August 2023

A Digital Twin Framework for Production Planning Optimization: Applications for Make-To-Order Manufacturers

Ron Mallach
University of Massachusetts Amherst

A DIGITAL TWIN FRAMEWORK FOR PRODUCTION PLANNING OPTIMIZATION: APPLICATIONS FOR MAKE-TO-ORDER MANUFACTURERS

A Dissertation Presented

by

RON MALLACH

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

MAY 2023

Industrial Engineering & Operations Research

A DIGITAL TWIN FRAMEWORK FOR PRODUCTION PLANNING OPTIMIZATION: APPLICATIONS FOR MAKE-TO-ORDER MANUFACTURERS

A Dissertation Presented

by

RON MALLACH

Approved as to style and content by:

Ana Murial, Chair

Chaitra Gopalappa, Member

Senay Solak, Member

Sundar Krishnamurty, Department Chair
Industrial Engineering & Operations Research

DEDICATION

This dissertation is dedicated to my family. To my father, mother, brother and sister. Thank you for your endless love, support and encouragement.

ACKNOWLEDGEMENTS

I would not have been able to complete this doctoral thesis without the help and support of the very kind people around me. First and foremost, I'd like to express my gratitude to my advisor and committee chair, Professor Ana Muriel, for her guidance throughout the development of this dissertation and for trusting me as a research assistant during my B.S, M.S, and Ph.D. studies at UMASS. I have learned so much from her and am grateful for the numerous opportunities and relationships I have been introduced to by her. I would not be in the position I am today without her.

Next, I would like to thank the express my gratitude to my thesis committee members, Chaitra Gopalappa and Senay Solak, for their time investments, encouragement and insightful comments throughout my graduate program. A special thanks as well to Michael Prokle, a great mentor to me academically and professionally. I'd also like to thank Raj Subbu, Ted Ackworth, and Vivek Saxena who provided me the opportunity to collaborate in the research efforts which culminated in the work represented in this thesis. Further, I'd like to thank those who supported me professionally during my studies. Thank you Kevin Coffey, Cindy Hulse, and David Mintz. Your guidance and patience with me over the years is much appreciated and has allowed me to complete this thesis.

ABSTRACT

A DIGITAL TWIN FRAMEWORK FOR PRODUCTION PLANNING OPTIMIZATION: APPLICATIONS FOR MAKE-TO-ORDER MANUFACTURERS

MAY 2023

RON MALLACH

B.S., UNIVERSITY OF MASSACHUSETTS AMHERST

M.S., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Ana Murial

In this dissertation, we develop a Digital Twin framework for manufacturing systems and apply it to various production planning and scheduling problems faced by Make-To-Order (MTO) firms. While this framework can be used to digitally represent a particular manufacturing environment with high fidelity, our focus is in using it to generate realistic settings to test production planning and scheduling algorithms in practice. These algorithms have traditionally been tested by either translating a practical situation into the necessary modeling constructs, without discussion of the assumptions and inaccuracies underlying this translation, or by generating random instances of the modeling constructs, without assessing the limitations in accurately representing production environments. The consequence has been a serious gap between theory advancement and industry practice. The major goal of this dissertation is to develop a framework that allows for practical testing, evaluation, and implementation of new approaches for seamless industry adoption. Throughout this dissertation, we emphasize the importance of the underlying scheduling problems which provide the basis for additional operational decision making. We focus on the computational

evaluation and comparisons of various modeling choices within the developed frameworks, with the objective of identifying models which are both effective and computationally efficient.

In Part 1 of this dissertation, we consider a class of Production Planning and Execution problems faced by job shop manufacturing systems. We are motivated by a collaboration with our industrial partner, a consultant firm supporting manufacturers in the aerospace supply chain. Manufacturers in this sector must generate production plans that span months in order to account for the long lead times experienced in the sector.

In the second chapter, we establish and validate a Digital Twin framework that acts as the test-bed for the application and evaluation of all the planning and scheduling formulations considered throughout this dissertation. We develop this framework as a modular software package and emphasize the practicality and configurability of the framework, such that minimal modelling effort is required to apply the framework to a multitude of optimization problems and manufacturing systems. Specifically, we develop: 1) algorithms capable of generating realistic problem scenarios faced by large-scale, production facilities, 2) heuristics capable of translating production plans into executable schedules, and 3) advanced analytics and visualizations for evaluating the resulting scenarios and schedules. In particular, we consider the practical nuances found in the aerospace manufacturing industry.

In the third chapter, we consider the Multi-Level Capacitated Lot Sizing Problem (MLCLSP), and develop an integrated solution procedure that: generates a production plan for an extended planning horizon, associates production decisions with specific customer sales orders, and creates an executable schedule which can be followed on the shop floor. The formulations we develop leverage novel modeling techniques, and include important practical considerations, which to our knowledge, have not been addressed in the literature of MLCLSP. We evaluate the proposed formulations and integrated solution procedure against several benchmark cases. We show our proposed approach to result in significant savings relative to an MRP-based implementation. Counter to intuition, a linear programming relaxation of the MLCLSP performs very well when tested in practice, indicating that it may be sufficient to guide large-scale practical operations avoiding the computational burden and limitations of MIP.

In Part 2 of this dissertation, we consider a class of scheduling problems faced by manufacturers whose production system is dominated by a single operation. We are motivated by a collabora-

tion with our industrial partner, a boutique manufacturer specializing in the production of mass customizable mosaic murals.

In the fourth chapter, we develop and compare several representations of the scheduling problem for the unrelated parallel machine problem and the flexible flow shop problem. The classes of scheduling formulations we consider are referred to as direct-positional and relative-positional scheduling models because sequence-dependent setup times are important in our industrial context. We extend the research of Ekin Koker [93] by building on the formulations and initial testing. Our contribution includes the generalization of the formulations presented by Koker [93], the development of an efficient implementation of the developed models using a Python/Gurobi interface, and an enhancement to the computational evaluation across a variety of practical manufacturing settings.

In the fifth chapter, we extend the scheduling problem from the fourth chapter to consider the implications of a class of uncertainties which stem from the demand generation process inherent to many MTO firms. This demand generation process is characterized by the back-and-forth negotiations between the firm and customer prior to the realization of demands. The existence of known demands which have not yet been confirmed, referred to as *contingent* demands, acts as a critical source of uncertainty for the firm. Despite being common place in practice, contingent demand is largely ignored in the literature. To address this, we develop several variations of direct-positional and time-indexed formulations for the unrelated parallel machine problem, subject to contingent demand, and evaluate their effectiveness and efficiency. In particular, we find that a novel hybrid time-indexed formulation in two timescales provides a good balance between solution accuracy and computational complexity. We formulate extensions to these scheduling problems to consider a suite of Revenue Management problems including the Order Acceptance & Scheduling Problem.

The sixth chapter summarizes our findings and concludes the dissertation highlighting our contributions and pointing to general future research directions.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1
INTRODUCTION

Today's supply chain and production facilities are becoming increasingly complex and manufacturers need to adapt to ever-increasing customer expectations, rising resource costs and an unprecedented level of uncertainty [94]. Recent advances in enabling technologies such as cloud computing and the Internet of Things as well as the maturation of information systems such as Transport Management Systems (TMS), Customer Relationship Management Systems (CRM), and Enterprise Resource Planning Systems (ERP) have provided companies the tools to manage these evolving supply chains [33]. The rapid development of these technologies have inspired a number of national advanced manufacturing strategies, including: "Industry 4.0" in Germany, "Made in China 2025" in China, and "Intelligent Manufacturing" in America [13]. The objective of these synonymous initiatives is to achieve autonomous, self-optimizing, and self-diagnostic capabilities to alleviate problems in complex manufacturing systems [196].

At this same time, enabling technologies have given customers the ability to *personalize* products through user-friendly interactive design interfaces and producers the ability to transfer those online customer designs into their production system [120]. The turn of the 21st century has seen these customer-driven environments become commonplace, leading to what is referred to as the era of Mass Customization (MC) in the manufacturing sector. The combination of these trends have led to an increasing number of manufacturers converting from a Make-to-Stock (MTS) production system to a Make-to-Order (MTO) operation scheme (also Assemble-to-Order and Engineer-to-Order) [114, 115, 186]. For simplicity of exposition and to capture the setting of our industrial partners, we will talk about the MTO manufacturing sector in this proposal, but all of our models and approaches can be applied to the service sector where worker capacity rather than equipment capacity will typically be the constraining resource.

The concept of Digital Twin for manufacturing systems (DTMS) provides an integrated solution for companies to overcome these challenges and achieve next-gen manufacturing strategies. Acting

as a centralized platform, a DTMS creates *living* digital models of their physical counterparts by ingesting real-time data from existing information systems [124]. These digital models can be applied to develop decision support systems, using real-time data, to provide users transparent, integrated and holistic solutions for the system they represent [47, 94].

However, the collection, storage and design of the data structures required to implement a DTMS represents a significant challenge in the field. In practice, manufacturers typically store various datasets in unique, disparate data environments and databases, with little to no capabilities of accessing these data points in a single interface. Further, researchers whom aim to develop applications for the optimization of processes and decision making for these manufacturers have even less access to relevant datasets. For the case of researchers, this challenge has led to the requirement of relying on abstracted data points, which are randomly generated, in order to develop optimization models. The solutions from these abstracted models are rarely able to be applied directly as a solution in the real-world context which they represent. For this reason, a major gap between the literature and practice exists, as manufacturers are not able to apply the models found in literature due to the abstractions which they rely on.

Our research objective is to address these trends and challenges faced by MTOs through the development of a novel end-to-end Digital Twin framework for Make-to-Order production facilities. This framework should be data-driven, and capable of enabling MTO firms the ability to make optimized decisions for a wide-array of use cases. We focus on problems related to production planning and scheduling. Throughout the development of the proposed Digital Twin framework and integrated solution procedures in this dissertation, we evaluate various representations of scheduling problems to identify the most promising implementations of these frameworks.

Specific to manufacturing and service operations, *machine scheduling* is defined as the allocation of tasks to available resources over some time horizon to best satisfy some set of criteria, subject to a series of constraints that capture the complexities of the environment at hand. The challenge in machine scheduling arises because each resource is limited in the number of jobs which it can process at any time and each job is limited in the number of resources it can be processed by at any time. To ensure the success of the firm, schedulers must allocate limited resources to the completion of tasks, typically with the goal of optimizing one or more objectives, such as minimizing the cost of penalties associated with tasks completed before or after their acceptable delivery windows.

We emphasize the importance of scheduling as the underlying problem which many other revenue management decision making problems rely on.

1.1 Production Planning and Execution Algorithms for Make-To-Order Firms

In Part 1 of this dissertation, we consider a class of Production Planning and Execution problems faced by job shop manufacturing systems. We are motivated by a collaboration with our industrial partner, a consultant firm supporting manufacturers in the aerospace supply chain. Manufacturers in this sector must generate production plans that span months in order to account for the long lead times experienced in the sector.

The supply chain of the aerospace industry is defined by corporations and companies which can be classified in one of four *tiers*, namely Original Equipment Manufacturers (OEMs), Tier I suppliers, Tier II suppliers, and Tier III suppliers. OEMs, such as Boeing and Airbus, carry out the design and assembly of the final products delivered to the end customers, i.e. commercial/combat aircrafts. Tier I suppliers, such as Pratt & Whitney, GE, United Technologies, etc., are the direct suppliers to OEMs and are responsible for manufacturing major sections of the aircraft, e.g. engines, wings, landing gear. Tier II suppliers supply Tier I firms and are responsible for producing the key sub-assemblies and sub-systems which compose the Tier I products. Tier III suppliers are the manufacturers of electronic components and raw materials in the aerospace supply chain.

In our study, we focus on the problems typically faced by Tier II suppliers. These companies, are generally smaller and less technically equipped than Tier I companies. However, most are fairly sophisticated in their capabilities and operations. These firms are subject to a High-Mix, Low-Volume (HMLV) demand backlog, requiring flexible manufacturing capabilities and agile decision making to ensure the delivery of finished goods that meet the terms agreed to with their customers. The products which Tier II manufacturers produce are complex assemblies composed of high-tech materials which require extensive lead times. As a result, the typical lead times associated with Tier II orders spans weeks and sometimes months, requiring a production plan which spans a correspondingly extensive period.

In practice, many Tier II MTO suppliers implement out-dated planning systems to schedule their operations. Advanced Planning Systems have been developed, but are expensive and require extensive resources and efforts to integrate into existing systems. Planning systems, such as Ma-

terials Resource Planning (MRP), prepare production plans in two steps: first, determining the materials requirements using customer orders and level-by-level Bill of Material (BOM) explosions, then calculating the amount of capacity required. Capacity is taken into account too late in the planning process, potentially resulting in infeasible and unexecutable schedules. In cases which feasible schedules were generated, optimization is not considered. MRP has since been replaced by MRPII which incorporates the consideration of capacity prior to materials requirements, but does so sequentially, leading to long process lead times [111].

To overcome these issues, the integration of materials and resource planning and scheduling is required. Our research objective is to develop a data-driven, generalized framework for the Multi-Level Capacitated Lot Sizing Problem (MLCLSP) that is capable of being *plugged into* existing Enterprise Resource Planning systems. This framework would act as a substitute to the expensive and difficult-to-integrate ERP-provided solutions which many firms are unable to afford/justify. Major challenges in this problem include: the extensive planning horizon which must be considered in the planning cycle, the different timescales in which different processes operate, the requirement that scheduling solutions must be at a detailed level which can be executed on the shop floor, and the practical considerations which need to be incorporated in order to develop a tool which would be usable by these Tier II manufacturers in practice.

1.2 Scheduling for Make-To-Order Firms Specializing in Mass Customization

In Part 2 of this dissertation, we are motivated by our industrial collaborator, Artaic – Innovative Mosaic. Artaic is a custom mosaic design studio and manufacturer in Boston, MA. They use robotic fabrication, which allows for fast, flexible and accurate assembly of unique tile work for their customers.

Artaic is an example of an MTO firm subject to the Mass Customization paradigm. The demands the firm faces is customer-driven, where each product design is established through many interactions with the customer. The challenges we consider stem from the demand generation process required for many MTO firms which offer this type of mass customization. Firms are often unaware of future demands until customers initiate communications with them through a product inquiry, which can lead to volatile and difficult-to-predict demand [50, 100]. Firms must submit quotations to customers, based on customer desired product specifications as well as a quoted price

and lead time. The timespan between when a firm submits a quotation and the customer informs their award decision may take weeks. During this time, the prospective job is a contingent demand, which may or may not be realized, but whose characteristics are fully known and whose terms (lead time, price) the firm is liable for. In order to maximize profits, the firm must be able to account for this uncertainty when submitting new proposals by balancing the increase in incoming revenue associated with shorter lead time offers with the potential delay penalties.

In recent decades, researchers and practitioners have implemented Revenue Management (RM) approaches to support these decisions. Revenue Management has been defined as the science that focuses on the implementation of sophisticated information technology systems, leveraging historical data and current congestion information, to enable firms, especially those facing fixed and perishable capacity constraints, the ability to identify optimal policies and/or make decisions with the objective of maximizing profits over some time horizon [38, 42]. However, there exists a gap in the literature when it comes to the science and understanding required by MTOs facing the uncertainties which derive from contingent demands. We plan to address this gap in our study through the development of an integrated framework that considers the underlying scheduling problem faced by Artaic. This framework will be capable of the incorporation of common RM decisions, such as the Order Acceptance (OA), Dynamic Pricing (DP), Due Date Setting (DDS) problems, and the Simultaneous Pricing, Due Date Setting and Scheduling Problem (SPDSP).

1.3 Dissertation Overview

This chapter introduces the reader to the industrial trends leading to the wide-spread adoption of Make-To-Order manufacturing paradigm and provides motivation for our research with the objective of developing a Digital Twin framework for manufacturing systems of small-medium sized firms.

In Chapter 2, we introduce the reader to the Digital Twin concept and develop the framework which is leveraged as the test-bed for the application and evaluation of all the planning and scheduling formulations considered throughout this dissertation. In Chapter 3, we address the Multi-Level Capacitated Lot-Sizing Problem (MLCLSP) from the perspective of a Tier-II manufacturer in the aerospace supply chain. We present an integrated solution procedure that: generates a production plan for an extended planning horizon, associates production decisions with specific

customer sales orders, and creates an executable schedule which can be followed on the shop floor. The formulations we develop leverage novel modeling techniques, and include important practical considerations, which to our knowledge, have not been addressed in the literature of MLCLSP.

In Chapter 4, we compare several direct-positional and relative-positional exact-method representations of the scheduling problems which MTO manufacturers, such as Artaic, are subject to. We provide a comprehensive computational analysis to evaluate the quality and computational efficiency of the presented formulations under various problem settings. In Chapter 5, we extend the scheduling problem introduced in Chapter 4 to include the considerations of uncertainties derived from contingent demands and provide a computational analysis of the proposed models. Chapter 6 concludes this dissertation and discusses opportunities for future research.

CHAPTER 2

A DIGITAL TWIN FRAMEWORK FOR MAKE-TO-ORDER PRODUCTION FACILITIES

2.1 Introduction

Today's supply chain and production facilities are becoming increasingly complex and manufacturers need to adapt to ever-increasing customer expectations, rising resource costs and an unprecedented level of uncertainty [94]. Recent advances in enabling technologies such as cloud computing and the Internet of Things as well as the maturation of information systems such as Transport Management Systems (TMS), Customer Relationship Management Systems (CRM), and Enterprise Resource Planning Systems (ERP) have provided companies the tools to manage these evolving supply chains [33]. The rapid development of these technologies has inspired a number of national advanced manufacturing strategies, including: "Industry 4.0" in Germany, "Made in China 2025" in China, and "Intelligent Manufacturing" in America [13]. The objective of these synonymous initiatives is to achieve autonomous, self-optimizing, and self-diagnostic capabilities to alleviate problems in complex manufacturing systems [196].

The concept of Digital Twin for manufacturing systems (DTMS) provides an integrated solution for companies to achieve these next-gen strategies. Acting as a centralized platform, a DTMS creates *living* digital models of their physical counterparts by ingesting real-time data from existing information systems [124]. These digital models can be applied to develop optimization decision-making models, using real-time data, to provide transparent, integrated and holistic views of the system they represent [47, 94]. Furthermore, these models can be applied off-line without disturbing the actual system to evaluate the impacts of unlikely scenarios, such as the disruptive events experienced during the SARS-CoV-2 pandemic in 2020 [32].

