *On the art of cutting metals*. Vol. 23. 1906: American society of mechanical engineers.
- [71] Glover, F. and E. Woolsey, *Converting the 0-1 polynomial programming problem to a 0-1 linear program*. Operations research, 1974. **22**(1): p. 180-182.
- [72] Doi, S. and S. Kato, *Chatter vibration of lathe tools*. Transactions of the American Society of Mechanical Engineers, 1956. **78**(5): p. 1127-1133.
- [73] Tobias, S. and W. Fishwick, *Theory of regenerative machine tool chatter*. The Engineer, 1958. **205**(7): p. 199-203.
- [74] Tlustý, J. and M. Polacek. *The Stability of the Machine Tool Against Self-Excited Vibration in Machining*. in *Proceedings of the ASME Production Engineering Research Conference, Pittsburgh, PA*. 1963.
- [75] Sridhar, R., R. Hohn, and G. Long, *A stability algorithm for the general milling process: contribution to machine tool chatter research—7*. Journal of Manufacturing Science and Engineering, 1968. **90**(2): p. 330-334.
- [76] Sweeney, G. and S. Tobias, *Survey of basic machine tool chatter research*. International Journal of Machine Tool Design and Research, 1969. **9**(3): p. 217-238.

- [77] Tyler, C.T., J.R. Troutman, and T.L. Schmitz, *A coupled dynamics, multiple degree of freedom process damping model, Part 2: Milling*. Precision Engineering, 2016. **46**: p. 73-80.
- [78] Schmitz, T.L., J. Couey, E. Marsh, N. Mauntler, and D. Hughes, *Runout effects in milling: Surface finish, surface location error, and stability*. International Journal of Machine Tools and Manufacture, 2007. **47**(5): p. 841-851.
- [79] Schmitz, T.L. and R. Donalson, *Predicting high-speed machining dynamics by substructure analysis*. CIRP Annals, 2000. **49**(1): p. 303-308.
- [80] Schmitz, T.L., M.A. Davies, and M.D. Kennedy, *Tool point frequency response prediction for high-speed machining by RCSA*. Journal of Manufacturing Science and Engineering, 2001. **123**(4): p. 700-707.
- [81] Smith, S. and J. Tlusty, *Efficient simulation programs for chatter in milling*. CIRP Annals, 1993. **42**(1): p. 463-466.
- [82] Smith, S. and J. Tlusty, *An overview of modeling and simulation of the milling process*. Journal of Manufacturing Science and Engineering, 1991. **113**(2): p. 169-175.
- [83] Tlusty, J. and F. Ismail, *Basic non-linearity in machining chatter*. CIRP Annals, 1981. **30**(1): p. 299-304.
- [84] Wiercigroch, M. and E. Budak, *Sources of nonlinearities, chatter generation and suppression in metal cutting*. Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences, 2001. **359**(1781): p. 663-693.

- [85] Vaishnav, S. and K. Desai. *Machine Learning-Based Cutting Constant Estimation for Mechanistic Force Models of End Milling Operation*. in *International Manufacturing Science and Engineering Conference*. 2022. American Society of Mechanical Engineers.
- [86] Rubeo, M.A. and T.L. Schmitz, *Mechanistic force model coefficients: A comparison of linear regression and nonlinear optimization*. Precision Engineering, 2016. **45**: p. 311-321.
- [87] Kim, J.-H., Y.J. Kim, and H.-G. Park, *Mechanics-based machine learning for failure classification of load-bearing walls*. Engineering Structures, 2025. **322**: p. 119110.
- [88] Cherukuri, H., M. Selles, and T.L. Schmitz, *A neural network approach for chatter prediction in turning*. Procedia Manufacturing, 2019. **34**: p. 885-892.
- [89] Greis, N.P., M.L. Nogueira, S. Bhattacharya, C. Spooner, and T. Schmitz, *Stability modeling for chatter avoidance in self-aware machining: an application of physics-guided machine learning*. Journal of Intelligent Manufacturing, 2023. **34**(1): p. 387-413.
- [90] Schmitz, T., *Physics-informed KNN milling stability model with process damping effects*. Journal of Manufacturing Processes, 2024. **120**: p. 1124-1129.
- [91] Postel, M., B. Bugdayci, and K. Wegener, *Ensemble transfer learning for refining stability predictions in milling using experimental stability states*. The International Journal of Advanced Manufacturing Technology, 2020. **107**: p. 4123-4139.

