Real-Time and Data-Driven Operation
Optimization and Knowledge Discovery
for an Enterprise Information System

by

Qing Duan

Department of Electrical and Computer Engineering
Duke University

Date: ____________________________
Approved:

______________________________
Krishnendu Chakrabarty, Supervisor

______________________________
Andrew Hilton

______________________________
Benjamin Lee

______________________________
Jun Yang

______________________________
Jun Zeng

Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Electrical and Computer Engineering in the Graduate School of Duke University 2014
ABSTRACT

Real-Time and Data-Driven Operation Optimization and Knowledge Discovery for an Enterprise Information System

by

Qing Duan

Department of Electrical and Computer Engineering
Duke University

Date: ____________________

Approved:

Krishnendu Chakrabarty, Supervisor

Andrew Hilton

Benjamin Lee

Jun Yang

Jun Zeng

An abstract of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Electrical and Computer Engineering in the Graduate School of Duke University 2014
Abstract

An enterprise information system (EIS) is an integrated data-applications platform characterized by diverse, heterogeneous, and distributed data sources. For many enterprises, a number of business processes still depend heavily on static rule-based methods and extensive human expertise. Enterprises are faced with the need for optimizing operation scheduling, improving resource utilization, discovering useful knowledge, and making data-driven decisions.

This thesis research is focused on real-time optimization and knowledge discovery that addresses workflow optimization, resource allocation, as well as data-driven predictions of process-execution times, order fulfillment, and enterprise service-level performance. In contrast to prior work on data analytics techniques for enterprise performance optimization, the emphasis here is on realizing scalable and real-time enterprise intelligence based on a combination of heterogeneous system simulation, combinatorial optimization, machine-learning algorithms, and statistical methods.

On-demand digital-print service is a representative enterprise requiring a powerful EIS. We use real-life data from Reischling Press, Inc. (RPI), a digit-print-service provider (PSP), to evaluate our optimization algorithms.

In order to handle the increase in volume and diversity of demands, we first present a high-performance, scalable, and real-time production scheduling algorithm for production automation based on an incremental genetic algorithm (IGA). The objective of this algorithm is to optimize the order dispatching sequence and balance
resource utilization. Compared to prior work, this solution is scalable for a high volume of orders and it provides fast scheduling solutions for orders that require complex fulfillment procedures. Experimental results highlight its potential benefit in reducing production inefficiencies and enhancing the productivity of an enterprise.

We next discuss analysis and prediction of different attributes involved in hierarchical components of an enterprise. We start from a study of the fundamental processes related to real-time prediction. Our process-execution time and process status prediction models integrate statistical methods with machine-learning algorithms. In addition to improved prediction accuracy compared to stand-alone machine-learning algorithms, it also performs a probabilistic estimation of the predicted status. An order generally consists of multiple series and parallel processes. We next introduce an order-fulfillment prediction model that combines advantages of multiple classification models by incorporating flexible decision-integration mechanisms. Experimental results show that adopting due dates recommended by the model can significantly reduce enterprise late-delivery ratio. Finally, we investigate service-level attributes that reflect the overall performance of an enterprise. We analyze and decompose time-series data into different components according to their hierarchical periodic nature, perform correlation analysis, and develop univariate prediction models for each component as well as multivariate models for correlated components. Predictions for the original time series are aggregated from the predictions of its components. In addition to a significant increase in mid-term prediction accuracy, this distributed modeling strategy also improves short-term time-series prediction accuracy.

In summary, this thesis research has led to a set of characterization, optimization, and prediction tools for an EIS to derive insightful knowledge from data and use them as guidance for production management. It is expected to provide solutions for enterprises to increase reconfigurability, accomplish more automated procedures, and obtain data-driven recommendations or effective decisions.
To my parents
## Contents

Abstract ......................................................... iv  
List of Tables ............................................... xiii  
List of Figures ............................................... xv  
List of Abbreviations ........................................ xx  
Acknowledgements ........................................... xxi  

1 Introduction .................................................. 1  
   1.1 Challenges and Opportunities for an Enterprise Information System ................... 5  
      1.1.1 Transient, Heterogeneous and Stochastic Nature ........................................ 5  
      1.1.2 Real-Time Decision Making ................................................................. 6  
      1.1.3 Diverse and Multi-Dimensional Big Data ................................................. 7  
   1.2 Introduction to Digital Print Production ....................................................... 8  
      1.2.1 Manual and Automated Rule-Based Scheduling and Resource Allocation ............. 10  
      1.2.2 Off-line Solutions .................................................................................... 11  
      1.2.3 Manual and Template-Based Order Acquisition ........................................... 12  
      1.2.4 Lack of Service-Level Forecasting and Capacity Planning ............................ 13  
   1.3 Review of State-of-the-Art .............................................................................. 14  
      1.3.1 Simulation ............................................................................................ 14  
      1.3.2 Operation Optimization ........................................................................... 14  
      1.3.3 Knowledge Discovery ............................................................................. 17  

vii
1.4 Outline of Thesis .................................................. 17
  1.4.1 Simulation ....................................................... 18
  1.4.2 Operation Optimization ....................................... 19
  1.4.3 Knowledge Discovery ......................................... 19

2 Production Simulation Platform ........................................... 21
  2.1 Background and Motivation ....................................... 21
  2.2 Introduction to Stochastic Discrete-Event Simulation .............. 22
    2.2.1 Ptolemy ...................................................... 22
  2.3 Virtual Print Factory ............................................. 23
    2.3.1 MySql Databases ............................................. 26
    2.3.2 Order, Product, and Part Hierarchy ............................ 26
    2.3.3 Resource Set and Task Set ................................... 28
    2.3.4 Successive Order Acceptance ................................ 29
    2.3.5 Stochastic Product Reprocessing .............................. 29
    2.3.6 Simulation Validation ....................................... 30

3 Production Workflow Optimization ......................................... 32
  3.1 Background and Motivation ....................................... 33
  3.2 Problem Description and Formulation .............................. 35
    3.2.1 Resources, Attributes, Parameters, and Task Sequencing Graph 35
    3.2.2 Risk-Aware Execution-Time Estimation ......................... 39
    3.2.3 Normalized Risk-Aware Slack ................................. 44
  3.3 Production Scheduler ............................................. 44
  3.4 Problem Complexity Analysis ...................................... 46
  3.5 Incremental Genetic Algorithm .................................... 48
  3.6 Dispatcher ....................................................... 49
4.4.2 Proposed Status-Prediction Method ............................. 91
4.4.3 Comparison Results and Discussions ............................ 95
4.5 Conclusion and Future Work ........................................... 97

5  Optimization of Order-Admission Policies .......................... 99

5.1 Background and Motivation ............................................. 100
  5.1.1 Related Prior Solutions ........................................... 102
  5.1.2 Costs for Service-Level Violation ............................... 103

5.2 Due-Date Validation Engine .......................................... 105
  5.2.1 Knowledge Base .................................................. 107
  5.2.2 Inputs to the Decision Engine ................................... 108
  5.2.3 Outputs of the Decision Engine ................................ 108
  5.2.4 Classifier Evaluation Metrics ................................... 109
  5.2.5 Support Vector Machines ....................................... 109
  5.2.6 Decision Tree .................................................. 114
  5.2.7 Bayesian Probabilistic Model .................................. 117
  5.2.8 Comparison of Classifiers ...................................... 120

5.3 Decision Integration .................................................... 121
  5.3.1 Dempster-Shafer theory-based decision integration approach . 121
  5.3.2 Decision fusion approach ...................................... 123
  5.3.3 Voting approach ............................................... 126
  5.3.4 Exploring new due dates ...................................... 126

5.4 Results and Discussions .............................................. 127
  5.4.1 Classifier Evaluation Strategy and Results ................. 128
  5.4.2 Factory Performance Evaluation Strategy and Results .... 130
  5.4.3 Simulation results for synthetic order data generated using real-life RPI data ........................................ 131
5.4.4 Discussion ........................................................................... 132
5.5 Conclusion .............................................................................. 137

6 Analysis and Prediction of Enterprise Service-Level Performance 138

6.1 Motivation and Prior Work ......................................................... 140
6.2 Problem Statement, Baseline Methods, and Data Source .......... 143
  6.2.1 Problem Statement ................................................................. 143
  6.2.2 Univariate Method ............................................................... 144
  6.2.3 Multivariate Method ............................................................. 146
  6.2.4 Data Source ........................................................................ 146
6.3 Mid-Term Time-Series Analysis and Prediction ......................... 148
  6.3.1 Time-Series Decomposition and Modeling ......................... 148
  6.3.2 Support Vector Regression ................................................. 152
  6.3.3 Baseline Methods ............................................................... 152
  6.3.4 Proposed Univariate Mid-Term Time-Series Prediction Method 154
  6.3.5 Results and Discussions ..................................................... 155
6.4 Multivariate Short-Term Time-Series Analysis and Prediction .... 157
  6.4.1 Time-Series Cross-Correlation Analysis ............................... 157
  6.4.2 Baseline Methods ............................................................... 158
  6.4.3 The Proposed Multivariate Short-Term Time-Series Prediction Method ..................................................... 159
  6.4.4 Results and Discussions ..................................................... 160
6.5 Conclusion .............................................................................. 161

7 Conclusion and Future Work 163

7.1 Thesis Contributions ............................................................... 163
7.2 Future Work .......................................................................... 166
  7.2.1 Topic 1: Dynamic Production Planning ............................... 167
7.2.2  Topic 2: Ranking of Attributes of Late Orders  . . . . . . . . . . 173
7.2.3  Topic 3: Conformance Checking of Production Workflow . . . 179
7.2.4  Topic 4: Worker Performance Evaluation . . . . . . . . . . . . 183
7.2.5  Summary of Thesis Proposal . . . . . . . . . . . . . . . . . . . . . 188

A Derivation of Equation (3.3)  . . . . . . . . . . . . . . . . . . . . . . 190
B Derivation of the PMF of Random Variable $X$ . . . . . . . . . . . . . 194
C Derivation of Equation (3.4)  . . . . . . . . . . . . . . . . . . . . . . 198
  C.1  Approximate the distribution of $X$ by an exponential distribution . . . 198
  C.2  The expectation of the maximum of exponentials . . . . . . . . . . . . . 198
D Introduction to SVR  . . . . . . . . . . . . . . . . . . . . . . . . . . . . 201

Bibliography  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 207
Biography  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 224
List of Tables

2.1 An example of a resource library. ........................................ 29

3.1 Mean (with standard deviation in parentheses) of average and cumulative on-time delivery ratio for (a) different simulation runs using real-life and synthetic order data; (b) varying number of dispatching GA threads for the production scheduler. ............................ 59

4.1 Confusion matrix of a status-prediction classification model. ..... 79

4.2 Features of the input vector. ............................................. 81

4.3 Relative mean error (RME) and root-mean-squared error (RMSE) of the baseline time-prediction method and the proposed time-prediction method tested through real-life EIS data. ............................... 89

4.4 Error percentage results of the two baseline status-prediction methods (Figure 4.11) and the proposed status-prediction method at some sampled predictive horizons. ........................................ 97

5.1 Attributes of the input vector. ........................................... 108

5.2 Four possible evidences of the completion status for an order. .... 122

5.3 Comparison results and tunable parameters for classifiers. ........ 129

5.4 Simulation results for classifiers over a synthetic order set. ......... 132

6.1 Time required for building the prediction model with a Windows workstation (8G memory and 2.5GHz CPU). .......................... 153

6.2 Cross correlation results between different time series. ............. 158

6.3 Relative square errors of different prediction methods for short-term (one-day-ahead) throughput prediction. .......................... 160

7.1 Mapping of Service Sets, workflow branches and class labels of Figure 7.3.171
7.2 Transformation of class labeling strategy (Figure 7.2) into structured output. .......................................................... 172

7.3 An example of the ranking of late-order attributes for an order. . . . . 178

D.1 Common kernel functions [1]. ......................................................... 205
## List of Figures

1.1 An overview of the hierarchical structure of a production enterprise [2]. 2

1.2 The overall vision of this research. Research components and key technologies are elaborated. 4

1.3 End-to-end fulfillment process of on-demand digital print. (Figure courtesy Jun Zeng [3].) 10

1.4 Examples of production templates. 13

1.5 An overview of thesis research. 18

2.1 An example of a lean manufacturing process for digital printing (Source: RPI). 24

2.2 “Virtual print factory” simulation platform architecture [3]. 25

2.3 Screenshot of the “virtual print factory” implemented on the Ptolemy simulation platform [3]. 26

2.4 An illustration of fulfillment paths. 27

2.5 An example of a task sequencing graph. 28

2.6 An example of normal incoming order flow (Source: RPI real-life data in year 2009). 30

3.1 A task sequencing graph corresponding to a single order. 36

3.2 Manufacturing process for a single-part product. 39

3.3 Manufacturing process for a multi-part product. 40

3.4 Manufacturing process for a multi-product order. 42

3.5 System architecture of the production scheduler. 45
3.6 Impact of operator weights on GA convergence performance (termination criterion is one TAKT interval). ........................................ 56
3.7 Effect of population size on GA convergence performance (termination criterion is one TAKT interval). ........................................ 57
3.8 GA convergence performance as a function of population size (p is the population size). ......................................................... 60
3.9 GA convergence performance as a function of orders in an order set (termination criterion is one TAKT interval). ...................... 60
3.10 GA convergence performance varies with the number of parallel threads. 61
3.11 GA convergence performance as a function of generations (n is the number of orders in an order set). ........................................ 61
3.12 An example of GA performance compared to ideal and practical lower bounds of the objective function. ................................. 66
3.13 VPF simulation results for a single run using real-life order data. .... 68
4.1 Illustration of the proposed process-execution time and process status prediction framework. ..................................................... 77
4.2 Event log structure. .................................................................. 80
4.3 Partitioning of data for training, testing, and validation. ............ 81
4.4 Illustration of (a) the baseline time-prediction method and (b) the proposed time-prediction method. ............................................. 82
4.5 The histogram and the probability mass function (PMF) of the relative error for the baseline time-prediction model. .................... 83
4.6 The histogram and the probability mass function (PMF) of the adjusted actual value derived from the prediction given by the baseline time-prediction model and its histogram/PMF of the relative error shown in Figure 4.5. .............................................. 85
4.7 The cumulative distribution functions (CDFs) of the adjusted actual value for different predicted values derived from the histogram/probability mass function (PMF) shown in Figure 4.6. ......................... 87
4.8 The cumulative distribution function (CDF) of the adjusted actual value for one predicted value derived from the histogram/PMF shown in Figure 4.6. ..................................................... 88
4.9 The final predicted value derived from the initial predicted value under the condition that the optimal time-prediction threshold equals 0.94. 90

4.10 Bias and variance contributing to the total error of a machine-learning algorithm. .................................................................................................................. 91

4.11 Illustration of two baseline status-prediction methods base on (a) the baseline time-prediction regression model (b) the baseline status-prediction classification model ..................................................... 92

4.12 Illustration of the proposed status-prediction method. ......................... 93

4.13 The updated cumulative distribution function (CDF) of the adjusted actual value derived from the histogram/probability mass function (PMF) shown in Figure 4.8 and the evidence that the process is in progress at time 140. ................................................................. 94

4.14 Variations of the optimal status-prediction threshold and the error rate with respect to the predictive horizon. ................................. 95

4.15 The error rate of the proposed status-prediction method and the first baseline status-prediction method with respect to the predictive horizon ((b) is a zoom-in version of (a)). ........................................ 96

5.1 An example of a production process for a two-product order. .............. 103

5.2 Illustration of the proposed order-admission framework. .................... 106

5.3 System architecture and activity diagram of the due-date validation engine. ........................................................................................................... 107

5.4 Geometric illustration of the decision boundary in an SVM. ............... 111

5.5 A illustration of a decision tree. .............................................................. 114

5.6 An illustration of the Bayesian probabilistic model. ............................ 118

5.7 Demonstration of the equivalence of the decision fusion approach to multiple voting strategies. Each tuple (n/p, n/p, n/p) represents the prediction results of classifier (A, B, C). .................................................. 125

5.8 Illustration of a phase-change point for a numerical attribute. .......... 127

5.9 Training, testing, and evaluation flow of the classifiers and the VPF. 129

5.10 Comparison results of TP and FP rates of classifiers. ....................... 130

xvii
5.11 Comparison results of missed-order percentage resulting from different classifiers and integration methods. 131
5.12 An example for the order completion-status boundary in (a) one dimension; (b) two dimensions. 135
6.1 Time-series prediction model. 145
6.2 Generation of the training data for an univariate time-series prediction method. 145
6.3 Generation of the training data for a multivariate time-series prediction method. 146
6.4 Data extracted from a large multi-national mass customization enterprise’s production event-log database. The first row shows the normalized daily counts of demand, throughput, and late delivery. The second row zooms into the region covered in the black box of the first row. 147
6.5 Time series in Figure 6.4 are decomposed to inter-year trend components. 150
6.6 Time series in Figure 6.4 are decomposed to intra-year season components. 150
6.7 Intra-season components in Figure 6.6 are decomposed to intra-year trend components. 151
6.8 Intra-season components in Figure 6.6 are decomposed to intra-week season components. 151
6.9 Hierarchical time-series aggregation strategy. 152
6.10 Predicted demand by different baseline methods. 152
6.11 Predicted time series constructed from predicted and repeated hierarchical components. 154
6.12 Predicted demand and throughput by different models. 156
6.13 Normalized demand, and 3-day offset throughput and late delivery daily count in one month from a large multi-national mass customization enterprise data. 158
6.14 Generation of the training data for the proposed multivariate time-series regression model. 159
6.15 Predicted one-day-ahead throughput data for day 1551 to day 1970. 161
7.1 An illustration of RPI’s partial production workflow. 169
7.2 An illustration of a decision point in the overall production workflow. 170
7.3 Converting a decision point to a classifier. 170
7.4 An example of a prediction path for a new service. 173
7.5 Illustration of a phase-change point for a numerical attribute. 176
7.6 Illustration of a phase-change point for a 4-category attribute. 177
7.7 An example of mined actual production workflow obtained from event log. 181
7.8 A simplified production workflow. 184
7.9 Production workflow in Figure 7.8 combined with worker information. 185
7.10 An example of the discovery of a crucial worker. 187
7.11 An example of a process parameter being influenced by a worker’s participation. 188
A.1 Manufacturing process for a 3-stage single-part product. 191
A.2 Manufacturing process for a \((N + 1)\)-stage single-part product. 192
C.1 The probability mass function derived from simulation results (left) and the probability density function approximated by an exponential distribution (right) for a 12-stage chain of sequential tasks with task execution times less than 1 time unit. 199
C.2 The probability mass function derived from simulation results (left) and the probability density function approximated by an exponential distribution (right) for a 12-stage chain of sequential tasks with task execution times no less than 30 time units. 200
D.1 The basic idea of SVM is to find a hyperplane in high-dimensional feature space that can separate instances into two classes. 202
D.2 SVR to fit a tube with radius \(\varepsilon\) to the data and positive slack variables \(\xi\) measuring the data points lying outside of the tube. 206
## List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPM</td>
<td>Bayesian probabilistic model.</td>
</tr>
<tr>
<td>DT</td>
<td>Decision tree.</td>
</tr>
<tr>
<td>EIS</td>
<td>Enterprise information system.</td>
</tr>
<tr>
<td>GA</td>
<td>Genetic algorithm.</td>
</tr>
<tr>
<td>IGA</td>
<td>Incremental genetic algorithm.</td>
</tr>
<tr>
<td>RPI</td>
<td>Reischling Press, Inc.</td>
</tr>
<tr>
<td>SVM</td>
<td>Support vector machine.</td>
</tr>
<tr>
<td>SVR</td>
<td>Support vector regression.</td>
</tr>
<tr>
<td>VPF</td>
<td>Virtual print factory.</td>
</tr>
</tbody>
</table>
Acknowledgements

My Ph.D. study at Duke University is a life-changing experience. I am most grateful to my advisor Professor Krishnendu Chakrabarty and mentor Dr. Jun Zeng who have brought me into this exciting data science field. It is my great fortune to study under their patient guidance. Their enthusiasm, endurance, and dedication have always encouraged me and become treasures in my life.

Besides, I would also like to extend my gratitude to all my dear friends for their support which has made this journey more meaningful and memorable. Special thanks go to the following individuals: my lab mates Fangming Ye, Kai Hu, Ran Wang, and Zipeng Li for all the days and nights when we studied together in the office; Ye Zhang, Hong Zhou, Julia Yao, and Lili Wang for being my lovely roommates at Durham; Yunyue Li, Yao Tong, Cheng Cheng, and Jingyi Chen for hosting me at Stanford; Wei Chen for supporting me in job search; Xingzhe Fan for being a great manager during my internship at Aspera; Ting Xu for her friendship no matter where we are; Tingzhen Ming for all his support. Last but no least, I would like to thank my parents for their unconditional love throughout these years.

Finally, I would like to thank the members of my committee for their time and effort in reading this dissertation and attending my Ph.D defense. Support received from HP Labs is greatly appreciated.
An enterprise is a business entity that comprises of individuals that share a common purpose and utilize their skills and resources to achieve specific and declared goals. Enterprise goals, organizational structures, business processes, as well as information infrastructures define the business scope of an enterprise and its relationship with the external environment [4]. An overview diagram of the hierarchical structure of a production enterprise is shown in Figure 1.1. At the highest level, a company or an affiliate is an entity that has to report its finances to the government on a timely basis. A plant can be a physical or a virtual place where goods are manufactured or traded. An enterprise may have its business presence across different locations. Its sales organization is responsible for selling goods as well as handling returns, credits, and materials at these locations. A distribution channel is the sales channel through which products are distributed, e.g., retail and wholesale. A division is purely related to the same type of product/service an enterprise is selling, e.g., hardware and software divisions. A storage location is the place where goods are stored before being sold. Each plant can be associated with multiple storage locations. It can be seen that in an enterprise, organizations or functional blocks are organized in a
In an enterprise, the fundamental activities are processes executed by resources. Parallel and series processes constitute different workflows. The completion of a service or an order requires the successful fulfillment of multiple workflows. As long as activities are occurring in an enterprise, various types of data are continuously generated and accumulated at different sites. The resulting production and service environment of an enterprise is distributed, heterogeneous, and stochastic.

It is critical for an enterprise to acquire a better understanding of the commercial context of its organizations, such as customers, the market, supply and resources, and even competitors. The enterprise information system (EIS) is designed for such purposes. An EIS is a large-scale, integrated application-software layer that uses the computational, data storage, and data transmission power of modern information technology to support business processes, information flows, reporting, and data an-
alytics within the enterprise [4,7]. Although different enterprises have their own business processes and IT operations, the EIS they adopt fundamentally share the same design principles in building its architecture [5]. An intelligent EIS is responsible for providing historical, current, and predictive views of various business operations in order to provide a high quality of service to customers. For instance, attribute characterization, online learning, real-time decision making (e.g., dynamic scheduling and resource provisioning), performance management, workflow management, and predictive analytics. In summary, without the EIS performing operation optimization and knowledge discovery, an enterprise may not be able to improve production efficiency, retain valuable customers, or perform effective market analysis [7].

Data-driven techniques enable an EIS to be self-learning, as well as adaptive to dynamic service requests and resource availability. These techniques can be integrated into the EIS to cooperatively provide enterprise-wide optimization [8]. Our goal is to develop data-driven techniques for designing, managing, and transforming an enterprise to realize a higher level of agility, performance, and operational efficiency. The key ideas behind these techniques have originated from different disciplines and has the potential to significantly improve the performance of enterprises.

This research vision and research components are highlighted in Figure 1.2, which shows an EIS that is interfaced to a pool of customers and a pool of resources. Both interfaces are dynamic since customers can start or end the business relationship with an enterprise, existing resources can become unavailable, and new resources can be added into the system. This EIS includes modules for scheduling and resource provisioning, data-driven learning, exception handling, customer social network, resource inventory and real-time decision making. An example scenario of dynamic adaptation is highlighted with the dotted line. An order from customer 2 is accepted by the enterprise. The order is then scheduled and allocated (Step 1). Certain processes of the order are allocated to Resource 2 (Step 2). However, Resource 2 becomes tem-
Figure 1.2: The overall vision of this research. Research components and key technologies are elaborated.

Porarily unavailable (e.g., due to machine breakdown, Step 3) and the order must be re-routed to Resource 1 (Step 4). All these real-time decisions taken by the EIS are hidden from the customer.

Commercial digital-print service is an example of a modern-day manufacturing-service enterprise that is primed for innovation. In the United States, the commercial print industry includes 30,000 companies and provides more than 500,000 jobs [9, 10]. Commercial print service traditionally produced homogeneous print products (e.g., thousands of copies of the same magazines) but today it is facing a paradigm
shift towards mass customization and personalization owing to the digital content explosion. This paradigm shift translates into many small-sized diverse orders that demand almost instant fulfillment. This thesis considers digital print as a sample enterprise for the evaluation of optimization methods.

The data-driven algorithms and models proposed for an EIS in this thesis will enable a transformation for enterprises by supporting dynamic integration and deployment of resources, graceful recovery from failures, scalability, and dynamic adaptation without any disruption in operations. These techniques were first developed using knowledge of the system’s mathematical model, and were subsequently and iteratively improved upon using reinforcement learning and optimization algorithms. The ability to carry out data-driven, online, and real-time optimization will have a profound impact on the quality of service provided by an enterprise.

The rest of this chapter is organized as follows. Section 1.1 presents an overview of the major challenges and opportunities for an EIS. Section 1.2 introduces digital-print service as an illustrative enterprise and discusses state-of-the-art solutions. Section 1.3 presents a literature review of the problems that have been tackled in the thesis. Finally, an outline of the thesis is presented in Section 1.4.

1.1 Challenges and Opportunities for an Enterprise Information System

In this section, we discuss challenges and opportunities that require and drive high-performance data-driven applications to be embedded into the EIS.

1.1.1 Transient, Heterogeneous and Stochastic Nature

The nature of an enterprise can be described as transient, heterogeneous, and stochastic due to the properties and the architecture of system components [11]. Heterogeneous components involve IT control and management components, error-prone
processes, repetitive manual work, diverse software and hardware resources, and so on [12]. Stochasticity is another significant feature. Stochastic trends of demand and supply, and the heterogeneous nature of customer requests all drive an EIS to implement the most adaptive data-driven strategies.

Let us take the print industry as an example. Print production activities exhibit a considerable amount of stochasticity [10]. In general, continuous order acceptance and job release activities cause the workflow in the factory to be highly variable. Products that fail during manufacturing must be rescheduled for reprocessing. If resource utilization is not well-balanced by making the workload balanced among the same type of resources, then resource overloading or idling becomes a serious problem, which can be hard to track and predict.

Therefore, an EIS must be able to handle optimization problems such as task scheduling and resource allocation, and these problems should be considered in a dynamic, transient and stochastic environment. Such challenge always requires optimization solutions to be robust enough to tackle variations rather than be effective only for the steady state. In addition to being driven by up-to-date situations, data-driven strategies must be guided by historical performance and even assisted by near-future predictions. It is essential for adaptive scheduling and resource allocation to be responsive to highly varying workloads from a large amount of customers.

1.1.2 Real-Time Decision Making

The rise of the internet helps to drive various enterprise businesses to become global business. To survive, an EIS needs to be quick and flexible, which makes response time the decisive and distinguishing factor for competitiveness. Market trends and enterprise service-level performance need to be identified and even predicted in real-time under the condition that the status of resources can be determined without any delay. Real-time decision making is needed in many production management
aspects such as modules for scheduling and resource allocation, data-driven learning from multi-dimensional correlated data, order admission and exception handling. In summary, processes that influence other processes should be optimized in real-time [13] [14].

For instance, in the print industry, print customers always expect delivery as soon as possible. Print service providers (PSPs) also compete for the ability to fulfill orders in the least amount of time. Facing such demand from customers, PSPs have been put in a tight squeeze in trying to meet the increased demand for variety with a shorter time-span to develop, produce, and deliver print products. The EIS of a PSP always needs to make real-time scheduling decisions such that acquired orders can be completed within the shortest possible time. Time-consuming scheduling methods might delay the manufacturing process of orders and also cause extra storage expense. Once an order contract has been established, the PSP should be able to achieve an optimized real-time schedule for this order. PSPs must not hesitate in responding to proposed contracts from customers or dealing with exceptional events that are affecting current manufacturing activities.

In summary, an EIS must be able to perform real-time decision-making in order to enable its business to be successful in the marketplace over a longer term [15].

1.1.3 Diverse and Multi-Dimensional Big Data

Comprehensive data warehouses, which integrate operational data from the customer, supplier, and market data, have resulted in an explosion of information for an EIS [16]. Nowadays, enterprises have valuable information stored in a number of systems and formats such as multiple relational databases, files, web pages, packaged applications, and so on [17]. In addition to large amounts of data, big data also means interactions between data from multiple sources, and even temporal correlations.

In the print industry, massive datasets are continuously being generated on the
factory floor. The EIS of a PSP has its own data warehouse in order to store all kinds of information about the PSP and its customers. For example, a large quantity of order information, such as customer information, order attributes, and order completion status, are recorded. Similarly, the enterprise event log records all major activities that take place in an enterprise, such as transactional activities, transfers of products through different departments, and exceptions. With the availability of such valuable data, the EIS is now looking for data-driven decision-making techniques that can derive insights from the available data. Data-driven approaches provide the advantage of not needing extensive expertise on production process modeling [18].

Knowledge discovery is expected to provide a high degree of integration while retaining flexibility [5]. Although data is being generated at tremendous speed, enterprises see new opportunities brought by recent advances in technologies that can help them in developing advanced solutions. Rapid improvements in computing power make it possible for an EIS to solve hard computational problems, which until recently used to be infeasible in a real-time environment. For example, the emergence of cloud computing offers enterprises creative ways to utilize IT resources [19, 20]. It allows optimization engineers to focus more on optimization objectives rather than be concerned about storage and memory limitations and insufficient computing power. Distributed computing also provides efficient solutions for an EIS to tackle hard computational problems [21, 22].

1.2 Introduction to Digital Print Production

Digital transformation of commercial print production brings new opportunities, including exploitation of embedded sensing and computing power, that enable real-time communication during various phases of the production chain and dynamic reconfiguration of the production flow [23]. Commercial print is a highly competitive service and quality-oriented business with annual retail sales of over US$700B
[9, 10]. Establishing a superior service engagement and fulfillment between clients and PSPs is key to creating value and ensuring profit for a commercial print business [24]. Figure 1.3 shows the end-to-end fulfillment process of digital print services. The starting point is the engagement with clients to obtain print services [3]. The clients, who are also the content suppliers, request print services and supply content for print. Clients range from traditional publishing agencies for newspapers, books and periodicals to web-based digital media companies, e.g., www.snapfish.com. The PSP offers its manufacturing capability to the content suppliers as a form of utility service in exchange for payment. Today, on-demand digital print service is leading the digitization of commercial print and commanding an impressive 8% annual growth rate [9]. Compared with traditional print services, which produce homogeneous print products (thousands of copies of the same content, also known as “long run-length”), on-demand digital print deals with high-frequency and high-volatility service demands [25].

In the commercial digital print industry, customers can request print products using multiple methods. They may physically visit the PSP or electronically submit print content to the PSP through the internet. An order is a contract for print service assigned to a PSP. It includes an order identifier, due date, number of copies, price and payment information, customer information, inks and presses, finishing and shipping methods. Several printing processes such as Raster Imaging Process (RIP), cover printing, page printing, folding, collating, binding, packaging and other related processing stages must be scheduled and finished before the completed order can be shipped to customers [26]. Besides offering high quality and good price, the PSP must deliver an order by its deadline since late delivery causes extra shipping cost and reduced customer satisfaction and/or loyalty.

In this section, we introduce the key features of digital printing production. Although illustrated through PSPs, these features are general and they exist in many
Figure 1.3: End-to-end fulfillment process of on-demand digital print. (Figure courtesy Jun Zeng [3].)

enterprises.

1.2.1 Manual and Automated Rule-Based Scheduling and Resource Allocation

An enterprise that provides mass-customization services has some unique characteristics [27]. From the product’s perspective, it simultaneously demands both product diversity and mass production; from an investment perspective, it is both capital-intensive and labor-intensive [10]. For instance, production scheduling on the basis of human intelligence is still the dominant practice in today’s commercial print industry [3]. Every morning, the production manager examines all the print orders that need to be fulfilled that day. Based on order attributes and corresponding production plans, the production manager then sketches the product priority and the assignment of the products to each department or resource. The production depart-
ments then operate according to such an arrangement. If an exception event occurs, the production departments report to the production manager, and then the production manager updates the production schedule on the fly. Rule-based scheduling such as first-come first-served (FCFS) and resource allocation strategy such as shortest queue are still then most-popular techniques [28].

These inefficient production solutions cannot deal with high volume and a diverse production environment. Such artisanal (or craftsman-based) and static decision-making practice have been identified as one of the key reasons that the productivity growth of the print industry as a whole is falling far behind that of other manufacturing industries [29].

1.2.2 Off-line Solutions

Scheduling problems applicable to digital print production have been tackled in operations research (OR) under the framework of job-shop scheduling. For example, mixed integer linear programming (MILP) has been applied to model such problems [30]. This approach provides optimized results; however, it takes a long time to obtain optimal solutions. Therefore, these techniques cannot be used in real-time for a high-volume production environment [31]. Similarly, process-simulation tools developed to simulate the print factory system also suffer from high computational complexity, which limit their use for real-time scenarios [3,9]. Such tools are mainly used for off-line testing of proposed solutions before they are adopted on the factory floor.

On-line scheduling solutions in operations research are based on repeatedly applying classical optimization algorithms (for instance, the aforementioned MILP) that are concerned with static input, static constraints, and static solutions. The optimization routines have an unrestricted start time when the scheduler is called. In an on-line application, this approach limits the problem size (production throughput
and complexity) because the scheduling solution must be provided within a defined time interval. In addition, exceptions in the production flow are difficult to account for in such a static optimization approach [32].

1.2.3 Manual and Template-Based Order Acquisition

Order admission is another important operation on the print factory floor. The dominant practice today is that orders are acquired by customer service representatives (CSRs) who are experienced and familiar with the history of transactions [25]. When a print buyer approaches the PSP for a potential service engagement, a CSR is assigned to this print buyer. The CSR takes the print buyer’s request i.e., job intent, consults the production manager, and then presents to the print buyer a quote of pricing and lead time. This approach is time-consuming and increasingly challenged by the trend of mass customization in print production.