In a practical DT, the status of physical systems and their digital counterparts must be frequently synchronized, requiring data transparency between various information systems [184]. However, given the current state of data architectures implemented in industry today, cross-functional

coordination and data-sharing across these systems is difficult, and in many cases, only partially implementable [33]. This shortcoming brings about a significant challenge concerning the availability and accessibility of the data required to develop the core functionalities and capabilities which the DT concept will enable. For example, in their state-of-the-art review of DTMS technologies, Tao et. al. [171] cite that the literature on production planning and scheduling based on digital twins is rare, primarily due to the challenges in acquiring the data, real or artificially generated, necessary to develop the optimization models that would be applied to the manufacturing systems.

This challenge leads to a significant gap between optimization applications related to production planning and scheduling in literature and the methods used in practice. Due to the lack of availability of realistic data researchers must apply abstractions and aggregations to data points in order to formulate and study applications. However, once these abstractions are applied to enable the development of optimization models, it is rare for researchers to consider the performance of resulting models when these abstractions, assumptions and aggregations are removed.

In this dissertation, we aim to address this gap by developing a Digital Twin framework which extends the typical problem definition of operation research applications to consider the pre-processing and post-processing steps required to consider the realistic data points which would be encountered in practice. The optimization applications developed within this framework translate relevant data points such that tractable models can be solved in reasonable computational time. Additionally, in this framework, the solutions from these models, i.e. production plans, are stripped of any abstractions and assumptions such that a user of the framework would be able to evaluate the performance of the optimization models under the context of the real-world system in which these solutions would be performed.

The objectives of this chapter is four-fold: 1) introduce the Digital Twin concept for manufacturing systems and develop a modular framework which will act as the test-bed to enable the application of the production planning and scheduling optimization models, 2) develop algorithms capable of generating problem instances of hyper-generalized production facilities, leveraging data models representative of those used in modern-day ERP-based information systems, 3) develop a novel scheduling and simulation algorithm capable of translating discrete-time production plans into continuous-time, executable schedules, and 4) provide an in-depth illustrative example to validate and showcase the capabilities and comprehensiveness of the proposed DT framework.

The rest of this chapter is organized as follows. A brief review of the history and application of Digital Twins in practice and academia is presented in Section 2.2. In Section 2.3, the proposed Digital Twin framework developed for this dissertation is introduced. In Sections 2.4-2.7, the various modules which make up the proposed Digital Twin framework are discussed. In Section 2.8, an illustrative example of the Digital Twin framework is presented, and Section 2.9 concludes the chapter.

2.2 Literature Review

First introduced by Professor Grieves [65] in 2003, the DT concept was defined as: "a set of virtual information constructs that fully describes a potential or actual physical manufactured product from the micro atomic level to the macro geometrical level. At its optimum, any information that could be obtained from inspecting a physical manufactured product can be obtained from its Digital Twin". The concept was developed as a means to enable "Intelligent Manufacturing", through the improvement of the Product Life Cycle Management (PLM) software. At the time, data collection of physical products depended on manual user-input and the amount of available data was lacking, so the mention of "Digital Twin" was rare in research between 2003 and 2010 [13, 99].

However, following advancements in enabling technologies, the DT concept has since been developed, defined and applied under various contexts and settings. NASA, the first to publish the term "Digital Twin", defined DT as: "an ultra-realistic, high scaling simulation, which uses the best available physical models, sensor data and historical data for mirroring one or more real systems" [176]. Busse et. al. [33] define the DT of a supply chain as: "a digital simulation model of a real logistics system, which features a long-term, bidirectional and timely data-link to that system." Kunath and Winkler [94] define the DT of a manufacturing system as: "a data-oriented representation of all elements of the manufacturing equipment system, the material flow system, the value stream system, the operating materials system and the human resource system in the information world, which are linked to their physical elements by the information system."

In recent years, three areas of literature have been widely discussed for the application of Digital Twins: smart cities, healthcare, and smart manufacturing/supply chain [99]. For the purposes of this research we consider and adopt the definition of the DT concept as applied to manufacturing

systems. It should be noted that no single approach has been agreed on for the best practices in the development of a Digital Twin, and as a result, the proposed capabilities and frameworks for DTs developed in the literature varies from use-case to use-case. However, most experts agree that a combination of simulation modeling, optimization and data analytics make up the full range of technologies required to create a DT [15].

Busse et. al. [33] develop a DT of a multi-modal supply chain for planning and control applications providing the following capabilities: supply chain visibility, data analytics (including predictions of future states of the system), and extensive decision support through the application of process optimization for planning and disruption handling. Shi et. al. [155] consider a DTMS of a batch production, mixed line assembly of aerospace products. In their framework, the authors describe a system architecture composed of three platforms: a hardware layer, a software layer, and an application level. The hardware layer is composed of the equipment and communication technologies responsible for collecting real-time data. The software layer includes the databases which store the data and the programs which make up the *module* functionalities. The application layer acts as the DT's user-interface and includes the visualizations representing the virtual representation of the manufacturing system. Wang et. al. [185] consider a DT model of a planning and scheduling system composed of five parts: the physical workshop, the virtual workshop (a digital mirroring of the physical workshop), the digital twin platform, the data prediction platform and the production planning system. All of the proposed frameworks rely on the bi-directional real-time sharing of data among the modules of the DT. Gerlach and Zarnitz [60] study the use of DTs for Logistics and Supply Chain Management and identify four levels of use: macro-level DTs considering multiple stakeholders in an inter-connected supply chain, DTs considering multiple stakeholders in an intra-connected supply chain, site level DTs (i.e. a warehouse, production facility, etc.), and asset-level DTs (i.e. trucks, forklifts, machines, etc.).

For the purposes of this dissertation, we consider a site-level DT framework. Baruffaldi et. al. [14] consider a site-level DT of a warehouse facility and exploit optimization and simulation techniques to quantify the impacts of information availability in decision-making processes. Schluse and Rossmann [153] consider a site-level DT of a production facility and develop a virtual test-bed for the application of various experimental simulations and databases. The authors consider their Digital Twin as the sum of all different simulation models and systems. Zhang et al. [199] also

consider a production facility DT and apply a proactive job-shop scheduling strategy in small (6 jobs, 6 machines) and large (20 jobs, 40 machines) environments to validate the effectiveness and scalability of the proposed framework. Zhou et. al. [201] also develop a test-bed for the validation of a Digital Twin of manufacturing cells. The authors provide applications of the DT in problems for intelligent process planning, production scheduling, process analysis and dynamic regulation of the production schedule to demonstrate the feasibility of their DT.

A notable trend in the literature of Digital Twins has been a shift in the focus of the application of DTs from being "tech-oriented" into being "decision-oriented" [184]. However, due the immaturity of the field (with most papers being written since 2018) it is difficult to describe the evolution of the application of DT solutions in managing supply chains [15]. A common theme found throughout the proposed DTs considers the *modularity* of the sub-systems which make up each framework. Guo et al. [67] consider a site-level DT of a factory. The authors highlight the benefits of implementing a modular construct to the DT architecture. The idea of a modular approach is to build reusable and *parameterized* software packages to reduce modeling times. Each module operates independently, allowing for drag-and-drop modeling, which also reduces the time required for modeling systems. This is achieved through parameterization, in which the strategic definition of each module allows a user to represent unique physical assets by inputting different user-defined parameters. The encapsulation of these modules into self-contained sub-systems also ensures that they are reusable, easy to modify, and seamlessly included or excluded in the DT framework as needed.

The development and application of a DT framework depends on the availability of datasets which represent the physical assets which are being modelled. In the absence of real-world data, which is common for researchers who develop models for production planning and scheduling problems, one of the most valuable sources of test data is that from a random generator [126]. However, the availability of public datasets and generators are limited in literature. For this reason, the applications found in literature are restricted to the problem contexts described by these available datasets or are reliant on the random generation algorithms developed for each specific use case. A major gap in the literature exists due to these challenges, as the developed algorithms for generating application specific datasets is often overlooked and not considered

While "standard reference" datasets are publicly available, see OR-Library [20], and widely used for certain problems in the literature of production planning and scheduling, i.e. the test instances developed by Tempelmeier and Derstroff [172] and Stadtler and Surie [165] for the Multi-Level Capacitated Lot-Sizing Problem (MLCLSP) and Brandimarte [29] for the Flexible Job Shop Scheduling Problem (FJSSP), datasets and data generators for more general use-cases, like the ones we consider in this dissertation, are not. For this reason, we must develop algorithms to generate the datasets which represent the problems our industrial partners face.

Considering the need for developing generators, Hall and Posner [71] discuss several principles for data generation and some desirable properties of generation schemes when related to machine scheduling applications. These principles include: 1) to generate data to satisfy the purposes of the experiment/use-case, 2) to make computational tests comparable, 3) to avoid biases, and 4) to make the generation scheme reproducible. Desirable properties of data generators include being: able to generating outputs which create a wide range of problem instances, practically relevance, describable, easy of use, and reproducible, i.e. through the implementation of pseudo-random generators.

Our contributions to the literature are as follows. We develop a novel, site-level, Digital Twin framework for manufacturing systems which will act as the test-bed for the optimization models developed throughout this dissertation. The Digital Twin framework is developed as a modular software package, leveraging methodologies inspired by Object-Oriented Programming such that all of the implementations can be extended, modified and/or replaced with minimal modeling efforts. This framework is also designed with practicality and configurability in mind, with efforts made to design the framework to be able to ingest datasets which represent the ones used in traditional Enterprise Resource Planning (ERP) systems. We also develop a novel algorithm for generating datasets representative of the Multi-Level Capacitated Lot Sizing Problem (MLCLSP). In this, we emphasize the importance of defining simple and measurable user-defined parameters that, when passed as input to the generator, allow for the generation of a wide-range of instances for a variety of problems, not just the MLCLSP. Further, we develop algorithms for translating production plan solutions (for those generated test instances) from a discretized/aggregated solution into an executable schedule which can then be evaluated in the contexts which it would be in the real-world. This is a consideration which is rarely addressed in the literature of production planning.

2.3 General Concepts

Our proposed Digital Twin framework can be thought of as a collection of programming *modules*. One benefit of a modular design is that it allows the user to *plug-in* additional modules that enhance the capabilities of the DT without modifying any other existing modules. Throughout this dissertation we leverage these *modularity* characteristics to apply multiple production planning optimization and revenue management decision-making models to various manufacturing environment settings. The proposed DT framework is composed of 5 modules: an Object Definition Module (ODM), a Scenario Generation Module (SGM), a Production Planning Module (PPM), a Scheduling Simulation Module (SSM), and an Application Programming Interface Module (API). A process map showing the relationship of each module is shown in Figure 2.1.

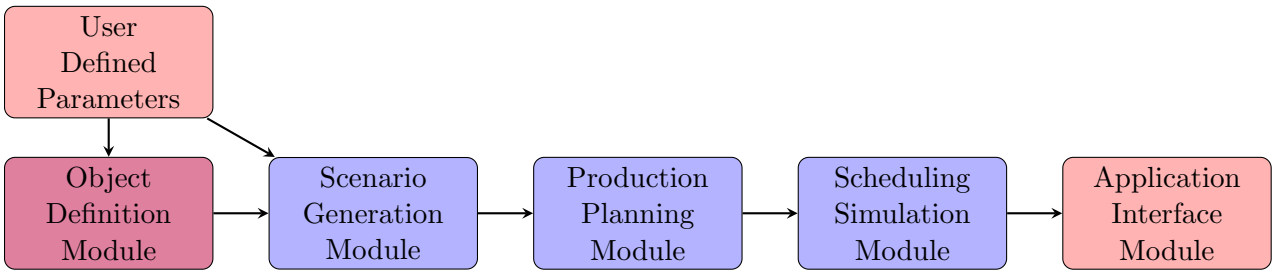


Figure 2.1. Proposed Digital Twin framework: modules and data flow

The Object Definition Module contains the constructs of the entities which are represented within the DT. Throughout this chapter, we use a `typescript` font to refer to instances of the following objects *models*: `Factory`, `PartClass`, `ResourceGroupClass`, `Part`, `ResourceGroup`, `Resource`, `Process`, `LineItem`, `Node`, `ScheduledReceipt`, `Shipment`, `WorkOrder`, `Task`. Each object model can be classified as either: 1) a parametric object model, 2) a static object model, or 3) a dynamic object model.

Parametric object models (`Factory`, `PartClass`, `ResourceGroupClass`) define the structure of the User-Defined Parameter (UDP) inputs which are required to generate a scenario in the Scenario Generation Module. The `Factory` parameter model includes general arguments which summarize the production environment that will be considered, such as the number of unique parts and the number of resources in the facility. The `PartClass` parameter model allows a user to associate specific traits that are inherent to *part classes*. Each generated `Part` will belong to

exactly one `PartClass` and will be assigned attributes based on the user-defined parameters within that `PartClass`. For example, the attributes associated with a *Raw Material* are likely to be very different from an *Assembled Part*. Similarly, the `ResourceGroupClass` parameter model allows a user to distinguish different classes of `ResourceGroup` objects. These parametric models, and the arguments which are required for each, are discussed in further detail in Section 2.4.

The collection of all UDP, once binded to the parametric object models, are input to the Scenario Generation Module. Each generated scenario is represented as a collection of instances of static object models (`Part`, `ResourceGroup`, `Resource`, `Process`, `LineItem`, `Node`, `ScheduledReceipt`). A `Part` is a physical entity which can either be produced as the output of one or more processes, procured from an external source, or delivered to an external customer to satisfy a demand. A `Process` describes the characteristics associated with a specific step in the production routing required to produce a part, including the `ResourceGroup` required to complete it. Each `ResourceGroup` is defined as a group of similar `Resource` objects capable of performing similar tasks at similar processing rates. Each `ScheduledReceipt` represents the arrival of a quantity of a specific part at a specific time. This includes initial inventory levels at the beginning of the planning horizon and items which are procured from external sources. Each `LineItem` represents an external customer demand for a quantity of a part at a specific time. In order to be completed, a `LineItem` requires the completion of a collection of *production requirements*. Production requirements, are represented by `Node` objects and are characterized as an *internal demand* for a quantity of a specific component/part/sub-assembly required to complete the end item which is demanded in a specific `LineItem`.

The output of the SGM is input to the Production Planning Module, where a production plan is generated using optimization and/or heuristic techniques. A production plan is represented as a collection of instances of dynamically generated object models (`Shipment`, `WorkOrder`, `Task`). Each `Shipment` represents an instruction to deliver a quantity of an end item at a specific time to a customer, satisfying a demand described by a `LineItem`. Each `WorkOrder` represents an instruction to produce a quantity of a specific part at a specific time. Existing work orders which must be followed in the production plan can also be passed as user-input. Each `Task` represents an instruction to execute a process required to complete an associated work order, as defined in the routing of the part which is being completed in that work order. The output of the PPM is

input to the Scheduling Simulation Module, where the production plan is scheduled and simulated using a continuous-time planning horizon, considering additional complexities and nuances, such as subtle differences in resource capacities, which may not be feasible or practical to consider in the PPM. The output of the SSM can be used as input to the API Module for analytic functionalities that evaluate and visualize the performance of the production plan for the generated test instance.

Figure 2.2 depicts the Object Relational Mapping of each of the object models we have defined in the Object Definition Module.

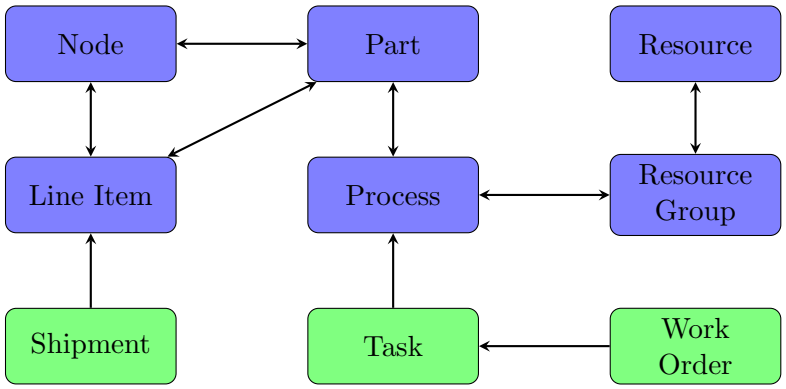


Figure 2.2. Proposed Digital Twin Framework: Object Relational Mapping diagram. Models are color-coded based on their classification: static (blue), dynamic (green). A directional-connection between models represents an inherent relationship between an instance of the source and destination object models, i.e. each `Resource` corresponds with a `ResourceGroup`, each `Task` with a `Process`, each `Part` with one or more `Process`, etc.

2.4 User Defined Parameters

User-defined parameters allow the user to specify the scope and characteristics of the scenario to be generated in the SGM. As mentioned, the user provides relevant parameters through the `Factory`, `PartClass`, and `ResourceGroupClass` parametric object models. Our objective in defining these models is to consider input fields for which data is typically available in practice or can be easily estimated. In the following, we discuss these parameter models in detail.

The `Factory` parameter model contains arguments which summarize the production environment to be generated. These arguments includes the specification of product structures, manufacturing process flows, the complexity of the routings associated with each item and the target

utilization rate of resource groups. These criteria and respective configurations are shown in Table 2.1.

Criteria	**Attribute**
Product structure	General Assembly
Process Flow	Non-cyclical Cyclical
Route complexity	Single-process Multi-process
Utilization Profile	90/90/90 70/70/70 50/50/50 90/70/50 50/70/90

Table 2.1. User-defined parameters to describe general production facility environment settings

The specification of a *product structure* restricts the parent-child relationships between parts in the Bill of Materials. A general product structure allows items to have multiple parent parts, while an assembly structure restricts the number of parents of any item to 1. See Figure 2.3 for an illustration of the differences associated with general vs assembly product structures. Note that these two representations of the example Bill of Materials show the component structures of the same top-level assemblies. A new naming convention for identifying each part must be adapted when translating the general product structure (left) into the assembly product structure (right). In the figure, the letters represent distinct *node* IDs in the assembly, while the corresponding part number is given in parenthesis.

Also note the annotation referencing the "BOM Level" of each *offset level* in the BOM of the networks. We define the BOM Level of each part in the general product structure BOM as the maximum *distance* that part can be from a top-level part (BOM Level = 0). This attribute is used in the Scenario Generation Module for dictating the parent-child relationships between parts and for determining which resources are allowed to process each part. This is discussed further in Section 2.5.