- [92] Cornelius, A., J. Karandikar, C. Tyler, and T. Schmitz, *Process damping identification using Bayesian learning and time domain simulation*. Journal of Manufacturing Science and Engineering, 2024. **146**(8).
- [93] Schmitz, T., A. Cornelius, J. Karandikar, C. Tyler, and S. Smith, *Receptance coupling substructure analysis and chatter frequency-informed machine learning for milling stability*. CIRP Annals, 2022. **71**(1): p. 321-324.
- [94] Cornelius, A., J. Karandikar, and T. Schmitz, *Bayesian stability and force modeling for uncertain machining processes*. npj Advanced Manufacturing, 2024. **1**(1): p. 11.
- [95] Altintas, Y., *Manufacturing automation: metal cutting mechanics, machine tool vibrations, and CNC design*. 2012: Cambridge University Press.
- [96] Schmitz, T.L. and K.S. Smith, *Machining Dynamics: Frequency Response to Improved Productivity*. Springer, New York, NY, 2009: p. 303.
- [97] Shi, T., M. Ma, H. Tran, and G. Zhang, *Compressive - Sensing - Assisted Mixed Integer Optimization for Dynamical System Discovery With Highly Noisy Data*. Numerical Methods for Partial Differential Equations, 2025. **41**(1): p. e23164.
- [98] Honeycutt, A. and T. Schmitz, *Milling bifurcations for strongly asymmetric, symmetric, and weakly asymmetric system dynamics*. Precision Engineering, 2019. **55**: p. 1-13.
- [99] Altıntaş, Y. and E. Budak, *Analytical prediction of stability lobes in milling*. CIRP Annals, 1995. **44**(1): p. 357-362.

- [100] Balachandran, B., *Nonlinear dynamics of milling processes*. Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences, 2001. **359**(1781): p. 793-819.
- [101] Kondo, Y., O. Kawano, and H. Sato, *Behavior of self-excited chatter due to multiple regenerative effect*. 1981.
- [102] Shi, H. and S. Tobias, *Theory of finite amplitude machine tool instability*. International Journal of Machine Tool Design and Research, 1984. **24**(1): p. 45-69.
- [103] Zhao, M. and B. Balachandran, *Dynamics and stability of milling process*. International journal of solids and structures, 2001. **38**(10-13): p. 2233-2248.
- [104] Balachandran, B. and M. Zhao, *A mechanics based model for study of dynamics of milling operations*. Meccanica, 2000. **35**(2): p. 89-109.
- [105] Davies, M. and B. Balachandran, *Impact dynamics in milling of thin-walled structures*. Nonlinear Dynamics, 2000. **22**(4): p. 375-392.
- [106] Wahi, P. and A. Chatterjee, *Self-interrupted regenerative metal cutting in turning*. International Journal of Non-Linear Mechanics, 2008. **43**(2): p. 111-123.
- [107] Szalai, R., G. Stépán, and S. John Hogan, *Continuation of bifurcations in periodic delay-differential equations using characteristic matrices*. SIAM Journal on Scientific Computing, 2006. **28**(4): p. 1301-1317.
- [108] Szalai, R., B. Mann, and G. Stépán. *Period-two and quasiperiodic vibrations of high-speed milling*. in *Proceedings of the 9th CIRP International Workshop on Modeling of Machining Operations*. 2006.

- [109] Stépán, G., R. Szalai, and T. Insperger, *Nonlinear dynamics of high-speed milling subjected to regenerative effect*. Nonlinear dynamics of production systems, 2004: p. 111-127.
- [110] Grabec, I., *Chaos generated by the cutting process*. Physics Letters A, 1986. **117**(8): p. 384-386.
- [111] Wiercigroch, M. and A.M. Krivtsov, *Frictional chatter in orthogonal metal cutting*. Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences, 2001. **359**(1781): p. 713-738.
- [112] Wiercigroch, M. and A.-D. Cheng, *Chaotic and stochastic dynamics of orthogonal metal cutting*. Chaos, Solitons & Fractals, 1997. **8**(4): p. 715-726.
- [113] Rusinek, R., M. Wiercigroch, and P. Wahi, *Modelling of frictional chatter in metal cutting*. International Journal of Mechanical Sciences, 2014. **89**: p. 167-176.
- [114] Stepan, G., R. Szalai, B.P. Mann, P.V. Bayly, T. Insperger, J. Gradisek, and E. Govekar, *Nonlinear dynamics of high-speed milling—analyses, numerics, and experiments*. Journal of Vibration and Acoustics, 2005. **127**(2): p. 197-203.
- [115] Kalmar-Nagy, T., J.R. Pratt, M.A. Davies, and M.D. Kennedy. *Experimental and analytical investigation of the subcritical instability in metal cutting*. in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. 1999. American Society of Mechanical Engineers.