A few of the industry-leading PSPs today implement an automated order ingestion process based on templatized product offering and service-level agreement [26]. For example, Reischling Press, Inc (RPI) spends several months to work with a strategic print buyer to develop a set of print products and rules to calculate life cycles of diverse products. This set of templates is developed prior to the production launch. Figure 1.4 illustrates the production templates for two different types of products. Once the production launches, the order ingestion process is automated using this set of templates. This approach provides sufficient efficiency for order ingestion but it relies on pre-determined static rules. These rules do not reflect unanticipated production events, hence they will inevitably and quickly diverge from the reality of production floor and lead to a demand-supply mismatch. Once the mismatch grows, such methods become extremely error-prone.
1.2.4 Lack of Service-Level Forecasting and Capacity Planning

The lack of service-level forecasting and capacity planning tools can greatly influence the profitability of an enterprise [15]. Service-level forecasting and capacity planning are quite challenging in commercial and industrial print service due to the lack of useful and useable predictive tools. Consequently PSPs postpone large capital equipment purchases (e.g., million-dollar printing presses) until they observe that production at the factory floor cannot sustain the agreed service-level quality. This has a severe negative business impact on customers as PSPs either have to turn orders away or be penalized by substantial late-service charges and the cost of print buyers’ goodwill [33].
1.3 Review of State-of-the-Art

1.3.1 Simulation

Enterprise simulation-based modeling can help to exploit optimization opportunities at both strategic and operational levels for an EIS [34]. It has been widely used for applications to design runtime policies. Simulation also provides performance evaluation of a system design specified by a given set of design parameters [35]. It has also been integrated with reinforcement learning to explore the significance of strategic decision making for enterprises. Simulation is very effective in helping to distinguish between different design alternatives [36].

A survey of the application of discrete-event simulation modeling to health-care clinics and systems of clinics is presented in [37]. Research on semiconductor enterprises were also studied through discrete-event simulation [38]. Applications of enterprise simulation in transportation, urban operations, supply chain management, and entertainment are described in [39].

1.3.2 Operation Optimization

Operation optimization is the process of selecting an optimal production plan and assigning limited resources for specific time periods to the set of both parallel and sequential processes for certain objectives under some practical constraints. The well-known static job-shop scheduling problems are typically NP-hard. For example, minimum makespan problems are impossible to find an optimal solution without the use of an enumerative algorithm [40].

Classical optimization techniques provide elegant static solutions to scheduling and resource management problems in an EIS [4, 41, 42]. However, these techniques are limited in their ability to handle high variability of incoming requests, dynamics of the demand and supply sides, especially when user requests are heterogeneous
in nature, and when the availability of services from a large number of resources is difficult to predict [41]. For example, enterprise workflow optimization has been studied in [43] and many approaches are suggested, but they are not adaptive to the increasing trend of data-driven decision-making patterns. Researchers proposed algorithms in [44] to address scheduling and resource allocation for heterogeneous sets of machines in an enterprise; however, the problem of complex process dependencies were not considered. Point solutions to specific scheduling and allocation problems exist in the literature, e.g., adaptive scheduling and resource allocation techniques for wireless communication networks [45, 46], but these methods tend to be more reactive than predictive, and they are not generalizable to other domains.

Therefore, without considering uncertain real-time machine workloads (machine status information, e.g., queue length, maintenance schedule, and exceptional events) and shop floor dynamics (e.g., unforeseen demand, stochastic rework, and additional or unavailable resources), process plans may become suboptimal or even invalid at the time of execution. Therefore, dynamic real-time operation optimization has to been studied to achieve adaptive solutions based on the up-to-the-moment situations [47]. Although dynamic operation optimization is even more complex, it attempts to fill the gap between scheduling theory and scheduling practice, and generates applicable solutions in real-world environments [40].

The dynamic flow optimization problem for data-center networks has gained much attention in recent years [48]. For solving dynamic flow optimization problems, the main effort is on designing scalable network structure, wherein hierarchical vertical- and horizontal- scaling patterns are explored. The optimization problem is typically modeled in terms of network flow. Heuristics algorithms, e.g., ad hoc scheduling, network flow, and meta-heuristics, e.g., simulated annealing have been used to solve this problem [49–53].

The operation-optimization problem in an EIS is more complex than in a data-
center network mainly due to the following reasons. First, a data-center network typically contains thousands of homogeneous clusters and a task can be assigned to any leaf cluster. In contrast, resources and tasks in an EIS are heterogeneous, introducing more complexity in task scheduling and resource allocation. Second, a data-center network has a tree topology with two or three levels of switches or routers. However, an enterprise production workflow usually involve multiple series or parallel processes. Third, a failed task in a data-center network can be re-assigned to any available cluster for rework, whereas an ad hoc rework policy has to be designed and executed depending on the specific production workflow.

As deterministic algorithms are not available for solving dynamic operation optimization problems, techniques based on heuristics (problem-specific scheduling methods), meta-heuristics (Tabu search, simulated annealing, and genetic algorithms), artificial intelligence (knowledge-based systems, fuzzy logic, Petri nets, and neural networks), and multi-agent systems (centralized and hierarchical models) have been studied [40]. Results indicate that a multi-agent approach that uses cooperative intelligent agents to develop global optimal solutions is more advantageous over traditional methods. Rather than exploring a standard architecture, multi-agent approaches consider specific challenges in a production environment and design of ad hoc strategies to achieve the best performance [47]. In addition, combining different techniques such as operational research and artificial intelligence in a multi-agent approach achieves real-time flexibility and robustness in a dynamic scheduling systems [40].

In summary, the major challenge for any EIS is to satisfy service-level agreements (SLAs), provide quality of service guarantees, and ensure profitability for providers. A transformative optimization approach is needed to analyze, predict, and decide in real-time such that the EIS provides utility to all parties.
1.3.3 Knowledge Discovery

Enterprise knowledge management/discovery is gaining popularity due to the increases in data quantity and dimensionality, as well as the rapid development in computing and storage techniques [54]. For instance, powerful and online analytical-processing tools in the EIS rely on data warehousing and multidimensional knowledge discovery. Capturing knowledge from different perspectives, and adaptively correlating and learning from them are the main directions that enterprises have been exploring in order to be more competitive [55].

The general goal in knowledge discovery for an EIS is to combine databases and data warehousing with algorithms from machine learning and methods from statistics to gain knowledge from data [5]. A review of applications of data mining in manufacturing enterprises, in particular production processes, operations, fault detection, maintenance, decision support, and product quality improvement is presented in [56]. Although significant achievements have been made in this area, there is still need for breakthroughs. For example, in the EIS decision-support system, research has shown the importance of using corporate data to derive and create higher-level information and knowledge, and the potential of combining different types of data [57]. Another motivation for further research is that machine-learning-based scheduling techniques in the EIS [58] are too simplistic to be generalizable to a distributed enterprise, especially since they tend to be offline solutions and they do not consider the larger problem of adaptive scheduling and policy management.

1.4 Outline of Thesis

This thesis research addresses a number of relevant operation-optimization and knowledge discovery problems with emphasis on data-driven, adaptive, and real-time functionality. The research topics cover different enterprise management issues, e.g., sys-
tem simulation, job scheduling and resource allocation, process-execution time prediction, order-fulfillment status prediction, order-admission policy optimization, and service-level performance forecasting. Designed for different optimization purposes and implemented by different techniques, these optimization components are combined to improve the efficiency of the enterprise for changing workload and resource-availability situations. Our algorithmic innovations have been tested, validated, and demonstrated using RPI’s real-life data. Developed through generalized models, these applications can be transferred to similar enterprise-software systems that need to address similar issues. The unified operation optimization and knowledge discovery framework is illustrated in Figure 1.5.

1.4.1 Simulation

Considering the heterogeneous, concurrent, and integrated characteristics of an enterprise, the first step is to adopt a holistic simulation-based approach that analyzes
the design and management of an enterprise as an integrated system. Chapter 2 presents our production simulation platform built through a discrete-event simulation tool. Section 2.2 introduces stochastic discrete-event simulation method and the tool we adopted. Section 2.3 presents the virtual factory simulation platform, which mimics enterprise production activities as shown in Figure 1.5.

1.4.2 Operation Optimization

An incremental evolutionary-algorithm based production scheduler which is used for workflow optimization is presented in Chapter 3. The background and motivation for workflow optimization is discussed in Section 3.1. Section 3.2 provides a formal problem description. Sections 3.3-3.7 describe the structure of the production scheduler and the underlying algorithms. Section 3.8 presents and analyzes experimental results. Section 3.9 concludes the chapter.

1.4.3 Knowledge Discovery

The remaining chapters present our online learning algorithms for hierarchical components in an enterprise. Using statistical analysis, data mining, and machine-learning techniques, we have designed and integrated new algorithms to improve the performance compared to state-of-the-art solutions.

In Chapter 4, we present a solution for enterprise process-execution time and process-execution status predictions. Section 4.1 discussed the background and motivation. Section 4.2 introduces the proposed prediction framework, problem statement, and the real-life data source used in the chapter. Section 4.3 describes in detail the state-of-the-art, the proposed time-prediction method, and the performance of method. Section 4.4 describes in detail the state-of-the-art, the proposed status-prediction method, and the performance of method. Finally, Section 4.5 presents conclusions and outlines ideas for further work.
Chapter 5 addresses order-admission policy optimization for enterprises. Section 5.1 discusses prior art and motivation. An order-fulfillment prediction model which includes multiple classification models has been built to estimate order life cycle. The main components and the structure of the model are discussed in Section 5.2. Section 5.3 describes decision-integration strategies for increasing the prediction accuracy of the model. In Section 5.4, we present experimental results and discuss the performance of the model. Section 5.5 concludes the chapter.

In Chapter 6, we address enterprise-wide service-level performance analysis and prediction. Section 6.1 discusses the background and motivation for enterprise service-level performance analysis and prediction. Section 6.2.4 introduces the state-of-the-art solutions and our data source. In Section 6.3, we present hierarchical time-series decomposition, the baseline model, and the proposed model for making mid-term time-series prediction. Following it, Section 6.4 presents time-series correlation analysis and the proposed multivariate short-term time-series prediction model used for short-term prediction. Section 6.5 concludes the chapter.

Finally, Chapter 7 summarizes the contributions of the thesis and identifies directions for future work.
2

Production Simulation Platform

2.1 Background and Motivation

Research on productivity improvement is gaining accelerated attention in enterprise industry in recent years. One notable example is the LDP solution by Xerox [59]. LDP is Xerox’s simulation-based service solution offered through Xerox Managed Services (XMS) that helps to enhance print shop productivity. There are distinctions between our and their simulation-based approaches. In our approach, we modeled the print production as a content-driven cyber-physical system [60]. Anticipating that the pervasive sensing and computing-based knowledge discovery will enable additional design space and flexibility, we have tested our optimization and knowledge discovery components on the simulation platform.

In collaboration with several leading digital printing companies, we have developed a production simulation platform, i.e., a system simulator that enables us to study the operational management of a digital print factory in a quantitative fashion [9] [10]. We have adapted Ptolemy II from UC Berkeley [61] and applied a statistical discrete-event modeling approach.
The rest of this chapter is organized as follows. Section 2.2 introduces discrete-event simulation and the design tool we have used. Section 2.3 presents in detail the simulation platform.

2.2 Introduction to Stochastic Discrete-Event Simulation

Discrete-event simulation refers to modeling the operation of a system as a discrete sequence of events in time [62]. Each event occurs at a particular instant in time. The state of the system changes as events continuously take place [63]. This contrasts with continuous simulation in which the simulation continuously tracks the system dynamics over time. Instead of simulating every time slice, discrete-event simulations can directly jump in time from one event to the next, therefore, they can typically run much faster than the corresponding continuous simulation. In stochastic simulation, the occurrence of an event can be stochastic and approximated by its underlying probability distribution [64]. For instance, customer arrival activities can be approximated by a Poisson distribution.

2.2.1 Ptolemy

We chose an open-source EDA toolkit, Ptolemy [61], as our modeling framework for print enterprise. Ptolemy is a Java-based, actor-oriented modeling framework for concurrent, real-time, embedded systems. Compared to object-oriented design practice, actor-oriented design emphasizes the concurrency and communication among components. Ptolemy implements a set of well-defined models of computation (for instance, continuous time, discrete event, finite state machine) that govern the component interactions. It provides a hierarchical component assembly design environment that enables the use of heterogeneous mixtures of models of computation (e.g., hybrid and mixed-signal models). It also simulates stochastic hybrid systems by adding random behavior to continuous-time models mixed with discrete events. The print
production system involves control, computation, logical and physical components, for which Ptolemy’s ability of blending different computational models provides the necessary simulation infrastructure support. In addition, compared with simulation methods as discussed in [65], we are already taking advantage of Ptolemy’s extensive customizability that only an open-source toolkit can offer.

2.3 Virtual Print Factory

It is very challenging to achieve high productivity in digital printing with non-homogeneous arrival of orders and varying customer requirements [9] [59]. The Ptolemy-based simulation platform was first developed in [9] for simulating commercial digital print operations. It monitors the resources (e.g., utilization, inventory, health) and traces the artifacts (e.g., time-stamping after each process, faulty rates). This information is recorded in the form of relational tables in MySQL as simulation outputs. The tables of simulation results serve as a basis for further analysis and discovery. Factory-level metrics such as throughput, order cycle time, quality of service and end-to-end cost can be obtained, and statistical properties can be derived, e.g., the likelihood of one process to be faulty, and the distribution of the interruptions of one machine; visualization tools can be applied to post-process the results tables and display the large amount of data intuitively. The simulated system accounts for the performance, efficiency, stability, and sustainability as organic system attributes. Its primary utilization was for understanding various lean manufacturing rules and runtime policies [66]. This platform is further used to evaluate optimization methods and decisions before their adoption on the real factory floor.

The idea of modeling document production as a manufacturing process was also described in [59]. Productivity solutions based on concepts derived from operations research were developed and deployed at several sites to accomplish substantial productivity and cost improvements for print shops [59]. The demand-driven model of
“Just-In-Time” fulfillment motivates digital PSPs to emphasize innovations in lean manufacturing productivity maximization [67] [68]. RPI is an example of such a PSP whose fulfillment procedures are centered on the concept of the “value stream” [69]. Figure 2.1 shows the RPI book factory organization.

The first part is IT services shared by all value streams, including order admission, production planning, pre-press and job release. Following the pre-press, a scheduler application evaluates order priority, decomposes orders into products, and dispatches products towards different fulfillment paths. Cover making and book block printing are the following two main streamlines. After all the products of an order have arrived at the shipping buffer, this order is assembled, packaged and shipped. The end-to-end workflow is monitored at each product level by the audit function performed by either software or wireless barcode scanners via operators. Once a part of an order fails, it is sent back to the pre-press stage for reprocessing. Every processing stage may fail following a certain stochastic probability distribution that can be derived from history data.

Figure 2.2 illustrates the architecture of the operations simulation software [23] developed as a virtual factory simulation platform based on Ptolemy [61].
2.3 shows a screenshot of the virtual factory on the Ptolemy simulation platform, in which the functional blocks corresponds to the main processes shown in Figure 2.1. The order stream is acquired from the printers’ enterprise resource management (ERM) system, which encompasses both the enterprise resource planning (ERP) system and the manufacturing execution system (MES) [3]; the fulfillment paths, resources and the operating policies are programmed in the model.

The benefit of this simulation platform is that it provides an environment that imitates the real situation on the print factory floor. Hence, we can evaluate new optimization techniques as a new scheduler application, as highlighted in Figure 2.1 and discussed in detail in Chapter 
2.3.1 MySQL Databases

MySQL, another open-source code, has been integrated as the database manager for all the input data and simulation results. Visual analytics tools interface with MySQL to display or extract both the factory level performance and cell-machine level dynamics. They also provide automated query and correlation discovery among different attributes.

2.3.2 Order, Product, and Part Hierarchy

An order is a contract established between customer and enterprise. For a PSP, the payload of an order is composed of several book titles. Each book title is referred to as a product. A product’s information contains the digital file that will be used for preprocessing and printing, quantity (number of copies) and a fulfillment path ID. There are several types of products a factory can offer. Every type of product
corresponds to its own fulfillment path, which defines a sequence of tasks or processing steps that completes the creation of a product of this type. A product may be further composed of more than one part. In case of multi-part products, the corresponding fulfillment path starts with parallel fabrication of the parts (e.g., printing the book cover and printing the book block), followed by part assembly into a product. After all products are fabricated, they can be merged to form a complete order. Based on these concepts, a processing stage can be categorized into an order-level, a product-level or a part-level stage [70].

Figure 2.4 shows an example of two simplified fulfillment paths. In the figure, we look at two different products within one order and they have fulfillment path A and fulfillment path B, respectively. Fulfillment path A has two parts, one part is used for printing cover and the second part for printing a book block. These two parts join together at the Binding stage to indicate the end of part processing. These two parts can be processed in parallel and are independent of each other, while Binding stage requires both parts to be completed before the Binding task of a product begins. Binding is therefore a product-level processing stage. The fulfillment path B processes a single-part product. The order-level stages Serializing and Packaging
are common to all products.

2.3.3 Resource Set and Task Set

The fulfillment of an order involves the processing of a sequence of steps that are serviced by different types of resources. Each resource is characterized by its functionality and speed. For example, a color printer’s printing rate is 1 second per page or a binder’s average service time is 42 seconds to bind two parts.

Based on the capabilities of each resource, the entire resource pool is partitioned into different resource sets. Resource sets for the fulfillment path example in Figure 2.4 are shown in Table 2.1. Resources within the same resource set can process the same type of tasks; these tasks compete with each other for resource allocation. We

![Figure 2.5: An example of a task sequencing graph.](image-url)
Table 2.1: An example of a resource library.

<table>
<thead>
<tr>
<th>Resource Set</th>
<th>Quantity</th>
<th>Speed (mean)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serializers</td>
<td>3</td>
<td>35</td>
</tr>
<tr>
<td>Image Processors</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>Book Printers</td>
<td>1</td>
<td>1 sec/sheet</td>
</tr>
<tr>
<td>Cover Printers</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Laminators</td>
<td>1</td>
<td>50</td>
</tr>
<tr>
<td>Cutters</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Creasers</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Binders</td>
<td>1</td>
<td>42</td>
</tr>
<tr>
<td>Packagers</td>
<td>3</td>
<td>23</td>
</tr>
</tbody>
</table>

define a task set as a set consisting of similar tasks competing with one another for the resources belonging to the same resource set. All tasks in the system can be partitioned into multiple task sets. As shown in Figure 2.5, the tasks in the same task set are interconnected by resource contention edges. There is a one-to-one correspondence between task sets and resource sets.

2.3.4 Successive Order Acceptance

In a real factory, orders are accepted in a dynamic and heterogeneous manner. The VPF uses the real-life order arrival data. The state of the VPF under a normal incoming order frequency and quantity is illustrated in Figure 2.6. The horizontal axis shows the factory time, and the vertical axis shows the number of orders accepted at 15-minute intervals.

2.3.5 Stochastic Product Reprocessing

All resources are subject to stochastic malfunctions caused by various failure modes on the factory floor. It results in reworking of some products. Due to the unrecoverable nature of physical print artifacts, no matter at which processing stage a product fails, this product is sent back to the pre-press stage in order to be reprocessed from the beginning as shown in Figure 2.1. These activities are implemented in the virtual factory as well, so that, once a product fails at any processing stage, its current man-
The manufacturing process is aborted and it is sent back to the pre-press stage immediately. If one or more products in a multi-product order require reworking, all other good products have to be stored and wait until all the reprocessed products are ready to be assembled into one order.

### 2.3.6 Simulation Validation

The accuracy of the simulation relies on the precision of component models included in the simulation. The fidelity of our VPF is measured by how closely it resembles reality, i.e., agreement with experimental measurements from real-life system provides highest confidence in the simulations.

The end-to-end fulfillment processes implemented and order data utilized in the VPF were directly pulled from RPI’s enterprise resource planning database, and integrated into the virtual factory. The results generated by the simulation platform with RPI’s internal factory audit data were validated and found to have good
correlation with the situation on the real factory floor [3].
Production Workflow Optimization

Operation optimization is a major function of an EIS. We present a high-performance and real-time production scheduler for an enterprise based on a dynamic incremental evolutionary algorithm. The optimization objective is to prioritize the dispatching sequence of orders and balance resource utilization. The scheduler is scalable for realistic problem instances and it provides solutions quickly for diverse products that require complex fulfillment procedures. Furthermore, it dynamically ingests the transient state of the enterprise, such as process information and resource failure probability in the production; therefore, it minimizes the management-production mismatch. Discrete-event simulation results show that the production scheduler leads to a higher and more stable order on-time delivery ratio compared to a rule-based heuristic. Its beneficial attributes collectively contribute to the reduction or elimination of the shortcomings that are inherent in today’s enterprise production environment and help to enhance an enterprise’s productivity and profitability.

This chapter addresses operation optimization in an EIS using digital-print industry as an example. The rest of this chapter is organized as follows. Section 3.1 discusses the background and motivation for operation optimization. Section 3.2
provides a formal problem description. Section 3.3 to 3.7 describes the structure of the production scheduler and the underlying algorithms. Section 3.8 presents and analyzes simulation results. Section 3.9 concludes this chapter.

3.1 Background and Motivation

Today’s on-demand digital printing industry is characterized by the drive for mass customization and personalization from all fronts e.g., consumer demand for personalized print products and broadly adopted targeted advertisement campaign in enterprise marketing. These trends are leading to a dramatic increase in the number of orders, yet the average size of an order is shrinking significantly [9]. In addition, print buyers are no longer willing to wait long for delivery. This scenario puts print service providers (PSPs) in a tight squeeze in trying to meet the increased demand for variety with a shorter time span to develop, produce, and deliver print products. Furthermore, print production is a “noisy” process in that there are high occurrences of exception events that are outside of the normal production planning process. Examples include rework resulting from the high failure probability of the production process and repair of the digital content due to file inconsistency or inadequate compliance [29]. In addition, precedence constraints between process steps within each order must be considered while scheduling. Precedence constraints increase the computational complexity of a scheduling problem [71].

Delivering orders on time is the primary goal of a PSP. If the fulfillment of an order takes longer than planned, the PSPs may incur several penalties. Upgrading shipping methods is one of the most common remedies for PSPs when an order is completed later than planned. This shipping cost upgrade can be significant; it can reach US$0.5M through a busy season [29]. The impact on goodwill is another type of penalty [23]. Printing is often a one step of a larger enterprise business process (e.g., a marketing campaign). If the printing step incurs significant delay compared
to the execution time promised by the PSP, it may jeopardize the enterprise’s larger business objective. Finally, delays lead to the imposition of the late penalty stated in the service contract.

This customization-responsiveness squeeze, coupled with the desire of buyers for affordable prices, forces PSPs to search for software-based automation solutions for production management in order to respond to current operational situation [10] [72] [73]. The state-of-the-art in software-based automation solutions for digital-printing production management suffers from some key limitations. Algorithms based on heuristic rules, which dominate today’s leading-edge practices, are fast but far from optimal. Techniques such as mixed-integer linear programming drawn from operational research are slow and cannot provide real-time solutions for production [30]. These techniques have largely been used for off-line strategic planning. In addition, exceptions in the production flow are difficult to account for in such a static optimization approach. The manual-intensive print production process is prone to faults, which are difficult to predict. Hence static optimization contributes to a mismatch between production management and the up-to-the-moment situation in a factory [3]. In print production today, such a mismatch between the production management and the factory reality results in different forms of inefficiency, for instance, larger inventories, larger work-in-progress (WIP) accumulated at the factory floor, more active ad hoc interventions of the production manager, longer execution time, and the need for shipping upgrades. Each of these inefficiencies has a significant negative impact on the factory’s profitability, e.g., larger inventories and work-in-progress (WIP) tighten up capital and larger execution time affects pricing and future business [9] [10].

To overcome the limitations of conventional methods, we present an evolutionary algorithm for optimization to achieve simultaneous production prioritization, task scheduling and resource allocation. The main contributions of this chapter are listed
below:

- A risk-aware order execution-time estimation model for diverse fulfillment requirements is formulated. It takes into account scheduling of orders that are yet to be dispatched, resource allocation of tasks, stochastic nature of resource malfunctions, and factory WIP workload.

- Simultaneous task scheduling and resource allocation problem is addressed in a hierarchical manner. The first (global) step is to acquire the optimal order priority and the second (local) step is to obtain detailed product priority and task resource allocation at a lower level of granularity. The computation is carried out in a distributed framework using parallel threads. Hence, it can address high volume of print orders in real-time.

- The proposed approach can handle new orders with minimal impact on the existing factory workflow. An incremental optimization strategy has been developed in the evolutionary algorithm to maximize efficiency.

- The production scheduler dynamically ingests factory process information and the resource failure probability in order to minimize the management-production mismatch.

3.2 Problem Description and Formulation

In this section, we first describe the main components of a digital print production system, their properties and inter-relationships. Following this, a risk-aware order lead-time estimation method is presented.

3.2.1 Resources, Attributes, Parameters, and Task Sequencing Graph

Each resource can be described by its processing rate and the function that the resource performs. The set $\mathcal{R}$ include all the resources in the factory. Parameter $n_\mathcal{R}$
denotes the number of different types of resources. Variable $s$ denotes the resource type, $1 \leq s \leq n_R$. The parameter $R_s$ denotes the set of resources of type $s$. The $j$th resource in $\mathcal{R}$ is denoted as $r_j$.

A task is defined as any factory operation that requires a resource. It is the atomic unit of work assigned to a resource. Tasks can be operations done by software (e.g., using computers), steps for manufacturing artifacts (require resources such as printers and binders), or shipping-related operations (require packing and shipping stations).

The fulfillment path of an order is composed of a sequence of tasks that are expanded into a series-parallel network. A task sequencing graph (TSG) is an abstract model of tasks at the system level, and it is determined by the fulfillment path and the nature of tasks. A TSG is a directed acyclic graph (DAG). Fig. 3.1 shows an example of a TSG that fulfills a single order. In a TSG $(V, A)$, each task is modeled as a vertex. The dependencies among tasks are modeled as directed arcs $\tau_{ij}$. Arc $\tau_{ij}$ indicates that task $v_i$ must be performed before $v_j$. The resource contention lines shown in Fig. 3.1 connect tasks that compete for resources in one resource set $R_s$.

An order is also an entity composed of several products and a product can further consist of parts. In the case of a multi-part product, parallel fabrication of parts is followed by assembly of parts into a product. In a similar manner, after the
fabrication of each product, several products are merged to form a complete order [70]. The order shown in Fig. 3.1 contains two products, where the first product contains two parts.

Attribute $L_i$ is the length of task $v_i$. Task length is an abstract quantity that is mapped to appropriate units based on the nature of task $v_i$. For example, a book printing task has the number of pages as its length unit. A binding task has the number of parts as its length unit.

Attribute $M_j$ is the maximum length of a task that resource $r_j$ can execute during each operation. Due to resource capacity, it cannot process a task with length greater than $M_j$ during each operation. For example, a book printer has maximum task length of a certain amount of pages, and a binder has a maximum task length of a certain amount of parts. During the establishment of the production plan of an order, each task’s length is limited by the maximum task length of the resources that can execute it.

Attribute $\beta_j$ is the intrinsic failure probability of resource $r_j$. It depends on the resource property such life span, utilization and maintenance. It is evaluated when resource $r_j$ executes a task with the maximum task length $M_j$.

Parameter $\alpha_{ij}$ is the failure probability of task $v_i$ if it is executed on resource $r_j$. It depends on the intrinsic failure probability $\beta_j$ of $r_j$ and the task length $L_i$ of $v_i$. It varies linearly with the length of $v_i$. It is obtained as: $\alpha_{ij} = \beta_j \times (L_i / M_j)$.

Parameter $\sigma_{ij}$ is the processing rate of resource $r_j$ when it executes task $v_i$. If $r_j$ cannot execute $v_i$, $\sigma_{ij}$ is not defined.

In our optimization framework, the production scheduler needs to estimate the total execution time (the time from entering the queue of a resource to exiting a resource after completion) of every task for each scheduling and resource allocation solution. We define functions and present steps in order to compute the total execution time of a task.
(1) The first step is to assign a resource to a task. Function $\Gamma(v_i)$ returns a resource that is assigned to execute task $v_i$.

(2) Once resource allocation is known, suppose that resource $r_j$ is assigned to task $v_i$, the second step is to compute the service time. Function $st_{ij}$ returns the service time of task $v_i$ on resource $r_j$. It is obtained by dividing the length of task $v_i$ by parameter $\sigma_{ij}$, i.e., $st_{ij} = (L_i/\sigma_{ij})$. The unit of $\sigma_{ij}$ depends on the unit of $L_i$. If resource $r_j$ cannot execute task $v_i$, then $st_{ij}$ is not defined.

(3) The third step is to compute the wait time of $v_i$. Since a resource executes tasks using a first-come first-served (FCFS) policy, the wait time of $v_i$ comes from the service time of tasks in front of it. There are two types of tasks that can wait in front of $v_i$ in the queue. One type is WIP tasks, i.e., tasks belonging to the orders that are currently being manufactured. The other type is scheduled tasks, i.e., tasks that are assigned to resource $r_j$ by the production scheduler and belonging to the orders that have not yet entered the production process.

In order to be executed, task $v_i$ needs to wait until resource $r_j$ finishes the execution of both WIP and scheduled tasks. Therefore, the wait time of $v_i$ consists of two components. The first component $qwt_{ij}$ is the service time of WIP tasks. The second component $swt_{ij}$ is the service time of scheduled tasks. The information of WIP tasks is provided by the resource management database, which records and updates resource information such as queue length. The value of $qwt_{ij}$ only depends on the status of resource $r_j$ when function $wt_{ij}$ is called. The value of $swt_{ij}$ depends on the number and size of scheduled tasks that have been mapped to resource $r_j$ before task $v_i$ is mapped to resource $r_j$ by function $\Gamma(v_i)$. Therefore, different scheduling and resource allocation solutions result in different values of $swt_{ij}$. Function $wt_{ij}$ returns the sum of the two components, i.e., $wt_{ij} = qwt_{ij} + swt_{ij}$.

(4) The last step is to compute the total estimated execution time, which consists of service time and wait time. Function $t_{ij}$ returns the sum of $st_{ij}$ and $wt_{ij}$, i.e.,
Manufacturing restarts from the first task

\[
\text{task } v_1 \rightarrow \ldots \rightarrow \text{task } v_i \rightarrow \ldots \rightarrow \text{task } v_N \rightarrow X
\]

FIGURE 3.2: Manufacturing process for a single-part product.

\[ t_{ij} = st_{ij} + wt_{ij}. \]

3.2.2 Risk-Aware Execution-Time Estimation

The execution time of an order, a product or a part is defined as the time to complete the manufacturing of an order, a product or a part, respectively. The most popular execution-time computation strategies are simple static rule-based methods [9]. We describe a more rigorous method for computing an order’s risk-aware execution time based on the factory up-to-the-moment status.

In the case of a single-part product, its manufacturing process involves successful completion of a sequence of tasks, shown in Fig. 3.2. The rework policy is defined such that if any task fails, the manufacturing process restarts from the first task. Each task has a specific resource assigned to it given by function \( \gamma \). Variable \( \delta_{ij} \) is a binary random variable such that \( \delta_{ij} = 1 \) if resource \( r_j \) is assigned to execute task \( v_i \), \( \delta_{ij} = 0 \) otherwise. Parameter \( w_i \) is the time required to execute task \( v_i \) on the resource to which it is mapped. It is obtained using Equation (3.1).

\[
w_i = \sum_{j=1}^{\vert \mathcal{R} \vert} \delta_{ij} \cdot t_{ij}, \quad \sum_{j=1}^{\vert \mathcal{R} \vert} \delta_{ij} = 1 \quad (3.1)
\]

Parameter \( \gamma_i \) is the failure probability of task \( v_i \) on the resource that is assigned to
execute it. It is obtained using Equation (3.2).

$$\gamma_i = \sum_{j=1}^{[R]} \delta_{ij} \cdot \alpha_{ij}, \quad \sum_{j=1}^{[R]} \delta_{ij} = 1 \quad (3.2)$$

The constraint $\sum_{j=1}^{[R]} \delta_{ij} = 1$ in Equation (3.1) and (3.2) ensures that each task can only have one resource assigned to it. Each task has a specific resource assigned to it given by function $\Gamma$. Parameter $w_i$ is the time required to execute task $v_i$ on the resource to which it is mapped. Parameter $\gamma_i$ is the failure probability of task $v_i$ on the resource that is assigned to execute it. Each task can only have one resource assigned to it. Random variable $X$ is the execution time of a single-part product. The expectation $E[X]$ of $X$ can be obtained by Equation (3.3):

$$E[X] = \sum_{i=1}^{N} \frac{w_i}{\prod_{k=i}^{N} (1 - \gamma_k)}. \quad (3.3)$$

Equation (3.3) is only applicable in the case of a chain of sequential tasks such as a single-part product. Derivation and Proof of Equation (3.3) are presented in Appendix A.

The manufacturing process for a multi-part product is shown in Fig. 3.3. Parts can be manufactured in parallel. Hence, the production process for an individual

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig3.png}
\caption{Manufacturing process for a multi-part product.}
\end{figure}
part is equivalent to a single-part product. After all the parts are manufactured, they are assembled into one product. Discrete random variable $Y$ is the execution time for completing all the parts. Therefore, $Y$ is defined as: $Y = \max_{j \in \{1, \ldots, H\}} \{W_j\}$, where $W_j$ is a discrete random variable that represents the execution time of the $j$th part. The expectation of random variable $Y$ is approximated as follows to facilitate real-time computation:

$$E[Y] \approx \max_{j \in \{1, \ldots, H\}} \{E[W_j]\} \cdot \sum_{j=1}^{H} \frac{1}{j}. \quad (3.4)$$

The expectation of $Y$ can be obtained from its probability mass function (PMF) and cumulative density function (CDF), which requires the PMF and CDF of each $W$ [74]. The computation time increases exponentially with the number of parts and the number of stages. Detailed derivation procedure is presented in Appendix B. From simulation results, we observed that $W$ can be approximated as an exponential distribution with rate $1/E[W]$. Detailed approximation procedure is presented in Appendix C. The expectation of the maximum of $H$ independent identically distributed exponential random variables with rate $1/\max\{E[W]\}$ is $\max\{E[W]\} \cdot \sum_{j=1}^{H} 1/j$ [75]. Therefore, we estimate the expectation of $Y$ as shown in Equation (3.4) for simplicity. The rework policy is defined such that if any part fails to merge into one product, then all the parts need reprocessing. We can view the set of tasks that manufacture all the parts of a product as a single task whose expected execution time equals $E[Y]$. After simplifying all the tasks for manufacturing all the parts as the first stage, we can compute the expectation of the execution time of a multi-part product by assuming that there are $N$ stages in the chain. Therefore, $E[X]$ can be obtained by the following equation:

$$E[X] = \frac{E[Y]}{\prod_{k=1}^{N} (1 - \gamma_k)} + \sum_{i=2}^{N} \frac{w_i}{\prod_{k=i}^{N} (1 - \gamma_k)}. \quad (3.5)$$

41
After substituting (3.4) into (3.5), we get:

$$E[X] = \frac{\max_{j \in \{1, \ldots, H\}} \{E[W_j]\} \sum_{j=1}^{H} \frac{1}{j}}{\prod_{k=1}^{N} (1 - \gamma_k)} + \sum_{i=2}^{N} \frac{w_i}{\prod_{k=i}^{N} (1 - \gamma_k)}.$$  \hspace{1cm} (3.6)

The manufacturing process for a multi-product order is shown in Fig. 3.4. After all the products have been manufactured, they are assembled into one product assembly. Note that, discrete random variable $Z$ is the execution time of completing all the products of an order, i.e., the execution time of the product assembly. Therefore, $Z$ is defined as: $Z = \max_{j \in \{1, \ldots, G\}} \{X_j\}$, where $X_j$ is the execution time of the $j$th product and $G$ is the number of products in a multi-product order.