The *process flow* criteria defines constraints, or lack thereof, in the sequence in which parts are allowed to visit resource groups in the facility. In a non-cyclical process flow, no part can

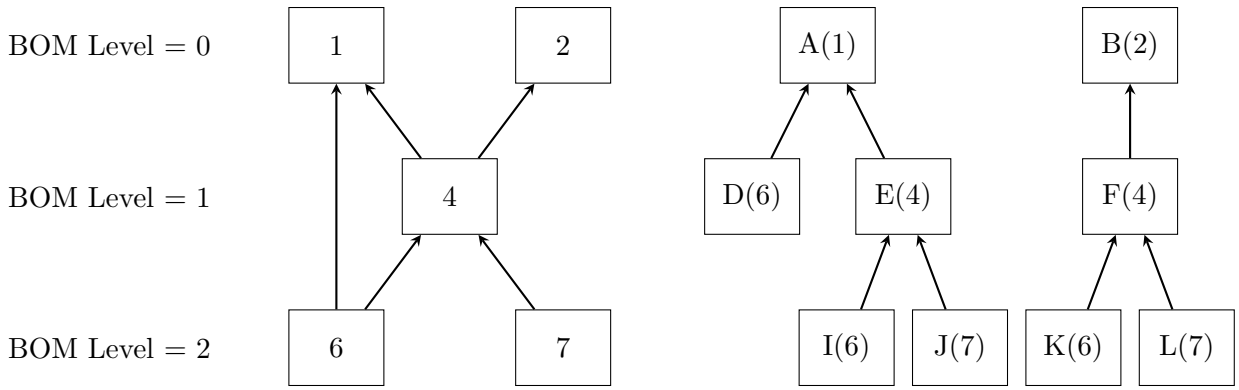


Figure 2.3. Example: General product structure (left) and Assembly product structure (right) representations of two end items. Numerical values represent part numbers. Alphabetic values represent node ID. Note that network nodes are duplicated for each part that is found in multiple assemblies, i.e. Part 6 is represented with Node D, Node I and Node K.

visit the same resource group more than once throughout its transformation from raw material to finished good. In a cyclical flow, no such restriction is enforced. *Route complexity* allows the user to represent items as the output of a series of processes or as the output of a single process. *Utilization profiles* define the approximate utilization rate to be realized by resources in the generated scenario. This allows the user the ability define bottlenecks in specific *phases* of the value chain, i.e. upstream/midstream/downstream processes. For example, a utilization profile of 50/90/70, would represent a factory with a bottleneck associated with the mid-stream processes as well as a higher expected utilization for downstream processes than upstream processes.

A summary of the remaining `Factory` parameters which dictate the scale and context of any test instance is given in Table 2.2.

Parameter	**Notation**	**Description**
`numI`	$\|I\|$	Number of unique items
`numJ`	$\|J\|$	Number of unique resource groups
`numLI`	$\|O\|$	Number of external line items
`numSR`	$\|A\|$	Number of scheduled receipts
`maxT`	$\overline{T}$	Length of the planning horizon (in weeks)
`maxRperRG`	$\overline{E}$	Max number of resources per resource group
`maxLIQty`	$\overline{Q}$	Max quantity demanded in any line item
`maxLev`	$\overline{\lambda}$	Max depth of any part's BOM, i.e. number of offset levels
`maxUPP`	$\overline{n}$	Max units-per-parent of any parent-child relationship

Table 2.2. User-defined parameters, notation, and definitions - `Factory` parameter model

Each `PartClass` is defined by a unique part class name and a series of user-defined inputs which set the bounds for the distributions of the attributes which each generated part of that class can inherit. For the purpose of this dissertation, we consider 4 unique part classes: raw materials, machined parts, assembled parts, and finished goods. It should be noted that the values associated with these parameters are also constrained by the user-specified `Factory` parameters. For example, the maximum BOM level of a part class cannot be greater than `maxLev`, as specified in the `Factory` parameter model. As another example, when the `Factory` parameter for `routeComp` is set to *Single-process*, the value for `maxRS` cannot be greater than 1. The arguments associated with each `PartClass` are summarized in Table 2.3.

Parameter	Notation	Description
`className`	c	Name of this part class
`classDensity`	d_c	Probability that any part belongs to this class
(`minChild`, `maxChild`]	$(\underline{K^{c+}}, \overline{K^{c-}}]$	Min/Max number of children for items in class c
(`minParent`, `maxParent`]	$(\underline{K^{c-}}, \overline{K^{c-}}]$	Min/Max number of parents for parts in class c
(`minRS`, `maxRS`]	$(\underline{s_c}, \overline{s_c}]$	Min/Max number of route-steps for parts in class c
(`minLevel`, `maxLevel`]	$(\underline{\lambda_c}, \overline{\lambda_c}]$	Min/Max BOM level for parts in class c

Table 2.3. User-defined parameters, notation, and definitions - `PartClass` parameter model

The `ResourceGroupClass` parametric model (see Table 2.4) sets the distributions of the attributes which each generated resource group, of that class, can inherit. Note that the relationship between an instance of a `ResourceGroupClass` parametric model and an instance of a `ResourceGroup` static model is that a `ResourceGroup` is a randomly generated instance that has its attributes generated based on the user-defined parameters described in a `ResourceGroupClass`. Further, each `ResourceGroup` is described by exactly one `ResourceGroupClass`, but multiple resource groups can belong to each `ResourceGroupClass`. Some examples of resource group classes include: a CNC-based class, a drilling class, an inspection class etc. Continuing with this example, a production facility could have multiple resource groups that carry out drilling operations, but at different stages of the value chain with unique processing characteristics. In this case, each randomly generated instance of a drilling resource group would have its attributes sampled from distributions that are bounded by the parameters described in the drilling `ResourceGroupClass`.

Each `ResourceGroupClass` parametric model includes the specification of `minLevel` and `maxLevel`, restricting which parts that resource group class can process (on the basis of each part's BOM level). Consider a downstream, packaging resource group; these resources would likely be associated with the processes of a finished good prior to shipping out to satisfy an external demand. The parameter, `timeline`, specifies the magnitude of the typical cycle time allowed to complete any production cycle (considering setup time and processing time) on a resource within the resource group of that class. This parameter allows the user to specify the typical cycle time of the operations which are processed by a class of resources (its use-case is discussed in detail in Section 3.3.1). `probSetup` is the probability that any process on a resource group of that class will require a setup activity. `minS` and `maxS` are the minimum/maximum proportion of time spent for a setup activity of any process on a resource in that class, relative to the value of the `timeline` input.

Parameter	Notation	Description
`className`	c	Name of resource group class
`classDensity`	d_c	Probability that any part belongs to this class
[`minLevel`, `maxLevel`]	$[\underline{\lambda_c}, \overline{\lambda_c}]$	Min/Max process-able BOM level for resource group in class
`timeline`	T^c	Timeline associated with this resource group
`probSetup`	$P(S_c > 0)$	Prob. that a process will require a setup
(`minS`, `maxS`]	$(\underline{S_c}, \overline{S_c}]$	Min/Max setup time of any process (proportion of `timeline`)

Table 2.4. User-defined parameters, notation, and definitions - `ResourceGroupClass` object model

2.5 Scenario Generation Module

In this section, we describe the algorithms and logic developed for the Scenario Generation Module (SGM) of the proposed Digital Twin framework. The objective of this module is to create realistic problem scenarios based on the user-defined inputs for the parameter models described in Section 2.4.

We first provide a high-level overview of the SGM developed for this dissertation. In the following subsections, we describe in detail the attributes of each static object model and the algorithms developed to generate them. For the purpose of exposition, we provide the overview so that the following subsections can act as supplemental content which is not critical to the understanding of this dissertation.

2.5.1 Overview

The SGM takes the user-defined parameters presented in Tables 2.2-2.4 as input. As discussed in Section 2.3, the primary output of the SGM is a collection of `Part`, `ResourceGroup`, `Resource`, `Process`, `LineItem`, `Node`, and `ScheduledReceipt` objects which accurately represent the processes and current situation of a realistic factory setting.

The procedure of generating a problem instance begins with the initialization of all `Part` objects. Each part is assigned to belong to exactly one part class and inherits attributes based on the user-defined parameters of that class. Following this, parent-child relationships are randomly created between parts. This results in a comprehensive Bill of Materials (BOM), for each of items the production facility is capable of producing. This BOM is represented in the *general* product structure (see Figure 2.3).

External demands are then generated, in the form of `LineItem` objects. The external demands, also referred to as sales orders, are analyzed to derive all internal part demands required to satisfy them. At this point, all `Part` objects are updated with relevant attributes, given their overall production requirements. Finally, an *assembly* product structure representation of the Bill of Materials is created, through the generation of `Node` objects. The representation of the BOM in an assembly product structure is relevant for the Order-Lot Matching Problem (OLMP), presented in Chapter 3, in which production lots are associated with the sales order they satisfy.

The next phase in the SGM concerns the generation of factory resources and the Bill of Operations (BOO). First, all `ResourceGroup` objects are initialized. Each resource group is assigned to belong to exactly on resource group class and inherits attributes based on the user-defined parameters of that class. `Resource` objects are then generated, each belonging to a specific resource group.

Following the creation of all resource groups and resources, a production routing is generated for each part, defining the sequence of resource groups it must visit. A `Process` object is initialized for each step in the routing of each part. Once all `Process` objects have been initialized, they are assigned relevant attributes considering the internal demands of the parts and resource groups associated with each process. A description of the notation used for the sets, subsets, and parameters associated with the outputs of the SGM can be found in Table 2.5.

In the following, we discuss this procedure in detail and provide the algorithms and equations implemented to generate factory scenarios that are consistent with practice and possess the desired characteristics, as specified in the user-defined parameters. Note, for instance, that ensuring the factory with complex BOMs and BOOs is loaded with a demand backlog (line items) that will lead to the desired resource utilization is very challenging.

2.5.2 Generating Parts and the Bill of Materials

The first stage in the Scenario Generation Model initializes all `Part` objects. We have a user-defined number of part objects, $|I|$, which must be generated. In an iterative process, parts are initialized, until $|I|$ part objects have been created. Each part is generated with a unique identifier, i, and will be randomly assigned to belong to one of the user-defined part classes, c, with probability d_c. Several attributes are then generated for the part on the basis of the part class which it belongs to. These attributes include: the number of children, $|K^{i+}|$, the number of parents $|K^{i-}|$, the BOM level, λ_i, and the number of processes required to produce the part, s_i.

Once all parts have been initialized, the Bill of Materials is created, represented as a collection of parent-child relationships between parts. The objective during the generation of the BOM is to create parent-child relationships for each part such that the resulting "network" adheres closely to the generated attributes of each part, specifically in the number of parents and children each part has.

This BOM network, G, is represented as a directed acyclic graph, where each part is represented as a node in the network and each parent-child relationship as a directed edge from the child node to parent node. The BOM network is generated in two steps. First, the network is initialized, then it is corrected through the implementation of several *patches*.

During the initialization step, in an iterative process, each part, i, is randomly assigned $|K^{i+}|$ children parts. To ensure an acyclic BOM structure, a part, k, is only allowed to be a child to part i if the BOM level of part k is greater than part i: $\lambda_k > \lambda_i$, i.e. a raw material, i, with BOM level $\lambda_i = 4$ can not be a parent of an assembly part, k, with BOM level $\lambda_k = 1$. For each selected child, a Units-Per-Parent (UPP) attribute, n_{ik}, is generated to describe the quantity of child part required to produce one unit of parent part.

Sets	Description
$i \in I$	Set of all `Part` objects
$j \in J$	Set of `ResourceGroup` objects
$o \in O$	Set of all `LineItem` objects
$\gamma \in \Gamma$	Set of all `Process` objects
$e \in E$	Set of all `Resource` objects
$f \in F$	Set of all `Node` objects
Subsets	**Description**
$k \in K$	Set of `Part` with children
$k \in K^{i+}$	Set of children parts for `Part` i
$k \in K^{i-}$	Set of parent parts for `Part` i
$i \in I^j$	Set of `Part` processed by `ResourceGroup` j
$j \in J^i$	Set of `ResourceGroup` visited in the route of `Part` i
$o \in O^i$	Set of `LineItem` that demand `Part` i
$\gamma \in \Gamma^j$	Set of `Process` processed by `ResourceGroup` j
$\gamma \in \Gamma^i$	Set of `Process` in routing of `Part` i
$e \in E^j$	Set of `Resource` that belong to `ResourceGroup` j
$f \in F^i$	Set of `Node` that represent `Part` i
$f \in F^o$	Set of `Node` required to be fulfilled to fulfill `LineItem` o
Mapping	**Description**
$i \leftarrow I^o$	`Part` demanded in `LineItem` o
$i \leftarrow I^f$	`Part` that is represented by `Node` f
$i \leftarrow I^\gamma$	`Part` that is processed in `Process` γ
$j \leftarrow J^\gamma$	`ResourceGroup` responsible for `Process` γ
$o \leftarrow O^f$	`LineItem` which `Node` f is a production requirement for
Parameters	**Description**
h_i	Per-unit per-period holding cost of `Part` i
H_i	Per-unit cost of under-production of `Part` i (relative to N_i)
L_i	Lead-time of `Part` i
N_i	Total production requirement of `Part` i
V_i	Approximated value of `Part` i
n_{ik}	Units-per-parent of `Part` i in `Part` k
Q_o	Quantity demanded in `LineItem` o
D_o	Due date of `LineItem` o
$\underline{P_o}$	Per-unit per-period penalty for tardiness of `LineItem` o
$\overline{P_o}$	Per-unit penalty for under-fulfillment `LineItem` o
ϵ_o	Per-unit per-period reward fro earliness of `LineItem` o
T^j	Maximum batch cycle time for any `Process` on `ResourceGroup` j
R_γ	Per-unit process rate of `Process` γ
S_γ	Per-batch setup time of `Process` γ
M_γ	Maximum quantity batch size of `Process` γ
s_γ	Step in the routing of `Process` γ
l_γ	Lead time offset between start of production of part until it reaches `Process` γ (i is trivial in γ)

Table 2.5. Sets, subsets, index-mappings and parameters generated as outputs of the Scenario Generation Module

Note that during the BOM initialization, there is no consideration for the objective of creating a BOM with the correct number of parents for each part, $|K^{i-}|$. For example, there is no control to

ensure that each part has a parent, potentially resulting in isolated sub-networks within the comprehensive BOM, G. Further, no restriction is enforced that limits the number of parents any part can have, which may result in a BOM network that is over-reliant on a single part or assembly. These concerns are addressed by iterating through all parts in the BOM network and adding/removing parent-child relationships accordingly. An overview of the `Part` object initialization and BOM generation process is shown in Algorithm 1.

Algorithm 1: Procedure for generating parts and the Bill of Materials

Input: $|I|$ - The number of `Part` objects to be generated
Input: `UDP` - User-Defined Parameters

```
# Initialize all Part objects
```
Initialize set of all parts, $I = \emptyset$
for $i \in |I|$ **do**
  Initialize a part object, $i \leftarrow$ `Part`
  Randomly assign part i to a part class, c, with probability d_c
  Set target number of children parts, $|K^{i+}| \leftarrow \mathtt{U}\big[\underline{K^{c+}}, \overline{K^{c+}}\big]$
  Set target number of parent parts, $|K^{i-}| \leftarrow \mathtt{U}\big[\underline{K^{c-}}, \overline{K^{c-}}\big]$
  Set BOM level of part, $\lambda_i \leftarrow \mathtt{U}\big[\underline{\lambda_c}, \overline{\lambda_c}\big]$
  Set number of route steps required for part, $s_i \leftarrow \mathtt{U}\big[\underline{s_c}, \overline{s_c}\big]$
  Add new part to set of all parts, $I \uplus i$

```
# Initialize all parent-child relationships
```
Initialize parameter $n_{ik} = 0$ for all $i \in I, k \in I : i \neq k$
for $i \in I$ **do**
  Let K be the set of eligible children parts for i, $K \subset I : \lambda_i < \lambda_k$
  Let K^{i+} be a random selection of $|K^{i+}|$ eligible children parts from K
  for $k \in K^{i+}$ **do**
    Set UPP relationship, $n_{ik} \leftarrow \mathtt{U}\big[1, \overline{n}\big]$
Let G be a directed acyclic graph describing all parent-child relationships

```
# Implement BOM patches
```
for $i \in I$ **do**
  if *this part doesn't have enough parents,* $deg^-(i) < |K^{i-}|$ **then**
    `# priority to eligible parts with least children, asc`$\big(deg^+(i)\big)$
    Add $|K^{i-}| - deg^-(i)$ parents
  if *this part has too many parents,* $deg^-(i) > |K^{i-}|$ **then**
    `# priority to this parts parents with most children, desc`$\big(deg^+(i)\big)$
    Remove $deg^-(i) - |K^{i-}|$ parents

Output: I - Set of all parts
Output: K - Set of all parts that have children, $K \subset I$
Output: K^{i+} - Set of all children parts for each part i
Output: K^{i-} - Set of all parent parts for each part i
Output: λ_i - BOM level of each part i
Output: s_i - Number of route-steps to produce each part i
Output: n_{ik} - Units-Per-Parents of part i required to produce part k

We assume that the relative value of each part is based on the costs of materials, labor and processing required to produce it. We propose the following equations for estimating the value of

each part. The resulting parameters, V_i, are later used as the basis for the calculation of additional part attributes, i.e. per-period holding cost, sales price, etc. The approximated *value* of any part, k is found as:

$$V_k = \sum_{i \in G^k} \sum_{p \in P(G_i^k)} \prod_{(a,b) \in p} n_{ab}$$

where:

G^k	is a subgraph of G (with root node, k)
$i \in G^k$	is the set of all nodes in the subgraph G^k
$p \in P(G_i^k)$	is the set of all *simple* paths from a source node, i, to the root node, k
$(a, b) \in p$	is the set of all child-parent relationships found in path p
n_{ab}	is the Units-Per-Parent of part a required to produce part b

Note that the Units-per-Assembly (UPA) relating any upstream component, i, and downstream assembly, k, can be found using a similar equation:

$$n_{ik} = \sum_{p \in P(G_i^k)} \prod_{a,b \in p} n_{ab}$$

This parameter, n_{ik}, can be used to derive the total production requirements of each part, i, throughout the duration of the planning horizon to satisfy all external demands.

2.5.3 Generating External and Internal Demands

Following the creation of the Bill of Materials, external demands are generated, in the form of `LineItem` objects. We have a user-defined number of customer demands, $|O|$, which need to be generated. Each line item, o, specifies an external demand for a quantity, Q_o, of specific part, $i \leftarrow I^o$, with due date, D_o. In an iterative process, these $|O|$ line items are created with values generated for the quantity, $Q_o = \mathtt{U}[1, \overline{Q}]$, of the end item demanded and a requested due date, $D_o = \mathtt{U}[1, \overline{T}]$, where $\overline{Q}$ and $\overline{T}$ are user-defined parameters from the `Factory` parameter model. The internal demand, N_i, for each item, $i \in I$, throughout the duration of the planning horizon can then be calculated as:

$$N_i = \sum_{o \in O} Q_o \, n_{i,I^o}$$

where O is the set of all external demands and I^o is the part requested in order o.