- [116] Stépán, G. and R. Szalai. *Nonlinear vibrations of highly interrupted machining*. in *Proceedings of the 2nd Workshop on Dynamics and Control of Mechanical Processing Organized by WG2 COST Action, Budapest*. 2001.
- [117] Molnár, T.G., T. Insperger, D. Bachrathy, and G. Stépán, *Extension of process damping to milling with low radial immersion*. The International Journal of Advanced Manufacturing Technology, 2017. **89**(9): p. 2545-2556.
- [118] Smith, S. and J. Tlusty, *An overview of modeling and simulation of the milling process*. 1991.
- [119] Schmitz, T.L., M.A. Davies, K. Medicus, and J. Snyder, *Improving high-speed machining material removal rates by rapid dynamic analysis*. CIRP Annals, 2001. **50**(1): p. 263-268.
- [120] Ding, Y., L. Zhu, X. Zhang, and H. Ding, *On a numerical method for simultaneous prediction of stability and surface location error in low radial immersion milling*. 2011.
- [121] Yan, Z., C. Zhang, J. Jia, B. Ma, X. Jiang, D. Wang, W. Wang, and C. Yang, *Two updated methods based on Simpson formula for chatter stability prediction in milling*. The International Journal of Advanced Manufacturing Technology, 2022. **121**(11): p. 8357-8378.
- [122] Budak, E. and Y. Altintas, *Analytical prediction of chatter stability in milling—part I: general formulation*. 1998.
- [123] Budak, E. and Y. Altintas, *Analytical prediction of chatter stability in milling—part II: application of the general formulation to common milling systems*. 1998.

- [124] Insperger, T. and G. Stépán, *Stability of the milling process*. Periodica polytechnica mechanical engineering, 2000. **44**(1): p. 47-57.
- [125] Insperger, T. and G. Stépán, *Semi - discretization method for delayed systems*. International Journal for numerical methods in engineering, 2002. **55**(5): p. 503-518.
- [126] Insperger, T. and G. Stépán, *Updated semi - discretization method for periodic delay - differential equations with discrete delay*. International journal for numerical methods in engineering, 2004. **61**(1): p. 117-141.
- [127] Insperger, T. and G. Ste' pa' n, *Vibration frequencies in high-speed milling processes or a positive answer to Davies, Pratt, Dutterer and Burns*. J. Manuf. Sci. Eng., 2004. **126**(3): p. 481-487.
- [128] Bayly, P., J. Halley, B.P. Mann, and M. Davies, *Stability of interrupted cutting by temporal finite element analysis*. J. Manuf. Sci. Eng., 2003. **125**(2): p. 220-225.
- [129] Bayly, P.V., B.P. Mann, T.L. Schmitz, D.A. Peters, G. Stepan, and T. Insperger. *Effects of radial immersion and cutting direction on chatter instability in end-milling*. in *ASME International Mechanical Engineering Congress and Exposition*. 2002.
- [130] Seguy, S., L. Arnaud, and T. Insperger, *Chatter in interrupted turning with geometrical defects: an industrial case study*. The International Journal of Advanced Manufacturing Technology, 2014. **75**(1): p. 45-56.
- [131] Parenti, P., F. Puglielli, M. Goletti, M. Annoni, and M. Monno, *An experimental investigation on Inconel 718 interrupted cutting with ceramic solid end mills*. The

- International Journal of Advanced Manufacturing Technology, 2021. **117**(7): p. 2173-2184.
- [132] Gradišek, J., E. Govekar, I. Grabec, M. Kalveram, K. Weinert, T. Insperger, and G. Stépán, *On stability prediction for low radial immersion milling*. Machine Science and Technology, 2005. **9**(1): p. 117-130.

VITA

Meiheng Ren was born in Liaoning, China. She earned a B.S. in Financial Mathematics and Statistics from University of California, Santa Barbara in 2021. In the fall of 2021, she started graduate school at the University of Tennessee, Knoxville with Dr. Tony Shi, majoring in Industrial and System Engineering. During her studies and work at UTK, her primary focus is on manufacturing operations management from supply chains, shop floors, to machine tools, i.e., from discrete-event dynamics to machining dynamics. She is expected to complete her Ph.D. degree in the fall of 2025.