The expectation of $Z$ is approximated as follows:

$$E[Z] \approx \max_{j \in \{1, \ldots, G\}} \{E[X_j]\} \cdot \sum_{j=1}^{G} \frac{1}{j}.$$  \hspace{1cm} (3.7)

The approximation process is similar to that of Equation (2). After all the products have been manufactured, remaining tasks comprise a sequence that is applied to the product assembly. The first of the remaining tasks is referred to as task $\theta$. It identifies defective products. The rework policy is defined such that flawed products are sent back for reprocessing from the beginning. In the meantime, qualified products wait at task $\theta$ for the reprocessed products to arrive.
In general, reprocessed products always have the highest priority in manufacturing. Print factory data shows that the failure probability of a task $\theta$, denoted by $\gamma_\theta$, is in the range of $10^{-4}$ to $10^{-3}$, where data has been obtained from Reischling Press, Inc. (RPI) [9].

As shown in Fig. 3.4, variable $U$ is the execution time of completing the remaining production process for an order after all the products have been manufactured. It can be calculated using the steps described for computing the execution time of a single-part product. Variable $Z$ can be obtained by Equation (3.7). Variable $T$ is the execution time of an entire order and it can be estimated using the following equation:

$$T = Z + (1 - \gamma_\theta) \cdot U + \gamma_\theta \cdot \left( \frac{\sum_{m=1}^G E[X_m]}{G} + U \right), \quad (3.8)$$

where $\sum_{m=1}^G E[X_m]/G$ is the expected time needed to reprocess the failed products. By applying the linearity property of expectation of the sum of random variables [74], we can compute the expectation of $T$ as follows:

$$E[T] = E[Z] + (1 - \gamma_\theta) \cdot E[U] + \gamma_\theta \cdot \left( \frac{\sum_{m=1}^G E[X_m]}{G} + E[U] \right). \quad (3.9)$$

In Equation (3.9), $E[Z]$ is computed using Equation (3.7), $E[U]$ is computed using Equation (3.3), and $E[X_m]$ is computed using Equation (3.3) or Equation (3.6) if the $m$th product is a single-part product or a multi-part product, respectively.

Therefore, we can obtain the expectation of the risk-aware execution time of any single or multi-product order, with either single or multi-part products by Equation (3.9).
3.2.3 Normalized Risk-Aware Slack

Every new order faces the risk of not being completed on time. This risk is incurred due to three factors: (i) task rework caused by stochastic machine malfunctions; (ii) influence from other new orders; (iii) manufacturing status of WIP orders. The probabilistic model that computes the expectation of the risk-aware execution time of an order comprehensively accounts for the main factors that are responsible for increasing the risk of late delivery.

The risk-aware execution time derived from Equation (3.9) is then used to compute the slack (the temporal difference between due time and completion time). For each order, we calculate the normalized risk-aware slack as follows:

\[ NS_i = \frac{d_i - t_j - E_i}{d_i - t_j} = 1 - \frac{E_i}{d_i - t_j}. \]

where \( E_i \) is the expectation of the risk-aware execution time of the \( i \)th order, \( d_i \) is the due time of the \( i \)th order. Variable \( t_j \) is the release time of this set of orders. It is the real time in the factory when factory releases this set of new orders for manufacturing at the \( j \)th \( TAKT \)Trigger signal (details are explained in Section 3.6). Therefore, \( d_i - t_j \) is the duration within which the \( i \)th order is expected to be completed.

3.3 Production Scheduler

Fig. 3.5 shows the system architecture and the activity diagram of the production scheduler in a print production environment. The production scheduler consists of two main components, the dispatcher and the scheduler. They cooperatively execute a two-step production scheduling algorithm.

There are notable differences as well as similarities between the dispatcher and the scheduler. After being assigned with a production plan and a due time, an
order enters the production scheduler and joins the order pool of the dispatcher, which contains orders that are yet to be dispatched for manufacturing. In the first step of creating production scheduling, the role of the dispatcher is to explore the optimal dispatching sequence for orders in the order pool. Top \( N \) orders from the optimal dispatching sequence will be sent to the scheduler. In the second step, the scheduler further determines the optimal dispatching sequence for all the products and resource allocation for every task derived from the top \( N \) orders. The value of \( N \) is dynamically determined by the changing throughput of the factory, which is further driven by the demand of customers.

This two-step incremental scheduling strategy is designed to first solve the order-level (highest granularity level) prioritization problem, then tackle the product-level (middle granularity level) prioritization and the task-level (lowest granularity level) resource allocation problem. This kind of scheduling strategy is different from tra-
ditional methods that try to achieve deterministic scheduling and/or resource allo-
cation at the lowest granularity level in one step [76]. Researchers have shown that
for many applications, partitioning of large optimization problems can lead to good
solutions in real-time [77]. For example, Fei, Meskensa and Chu [78] addressed a
planning and scheduling problem for an operating theatre in two phases and demon-
strated enhanced efficiency. Note that, in a streamlined factory, it is not necessary
to explore task-level prioritization. Only orders and products can be sorted and
dispatched according to a priority sequence at certain stages of the manufacturing
process. Tasks naturally inherit priorities from products since they are executed by
resources obeying first-come first-served (FCFS) policy in a streamlined manufac-
turing environment. Although FCFS does not necessarily result in optimal or even
superior solutions, FCFS results in fairness in terms of resources made available to
incoming orders. In addition, FCFS is easy to implement. The application of FCFS
can therefore be widely observed in practice [79].

3.4 Problem Complexity Analysis

The optimization problems tackled by the production scheduler are NP-hard. Heuris-
tic algorithms are needed for practical problem instances. It has been shown in [80]
that sequencing orders with random execution times to minimize the variances of
completion times is an NP-complete problem. For convenience, we denote this prob-
lem by $P1$. The dispatcher attempts to minimize the sum and variance of normalized
risk-aware slack times. Suppose that orders have the same due time. Our problem
is denoted by $P2$. The objective function of $P1$ is $obj_1 = Var(C(S))$, where $S$ rep-
resents an order sequence, $C_i(S)$ is the completion time of order $i$ under $S$. The
objective function of $P2$ is $obj_2 = Var(NS(S))$, where $NS_i(S)$ is the normalized
slack time of order $i$ under $S$. Completion time and slack time are related as follows:

$$NS(S) = \frac{DueTime - C(S)}{DueTime - ReleaseTime} = c_1 - c_2 C(S) \quad (3.10)$$

where $c_1$ and $c_2$ are positive constants. Therefore $Var(NS(S)) = Var(c_1 - c_2 C(S))$.

By applying these properties of variance: $Var(X + a) = Var(X)$, and $Var(aX) = a^2 Var(X)$ [74], where $a$ is a constant, we can get the transformation as $obj_2 = c \cdot obj_1$, where $c$ is a positive constant, $c = 1/c_2^2$.

For any given instance of $P1$, we construct an instance of $P2$ by multiplying $obj_1$ with the positive constant $c$. The resultant instance is one of $P2$. This construction takes polynomial time. A solution of the transformed problem instance minimizes $obj_2$, which also minimizes $obj_1$. Hence, this solution is also a valid solution for the initial problem instance. Conversely, a solution of the initial problem instance also minimizes $obj_2$, which means it is also a solution of the transformed problem instance. Therefore, we have shown that $P2$ can be reduced from $P1$ in polynomial time. The order-prioritizing problem that the dispatcher attempts to solve is therefore NP-hard.

The optimization problem of the scheduler is to find the optimal scheduling sequence for products as well as optimize resource allocation for tasks. The problem is subject to resource constraints that there are one-to-one mappings between resource sets and task sets. As discussed for the dispatcher, sequencing products is at least as hard as a known NP-complete problem. Several scheduling problems with resource constraints are discussed in [81] [82] and optimal resource allocation problems are discussed in [83]. They are at least as hard as known NP-complete problems. Our objective combines the two objectives. Therefore, based on the principle of restriction, it follows that the problem the production scheduler attempts to solve is computationally at least as hard as know NP-complete problems.
3.5 Incremental Genetic Algorithm

Incremental genetic algorithms (IGA) [84] are used in both the dispatcher and the scheduler as the main heuristics. The core algorithm, GA, is an evolutionary population-based meta-heuristic optimization strategy [85]. A chromosome is a sequence of numbers that encodes a proposed solution to the problem that the GA algorithm is trying to solve. Analogous to biological evolution process, reproduction, mutation, recombination and selection mechanisms are implemented in GA. The evolution process of a population is realized by repeated applications of these mechanisms so that the quality of the next population can be improved based on the previous population. A fitness function in GA evaluates a population of candidate solutions so that the best solution can be distinguished. After reaching a non-improving best solution or a computation time limit, as well as receiving a triggering signal that requires a solution, GA returns the best solution obtained so far. GA-based strategies have been recognized as efficient solutions for heuristically solving complex and intractable optimization problems across various domains, especially in scheduling problems [86] [87] [88] [89].

Furthermore, a general characteristic has been identified in our optimization problem. Whenever an order joins or leaves the order pool of the dispatcher, impact on the optimization problem is minor, i.e., addition or removal of one order only causes usually minor changes to the optimal dispatching sequence of other orders. Considering this characteristic, an IGA is adopted in order to reduce the time cost to reoptimize the problem after each change of the order pool. IGA is similar to classical GA except that it starts with an initial population that contains chromosomes generated based on the best chromosome obtained from solving the previous problem, which corresponds to the problem prior to a change of the order pool.

In the dispatcher, the computation is carried out in a distributed framework
using parallel threads of GAs. Each thread is referred to as a dispatching GA thread. Similarly, a GA thread in the scheduler is referred to as a scheduling GA thread.

3.6 Dispatcher

The dispatcher is a dynamic incremental optimization engine that is continuously running in the system. Its main function is to maintain a prioritized sequence of orders that are going to be dispatched for manufacturing. Every new order is first sent to the dispatcher.

Activities of the dispatcher are triggered by two external control signals. The first signal is a periodic signal called \textit{TAKTTrigger}. The \textit{TAKT} clock in Fig. 3.5 generates a new \textit{TAKTTrigger} signal with a constant time interval, called \textit{TAKT} interval. An example value of the \textit{TAKT} interval is 15 minutes, which is the actual setting used by RPI. The second signal, called \textit{IncomingOrder} signal, is generated whenever a new order is sent to the dispatcher.

Whenever there is a \textit{TAKTTrigger} signal, the dispatcher pulls the newest resource information such as failure probability and queue length from a resource database. Data are updated in real time by the production surveillance system. Therefore, parameters that the dispatcher uses are always up-to-the-moment. Updating information is crucial for minimizing the management-production mismatch. At each \textit{TAKTTrigger} signal, the dispatcher also releases a prioritized list of orders to the scheduler. Simultaneously, these orders are also removed from the order pool of the dispatcher.

There can be more than one dispatching GA threads running in the dispatcher. The order pool of the dispatcher is partitioned into many smaller \textit{buckets} according to order due time. The partition is implemented such that all the buckets have similar quantities of orders with close due times. The \textit{due time bound} of each bucket is determined by the earliest and latest due times of orders in this bucket. Each
bucket corresponds to a dispatching GA thread.

Whenever there is a new order, it is inserted into a bucket whose due time bound covers or is closest to its due time. Meanwhile, other buckets are not affected. When the loads of buckets become imbalanced to a certain degree, an automatic partitioning process applies to re-distribute orders into new buckets.

The dispatching GA in the host bucket of the new order inserts this order to its intermediate solutions by adding one gene to its chromosome. After modifying all chromosomes, the GA thread resumes running. In this way, the candidate solutions of the updated problem that contain the new order are built upon the intermediate solutions of the previous problem that do not contain the new order. The incremental strategy is used to accelerate the convergence to the new optimal solution. At the same time, only one dispatching GA is interrupted, whereas other GA threads are isolated from the changes resulting from the new order.

Order priority sequences obtained from buckets are then assembled into a global prioritized order list according to the due time bounds. The above methodology follows a distributed framework. The production scheduler as shown in Fig. 3.5 partitions a large computational problem over an extensive dataset into distributed sub-problems and then integrates results from sub-problems. It overcomes the difficulty that the original large problem cannot achieve a satisfactory solution in real time.

3.6.1 Scheduling Priority, Resource Allocation Policy, and Fitness Function in the Dispatching GA

In the dispatching GA, priority is assigned at order level. Therefore, products from the same order have no difference in terms of manufacturing sequence. Hence, for each resource, the execution sequence of tasks from different products of the same order can be arbitrary. However, tasks from different orders have a priority sequence
decided by the priority sequence of their parent orders.

GA manipulates a set of candidate solutions in each generation. A solution is encoded into a chromosome, which is a vector of genes. We utilize a random key representation technique, which allows us to realize a good chromosome representation for easy and unambiguous decoding of a chromosome into a valid solution [90]. Each gene in the chromosome is a random fractional number between 0 and 1. The genes are used for encoding the priorities of orders. If the total number of orders in a dispatching GA is \( n \), and orders are numbered from 1 to \( n \), then each chromosome can be represented as a vector \( chromosome=(gene(1), ..., gene(m)) \). The priority of order \( i \) is given by \( gene(i) \). Another advantage of such chromosome representation is that after applying simple evolutionary techniques, such as crossover, the resulting chromosome is still a valid solution.

Crossover is a technique to exchange genes between two chromosomes. We use the parameterized uniform crossover approach in this work [91]. In mutation, the remaining part of the population is randomly created in the next generation.

In the dispatching GA, decoding each chromosome determines order priority, whereas the resource allocation information is not determined. However, in order to compute wait time and service time of each task, resource allocation information must be known. Therefore, the dispatching GA requires a good approach to estimate resource allocation such that it does not deviate too much from resource allocation results given by the scheduling GA in the following computation.

An analogy to a classical optimization problem has been identified in our work. The process of computing wait time and service time for tasks that need to be processed by resources in the same resource set \( R_s, 1 \leq s \leq n_R \), is similar to the “minimum makespan” scheduling problem [92]. “Minimum makespan” scheduling problem is formulated as follows. Given a set of jobs and a set of machines, the jobs have either identical or different processing times on the given machines. The
goal is to assign jobs to machines so that the maximum completion time which is
referred to as makespan is minimized. The order in which the jobs are processed on
a particular machine does not matter. Similarly, our problem of scheduling tasks of
the same type and from the same order is formulated as follows. Given a set of tasks
from the same order that use the same set of resources, a task $v_i$ has service times $t_{ij}$
if mapped to resource $r_j$, the problem is to find an assignment of tasks to resources
such that the maximum completion time of tasks is minimized. Since tasks are from
the same order, there are no precedence constraints between them.

One policy is as follows: (i) sequence tasks arbitrarily; (ii) schedule the next task
on the resource that has been assigned the least amount of workload so far, i.e.,
the least accumulated service time for tasks in the resource queue. The minimum
makespan scheduling theorem shows that this solution is a $(2 - 1/|R_s|)$-approximation
to the optimal minimum completion time for tasks [93]. This policy is referred to as
minimum makespan scheduling and resource allocation policy.

In the dispatching GA, tasks are first sequenced by order priority and then as-
signed resources according to the minimum makespan policy. The wait time and
service time for each task is computed afterwards.

Although optimal resource allocation can only be achieved by applying task-
level priority assignment, by adopting order-level priority assignment and the min-
imum makespan policy, the dispatching GA can still achieve a certain degree of
near-optimality. Since the scheduling GA attempts to optimize the scheduling and
resource allocation at the lowest granularity level, the results of the dispatching GA
are at most $(2 - 1/|R_s|)$ times worse than the scheduling GA [93].

We have discussed that the dispatching GA adopts dynamic estimation of the
risk-aware slack. It is neither the static estimation used by rule-based heuristics
nor worst-case estimation as in our previous work [70]. It is close to the result
that can be achieved by the scheduling GA. This is important since orders and
products are eventually dispatched for manufacturing according to solutions given by the scheduling GA. We need to minimize the difference between the estimates of the dispatching GA and results of the scheduling GA.

The objective of the dispatching GA is to explore the optimal priority sequence of orders while honoring each order’s due-date commitment, as well as accounting for stochastic factors.

A two-step fitness function evaluation framework is adopted in the dispatching GA [86]. The first fitness function is to minimize the average of the normalized risk-aware slacks $\overline{NS}$ of orders, i.e., $\overline{NS} = 1/P \sum_{i=1}^{P} NS_i$. It is formulated as follows:

$$\text{minimize } f_1 = \overline{NS}$$

(3.11)

where $P$ is the number of orders in a dispatching GA. The objective of fitness function (3.11) is to meet JIT production requirement since storage of physical products on the factory floor can be expensive [94].

The second fitness function is designed to minimize the sum of the variance of normalized risk-aware slack of orders.

$$\text{minimize } f_2 = \sum (NS_i - \overline{NS})^2$$

(3.12)

The objective of fitness function (3.12) is to preserve a reasonable amount of slack for orders according to order attributes and factory status.

Each chromosome in a dispatching GA is first evaluated by fitness function $f_1$. If multiple chromosomes have the same fitness value according to $f_1$, then $f_2$ is further computed. Therefore, the best chromosome has the minimum values of both $f_1$ and $f_2$.

In addition, two candidate chromosomes that encode EDD (earliest due date) and min-slack (minimum slack time) scheduling solutions are included in the initial
population of the dispatching GA to guarantee that the solution is at least as good as what can be derived from rule-based heuristics.

3.7 Scheduler

The scheduler contains only one scheduling GA thread. It is applied over the prioritized orders sent by the dispatcher. The activation and termination of the scheduling GA are controlled by \textit{TAKTTrigger} signal. Once a \textit{TAKTTrigger} signal arrives, current scheduling GA terminates and produces the production scheduling solution. The production scheduler dispatches these orders for manufacturing. A new scheduling GA starts running over the new set of prioritized orders sent by the dispatcher.

3.7.1 Scheduling Priority, Resource Allocation Policy, and Fitness Function in the Scheduling GA

The scheduling GA gives the final production scheduling solution, in which the priority is assigned at product level. The resource allocation for each task is also given by a scheduling solution. Based on the order priority sequence given by the dispatcher, the scheduler attempts to obtain the optimal priority sequence of products and resource allocation for each task within one \textit{TAKT} interval.

A chromosome of the scheduling GA contains two types of information. The first part encodes the product priority, the second part encodes the resource allocation for every task. The scheduling GA has a thoroughly determined resource allocation information for every task encoded in each chromosome, therefore, the resource allocation policy followed by the scheduling GA is to directly decode a chromosome. Decoding product priority is similar to decoding order priority as discussed for the dispatching GA. Decoding resource allocation is done as follows. The resource allocation gene of task $v_i$ is $gene(M+i)$, where $M$ is the total number of products. If $v_i$ is mapped to resource set $R_s$, then resource $[gene(M+i) \times |R_s|]$ in $R_s$ is assigned
to execute \( v_i \). The length of a chromosome in the scheduling GA is equal to the sum of the number of products and the number of tasks from all orders. Therefore, chromosome length is much larger than it in the dispatching GA even if they contain the same set of orders.

The dispatcher has already optimized order-level priority, therefore, the main objective of the scheduler is to achieve an optimal resource utilization solution. Furthermore, the production scheduler is designed for high volume and tight due-date orders, when long queues at resources can easily become bottlenecks in the manufacturing system if an ineffective resource allocation policy is adopted. Resource utilization can be measured in terms of time. It is the total time that a resource is executing tasks \([83]\).

Fair resource utilization is achieved when in each resource set \( R_s \), the utilization of the most heavily loaded resource is close to the utilization of the least loaded resource. The fitness function of the scheduling GA is designed to minimize the sum of the normalized utilizations of the most heavily loaded resources from all the resource sets and it is formulated in Equation (3.13).

\[
\text{minimize } f_3 = \sum_{s=1}^{n_R} \frac{\max\{u(r_j) | r_j \in R_s\}}{\sum_{r_j \in R_s} u(r_j)}
\]  

(3.13)

After a chromosome in the scheduling GA is decoded, product priority and task resource allocation are all determined. Then risk-aware execution time is computed for each order. With all the information available, the scheduling GA computes the value of fitness function \( f_3 \). If multiple candidate chromosomes have the same minimal fitness value according to \( f_3 \), then the scheduling GA selects the best candidates according to the two-step fitness function framework of the dispatching GA, as discussed in Section 3.6.1.
3.8 Validation

3.8.1 Evaluation Metrics

The non-homogeneous arrival of orders is described by order arrival rate, which is the number of orders arrived during a certain interval. Parameter throughput is the number of orders delivered during a certain interval. Instantaneous workload is the number of WIP orders being processed at a specific timestamp. Average workload is the average of instantaneous workload during a certain interval. Normalized average workload is the average workload divided by the maximum average workload. Parameter on-time delivery ratio is an interval-based measure of production efficiency, which determines how efficiently a factory meets its customer’s deadlines for an interval. It is defined as the ratio of the number of orders delivered on time versus the total number of orders delivered during a certain interval. Similarly, parameter

![Graph showing the impact of operator weights on GA convergence performance.](attachment:image.jpg)

**Figure 3.6**: Impact of operator weights on GA convergence performance (termination criterion is one TAKT interval).
cumulative on-time delivery ratio is the total number of orders delivered on time versus the total number of orders delivered from time zero to a given time.

3.8.2 Simulation Settings

The production scheduler has been integrated in the VPF. The rule-based heuristic used by RPI and implemented in the VPF is min-slack for product prioritization and round-robin for task resource allocation. The min-slack approach assigns priorities to product by sequencing them according to slack time estimated based on a set of fixed values. Resources in the same resource set take turns to execute tasks by the round-robin approach. All tests and simulations were run on a 32-bit Linux operating system with 2 GB of RAM.

As shown in Fig. 2.6, we considered a real-life order-arrival data from RPI which

![Graph](image)

**Figure 3.7**: Effect of population size on GA convergence performance (termination criterion is one TAKT interval).
has a TAKT interval equals 15 minutes. There are 12 different product types, the number of tasks in a product ranged from two to 14, the number of products in an order ranged from one to 400, the number of resource set is 22, and the average number of tasks in an order is 90.

3.8.3 GA Configuration and Convergence Performance

This subsection discusses methods used to derive the configuration of the GA. We adopted the weights in constructing the next generation for reproduction, mutation, and crossover as 30%, 20%, and 50%, respectively, and a population size of 100 since they were found to provide the best convergence performance.

In the following, we discuss how to determine the relationship between parameters such that an acceptable local optimal solution for each GA run can be achieved within one TAKT interval.

Reproduction and mutation are the key processes that increase the probability of convergence to a globally optimal solution [95] [96]. However, it still remains an open question as to what values of GA parameters (such as population size, choice of GA operators, operator weights, and others) to use for a specific problem. Based on the studies in [97], it is recommended that crossover is the key operator that should be used for generating most chromosomes in the next generation.

Since GA is a stochastic optimization process, even over the same order set, GA runs seldom converge to the same solution. We therefore consider results that are averaged over 15 runs. In our production scheduler application, the termination criterion is one TAKT interval, i.e., each GA run lasts for 15 minutes. We used RPI real-life orders for testing the convergence performance.

Simulation results presented in Fig. 3.6 show that as the mutation ratio decreases, variance of the fitness value also decreases. A high reproduction ratio increases the likelihood of early convergence. A relatively low reproduction ratio decreases the
Table 3.1: Mean (with standard deviation in parentheses) of average and cumulative on-time delivery ratio for (a) different simulation runs using real-life and synthetic order data; (b) varying number of dispatching GA threads for the production scheduler.

(a)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Performance metrics</th>
<th>Run 1</th>
<th>Run 2</th>
<th>Run 3</th>
<th>Run 4</th>
<th>Run 5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>on-time delivery ratio</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>production</td>
<td>average</td>
<td>0.87336</td>
<td>0.89095</td>
<td>0.89457</td>
<td>0.88276</td>
<td>0.88806</td>
</tr>
<tr>
<td>scheduler</td>
<td>(0.02004)</td>
<td>(0.00650)</td>
<td>(0.00623)</td>
<td>(0.01322)</td>
<td>(0.00977)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>cumulative</td>
<td>0.88404</td>
<td>0.89037</td>
<td>0.89002</td>
<td>0.88715</td>
<td>0.88864</td>
</tr>
<tr>
<td></td>
<td>(0.00909)</td>
<td>(0.00654)</td>
<td>(0.00790)</td>
<td>(0.00802)</td>
<td>(0.00774)</td>
<td></td>
</tr>
<tr>
<td>min-slack</td>
<td>average</td>
<td>0.72968</td>
<td>0.74717</td>
<td>0.74715</td>
<td>0.73842</td>
<td>0.74279</td>
</tr>
<tr>
<td>heuristic</td>
<td>(0.10505)</td>
<td>(0.11592)</td>
<td>(0.11594)</td>
<td>(0.11048)</td>
<td>(0.11321)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>cumulative</td>
<td>0.75274</td>
<td>0.74986</td>
<td>0.75033</td>
<td>0.75138</td>
<td>0.75078</td>
</tr>
<tr>
<td></td>
<td>(0.01158)</td>
<td>(0.01460)</td>
<td>(0.01778)</td>
<td>(0.01362)</td>
<td>(0.01517)</td>
<td></td>
</tr>
</tbody>
</table>

(b)

<table>
<thead>
<tr>
<th>Performance metrics</th>
<th>1 thread</th>
<th>2 threads</th>
<th>3 threads</th>
<th>4 threads</th>
<th>5 threads</th>
</tr>
</thead>
<tbody>
<tr>
<td>average on-time delivery ratio</td>
<td>0.87071</td>
<td>0.87336</td>
<td>0.86477</td>
<td>0.77777</td>
<td>0.75154</td>
</tr>
<tr>
<td>(0.02745)</td>
<td>(0.02004)</td>
<td>(0.02885)</td>
<td>(0.06933)</td>
<td>(0.09845)</td>
<td></td>
</tr>
<tr>
<td>average on-time delivery ratio</td>
<td>0.88176</td>
<td>0.88404</td>
<td>0.87506</td>
<td>0.78264</td>
<td>0.76087</td>
</tr>
<tr>
<td>(0.01533)</td>
<td>(0.00909)</td>
<td>(0.01279)</td>
<td>(0.00778)</td>
<td>(0.01226)</td>
<td></td>
</tr>
<tr>
<td>cumulative on-time delivery ratio</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cumulative on-time delivery ratio</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We next optimize the population size. Studies have shown that satisfactory results can be obtained with a small population size, i.e., the population size does not have to increase exponentially with the complexity of the problem [98]. The complexity of our problem increases exponentially with the number of orders.

Simulation results presented in Fig. 3.7 show that the time to complete one generation increases linearly with the size of the population. The larger the population size, the fewer the generations during one TAKT interval. A small population size causes premature convergence even though it has the most number of generations. Fewer generations limit the convergence performance for a larger population size. In addition, a long duration for each generation should be avoided if new orders arrive frequently since every new order interrupts one dispatching GA thread. Furthermore, no clear dependency between order set size and population size has been found. Fig. 3.8 shows that for an order set of 100 orders, GA convergence performance does not clearly depend on the population size.
Figure 3.8: GA convergence performance as a function of population size ($p$ is the population size).

Figure 3.9: GA convergence performance as a function of orders in an order set (termination criterion is one TAKT interval).
**Figure 3.10:** GA convergence performance varies with the number of parallel threads.

**Figure 3.11:** GA convergence performance as a function of generations ($n$ is the number of orders in an order set).
After setting the weights for operators, and the population size, we investigate how the convergence performance varies with order sets of different sizes. Simulation results presented in Fig. 3.9 shows that the time for each generation increases linearly with the number of orders. Therefore, the average number of generations during the same TAKT interval decreases with the number of orders, whereas the variance in fitness value increases. Smaller the order set, the lower is the fitness value. This can be attributed to two factors. One reason is that the slack time allocated to each order in a smaller order set is more than it is in a larger order set. The second reason is that more generations are allowed for searching an optimal solution during the same interval.

We next evaluate the computational cost (time per generation) of parallel GAs. Population size is set to 100 regardless of order set size. Simulation results presented in Fig. 3.10 show that, since parallel threads are introduced to handle portions of a big order set, as the number of threads increases, computational cost decreases as well. The decrease rate reaches the highest from a single thread to two parallel threads.

A brute-force method that exhaustively computes the fitness value for each order priority sequence is feasible only when an order set has no more than eight orders. The GA can find the same optimal solution as the brute-force method. For larger problem instances, as shown in Fig. 3.11, GA converges to at least locally optimal solutions after sufficient numbers of generations.

### 3.8.4 ILP Model for GA Performance Evaluation

In order to evaluate the quality of the results obtained by the GA algorithm in terms of how close it is to globally optimized results, we have developed an integer linear programming (ILP) model that carries out global optimization for the same objective function (9). Notations used in the ILP model are the same as defined for the GA
algorithm in Section II. Details of the ILP model are listed below.

1. Constants:

   - Array $PT = [PT_i]$, where $i$ denotes task $i$ and $PT_i$ is the set which contains task $i$’s parent tasks. For the first task of any order, $PT_i$ is empty.

   - Array $ES = [ES_i]$, where $ES_i$ is the earliest possible start time of task $i$. The value of $ES_i$ for the first task of an order is the manufacturing release time for the order set. For other tasks, the value of $ES_i$ is calculated under the ideal scenario when sufficient resources are available, i.e., there are no WIP tasks and no resource contention for tasks of the same type.

2. Variables:

Variables in the ILP model are either binary or non-binary variables. They are listed below:

\[ s_i : \text{start time of task } i \text{ (integer variable)} \]

\[ e_i : \text{end time of task } i \text{ (integer variable)} \]

\[ r_{ij} = \begin{cases} 
1 & \text{if task } i \text{ is assigned to resource } j, \\
0 & \text{otherwise.} 
\end{cases} \]

\[ \alpha_{ik} = \begin{cases} 
1 & \text{if } (s_i \leq s_k), \text{ i.e., task } i \text{ starts no later than task } k, \\
0 & \text{otherwise.} 
\end{cases} \]

\[ \beta_{ik} = \begin{cases} 
1 & \text{if } (e_i \geq s_k), \text{ i.e., task } i \text{ ends no earlier than task } k, \\
0 & \text{otherwise.} 
\end{cases} \]

\[ \delta_{ik} = \begin{cases} 
1 & \text{if task } i \text{ overlaps with task } k \text{ and task } i \text{ ends no later than task } k \\
0 & \text{otherwise.} 
\end{cases} \]
3. Constraints:

Constraints in the ILP model include both linear and nonlinear constraints. Linear constraints are as follows:

\[
\sum_{j=1}^{J} r_{ij} = 1, \tag{3.14}
\]

\[
e_i - sw_i \geq e_k, \quad \forall k \in PT_i \tag{3.15}
\]

\[
e_i - sw_i \geq ES_i. \tag{3.16}
\]

Constraint (12) ensures that each task has a resource assigned to it. Constraint (13)(14) ensure that each task cannot start earlier than its parent tasks or its earliest possible start time. The nonlinear constraints are:

\[
\delta_{ik} = \alpha_{ik} \times \beta_{ik} \tag{3.17}
\]

\[
(s_i - s_k)(1 - \alpha_{ik}) + \alpha_{ik} \geq 1 \tag{3.18}
\]

\[
\alpha_{ik}(s_i - s_k) \leq 0 \tag{3.19}
\]

\[
(s_k - e_i)(1 - \beta_{ik}) + \beta_{ik} \geq 1 \tag{3.20}
\]

\[
\beta_{ik}(s_k - e_i) \leq 0 \tag{3.21}
\]

\[
\delta_{ik}(1 - r_{ij} - r_{kj}) \geq 0 \tag{3.22}
\]

Constraint (15) models tasks that overlap with each other. Constraint (16)(17) force binary variable \(\alpha_{ik}\) to be 1 when task \(i\) starts no later than task \(k\). Constraint (18)(19) force binary variable \(\beta_{ik}\) to be 1 when task \(i\) finishes no earlier than task \(k\). Constraint (20) guarantees that under the situation when task \(i\) and task \(k\) have a time conflict and the possibility of using the same resource, they cannot be assigned to the same resource at the same time.

In order to linearize constraint (15), new binary variable \(\gamma_{ij}\) is introduced to replace variable \(\delta_{ij}\) by standard techniques [99]. The following two constraints are
also added to replace constraint (15) for all tasks.

\[ \alpha_{ij} + \beta_{ij} \leq \gamma_{ij} + 1, \quad (3.23) \]

\[ \alpha_{ij} + \beta_{ij} \geq 2 \times \gamma_{ij}. \quad (3.24) \]

Similar techniques are applied to replace product terms involving two variables.

4. Mapping between ILP and GA:

In the ILP model, there is no estimated execution time (introduced in Section II-A). The estimated execution time is calculated as follows:

\[ w_i = ES_i - s_i + st_{ij} \quad (3.25) \]

Therefore, in the ILP model the estimated execution time is derived from the estimated earliest start time minus the start time (the subtraction is the estimated wait time), plus the service time. Once the estimated execution time is obtained, the normalized risk-aware slack time can be estimated as described in Section II.

5. Optimality and lower-bound analysis:

The optimal order priority sequence for up to eight orders are the same for the ILP model and the GA algorithm. Due to the large number of constraints, the computation time exceeds 24 hours when more orders are considered.

For more than eight orders, we consider ideal and practical lower bounds of the objective function. The ideal lower bound can be obtained under the ideal scenario when sufficient resources are available, i.e., no resource contention. In the ideal scenario, since every order can be scheduled in the desired just-in-time manner, the value of the objective function is zero. The practical lower bound is the best result the GA algorithm can obtain without computation-time limitation. 48 runs of GA were conducted with different combinations of parameter settings. Crossover varies among 40%, 50%, and 60%. The ratio of reproduction and mutation varies among 1:4, 2:3, 3:2, and 4:1. The population size varies among 50, 100, 150, and 200. Instead of being terminated within one TAKT interval, GAs were terminated when
there was no improvement in the fitness value over 1000 consecutive generations. The best result of those 48 GAs was referred to as the practical lower bound. Note that, the value of the practical lower bound varies with each problem instance.

Furthermore, as discussed in Section III-D, two candidate chromosomes that encode EDD and min-slack scheduling solutions are generated in the initial population of the GA algorithm to guarantee that the solution is at least as good as what can be obtained from rule-based heuristics.

As discussed in the beginning of Section IV-D, we run the performance test based on the optimal configuration of the GA algorithm. Figure 3.12 shows simulation results obtained from 15 runs for the same order set which contains 100 orders (the maximum number of orders each GA thread is configured to handle).

Finding optimal solutions to our problem is not practical due to the problem complexity, e.g., the ILP model for large problem instances takes vast amount of computation time. The ILP model for small problem instances shows that the GA
algorithm can obtain optimal results for small problem instances. The lower-bound performance analysis for large problem instances shows that although the GA algorithm cannot obtain the result as good as the ideal lower bound, its performance is superior to rule-based heuristics. In addition, if more computation time is allowed, GA can obtain even better performance.