Algorithm 2: Procedure for generating external demands and populating part attributes

Input: G : Bill of Materials network, composition of all finished good BOMs
Input: $|O|$: The number of line items to generate
Input: n_{ik} : Units-Per-Parent relationship of child part i and parent part k
Input: h,H,P,$\overline{P}$, ϵ : User-defined parameters for determining objective function coefficients

`# Calculate approximate value of each part`
for *each part, i, in the BOM network, $i \in G$* **do**
  Let G^i be a subgraph of G, with root node, i, and all ancestors of k, $\overline{K^{i-}}$
  Let $V_i = 0$
  for *each part, k, in the BOM of i, $k \in G^i$* **do**
    Let $n_{ik} \leftarrow \sum_{p \in P(G^i_k)} \prod_{a,b \in p} n_{ab}$
    $V_i \leftarrow V_i + n_{ik}$

  `# Populate attributes of` i `using` V_i
  Let $h_i \leftarrow V_i\ h$ be the per-unit per-period holding cost of i
  Let $H_i \leftarrow V_i\ H$ be the per-unit cost for underproduction of i

`# Generate LineItem objects`
for $o \in |O|$ **do**
  Initialize a `LineItem` object, o
  Let $I^o \leftarrow i$ be a randomly selected `Part` of class *finishedGood*
  Let Q_o be the quantity demanded, $Q_o \leftarrow \mathtt{U}\big[1, \overline{Q}\big]$
  Let D_o be the due date, $D_o \leftarrow \mathtt{U}\big[0, \overline{T}\big]$
  Let P_o be the per-unit per-period penalty for tardiness, $P_o \leftarrow V_i\ P : i \leftarrow I^o$
  Let ϵ_o be the per-unit per-period reward for earliness, $\epsilon_o \leftarrow V_i\ \epsilon : i \leftarrow I^o$
  Let $\overline{P}_o$ be the per-unit penalty for under-delivery, $\overline{P}_o \leftarrow V_i\ \overline{P} : i \leftarrow I^o$

`# Calculate total demand profile of each part during planning horizon`
for $i \in I$ **do**
  Let $N_i = \sum_{o \in O} Q_o\ n_{i\ I^o}$ be total demand of part i across all orders

Output: H_i : Per-unit penalty for under production (relative to requirements) of each part i
Output: N_i : Production requirements of each part i to satisfy all internal and external demands
Output: V_i : Value coefficient for each part i
Output: h_i : Per-unit per-period holding cost for each part i
Output: O : The set of all external demand line items
Output: I^o : Mapping of the part i requested in line item o
Output: D_o : Requested due date of each line item o
Output: Q_o : Quantity of end item requested for each line item o
Output: P_o : Per-period penalty for tardiness of shipped units satisfying line item o
Output: ϵ_o : Per-period reward for earliness of shipped units satisfying line item o
Output: $\overline{P}_o$: Per-unit penalty for unfulfillment of requrested units satisfying line item o

As mentioned, any general product structure network can be represented as an assembly product structure network. We leverage this translation to derive mechanisms which provide visibility and transparency to the resulting production plan and execution schedule. As a reminder, an assembly representation of a product structure restricts the number of parents that any *node* in the BOM network can have to be no more than 1. In cases where a part in the general product structure BOM network, G, has more than 1 parent, the node must be duplicated for as many parents it has. During this translation, each duplicated node must be renamed such that it has with a unique identifier. We refer the reader to Appendix C for a description of the algorithm and naming conventions implemented for this procedure.

Following the generation of all external demands, $o \in O$, an assembly product structure network (ASPN) is created for each `LineItem`. This sub-network, $G^o \subseteq G$ is defined as the subset of parts from G which are required to produce the end item associated with order o. The sub-network, G^o, is transformed from a general structure into an ASPN, Y^o, and new node identifiers, f, are systematically generated for each of the nodes of the new network. Note that node identifiers associated with unique line items must be unique, even if the end item demanded in the line item is the same. The collection of all node identifiers across all orders, is $f \in F$. Similarly, the composition of all ASPN sub-networks is Y.

A `Node` object is created for each node in the network Y and is populated with attributes that provide mappings back its representative part in the general product network, G. These attributes include: 1) the part, i, which node f represents, $i \leftarrow I^f$, 2) the line item, o, which node f is a production requirement for, $o \leftarrow O^f$. Each `Node` also inherits all relevant attributes from its representative part i and order o, i.e. holding cost, h_f, units-per-parent between nodes f and k, n_{fk}, etc. Additional subsets mappings are created to related each node to its original part and order. For example, $f \in F^i$ represents the set of all nodes which represent part i, and $f \in F^o$ is the set of all nodes that are required to produce order o. The benefits of the transparency gained in this transformation will be explored throughout this dissertation.

2.5.4 Generating Resources and Production Routings

The Bill of Operations (BOO) describes all of the data associated with the processes and resources required to produce all of the parts described in the Bill of Materials. This includes the

description of all `Resource`, `ResourceGroup` and `Process` objects. In the following, we present the procedure for generating these objects.

The first phase in generating the Bill of Operations is initializing a user-defined number of resource groups, $|J|$. Each `ResourceGroup` object has a unique identifier, j, and belongs to exactly one `ResourceGroupClass`. The set of all resource groups is J. Each resource group is comprised of one or more resources. The number of resources which make up each resource group is generated as $|E^j| = \mathtt{U}[\,1, \overline{E}\,]$. A total of $|E^j|$ `Resource` objects are initialized for each resource group, j. Each resource, e, belongs to exactly one resource group. The set of all resources belonging to resource group j is $e \in E^j$, and the set of all resources is $e \in E$.

The user-defined parameters within a `ResourceGroupClass` dictate the parts which each resource/resource group is capable of processing. Specifically, this is controlled by the parameters, $\underline{\lambda_j}$ and $\overline{\lambda_j}$, representing the minimum and maximum BOM levels of any part that can be processed by the resources in the resource group, j. This allows the user to define cases such as: 1) a packaging resource group restricted to processing only finished goods, i.e. parts that have a BOM level $\lambda_i < 1$, 2) a receiving & inspection resource group restricted to processing only raw materials, i.e. parts that have a BOM level $\lambda_i = \overline{\lambda}$.

The specification of $[\underline{\lambda_j}, \overline{\lambda_j}]$ also dictates the *processing phase* which a resource group belongs to. This classification establishes a mapping of each resource group, to a specific target utilization rate as specified in the user-defined utilization profile, U. Specifically, the processing phase, u_j associated with a resource group, j, is found as:

$$u_j = |U| - \left\lceil \frac{|U|}{\overline{\lambda}} \frac{\left(\underline{\lambda_j} + \overline{\lambda_j}\right)}{2} \right\rceil$$

where $\overline{\lambda}$ is a user-specified parameter describing the maximum BOM level of any part, and $|U|$ is the number of processing phases described in the user-defined parameter for the utilization profile, U. Consider an example where $\underline{\lambda_j} = 0, \overline{\lambda_j} = 2$, $\overline{\lambda} = 5$, and $U = [50, 70, 90]$ such that $|U| = 3$. Then:

$$u_j = 3 - \left\lceil \frac{3}{5} \frac{0+2}{2} \right\rceil = 3 - \left\lceil \frac{3}{5} \right\rceil = 3 - 1 = 2$$

The target utilization, U_j, of resource group j is the u_j-th element in U (0-indexed), in this case $U_j = 90$. If instead, $\underline{\lambda_j} = 3, \overline{\lambda_j} = 5$; $u_j = 3 - \left\lceil \frac{3}{5} \frac{3+5}{2} \right\rceil = 3 - \left\lceil \frac{12}{5} \right\rceil = 3 - 3 = 0 \rightarrow U_j = 50$.

Each resource group also inherits additional attributes, on the basis of the resource group class it belongs to, which dictate the generation of attributes associated with the processes it will be assigned to be responsible for. The full procedure for generating `ResourceGroup` and `Resource` objects is shown in Algorithm 3.

Algorithm 3: Procedure for the generation of resource groups and resources

Input: $|J|$: User-defined number of resource groups to be generated
Input: U : User-defined utilization profile
Input: $\overline{\lambda}$: User-defined maximum BOM network depth (offset levels)
Input: $\overline{E}$: User-defined maximum number of resources for any resource group

Initialize set of all resource groups, $J = \emptyset$ and resources $E = \emptyset$
for $j \in |J|$ **do**
  Initialize a resource group object, $j \leftarrow$ `ResourceGroup`
  Assign j to a resource group class, $c \leftarrow$ `randomChoice`(`ResourceGroupClass`)

  `# Inherit attributes from ResourceGroupClass`
  Let $P(S)_j \leftarrow P(S)_c$ be the probability that any process on this resource group will require a setup
  Let $T^j \leftarrow T^c$ be the maximum cycle time of any process on resource group j
  Let $\underline{S}_j \leftarrow \underline{S}_c$ and $\overline{S}_j \leftarrow \overline{S}_c$ be the min/max proportion of T^j spent for setup activities of any process on resource group j
  Let $\underline{\lambda_j} \leftarrow \underline{\lambda_c}$ and $\overline{\lambda_j} \leftarrow \overline{\lambda_c}$ be the min/max BOM levels of parts j can process

  `# Associate` j `with a processing phase and target utilization`
  Let $u_j = |U| - \left\lceil \frac{|U|}{\overline{\lambda}} \frac{\left(\underline{\lambda_j} + \overline{\lambda_j}\right)}{2} \right\rceil$ be the processing phase of j
  Let U_j be the u_j^{th} element in U, and be the target utilization rate of j

  `# Initialize Resource objects`
  Let $|E^j| \leftarrow \mathtt{U}\left[1, \overline{E}\right]$ be the number of resources belonging to resource group j
  for $e \in |E^j|$ **do**
    Initialize a `Resource` object that belongs to j
    Add new `Resource` to set of all resource groups, $E \uplus e$
  Add new `ResourceGroup` to set of all resource groups, $J \uplus j$

Output: E : The set of all resources
Output: J : The set of all resource groups
Output: T^j : The timeline associated with each resource group, j
Output: U_j : Target utilization rate of each resource group, j
Output: $\underline{\lambda_c}$, $\overline{\lambda_c}$: Min/Max process-able BOM level for each resource group, j
Output: $P(S)_j$: Prob. of setup being required for any process on each resource group, j
Output: $\underline{S_j}$, $\overline{S_j}$: Min/Max possible setup time of any process for each resource group, j

Following the generation of resources and resource groups, production routings are created for each part. A part's production routing is defined as the series of processes required to transform that part's children, K^{i+}, into the part, i. Each route is composed of one or more route-steps and each route-step is described by a `Process` object. Each process can only be completed by one resource group. In the following, we propose our methodology for generating routes and initializing `Process` objects.

The production routing of a part is defined as an ordered-list of the resource groups which will process it. The first step in defining the route is to identify which resource groups are capable of processing it. The subset of resource groups capable of processing a part, i, is dependent on the BOM level of the part, λ_i, and minimum/maximum process-able BOM levels of each resource group: $[\underline{\lambda_j}, \overline{\lambda_j}]$. The subset of resource groups, J^{i+}, capable of processing the part i, is defined as:

$$J^{i+} \subseteq J : \underline{\lambda_j} \leq \lambda_i \leq \overline{\lambda_j}$$

Following the identification of the capable resources, $j \in J^{i+}$, a routing is defined by randomly selecting s_i resources from J^{i+}. If cyclical routings are allowed, the selection of resources from J^{i+} is made **with replacement**. As a reminder, s_i is the number of route-steps required to produce part i. Let J^i be the set of randomly chosen resource groups from the eligible resource groups, where $J^i \subseteq J^{i+}$. Once the collection, J^i, has been selected as the routing of part, i, a `Process` object, with identifier γ, is initialized for each route-step. The set of all processes is $\gamma \in \Gamma$, and the set of all processes in the routing of part i is $\gamma \in \Gamma^i$. Additionally, the part associated with any process, γ, is $i \leftarrow I^\gamma$, and the resource group responsible for processing it is, $j \leftarrow J^\gamma$. The procedure for generating part routings and initializing process objects is summarized in Algorithm 4.

2.5.5 Generating Processes and the Bill of Operations

Once all `Process` objects are initialized for all parts, their attributes (i.e. processing rates and maximum batch sizes) are populated simultaneously. The objective in populating `Process` attributes is to balance the expected workload of each resource group (required to complete all of

Algorithm 4: Procedure for generating production routings and initializing process objects

Input: I : The set of all parts
Input: J : The set of all resource groups

Initialize set of all processes, $\Gamma = \emptyset$
Initialize set of processes belonging to j, $\Gamma^j = \emptyset$

```
# Create the routing for each part
```
for $i \in I$ **do**
  Let λ_i be the BOM level of this part, i
  Let s_i be the number of steps needed to produce this part
  Let $J^{i+} \subseteq J : \underline{\lambda_j} \leq \lambda_i \leq \overline{\lambda_j}$ be the subset of resource groups capable of processing i
  Let J^i be a random subset of s_i resources from J^{i+} (WITH replacement)

```
  # For each step in the routing, initialize a Process object
```
  for $j \in J^i$ **do**
    Initialize a `Process` object, γ
    Let s_γ be the step in the routing of this `Part` that this process is
    Let $I^\gamma \leftarrow i$ be the `Part` being processed by γ
    Let $J^\gamma \leftarrow j$ be the `ResourceGroup` responsible for processing γ
    Add new process to set of all processes, $\Gamma \uplus \gamma$
    Add new process to set of all processes, $\Gamma^j \uplus \gamma$

Output: Γ : The set of all processes, $\gamma \in \Gamma$
Output: I^γ : Mapping of the part i associated with process γ
Output: J^γ : Mapping of the resource group j associated with process γ
Output: J^i : The subset of resource groups which are required to produce part i
Output: Γ^j : The set of processes which resource group j is capable of
Output: s_γ : The step number in the routing of the part I^γ, which each process, γ is

the production requirements of each part it is responsible for processing) with the target utilization rates of each resource group, U_j.

Our logic in generating these attributes assumes three characteristics relating a process, a, which has a larger production requirement (i.e. $N_i = 1000$ units over the course of the planning horizon), to another process, b, which has a lower production requirement (i,e. $N_i = 100$ units over the course of the planning horizon). The following assumptions are considered:

- Process a is likely to have a per-unit processing rate lower than process b
- Process a is likely to consume more resource capacity than process b

- Process a is likely to have a maximum allowable batch size larger than process b

We acknowledge that this relationship between the total demand requirements of a process are not likely linearly correlated with total demand of a process‘ output and that many exceptions exist to the assumptions stated above. We also acknowledge that these attributes are highly dependent on the types of processes, resource groups and parts which are produced. However, for the purpose of this research, we must define an approximate transformation which will provide a relatively accurate reflection on the relationship between processing rates, maximum batch sizes and total production requirements. We propose the procedure, shown in Algorithm 5, to populate attributes for setup times, processing rates, maximum batch sizes, etc.:

Algorithm 5: Procedure for generating processing rates, setup times and maximum batch sizes with objective of balancing workload with target utilization

Input: I : The set of all parts
Input: J : The set of all resource groups
Input: Γ : The set of all processes
Input: $\overline{T}$: The length of the planning horizon, in hours

```
# For each resource group
```

for $j \in J$ **do**

```
# Get target usage and subset of processes j is responsible for
```

Let $\overline{C}_j = |E^j| * U_j * \overline{T}$ be the target capacity of j
Let $\Gamma^j \subseteq \Gamma$ be the subset of all processes which are processed by j

```
# Get weightings for each process on this resource group
```

for $\gamma \in \Gamma^j$ **do**
Let $N_\gamma = N_i$ be the total output requirement of process γ
Let $f(N_\gamma)$ be a transformation function of N_γ, i.e. $log(N_\gamma)$ or $\sqrt{N_\gamma}$
Let $W_{\Gamma^j} = \sum_{\gamma \in \Gamma^j} f(N_\gamma)$ be the sum of the transformed requirements of $\gamma \in \Gamma^j$

```
# Get the proportion of time dedicated for each process on this resource
  group, then set Process attributes accordingly
```

for $\gamma \in \Gamma^j$ **do**
Let $P_\gamma = f(N_\gamma)/W_{\Gamma^j}$ be proportion of transformed requirements of γ over Γ^j
Let $\overline{C}_\gamma = \overline{C}_j P_\gamma$ be the dedicated capacity of resource group, j for process, γ

Let $S^*_\gamma = \texttt{Bernoulli}(P(S)_j) \times \texttt{U}[\underline{S}_j, \overline{S}_j]$ be the proportion of per-unit setup rate vs processing rate (assuming a max batch run)
Let $R^*_\gamma = 1 - S^*_\gamma$ be the proportion of per-unit processing vs setup rate
Let $\overline{R}_\gamma = \overline{C}_\gamma \; R^*_\gamma$ be the dedicated processing time for this process γ

Let $R_\gamma = \overline{R}_\gamma / N_\gamma$ be the per-unit processing rate of process γ
Let $S_\gamma = T^j \; S^*_\gamma$ be the setup time associated with each batch of process γ
Let $M_\gamma = (T^j - S_\gamma)/R_\gamma$ be the maximum batch size of process γ

Output: S_γ : The setup time of each process, γ
Output: R_γ : The per-unit processing rate of each process, γ
Output: M_γ : The maximum allowable lot size per-period of each process, γ

Following the generation and population of all `Process` attributes, the attributes of each `Part` object can be populated with relevant lead time data. The lead time of part i is L_i, and is estimated to be the sum of the maximum cycle times associated with each process in its routing. This calculation is found as:

$$L_i = \sum_{\gamma \in \Gamma^i} (M_\gamma R_\gamma) + S_\gamma$$

Further, we define the expected *lead time offset* of when a part will reach process γ once it has begun processing at its first route-step as l_γ. This value is calculated as:

$$l_\gamma = \sum_{\substack{a \in \Gamma^i: \\ s_a < s_\gamma}} (M_a R_a) + S_a$$

where s_γ is the step number in the routing of i. Note that the lead-time offset of the first route-step is $l_\gamma = 0$. Further, the lead-time offset of the last route-step will be L_i minus the cycle time of the last route-step. Note that both of these lead time parameters (L_i, l_γ) assume that the batch of size of a process is always M_γ. This results in an over-estimation of the lead times and acts as a buffer within the Production Planning Module and Scheduling Simulation Module.