3.8.5 Production Scheduler Configuration

Multiple GA threads are run in parallel, which contains one scheduling GA and the others are dispatching GAs. Fig. 3.13 presents results for two parallel dispatching GA threads. Comparison results for different number of dispatching GAs are presented in Table I(b). All the evaluation metrics are evaluated over an interval of six hours. Intervals ranging from one to twelve hours show similar results. A single-run simulation ranges from seven to 10 days in terms of factory time. Order due dates are tight. For loose due dates, any scheduling method can be used.

3.8.6 Results and Discussions

1) Results from a single run of real-life RPI data

We consider factory optimization for a real-life order pool provided by RPI. The performance of the production scheduler and the rule-based heuristic are compared. Fig. 3.13(a) shows the normalized average workload. Fig. 3.13(b) shows the average on-time delivery ratio. Fig. 3.13(c) shows cumulative on-time delivery ratio, where the interval is from the start of factory operation to the current time. Fig. 3.13(d) shows the average on-time delivery ratio versus average throughput. The results are listed in Table I(a) ("Run 1").

Based on the real-life order data from RPI, the following conclusions can be drawn from simulation results:

(i) Fig. 3.13(a) shows that factory workloads resulting from the two scheduling algorithms are different. This is because of the difference in order dispatching mechanisms of the two scheduling algorithms. Rule-based heuristics instantly generate a schedule
Figure 3.13: VPF simulation results for a single run using real-life order data.
for an order, which gives the exact time when this order should be dispatched for manufacturing. The production scheduler however, does not dispatch an order right after its arrival. As discussed in Section 3.3, it takes at least two TAKT intervals before an order can be dispatched. Fig. 3.13(b) shows that the proposed schedule provides a higher on-time delivery ratio than the rule-based heuristic used by RPI. Moreover, it provides an on-time delivery ratio with less variance. In contrast, the performance of the rule-based heuristic is highly variable (and hence unpredictable).

(ii) Fig. 3.13(b)-(c) show that the production scheduler can stabilize and improve the on-time delivery ratio significantly compared to rule-based heuristics. Furthermore, Fig. 3.13(c) shows that the cumulative on-time delivery ratio of the production scheduler is always higher and more stable than that for rule-based heuristics. This is because the production scheduler is a global scheduler. It generates a schedule for an order while considering the influence of scheduling other orders as well as current factory status; important considerations that are ignored by the rule-based heuristic currently deployed in practice.

(iii) Fig. 3.13(a)-(b) show that, as expected, an increase (decrease) in workload always leads to a decrease (increase) in on-time delivery ratio regardless of scheduling algorithms. This is because a heavier workload of the factory increases the amount of tasks accumulated in resource queues, which subsequently increases the average execution time of an order. The likelihood of an order missing its due date is proportional to the factory workload during the period when it was being processed. Note however that the production scheduler is able to handle high workloads more efficiently.

(iv) Fig. 3.13(d) shows that for the same factory throughput, the on-time delivery ratio resulting from the production scheduler is always larger than or comparable to that for the rule-based heuristics. The variation of on-time delivery ratio of the production scheduler is also less. When the same number of orders have been
delivered during the same interval, the production scheduler results in fewer late orders in spite of the computation time required by GAs. These results demonstrate the higher scheduling efficiency of the production scheduler.

2) Results from other runs

We ran the VPF using the same RPI data two additional times. Compared to the first run, the amount of rework is different because of the stochastic nature of fault occurrences. Simulation results for the second and third run of the real-life RPI data are included in Table I(a).

We deployed the production scheduler for the fourth and fifth run on data sets that have been generated based on the real-life RPI data. We randomly select pairs of orders and randomly exchange some attributes except for due dates. Simulation results are also shown in Table I(a) (“Run 4” and “Run 5”).

3) Parallel performance

Simulation results for parallel dispatching GAs on the RPI order data are presented in Table I(b). The following conclusions can be drawn from simulation results: (i) Two parallel dispatching GA threads are more effective then a single thread, as well as more than two threads. Two parallel threads leads to higher on-time delivery ratio and less variance in the results. As the number of orders to be processed during one TAKT interval increases, the probability that a single thread cannot converge to an effective solution increases. Two parallel threads partition the order set, therefore, each of them has a higher probability of converging to an optimal solution over the half of the original problem. However, note that orders are partitioned according to order due date. When there are more than two threads, the assembled solution from each thread can no longer capture other factors that influence order priorities. (ii) The optimal number of dispatching GAs depends on problem attributes such as order pool size, due date, and order attribute diversity. The book-keeping and merging overhead resulting from an unnecessarily large number of threads offsets the
benefit gained from parallelism. In terms of RPI’s dataset, two parallel dispatching GA threads is the most appropriate setting for the production scheduler. A theoretical characterization of the optimum number of threads in general cases needs more study and it is left for future work.

3.9 Conclusion

We have presented a production scheduler based on an IGA. The production scheduler is used to provide an effective dispatching sequence of orders and products, and simultaneously assign resources to tasks. It adopts an distributed computing framework using parallel threads. Each thread provides a partial solution and the global solution is integrated from partial solutions. The production scheduler also dynamically ingests the up-to-the-moment factory information to update parameters used for computation. We have implemented the production scheduler in a virtual print factory, and compared it to a heuristic method that is currently deployed in practice. We have shown that the production scheduler clearly outperforms the rule-based heuristic for a real-life order set and synthetic order set.

In future work, we will further investigate the stochastic nature of the optimization problem. We will study machine-learning techniques to identify outliers, tolerate noise in the factory and order data, and provide predictions with probabilities that can be incorporated in decision-making for scheduling and resource allocation.
Predictions of Process-Execution Time and Process-Execution Status

As shown in Chapter 3, process-execution time is a fundamental measure in an EIS. Our risk-aware execution-time estimation method (Section 3.2.2) has demonstrated improved performance over static rule-based methods. However, in addition to performing real-time production scheduling, an EIS should also be able to carry out planning for the future. Therefore, accurate predictions of both process-execution time and process status are crucial for the development of an intelligent EIS. We propose new process-execution time-prediction and process status-prediction methods for an EIS. Process-execution time prediction is a regression problem and state-of-the-art (baseline) time-prediction methods use a machine-learning regression model. Process status prediction is a binary classification problem in which a class labeled “completed” or “in-progress” is assigned to a process with respect to an arbitrary predictive horizon (i.e., the future time given by the method user). A state-of-the-art (baseline) status-prediction method compares the predicted process-execution time with the predictive horizon to check whether the process is still in progress. Another baseline method builds individual status-prediction classifiers for all possible
predictive horizons. The methods proposed in this chapter are developed on the basis of the baseline time-prediction methods by integrating statistical methods with machine-learning algorithms. Comparison results obtained from the real-life data of a digital-print enterprise show that the proposed time-prediction method reduces both the relative mean error and the root-mean-squared error of the regression model. Furthermore, the proposed status-prediction method not only achieves higher classification accuracy than both state-of-the-art methods, it also estimates the probability of the predicted status. In addition, algorithm development and training phases of the proposed methods do not rely on any arbitrary predictive horizon. Therefore, a single time-prediction model as proposed is sufficient for status prediction as opposed to a baseline status-prediction method that requires classification models for all potential predictive horizons.

The rest of this chapter is organized as follows. Section 4.1 discussed the background and motivation. Section 4.2 introduces the proposed prediction framework, problem statement, and the real-life data source used in this chapter. Section 4.3 describes in detail the state-of-the-art, the proposed time-prediction method, and the performance of method. Section 4.4 describes in detail the state-of-the-art, the proposed status-prediction method, and the performance of method. Finally, Section 4.5 presents conclusions and outlines ideas for further work. Since SVR is the key solution technique used in this chapter, an introduction to it is provided in Appendix D.

4.1 Background and Motivation

An enterprise offers diverse, and often on-demand, customized services. Each service can involve multiple processes, which may be parallel, in series, or in any combination thereof. A process is defined as any enterprise operation that requires a resource. Process-execution time is defined as the time spent by the enterprise executing that
process. The status of a process can be either “completed” or “in-progress” with respect to an arbitrary predictive horizon (i.e., the future time of interest) [100].

An EIS is responsible for recording, monitoring, and predicting the attributes related to each process. Process-execution time and process status serve as the basis for a number of performance measures in an EIS [7]. For instance, they are very informative for managers for making decisions or planning schedules. Process-execution time and process status can be used in production planning, system design, operational control, and evaluation. With precise time and status prediction, an EIS can suggest optimal resource allocation for processes and estimate resource availability, allowing managers to make higher-level management decisions based on predicted process-execution times for lower-level processes [33]. Process-execution time prediction can be used to schedule many other activities, such as maintenance interruptions. Being able to predict the time and status of a process in real time is critical for service and process-flow management in an EIS [41].

If we take the digital print industry as an example, a use case can be as follows. During the morning shift, the floor manager looks at the queue of a web-press. Suppose there are 14 jobs in the queue. The manager might want to know how many of them will be left for the afternoon shift. Another use case can be a process-status dashboard that periodically predicts which processes in the queue can be completed within the next hour [26].

Process-execution time and process status are influenced by many features. For instance, a resource malfunction event delays scheduled processes. An older-version resource may take more time to execute a process than a newer version of the resource. Process workload also affects execution time. Related processes, e.g., processes that share the same resources, may influence the execution times of one another. These dynamic features make time and status predictions very complex, making it difficult to achieve high accuracy [101]. Furthermore, features for making predictions are
selected through several iterations of data selection, pre-processing, model building, and model analysis.

Time prediction has been studied in many EISs. A common traditional approach is based on the multiple-factor linear combination (MFLC) method [102]. This solution is extremely inaccurate when the attributes of a process are dynamic, e.g., when manual work is involved in a process. It is obvious that it cannot be adopted in transient, heterogeneous, and stochastic enterprise environments. A statistical modeling method that considers uncertain factors and nonlinear relationships between features is a significant improvement over the MFLC method [102]. Although this solution does not require a clear understanding of the underlying physical processes, assumptions made for building the model must be able to depict the actual situation as clearly as possible.

Production simulation is another advanced solution for process-execution time and process status predictions [103]. However, this solution requires a clear understanding of the underlying physical processes. Developing models for simulation can be expensive and the execution can be lengthy if the predictive horizon is long [104]. Therefore, production simulation for execution-time prediction is not sufficiently general or flexible for most EISs [36].

Recently, data-driven methods have become mainstream [105]. An advantage of data-driven approaches is that they do not require extensive expertise in process modeling. These data-driven methods correlate historical, current, and future data, without explicitly addressing the physical processes in the way model-based approaches do. There are many successful applications of time and status predictions, for instance, vehicle-travel time prediction based on support vector regression (SVR) [106], software execution time and reliability prediction based on SVR [107], aircraft taxi time prediction based on different machine-learning algorithms [108], and tumor status prediction based on SVR [109].
Recent work has therefore shown that baseline process-execution time prediction can be realized by using a machine-learning regression model. However, the prediction performance fluctuates and even deteriorates as predictions become more complex. Errors accumulate for predictions made using other predicted values as input information. We consider two start-of-the-art status-prediction methods as baselines in this work. The first baseline status-prediction method compares the predicted process-execution time with the predictive horizon to check whether the process is still in progress. A drawback is that it does not differentiate potential impact of the predictive horizon. Intuitively, the larger the predictive horizon, the higher the probability it is that a process can be completed by that predictive horizon. The second baseline status-prediction method builds individual status-prediction classifiers for all possible predictive horizons. The disadvantage with this approach is that it needs to know the predictive horizons in advance in order to train each classifier [101].

In this chapter, we design a time-prediction method and a status-prediction method. The proposed methods provide higher prediction accuracy with only one regression model.

The main contributions of this chapter are as follows.

- By integrating statistical analysis with machine-learning algorithms, the proposed time-prediction method takes into consideration the performance deviation of the baseline time-prediction method with respect to different instances. It modulates the initial predicted value to get the final predicted value such that higher prediction accuracy can be achieved. The proposed method also absorbs the effect of outliers as long as a sufficient number of testing instances is provided.

- Along with a binary process status prediction, the proposed status-prediction method also estimates the probability of the predicted status. It further enables
conditional prediction if the intermediate process status is considered to be new evidence.

- By replacing multiple classification models with a single regression model, the proposed status-prediction method greatly reduces the cost of building extra classification models. It also considers the potential impact of the arbitrary nature of a predictive horizon.

4.2 Problem Statement and Data Source

4.2.1 Problem Statement

Figure 4.1 is an illustration of the proposed process-execution time and process status prediction framework. The framework can be established for any type of process whose execution time and status information is important for production management. Given an arbitrary predictive horizon, the framework is able to predict in real time the status and the probabilities of the status for the processes in the resource queue.

*Time-Prediction Problem Statement*

Process-execution time prediction is formulated as a regression problem. Each data instance is denoted as \((x, y)\), where vector \(x\) is the feature vector of the process and

![Figure 4.1](image-url): Illustration of the proposed process-execution time and process status prediction framework.
the numerical value $y$ is the prediction target, i.e., the execution time of the process. For instance, the number of pages, the width of one page, and the height of one page can be three features of a printing process. Given a feature vector $\mathbf{x}=(200, 11, 17)$, the proposed method may predict, for example, that it takes 214 seconds to print 200 A4 pages. The feature selection and construction process of the feature vector $\mathbf{x}$ are explained in detail in Section 4.2.4. A regression analysis process, which estimates the relationships among input features, has to be conducted in order to predict the numerical value of the process-execution time.

Relative mean error (RME) and root-mean-squared error (RMSE) are two performance metrics that can be used to evaluate a regression model. They are defined in Equation (4.1) and Equation (4.2), respectively:

\[
RME = \frac{1}{N} \sum_{i=1}^{N} \left| \frac{y_i - y_i^*}{y_i} \right|
\]

\[
RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left( \frac{y_i - y_i^*}{y_i} \right)^2}
\]

where $y_i$ is the actual value, $y_i^*$ is the predicted value, and $N$ is the number of instances that have been evaluated. The less the error, the higher the model accuracy.

### 4.2.2 Status-Prediction Problem Statement

Process status prediction is formulated as a classification problem. Each data instance is denoted as $(\mathbf{x}, y)$, where vector $\mathbf{x}$ is the feature vector of the process as defined in Section 4.2.1 and categorical value $y$ is the prediction target, i.e., the binary status (“completed” or “in-progress”) of the process. For instance, the number of books and the pages of one book can be two features of a binding process. Given a feature vector $\mathbf{x}=(12, 200)$, the proposed method predicts that after five minutes the binding process of 12 200-page books will still be “in-progress” with a probability of
0.9, and after ten minutes the binding process will be “completed” with a probability of 0.98.

Table 4.1 shows the confusion matrix which depicts how binary-class predictions on instances can be distinguished. The performance of a classifier can be evaluated by error rate which is defined in Equation (4.3).

\[
\text{error rate} = \frac{FP + FN}{FP + TP + FN + TN} \quad \text{(4.3)}
\]

\[
\text{error percentage} = \text{error rate} \times 100\% \quad \text{(4.4)}
\]

where \( FP \), \( FN \), \( TP \), and \( TN \) are described in Table 4.1.

4.2.3 Production Event Log

The surveillance system of an EIS records all the major activities and stores raw information into databases [9]. Before using this raw data, standard database preprocessing needs to be performed, such as extracting useful and correct data, transforming data into an appropriate format, and loading data into the target data warehouse [110]. After preprocessing all the records, we can get an event log organized on the basis of trace and event.

Figure 4.2 shows the event log structure. Each trace corresponds to a service. Each event corresponds to a process. Each event records key information, such as service ID, process ID, process description, the ID of the worker who executed this process and timestamps of the start and end of the process. The precedence relations among processes are reflected by the timestamps associated with each event.

Table 4.1: Confusion matrix of a status-prediction classification model.

<table>
<thead>
<tr>
<th>Actual class</th>
<th>Predicted class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive (completed)</td>
<td>True Positive (TP)</td>
</tr>
<tr>
<td>Negative (in-progress)</td>
<td>False Negative (FN)</td>
</tr>
<tr>
<td>Positive (completed)</td>
<td>False Positive (FP)</td>
</tr>
<tr>
<td>Negative (in-progress)</td>
<td>True Negative (TN)</td>
</tr>
</tbody>
</table>
4.2.4 Data Source

In this chapter, we use real-life data from the EIS production event log of a large multi-national mass customization manufacturer as a test case to demonstrate the building, optimization, and evaluation procedures of the proposed prediction framework.

In the test case, we have identified 13 ad-hoc features that contribute in predicting the process-execution time and status of one printing process which uses press as the resource. The resulting feature vector $x = (A_1, A_2, ..., A_{13})$ is explained in Table 4.2. Features consist of attributes of the process ($A_1$ to $A_9$), properties of the resource ($A_{10}$ to $A_{11}$), and the attributes of the resource queue ($A_{12}$ to $A_{13}$). This set of features include the attributes of the process ($A_1$ to $A_9$) as well as the interactive influence from other processes ($A_{10}$ to $A_{13}$). Feature data were extracted from EIS event logs and transformed into the right format for building the predictive models.

As shown in Fig. 4.3, the training, testing, and evaluation data were extracted from three consecutive weeks of real-life data. We used 4896 instances as the training data to build the model; 4637 instances as the testing data to tune parameters and
Table 4.2: Features of the input vector.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Property</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_1$</td>
<td>Numerical</td>
<td>Number of pages</td>
</tr>
<tr>
<td>$A_2$</td>
<td>Numerical</td>
<td>Width of one page</td>
</tr>
<tr>
<td>$A_3$</td>
<td>Numerical</td>
<td>Height of one page</td>
</tr>
<tr>
<td>$A_4$</td>
<td>Binary</td>
<td>Indicates whether print in color</td>
</tr>
<tr>
<td>$A_5$</td>
<td>Binary</td>
<td>Indicates whether print in black and white</td>
</tr>
<tr>
<td>$A_6$</td>
<td>Binary</td>
<td>Indicates whether print on book paper</td>
</tr>
<tr>
<td>$A_7$</td>
<td>Binary</td>
<td>Indicates whether print on glossy paper</td>
</tr>
<tr>
<td>$A_8$</td>
<td>Binary</td>
<td>Indicates whether print on one side</td>
</tr>
<tr>
<td>$A_9$</td>
<td>Binary</td>
<td>Indicates whether print on both sides</td>
</tr>
<tr>
<td>$A_{10}$</td>
<td>Numerical</td>
<td>Elapsed time since last maintenance</td>
</tr>
<tr>
<td>$A_{11}$</td>
<td>Numerical</td>
<td>Number of executed processes since last maintenance</td>
</tr>
<tr>
<td>$A_{12}$</td>
<td>Numerical</td>
<td>Number of processes in the queue that are ahead of this process</td>
</tr>
<tr>
<td>$A_{13}$</td>
<td>Numerical</td>
<td>Sum of the predicted execution times for processes in the queue that are ahead of this process</td>
</tr>
</tbody>
</table>

Figure 4.3: Partitioning of data for training, testing, and validation.

optimize the framework performance; and 4725 instances as the evaluation data to evaluate the performance of our proposed methods.

4.3 Process-Execution Time Prediction

4.3.1 Baseline Time-Prediction Method

An SVR model is adopted as the baseline time-prediction model. An introduction of SVR is presented in D. We select the popular sequential minimal optimization (SMO) algorithm to train the SVR model since it is known for its fast convergence [111]. Weka, the open source software [112], has been used to realize the algorithm. Parameters of the baseline time-prediction model built from the training data have
been tuned such that both RME and RMSE are minimized for the testing data. The baseline time-prediction model has maximum generalization ability for the training data and minimum error for the testing data. Figure 4.4(a) shows the procedure of the baseline time-prediction method for making a new prediction. The initial predicted time given by the baseline time-prediction model is treated as the result of the baseline method.

4.3.2 Proposed Time-Prediction Method

The proposed time-prediction method as shown in Figure 4.4(b) is designed upon and developed from the baseline time-prediction method. The procedure and the main components of the proposed time-prediction method are discussed as follows.

*Probability Mass Function (PMF) of the Relative Error*

After optimizing the parameters of the baseline time-prediction model, we derive the histogram and the probability mass function (PMF) of the model's relative error. The
PMF can be derived from the histogram; probability density is derived by dividing the frequency value by the number of instances. The relative error (RE) for an instance is defined as follows:

\[
RE = \frac{y^* - y}{y}
\]  

(4.5)

where \( y \) is the actual value and \( y^* \) is the predicted value given by the baseline time-prediction model. The histogram/PMF of the relative error by evaluating the testing data through the baseline time-prediction model is shown in Figure 4.5.

The key feature of the baseline time-prediction model is that its histogram or PMF of the relative error is uniquely determined by its performance over the testing data. Therefore, we can see that the histogram/PMF of the relative error can be used to describe the performance of the baseline time-prediction model. For instance,
as shown in Figure 4.5, the distribution of the relative error is highly skewed. The baseline time-prediction model makes overestimation more frequently than underestimation. The overestimation errors are also more severe than underestimation errors.

The more the number of testing instances, the more accurate is the histogram/PMF. Rather than fitting the PMF with any parametrized distribution model, we record the PMF of the baseline time-prediction model. The following procedure is designed to make use of the information provided by the histogram/PMF.

**Probability Mass Function (PMF) of the Adjusted Actual Value**

For every new instance, the baseline time-prediction model generates the initial predicted value. We made an assumption that the relative error of the initial predicted value follows the same distribution of the relative errors derived from historical data maintained by the baseline time-prediction model, i.e., the histogram/PMF of the relative error derived from the testing data. It is a reasonable assumption since the new instance is randomly generated from the the same unknown distributions as the testing data.

By transforming Equation (4.5), we can derive the following relationship:

\[
\text{RE} = \frac{y^* - y}{y} \Rightarrow y_a = \frac{y^*}{1 + \text{RE}} \tag{4.6}
\]

where \(y_a\) is the adjusted actual value of the new instance derived from the predicted value \(y^*\) given by the baseline time-prediction model and a given relative error (RE). Therefore, for every given \(\text{RE}\), an adjusted actual value can be derived from the predicted value through Equation (4.6). Due to such one-to-one mapping between the relative error and the adjusted actual value, the PMF of the adjusted actual value can be derived from the PMF of the relative error. That is, an adjusted actual
Figure 4.6: The histogram and the probability mass function (PMF) of the adjusted actual value derived from the prediction given by the baseline time-prediction model and its histogram/PMF of the relative error shown in Figure 4.5.

value has the same probability density as the relative error used to derive it.

Therefore, for every new prediction made by the baseline time-prediction model, we can derive the histogram/PMF of the adjusted actual value. For example, Figure 4.6 shows the histogram/PMF of the adjusted actual value derived from the histogram/PMF of the relative error shown in Figure 4.5 and the initial predicted value of 150 time units generated by the baseline time-prediction model.

Furthermore, once a new instance has been predicted by the baseline time-prediction model and evaluated by its actual value, it can be used to update the histogram/PMF of the relative error of the baseline time-prediction model. In this manner, the histogram/PMF of the relative error is generalized to all testing instances that the baseline time-prediction model has been applied to. In the following,
we consider the case when the histogram/PMF derived from initial testing data is not dynamically updated by new instances.

*Cumulative Density Function (CDF) of the Adjusted Actual Value*

We can further obtain the cumulative distribution function (CDF) of the adjusted actual value for a predicted value. If the histogram/PMF of the relative error is not dynamically updated, then two instances that obtain the same initial predicted value from the baseline time-prediction model will have the same CDF of the adjusted actual value. Only for different predicted values, the CDF of the adjusted actual values are different. Figure 4.7 shows the CDFs of adjusted values for different initial predicted values. For any point on the CDF curve, the value on the vertical axis is the probability that the process-execution time is no greater than its corresponding adjusted actual value on the horizontal axis. Therefore, the value on the vertical axis is also the process completion probability by the predictive horizon value on the horizontal axis.

*Final Predicted Value*

The final predicted process-execution time of the proposed time-prediction method is a candidate among the adjusted actual values. The CDF of the adjusted actual value is used for exploring the final predicted time. The proposed solution is to explore a time-prediction threshold and use its corresponding adjusted actual value on the CDF as the final predicted value. For instance, figure 4.8 shows the CDF of the adjusted actual value derived from the histogram/PMF shown in Figure 4.6 and an initial predicted value equals 150 time units. If the time-prediction threshold is 0.9, then the final predicted value adjusted from the initial predicted value is 141 time units.
Figure 4.7: The cumulative distribution functions (CDFs) of the adjusted actual value for different predicted values derived from the histogram/probability mass function (PMF) shown in Figure 4.6.

Time-Prediction Threshold

In order to make the most accurate process-execution time prediction, the time-prediction threshold should be optimized. Since the CDF of the adjusted actual value has a one-to-one mapping to the initial predicted value, the time-prediction threshold should be separately optimized for each initial predicted value. Consequently, each initial predicted value corresponds to a specific final predicted value.

However, the time-prediction threshold optimization requires that each initial predicted value appears for a sufficient number of times. Even if a large number of testing instances can be evaluated, such assumption cannot be guaranteed. Therefore, we optimize a single time-prediction threshold to modulate all initial predicted values, such that the final predicted values make the proposed method to achieve both less relative mean error (RME) and root-mean-squared error (RMSE) than the initial predicted values.

Figure 4.9 demonstrates the one-to-one mapping of the initial predicted value
Figure 4.8: The cumulative distribution function (CDF) of the adjusted actual value for one predicted value derived from the histogram/PMF shown in Figure 4.6.

to the final predicted value under the condition that the optimal time-prediction threshold equals 0.94.

4.3.3 Comparison Results and Discussions

Table 4.3 shows the relative mean error (RME) and root-mean-squared error (RMSE) of the baseline time-prediction method and the proposed time-prediction method tested through real-life EIS data. For instance, when evaluated using the demonstration test case, the RME and the RMSE of the baseline time-prediction method are 0.226 and 0.144, respectively. The proposed time-prediction method achieved smaller values of both RME and RMSE, 0.205 and 0.134, respectively. When evaluated using six other real-life test cases, the proposed time-prediction method has always achieved less RME and RMSE than the baseline time-prediction method.

The proposed time-prediction method generates the final prediction based on the
performance of the baseline time-prediction method. For instance, if the baseline
time-prediction method tends to overestimate (as shown in Figure 4.5), then the
proposed time-prediction method modulates the final predicted value to a smaller
value through the time-prediction threshold and the CDF of the adjusted actual
value (as shown in Figure 4.9).

The improvement of the proposed time-prediction method over the baseline time-
prediction method can be explained as follows. The error of a machine-learning
algorithm is due to both bias and variance as shown in Figure 4.10 [113]. Bias is
reduced and variance is increased in relation to the regression model complexity.
Complex models tend to overfit the training data and are not generalized enough
towards the testing data. The baseline time-prediction model has been trained to
minimize the error for testing data which is equivalent to minimize the total error of
the model. Moreover, the proposed time-prediction method associates the validation
data to the testing data. By modulating the initial predicted values for the validation
data through the histogram/PMF of the relative error derived from the testing data,
it reduces the error caused by variance even further. Therefore, it has achieved better
performance than the baseline time-prediction method.

Table 4.3: Relative mean error (RME) and root-mean-squared error (RMSE) of the
baseline time-prediction method and the proposed time-prediction method tested
through real-life EIS data.

<table>
<thead>
<tr>
<th>Test case based on real-life enterprise data</th>
<th>Relative mean error</th>
<th>Root-mean-squared error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demonstration</td>
<td>Baseline</td>
<td>Proposed</td>
</tr>
<tr>
<td>0.226</td>
<td>0.205</td>
<td>0.144</td>
</tr>
<tr>
<td>0.324</td>
<td>0.293</td>
<td>0.234</td>
</tr>
<tr>
<td>0.456</td>
<td>0.444</td>
<td>0.365</td>
</tr>
<tr>
<td>0.123</td>
<td>0.098</td>
<td>0.102</td>
</tr>
<tr>
<td>0.286</td>
<td>0.265</td>
<td>0.201</td>
</tr>
<tr>
<td>0.334</td>
<td>0.302</td>
<td>0.267</td>
</tr>
<tr>
<td>0.224</td>
<td>0.198</td>
<td>0.172</td>
</tr>
</tbody>
</table>
4.4 Process Status Prediction

4.4.1 Baseline Status-Prediction Methods

There are two baseline status-prediction methods that we consider in this chapter. The first method shown in Figure 4.11(a) compares the initial predicted process-execution time with the predictive horizon to check whether the process is still in progress. Such a method uses a regression model to realize a classifier. As long as time prediction is accurate enough, it can be further used to realize status prediction. If time prediction has a large error, then the error propagates to status prediction.

The second baseline status-prediction method shown in Figure 4.11(b) builds individual status-prediction classification models for all possible predictive horizons. A status-prediction classification model is used to directly predict the process status. By replacing the numerical target value (i.e., the actual process-execution time)
with the binary categorical target value (i.e., the actual process status at the given predictive horizon), a regression model can be easily converted into a classification model that is specifically built for the given prediction horizon. Clearly, the training process of the classification model requires the value of the predictive horizon to be known in advance to decide the target process status. Therefore, in order to predict process status for an arbitrary predictive horizon, one classification model has to be built for each predictive horizon. Such a feature makes the second baseline method computationally expensive due to the arbitrary nature of the predictive horizon.

4.4.2 Proposed Status-Prediction Method

Figure 4.12 shows the proposed status-prediction method. The detailed procedure is discussed as below.

**Initial Prediction**

Given an arbitrary predictive horizon, the probability of process completion by the predictive horizon can be directly obtained from the CDF of the adjusted actual value. It equals the probability that the process execution time is no greater than
the predictive horizon. For instance, as shown in Figure 4.8, if the predictive horizon is 138 time units, then the probability of process completion by 138 time units once it has been started is 0.55. The CDF also shows that it is highly unlikely that the process can be completed within 90 time units and will most likely be completed after 160 time units once it has been started.

**Conditional In-Progress Prediction**

Furthermore, according to probability theory [74], if a process has already been started and is still in progress, the probability of process completion by a predictive horizon is the conditional probability.

\[
P(\text{completed by } t_2 | \text{in-progress at } t_1) = \frac{P(\text{completed between } t_1 \text{ to } t_2)}{P(\text{completed after } t_1)}
\]  

(4.7)

Equation (4.7) shows that the probability of process completion by time \( t_2 \) given the evidence that the process is still in-progress at time \( t_1 \) equals the probability of process completion between time \( t_1 \) and \( t_2 \) divided by the probability of process com-
pletion after time $t_1$. The CDF of the adjusted actual value should also be updated by the given evidence that the process is still in-progress at time $t_1$. For example, Figure 4.13 shows the updated CDF of the adjusted actual value by Equation (4.7) from Figure 4.8, given that the process is still in-progress by 140 time units.

**Status-Prediction Threshold**

As demonstrated through Figure 4.8 and Figure 4.13, given a predictive horizon, the probability of process completion can be found through the CDF of the adjusted actual value. We define a status-prediction threshold in order to determine the class label as shown in Figure 4.12. By comparing the probability of completion to the status-prediction threshold, the final class label is “completed” if the probability is greater than the threshold, and “in-progress” if it is lower than the threshold.

The probability of process completion varies with the predictive horizon. Intuitively, the larger the predictive horizon, the higher the probability that the predicted status is “completed”. Therefore, the optimal status-prediction threshold that results in the minimal error rate status-prediction classifier should vary with the predictive
Figure 4.13: The updated cumulative distribution function (CDF) of the adjusted actual value derived from the histogram/probability mass function (PMF) shown in Figure 4.8 and the evidence that the process is in progress at time 140.

Figure 4.14 is obtained from optimizing the status-prediction thresholds for the demonstration test case. It shows that both the status-prediction threshold and the minimal error rate varies with the predictive horizon. Therefore, for a certain predictive horizon, the status-prediction model should select its corresponding status-prediction threshold for making the final classification.

Different from optimizing the proposed time-prediction method (discussed in Section 4.3.2, in which the time-prediction threshold can be optimized with respect to the initial predicted value generated by the baseline time-prediction model), the optimal status-prediction threshold of the proposed status-prediction method is explored with respect to the predictive horizon given by the model user.
4.4.3 Comparison Results and Discussions

We evaluate the performance of the proposed status-prediction method compared to the baseline methods. In the proposed method, we compare the process-completion probability (obtained from the CDF of the adjusted actual value, derived in Section 4.4.2 and Section 4.4.2) with the optimal status-prediction threshold of the predictive horizon (derived in Section 4.4.2). In the first baseline method, the process status is determined by comparing the initial predicted value with the predictive horizon.

Figure 4.15(a) shows that the performance of the proposed method is always superior than the first baseline method regardless of the predictive horizon. Figure 4.15(b) further zooms into the predictive horizon ranged from 230 to 590 time units. Both of them show that the error rate varies with respect to the predictive horizon.

As discussed in Section 4.4.1, the second baseline status-prediction method re-
Figure 4.15: The error rate of the proposed status-prediction method and the first baseline status-prediction method with respect to the predictive horizon ((b) is a zoom-in version of (a)).

requires building a status-prediction classifier for each predictive horizon. Therefore, we tested the second baseline method with some sampled predictive horizons. Comparison results are shown in Table 4.4. For instance, it shows that when the predictive horizon equals to 410 time units (i.e., the predictive horizon for which the proposed status-prediction method gives the highest error rate as shown in Figure 4.15), the error rate of the proposed method, the first baseline status-prediction method (a), and the second baseline status-prediction method (b) is 4.343%, 9.899%, and 8.536% respectively. It can be seen that the proposed status-prediction method demonstrates better performance than both the baseline status-prediction methods.

Furthermore, we have tested other real-life test cases, and the proposed method always achieve higher accuracies in both process-execution time (Table 4.3) and process status predictions. The improvement of the proposed status-prediction method over the baseline status-prediction methods is mainly due to the improvement of the proposed time-prediction method over the baseline time-prediction method (discussed in Section 4.3.3). In addition, the status-prediction threshold takes into consideration the impact of the arbitrary nature of the predictive horizon. Status-
Table 4.4: *Error percentage* results of the two baseline status-prediction methods (Figure 4.11) and the proposed status-prediction method at some sampled predictive horizons.

<table>
<thead>
<tr>
<th>Predictive horizon (time units)</th>
<th>Error percentage</th>
<th>Baseline methods (a)</th>
<th>Baseline methods (b)</th>
<th>Proposed method</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>5.859%</td>
<td>5.486%</td>
<td>1.818%</td>
<td></td>
</tr>
<tr>
<td>150</td>
<td>7.525%</td>
<td>7.563%</td>
<td>2.071%</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>7.879%</td>
<td>7.623%</td>
<td>2.121%</td>
<td></td>
</tr>
<tr>
<td>250</td>
<td>7.475%</td>
<td>7.195%</td>
<td>2.576%</td>
<td></td>
</tr>
<tr>
<td>300</td>
<td>7.626%</td>
<td>7.236%</td>
<td>2.879%</td>
<td></td>
</tr>
<tr>
<td>350</td>
<td>8.939%</td>
<td>7.852%</td>
<td>2.929%</td>
<td></td>
</tr>
<tr>
<td>410</td>
<td>9.899%</td>
<td>8.536%</td>
<td>4.343%</td>
<td></td>
</tr>
<tr>
<td>450</td>
<td>9.343%</td>
<td>8.125%</td>
<td>3.939%</td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>8.081%</td>
<td>8.227%</td>
<td>4.091%</td>
<td></td>
</tr>
<tr>
<td>550</td>
<td>5.909%</td>
<td>5.365%</td>
<td>3.939%</td>
<td></td>
</tr>
</tbody>
</table>

The prediction threshold serves as a new tunable parameter that can further enhance prediction accuracy.