To summarize, in this section we have described the algorithms and procedures developed to generate a test instance of a generalized production facility with general product structures, and multi-step, cyclical routings. The outputs of the SGM are shown in Table 2.5.

2.6 Production Planning Module

The purpose of the Production Planning Module (PPM) is to facilitate the application of traditional production planning, scheduling, revenue management models and/or heuristics to the test instances generated in the SGM. Due to the *plug-and-play* capability of the proposed Digital Twin framework, the method/model used in the PPM is extremely flexible. Within the Digital Twin framework, a PPM application can be an optimization model, meta-heuristic (i.e. simulated annealing, genetic algorithm, ant colony optimization, etc) or a priority- or dispatching-heuristic. Throughout this dissertation, we develop and evaluate optimization models which plug-into the PPM for various decision-making problems.

Due to the complexity of real-world problems, researchers and practitioners are limited in the amount of detail which can be considered in optimization models. For example, it is not realistic to consider a continuous-time planning horizon in a long-term production planning problem. Instead, a discrete-time horizon with time increments at a daily/weekly/etc. aggregation would need to be applied. The solutions derived in an aggregated context such as this are rarely evaluated in the unaggregated context which they consider. However, in practice, the true effectiveness of an application must be considered in that unaggregated context. This disparity represents a major gap in the literature.

The input of the PPM includes all of the generated objects and associated attributes from the Scenario Generation Module (see Table 2.5). The output of the PPM, which acts as input to the SSM, includes a collection of dynamically generated `Shipment` objects and `WorkOrder` objects, $w \in W$, and `Task` objects, $x \in X$, which satisfy production requirements from the generated factory instance. Note that both shipments and work orders belong to the set of *orders*, W; they are considered similarly in the Scheduling Simulation Module.

Each shipment, w, calls for the delivery of a quantity of a part, i, which satisfies a requirement of a customer line item, o, during time period t. Each work order, w, calls for the production of a quantity of a specific part, i, to begin production during a specific time period, t. The time, t, associated with each generated shipment and work order in the PPM is referred to as the *planned start time*. The generation of a `WorkOrder` object, w, will also trigger the generation a collection of `Task` objects, $x \in X^w$, which must be completed in order to produce the part specified in that work order. These tasks are generated based on the `Process` objects defined in the routing of the part associated with the work order. Each `Task` object represents an instruction for executing an operation as described in the appropriate `Process` object.

2.6.1 A MRP-based Production Planning Heuristic

In the following, we present an MRP-based heuristic to illustrate an implementation the proposed Digital Twin framework (See Section 2.8). The proposed MRP-based heuristic (Algorithm 6) is capable of generating a production plan for a multi-level, multi-product, lot-sizing production environment. The heuristic does not consider capacity constraints, similar to traditional Material Requirement Planning (MRP) methods. The objective of the heuristic is to generate one produc-

tion `WorkOrder` for each production requirement, $f \in F$, and one `Shipment` for each customer order, $o \in O$.

In this heuristic, the planned start time of each production `WorkOrder` is found as the minimum of: 1) the cumulative lead time of the associated production requirement, CLT_f, and 2) the latest possible starting time of the production requirement without resulting in the late delivery of the order, o. The planned start time of each `Shipment` is the minimum of: 1) the due date of the order, D_o, and 2) the cumulative lead time of the finished good deliverable, $CLT_k : k \leftarrow F^o$. The heuristic is presented in Algorithm 6.

2.7 Scheduling Simulation Module

The purpose of the Scheduling Simulation Module (SSM) is to transform the aggregated, discrete-time, lot-size based production plan from the PPM into a continuous-time, assignment-based schedule with enough detail to be executed on a shop floor. To accomplish this, we develop the algorithm: Continuous-Time Scheduling Heuristic (CTSH). The proposed CTSH is a myopic, forward-constructed scheduling heuristic that iteratively schedules all of the `Shipment`, `WorkOrder` and `Task` objects generated in the PPM.

Throughout the progression of the CTSH, each `WorkOrder` and `Task` object is assigned to be completed by a specific resource and/or sent out for delivery to fulfill a specific external demand. The start and end time of each production task and the inventory levels of each part is tracked in continuous-time. The results of the simulated events are then able to be analyzed and visualized (see Section 2.8).

The primary challenge in translating a discrete-time production plan into a continuous-time schedule concerns the relaxation of assumptions made to formulate a tractable production planning model. Several examples of these assumptions include: the discretization of time, the aggregation of resources into resource groups, simplified representations of complex setups, etc. In an executable schedule, resource capacity can no longer be aggregated across a resource group. Instead, each task must be *assigned* to begin processing at a specific time on a specific resource within the required resource group. Relaxing this assumption introduces additional assignment constraints, i.e. each resource is only capable of processing one task at a time. When considering resource assignment, maximum batch sizes must also be considered. The production plan solution generates work orders

Algorithm 6: MRP Benchmark Algorithm

Input: I : The set of all parts
Input: O : The set of all line items
Input: F : The set of all nodes (production requirements)
Input: CLT_f : The cumulative lead time of each node, f
Input: D_o : The requested due date of each line item, o
Input: L_i : The lead time of each part, i

```
for each node in the production requirements, f ∈ F do
    Let o ← O^f be the line item associated with node f
    Let k ← F^o be the end item node associated with line item o

    Let TTC_f = CLT_k − CLT_f be the "time to completion" of the finished good, k, once
      node f has been produced
    Let MSB_f = D_o − TTC_f − L_i be the "must start by" time of production of node f in
      order NOT to delay the delivery of order o past its due date

    if MSB_f ≥ CLT_f then
        Let t_1 ← MSB_f be the start time of production of node f
    else
        Let t_1 ← CLT_f be the start time of production of node f

    Generate a WorkOrder, w, for a quantity of N_f of part i to begin at time t_1
    Generate a Task, x, for each route-step in the WorkOrder, w

    if f = k (this node IS the deliverable) then
        Let t_2 = t_1 + L_i be the expected completion time of node f
        Generate a Shipment, w, for qty N_f of part i to satisfy order o at time t_2
```

Output: W : The set of all dynamically generated `Shipment` and `WorkOrder` objects
Output: X : The set of all dynamically generated `Task` objects
Output: TTC_f : The time to completion of the associated end item for each node, f
Output: MSB_f : The must start by time of each node, f, to avoid a tardy delivery of the associated line item o

at the *production lot* level. However, these lots may be for a quantity that is larger than the maximum *batch* size of a resource required in the part's routing. In this case, the production lot must be split into batches which comply with the maximum batch constraints of each resource. Batches may be processed in parallel by different resources belonging to the same resource group.

In cases which a production plan calls for multiple tasks to be processed by the same resource group during the same time period, we must establish a prioritization scheme that ranks each competing task such that the most "critical" tasks are given priority in the resource assignment

procedure. We are able to leverage information from each work order (and the production requirements that it satisfies) to formulate these prioritization schemes. Several examples of prioritization schemes include sorting by: 1) the current *lateness* of each work order (relative to its planned start time), 2) the expected delay penalty associated with all of the `LineItem` objects the work order satisfies, etc.

We now introduce several dynamic subsets of all shipments and work orders, $w \in W$, that represent the *state* of each `Shipment` and `WorkOrder` object at any given time in the simulation. For simplicity of exposition, we refer to the collection of all shipments and work orders as the set of all "orders". We denote the following:

- W^0: is the set of orders which are *unreleased* to the shop floor. A work order will remain in this state until a user-defined time prior to its *planned start time* as specified in the production plan
- W^1: is the set of orders which are *released* but not yet scheduled. These are the orders which will be attempted to be scheduled during the current epoch of the CTSH
- W^2: is the set of all *scheduled* but not yet completed orders. These orders are considered Work-In-Progress
- W^3: is the set of all *completed* orders.

Orders progress through these states in the following order: $W^0 \rightarrow W^1 \rightarrow W^2 \rightarrow W^3$.

The CTSH algorithm is initialized by updating the inventory levels for each part's initial inventory. Then all orders, $w \in W$, are added to the set W^0. The current time in the simulation is kept with the variable, t, and is initialized at $t = 0$. The simulator will run until the maximum time period in the planning horizon, $\overline{T}$.

During each epoch, t, the following steps will occur. First, all orders belonging to the unreleased order set, W^0, having a planned start time less than t, are moved to the released, unscheduled order set, W^1. Then, any scheduled orders, $w \in W^2$, having a scheduled completion time less than the current time, t, are moved to the set W^3. The inventory levels of each part associated with the now completed orders are updated accordingly. Following this, each of the unscheduled, released orders, $w \in W^1$, are prioritized using a user-defined prioritization scheme. In an iterative fashion,

each work order in the sorted set W^1 is attempted to be scheduled. A released order, w, is eligible to be scheduled if all required part inventory is available.

If the order is a `Shipment`, the inventory level of the associated part must be greater than the quantity specified to be delivered in the order. If this condition is satisfied, the delivery is simulated to be sent to the customer. The order, w, is moved from $W^1 \rightarrow W^2 \rightarrow W^3$.

If the order is `WorkOrder`, the inventory level of each child of the part specified in the work order is evaluated. If all required components have enough inventory, then all `Task` objects specified in the `WorkOrder` will be scheduled. If the work order production lot quantity is larger than the maximum batch size associated with ANY of the tasks in the part's routing, then the lot is split into an appropriate number of batches. Each batch is then scheduled for each task in the work order. Batches are assigned to be processed by a specific resource in the resource group associated with the process specified by each task. Specifically, each batch is assigned to be completed by the resource which is able to complete it the earliest. Restrictions specify that a batch can not be started before the current time, t, or prior to the completion time of any earlier route-steps. Further, the existing schedule of a resource can not be modified when scheduling a batch.

Each batch is allowed to move on to the subsequent route-step following its completion at the current route-step. However, each batch will remain in a Work-in-Progress state until all batches in the production lot have completed all route-steps. Restricting completed batches within a production lot to remain in WIP represents the practice of "paper routing" which allows manufacturers to trace parts throughout the factory as they progress along the value chain. This concept is discussed further in Chapter 3.

Note that this particular representation of paper routing represents a loose enforcement of the practice. A stricter enforcement could restrict batches to remain in WIP at each route-step until all batches in the production lot are completed prior to moving to the next route-step. Further, we acknowledge that this batch-resource assignment method may result in sub-optimal schedules due to the restriction that the existing schedule of each resource can not be modified, even if a modification may result in a superior schedule. A potential enhancement to the proposed CTSH is to allow for a schedule to be modified in cases which a modification would improve the schedule's overall performance.

Following assignment, all related object attributes are updated accordingly. If the task is the first step in the work order, the inventory levels of all required component parts are updated accordingly. Once all tasks in the work order, $x \in X^w$, are scheduled, the work order, w, is moved from $W^1 \rightarrow W^2$. The scheduled completion time of the work order is set to be the completion time of the last batch at the last task in that part's routing.

If a work order can not be scheduled due to lack of available inventory, it will remain in a released, unscheduled state, W^1, until it can be scheduled. Once all released, unscheduled orders, $w \in W^1$, have attempted to be scheduled, the current time, t, is incremented by a user-defined t_{step}, and the next epoch begins. An overview of the logic flow of the CTSH is shown in Algorithm 7.

Algorithm 7: Continuous Time Scheduling Heuristic: Overview

```
Set t = 0 , t_step = 1, Initialize inventory
while t ≤ α do
    Move all released, unscheduled orders, W^0 → W^1
    Move all completed orders, W^2 → W^3, release inventory from WIP
    Calculate and sort orders, w ∈ W^1, by prioritization metric
    for each released order in sorted set, w ∈ W^1 do
        if All required component inventory is available then
            if the order is a Shipment then
                Simulate a delivery
                Move Shipment, w, from released W^1 → W^2 → W^3 completed
            else
                Split work order into batches
                for batch in work order do
                    for task in WorkOrder, x ∈ X^w do
                        Add batch to the schedule of a Resource
                        if Task is first step in WorkOrder then
                            Update inventory of children parts accordingly
                Move WorkOrder, w, from released W^1 → W^2 scheduled
    t ← t + t_step
```

2.8 Illustrative Example

In the following, we present a series of outputs from the developed Digital Twin framework which describe and showcase the power of integrating object-oriented programming with mathematical programming. The objective of this section is to explore a generated test instance and analyze the solutions obtained by the proposed production planning models (presented in Chapter 3) and the CTSH (presented in Section 2.7). Specifically, we provide an illustrative example of a scenario faced by job shop facility representative of an aerospace manufacturing facility capable of producing complex, multi-level products.

In this illustrative example, we consider a factory which is comprised of 50 unique parts, 6 resource groups and 25 customer orders. Products are represented in a general product structure. Cyclical and multi-step routings are allowed. The production facility is structured into 3 production phases, to differentiate the characteristics of early, middle and final stage processes. Each with a target utilization of 0.80. Each part belongs to one of four part classes and each resource group belongs to one of six resource group classes (see Table 3.6 and 3.7). The Bill of Materials of any part can have a maximum of 4 offset levels, and each part can have up to 5 children. The numer of units per parent are in the range [1, 5] and the routing of each part can have up to 5 route steps. Each resource group is made up of up to 3 resources. Each customer order represents a demand for a single part (belonging to part class `FinishedGood`) for a quantity in the range [1,20]. The planning horizon spans 26 weeks, with 40 hours in a week. Three timelines are considered in the planning horizon with time steps of 10 hours, 20 hours and 40 hours, respectively.

Parameter	Notation	Default Value	Description
`prodStr`		General	Specification of Product Structure scheme
`resAssign`		Cyclical	Specification of Resource Assignment scheme
`routeComp`		Multi-Step	Specification of Routing Complexity scheme
`utilProf`		[80, 80, 80]	Specification of Utilization Profile scheme
`numI`	$\|I\|$	50	Number of unique items
`numJ`	$\|J\|$	6	Number of unique resource groups
`maxT`	$\|T\|$	26	Length of the planning horizon (weeks)
`numLI`	$\|O\|$	25	number of external demand orders
`numSR`	$\|A\|$	0	number of scheduled receipts
`maxRperRG`	$\|\overline{E}\|$	3	max number of resources per resource group
`maxLiQty`	$\overline{Q}$	20	Max qty. associated with any line item
`maxLev`	$\overline{\lambda}$	4	Max level in a BOM
`maxUPP`	$\overline{n}$	5	Max units-per-parent of any parent-child relationship

Table 2.6. Illustrative example: User-defined parameters, notation, and values - `Factory` parameter model

`className`	Raw Material	MachinedPart	AssembledPart	FinishedGood
`density`	0.1	0.4	0.3	0.2
[`minChild`, `maxChild`]	[0, 0]	[1, 3]	[1, 5]	[1, 5]
[`minParent`, `maxParent`]	[1, 10]	[1, 3]	[1, 3]	[0, 0]
[`minLevel`, `maxLevel`]	[4, 4]	[1, 3]	[1, 2]	[0, 0]
[`minRS`, `maxRS`]	[0, 0]	[1, 4]	[1, 3]	[1, 2]

Table 2.7. `PartClass` parameters used throughout numerical implementation

`className`	RG1	RG2	RG3	RG4	RG5	RG6
[`minLevel`, `maxLevel`]	[2, 3]	[2, 3]	[0, 3]	[0, 3]	[0, 1]	[0, 1]
[`minS`, `maxS`]	[0, 0.5]	[0, 0.5]	[0, 0.5]	[0, 0.5]	[0, 0.5]	[0, 0.5]
`probS`	0.25	0.75	0.50	0.50	0.25	0.75
`timeline`	20	20	10	40	20	20

Table 2.8. `ResourceGroupClass` parameters used throughout the numerical implementation

Throughout this illustrative example, we focus our analysis on a single customer order, `LineItem` 1. Table 2.9 summarizes the Bill of Materials for the end item that is requested in this customer order. Figure 2.4 provides a visual representation of the finished good in both a general product structure (top) and as an assembly product structure (bottom). Each node in the BOM network is annotated with the Part ID which it represents. Note that each part that exists in multiple

assemblies within the end item is represented by several nodes in the assembly product structure network (bottom). The offset (row which the node is drawn in the network) in the general product representation is dictated by each part's BOM level, λ_i. In the assembly network representation (bottom), the offset of each node is dictated by it's *distance* from the finished good.

Table 2.9. Illustrative example: Indented Bill of Materials report of the FinishedGood item associated with `LineItem` 1

	Offset Level					BOM Attributes			
PartID	0	1	2	3	4	UPP	Part Class	# Children	# Parent
31	31						Finished Good	4	0
24		24				2	Machined Part	2	1
7			7			3	Raw Material	0	6
43			43			1	Raw Material	0	14
27		27				3	Assembled Part	1	1
43			43			3	Raw Material	0	14
33		33				4	Machined Part	3	1
14			14			4	Assembled Part	1	1
4				4		1	Machined Part	2	2
7					7	4	Raw Material	0	6
43					43	1	Raw Material	0	14
35			35			4	Assembled Part	2	2
6				6		3	Raw Material	0	8
43				43		4	Raw Material	0	14
42			42			1	Assembled Part	1	3
43				43		2	Raw Material	0	14
39		39				1	Machined Part	1	2
3			3			2	Raw Material	0	4
...	...	...	...	...	...	...	...	...	...