### 4.5 Conclusion and Future Work

In this chapter, we have developed a framework for process-execution time and status predictions, and compared results to baseline (state-of-the-art) approaches using real-life EIS data.

We have shown that, although machine-learning algorithms have demonstrated their successes in time prediction, little work has been done to further explore its capability and flexibility. The proposed models extend baseline models and have achieved better performance as well as new functions and more flexibility.

The advantages of the proposed models and the reasons for achieving superior performance than baseline approaches can be summarized as follows:

- By integrating statistical analysis and machine-learning algorithms, the proposed time-prediction model takes into consideration the performance devia-
tion of the baseline time-prediction model with respect to different instances. It reasonably adjusts the initial predicted value to get the final predicted value such that less relative mean error and less root-mean-squared error can be achieved. The proposed model also absorbs the effect of outliers as long as its histogram of the relative error is derived from a sufficient number of testing instances.

- Along with a binary status prediction, the proposed process status prediction model also provides a probability estimation of the predicted status. It further enables conditional predictions if the intermediate status becomes known as evidence.

- By replacing multiple classification models by a single regression model, our status-prediction model greatly reduces the cost of building extra models. It also considers the effects caused by the arbitrary predictive horizon.

Finally, in our future work, the performance of the proposed system will be tested through a discrete-event simulation platform as an embedded artificial intelligent functional block in the whole EIS.
Optimization of Order-Admission Policies

A mass-customization enterprise offers personalized manufacturing services. Once an order is submitted to the enterprise by a client, the EIS needs to make a real-time decision on whether to accept or refuse this order. Based on the enterprise current capacity, and the order’s properties and requirements, an order is refused if its acceptance is not profitable for the enterprise. The order is accepted with the most appropriate due date in order to maximize the profit that can result from this order. We have developed an intelligent order-admission framework that provides admission decisions in real-time for new orders using machine-learning and decision-integration techniques. The framework consists of three classifiers: Support Vector Machine (SVM), Decision Tree (DT), and Bayesian Probabilistic Model (BPM). The classifiers are trained by history orders and used to predict completion status for new orders. A decision integration technique is implemented to combine the results of the classifiers and predict due dates. Experimental results derived using real factory data from a leading print-service provider and Weka open source software show that the order completion-status prediction accuracy is significantly improved by the decision-integration strategy. The proposed multi-classifier model also outperforms
a stand-alone regression model.

This chapter addresses order-admission policy optimization in an EIS using digital-print industry as an example. The rest of this chapter is organized as follows. Section 5.1 discusses prior art and motivation. An order-fulfillment prediction model which includes multiple classification models has been built to estimate order life cycle. The main components and the structure of the model are discussed in Section 5.2. Section 5.3 describes decision integration strategies for increasing the accuracy of model. In Section 5.4, we present simulation results and discuss the performance of the model. Section 5.5 concludes the chapter.

5.1 Background and Motivation

Commercial print is a highly competitive and service quality-oriented business with annual retail sales of over US$700B [9] [10]. Establishing a superior service engagement and fulfillment between clients and the print service providers (PSPs) is key to creating value and ensuring profit for a commercial print business [24]. The end-to-end fulfillment process of digital-print service starts with clients’ engagement to obtain print services [3]. The clients, who are also the content suppliers, request print services and supply content for print. Clients range from traditional publishing agencies for newspapers, books, and periodicals, to web-based digital media companies, e.g., www.snapfish.com. The PSP offers its manufacturing capability to the content suppliers as a form of utility service in exchange for payment. Today, on-demand digital-print services are leading the digitization of commercial print and commanding an impressive 8% annual growth rate [10]. Compared with traditional print services, which produce homogeneous print products (thousands of copies of the same content, also known as “long run-length”), on-demand digital-print deals with high-frequency and high-volatility service demands [25].

In the commercial digital-print industry, customers can request print products
using multiple methods. They may physically visit the PSP or electronically submit print content to the PSP through the internet. In order to describe a potential order, a document called “Statement of Work” (SoW) is created which provides three crucial pieces of information. The first is order intent, which specifies the customer’s print intent, for example, the types of products, the number of copies, as well as particular instructions such as laminating and quality requirements. The second is the expected due date which can be proposed from two methods. It can be requested by the client or given by the PSP. The third is the price of this order, which similarly, may come from either the client’s offer or the PSP’s proposal.

After a negotiation process, if the order is accepted, the price and the due date are finalized and a contract for this order is established. A production plan that specifies all the production steps to fulfill this order, is also devised and disclosed to the client. The production steps include sequences of production operations on the factory floor, including pre-press production, cover making, book block printing, laminating, cutting, collating, binding, packaging and shipping, etc. [26]. Besides offering high quality and good price, the PSP must deliver an order by its deadline since late delivery leads to extra shipping cost and reduces customer satisfaction or loyalty.

The most popular practice in today’s commercial print industry for order acquisition is based on human involvement [10]. This kind of contract establishment process between a PSP and the clients mainly depends on human judgment. When a potential service engagement arrives at a PSP, the production managers and the customer service representatives cooperate and then present to the client a tentative price and completion date. Such an approach is clearly incompatible with today’s mass customized production due to increasing diversity and complexity [25].
5.1.1 Related Prior Solutions

A more advanced technique implemented by a few of the industry-leading PSPs (e.g., Reischling Press, Inc. (RPI)), is a template-based automated order acquisition process [26]. First, a set of pre-determined static templates are developed through business cooperation with a strategic print buyer over several months. Each template represents a type of product and includes the associated rules to calculate the lead time of a product that can be mapped into this template. Once the templates are well established, they can be deployed so that the order acquisition process can be automated by using them. A new set of templates is required to be developed once the situation in the factory changes. Fig. 5.1 demonstrates the template-based lead-time estimation process for an order involving two products. Other examples of similar approaches include a case-based reasoning system for predicting job completions [114], and a fuzzy rule-based prediction of monthly precipitation [115].

None of the above solutions can account for the up-to-the-moment factory production situation. These techniques are static, non-adaptive and susceptible to faults. [116] has discussed the difficulty associated with decision-making in practical production management domains and identified the general opportunities that data-driven artificial intelligence systems can provide. For a real-life problem, [117] demonstrated that a data-driven machine-learning solution was needed when simple matching based on observations became ineffective for detecting target signals. In [118], memory based reasoning (MBR) and neural network (NN) learning are combined for yield improvement and an integrated framework is proposed for a yield management system based on hybrid machine-learning techniques. In [119], a set of heterogeneous independent learning components were coordinated by a central meta-reasoning executive. It has been demonstrated that machine-learning techniques assist us to manage complexity, diversity, and uncertainties in manufacturing
process management [120].

5.1.2 Costs for Service-Level Violation

Print production suffers from high process variation due to nontrivial occurrence of exception events that are beyond the planned production process. Typical examples are product reworking induced by machine malfunctions, digital content repair due to inconsistent files, service maintenance and replenishment of the consumables (paper
rolls, inks, etc.). All these exception events negatively impact the realization of the due-date commitment. In general, if an order cannot be completed on time, the PSPs may incur extra costs, for instance:

- Cost to upgrade shipping methods. This common remedial action for PSPs is used for compensating for the extra time consumed by manufacturing. Such unexpected shipping cost can be significant; it can reach US$0.5M through a busy season [25].

- Cost of goodwill. This cost is client-dependent. It is particularly important when PSPs conduct business with enterprise print buyers. A significant delivery delay beyond the promised due date given by the PSP can jeopardize the enterprise’s own business objective. Negative service experience consequently hurts a PSP’s re-occurring contracts and business development effort.

- Possible late penalty stated in the service contract.

On the other hand, PSPs can choose to reduce the price from what has been advertised if the customer agrees on a postponed due date. This type of cost is the price difference between the reduced price caused by postponing the due date and the original price. In this scenario, a postponed due date increases the likelihood of timely delivery, therefore, PSPs can accept the price penalty in order to avoid the potentially higher amount of late cost.

The cost to upgrade shipping methods can be quantified using the cost table provided by the shipping companies; the cost depends on the size (weight, volume) of the print product and the reduction of the shipping days. The cost of goodwill can be quantified by examining the Customer Relation Management (CRM) database and drawing correlations between the previous late service events and lost business opportunities. The cost stated in the service contract can be quantified by analyzing
the service contract. The cost can also be quantified by examining the history of postponed service events.

The goal of this chapter is to introduce a new order-admission framework and to show the efficiency of data-driven approaches adopted by the framework. The key contributions of this chapter are listed below:

- Relevant order- and factory-information attributes have been identified and used in a classification framework to make predictions for new orders.

- Decision integration techniques have been used to improve the prediction accuracy.

- Order due dates are explored through order completion-status classification results.

- We show that the proposed order-admission framework reduces the number of orders that miss their due dates.

5.2 Due-Date Validation Engine

As shown in Fig. 5.2, whenever a new order arrives at the factory, relevant information about this order along with current factory information is sent to the due-date validation engine. Based on the initial due date required by the customer, the due-date validation engine makes a prediction whether it can be delivered on time. If it can be completed as expected, then the contract can be established between the PSP and the customer. If it cannot be completed as expected based on current factory status, then the due-date validation engine proposes a new due date. The customer can decide whether the initial due date can be postponed. If not, the customer can pay more to increase the priority of this order. High-priority orders take less time to be completed since they are also scheduled in advance compared to regular orders.
The due-date validation engine is designed to automate the order admission process. The functionality of the engine is to output an instantaneous response to the contract on whether it can be delivered on time, and if not, the most appropriate due date. Fig. 5.3 shows the schematic of the due-date validation engine. The information of every incoming prospective contract is fed into the engine before a contract can be established.

The main difficulty in achieving high predictability is caused by two factors. One factor is the uncertainty that is inherent in the arrival of new orders. The other factor is the inevitably insufficient amount of input information. The available information for prediction at the moment when an order arrives only includes the history and the up-to-the-moment factory status information. However, whether an order can be completed on time is also affected by the factory status during the manufacturing process, which is unknown at the time when the prediction must be made. In order to address the first obstacle, instead of relying on just one type of machine learning
algorithm, we search for different algorithms to perform the prediction such that a broader coverage can enhance the overall prediction accuracy. It has been shown that integrating the advantages of multiple machine-learning algorithms leads to higher efficiency in practical scenarios [121] [122]. The second problem can be overcome by forecasting the factory performance, which is beyond the scope of this chapter and left for future research. The key components and functions of the proposed due-date validation engine are explained in detail in the following subsections.

5.2.1 Knowledge Base

The knowledge base pulls historical records from the factory surveillance system either periodically or on-demand (driven by an external request) [10]. It is composed of a historical record database that provides the past and current information of the
Table 5.1: Attributes of the input vector.

<table>
<thead>
<tr>
<th>Index</th>
<th>Attribute</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>OrderLeadTime</td>
<td>the proposed due date minus the accepted time</td>
</tr>
<tr>
<td>A2</td>
<td>ProductQuantity</td>
<td>the number of products in this order</td>
</tr>
<tr>
<td>A3</td>
<td>BookBlockQuantity</td>
<td>the number of book blocks in this order</td>
</tr>
<tr>
<td>A4</td>
<td>BookCoverQuantity</td>
<td>the number of book covers in this order</td>
</tr>
<tr>
<td>A5</td>
<td>ValueStreamA</td>
<td>indicates whether value stream A is required</td>
</tr>
<tr>
<td>A6</td>
<td>ValueStreamB</td>
<td>indicates whether value stream B is required</td>
</tr>
<tr>
<td>A7</td>
<td>ValueStreamC</td>
<td>indicates whether value stream C is required</td>
</tr>
<tr>
<td>A8</td>
<td>ValueStreamD</td>
<td>indicates whether value stream D is required</td>
</tr>
<tr>
<td>A9</td>
<td>ValueStreamE</td>
<td>indicates whether value stream E is required</td>
</tr>
<tr>
<td>A10</td>
<td>ValueStreamF</td>
<td>indicates whether value stream F is required</td>
</tr>
<tr>
<td>A11</td>
<td>ValueStreamG</td>
<td>indicates whether value stream G is required</td>
</tr>
<tr>
<td>A12</td>
<td>WorkInProgress</td>
<td>the number of products in the factory</td>
</tr>
<tr>
<td>A13</td>
<td>ResourceAQueueLength</td>
<td>task queue length of book-block presses</td>
</tr>
<tr>
<td>A14</td>
<td>ResourceBQueueLength</td>
<td>task queue length of book-cover presses</td>
</tr>
</tbody>
</table>

factory. The simulation engine is a functional block that can provide factory status forecast based on previous and current information about the factory.

5.2.2 Inputs to the Decision Engine

An attribute vector $\mathbf{x}$, which consists of 14 carefully selected key attributes from both the contract and the factory is constructed for every prospective contract. The same vector $\mathbf{x}$ is sent to all classifiers each time. Critical contract attributes are identified from the “Statement of Work”. The key factory status information is supplied by the knowledge base once the contract arrives at the factory. An example input vector $\mathbf{x}$ is shown as $\mathbf{x} = (A_1, A_2, ..., A_{14})$, and each attribute is explained in Table 5.1.

5.2.3 Outputs of the Decision Engine

Classifiers in the decision engine shown in Fig. 5.3 have been chosen because they are considered to be among the top-10 most influential machine-learning algorithms [123]. Each produces a prediction about the prospective contract’s completion status and a probability estimation $\text{Prob}$. There are two possible completion status that
can be viewed as two class labels. The “On-time order" label means that the factory is confident of delivering this order on time, and “Late order” implies that this order will potentially be late if it is due by the proposed due date. The probability estimate \( \text{Prob} \) is the confidence in making the prediction. Therefore, the output space \( Y \) for classifiers is a two-element set of class labels, \( Y = \{ \text{“On-time order”}, \text{“Late order”}\} \). The probability estimate \( \text{Prob} \in [50\%, 100\%] \). For example, if the prediction is “On-time order”, then \( \text{Prob} = \text{Prob}(N) \) and \( \text{Prob}(N) + \text{Prob}(P) = 1 \), where \( \text{Prob}(N) \) and \( \text{Prob}(P) \) are the probability estimations that an order is “On-time” or “Late” order.

5.2.4 Classifier Evaluation Metrics

Predictive accuracy is the main evaluation criterion for the predictive performance of classifiers [124]. There are four important metrics used for evaluating a classifier’s performance and only two of them are independent of each other [125]. They are true positive rate \( (TP) \), true negative rate \( (TN) \), false negative rate \( (FN) \), and false positive rate \( (FP) \). They represent the rate of correctly classifying a late order as “Late order”, correctly classifying an on-time order as “On-time order”, falsely classifying a late order as “On-time order”, and falsely classifying an on-time order as “Late order”, respectively. They are related as follows:

\[
TP + FN = 1 \tag{5.1}
\]
\[
TN + FP = 1 \tag{5.2}
\]

Note that, classification is done at the moment when an order arrives; however, the actual class label is known only after the order is completed.

5.2.5 Support Vector Machines

A nonlinear two-class Support Vector Machine (SVM) is used as one classifier. An SVM is a supervised machine-learning algorithm [126]. It can be trained for solving
binary classification problems. Although similar to neural networks [127], the quality of generalization and the ease of training exceed that of traditional methods. An SVM performs well on datasets that have many attributes. Even if very few training instances are available, as long as they are representative in the problem space, an SVM can still perform well. In addition, there is no upper limit to the number of attributes; hardware capabilities are the only constraints. SVMs have been successfully applied to model complex and real-world problems such as text and image classification, hand-writing recognition, as well as bioinformatics and biosequence analysis [128] [129]. A brief introduction to an SVM, illustrated by our problem, is presented below.

Each training instance corresponds to a history order. The training instance of order $i$ consists of an attribute vector $x_i$ and a “target value” $y_i$ (i.e., the class label, either “On-time order” or “Late order”). The goal of an SVM is to generate a model from training instances. The model can then make predictions of the class labels for new orders. A prediction is made by evaluating the attribute vector of a new order.

An SVM requires that each data instance be represented as a vector of real numbers [130]. However, our attributes include categorical features. Hence, before implementation, we need to pre-process data by converting attributes of categorical values into numerical values. We adopt $m$ numbers to represent an $m$—category attribute. Only one of the $m$ numbers is one while the others are zero. For instance, a two-category attribute such as {BookBlock, BookCover} can be represented as (0,1) and (1,0), respectively. Similarly, for class labels, we use $y_i = 1$ and $y_i = -1$ to represent “On-time order” and “Late order”, respectively. This coding technique performs better than the use of a single number to distinguish an attribute of different categories [130].

Another possible conversion of categorical attributes into numerical attributes is to establish a meaningful mapping of each attribute value into a numerical value.
For instance, based on manufacturing process complexity of soft-cover types, which is on a scale of (0 to 1), “landscape” corresponds to 0.5 and “minibook” corresponds to 0.7. This type of conversion requires detailed understanding of the correlation between categorical attributes and manufacturing processes. We do not consider such a targeted mapping technique here since our goal is to study due-date prediction in a general scenario. The proposed method can nevertheless be used with any specific mapping technique.

In Fig. 5.4, we illustrate our 14-dimensional problem using a two-dimensional representation. An SVM defines an optimal decision boundary to separate on-time and late orders. The margin from the optimal decision boundary to its closest vectors is the maximum among all possible separating boundaries to their closest vectors. Vectors that are either closest to the decision boundary or on the wrong side of the boundary are called support vectors. Only support vectors are maintained by
SVM while other vectors are dumped after training since the decision boundary can be uniquely determined by the support vectors. This is an important advantage of SVMs: the classification accuracy does not depend on the size of the training dataset [1].

Our optimization problem can be formulated as follows. We are given a set $TD_N$, the training dataset of $N$ order/instance-label/class pairs $\{x_i, y_i\}$, where $i = 1, ..., N$, $x_i \in R^{14}$, and $y \in \{1, -1\}$. There are $N$ history orders in the dataset, the number of on-time orders is $N_1$, and the number of late orders is $N_2$. The SVM computes the unique solution to the following optimization problem [130]:

$$\min_{w, b, \xi} \frac{1}{2} w^T w + C_{\text{on-time}} \sum_{i=1}^{N_1} \xi_i + C_{\text{late}} \sum_{j=N_1+1}^{N} \xi_j$$  \hspace{1cm} (5.3)

subject to:

$$y_{ij}(w^T \phi(x_{ij}) + b) \geq 1 - \xi_{ij},$$  \hspace{1cm} (5.4)

where $\xi_{ij} \geq 0, i = 1, ..., N_1, j = N_1 + 1, ..., N$. The SVM needs to explore values for vector $w$, variable $b$, and vector $\xi$ that minimize the objective function (5.3). According to the properties of an SVM, the distance from the closest vector to the optimal decision boundary is $1/\|w\|$, where $w$ is a 14-dimensional vector. The margin of the decision boundary, which is shown in Fig. 5.4, is $2/\|w\|$. Slack variable $\xi = (\xi_1, ..., \xi_i, ..., \xi_{N_1}, ..., \xi_j, ..., \xi_N)$ is a $N$-dimensional vector. Its element $\xi_i/\xi_j$ represents an error when an on-time/late order instance is misclassified into the wrong class. An error is the distance of a wrongly classified instance to the decision boundary. Parameters $C_{\text{late}}$ and $C_{\text{on-time}}$ are referred to as the penalty parameters of the error terms. Since we are dealing with highly imbalanced dataset (more on-time than late orders), penalty parameters for on-time and late orders have different values. The larger the value of $C$, the heavier the penalty. For instance, if $C_{\text{late}}$ is greater than
a heavier penalty is added for misclassifying a late order as an on-time order than misclassifying an on-time order as a late order.

Furthermore, since orders are inseparable in the linear space defined by the attribute set, the SVM needs to find a linear separating hyperplane (decision boundary plane) with the maximum margin in a higher dimensional space. Instead of showing a straight line in Fig. 5.4, a curve is used to represent the decision boundary projected from the higher dimensional space to the 2-dimensional space. Vectors \( \mathbf{x}_{i/j} \) are mapped into a higher dimensional space by the function \( \phi \). Function \( K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) \) is called the kernel function [1]. We adopt a radial basis kernel function (RBF) defined as follows:

\[
K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2), \quad \gamma > 0
\]

where \( \gamma \) is the RBF kernel parameter. The RBF kernel outperforms other kernels such as polynomial and sigmoid kernels in our problem. The term \( \mathbf{w}^T \phi(\mathbf{x}_{i/j}) \) in Constraint (5.4) is the inner product of \( \mathbf{w} \) and \( \phi(\mathbf{x}_{i/j}) \). Constraint (5.4) is illustrated in Fig. 5.4, which indicates for both classes, the allowed error distance is within \( \xi_{i/j} \). The first term in Equation (5.3) is the inverse of the margin, while the remaining terms equal the sum of errors. The objective function (5.3) is constructed to maximize the margin of the decision boundary while minimizing the sum of classification errors.

When classifying a new order, the SVM computes the value \( d = \mathbf{w}^T \phi(\mathbf{x}) + b \) which represents the distance of the new order to the decision boundary. The sign of \( d \) determines the class label. The magnitude of \( |d| \) determines the value of \( \text{Prob} \) as follows:

\[
\text{Prob} = \begin{cases} 
1 & \text{if } |d| \geq 1, \\
0.5|d| + 0.5 & \text{otherwise}.
\end{cases}
\]

In summary, there are three key tuning parameters in our SVM that must be
Figure 5.5: A illustration of a decision tree.

optimized. They are penalty parameters $C_{late}$ and $C_{ontime}$, and the parameter $\gamma$ in the RBF kernel. In addition, the popular Sequential Minimal Optimization (SMO) algorithm is implemented to train the SVM model since it is known for its fast convergence [111].

5.2.6 Decision Tree

A decision tree learner uses a divide-and-conquer algorithm to construct a tree from a training dataset. It is mostly used to explore features and extract patterns in large databases for classification, regression, and prediction [131]. A decision tree has desirable characteristics and merits such as high accuracy, fast computation, virtually
infinite capacity, and intuitive interpretation. Decision trees have been extensively implemented for more than two decades in various areas involving exploratory data analysis and predictive modeling applications [132].

We adopt a nonlinear two-class (binary) decision tree as the second classifier. Fig. 5.5 shows a simple decision tree. It discovers the underlying rules and regulations to best separate the training and testing datasets into “On-time” and “Late” orders with the least amount of errors. An error refers to a misclassification of an order instance. A brief introduction to decision trees is presented and exemplified by our problem.

A decision tree is a directed acyclic graph consisting of nodes. A decision tree classifier is defined as a classification procedure that recursively partitions a dataset into smaller subdivisions on the basis of a set of rules defined at each node in the tree. In Fig. 5.5, nodes are shown as squares. The tree is composed of a root node (which contains the entire training dataset $TD_N$), a set of internal nodes, and a set of terminal nodes or leaves [133]. Each node in a binary decision tree has only one parent node and zero or two descendant nodes.

Every node contains a set of orders denoted by $STD_M$, where $M$ is the size of $STD_M$ and $STD_M \subseteq TD_N$ (i.e., $STD_M$ is a subset of $TD_N$, $M \leq N$; for the root node, $STD_M = TD_N$). In each dataset $STD_M$, the number of late and on-time orders is denoted by $M_P$ and $M_N$, respectively. An evaluation matrix called information value or information entropy [134], which is denoted by parameter $info$, is defined and evaluated at each node as follows:

$$info(STD_M) = info([M_P, M_N])$$  \hspace{1cm} (5.6)

$$= -\frac{M_P}{M} \log \frac{M_P}{M} - \frac{M_N}{M} \log \frac{M_N}{M}$$  \hspace{1cm} (5.7)

where $M_P + M_N = M$. The property of $info$ can be expressed as follows. If $M_P$ or $M_N$ equal to 0, then $info(STD_M) = 0$. A node containing only late or on-time orders
has an $info$ value equals 0. On the other hand, if $M_p = M_N$, then $info(STD_M) = 1$ and it reaches its maximum value. A node with $info = 1$ has the same number of late and on-time orders. Hence, regularity and unambiguity of the order completion status at a node are characterized by small $info$, whereas, randomness and ambiguity are characterized by large $info$.

We adopt the popular greedy decision-tree algorithm “maximize the information gain” to construct the decision tree [131]. Starting from the root node, if a node has nonzero $info$, then its dataset $STD_M$ should be further divided into two non-overlapping subsets $STD_{M1}$ and $STD_{M2}$ by a set of rules. These rules divide the orders by checking their attribute values. Two descendent nodes are created and each of them contains $STD_{M1}$ and $STD_{M2}$, respectively. The algorithm requires that after the division of $STD_M$ at a node, the information value gained from this separation should be maximized. The objective is formulated as follows:

$$\text{maximize} \quad info(STD_M) - (info(STD_{M1}) + info(STD_{M2})) \quad (5.8)$$

A decision tree is constructed by exploring the rules that maximize the above objective at each node until all descendant nodes are leaves. There are three conditions under which a node becomes a leaf. (i) If its $info$ is zero, i.e., a node contains only one type of orders. (ii) Its $info$ is nonzero, however, orders in its dataset are indistinguishable by any attribute except for their class labels, i.e., its dataset cannot be separated by any rule. (iii) Its $info$ is nonzero, the ratio $\alpha$, which is defined as the ratio of the number of “Late” orders versus “On-time” orders, is below a certain threshold $\alpha_{th}$. Parameter $\alpha_{th}$ is used to limit the depth of a decision tree.

Once constructed from a training dataset, a decision tree can be used for classification. A new order instance represented by input vector $x_j$ traverses the decision tree from the root node to a leaf. The new order $x_j$ is classified as the dominant order type at the leaf. The probability estimation $Prob$ is the ratio of the number
of dominant orders versus the number of all orders at the leaf. A leaf represents a classification decision. In our problem, a leaf is either an “On-time order” or a “Late order” class label. Every path from the root node to a leaf can be expressed as a decision rule, which is a combination of rules at the nodes along the path. An example decision rule is as follows:

if $A_2$ is Cover Only and $A_4 > 1$ day, then $y = “On-time order”$

where $A_2$ and $A_4$ are the second and the fourth attribute in the input vector, respectively. This decision rule corresponds to the left-most path in the decision tree shown in Fig. 5.5.

In the real-life case studies that we considered, the training dataset contained more than 4000 order instances, the number of internal nodes in the decision tree was over 117, and the number of leaves was over 59.

5.2.7 Bayesian Probabilistic Model

The third classifier that we adopt is an efficient probabilistic model based on Bayes’ theorem [135]. An illustration of this model is shown in Fig. 5.6. By counting the number of on-time and late history orders that have been completed, we compute two prior probabilities $P(OnTime)$ and $P(Late)$, where $OnTime$ and $Late$ are the events that an order is delivered on time or late, respectively. Probability $P(OnTime)$ is approximated by the ratio of the number of on-time orders versus all orders. Probability $P(Late)$ equals $1 - P(OnTime)$. Two likelihoods for every value $a$ of every attribute $A$ of history orders, $P(A = a|OnTime)$ and $P(A = a|Late)$, are also computed by calculating the occurrence frequencies of $A = a$ for on-time and late orders, respectively. All these prior probabilities and likelihoods are stored in the knowledge base and updated whenever there are orders shipped on time or late. Therefore, unlike an SVM or a decision tree, the quantity of training orders is
important in a probabilistic model. The more the orders, the closer the occurrence frequency approximates the actual probability distribution.

When given a new input vector $x_j$ for prediction, the probabilistic model first applies Bayes’ theorem to obtain the posterior probabilities conditioned on each attribute. Given an attribute $A = a'$ in $x_j$, its two posterior probabilities are calculated as follows:

$$
P(OnTime|A = a') = \frac{P(A = a'|OnTime) \cdot P(OnTime)}{P(A = a')} \tag{5.9}
$$

$$
P(Late|A = a') = \frac{P(A = a'|Late) \cdot P(Late)}{P(A = a')} \tag{5.10}
$$

where $P(OnTime|A = a')$ and $P(Late|A = a')$ are the probabilities that this contract $x_j$ is going to be shipped on time or late conditioned on its attribute $A$ equals $a'$, respectively. The common term $P(A = a')$ in the denominators is set to 1.

We also assume that all the attributes $A_1, A_2, ..., A_{14}$ are independent of each other. This assumption does not lose generality since orders are very diverse in

FIGURE 5.6: An illustration of the Bayesian probabilistic model.
terms of attributes, and the quantity of past orders is considerable. Furthermore, no significant correlation was found among any two attributes in our evaluation. The property of independent events states that events $B$ and $C$ are said to be independent if [74]:

$$P(B \cap C) = P(B) \cdot P(C)$$  \hspace{1cm} (5.11)

Therefore, by applying the property of independent events, the probability that order $x_j$ will be shipped on time or late while considering all the attributes is computed as follows:

$$P(OnTime|x_j) = P(OnTime|A_1 = a_1) \cdot \ldots \cdot P(OnTime|A_{14} = a_{14})$$  \hspace{1cm} (5.12)

$$P(Late|x_j) = P(Late|A_1 = a_1) \cdot \ldots \cdot P(Late|A_{14} = a_{14})$$  \hspace{1cm} (5.13)

where each term on the right hand side is calculated by Equation (5.9) or (5.10).

In addition, we define a parameter $\delta$ to enhance the adaptability and controllability of the probabilistic model. As mentioned before, the numbers of late and on-time orders are heavily imbalanced since there were more on-time orders than late ones. The due-date validation engine will further amplify the imbalance. If decisions given by the engine were deployed, more orders could be delivered on time. Parameter $\delta$ is used to modify the prior probabilities, such that,

$$P'(Late) = P(Late) - \delta$$  \hspace{1cm} (5.14)

$$P'(OnTime) = P(OnTime) + \delta$$  \hspace{1cm} (5.15)

Whenever an order is shipped, it becomes a training instance for the probabilistic model. Consequently, the values of all prior probabilities and likelihoods are transient. Therefore, the value of $\delta$ is also trained and updated dynamically to achieve the least amount of classification errors.

In the final step of prediction, since probabilities $P'(OnTime|x_j)$ and $P'(Late|x_j)$ are calculated, the probabilistic model sets the predicted class label for order $x_j$ to
be the one that has a higher value. Therefore the probability estimation Prob is the corresponding $P'$.  

### 5.2.8 Comparison of Classifiers

This subsection summarizes similarities and differences between the three machine-learning classifiers. First, an SVM is an instance-based classifier whose decision boundary is defined by representative instances known as support vectors. A decision tree is a rule-based classifier and it is equivalent to a set of decision rules. A probabilistic model is a probability-based classifier, whose final result is obtained by comparing probabilities. Second, an SVM is a nonlinear classifier in the attribute space. It finds the decision boundary by transforming the attribute space into a higher-dimensional space with the application of a kernel function. A decision tree is a nonlinear classifier in the attribute space [133]. A probabilistic model can be viewed as a linear classifier. By taking logs of both sides of Equation (5.12) and (5.13), we obtain the following transformations:

\[
\log(P(OnTime|x_j)) = \sum_{i=1}^{14} \log(P(OnTime|A_i = a_i)) \quad (5.16)
\]

\[
\log(P(Late|x_j)) = \sum_{i=1}^{14} \log(P(Late|A_i = a_i)) \quad (5.17)
\]

Therefore, the value of $\log(P(OnTime|x_j))$ or $\log(P(Late|x_j))$ is a linear combination of $\log(P(OnTime|A_i = a_i))$ or $\log(P(Late|A_i = a_i))$, where $1 \leq i \leq 14$. Furthermore, the performance of an SVM or a decision tree depends on the representative order instances rather than the quantity of instances. A probabilistic model requires sufficient number of order instances to compute prior probabilities and likelihoods [136]. Since the classifiers are diverse as discussed above, we consider a combination of them to achieve better prediction.
5.3 Decision Integration

In this section, we describe decision integration strategies for increasing the accuracy of classification. Inputs to the decision integration center are the classification results (class labels and probability estimations) given by each classifier. Each classifier has its own advantages, constraints and applicability determined by their underlying algorithms. Therefore, the predictions for the same order given by the three classifiers may be different. Decision integration is implemented in the interest of preserving the advantage of each classifier. There are some popular candidates such as majority voting, weighted voting, and so on [137]. We considered three decision integration approaches: Dempster-Shafer theory based decision integration approach, decision fusion approach, and voting approach. The decision integration center can select an approach for a given target scenario.

5.3.1 Dempster-Shafer theory-based decision integration approach

Dempster-Shafer theory (DST) is a mathematical theory of evidence that allows one to combine evidence from different sources and arrive at a degree of belief that takes into account all the available evidence [138]. It is often used to deal with problems that are lack of parametric probability distribution models or accurate mathematical analysis [139]. The implementation of DST in our problem is presented below.

In a Dempster-Shafer reasoning system, all possible mutually exclusive facts are enumerated in “the frame of discernment Θ” [140]. In our problem, an order is either a “Late” or an “On-time” order, which are represented as facts $P$ (positive) and $N$ (negative), respectively. Therefore, in our problem, $Θ = \{P, N\}$ to indicate whether an order is late or on time.

Classifier $S_i$ contributes its prediction for an order by assigning its beliefs over $Θ$. This assignment function is called the “probability mass function” of classifier $S_i$, and
it is denoted by \( m_i \). According to classifier \( S_i \)'s prediction, the probability that “the order is late” is indicated by a “confidence interval” [\( \text{Belief}_i(P), \text{Plausibility}_i(P) \)]. For every classifier, there are only four possible evidences for an order and they are described in Table 5.2, in which \( TP, TN, FP, \) and \( FN \) rates are classifier performance metrics (Section 5.2.4), and parameters \( \text{Prob}(P) \) and \( \text{Prob}(N) \) are classifier outputs (Section 5.2.3). Evidence \( E_1 \) and \( E_2 \) support proposition \( P \) (\( A = \{E_1, E_2\} \)), whereas evidence \( E_3 \) and \( E_4 \) support proposition \( N \).

Table 5.2: Four possible evidences of the completion status for an order.

<table>
<thead>
<tr>
<th>evidence</th>
<th>real order status</th>
<th>predicted order status</th>
<th>( m(E_k) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( E_1 )</td>
<td>( P )</td>
<td>( P )</td>
<td>( TP \cdot \text{Prob}(P) )</td>
</tr>
<tr>
<td>( E_2 )</td>
<td>( P )</td>
<td>( N )</td>
<td>( FN \cdot \text{Prob}(N) )</td>
</tr>
<tr>
<td>( E_3 )</td>
<td>( N )</td>
<td>( N )</td>
<td>( TN \cdot \text{Prob}(N) )</td>
</tr>
<tr>
<td>( E_4 )</td>
<td>( N )</td>
<td>( P )</td>
<td>( FP \cdot \text{Prob}(P) )</td>
</tr>
</tbody>
</table>

The lower bound of the confidence interval is the belief confidence, which accounts for all evidence \( E_k \) that supports the given proposition \( P \). Therefore:

\[
\text{Belief}_i(P) = \sum_{E_1, E_2 \subseteq A} m_i(E_k) = m_i(E_1) + m_i(E_2) \quad (5.18)
\]

The upper bound of the confidence interval is the plausibility confidence, which accounts for all the evidence that do not rule out the given proposition \( P \). Therefore,

\[
\text{Plausibility}_i(P) = 1 - \sum_{E_3, E_4 \cap A = \emptyset} m_i(E_k) = 1 - m_i(E_3) - m_i(E_4) \quad (5.19)
\]

For each possible proposition, e.g., \( P \), DST gives a rule of combining classifier \( S_i \)'s prediction \( m_i \) and classifier \( S_j \)'s prediction \( m_j \) as follows:

\[
(m_i \boxplus m_j)(P) = \frac{\sum_{E_k \cap E_k' \subseteq A} m_i(E_k)m_j(E_k')} {1 - \sum_{E_k \cap E_k' = \emptyset} m_i(E_k)m_j(E_k')} \quad (5.20)
\]
Furthermore, this combination rule can be generalized using iteration. After combining the predictions of two classifiers, the combined result can be treated as a new classifier such that it can be further combined with the third classifier.

### 5.3.2 Decision fusion approach

In this subsection, we introduce the decision fusion approach, demonstrate the implementation in our problem, and discuss its properties.

Decision fusion techniques have been applied to areas such as computer-aided diagnostic models for breast cancer [141], distributed sensor surveillance systems [142], and other fields. In these fields, for instance, medical imaging and distrusted sensing, the sensing events are immersed in significant noise from diverse sources. Decision fusion techniques are used to primarily detect if the true signature is present.

**Implementation**

The due-date validation engine also uses decision fusion technique as one integration framework for multiple classifiers. In our problem, late deliveries occur less frequently than on-time deliveries and incur extra costs for the factory. Late deliveries should be captured more accurately than on-time deliveries, therefore, they are referred to as positive signals that need to be detected.

Each classifier makes either a “positive” (late order) or a “negative” (on-time order) prediction for a new order. Note that, alternatively, on-time orders can be treated as positive signals. As long as classifiers are the same, the integration results are not affected by taking which class label as the positive signal.

There are four conditional probabilities denoted as $P(p|p)$, $P(n|p)$, $P(p|n)$, and $P(n|n)$, which represent the probability of classifying an order as positive/negative conditioned on its actual completion-status as positive/negative. For example, parameter $P(p|p)$ is the probability of correctly classifying a late order. Therefore,
$P(p|p)$ can be estimated by the $TP$ rate.

For the $i$th classifier, variable $d_i$ denotes its prediction decision for an order, where $d_i \in \{p, n\}$. We define a parameter $\lambda(d_i) = P(d_i|p)/P(d_i|n)$. Therefore, if $d_i = p$, then $\lambda(d_i) = P_i(p|p)/P_i(p|n) = TP_i/FP_i$. Similarly, if $d_i = n$, then $\lambda(d_i) = P_i(n|p)/P_i(n|n) = FN_i/TN_i$. Note that, since each classifier makes a binary decision, there are only two possible values of $\lambda(d_i)$, i.e., $\lambda(d_i = p)$ and $\lambda(d_i = n)$.

In decision fusion approach, positive signal $\lambda(d_i = p)$ should always be greater than negative signal $\lambda(d_i = n)$. Otherwise, the value of $\lambda(d_i)$ is in contradiction to signal definition. We then have the following:

\[
\begin{align*}
\lambda(d_i = p) & > \lambda(d_i = n) \\
\frac{TP}{FP} & > \frac{1 - TP}{1 - FP} \\
\Rightarrow TP & > FP
\end{align*}
\]

Therefore, in order for a classifier to be used in decision fusion approach, its $TP$ must by greater than $FP$.

The prediction of each classifier is independent. Parameter $\lambda(d)$ is defined as $\lambda(d) = \lambda(d_1) \cdot \lambda(d_2) \cdots \lambda(d_N)$. Therefore, each new order has its value $\lambda$ computed based on predictions from classifiers as follows:

\[
\lambda(d) = \frac{\prod_{i=1}^{N} p(d_i|p)}{\prod_{i=1}^{N} p(d_i|n)} \quad (5.21)
\]

Note that, since there are only two possible values of $\lambda(d_i)$, the total possible values of $\lambda$ for a new order is $2^N$. They can be referred to as $\lambda_1$ to $\lambda_{2^N}$ in ascending order.

We then define parameter $\lambda_{th}$ as a threshold value which is used to determine the final prediction. If $\lambda < \lambda_{th}$, then the final prediction is negative. If $\lambda > \lambda_{th}$, then the final prediction is positive.
As shown in Fig. 5.7, the number of possible positions for $\lambda_{th}$ when $N = 3$ is $2^3 - 1 = 7$. Note that, it is useless for $\lambda_{th}$ to be less than $\lambda_1$ or greater than $\lambda_{2N}$ since otherwise the final prediction is uniformly negative or positive. We compute

$$\lambda_{thj} = (\lambda_j + \lambda_{j+1})/2.$$ 

Therefore, based on each threshold value, the three classifiers generate a new classifier. Its performance can be evaluated by the same metrics as for other classifiers.

**Properties**

In this subsection, we demonstrate that the decision fusion approach covers multiple voting strategies that can be used for integrating multiple decisions. First, we rank the three classifiers by $\lambda(d_i = p)/\lambda(d_i = n)$ in descending order and rename them as
classifier A, B, and C. As shown in Fig. 5.7, each tuple \((n/p, n/p, n/p)\) represents
the prediction results of classifier \((A, B, C)\). Furthermore, each tuple corresponds to
a unique value of \(\lambda\), which is shown in Fig. 5.7. As we have discussed, the seven
threshold values \(\lambda_{th}\) correspond to seven classifiers, and they further correspond to
seven voting strategies as demonstrated in Fig. 5.7. For example, only a tuple with
at least two positive predictions has a \(\lambda\) greater than \(\lambda_{th4}\). Therefore, parameter \(\lambda_{th4}\)
corresponds to majority voting strategy.

5.3.3 Voting approach

The voting approach deploys a simple and efficient joint method to integrate three
class labels and it determines that the final class label is “Late order” only when all
the three input class labels are “Late order”. If any classifier inputs an “On-time
order” label, the final class label is “On-time order”.

The reason for deploying this voting strategy is because \(FP\) rates of all classifiers
are high. This kind of strategy takes the advantage of the differences among
classifiers. Only on-time orders that they all misclassified as late orders are the final
misclassified orders that contribute to the final \(FP\) rate. The smaller the overlap,
the lower the final \(FP\) rate.

Similar to the DST-based approach, the voting approach also generates one new
classifier. Therefore, when we apply three decision integration approaches and use
three classifiers, there can be a total \(3 + 1 + 7 + 1 = 12\) classifiers that can run in
parallel.

5.3.4 Exploring new due dates

After a new order is predicted to be late, the next step is to explore the appropriate
due date. If we continue increasing the due date, there is a value that changes the
predicted order-completion status from late to on-time. This value is illustrated

126
in Fig. 7.5 and denoted as $dp$, where parameter $DL$ and $DU$ denote the lowest and highest allocated manufacturing time for orders that have ever seen in history. Therefore, we define a “phase-change point” for due date to be a value that flips the predicted order-completion status from “late” to “on-time”. In order to enhance the efficiency of reaching to the phase-change point, we adopted a binary search method and ten steps showed good convergence performance for real-life order data.

5.4 Results and Discussions

In this section, we evaluate the proposed due-date validation engine using real-life order data from RPI. Our goal is to show that we can accurately predict which orders will miss their designated due dates. Such prediction results can be used to negotiate new due dates with customers.

The SVM and decision tree model were implemented using the open-source machine learning Weka software [112]. Original order due dates were assigned by the
template-based method as discussed in Section 5.1.1.

5.4.1 Classifier Evaluation Strategy and Results

Discrete-event simulation is an important tool for analyzing the characteristics of complex manufacturing systems [143]. A virtual print factory (VPF) for simulating commercial digital-print operations based on Ptolemy operation-simulation software [61] was first developed in [10]. The end-to-end fulfillment processes implemented and order data utilized in the VPF were directly pulled from RPI’s enterprise resource planning database. The results generated by the simulation platform with RPI’s internal factory audit data were validated and found to have good correlation with the situation on the real factory floor [3] [23].

The due-date validation engine has been integrated into the VPF. With the VPF running continuously, two sets of data were collected from the knowledge base during two consecutive periods (each period is seven days in terms of the factory time). The first dataset was used as training data and the second dataset was used as testing data.

After the first two periods, parameters in the three classifiers were optimized, such that each classifier can reach a $TP$ rate no less than 0.95 and a $FP$ rate as low as possible. During the third consecutive period, whenever a new order is accepted by the order admission framework, each classifier predicts a class label as either “On-time order” or “Late order”. We record the predictions made by each individual classifier as well as different decision integration approaches. As the simulation continues, orders accepted during the third period are completed either on time or late according to their due dates. The actual order completion status is compared to the predicted class label in order to evaluate the prediction accuracy of each classifier. This process is shown in Fig. 5.9. The comparison results for different classifiers and evaluation metrics are shown in Table 5.3, Fig. 5.10, and
Start training (3 classifiers)

Period 1
Collect training data

Factory time

Period 2
Collect testing data

All classifier parameters have been optimized

Period 3
Turn on due-date validation engine

Evaluate prediction performance

Period 4
Turn on due-date validation engine

Evaluate factory performance

Due-date validation engine off

Due-date validation engine on

Figure 5.9: Training, testing, and evaluation flow of the classifiers and the VPF.

Table 5.3: Comparison results and tunable parameters for classifiers.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>TP rate</th>
<th>FP rate</th>
<th>Tunable parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>0.969</td>
<td>0.508</td>
<td>$C_{late}, C_{ontime}, \gamma$</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>0.994</td>
<td>0.708</td>
<td>$\alpha_{th}$</td>
</tr>
<tr>
<td>Bayesian Probabilistic Model</td>
<td>0.992</td>
<td>0.508</td>
<td>$\delta$</td>
</tr>
<tr>
<td>Dempster-Shafer theory</td>
<td>0.975</td>
<td>0.178</td>
<td>none</td>
</tr>
<tr>
<td>$\lambda_{th1}$</td>
<td>1.0</td>
<td>0.771</td>
<td></td>
</tr>
<tr>
<td>$\lambda_{th2}$</td>
<td>0.999</td>
<td>0.625</td>
<td></td>
</tr>
<tr>
<td>$\lambda_{th3}$</td>
<td>0.996</td>
<td>0.569</td>
<td></td>
</tr>
<tr>
<td>$\lambda_{th4}$</td>
<td>0.991</td>
<td>0.500</td>
<td></td>
</tr>
<tr>
<td>$\lambda_{th5}$</td>
<td>0.970</td>
<td>0.168</td>
<td>$\lambda_{th}$</td>
</tr>
<tr>
<td>$\lambda_{th6}$</td>
<td>0.970</td>
<td>0.167</td>
<td></td>
</tr>
<tr>
<td>$\lambda_{th7}$</td>
<td>0.947</td>
<td>0.142</td>
<td></td>
</tr>
<tr>
<td>Voting approach</td>
<td>0.947</td>
<td>0.142</td>
<td>none</td>
</tr>
<tr>
<td>Decision Integration Center</td>
<td>none</td>
<td>Integration approaches</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 5.11. Table 5.3 also summarizes the parameters that need to be optimized for each classifier. They are the penalty parameters $C_{late}$ and $C_{ontime}$ shown in Equation (5.3), parameter $\gamma$ in a RBF kernel shown in Equation (5.5), parameter $\alpha_{th}$ used for constructing a decision tree, parameter $\delta$ shown in Equation (5.14), and the most efficient decision integration approach among the three options we have evaluated. If the decision-fusion approach is adopted, then the value of $\lambda_{th}$ should be optimized.

The percentage of orders that miss deadlines (missed-order percentage) is an important metric used for evaluating a factory’s service-level violation performance.
Figure 5.10: Comparison results of TP and FP rates of classifiers.

[66]. It is defined as the ratio of the number of late orders versus total number of orders completed during a certain period. In order to reduce the costs for service-level violation, a factory requires a low missed order percentage.

5.4.2 Factory Performance Evaluation Strategy and Results

After the evaluation of classifiers’ prediction accuracy, we tested the performance of the order admission framework. New simulations were carried out. The evaluation strategy is shown in Fig. 5.9 and described as follows:

(i) Classifiers in the due-date validation engine are optimized after the first two periods (Section 5.4.1).

(ii) In the remaining time, if a new order is predicted as a “Late order”, the due-date validation engine assigns it a new due date corresponding to the “phase-change point” (Section 5.3.4).
(iii) Simulation continues until the end of the fourth period. Completion status of orders are obtained according to new due dates. We record the new due dates assigned by the decision integration center as well as each individual classifier. Missed order percentages of the fourth period according to different classifiers were measured and the results are presented in Fig. 5.11.

5.4.3 Simulation results for synthetic order data generated using real-life RPI data

We adopted the proposed order-admission framework to another data set that was generated using the real-life RPI data. We randomly selected pairs of orders and randomly exchanged some attributes (but not due dates). Simulation results are shown in Table 5.4.

![Figure 5.11: Comparison results of missed-order percentage resulting from different classifiers and integration methods.](image)
5.4.4 Discussion

The following key conclusions can be drawn from the simulation results:

1. Each individual classifier offers a trade-off between a high $TP$ rate, i.e., “Late order” prediction accuracy, and a low $FP$ rate, i.e., “On-time order” misclassification probability. However, the optimal classifier should have a high $TP$ rate and a low $FP$ rate. Even though $TP$ rate can be made as high as 0.96, the value of $FP$ rate is no less than 0.5.

2. The SVM classifier has the lowest $FP$ rate but also the lowest $TP$ rate, whereas, the decision tree classifier has the highest $TP$ rate but also the highest $FP$ rate. It is very hard to find a single machine-learning algorithm that can achieve lowest $FP$ rate and the highest $TP$ rate simultaneously.

3. If the decision integration center implements the voting approach, the value of $FP$ rate is greatly reduced from more than 0.7 (given by the decision tree classifier) to less than 0.15 with a drop of $TP$ rate from 0.96 (given by the SVM classifier) to 0.94. Therefore, by deploying the voting approach, a much

<table>
<thead>
<tr>
<th>Classifier</th>
<th>$TP$ rate</th>
<th>$FP$ rate</th>
<th>Missed order percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>0.975</td>
<td>0.503</td>
<td>1.042</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>0.987</td>
<td>0.721</td>
<td>0.205</td>
</tr>
<tr>
<td>Bayesian Probabilistic Model</td>
<td>0.996</td>
<td>0.518</td>
<td>0.911</td>
</tr>
<tr>
<td>Dempster-Shafer theory</td>
<td>0.960</td>
<td>0.166</td>
<td>1.223</td>
</tr>
<tr>
<td>$\lambda_{th1}$</td>
<td>0.998</td>
<td>0.755</td>
<td>0.043</td>
</tr>
<tr>
<td>$\lambda_{th2}$</td>
<td>0.991</td>
<td>0.685</td>
<td>0.320</td>
</tr>
<tr>
<td>$\lambda_{th3}$</td>
<td>0.986</td>
<td>0.620</td>
<td>0.548</td>
</tr>
<tr>
<td>$\lambda_{th4}$</td>
<td>0.986</td>
<td>0.564</td>
<td>0.665</td>
</tr>
<tr>
<td>$\lambda_{th5}$</td>
<td>0.982</td>
<td>0.555</td>
<td>0.868</td>
</tr>
<tr>
<td>$\lambda_{th6}$</td>
<td>0.968</td>
<td>0.150</td>
<td>1.471</td>
</tr>
<tr>
<td>$\lambda_{th7}$</td>
<td>0.945</td>
<td>0.149</td>
<td>1.477</td>
</tr>
<tr>
<td>Voting approach</td>
<td>0.950</td>
<td>0.160</td>
<td>1.275</td>
</tr>
</tbody>
</table>

Table 5.4: Simulation results for classifiers over a synthetic order set.
lower FP rate can be achieved with negligible reduction in the TP rate.

In addition, we draw the following conclusions regarding the choice of classification and decision integration methods.

1. The performance of the DST-based approach is similar to classifications resulting from the decision fusion approach. For the decision fusion approach, when $\lambda_{th7}$ is used, the performance is the same as the voting approach. This can be explained by the properties of the decision fusion approach discussed in Section 3.2.2.

2. Different $\lambda_{th}$ value results in different classifier performance. Depending on the requirements of TP or TN rates, the decision fusion approach provides more choices for optimizing the integrated results compared to other approaches.

3. By comparing Fig. 5.7 and Table 5.3, we can see that classifiers correspond to $\lambda_{th1}$, $\lambda_{th2}$, and $\lambda_{th3}$ can still increase TP rate compared to individual classifiers. Starting from $\lambda_{th4}$, the TP rate becomes lower than individual classifiers. This is because the voting strategy of an order being classified as a late order becomes stricter as $\lambda_{th}$ increases. As demonstrated by Fig. 5.7, fewer orders can be classified as late orders as $\lambda_{th}$ increases. Although some late orders have been correctly identified by individual classifiers, the voting strategy rules them out.

4. Demands on TP and TN rates vary with the factory needs. For instance, if late-delivery associated cost is high, then a high TP rate is desired. If postponing due dates cost more, then a low FP rate is most demanding. Decision fusion approach provides more options on TP and TN rates. The threshold value can be adjusted according to the actual need.

5. In the following discussion, we assume that the decision integration center implements the voting approach if a low FP rate is the most desirable per-
formance. Analysis for other integration approaches are similar to the voting approach. The order admission framework accepts orders that are predicted to be “On-time” immediately and postpones the due dates of orders according to recommendations given by the due-date validation engine. The due-date validation engine can timely identify potential late orders and assign new due dates for them based on up-to-the-moment situation. Therefore, the missed order percentage can be reduced compared to the situation when due dates were assigned by static rules.

6. Fig. 5.11 shows that the voting approach and the decision fusion approach with $\lambda_{th7}$ have the highest missed order percentage among all classifiers. It appears that decision fusion approach with $\lambda_{th1}$ improves the factory-level performance the most since it generates the lowest missed order percentage. However, $FP$ rate determines the likelihood of due dates being postponed for potential “On-time” orders. Unnecessary postponement reduces potential profits by underestimating the productivity of the factory. This phenomenon is highlighted by the SVM and the probabilistic model. Although the two classifiers have the same $FP$ rate, the probabilistic model has a higher $TP$ rate. Consequently, the probabilistic model has a lower missed order percentage than the SVM model. The voting approach and the decision fusion approach with $\lambda_{th7}$ have the lowest likelihood of postponing due dates for potential “On-time” orders.

7. In addition, although the training of a classifier takes a certain amount of time, predictions can be done instantly. For example, the training of the SVM classifier takes a few minutes. Whereas, the training of the decision tree and probabilistic model is instantaneous. The training time depends on the algorithm, size and complexity of the attribute vector, as well as the size of the dataset. Therefore, the training of classifiers can be done in advance, whereas,
predictions are made in real time.

8. Similar to the real-life order set, the synthetic order set also shows that decision integration approach can be a good choice to improve prediction accuracy. The proposed order-admission framework can be used to reduce the number of missed orders for a print factory similar to RPI.

In summary, the problem we have attempted to solve is to enhance the accuracy of identifying orders that are close to the order completion-status boundary. The main difficulty in prediction is caused by overlapping of on-time and late orders in the order-attribute space. The boundary can be placed in favor of decreasing $FN$ rate with the inevitable increase in $FP$ rate, or vice versa. For instance, we can aim for much higher $FP$ rates than $FN$ rates of all three classifiers.

Even if the boundary can be well-defined, as illustrated in Fig. 5.12(a), the lowest $FP$ rate that the static template-based method can achieve is as good as an individual classifier (e.g., 0.508), since the static method cannot distinguish orders within the overlap area. By adopting decision integration, the data-driven approach
significantly reduces the $FP$ rate (e.g., to only 0.142). Based on history and up-to-date orders, the data-driven approach tackles this challenge by estimating the probability that an order is placed on either side of and close to the boundary.

Decision integration provides more options for making the final prediction. It is equivalent to adding an extra attribute to the attribute space. Compared to Fig. 5.12(a), an extra attribute $b$ is added by adopting decision integration, therefore, boundary $b$ can be established. The overlap area is divided into four regions by the two boundaries. Region 1 contains late orders that are misclassified as on-time orders. Region 2 contains on-time orders that are misclassified as late orders. Region 3 and 4 have conflict classifications given by the two boundaries. One possible integration strategy can be defined such that orders in both region 3 and 4 are all classified as on-time orders.

We then evaluate $FN$ and $FP$ rates in both cases. Areas of late and on-time orders are the same and denoted by area\{late orders\} and area\{on-time orders\}, respectively. Let $FN(a)$ denote the false negative rate of classification in case (a). Hence, $FN(a) = \text{area}\{\text{region 1}\}/\text{area}\{\text{late orders}\} = 6/36$, $FP(a) = \text{area}\{\text{region 2}\}/\text{area}\{\text{on-time orders}\} = 12/36$. Similarly, let $FN(b)$ denote the false negative rate of classification in case (b). Hence, $FN(b) = \text{area}\{\text{region 1, 3, 4}\}/\text{area}\{\text{late orders}\} = 7/36$, $FP(a) = \text{area}\{\text{region 2}\}/\text{area}\{\text{on-time orders}\} = 2/36$. We can see that according to the decision integration strategy, $FP$ rate is reduced from 12/26 to 2/36, whereas $FN$ rate is only increased from 6/36 to 7/36.

Therefore, as the complexity and diversity of both orders and the factory increase, a static prediction method becomes even more inaccurate, and a data-driven order-admission policy, as presented in this chapter, must be adopted to optimize performance.
5.5 Conclusion

We have presented an intelligent order-admission framework that predicts order completion status and due dates in real time for each incoming order using machine-learning and decision-integration techniques. We have considered the impact of diverse order attributes and dynamic factory status. We have deployed three powerful machine-learning algorithms in the framework: support vector machine, decision tree, and Bayesian probabilistic model. The classifiers have been trained and optimized using history orders. Three decision integration approaches have been applied to derive a final prediction. The performance of classifiers has been tested on the virtual print factory simulation platform using real-order data from a PSP. Results show that the integration strategy outperforms stand-alone classification methods. The number of orders missed deadlines has been significantly reduced compared to traditional static template-based method.
In addition to production workflow optimization (Chapter 3), the analysis and prediction of the execution performance of an individual process (Chapter 4), and the analysis and prediction of the fulfillment performance of an individual order (Chapter 5), the EIS should further be able to analyze and predict enterprise service-level performance. The collaborative goal of the preceding operation-optimization and knowledge-discovery efforts is to assist an enterprise to achieve superior service-level performance. The eventual concern of an enterprise is to satisfy service-level agreements (SLAs), to provide quality of service guarantees, and to ensure profitability. Therefore, in this chapter we study the analysis and prediction of enterprise service-level performance.

An enterprise service-level performance time series is a sequence of data points that quantify demand, throughput, average order-delivery time, quality of service, or end-to-end cost. Analytical and predictive models of such time series within an EIS can provide meaningful insights into potential business problems and generate guidance for appropriate solutions. Time-series analysis includes periodicity detection,
decomposition, and correlation analysis. Time-series prediction can be modeled as a regression problem to forecast a sequence of future time-series data points based on the given time series. The state-of-the-art (baseline) methods employed in time-series prediction generally apply advanced machine-learning algorithms. In this chapter, we propose a new univariate method in dealing with mid-term time-series prediction. The proposed method first analyzes the hierarchical periodic structure in one time series and decomposes it into trend, season, and noise components. By discarding the noise component, the proposed method only focuses on predicting repetitive season and smoothed trend components. As a result, this method significantly improves upon the performance of baseline methods in mid-term time-series prediction. Moreover, we propose a new multivariate method in dealing with short-term time-series prediction. The proposed method utilizes cross-correlation information derived from multiple time series. The amount of data taken from each time series for training the regression model is determined by results from hierarchical cross-correlation analysis. Such a data-filtering strategy leads to improved algorithm efficiency and prediction accuracy. In conclusion, by combining statistical methods with advanced machine-learning algorithms, we have achieved a significantly superior performance in both short-term and mid-term time-series predictions compared to state-of-the-art (baseline) methods.

The rest of this chapter is organized as follows. Section 6.1 discusses the background and motivation for enterprise service-level performance analysis and prediction. Section 6.2.4 introduces the state-of-the-art solutions and our data source. In Section 6.3, we present hierarchical time-series decomposition, the baseline model, and the proposed model for making mid-term time-series prediction. Following it, Section 6.4 presents time-series correlation analysis and the proposed multivariate short-term time-series prediction model used for short-term prediction. Section 6.5 concludes the chapter.
6.1 Motivation and Prior Work

Enterprise service-level objectives (SLOs) include demand, throughput, order delivery time, quality of service, and end-to-end cost [4]. They are usually dominated by on-time delivery or due-date constraints [23]. An enterprise service-level performance time series is a sequence of data points on one of its SLOs [144]. There are many key purposes for analyzing and predicting enterprise service-level performance: (1) to identify and adapt to changing demand patterns; (2) to support the generation of synthetic demand that represents future demand, e.g., demand for several days, weeks or months into the future to support capacity planning exercises; (3) to derive models that can be used for forecasting purposes; and (4) to study how demand influences other service-level performances such as throughput and on-time/late delivery [145]. Analytical and predictive models of such time series can provide meaningful insights into potential business problems and generate guidance for appropriate solutions.

Time-series analysis includes periodicity detection, decomposition, and correlation analysis. Periodicity of a time series can be identified through classical Fourier-transformation-based spectral analysis [146] and auto-correlation analysis [147]. Studies in [148] have successfully demonstrated the integration of these two approaches for periodicity detection. Even more sophisticated algorithms are introduced in [149] for the same purpose. Periodicity can be used for time-series decomposition. A season-trend decomposition procedure based on the LOESS moving-average algorithm was introduced in [150]. The procedure decomposes a time series into additive trend, season, and remainder. Multiple interdependent time series are becoming more correlated with each other; this is the main motivation for performing multivariate analysis for multiple time series.

Time-series predictions mainly involve short-term and mid-term predictions. Short-term prediction focuses on predicting only a few future data points for a given time
series. In contrast, mid-term prediction forecasts a sequence of future data points, including at least one period of the given time series. Statistical methods have often been adopted in time-series prediction. Historically, enterprise time-series prediction has relied upon curve fitting and queueing models. Unfortunately, these approaches are typically human-intensive and expensive processes \cite{151} \cite{145}. They are difficult to automate and scale for broader use. In addition, enterprise time-series data are inherently noisy and nonstationary \cite{4}. The nonstationary characteristic implies that the distribution of the time series changes over time. It becomes impossible to statically model the gradual changes in a time series. Techniques for short-term predictions often use approaches such as ARMA \cite{147} or GARCH \cite{152} models. While these approaches may be good enough for the short-term, their predictions quickly converge undesirably to a mean value for longer terms \cite{145}. A review in \cite{153} discusses more methods and progress made in time-series forecasting.

In recent years, machine-learning algorithms have been developed to address time-series or sequential data-prediction problems. Popular algorithms include sliding window methods, hidden Markov models, conditional random fields, and graph transformer networks, and so on \cite{154}. Neural networks have been successfully used for modeling financial time series \cite{155}. Neural networks are universal function approximators that can map any nonlinear function without \textit{a priori} assumptions about the properties of the data \cite{156}. Unlike traditional statistical models, neural networks are data-driven, nonparametric models. They are also more noise-tolerant in terms of having the ability to learn complex systems with incomplete and corrupt data. Among different machine-learning algorithms, support vector machines (SVMs) have gained traction in time-series prediction applications ranging from the original application in pattern recognition \cite{157,158} to the extended application of regression estimation \cite{159}. SVMs usually achieve higher generalization performance than traditional neural networks. Training an SVM is equivalent to solving a linearly
constrained quadratic programming problem so that the solution is always unique and globally optimal. The remarkable characteristics of SVMs such as good generalization performance, the absence of local minima, and sparse representation of the solution, have contributed their popularity. In summary, state-of-the-art time-series prediction methods use either statistical methods or machine-learning algorithms.

However, univariate mid-term prediction and multivariate short-term prediction are still bottlenecks for real-time responsiveness in an enterprise system [160]. Short-term and mid-term predictions of enterprise service-level performance are difficult to make due to the following factors. First, there exists a large amount of variations in multiple domains. Such as diverse service requests that arrive in a bursty or unpredictable manner, processes that rely on a significant amount of manual work, and the occurrence of stochastic failures. Client demands can also vary dramatically, e.g., “peak-to-peak” ratios may differ by an order of magnitude or more [161]. Moreover, due to the lack of adequate system understanding, some key features might be missed during forecasting. Finally, computationally expensive offline optimization solutions cannot be deployed in real-time applications [160].

Take the digital-print industry as an example. Today, print service providers (PSPs) rely on skilled managers for service-level time-series prediction, which is an error-prone process. It impedes a thorough understanding of the enterprise and the subsequent capacity-planning process.

In this chapter, we propose a univariate mid-term time-series prediction method and a multivariate short-term time-series prediction method. The proposed methods account for a hierarchical periodic structure in any given time series and temporal correlations between different time series. By combining statistical methods and advanced machine-learning algorithms, we have achieved improved performance in both short-term and mid-term time-series predictions as opposed to state-of-the-art (baseline) solutions.
The main contributions of this chapter are as follows.

- Hierarchical season-trend decomposition enables a better understanding of the time-series structure. Season components tend to repeat themselves, whereas noise components are highly unpredictable. By identifying and discarding noise components, the proposed univariate mid-term prediction method only focuses on predicting repetitive season and smoothed trend components. This method has significantly improved upon the performance of state-of-the-art (baseline) methods in mid-term time-series prediction.

- Studying components obtained from hierarchical decomposition also helps to perform and validate cross-correlation analysis between multiple time series. Cross-correlation information reveals a significant amount of cause-and-effect relationships between different time series.

- The proposed multivariate short-term time-series prediction method uses cross-correlation information explored from multiple time series. The amount of historical data of each time series used for training the regression model is determined through hierarchical cross-correlation analysis. The inclusion of only necessary information improves both algorithm efficiency and prediction accuracy.

6.2 Problem Statement, Baseline Methods, and Data Source

6.2.1 Problem Statement

We can separate a time series into two parts: known (including historical and current) and predictive data. As illustrated in Figure 6.1, suppose that given a known time-series data \( f(t), f(t-1), \ldots, f(t-L) \) at time \( t, t-1, \ldots, t-L \), respectively, the model can make an \( n \)-step ahead prediction (i.e., predict the future values of \( f(t+1) \), \( f(t+2) \), \ldots, \( f(t+n) \)).
by analyzing known data. The constant $L$ is the length of the known time series (the number of historical data points). The parameter $n$ is the furthest data point in the future that is of interest for prediction.

Relative squared error (RSE) can be used to evaluate time-series models. It is defined in Equation (6.1) as follows.

\[
RSE = \frac{\sum_{i=1}^{N} (Y_i^* - Y_i)^2}{\sum_{i=1}^{N} (\bar{Y}_i - Y_i)^2}
\]

where $Y_i$ is the actual value, $Y_i^*$ is the modeled or predicted value, and $N$ is the total number of data points that have been evaluated. RSE can be compared between time series whose values are measured in different units.

There are two methods to make an $n$-step ahead prediction. Note that, only one future data point is predicted each time. Once a data point is predicted, it will be treated as a known data point and used as input to predict the next data point. One method is iterative prediction, in which whenever a data point is predicted, the model will be re-trained by adding the predicted data as known data. For an $n$-step ahead iterative prediction, $n$ models need to be trained to make all the predictions. The other method is one-time prediction, i.e., all the future data points are predicted by the model which was initially trained by the original time series. In general, iterative prediction makes less error than one-time prediction. However, the quality of the results deteriorates very quickly as the predictive horizon increases due to the accumulation of prediction errors, regardless of the prediction algorithm [162].

### 6.2.2 Univariate Method

Data-driven methods for single time-series prediction can be characterized as a univariate method using existing data points as input to predict future data points for the time series. For a machine-learning based univariate time-series prediction method, each input instance for training the model is a vector $(x_i, y_i)$, where
Figure 6.1: Time-series prediction model.

Figure 6.2: Generation of the training data for an univariate time-series prediction method.

$x_i \in X \subseteq R^m$, $m$ is the number of features, and $y$ is the actual value. The training data includes multiples of such instances. The generation of the initial training data for a time series with $L$ known data points is illustrated in Figure 6.2. In order to predict one data point, a sequence of $g$ historical data points should be provided. In this case $m = g$. Parameter $g$ is referred to as the lag of the model. It can be seen that the number of instances in the training data is $L - g$. Theoretically, the longer the known time series ($L$) and the more the historical data ($g$) in each instance, the more informative is the training data.
6.2.3 Multivariate Method

Compared with the univariate method illustrated in Figure 6.2, the multivariate method involves one target time series and multiple assisting time series as shown in Figure 6.3. This method uses the information of the target and multiple assisting time series in order to predict new data for the target time series. Note that the input feature vector $x_i$ now includes $g$-lag data from the target and the assisting time series, i.e., $x_i = (t_i, a_i^1, a_i^2, \ldots)$, $1 \leq i \leq (L - g)$, where $L$ is the length of the target time series, parameter $g$ is the lag of the model, $t$ is the historical data of the target time series, and $a_j^i$ is the historical data of the $j$th assisting time series.

6.2.4 Data Source

The IT stack (enterprise resource planning, manufacturing execution and workflow management systems) in an EIS tracks the lifecycle of an order and documents the production activities (e.g., receiving, negotiating, admitting, planning, realizing, and delivering an order). Each tracking event produces an event log written to an event-log database. An event log includes a timestamp, a process ID, an equipment ID, an employee ID, IDs for this order, this product and this part, and other ancillary infor-
Figure 6.4: Data extracted from a large multi-national mass customization enterprise’s production event-log database. The first row shows the normalized daily counts of demand, throughput, and late delivery. The second row zooms into the region covered in the black box of the first row.

The factory managers use the production event logs to debug the production process in case things go wrong. Our experimental environment is composed of an event-log database obtained from a large multi-national mass customization enterprise. It includes 664 million records. It is managed by a 5-node VERTICA cluster (v6.1.1-0, Ubuntu 64 bit) in support of fast computation. We mine this production event-log database to uncover the production reality and extrapolate the trend into the future. In compliance with our confidentiality agreement with the enterprise, real-life data have been normalized by nonlinear functions only for visualization purpose. Therefore no actual value of time or quantity is presented.

Figure 6.4 shows the data from day 1 to day 1940 that we have extracted from
the production event-log database. The first row shows the flows of the demand, production throughput, and late delivery in terms of normalized daily product count. The second row zooms into the region inside the black box on the first row. It reveals the weekly periodic patterns that cannot be determined from the lower-resolution view in the first row.