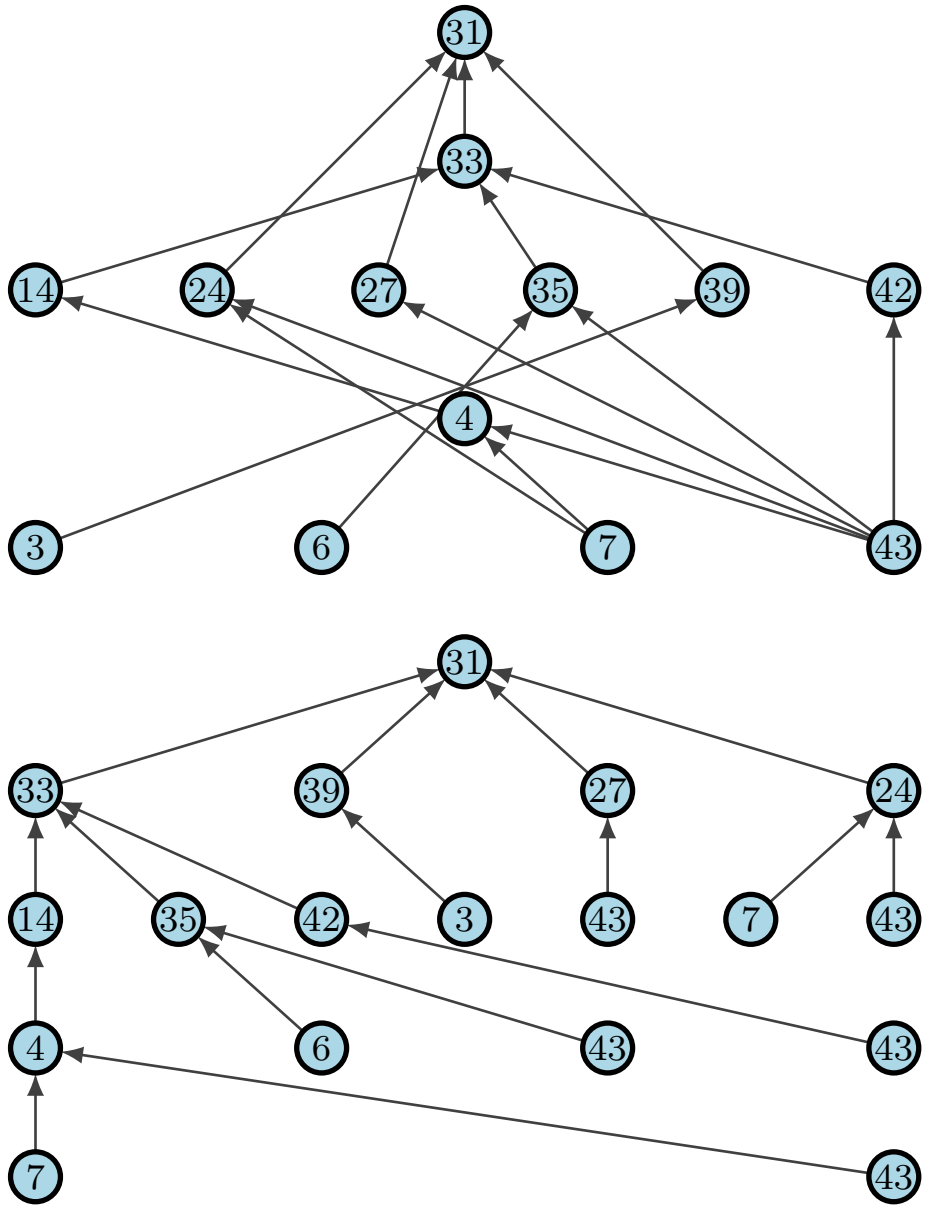


Figure 2.4. Illustrative Example: Generated Bill of Materials of the end item associated with customer order, `LineItem` 1, represented as a general product structure (top) and as an assembly product (bottom).

Table 2.10 describes a subset of the Bill of Operations detailing the processes which are required to produce the finished good of `LineItem` 1. Each *ProcessID* is composed of the *Part* number of the part being produced and the step number of the part's routing. The resource group responsible for each process is shown in column *Resource*, as well as information regarding the setup time, processing rate and maximum batch size of each process. A visual representation of the combination

of the Bill of Materials and Bill of Operations for this end item is shown in Figure 2.5. In this representation, each node in the graph represents either a physical part (square nodes) or a resource group responsible for the process required to produce a part (circle nodes). Each part and resource group node is color coded based on the part class and resource group class associated with each node. Note that a square node is presented twice for each part which has a routing in the Bill of Operations; one for the *start* of the routing (suffix "-s") and one for the *finish* of the routing (suffix "-f").

Table 2.10. Illustrative example: Subset of Bill of Operations for parts required for the Finished-Good item associated with lineItem 1. Rows are sorted in order of part-step.

ProcessID	Part	Step	Resource	Setup Time	Process Rate	Max Batch Quantity	Max Lot Time
4_1	4	1	B	-	2.4	8.5	20
14_1	14	1	D	1.6	0.4	101.0	40
14_2	14	2	D	-	0.4	101.0	40
24_1	24	1	E	-	2.6	3.8	10
27_1	27	1	D	-	1.9	21.4	40
27_2	27	2	F	-	2.7	7.5	20
31_1	31	1	E	0.9	3.9	2.3	10
31_2	31	2	D	-	4.2	9.5	40
33_1	33	1	F	-	1.8	11.4	20
33_2	33	2	A	1.8	1.6	11.4	20
35_1	35	1	E	0.8	0.4	25.4	10
35_2	35	2	A	0.2	0.5	36.5	20
35_3	35	3	D	-	0.4	104.3	40
39_1	39	1	A	-	6.0	3.3	20
39_2	39	2	F	7.4	3.8	3.3	20
39_3	39	3	D	14.4	2.7	9.5	40
42_1	42	1	D	10.8	0.4	78.3	40
42_2	42	2	F	-	0.7	27.5	20

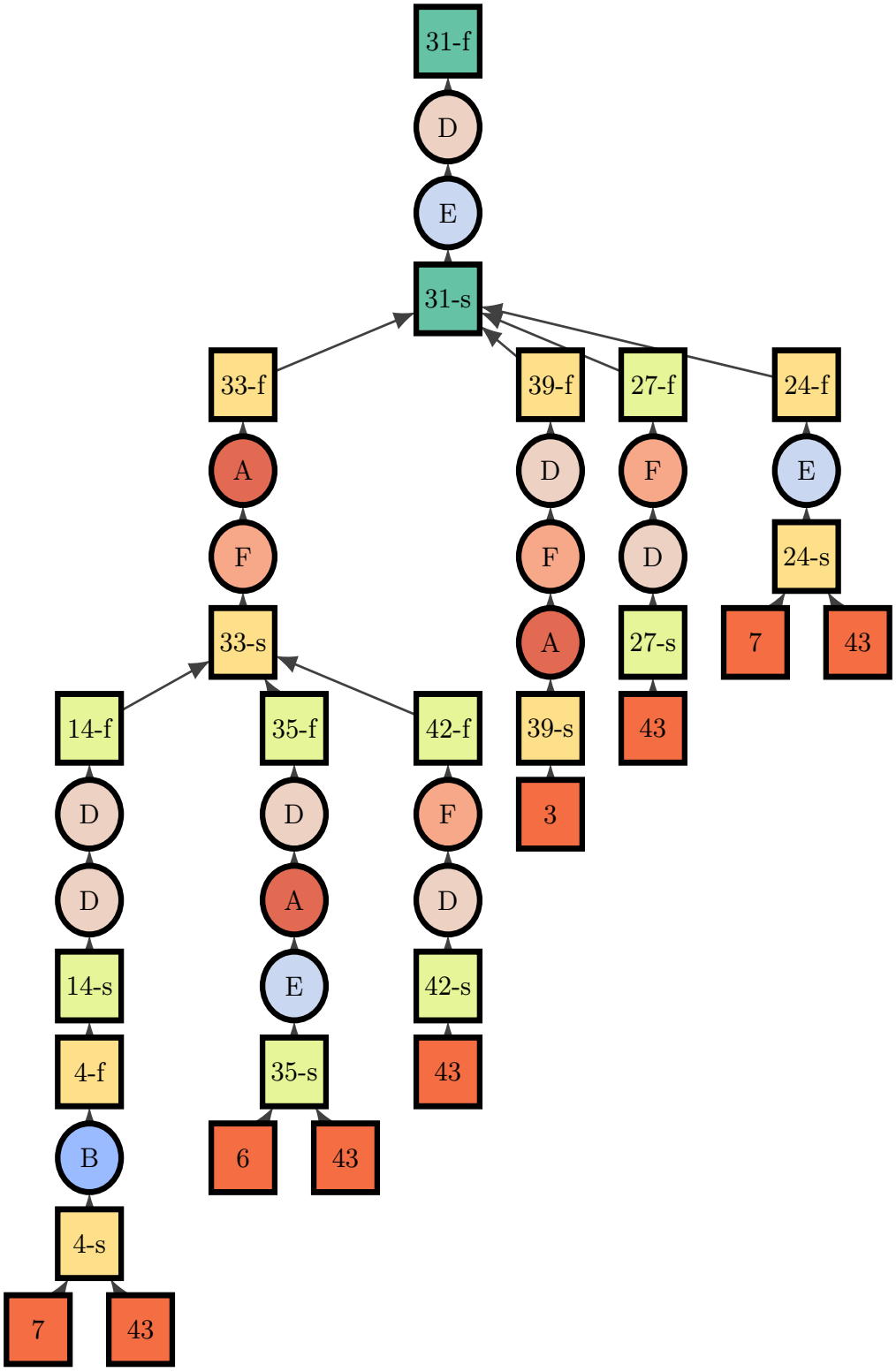


Figure 2.5. Illustrative Example: Generated Bill of Operations of the end item associated with customer order, `LineItem` 1. In this diagram, parts are represented as squares and resource groups which are visited in the production routing of parts as circles. Nodes in this diagram are color coded based on the class of each part/resource group.

Table 2.11. Illustrative example: Subset of dynamically generated `WorkOrder` objects which satisfy the production requirements of parts required for the end item associated with `LineItem` 1.

Work Order	Work Order Details			Completion Time		
	Part	Lot Qty.	# Tasks	Planned	Actual	Delay
20	4	25.4	1	40	20.0	-20.0
23	4	11.8	1	100	80.0	-20.0
24	4	12.5	1	120	100.0	-20.0
27	4	11.8	1	180	160.0	-20.0
88	14	30.3	2	200	125.1	-74.9
89	14	38.3	2	280	244.6	-35.4
162	24	5.4	1	200	252.9	52.9
163	24	3.6	1	240	275.3	35.3
164	24	4.9	1	340	395.3	55.3
172	27	0.3	2	100	1.3	-98.7
173	27	12.9	2	140	114.4	-25.6
174	27	0.3	2	180	120.8	-59.2
175	27	1.5	2	260	220.5	-39.5
202	31	2.7	2	260	430.4	170.4
203	31	1.8	2	290	445.6	155.6
204	31	2.5	2	400	464.8	64.8
252	33	10.8	2	200	338.9	138.9
253	33	7.3	2	240	373.6	133.6
254	33	9.8	2	340	391.1	51.1
268	35	59.5	3	160	205.6	45.6
269	35	66.0	3	200	263.2	63.2
309	39	7.8	3	120	135.3	15.3
320	42	51.0	2	100	60.8	-39.2
321	42	19.9	2	180	151.5	-28.5
323	42	19.2	2	300	268.4	-31.6
...	...	...	...	...	...	...

Table 2.11 shows a subset of the `WorkOrder` objects generated to satisfy `LineItem` 1, as an output of the Production Planning Module and Scheduling Simulation Module. A total of 25 work orders are shown, including a description of the work order and the completion time of the work order in the production plan (*Planned*) and in the simulated schedule (*Actual*). Note that the *Delay* experienced in the simulated schedule can be negative, representing a work order which is completed earlier than in the plan. One cause for this includes an over-estimation in the lead time of the routing of work orders, especially when the lot size of the work order is less than the maximum production lot size for the produced part.

A visual representation of a subset of the realized schedule is shown in Figure 2.6. Specifically, this subset represents the work orders which were dynamically generated to satisfy the production requirements of `LineItem` 1 (see Table 2.9). Each faceted row in this figure represents the realized production schedule of a specific part required to produce this end item. Each row *within* each facet represents a unique work order dedicated to producing that respective part. Each block in the Gantt chart is color coded by the resource group which is responsible for performing each task in the work order.

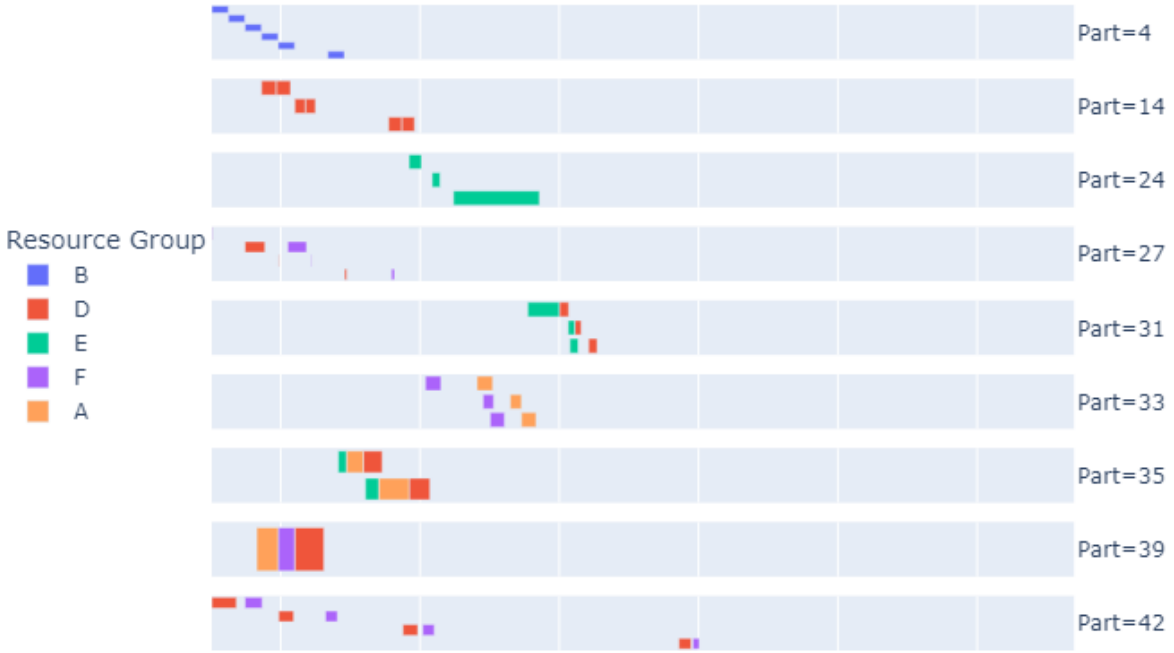


Figure 2.6. Illustrative Example: Gantt chart of dynamically generated work orders which produce parts (right axis) which satisfy the production requirements for the end item associated with external demand, `LineItem` 1. Blocks are color-coded based on the resource group which is visited for each operation. Each row within each facet is representative of a unique work order.

Note that some of these work orders have spaces between the completion time of one task and the start time of the next. This occurs when the Work-In-Progress inventory is waiting in the queue of a resource within the resource group that must process it. As a reminder, queuing is not considered in the Production Planning Module (only the aggregated capacity of the resources in each resource group), and is a cause for delays between the production plan and the simulated schedule.

Figure 2.7 shows another Gantt chart with respect to the schedule of the resources within a one of the resource groups in the generated facility. Each facet row in this figure represents the schedule for a resource within the resource group, each row within the facet represent the part being produced, and blocks are color-coded by the process which is performed.

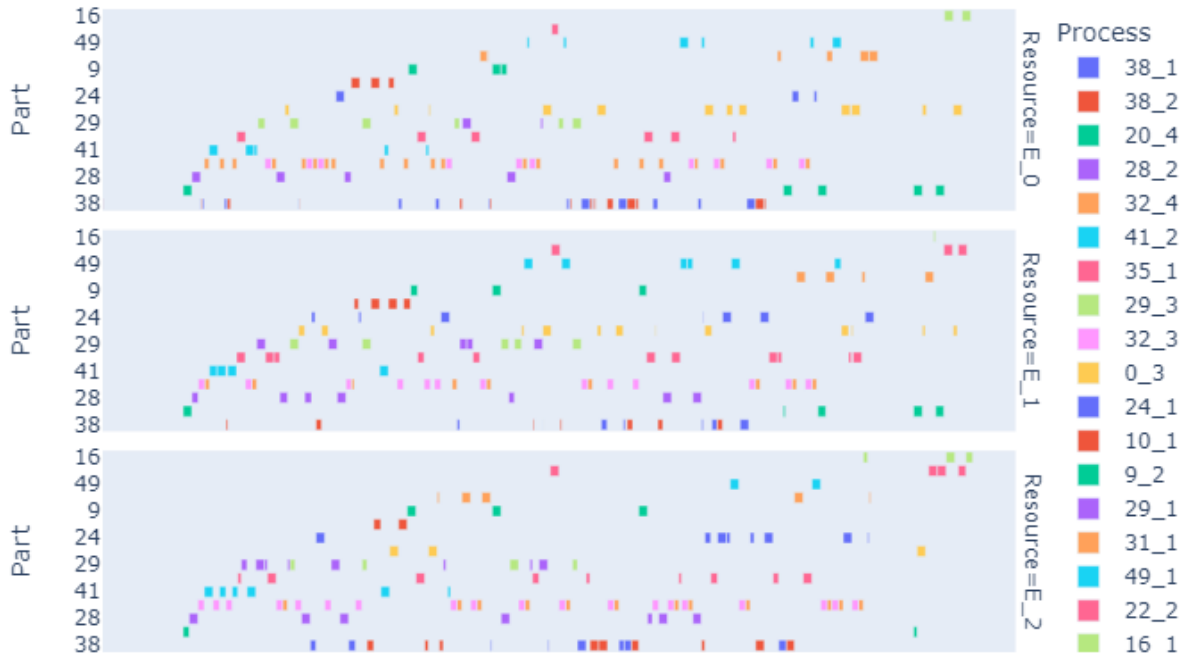


Figure 2.7. Illustrative Example: Gantt chart of dynamically generated work orders which are processed by the resources in a resource group. Each facet row represents a resource in the resource group. Each row within the facet row represents a specific part, and blocks are color coded based on the process which is performed in each work order.

The realized production schedule of these resources can be aggregated to analyze the utilization rates of each resource. An example showing the utilization of all resource groups is shown in Figure 2.8. Each block represents a week within the schedule of a specific resource group in a specific week. Figure 2.9 provides another view, where each block represents a month in the schedule of a specific resource.