We partition the data into training (time series from day 1 to day 1550) and testing data (time series from day 1551 to day 1970) to train and test the proposed methods.

6.3 Mid-Term Time-Series Analysis and Prediction

6.3.1 Time-Series Decomposition and Modeling

Enterprise time-series data often contain multiple periodic patterns. Daily, weekly, monthly, quarterly and yearly patterns can be observed at different scales. For instance, daily patterns distinguish peak and off-peak hour and weekly patterns distinguish weekday and weekend trend, while yearly patterns distinguish holidays and work days. It is obvious that understanding this time-varying feature is the key to enterprise time-series modeling.

Classical Fourier-transformation based spectral analysis can be used to identify time-series periodicity [146]. A Fourier transformation gives an overlay of harmonics for the time-varying magnitude of the time series. The most dominant frequencies (frequencies showing large intensities) provide candidate periods. Auto-correlation analysis can also be used to identify periodicity [147]. Auto-correlation reveals similarity between a time series and its shifted version by different lags. A high value for the auto-correlation at lag $g$ denotes that the original time series looks similar to itself when shifted by $g$ time steps. Therefore, if the auto-correlation shows local extrema at multiples of a lag $g$, it is a strong indicator that $g$ is a candidate period. Cross-
correlation can be used to identify similarity among different time series. Formal definition and further details on auto-correlation are introduced in [147]. Since there are many standard techniques for periodicity detection, we directly use candidate periods obtained from observing the data and confirmed by these techniques.

After identifying candidate periods, we decompose the time series with respect to them. According to a classical time-series additive model [150], a time series consists of a trend component, a season component, and a remainder. That is, suppose the time-series data, the trend component, the season component, and the remainder are denoted by $Y_v$, $T_v$, $S_v$, and $R_v$, respectively, for $v = 1$ to $N$. Then

$$Y_v = T_v + S_v + R_v \quad (6.2)$$

The trend component is the low-frequency variation in the time-series data together with nonstationary long-term changes. The season component is the variation in the time-series data at or near the seasonal period. The remainder usually characterizes the influence of noises which is the variation in the time-series data beyond that in the season and trend components. We apply a season-trend decomposition procedure based on LOESS moving-average algorithms (STL) as discussed in [150] on the enterprise demand, throughput, and late-delivery time series shown in Figure 6.4.

We first decompose the time series with the largest candidate period (i.e., a yearly period) using the STL procedure as presented in [150]. The resulting inter-year trend components are shown in Figure 6.5. Seasonal components are repeated over yearly period. Figure 6.6 shows the intra-year season components, in which weekly pattern and burstiness are revealed more clearly. The remainders were treated as white noises and discarded.

Since weekly periodicity is the next smaller candidate period with strong intensity in spectral analysis, we further apply the STL procedure to decompose the intra-year season component (shown in Figure 6.6) to intra-year trend component (shown in
Although demonstrated through a large multi-national mass customization enterprise’s data, the same hierarchical decomposition procedure can be applied to any time-series data. In summary, hierarchical time-series decomposition involves the following steps: (1) hierarchical periodicity detection through visual inspection, spectral analysis, and auto-correlation methods; (2) first-level STL decomposition starting from the largest candidate period; (3) remaining STL decompositions always start with the next smaller candidate period with strong intensity in spectral analysis. Note that a multiple of a candidate period is also another valid candidate period.
We then model the original time series shown in Figure 6.4 according to the hierarchical aggregation strategy shown in Figure 6.9. We aggregate three components. The first component is the inter-year trend (Figure 6.5). The second component is the yearly-repeated intra-year trend (Figure 6.7). The third component is the weekly-repeated intra-week season (Figure 6.8). Note that the aggregation of the second and the third components replaces the noisy yearly season obtained from the first-level decomposition procedure. Therefore, the hierarchical aggregation strategy
6.3.2 Support Vector Regression

Support Vector Regression (SVR) has been demonstrated as a powerful technique for time-series prediction [159]. A brief introduction to SVR is presented in D. Considering its advantages as discussed in Section 6.1, SVR is adopted in this chapter for building regression models.

6.3.3 Baseline Methods

The baseline univariate method is introduced in Section 6.2.2. Today’s data-driven time-series prediction approaches (e.g., WEKA [112]) treat the data as being flat and rely on machine-learning algorithms (e.g., SVM) to uncover the patterns. This class of approaches can achieve high accuracy in short-term prediction oftentimes

Excludes noises obtained from both the first-level and the second-level decomposition procedures. Relative squared errors of the modeled demand, throughput, and late delivery time series are 0.13, 0.28, and 0.63, respectively. It can be seen that the demand and throughput time series are better structured than the late delivery time series. Consequently, the more structured the original time series, the better the modeled time series approaches the original one.
requiring extensive manual tuning of model parameters. As the predictive horizon increases, performance deteriorates. This happens because overfitting of the known time series is beneficial for making short-term predictions but detrimental for making longer-term predictions [162].

Theoretically, as discussed in Section 6.2.2, the longer the known time series \((L)\) and the more the historical data in each instance \((g)\), the more informative is the training data. However, the feasible values of \(L\) and \(g\) are limited in real-time applications because the training time of an SVR scales between quadratic and cubic with respect to \(L - g\) (the number of training samples) and \(g\) (the number of input features) [163].

The time required for training an SVR model is the main time consumption in a real-time application. Since demand time series is the most structured data, we use demand data to evaluate time required for training an SVR model. Experimental results on the average time to train the model and to make predictions are shown in Table 6.1. It can be clearly seen that although making new predictions can be done instantly, training a model with large lag values can be time consuming. Furthermore, an iterative model with a lag of 364 days (i.e., the largest candidate period) is not practical for reasonable computation resources (e.g., Windows workstation with 8G memory and 2.5GHz CPU in our experiments).

Figure 6.10 shows the comparison results for different baseline methods. It can be seen that baseline methods with less lags cannot capture the yearly pattern of the data. On the other hand, iterative baseline methods with a one-year lag are

<table>
<thead>
<tr>
<th>Time required</th>
<th>Lag = 30</th>
<th>Lag = 364</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 step ahead</td>
<td>420-step ahead</td>
</tr>
<tr>
<td>Iterative</td>
<td>9 seconds</td>
<td>1 hour</td>
</tr>
<tr>
<td>One-time</td>
<td>9 seconds</td>
<td>9 seconds</td>
</tr>
</tbody>
</table>
infeasible for real-time applications. Therefore, we select the one-year lag one-time model as the baseline method to compare the performance of our proposed method, as discussed in Section 6.3.4.

6.3.4 Proposed Univariate Mid-Term Time-Series Prediction Method

As discussed in Section 6.3.3, the main difficulty in mid-term time-series prediction is the deteriorating performance as the predictive horizon increases. The main motivation for the hierarchical decomposition of a time series is to explore its structure more thoroughly and reduce the unpredictable components. Instead of predicting the whole data, which contains white noise, we investigate distributed predictions of each trend component and then reconstruct the whole data by integrating predicted trend components and repeated season components.

We used SVR models to learn and predict the data from day 1551 to day 1970. According to the hierarchical decomposition procedure shown in Section 6.3.1 and the aggregation procedure shown in Figure 6.9, we predict the data for day 1551 to day 1970 by aggregating the predicted data of three components as illustrated in Figure 6.11. The first component is the predicted inter-year trend for day 1551 to
day 1970. The second component is the historical intra-year trend. The historical intra-week season component in Figure 6.8 is repeated for 52 times as the third component. Therefore, the predicted data for from day 1551 to day 1970 is the aggregation of the predicted inter-year trend, the historical intra-year trend, and the repeated intra-week season. This is because after the two-level decomposition, the intra-year season component is replaced by the aggregation of the intra-year trend and the repeated intra-week season components. The complete time series for day 1551 to day 1970 is constructed by aggregating each components according to the hierarchical structure as shown in Figure 6.11.

6.3.5 Results and Discussions

Figure 6.12 shows the comparison between the proposed method and the baseline method. Relative square errors (RSEs) of one-year-ahead predicted demand and throughput given by the proposed method are 0.23 and 0.32, respectively. The corresponding errors are 0.41 and 0.94 for the baseline method. Evaluated using RSE, the proposed method demonstrates better performance than the baseline method. Furthermore, it can be seen that for the throughput time series, which is less structured than the demand data, the baseline method deteriorates significantly as the

---

**Figure 6.11:** Predicted time series constructed from predicted and repeated hierarchical components.
predictive horizon increases. Therefore, we can predict scenarios for the least structured late-delivery data. The baseline method clearly cannot be used for predicting one-year-ahead throughput and late delivery due to large errors. The improvement for such less structured data through the proposed method is more significant. The proposed method demonstrated its robustness for less structured time-series data, which is a major bottleneck for the baseline method.

Other than a thorough understanding of the periodic nature of a time series, the improved performance gained by the proposed method also demonstrates another motivation for hierarchical time-series decomposition. The amount of noise that is
removed in the decomposition is proportional to the number of hierarchical levels involved in the process.

Furthermore, another advantage of hierarchical decomposition is that it helps to improve \( n \)-step ahead prediction when \( n \) is even larger than the largest period \( P_{\text{max}} \). In such a case, the original time series should be initially decomposed by a multiple factor of its largest period such that \( c \cdot P_{\text{max}} \geq n \), where \( c \) is a constant. Meanwhile, the largest period \( P_{\text{max}} \) is updated to a multiplication of the original value, \( P'_{\text{max}} = c \cdot P_{\text{max}} \). It is an advantageous feature for time-series data with smaller periods but require longer-term predictions. Considering the real enterprise data, if weekly period was its largest detected period, we can still first decompose it with a yearly period since yearly period is a multiplication of weekly period.

6.4 Multivariate Short-Term Time-Series Analysis and Prediction

6.4.1 Time-Series Cross-Correlation Analysis

The decomposition procedure and the hierarchical patterns of the demand, throughput and late-delivery curves led us to examine the cross correlations between them. There are clear lags of peaks in throughput and late delivery compared to demand as shown in Figure 6.4.

For modeled integrated time series, intra-week season pattern (Figure 6.8), intra-year trend pattern (Figure 6.7), and inter-year trend pattern (Figure 6.5), we compute cross correlations among two different time series. The steps for performing cross-correlation analysis among two time series can be found in [147]. Results in Table 6.2 show that demand leads throughput and late delivery by one to three days. This agrees to our supply-chain empirical knowledge about the enterprise.

Figure 6.13 shows the normalized data in one month by offsetting throughput and late delivery 3 days ahead to provide the visual contrast about such cross-correlation.
FIGURE 6.13: Normalized demand, and 3-day offset throughput and late delivery daily count in one month from a large multi-national mass customization enterprise data.

6.4.2 Baseline Methods

The results from cross-correlation analysis motivate us to propose that historical demand, throughput, and late delivery data together can improve the quality of the prediction. To validate this idea, we built a multivariate model that predicts the next data point (one-step ahead) of a time series by utilizing multiple assisting time series.

We used both the standard univariate (Section 6.2.2) and multivariate (Section 6.2.3) time-series prediction methods implemented in Weka [112] as baseline methods

<table>
<thead>
<tr>
<th>Most correlated lag between two time series</th>
<th>demand &amp; throughput</th>
<th>demand &amp; late delivery</th>
<th>throughput &amp; late delivery</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modeled time series</td>
<td>-1</td>
<td>-3</td>
<td>0</td>
</tr>
<tr>
<td>Intra-week pattern</td>
<td>0</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>Intra-year pattern</td>
<td>-3</td>
<td>-3</td>
<td>-1</td>
</tr>
<tr>
<td>Inter-year pattern</td>
<td>-3</td>
<td>-3</td>
<td>-1</td>
</tr>
</tbody>
</table>
6.4.3 The Proposed Multivariate Short-Term Time-Series Prediction Method

We design a new multivariate short-term time-series prediction method based on the conventional multivariate method (Section 6.2.3). Considering that the training time for an SVR model scales between quadratic and cubic with respect to the number of training samples and the number of input features [163], we try to use only the most informative features while guaranteeing the prediction performance.

The improvement over the conventional multivariate method is illustrated in Figure 6.14. Rather than including as many historical data points as the target time series, the number of historical data points from the assisting time series is wisely determined. Different weights of assisting time series for making the prediction are reflected through selecting different amounts of historical data points from each assisting time series. The proposed multivariate short-term time-series prediction method uses the cross-correlation analysis results to determine the length of historical assisting time series that should be used for prediction.

**Figure 6.14**: Generation of the training data for the proposed multivariate time-series regression model.

for comparision.
Table 6.3: Relative square errors of different prediction methods for short-term (one-day-ahead) throughput prediction.

<table>
<thead>
<tr>
<th>Method</th>
<th>Regression model parameters</th>
<th>Relative square error target time series</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline method</td>
<td>univariant</td>
<td>0.316 0.318</td>
</tr>
<tr>
<td>multivariate</td>
<td></td>
<td>0.543 0.548</td>
</tr>
<tr>
<td>the proposed multivariate method</td>
<td></td>
<td>0.179 0.175 0.174 0.174 0.184 0.179 0.196 0.199</td>
</tr>
<tr>
<td>assisting time series lag</td>
<td>Lag = 364 Lag = 30</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6.4.4 Results and Discussions

Prediction results for day 1551 to day 1970 for throughput (target time series) using demand as the assisting time series are shown in Table 6.3. It shows that, for all methods, year-long historical throughput gives better results than month-long historical throughput. This is because the largest period of the time series is one year. The proposed multivariate method outperforms the baseline methods. In addition, for the proposed multivariate method, using 4/5-day historical demand as the assisting time series gives the highest prediction accuracy. The best results are plotted in Figure 6.15.

We conduct the same experiment for late delivery. The minimum RSE from the baseline methods is 0.713, whereas the best performance of the proposed multivariate method is 0.386. It can be seen that, more data points from the assisting time series cannot guarantee a higher prediction accuracy. Including unnecessary features can even deteriorate model accuracy. Rather than including more data points, lags of the target time series as well as the assisting time series should be selected based on the decomposition and cross-correlation analysis results. Demonstrated through
a large multi-national mass customization enterprise real-life data, the length of the target time series should be its largest period, and the length of an assisting time series should be no less than its number of leading lags with respect to the target time series.

6.5 Conclusion

In this chapter, we have proposed an enterprise service-level performance analysis and prediction system. Prediction results using this system for a large multi-national mass customization enterprise real-life data have been compared to baseline methods.

It can be seen that, although time-series prediction using machine-learning algorithms have been studied and successfully applied in many applications, the advantages of combining statistical analysis results were not fully investigated. The proposed system has demonstrated the importance of thorough analysis of correlated time series for making mid-term and short-term time-series predictions. It
clearly outperforms stand-alone machine-learning based methods.

The advantages of the proposed system and the reasons for superior performance compared to baseline methods can be summarized as follows:

- Hierarchical season-trend decomposition helps to get a better understanding of the time-series data structure. By performing hierarchical decomposition, the proposed method greatly reduced unpredictable components from the original time-series data by removing noise components. This feature has significantly upon the performance of state-of-the-art (baseline) methods in mid-term time-series prediction.

- Hierarchical decomposition results also helps to perform and validate cross-correlation analysis between different time series. Cross-correlation information reveals cause-and-effect relationships between different time series.

- A multivariate time-series method uses additional useful information, therefore, it outperforms a univariate method. In the proposed multivariate short-term time-series prediction method, cross-correlation results have been used to determine the amount of historical data to be used by the regression model. Providing only the necessary information improves both algorithm efficiency and model accuracy.
Conclusion and Future Work

7.1 Thesis Contributions

This thesis has presented operation optimization and knowledge discovery solutions for an EIS. Topics covered include production workflow optimization, operation scheduling, resource allocation, process-execution time and process status prediction, order/service fulfillment prediction, and enterprise service-level performance analysis and prediction. In contrast to state-of-the-art methods, which are based on stand-alone statistical methods, analytical methods, or machine-learning algorithms, this thesis has focused on designing unified real-time and data-driven applications that integrate statistical methods and machine-learning algorithms. Therefore, correlated objectives of an EIS can be optimized in a comprehensive data-driven framework.

A number of realistic issues have been considered for data-driven operation optimization and knowledge discovery techniques of an EIS. Due to heterogeneity of resources, stochastic resource malfunctions, time-varying resource queue status, and diverse process attributes, a fundamental design challenge is the estimation of process-execution time. We have presented two different techniques to solve the
this challenge. First, a risk-aware process-execution time estimation model has been built to incorporate heterogeneous, transient, and stochastic factors. The reliability of this model is based on thorough understanding and highly accurate simulations of a real enterprise production environment. Second, a machine-learning and statistical-analysis integrated model has been built to perform the prediction. Such data-driven approach enables more flexibility by releasing the need of establishing parametric relationships between model features. It also overcomes the limitation caused by making assumptions as opposed to stand-alone statistical methods.

Another key design challenge is the realization of real-time and simultaneous scheduling and resource allocation for a large number of heterogeneous processes, which has been intensified by the increase in volume and diversity of demands. An evolutionary algorithm based production scheduler has been designed to address this challenge. It first acquires optimal order dispatching priority at the global level and next obtains detailed process priority and resource allocation at a lower level of granularity. The production scheduler handles large volume of orders in a distributed manner and deals with new orders with minimal impact on existing solutions. With superior performance compared to current solutions, the production scheduler has highlighted its potential benefit in reducing or eliminating production inefficiencies and enhancing the productivity of an enterprise.

The event-log databases of an EIS record major activities occurred on the service and production environment. It is of great value to the realization of different knowledge discoveries for an enterprise.

First, accurate predictions of both process-execution time and process status are crucial for the development of an intelligent EIS. We have proposed new process-execution time-prediction and process status-prediction methods for an EIS by integrating statistical methods with machine-learning algorithms. Comparison results obtained from the real-life data of an enterprise show that the proposed time-
prediction method reduces both the relative mean error and the root-mean-squared error of predictions. Furthermore, the proposed status-prediction method not only achieves higher classification accuracy than both state-of-the-art methods, it also estimates the probability of the predicted status. In addition, algorithm development and training phases of the proposed methods do not rely on any arbitrary predictive horizon. Therefore, a single time-prediction model as proposed is sufficient for status prediction as opposed to a baseline status-prediction method that requires classification models for all potential predictive horizons.

Second, optimization of order-admission or service-establishment processes for an enterprise requires accurate prediction of order life cycle and fulfillment status. We have developed an intelligent order-admission framework that provides admission decisions in real-time for new orders using machine-learning and decision-integration techniques. The framework consists of three classifiers: Support Vector Machine (SVM), Decision Tree (DT), and Bayesian Probabilistic Model (BPM). The classifiers are trained by history orders and used to predict completion status for new orders. A decision integration technique is implemented to combine the results of the classifiers and predict due dates. Experimental results derived using real enterprise data show that the order completion-status prediction accuracy is significantly improved by the decision-integration strategy. The proposed multi-classifier model also outperforms a stand-alone regression model.

Finally, superior enterprise service-level performance is the collective goal of data-driven applications in an EIS. Analytical and predictive models of enterprise service-level time series within an EIS can provide meaningful insights into potential business problems and generate guidance for appropriate solutions. We have proposed a new univariate method in dealing with mid-term time-series prediction. The proposed method first analyzes the hierarchical periodic structure in one time series and decomposes it into trend, season, and noise components. By discarding the noise
component, the proposed method only focuses on predicting repetitive season and smoothed trend components. As a result, this method significantly improves upon the performance of baseline methods in mid-term time-series prediction. Moreover, we have proposed a new multivariate method in dealing with short-term time-series prediction. The proposed method utilizes cross-correlation information derived from multiple time series. The amount of data taken from each time series for training the regression model is determined by results from hierarchical cross-correlation analysis. Such a data-filtering strategy leads to improved algorithm efficiency and prediction accuracy. In conclusion, by combining statistical methods with advanced machine-learning algorithms, we have achieved a significantly superior performance in both short-term and mid-term time-series predictions compared to state-of-the-art (baseline) methods.

7.2 Future Work

The contents of this thesis open up more exciting research directions for the development of an intelligent EIS. In this section, we describe proposed research on automatic knowledge discovery and production planning for PSPs. Analysis of order information, factory status information, and event logs can yield important insights that can help PSPs to improve the quality of their services, enhance production efficiency, reduce extra costs, and increase profits. The proposed methods are also potentially applicable for other enterprises that face similar optimization problems [164] [165].

Our focus is on problems that currently heavily rely on ad hoc rule-based methods or manual work. They are becoming obstacles for PSPs as they strive to improve their performance. The diverse and transient situation in an enterprise, the huge amount of data, and the availability of computational power all drive the PSPs to search for advanced solutions. The optimization methods we propose here are built on state-of-the-art knowledge discovery, machine learning, data mining, and statisti-
cal analysis techniques. Their strengths are leveraged so that novel applications can be realized, real-time and insightful recommendation can be provided, and decision-making for PSPs can be automated. Based on our preliminary research work, our proposed innovations can, at an early stage, provide automatic planning of production activities and other recommendations.

Section 7.2.1 describes a dynamic production planning recommender. It deploys decision mining and machine learning techniques in order to recommend, for each new order, a production workflow that overcomes the limitations of static methods such as matching orders with templates. Section 7.2.2 describes a late-order features ranking engine. It predicts the key features that might lead to a late delivery, and further provides useful information for the PSPs to improve on these key features. Section 7.2.3 describes the research plan for a production workflow conformance checking engine. It includes the mining process of a production workflow through event logs and comparison of workflows. Section 7.2.4 describes the research plan for a worker performance-evaluation recommender. It includes process mining, social network mining, and graph comparison techniques.

7.2.1 Topic 1: Dynamic Production Planning

In today’s EIS, a new service type needs to be dynamically assigned a production workflow. In current implementations, this is done manually on the basis of human expertise or by searching similar services and matching a new service with a pre-designed workflow template. In such traditional methods, the large quantity of history of service data is not utilized to provide automatic guidance. We therefore propose a production planning recommender that identifies, recommends, and guides the EIS towards the best production plan for any incoming-new service based on decision mining [166] and structured-output SVM machine-learning techniques [167].

We first describe the problem in more detail and then discuss the proposed solu-
Problem Statement

Overall Production Workflow: An EIS has an overall production workflow that can be represented by a directed graph $G = (V, E)$ to show all possible production workflow instances. The vertices in $V$ are different types of processing activities and the edges in $E$ determine the precedence relations among processing activities. Every service that has been successfully produced in an enterprise has gone through certain paths inside the overall production workflow. For each service, certain paths to take are determined by the requirement of the service. We will analyze how service attributes influence the choices made in the process of assigning a production plan based on the history of services that have been successfully realized by the enterprise. Figure 7.1 is an illustration of RPI’s partial production workflow.

Decision Point: Each fork in the overall production workflow is a decision point. Based on the attributes of a service and the history of successful completion, our goal is to determine how decisions are made for a service to follow one or more branches. Therefore, after obtaining the overall production workflow, we need to identify each decision point on the graph. The depth-first search (DFS) [168] algorithm can be applied to identify all the forks in the overall production workflow $G = (V, E)$. Figure 7.2 shows an example of a decision point.

Proposed Solution

In our proposed optimization work, we will convert each decision point to a classification problem. Advanced classification algorithms can be then applied to solve this problem. The steps in the proposed solution are listed below.
Figure 7.1: An illustration of RPI’s partial production workflow.

- **Analysis of Decision Points**

Decision-point analysis aims at the detection of service-attribute dependencies that affect the routing of a service to certain paths of the production workflow. Every decision point is analyzed, and if possible, linked to properties of individual services. The main idea is to convert every decision point into a classification problem, where the classes are the different branches of the production workflow that can be followed. Figure 7.3 shows the mapping of selection of workflow branches into classification of
**Figure 7.2:** An illustration of a decision point in the overall production workflow.

**Figure 7.3:** Converting a decision point to a classifier.

different classes, where the entire Service Set is composed of Service Set 1, Service Set 2 and Service Set 3.

- **Building Classification Model**

The classification model for each classifier is built as follows:

- Input: service attribute vector $\mathbf{x}$.

- Output: class label $y$.

- Schema: Each branch is converted into a class label; moreover, if more than one branch is involved in an service’s production workflow, then the combination of these branches is labeled as a new class. For example, for Figure 7.3, the corresponding
class labeling strategy is shown in Table 7.1.

- *Selection of Classification Algorithm*

Once the classification model is established, the next challenge is to select an appropriate classification algorithm. We propose to use structured-output SVM, a powerful classification algorithm that can be applied to problems with structured class labels such as trees, sequences, or sets [167]. It learns a mapping from input vectors or patterns $\mathbf{x} \in X$ to discrete response variables $\mathbf{y} \in Y$, based on a training sample of input-output pairs $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n) \in X \times Y$, where $X$ and $Y$ are the input and structured output spaces, respectively. Unlike multiclass classification, where the output space consists of an arbitrary finite set of labels or class identifiers, or regression where the response variable is a scalar, the output of structured output SVM can be more complex and convey more information.

The advantages of using structured-output SVM can be explained as follows. If we apply traditional classifiers that predict simple discrete class labels (as shown in Table 7.3), then the number of different branches at one decision point cannot be too large, otherwise the number of possible classes increases exponentially. Suppose the number of branches is $n$ then the number of possible classes can be as large as $\sum_{i=1}^{n} \binom{n}{i} = \mathcal{O}(2^n)$. On the other hand, if we convert discrete class labels into a vector as shown in Table 7.2, the length of the structure output vector $\mathbf{y}$ is always $n$. The standard structured-output SVM primal formulation is to minimize the following
Table 7.2: Transformation of class labeling strategy (Figure 7.2) into structured output.

<table>
<thead>
<tr>
<th>Branches that are in an service’s production workflow</th>
<th>Class label</th>
<th>Structured output</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Service Set 1</td>
<td>y = (1, 0)</td>
</tr>
<tr>
<td>B</td>
<td>Service Set 2</td>
<td>y = (0, 1)</td>
</tr>
<tr>
<td>B and C</td>
<td>Service Set 3</td>
<td>y = (1, 1)</td>
</tr>
</tbody>
</table>

regularized risk function:

$$\min_{w, \xi} \|w\|^2 + C \sum_{n=1}^{l} \xi_n$$  \hspace{1cm} (7.1)

such that $w^T \Psi(x_n, y_n) - w^T \Psi(x_n, y) + \xi_n \geq \Delta(y_n, y)$, $n = 1, ..., l, \forall y \in Y, \xi_n \geq 0$, where we are given a set of $l$ training service instances $(x_n, y_n) \in X \times Y$, and $\xi_n$ is the slack variables for each training sample.

- **Planning Workflow for New Services**

For every new service, based on the service attributes, this service can be automatically assigned a certain production workflow following the steps listed below:

1. Construct the attribute vector of this service $x$.

2. Start from the root node of the overall production workflow $G$. The new service attribute vector will go through the first decision point, and based on the selection of a branch at the first decision point, some (and not all) of the remaining decision points in the overall production workflow will be visited by this new service attribute vector. If the service input vector is $x$, the prediction function is:

$$f(x) = \arg \max_{y \in Y} w^T \Psi(x, y)$$  \hspace{1cm} (7.2)

The resulting class vector is the one that maximize the value of $f(x)$ [167].

As shown in Figure 7.4, the final production workflow planned for this service will be $A \rightarrow B \rightarrow D \rightarrow E$. Decision points 1 and 2 are visited, whereas decision point 3 is never reached since branch $A \rightarrow C$ is not selected at the first decision point.
Detecting Outliers

Not all the new services can be matched to a production workflow. If the likelihood of classifying a new service into any branch is below a certain threshold $th$, e.g., $th = 40\%$, at any decision point, then this new service can be deemed to be new (not handled before) for the factory. The production planning recommender should provide this new service to a human manager so that a new production workflow pattern can be specifically designed to produce this service.

7.2.2 Topic 2: Ranking of Attributes of Late Orders

This part of the proposed research is based on the preliminary work of order lead-time prediction. Once an order is predicted to be late by the decision fusion center discussed in Section 5.2, the EIS should be able to generate a ranking of order attributes that can potentially lead to the late delivery. The attributes that may cause late delivery are referred to as late-order attributes. Other than tight due dates, factors such as high print-quality requirement are also key attributes that cause late delivery.

The state-of-the-art today relies on judgment driven by human expertise [23]. Human experience does not however always reflect the most up-to-date situation.
in an enterprise. Our proposed solution is based on knowledge discovery of history data and up-to-the-moment performance of the factory. Moreover, after identifying late-order attributes, the PSP can focus on solving the limitations caused by the key attributes that have been frequently identified as being bottlenecks.

**Problem Statement**

**Attribute Vector:** In order to predict the completion status of a new order, two sources of information are essential. One is order attributes, which comprehensively define an order. The other is the factory status information acquired at the moment when this order is accepted. Every order can be represented by an attribute vector, which we call *OrderAttributes*. The factory status information can be represented by a vector *FactoryStatus*, which includes key information about the factory, for example, the work-in-progress in an enterprise, the number of queued jobs at the departments, the number of available resources at the departments and so on.

**Transformation:** Some machine-learning techniques such as SVM require that each data instance is represented as a vector of real numbers [130]. Hence, in order to apply the same data set to all kinds of machine-learning algorithms, we convert categorical attributes into numerical format. One solution is to use a vector of length *m* to represent an *m*-category attribute. Only one of the *m* numbers is one, and others are zero. For example, a three-category attribute such as \{glossy, silk, linen\} for book cover can be represented as \{(0,0,1), (0,1,0), and (1,0,0)\}. Therefore, after the transformation, the entire order attribute vector consists of two parts, the first part describes this order attributes, and the second part describes the factory status. In other words, the order instance vector \(x\) can be described as:

\[
x = (\text{OrderAttributes, FactoryStatus}) = (A_1,\ldots,A_n)
\]

**Variation Range:** Each attribute in its numerically encoded form can take a range
of values, which can be obtained from history data. Note that, after transformation, all the attributes are numerical values. Therefore, for each numerical attribute, there is an upper and lower bound, i.e., \( A_i \leq AU_i \), and \( A_i \geq AL_i \), where \( AU_i \) and \( AL_i \) are the largest and smallest values that \( A_i \) has ever taken in this history. For each \( m \)-category discrete attribute, there is a set that contains \( m \) vectors, i.e., set \( AUL_i = \{ (0, ..., 0, 1), (0, ..., 1, 0), ..., (1, ..., 0, 0) \} \), and \( A_i \in AUL_i \), where each element represents a category.

**Proposed Solution**

**Phase-Change Point:** After a new order is predicted to be late, the next step is to determine the late-order attributes. For a numerical attribute \( A_i \), which takes a values in a range determined by \( AU_i \) and \( AL_i \), if we search through the range, there might be another value that changes the predicted order-completion status from late to on-time. This situation is illustrated in Figure 7.5. A similar situation can exist for a categorical attribute \( A_j \), which takes a discrete vector value in a set. If we enumerate all the possible choices, there might be another value that changes the predicted order-completion status from “late” to “on-time”. This situation is illustrated in Figure 7.6. Therefore, we define a “Phase-Change Point” for both numerical and categorical attributes to be a value that flips the predicted order-completion status from “late” to “on-time”. Note that, for each attribute there might be no phase-change point or multiple phase-change points. Phase-change point is only defined for an order that is predicted to be late.

- **Search phase-change point**

While searching the phase-change points for each attribute, the values of other attributes are kept as their original values in \( x \). For a numerical attribute, only the closest phase-change point to the original attribute value is of interest. We pro-
Figures 7.5: Illustration of a phase-change point for a numerical attribute.

pose to search from the lower bound to the upper bound using a binary search algorithm [169]. For a categorical attribute which is represented by a vector, we enumerate all the alternative vectors and count the number of phase-change points to be \( n \). For example, for the 4-category attribute shown in Figure 7.6, \( n = 2 \) if the original value is represented by vector \((0, 0, 0, 1)\).

- **Compute Normalized Variation Degree**

After obtaining the phase-change points for all the attributes, the next step is to quantify the degree of variation necessary to change the predicted order-completion status.

We define a normalized variation degree (\(NVD\)) for each numerical attribute as follows:

\[
NVD = \frac{|a_i - ap_i|}{AU_i - AL_i}
\]  

(7.3)

where \(a_i\) denotes the original value of attribute \(A_i\), and \(ap_i\) denotes the value of \(A_i\)’s phase-change point as shown in Figure 7.5.

176
Similarly, the $NVD$ for each $m$-category attribute is defined as follows:

$$NVD = \frac{m - n}{m} = 1 - \frac{n}{m} \quad (7.4)$$

where $m$ denotes the number of categories and $n$ denotes the number of phase-change points for attribute $A_i$ as shown in Figure 7.6.

Therefore, the closer the phase-change point to the original numerical attribute value or the more the phase-change points of a categorical attribute, the smaller the value of $NVD$. A less value of $NVD$ signifies a higher likelihood of causing late delivery.

If no phase-change point can be found for an attribute, then $NVD$ is not defined for this attribute. It is then likely that this attribute is not the cause of late delivery since varying the value of this attribute throughout all its possible choices cannot avoid a late-delivery prediction.
Table 7.3: An example of the ranking of late-order attributes for an order.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Property</th>
<th>NVD</th>
<th>Original Value</th>
<th>Phase-Change Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>Numerical</td>
<td>0.1</td>
<td>a</td>
<td>ap1</td>
</tr>
<tr>
<td>A2</td>
<td>Numerical</td>
<td>0.2</td>
<td>a</td>
<td>ap2</td>
</tr>
<tr>
<td>A3</td>
<td>Categorical</td>
<td>0.33</td>
<td>(1, 0, 0)</td>
<td>(0, 1, 0), (0, 1, 1)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>An</td>
<td>Categorical</td>
<td>0.8</td>
<td>(0, 0, 1, 0)</td>
<td>(0, 0, 0, 1)</td>
</tr>
</tbody>
</table>

- **Ranking Late-order Attributes**

Only an attribute with a valid NVD value is recorded as being a late-order attribute. The significance of late-order attributes can be ranked by their NVD. An attribute with the smallest NVD is treated as the most-significant attribute that can cause late delivery since a relatively small change in its value has the highest likelihood of leading to an on-time delivery.

An example of the ranking of late-order attributes for a predicted-to-be-late order is shown in Table 7.3.

- **Recommend the Most Appropriate Alternative Choices**

If a PSP wants to negotiate with a customer about modifying an order’s requirement so that this order can be delivered on time, then the values of the phase-change points can be provided as recommendations for the most appropriate alternative choices. Note that only attributes from **OrderAttributes** can be provided to a customer for negotiation. A customer can choose to vary one or even more attribute requirements, for instance, choose a looser due time or a lower printing quality. A PSP has high confidence in completing the resulting new order contract on time.

- **Analyze Late-order Attributes**

After the accumulation of a sufficient amount of data, statistical analysis techniques can be applied. If a late-order attribute is from **FactoryStatus**, it is the factory’s
responsibility to improve its productivity in order to deliver orders on time. This information is valuable since it helps a PSP to identify bottlenecks in an enterprise that always causes late delivery of orders. For example, if the queue length of a server is always predicted to be reduced to guarantee on-time delivery, then the late-order attributes ranking engine can provide a recommendation to the factory that the quantity of this server should be increased to improve the productivity of the factory.

- **Update Data**

Furthermore, since the situation on the print factory floor is transient, for example, order quantity of a certain type may be boosted during a specific season and the resources on the factory can be upgraded; therefore, order-delivery information is of value for only a limited period of time. Therefore, outdated history data should be eliminated and new data added to guarantee updated training of all order-status prediction classifiers. As long as the prediction of order status always relies on the up-to-date situation, the ranking of late-order attributes can reflect the current situation in an enterprise.

### 7.2.3 Topic 3: Conformance Checking of Production Workflow

An EIS has a designed production workflow. All the production activities of an enterprise are supposed to follow the patterns embedded in the designed production workflow. The designed production workflow is used by the EIS to define and monitor its production capability. However, the actual production activities performed by the enterprise might be different from the designed version. A discrepancy can be caused by either incorrectly processed services or the incompleteness of the designed production workflow [170]. Currently, there is no efficient automatic solution for an EIS to discover such discrepancies. These discrepancies are very subtle, which
makes manual discovery slow and difficult. Our proposed solution allows an EIS to
dynamically discover its actual production workflow and timely update the designed
production workflow such that the designed production workflow is maintained as
a comprehensive and correct version. It is important that the actual production
activities match the designed production workflow, since activities such as worker
responsibility assignment relies on the designed production workflow.

*Proposed Solution*

- **Mine the Actual Production Workflow**

Process mining techniques, such as fuzzy mining technique, can be applied to the
event log in order to get the actual production workflow [171]. A production workflow
is represented by a directed graph as illustrated in Figure 7.7. Each node corresponds
to a different process type, i.e. there is a one-to-one correspondence between a node
and a process type. The arcs represent the series or parallel relations among different
process types. For example, print book block and print book cover can be processed
in parallel, whereas, book cover must be laminated before being trimmed.

There are other process mining algorithms in the literature, such as $\alpha$-algorithm
[172] and heuristics-miner algorithm [173]. We propose to adopt the fuzzy mining
algorithm since it is robust to event logs that contain low-frequency disturbances
of exceptional events [171]. Considering the stochastic nature of the occurrence of
exceptions in an enterprise, fuzzy mining algorithm will be considered as the primary
candidate.

- **Graph Matching of Two Workflows**

Graph matching algorithm can be used to compare the actual production workflow
and the designed production workflow to find out discrepancies [174].

The graph matching model can be established as follows:
Figure 7.7: An example of mined actual production workflow obtained from event log.

Given two graphs: the actual production workflow graph $G_A = (V_A, E_A)$ and the designed production workflow graph $G_D = (V_D, E_D)$. There are two possible situations:

1. When $|V_A| = |V_D|$, i.e., they have the same set of nodes, the problem is to find a one-to-one mapping $f : V_A \rightarrow V_D$ such that $(u, v) \in E_A$ iff $(f(u), f(v)) \in E_D$, when such a mapping $f$ exists, this is called an isomorphism, and $G_D$ is said to be isomorphic to $G_A$, otherwise, any $(u, v) \in E_A$ that cannot be mapped into $G_D$ are discrepancies, for example, an extra edge in $G_A$ or $G_D$.

2. When $|V_A| \neq |V_D|$, the problem is to find a mapping: $f' : V_A \rightarrow V_D$ such that $(u, v) \in E_A$ iff $(f'(u), f'(v)) \in E_D$. This corresponds to the graph embedding
problem. Any vertices from the larger graph and any edges that cannot be mapped by $f'$ are identified as discrepancies, for example, additional nodes and edges in $G_A$ or $G_D$.

- **Identify the Causes of Discrepancies**

Discrepancies between $G_A$ and $G_D$ are caused by services that are processed differently from $G_D$. Once a discrepancy is found by graph matching process, which can be represented by an extra node $v'$ or an extra edge $e' = (u', v')$ in $G_A$ or $G_D$, events that correspond to the process types represented by the $u'$ and $u'$ can be pulled out from the event log. From these events the corresponding services that caused a discrepancy can be identified by service ID.

If the extra nodes and edges are only from $G_D$, they can be temporarily viewed as being consistent, since it is possible that the event log being used to mine $G_A$ does not include any service that involves process types or process paths that map to the extra nodes or edges.

- **Confirm the Causes of Discrepancies**

In order to confirm the identified causes of discrepancies, we can apply process-mining algorithm over a modified version of the event log that has eliminated events related to discrepancies. After excluding such events, we expect that the newly mined production workflow $G'_A$ should match the designed production workflow $G_D$ or at least exclude the discrepancy found before.

- **Update the Designed Production Workflow**

There are two causes of discrepancies. The first corresponds to services that are incorrectly processed in an enterprise. Such services have exception events recorded under their service ID. Information of these services is valuable because it can be further analyzed to find out the cause of exceptional events, for instance, a certain
type of paper material always causes paper jam and requires reprocessing. The other cause is the incompleteness of the designed production flow. If the identified services related to discrepancies do not have any exceptional or rework events associated to them, then these successfully manufactured services show that the designed production workflow can be expanded to include the production workflow involved in producing these services. The designed production workflow can be automatically updated to a comprehensive and accurate version.

7.2.4 Topic 4: Worker Performance Evaluation

Almost all the production activities on a print factory floor are executed or monitored by workers on different resources. The activities of workers are recorded in the event log. The print factory also has a planned work assignment and minimum expectation for each worker [23]. The evaluation of worker performance is difficult. Traditional evaluation methods of worker performance involve many subjective and biased judgments, which are neither convincing or accurate [175].

Event log records the actual activities occurred in an enterprise, the knowledge discovered from this data set, such as the handover-of-work relations among workers can be viewed as reliable under the assumption that the event log is comprehensive and accurate [176]. From the event log, the knowledge about the performance of workers can be mined out by applying a combination of process-mining and social-network-mining techniques [177].

Automatic worker performance evaluation is still an open problem for PSPs. Our proposed solution provides a PSP with useful recommendations to evaluate the performance of its workers.
Problem Statement

**A Combination of Production Workflow and Worker Information:** An overall production workflow can be mined out based on the event log by process mining techniques as described in proposed research Section 7.2.3. Since events also have worker ID information associated with, we can also include worker-participation information in the mined production workflow. This combined version of production workflow and worker information reflects the causal relations among different process types and the importance of the worker. Shown in Figure 7.8 is a directed graph of a simplified production workflow. Figure 7.9 is an extension, since it includes the worker information into the production workflow shown in Figure 7.8. Different workers are represented by different colors. The pie chart shows the weights of worker participation measured in terms of event frequency. For example, 50% in the pie chart in process 1 means that among all the events corresponding to process 1, half of them are executed by the same worker (colored in dark green).
Figure 7.9: Production workflow in Figure 7.8 combined with worker information.

Proposed Solution

- Extract Information Related to Worker Performance

A workflow graph is an abstract model of production activities at the system level. We define a complete workflow graph $G_C$, i.e. the graph mined through a complete version of the event log that includes the activities of every worker. A partial workflow graph $G_P$ is mined through a partial version of the event log that deliberately excludes the activities of a specific worker whose performance is being evaluated. Note that, there is an unique version of $G_C$, whereas a specific $G_P$ is generated for each worker that is being evaluated. Since the production-activity records related to a worker have been removed from the complete event log, $G_P$ can be different from $G_C$.

The participation of a worker can influence the the flow and efficiency of the
production activities in different ways. We illustrate two situations here that can be reflected by the differences between the $G_C$ and a $G_P$.

(1) Break the workflow

While comparing the complete workflow graph $G_C$ and a partial workflow graph $G_P$, a break of the workflow is defined as the absence of an edge $e = (u, v)$ that only present in $G_C$ but absent in $G_P$. In other words, production process $u$ and $v$ no longer have a precedence constraint defined by an edge from $u$ to $v$ in $G_P$, whereas in $G_C$ it is defined by edge $e$ that process $u$ is the preceding step of process $v$.

If a break occurs, it means that products can no longer be handed over from one process to another process. It also reveals that the worker who is being evaluated performs the handover-of-work task from process $u$ to process $v$. The influence of breaking the workflow can be deemed as of high impact to the factory since the manufacturing process of some products is disabled. Therefore, the worker being evaluated can be marked as a crucial worker. The worker performance-evaluation recommender can provide such information to human managers. Subsequently, decisions such as training another worker to perform the same task can be made.

An example of the discovery of a crucial worker is shown in Figure 7.10. It is shown that excluding activities performed by the worker being evaluated breaks the workflow connection from process 1 to process 2.

(2) Change the process parameter

An important parameter of a production process is the unit processing time $t$, which is defined as the average time used to process a unit workload. A unit workload is the minimum amount of workload for a process. For example, a unit workload for a binder is two products and the unit processing time of the binding process is 42 seconds. In other words, the least amount of binding workload takes 42 seconds to complete.

Since information of event timestamp (i.e. the start or end time) and workload
quantity is being recorded in the database and can be written in the event log, we can compute the unit processing time and include it in both $G_C$ and $G_P$ as a parameter associated to each node. Other than changing the pattern of workflow graph, the activities of a worker can influence the unit processing time of a process.

If excluding activities of the worker being evaluated reduces the unit processing time of a process, the worker performance-evaluation recommender can provide a piece of information showing that this worker’s efficiency is below the average level since without the participation of this worker, the unit processing time can be reduced. An example situation is shown in Figure 7.11, where excluding activities of the worker being evaluated does not influence the pattern of the workflow; however, it changes the unit processing time of process 2 from two hours to one hour.

Furthermore, there are other possible situations that the participation of a worker can differentiate $G_P$ from $G_C$, such as causing the absence of a node that corresponds to a certain process type. The influence of a worker can be a combination of differ-

**Figure 7.10:** An example of the discovery of a crucial worker.
Figure 7.11: An example of a process parameter being influenced by a worker’s participation.

The proposed research work aims to tackle several open problems in the print factory management area. In summary, the specific problems to be investigated are as follows:

1. Dynamic production planning, which assigns new orders production workflows by utilizing decision-mining and machine-learning techniques.

2. Ranking the late-order attributes by their significance and providing recommendations for the most appropriate alternative choices.
3. Conformance checking of the actual production workflow with the designed production workflow and updating the designed workflow if discrepancies are identified.

4. Evaluating worker performance by investigating the variation of the actual production workflow with and without the participation of a worker.
Appendix A

Derivation of Equation (3.3)

When \( N = 0 \), Equation (3.3) equals zero. When \( N \) approaches infinity, Equation (3.3) should also approaches infinity.

Define:

\[
L = \left( \frac{E[X_{N+1}]}{E[X_N]} \right) = \frac{\sum_{i=1}^{N+1} \frac{w_i}{\prod_{k=1}^{N+1}(1-\gamma_k)}}{\sum_{i=1}^{N} \frac{w_i}{\prod_{k=1}^{N}(1-\gamma_k)}} \quad (A.1)
\]

\[
= \frac{1}{1-\gamma_{N+1}} \cdot \left[ \frac{\sum_{i=1}^{N} \frac{w_i}{\prod_{k=1}^{N}(1-\gamma_k)} + \frac{w_{N+1}}{\prod_{k=1}^{N+1}(1-\gamma_k)}}{\sum_{i=1}^{N} \frac{w_i}{\prod_{k=1}^{N}(1-\gamma_k)}} \right] \quad (A.2)
\]

\[
= \frac{1}{1-\gamma_{N+1}} \left[ 1 + \frac{\prod_{k=1}^{N+1}(1-\gamma_k)}{\sum_{i=1}^{N} \frac{w_i}{\prod_{k=1}^{N}(1-\gamma_k)}} \right] \quad (A.3)
\]

\[
> \frac{1}{1-\gamma_{N+1}} \quad (A.4)
\]

\[
> 1. \quad (A.5)
\]

Since Equation (3.3) monotonically increases as \( N \) increases with a rate greater than 1, therefore, Equation (3.3) diverges with \( N \).
Figure A.1: Manufacturing process for a 3-stage single-part product.

An illustration of Figure 3.2 with three sequential tasks is shown in Figure A.1. Variables $X_1$, $X_2$, and $X_3$ represent the time to complete all three tasks, the last two tasks, the last task, respectively. They are related by the follow equations:

\[
E[X_3] = w_3 + \gamma_3 \cdot E[X_1] + (1 - \gamma_3) \cdot 0 \quad \text{(A.6)}
\]
\[
E[X_2] = w_2 + \gamma_2 \cdot E[X_1] + (1 - \gamma_2) \cdot E[X_3] \quad \text{(A.7)}
\]
\[
E[X_1] = w_1 + \gamma_1 \cdot E[X_1] + (1 - \gamma_1) \cdot E[X_2] \quad \text{(A.8)}
\]

by solving Equations (A.6) to (A.8), we can obtain:

\[
E[X_1] = \frac{w_1}{(1 - \gamma_1)(1 - \gamma_2)(1 - \gamma_3)} + \frac{w_2}{(1 - \gamma_2)(1 - \gamma_3)} + \frac{w_3}{1 - \gamma_3} \quad \text{(A.9)}
\]

\[
= \sum_{i=1}^{3} \frac{w_i}{\prod_{k=1}^{3} (1 - \gamma_k)}.
\]

(A.10)

If the number of stages is extended to $N$, by replacing 3 by $N$ in Equation A.10, we can get the same expression as Equation (3.3), which is repeated here:

\[
E[X] = \sum_{i=1}^{N} \frac{w_i}{\prod_{k=i}^{N} (1 - \gamma_k)}.
\]

(A.11)

Therefore, Equation (3.3) holds true for all the values of $N$. 191
Equation (3.3) can be validated by mathematical induction as follows:

In the base case where \( N = 1 \), according to the definition of expectation, the expectation, \( E[X_1] \), is defined by [74]:

\[
E[X_1] = \sum x_i p_X(x_i) \tag{A.12}
\]

\[
= \sum_{q_1=0}^{+\infty} (q_1 + 1) \cdot w_1 \cdot \gamma_1^{q_1} \cdot (1 - \gamma_1) \tag{A.13}
\]

\[
= w_1 \cdot (1 - \gamma_1) \sum_{q_1=0}^{+\infty} (q_1 + 1) \cdot \gamma_1^{q_1} \tag{A.14}
\]

\[
= \frac{w_1}{1 - \gamma_1} \tag{A.15}
\]

where \( x_i \) is given by Equation (B.3) and \( p_X(x_i) \) is given by Equation (B.7). Therefore, Equation (3.3) is correct for \( N = 1 \).

Assume Equation (3.3) to be true for some \( N \geq 2 \). If one more stage is added at the end of the chain, then there are \( N + 1 \) stages. Figure A.2 shows the case when there are \( N + 1 \) sequential stages. The first \( N \) stages are simplified as one stage. Variables \( X_N \) and \( X_{N+1} \) represent the time to complete the first \( N \) tasks and the
time to complete all $N + 1$ tasks, respectively. They are related as follows:

$$E[X_{N+1}] = E[X_N] + w_{N+1} + (1 - \gamma_{N+1}) \cdot 0 + \gamma_{N+1} \cdot E[X_{N+1}]$$ \hspace{1cm} (A.16)

$$E[X_N] = \frac{E[X_N]}{1 - \gamma_{N+1}} + \frac{w_{N+1}}{1 - \gamma_{N+1}}$$ \hspace{1cm} (A.17)

by plugging in Equation (3.3) into Equation (A.17), we get:

$$E[X_{N+1}] = \sum_{i=1}^{N} \frac{w_i}{\prod_{k=i}^{N+1} \left(1 - \gamma_k\right)} + \frac{w_{N+1}}{1 - \gamma_{N+1}}$$ \hspace{1cm} (A.18)

$$= \sum_{i=1}^{N+1} \frac{w_i}{\prod_{k=i}^{N+1} \left(1 - \gamma_k\right)}.$$ \hspace{1cm} (A.19)

Therefore, if Equation (3.3) is true for some $N \geq 2$, it is also true for $N + 1$. Since it has been proved to be true for $N = 1$, by mathematical induction, Equation (3.3) holds true for all the values of $N$. 

193
Appendix B

Derivation of the PMF of Random Variable $X$

Random variable $Q_i$ is the number of failures at task $v_i$. The value of $Q_i$ is denoted by $q_i$. Failures are independent events. The joint probability of the event that $Q_1 = q_1, ..., Q_N = q_N$ is computed by Equation (B.1):

$$P(Q_1 = q_1, ..., Q_N = q_N) = \frac{\left(\sum_{i=1}^{N} q_i\right)!}{\prod_{i=1}^{N} q_i!} \cdot \prod_{i=1}^{N} \gamma_i^{q_i} \cdot \prod_{i=1}^{N} (1 - \gamma_i)^{\sum_{k=i+1}^{N} (q_k + 1)}$$

where $\gamma_i$ is the failure probability of task $v_i$ (defined in Equation (3.2)).

Discrete random variable $X$ is the execution time of a chain of sequential tasks. The value of $X$, which is denoted by $x$, depends on the values of $Q_1$ to $Q_N$ as follows:

$$X = \sum_{i=1}^{N} \left[ \left( \sum_{k=i}^{N} Q_k + 1 \right) \times w_i \right]$$

where $w_i$ is the execution time of task $v_i$ (defined in Equation (3.1)).

Therefore, a combination of values $(q_1, ..., q_N)$ for variables $(Q_1, ..., Q_N)$ can result in a value $x$ for variable $X$. Note that some combinations of $(q_1, ..., q_N)$ can result in
the same value of $X$. Set $\Phi_{X=x}$ contains all combinations such that they all result in the same $x$ for $X$. Therefore, value $x$ is computed as follows:

$$x = \sum_{i=1}^{N} \left[ \sum_{k=i}^{N} q_k + 1 \right] \times w_i, \quad (q_1, ..., q_N) \in \Phi_{X=x}. \quad (B.3)$$

The probability of the event that $X = x$ equals the sum of the probabilities of events $Q_1 = q_1, ..., Q_N = q_N$, for all $(q_1, ..., q_N) \in \Phi_{X=x}$. The PMF of $X$ can be expressed as follows:

$$p_X(x) = P(X = x) = \sum_{(q_1, ..., q_N) \in \Phi_{X=x}} P(Q_1 = q_1, ..., Q_N = q_N). \quad (B.4)$$

When $N \geq 2$, by plugging in Equation (B.1), Equation (B.5) can be written as:

$$p_X(x) = \sum_{(q_1, ..., q_N) \in \Phi_{X=x}} \left[ \frac{(\sum_{i=1}^{N} q_i)!}{\prod_{i=1}^{N} q_i!} \cdot \prod_{i=1}^{N} \gamma_i^{q_i} \cdot \prod_{i=1}^{N} (1 - \gamma_i)^{q_i} \left(\sum_{k=i+1}^{N} q_k + 1\right) \right]. \quad (B.6)$$

whereas when $N = 1$, Equation (B.5) can be written as:

$$p_X(x) = \gamma_1^{q_1} \cdot (1 - \gamma_1). \quad (B.7)$$

When $N = 1$ (i.e., only one task is in the product), value $x = (q_1 + 1) \cdot w_1$, which means $x$ equals multiples of $w_1$. No matter how many times task $v_1$ fails, as soon as it succeeds, the one-task product gets manufactured. Term $\gamma_1^{q_1} \cdot (1 - \gamma_1)$ in Equation (B.7) is the joint probability of the $q_1$ times failures and the one time success.

When $N \geq 2$, value $x$ is a linear combination of $w_1$ to $w_N$ as shown in Equation (B.3). For every combination $(q_1, ..., q_N) \in \Phi_{X=x}$, although they derive the same value $x$, they represent different situations of failures. Therefore, a combination $(q_1, ..., q_N)$ represents a failure situation. For each failure situation, set $\Delta$ contains all failure
sequences under this situation. A failure sequence describes the sequence of failures, i.e., the sequence in which failures occur. In an example when \( N = 2 \), there are two sequential tasks \( v_1 \) and \( v_2 \) that must be executed to manufacture a product. One possible failure situation is \( (q_1 = 2, q_2 = 1) \). It means that task \( v_1 \) fails twice and task \( v_2 \) fails once. Set \( \Delta \) under this situation contains three failure sequences: \( v_1 - v_1 - v_2 \), \( v_1 - v_2 - v_1 \), and \( v_2 - v_1 - v_1 \). Therefore, \(|\Delta| = 3\). They all result in the same value \( x \) by Equation (B.3):

\[
x = (2 + 1 + 1) \cdot w_1 + (1 + 1) \cdot w_2
\]
even though they are different failure sequences.

Failures at the same task have no difference. Term \((\sum_{i=1}^{N} q_i)!\) in Equation (B.6) is the number of failure sequences only if all failures are different. Term \((\sum_{i=1}^{N} q_i)!/\prod_{i=1}^{N} q_i!\) is the actual number of failure sequences. It is derived by excluding indistinguishable failure sequences caused by multiple failures at the same task from term \((\sum_{i=1}^{N} q_i)!\).

In the \( N = 2 \) example, for failure situation \((q_1 = 2, q_2 = 1)\), the number of distinct failure sequences is obtained by:

\[
|\Delta| = \frac{(q_1 + q_2)!}{q_1!q_2!} = \frac{(2 + 1)!}{2!1!} = 3.
\]

According to the rework policy, every failure at task \( v_i \), where \( 1 \leq j < i \leq N \), means a success at task \( v_j \). Term \( \sum_{k=i+1}^{N} q_k \) is the number of successes at task \( v_i \) caused by failures at task \( v_k \), where \( i < k \leq N \). Note that, a product can only be manufactured when all tasks uninterruptedly succeed from the first to the last task. Therefore, task \( v_i \) must succeed at least once. Term \((\sum_{k=i+1}^{N} q_k + 1)\) in Equation (B.6) is the total number of successes at task \( v_i \). Hence term \( \prod_{i=1}^{N} (1 - \gamma_i)^{\sum_{k=i+1}^{N} q_k + 1} \)
is the joint probability of successes at all tasks. Term $\prod_{i=1}^{N} \gamma_i^{q_i}$ in Equation (B.6) is the joint probability of failures at all tasks.

Due to the independence of failures, the probability of any failure sequence in the set $\Delta$ under the same failure situation is the same. A failure sequence implies successes, therefore, term $\prod_{i=1}^{N} \gamma_i^{q_i} \cdot \prod_{i=1}^{N} (1 - \gamma_i)^{\sum_{k=i+1}^{N} q_k + 1}$ is the probability of a failure sequence in set $\Delta$ under failure situation $(q_1, ..., q_N)$.

In summary, when $N \geq 2$, term $(\sum_{i=1}^{N} q_i)!/\prod_{i=1}^{N} q_i!$ is the number of failure sequences under failure situation $(q_1, ..., q_N)$; term $\prod_{i=1}^{N} \gamma_i^{q_i} \cdot \prod_{i=1}^{N} (1 - \gamma_i)^{\sum_{k=i+1}^{N} q_k + 1}$ is the probability of a failure sequence under failure situation $(q_1, ..., q_N)$; and set $\Phi_{X=x}$ contains all failure situations that result in $X = x$. Therefore, the probability of $X = x$ equals the sum of the probabilities of failure sequences under each failure situation in set $\Phi_{X=x}$. It can be calculated by Equation (B.6). Hence Equation (B.5) is the PMF of random variable $X$. 

197
Appendix C

Derivation of Equation (3.4)

C.1 Approximate the distribution of $X$ by an exponential distribution

Discrete random variable $X$ is the execution time of a chain of sequential tasks. As discussed in Section B, Equation (B.5) is the PMF of random variable $X$. Section A discussed that the expectation of $X$ can be computed by Equation (3.3). If task execution time $w_i$ at each stage $i$ is small enough, then the intervals between possible values of $X$ are also small. Therefore, the execution time $X$ can be viewed as a continuous random variable. We observed that, the probability mass function of $X$ can be approximated by the probability density function of an exponentially distributed random variable. The rate of the exponential distribution equals $1/E[X]$. Figure C.1 and Figure C.2 demonstrate such approximation.

C.2 The expectation of the maximum of exponentials

Let random variables $W_1,..., W_H$ be independent exponentially distributed random variables with the same rate $\lambda = 1/E[X]$, $Y = \max(W_1,..., W_H)$. It is proved in [178]
Figure C.1: The probability mass function derived from simulation results (left) and the probability density function approximated by an exponential distribution (right) for a 12-stage chain of sequential tasks with task execution times less than 1 time unit.

that the expectation of $Y$ is given by the follow equation:

\[
E[Y] = \frac{1}{\lambda} \cdot \left(1 + \frac{1}{2} + \ldots + \frac{1}{H}\right) \tag{C.1}
\]

\[
= E[X] \cdot \sum_{j=1}^{H} \frac{1}{j}. \tag{C.2}
\]

Based on the discussed in Section C.1 and a worst case estimation, we approximate the distributions of execution times for parallel parts with an identical exponential distribution whose rate equals $\lambda = \max(1/E[W_j]),$ where $j \in \{1, \ldots, H\}$. Hence we estimate the execution time of all the parts by Equation (3.4), which is repeated
Figure C.2: The probability mass function derived from simulation results (left) and the probability density function approximated by an exponential distribution (right) for a 12-stage chain of sequential tasks with task execution times no less than 30 time units.

Here:

$$E[Y] \approx \max_{j \in \{1, \ldots, H\}} \{E[W_j]\} \cdot \sum_{j=1}^{H} \frac{1}{j}.$$  \hspace{1cm} (C.3)

We can obtain $E[Y]$ from $Y$’s PMF and CDF, which requires the PMF and CDF of each $W_j$. Rigorous computation is too complicated to be realized in real time. Therefore, we adopt Equation (3.4) for simplicity and fast computation.
Support-vector machines (SVMs) [179] have been extensively used for classification and regression [180, 181]. The key idea of an SVM is based on computing a linear regression function in a high-dimensional feature space by mapping the input data via a kernel function. It has been shown that an SVM offers major advantages in high-dimensionality space since its optimization is less dependent on the dimensionality of the input space. Compared to the empirical risk minimization (ERM) principle adopted in neural networks, the structural risk minimization (SRM) principle in SVM has greater generation ability [182]. It is robust in recognizing subtle patterns from complex data sets, even with corrupted data. Applications in pattern recognition [183, 184], vision-based head recognition [157] and vehicle detection [158] have already demonstrated that SVM is rigorous from a theoretical point of view and can lead to superior performance and great potential.

The application of SVM to regression is called support vector regression (SVR). The tutorial in [185] describes the process of selecting algorithms for training SVR models. We are therefore motivated by the many successful applications of SVR
The basic idea of SVM is to find a hyperplane in high-dimensional feature space that can separate instances into two classes. We have found that SVR achieves the highest prediction accuracy for our testing data as compared with other algorithms such as linear regression and neural networks.

We first provide a brief introduction to support vector machine (SVM) in regression approximation [106, 163]. The basic idea of SVM is to map the training data from the input space into a high-dimensional feature space via a function \( \phi \) and then construct a separating hyperplane with maximum margin in the feature space. The set of training data points is denoted as \((x_1, y_1), (x_2, y_2), \ldots, (x_l, y_l)\), where \( x_i \in X \subseteq \mathbb{R}^n \), \( y_i = \pm 1 \) denotes two class labels, and \( l \) is the number of training data.

An SVM will find a hyperplane direction vector \( w \) and an offset scalar \( b \) such that \( f(x) = w \cdot \phi(x) + b \geq 0 \) for positive samples and \( f(x) = w \cdot \phi(x) + b \leq 0 \) for negative samples, which is illustrated in Figure D.1. Therefore, even if a linear function cannot be found in the input space to decide which class the given data belongs to, the function of \( \phi \) can be used to find an optimal hyperplane that can clearly distinguish between instances of two classes.

SVR is derived from the idea of an SVM. The set of training data points is denoted as \((x_1, y_1), (x_2, y_2), \ldots, (x_l, y_l)\), where \( x_i \in X \subseteq \mathbb{R}^n \), \( y_i \in Y \subseteq \mathbb{R} \), \( l \) is the number of training data. These data points are randomly and independently generated from the same distributions. An SVM approximates the decision function using the following

![Figure D.1: The basic idea of SVM is to find a hyperplane in high-dimensional feature space that can separate instances into two classes.](image)
form:

\[ f(x) = w \cdot \phi(x) + b \]

(D.1)

where \( \phi(x) \) represents the high-dimensional feature space that is obtained through a nonlinear mapping from the input space \( x \). The hyperplane direction vector \( w \) and offset scalar \( b \) are estimated by minimizing the regularized risk function (D.2):

\[
\text{minimize} \quad R_{\text{reg}}(f) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{l} L_{\epsilon}(y_i, f(x_i))
\]

(D.2)

where \( L_{\epsilon}(\cdot) \) is the loss function [186]. The second term \((1/l) \sum_{i=1}^{l} L_{\epsilon}(y_i, f(x_i))\) is the empirical error measured by the \( \epsilon \)-insensitive loss function. The loss function provides the advantage of using sparse data points to represent the regularized risk function (D.2). Parameter \( C \) is the regularization factor, which determines the penalty of a data point at the wrong side of the hyperplane, and \( \epsilon \) is called the tube size, which determines the data inside the \( \epsilon \) tube to be ignored in regression. They are user-prescribed parameters and need to be empirically optimized while building the model.

In order to get estimates of \( w \) and \( b \), the regularized risk function (D.2) is transformed to the primal objective function (D.4) by introducing the positive slack variables \( \xi_i \) and \( \xi_i^* \), which are used to measure errors outside of the \( \epsilon \) tube.

\[
\text{minimize} \quad R_{\text{reg}}(f) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{l} (\xi_i + \xi_i^*)
\]

(D.4)
subject to  
\[ y_i - w \cdot \phi(x_i) - b \leq \varepsilon + \xi_i \]
\[ w \cdot \phi(x_i) + b - y_i \leq \varepsilon + \xi_i^*, \quad i = 1, \ldots, l \]
\[ \xi_i^{(*)} \geq 0 \quad \text{(D.4)} \]

where \( \xi_i^{(*)} \) denotes variables with and without *.

Finally, by introducing Lagrange multipliers and exploiting the optimality constraints, the decision function (D.1) has the following explicit form:

\[ f(x) = \sum_{i=1}^{l} (a_i - a_i^*) K(x_i, x) + b \quad \text{(D.5)} \]

In function (D.5), \( a_i^{(*)} \) are Lagrange multipliers. They satisfy the equalities \( a_i \times a_i^* = 0 \), and \( a_i^{(*)} \geq 0 \) where \( i = 1, \ldots, l \), and they are obtained by maximizing the dual of the primal objective function (D.4), which has the following form:

\[ W \left( a_i^{(*)} \right) = \sum_{i=1}^{l} y_i (a_i - a_i^*) - \varepsilon \sum_{i=1}^{l} y_i (a_i + a_i^*) \]
\[ -\frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} (a_i - a_i^*) (a_j - a_j^*) K(x_i, x_j) \quad \text{(D.6)} \]

with the following constraints:

\[ \sum_{i=1}^{l} (a_i - a_i^*) = 0, \quad 0 \leq a_i^{(*)} \leq C, \quad i = 1, \ldots, l. \]

Vector \( w \) is written in terms of data points as:

\[ w = \sum_{i=1}^{l} (a_i - a_i^*) \phi(x_i) \quad \text{(D.7)} \]

Function \( K(x_i, x_j) \) is defined as the kernel function [1]. The value of the kernel is equal to the inner product of two vectors \( x_i \) and \( x_j \) in the feature space \( \phi(x_i) \) and...
Table D.1: Common kernel functions [1].

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Function $K(x_i, x_j)$</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>$x_i \cdot x_j$</td>
<td></td>
</tr>
<tr>
<td>Polynomial</td>
<td>$(x_i \cdot x_j + 1)^d$</td>
<td>$d$</td>
</tr>
<tr>
<td>Gaussian</td>
<td>$\exp\left(-\frac{1}{\sigma^2}(x_i - x_j)^2\right)$</td>
<td>$\sigma$</td>
</tr>
<tr>
<td>Radial Basis Function (RBF)</td>
<td>$\exp\left(-\gamma(x_i - x_j)^2\right)$</td>
<td>$\gamma$</td>
</tr>
</tbody>
</table>

ϕ($x_j$), that is, $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$. The reason for applying kernel function is that SVR can perform a dot product in high-dimensional feature space using low-dimensional space data without knowing the transformation $\phi$. Any function that satisfies Mercer’s condition [126] can be used as the kernel function.

Some commonly used kernel functions are shown in Table D.1. We experimented with these kernels and the RBF kernel demonstrated the best performance.

After solving the value of $w$ in terms of the Lagrange multipliers, variable $b$ can be computed by applying the Karush-Kuhn-Tucker (KKT) conditions [187]. Based on the KKT conditions, only a few coefficients $(a_i - a_i^*)$ in decision function (D.5) have nonzero values, and the corresponding training data points have approximation errors equal to or larger than the tube with radius $\varepsilon$, and these data points are referred as support vectors. For all points inside the $\varepsilon$ tube, the Lagrange multipliers that equal to zero do not contribute to the decision function. Lagrange multipliers may be nonzero values and used as support vectors only if the requirement $\varepsilon_i \leq |f(x_i) - y_i| \leq \xi_i$ is satisfied (shown in Figure D.2). In this case, the product of the Lagrange multipliers and constrains has to be equal to 0.

$$a_i (\varepsilon + \xi_i - y_i + (w, x_i) + b) = 0$$
$$a_i^* (\varepsilon + \xi_i^* + y_i - (w, x_i) - b) = 0$$
$$(C - a_i^{(s)})\xi_i^{(s)} = 0$$ (D.8)
Figure D.2: SVR to fit a tube with radius $\varepsilon$ to the data and positive slack variables $\xi$ measuring the data points lying outside of the tube.

Since $a_i^{(s)} = 0$, and $\xi_i^{(s)} = 0$ for $a_i^{(s)} \in (0, C)$, $b$ can be computed as:

$$b = y_i - (w, x_i) - \xi_i \quad \text{for} \quad a_i \in (0, C)$$

$$b = y_i - (w, x_i) + \xi_i \quad \text{for} \quad a_i^{(s)} \in (0, C). \quad (D.9)$$

Therefore, as seen from the decision function (D.5), only the support vectors are used to determine the function value, as the values of $(a_i - a_i^{(s)})$ for the other training data points are equal to zero. This highlights the main advantage of SVR, i.e., support vectors are usually only a small subset of the training data points; this is referred to as the sparsity of the solution.

In summary, in order to train a SVR model, we need to optimize the kernel function $K(\cdot)$, the regularization factor $C$, and the tube size $\varepsilon$ through the procedures described in tutorial such as [185].
Bibliography


cost and resource constraints.” IEEE International Conference on Industrial Informatics, 2011.


[149] Z. Li and J. Han, “Mining periodicity from dynamic and incomplete spatiotemporal data,” in *Data Mining and Knowledge Discovery for Big Data*. Springer, 2014, pp. 41–81.


Biography

• Qing Duan

• Place of birth: Yang Ling, Shaan Xi, China

• Date of birth: October 21st, 1986

EDUCATION

• Ph.D., Duke University, Durham, NC, - present.

• B.S., Xi’an Jiaotong University, China, 2008.

PUBLICATIONS


PATENT