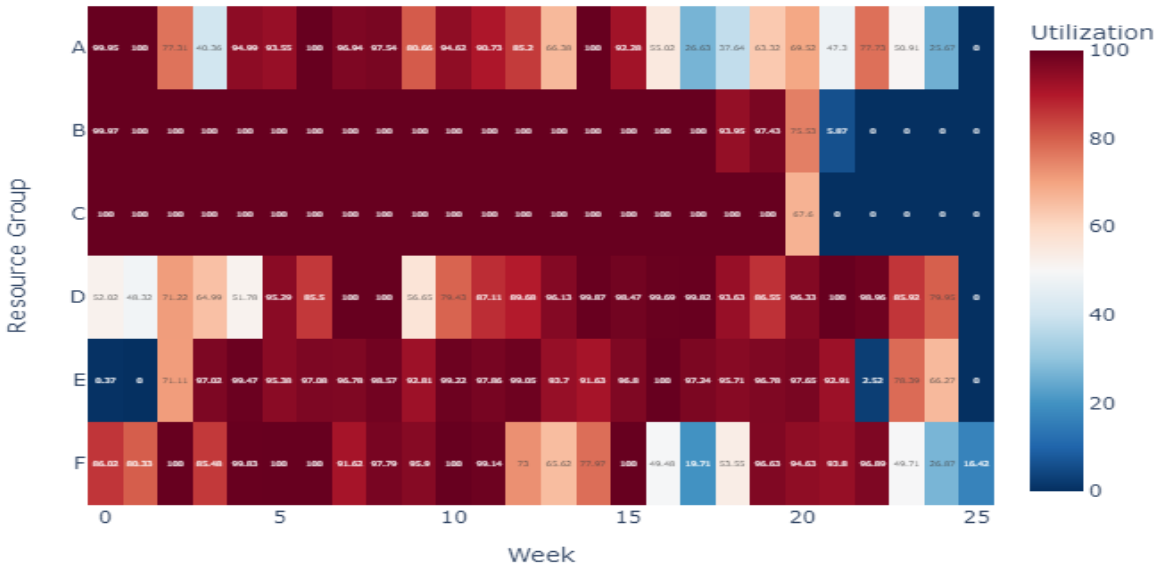


Figure 2.8. Illustrative Example: Utilization rates of each resource group (row) and week (column)

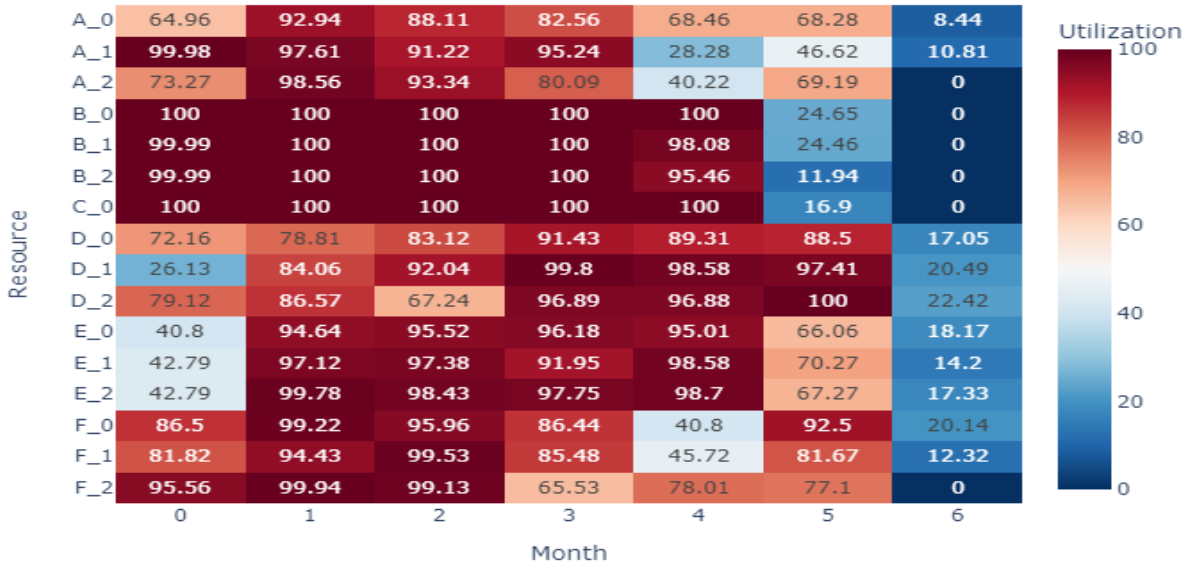


Figure 2.9. Illustrative Example: Utilization rates of each resource (row) and month (column)

Table 2.12 details the shipments generated in the production plan. Note that several shipments may be generated to satisfy each order. The *Lateness* and *Tardiness* of each shipment is shown, where a negative value for *Lateness* represents a shipment that was delivered prior to the associated order's due date. A shipment that is shipped early will have a Tardiness of 0. Each shipment is also evaluated for the quantity of units of the end item which were shipped, the monetary value of those units, as well as whether the shipment "completed" the line item, i.e. shipped the last remaining units requested by the customer.

Table 2.12. Illustrative example: Subset of dynamically generated `Shipment` objects which satisfy external demands.

orderID	ShipID	Due Date	Time Delivered	Lateness	Tardiness	Units	Dollars	Order Completed
0	0	480	360	-120	0	7.4	230.3	0
	1	480	560	80	80	1.6	48.4	1
1	2	850	920	70	70	8.0	512.0	1
2	3	510	520	10	10	3.9	446.2	0
	4	510	460	-50	0	5.9	680.8	0
4	6	270	220	-50	0	2.7	758.6	0
	7	270	250	-20	0	1.8	513.2	0
	8	270	360	90	90	0.5	132.5	1
5	9	190	100	-90	0	3.0	38.7	0
	10	190	520	330	330	2.5	32.8	0
6	11	200	220	20	20	3.3	316.4	0
	12	200	260	60	60	5.3	506.4	0
	13	200	340	140	140	0.5	45.6	0
	14	200	380	180	180	0.3	26.6	0
	15	200	500	300	300	1.3	120.7	1
	16	200	300	100	100	4.3	406.6	0
7	17	440	320	-120	0	9.2	768.6	0
	18	440	390	-50	0	2.6	219.2	0
8	19	920	920	0	0	2.2	33.8	0
	20	920	800	-120	0	0.5	7.8	0
	21	920	880	-40	0	2.6	39.6	0
9	22	390	340	-50	0	3.4	218.9	0
	23	390	360	-30	0	5.9	375.7	0
	24	390	440	50	50	9.7	620.2	1
10	25	250	880	630	630	10.0	50.0	1
...	...	...	...	...	...	...	...	...

These KPI's can be aggregated to evaluate the performance of the production plan from the PPM. Figure 2.10 provides a view of the distribution of the *Lateness* of the shipments of Completed

Orders/Units/Dollars throughout the planning horizon. The X-axis in the histogram represents the lateness of each shipment relative to the due date of the order which each shipment satisfies.

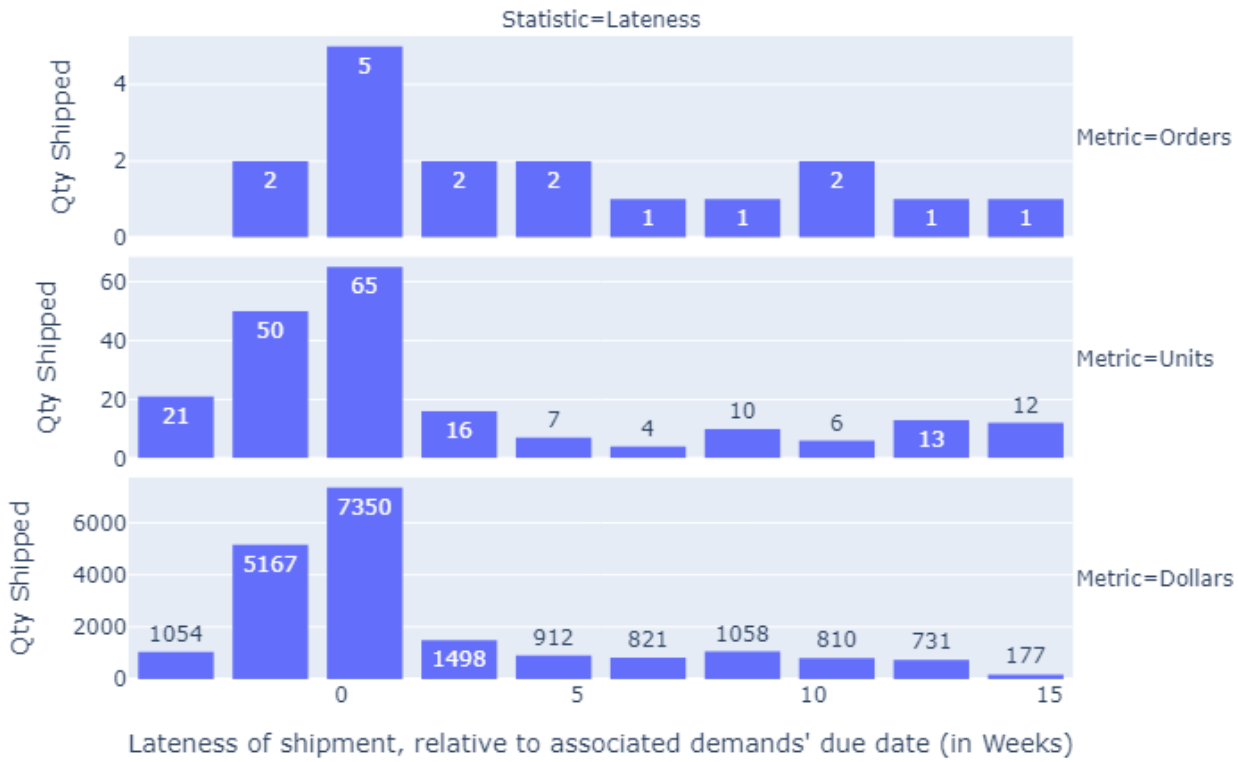


Figure 2.10. Illustrative Example: Histogram of the Lateness of all dynamically generated `Shipment` objects in the planning horizon. Each facet row represents a different aggregation of the product (right axis).

Figure 2.11 presents another view in delivery shipment performance. This visualization is a stacked area chart and shows the size of the existing overdue backlog relative to the total shipments, with respect to completed orders/units/dollars, throughout the planning horizon. Figure 2.12 provides yet another view of delivery shipment performance. This visualization depicts the cumulative requested and shipped orders/units/dollars compared to the existing backlog during the planning horizon.

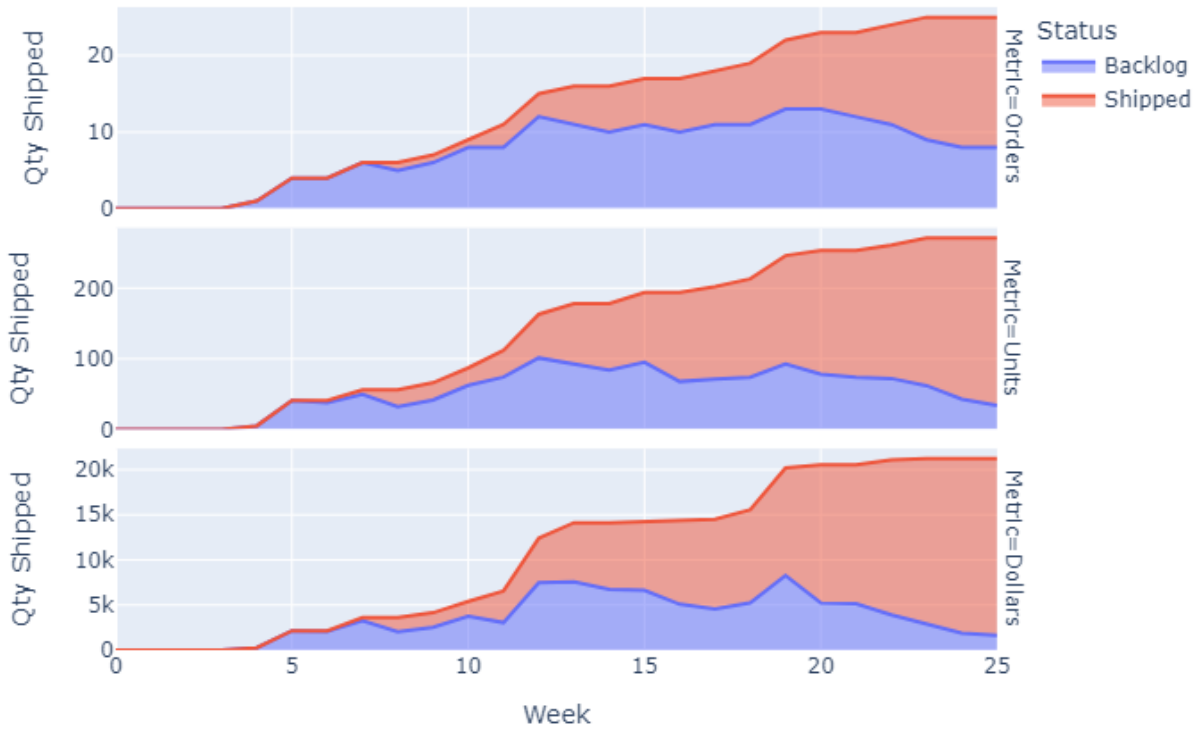


Figure 2.11. Illustrative Example: Stacked area graph displaying the magnitude of the existing overdue backlog of demands vs cumulative shipments delivered during the planning horizon. Each facet row represents a different aggregation of the product (right axis).

2.9 Conclusion

This chapter introduces the Digital Twin framework for Make-to-Order production facilities developed for the purposes of this dissertation. This framework is capable of generating realistic problem instances on the basis of adjustable parameters, creating production plans for those scenarios using optimization models, then simulating and evaluating the execution of those production plans. Our objective in developing this Digital Twin framework was to build the test-bed platform that will enable the development, testing, evaluation and comparison of production planning and scheduling optimization models throughout the remainder of this dissertation.

This proposed DT framework is comprised of 5 unique modules: the Object Definition Module, the Scenario Generation Module, the Planning Module, the Scheduling Module and the Application

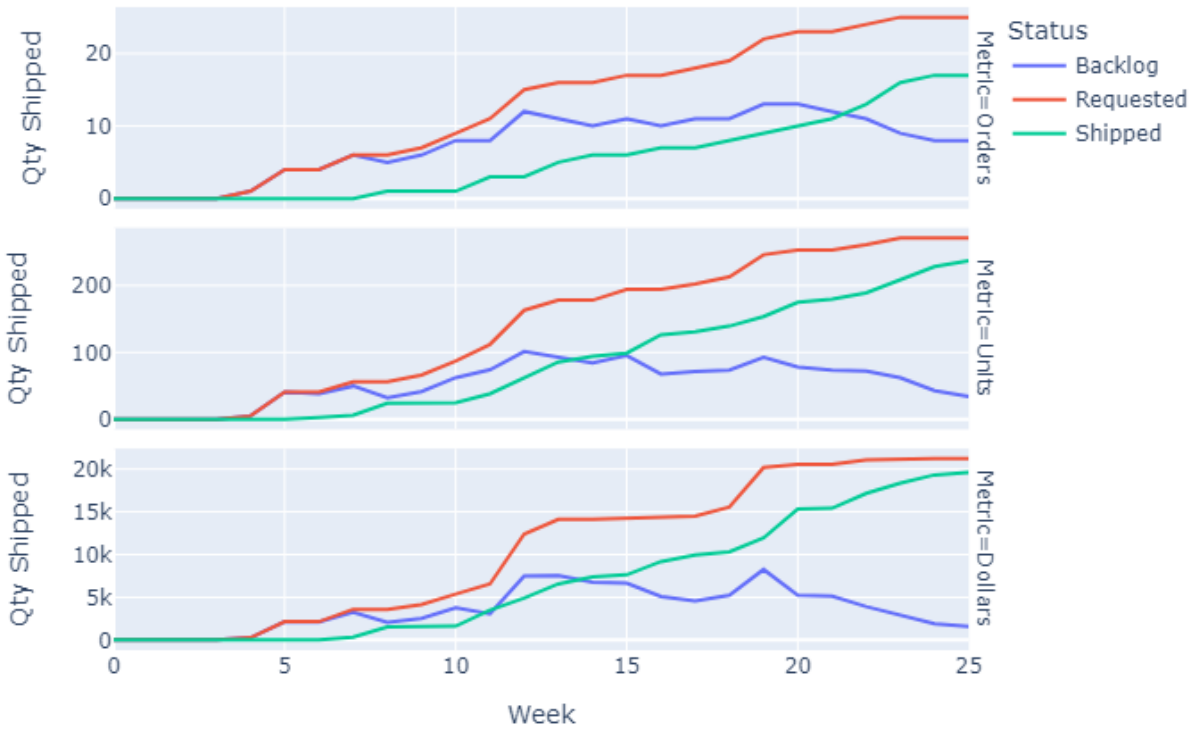


Figure 2.12. Illustrative Example: Line graph displaying the magnitude of the existing overdue backlog of demands vs cumulative requests and shipments delivered during the planning horizon. Each facet row represents a different aggregation of the product (right axis).

Programming Interface Module. We emphasize the *plug-and-play* capabilities of the framework in which modules can be swapped or modified to enable the framework to become applicable to a variety of highly unique problem contexts.

For the purposes of this dissertation, we develop novel algorithms capable of generating large-scale, realistic problem instances for the Multi-Level Capacitated Lot-Sizing Problem (discussed in detail in Chapter 3). We also develop an MRP-based production planning heuristic which will be used as a benchmark for evaluating the optimization models developed in Chapter 3. Further, we develop a discrete-to-continuous-time scheduling heuristic capable of translating a production plan into an executable schedule. This heuristic enables the enhanced evaluation of planning techniques considering the complexities and nuances of manufacturing settings which are too challenging to

incorporate into production planning optimization models. The evaluation of a production plans following it's simulation represents a more realistic evaluation of the plan, i.e. closer to what can be expected in practice when implementing the plan.

To validate the complexity and comprehensiveness of the proposed DT framework, we provide an illustrative example of a randomly generated production facility. We show that the algorithms developed in the Scenario Generation Module are capable of randomly generating realistic and unique Bill of Materials and Bill of Operations while adhering closely to the user-defined parameters which provide context to the desired characteristics of the problem instance. Future research directions for the proposed DT framework include: 1) enhancing the algorithms developed for scenario generation to include more specific considerations per use-case, 2) improving the Continuous-Time Scheduling Heuristic (presented in Section 2.7), and 3) expanding the analytic and visualization packages exhibited in the illustrative example, i.e. to enable the visual comparison of implementation of competing production planning or scheduling practices.

CHAPTER 3

HIERARCHICAL PRODUCTION PLANNING FOR JOB SHOP SCHEDULE OPTIMIZATION

3.1 Introduction

While large corporations have invested heavily in Enterprise Resource Planning (ERP) and Manufacturing Execution Systems (MES) softwares to plan and schedule their production, Small-Medium Enterprises (SMEs) have lagged behind because of its cost, complexity and rigidity once implemented [43]. Our research is motivated by a medium-sized aerospace parts manufacturer struggling to manage its capacity effectively and meet customer due dates. Hierarchical job shop schedule planning and execution is a particularly challenging problem for shops such as this one, facing high product variety, long production lead times and scarce capacity.

At the tactical level, given a set of customer orders and the current status of the shop, gross production of all necessary parts must be planned for, considering the availability of limited resources. Planning must account for a sufficiently large horizon that allows for the completion of these long lead time orders. In cases with severe capacity limitations, some orders may remain unfinished by the end of the planning horizon; the plan must get them as far along as possible. At the operational level, a finer level of detail is required to put the plan, or at least the initial part of the plan, into execution, with a comprehensive schedule that can be directly followed by workers on the shop floor. The resulting plans will be periodically updated to incorporate new orders and to account for changes in status of the shop, which may have deviated as variability, equipment breakdown or malfunction, absenteeism, quality issues, etc., impact production. We refer to the problems at the tactical and operational levels as the Planning Problem and Execution Problem, respectively.

Production plans are presented in the form of work orders which call for the production of a specific quantity (production lot) of a specific item on the shop floor. The item specified as the output of a work order may require multiple operations to transform its input components into

the finished part/assembly. The series of steps (tasks) required to complete the production of the lot is the part's routing. Routings specify: the operations required to produce an item, the work centers/operators/tools required to complete those operations, the costs/times associated with each setup activity required, the processing rates for each unit, and potentially the maximum allowable batch size associated with each step in the routing (route-step).

In most factories, materials within a production lot travel together from operation to operation, typically attached to a document, referred to as a "paper routing". This document includes the details about: the flow of required work that needs to be achieved, where these tasks will be completed, and what materials/components will be required. If a production lot must be split and processed in multiple batches (as a result of limited resource capacity), the first batches to be completed will be held as Work-In-Progress inventory at the work center. Once all batches have been completed, the production lot will move together to the next operation/work center.

A practically desirable property of a production plan is the direct association of internal demands (work orders) to external demands (sales orders) so as to provide transparency and traceability to the production plan and shop floor. The linking of production plans to the sales orders is referred to as *order pegging*. A contribution of this research is dedicated towards the development of an integrated solution procedure which sequentially creates a production plan, then solves the Order-Lot Matching Problem (OLMP), which allocates inventory from a production plan to the specific bill of materials (BOM) requirements of each sales orders. We define a BOM requirement as the satisfaction of a *node* in the assembly-styled network representation of the BOM associated with a sales order. In this study, we refer to the mapping of a production plan to specific order-centric requirements as the Node Pegging Problem.

An important complicating factor in solving the Planning and Execution Problems is the presence of setup times at work centers to account for the necessary tooling changes and preparations required when shifting production from one product to the next. Setups consume precious capacity and need to be avoided by aggregating the production of parts that satisfy multiple orders into production lots. In our application context, setups range from a few minutes to a few days, and so do production times.

The general job shop scheduling problem described above has a common name: the Multi-Level Capacitated Lot-Sizing Problem (MLCLSP). Multi-Level refers to the product structure described

by the bill of materials (BOM), where the final assembly (level 0) is composed of several sub-assemblies (level 1) and each of these in turn is composed of components (level 2), which may be further broken down into other components. Capacitated refers to the limited resources, and lot-sizing is required in the presence of setups. The complexity of the MLCLSP has led researchers to develop many heuristics to provide high-quality solutions for industrial sized problems (see Robinson and Lawrence [143] and Tempelmeier and Buschkühl [32] for actual industrial applications). Exact methods, however, have not yet been explored to their full potential. The development of more powerful computers and the availability of improved commercial optimization software in recent years, together with the use of tight formulations, have led us to consider an exact solution methodology.

In this study, we propose mathematical models and heuristics for the challenging Planning and Execution Problem. We develop computational methods to decrease the size of the problem to allow commercial solvers to find high-quality solutions (within a desired MIP gap) in reasonable computational time. Our objective is to develop a highly generalized production planning and execution decision support system framework that will enable capacity-constrained aerospace manufacturers to generate clear and concise (traceable) schedules. This framework is developed with *practicality* and *configurability* in mind, i.e. we develop many extensions that can *plug-into* the MLCLSP formulation, as needed. These practical considerations are inspired by our collaboration with our industrial partner, and leverage ERP-based datasets as input.

The proposed solution procedure solves two optimization problems sequentially: first the Planning Model (PM) generates a production plan, then the Node Pegging Model (NM) matches the inventories from the production plan to the customer sales orders they satisfy. The PM solves the MLCLSP for multiple items considering a general product structure, multiple classes of resources, and setups. The NM translates the general product structure into an assembly structure to promote solution transparency and traceability when executed on the shop floor. Following the generation of a production plan with the PM and NM optimization models, we apply the Continuous-Time Scheduling Heuristic (see Section 2.7) to convert the big-bucket lot-sizing solution for the MLCLSP into an assignment-based, executable schedule.

In the PM, we define production at the route level, where the capacity consumption of all the required resources required to produce each item are linked to a single decision variable. The timing

of capacity consumption of each resource within the routing is staggered to account for the planned lead times of each route step. These considerations allow for the ability to incorporate existing work orders into the production plan seamlessly, while also ensuring the materials in a production lot travel together throughout the routing, resembling the practice of *paper routing*. We consider a finite planning horizon, positive lead times similar to Helber and Sahling [74], backlogging similar to Wu et al. [190], but we also consider the case of incomplete orders within the time horizon.

The remainder of this paper is organized as follows. In Section 3.2, we briefly review the relevant literature. In Section 3.3 we present a formal statement of the optimization problem. In Section 3.4, we present our mathematical formulations for the Planning and Pegging Models. In Section 3.5, we provide numerical results. In Section 3.6, we present several practical extensions to the formulations developed in this chapter, and in Section 3.7 we conclude on our findings.

3.2 Literature Review

The importance of effective production lot sizes in the presence of fixed costs was first recognized at the turn of the 20th century, with the introduction of the classic Economic Ordering Quantity (EOQ) problem by Harris [73]. A rich literature ensued to tackle the many related problems that arise as assumptions of this basic problem are relaxed. In particular, for the case of dynamic demand over T periods, which is known as the Lot Sizing Problem (LSP), Wagner and Whitin [181] present an exact dynamic programming algorithm that runs in time $O(T^2)$. Linear run time algorithms are now available for this problem; e.g. Wagelmans, van Hoesel and Kolen [180]. The capacitated version of LSP (CLSP) was introduced by Manne [112], to account for the existence of resources with limited capacities. The multi-level version of CLSP (MLCLSP) was introduced by Billington and McClain [25]. This problem considers a BOM structure where components are assembled into sub-assemblies and final products.

Lot sizing problems have been studied extensively and it is well known that a vast majority of the problems are NP-Hard. The complexity of the mathematical models develops for these LSPs increases as more simplifying assumptions are relaxed, resulting in the need for the integration of efficient algorithms and tighter models. We refer the reader to Buschkul et al. [32] for a review of solution approaches found in literature for dynamic capacitated LSPs. In this review, the authors classify solution approaches into five groups: mathematical programming heuristics, Lagrangean

heuristics, decomposition and aggregation heuristics, meta-heuristics and problem specific greedy heuristics. We provide a brief review of these solution approaches in Appendix A.

Kamiri, Ghomi and Wilson [82] provide a relatively recent review on the CLSP and classify models using eight defining characteristics. Some of these modelling dimensions include: definition of product structure (i.e. single-level, serial, parallel, assembly, general), definition of the planning horizon (finite vs infinite) and time periods (small- vs big-bucket), the number of final products in the production system (single vs multiple), resource constraints (uncapacitated vs capacitated), the representation of demand (deterministic vs stochastic), and whether backlogging is allowed or if shortages results in lost sales. The authors also cite the consideration of product lead times and deteriorating items (i.e. food products that will spoil) as additional defining characteristic of lot sizing problems. The production system we consider faces known, deterministic demands, offers multiple non-deteriorating products, is subject to capacitated resources, considers product lead times, allows backlogging and is represented using a finite, big-bucket planning horizon.

In the following, we briefly introduce concepts considering the representation of production changeovers and production lead times as they relate to the literature of the MLCLSP.

Production changeovers between different operations on a resource can incur a setup time/cost. In mathematical models, this is typically represented using a binary variable, taking a value of 1 if a setup activity is performed. The representation of setup activities in mathematical models can be categorized as either simple or complex. A simple setup structure has a time and cost which is independent of the sequence and decisions from previous periods. Many forms of complex setup structures exist, including sequence-dependent, carry-over, and family-based setup structures.

Sequence-dependent setup activities incur a time/cost dependent on the production activity which occurred prior to it, see Mohammadi et al. [116], Ramezanian, Saidi-Mehrabad and Fattahi [140]. Family-based (also referred to as major/minor) setup structures are applicable when similarities in manufacturing processes allow for a *minor* setup activity for successive production runs of items in the same family. Alternatively, a *major* setup activity would be required when there is a changeover of items in different families, see Robinson and Lawrence [143]. Carry-over setup structures describe the case in which a production run from the previous period can continue into the current period. A setup activity would not need to be performed in the current period. Carry-over setups are relevant when considering small-bucket time periods, where the timespan

of a production run spans multiple periods, see Caserta, Ramirez and Voss (2010) [37], Sahling et al. [150], and Stadtler [164]. The application of carry-over setups also applies to big-bucket time periods in the case which the variability of setup times is great enough that some activities' duration exceeds the duration of a big-bucket period, as is the case we consider.

While researchers have tried to consider more realistic classes of MLCLSP, a majority of models still assume no product lead times, allowing predecessors and successors to be produced in the same period [5]. We consider *planned lead times*, which span multiple periods, where the lead time of each product is fixed and does not depend on the production quantity associated with the production lot (also referred to as *timed-route* [21]). We assume these lead times with 100% reliability, which is common practice in the literature (see Bertrand et al. [24], Hopp and Spearman [79]). Unlike Billington et al. [25], who assume that production orders which are released during the start of a period will also be completed during that period (and thus only consume capacity during the period it is assigned to begin), we consider multi-period lead times. In this, the period which capacity is required for each route-step depends on the intra-route lead time of each work order and the period which it will each required resource. This technique is more in line with Spitter et. al. [162], who argue that the benefit of multi-period capacity consumption is in its ability to decouple material release from resource capacity consumption, providing aspects of scheduling problems into the production planning problem. One disadvantage of considering planned lead times is that production flows are not flexible within the production routing [21]. We address this disadvantage in Section 3.6.

The integrated production planning and scheduling framework we consider also addresses the Order-Lot Matching Problem (also referred to as *order pegging*). Most commonly addressed in the context of semiconductor manufacturing, where production routings can contain hundreds of route-steps, the Order-Lot Matching Problem (OLMP) has rarely been addressed in the context of aerospace manufacturing [89]. The solution to OLMP associates production lots with customer orders such that the due dates of each order can be met effectively. This association also provides transparency to the existing schedule and provides planners visibility into the current state of the system with respect to the expected performance of the shop. The main difference between the OLMP and the MLCLSP is that production lot sizes in the OLMP are fixed and do not need to be considered, resulting in an easier problem. Much of the literature considering the OLMP aim

to develop dominance conditions and heuristics for solving the problem (see Kim and Lim [90], Mönch, Shen, and Fowler [117], and Sun et. al. [169]).

Our main contributions to the literature is in the novel modeling techniques developed for solving the Planning and Execution problem. This includes the development of: 1) an integrated solution procedure which sequentially creates a production plan for the MLCLSP, provides order-lot node-pegging associations for all production activities, then generates an assignment-based schedule for the long-term production plan in a practical context, i.e. in a continuous-time, executable schedule, 2) a heterogeneous discretization of the planning horizon which associates each resource group with a unique timeline on the basis of the processes they typically face. We implement this methodology as a means to account for large variabilities in the experienced setup times and processing rates across the production facility. This methodology allows us to avoid the requirement of complex setup structures while also modelling the internal dynamics of the facility at the greatest level of detail, 3) the enforcement of paper routing practices while considering multi-step, cyclical, production routings with intra-route lead-times.

We also formulate additional modeling complexities which, to our knowledge, have not been addressed in the context of the MLCLSP, including: 1) the sequencing of existing work orders, having a fixed production quantity, in the production plan alongside the dynamically generated work orders, having optimized production quantities, 2) the consideration of resource sub-groups, which exhibit unique qualities for other sub-groups in the resource group, 3) the introduction of multiple variations of timed-routes for similar parts to introduce flexibility to the production plan via delays in the planned lead times of parts, 4) the consideration of target safety stock levels. These extensions are presented in Section 3.6.

3.3 Modeling Approach

Parts and the Bill of Materials

Aerospace part manufacturing involves highly complex Bill of Materials (BOM) structures, where the end products are composed of many parts and intermediate sub-assemblies, each requiring multiple operations and many hours of processing. Each unique part/item, i, that a factory may hold or produce is represented in the set $i \in I$. The set K: $K \subseteq I$, is used to represent all parent parts, i.e. parts with children. We define the subset mapping, $K^i \subseteq K$, as the subset of all parent

parts which contain a child part i. Further, we define $\overline{K^i} \subseteq K$ as the set of all *ancestors* of part i, i.e. all of the assemblies which part i is a component of. For each descendant-ancestor pair, we define a Units Per Parent (UPP) parameter, n_{ik}, to represent the quantity of part i required to produce a single unit of part k. Each part, i, is also characterized by a per-unit per-period holding cost, h_i, and a deterministic production lead time, L_i. We define A_{it} as the quantity of part i which is scheduled to arrive as inventory from existing or external sources at the beginning of time period t. For the case of representing initial inventories, $t = 0$. Time periods, t, are represented as big-bucket periods (of duration Δ) in the planning horizon, $t \in T$.

The decision variable u_{it} describes the production lot size of part i which starts production in time period t. The variable v_{it} defines the inventory level of part i at the beginning of time period t.

External and Internal Demands

External demands are represented in the form of customer line items, $o \in O$ (also referred to as *sales orders*). Each sales order, o, is a demand for a specific part, i. The quantity demanded is Q_o. We define D_o as the due date of line item o, P_o as the per-unit per-period penalty for under delivery of overdue sales orders, $\overline{P_o}$ as the per-unit penalty for demands which are unfulfilled orders by the end of the planning horizon, and ϵ_o as the per-period per-unit reward for early deliveries. We define the subset $O^i \subseteq O$ as the set of all sales orders which call for the delivery of part i.

Due to the long lead times observed in industries such as aerospace, the complexity of the MLCSLP and limitations in commercial solver's abilities to handle large-scale problems, it may not be possible to consider a planning horizon long enough to allow for the completion of all existing line items. It is important to incentivize the production of incomplete orders. To accomplish this, we introduce parameters N_i, which represent the cumulative production/procurement requirements of each part, i, to complete all line items, and H_i as the per-unit penalty for the under-production/procurement of part i relative to N_i. We calculate the total production requirements, N_i of each part i as:

$$N_i = \sum_{o \in O} Q_o \, n_{i,I^o}$$

where I^o is the part which is demanded as the end item in order o. Finally, we define the variable r_i to track the *remaining* units of part i, that is, the difference between the production requirements, N_i, and total cumulative production and procurement scheduled/planned to occur during the planning horizon.

The decision variable, y_{ot}, represents the quantity of units delivered for line item o at the beginning of time period t.

Resources and the Bill of Operations

Production within factories requires the use of value-adding resources that transform raw materials into finished goods. We consider the existence of resources and resource groups. A resource group, j, is composed of a collection of similar resources which have similar capabilities. We refer to the set of all resource groups as $j \in J$ and the set of all resources as $e \in E$. The subset of resources which belong to a resource group, j, is $e \in E^j$, and the number of resources in a resource group, j, is $|E^j|$. The subset $I^j \subseteq I$ refers to the set of parts which require capacity from resource group j. The available capacity of each resource group, j, in each time period, t, is C_{jt}. The capacity of each resource group is represented as the aggregation of the capacities of each resource within the group.

The Bill of Operations describes the production routings of all parts that a firm can produce. A part's routing is decomposed into route-steps. Each route-step describes which resources are required to process the operation and what type of setup activity (if any) is required prior to processing. It is possible for the routing of a part to call for several visits to the same resource group. We define the parameters B_{ij} to denote the number of times the routing for part i requires capacity from resource group j. This is required, as the operations which are performed at each visit may be different, and thus requires an identifier, b, to specify which operation is actually occurring. We denote the index, $\gamma = (i, j, b)$, to represent each unique *process* the shop floor is capable of and the set $\gamma \in \Gamma$ as the collection of all processes. We define the collection of all processes which a resource group, j, is capable of performing as $\gamma \in \Gamma^j$, and the collection of processes required to produce a part, i, as $\gamma \in \Gamma^i$. The parameters R_γ and S_γ are the per-unit processing rate and setup time required to run a batch of parts through process γ, respectively. Processes which do

not require a setup will have $S_\gamma = 0$. Finally, we define l_γ to be the intra-route lead-time from the beginning of the production of the first route-step of part i to the time it reaches process γ.

3.3.1 Resource Dependent Timelines

A major contribution of this research is the methodology developed to represent the capacity of each resource group throughout the duration of the planning horizon. Specifically, we discretize the planning horizon of each resource group dynamically, using appropriately sized time buckets that capture the requirements of the processes that particular resource group is responsible for. That is, each resource may use a different time discretization. We consider this methodology as a means to avoid the requirement of carryover setups for the resource groups with long setup and processing times, while capturing the overall factory dynamics at an appropriate granularity. This is a must in factories with wide-ranging processing times.

To accomplish this we define multiple timelines, each with bucket durations twice as large as the prior. Each resource group is assigned to be represented by the timeline which has the shortest bucket duration greater than the maximum amount of time required for any process it is capable of (considering setup time, processing rates, and a maximum batch size).

We define τ as the number of timelines considered in the planning horizon. In this format, each timeline is unique in the time-step increment for each successive period in the timeline. The timeline with the shortest duration bucket is referred to as the *base* timeline (also referred to as Timeline 1). We notate the base timeline as T. The duration of each period in the base timeline, T, is Δ.

Timeline	**Time-step**	**Set of period start times**
Timeline 1 (Base)	10	{0, 10, 20, 30, 40, 50, ..., 350, 360, 375, 380, 390, 400}
Timeline 2	20	{0, 20, 40, 60, 80, 100, ..., 300, 320, 340, 360, 380, 400}
Timeline 3	40	{0, 40, 80, 120, 160, 200, 240, 280, 320, 360, 400}

Table 3.1. Example - Resource Dependent Timelines: Set of period start times, considering 3 unique timelines, with the base timeline having time-step, $\Delta = 10$ hours. This example considers a planning horizon of 400 hours (10 weeks).

<table>
<tr><td>Timeline 1 (Base)</td><td>0</td><td>10</td><td>20</td><td>30</td><td>40</td><td>50</td><td>60</td><td>70</td><td>80</td><td>90</td><td>100</td><td>110</td></tr>
<tr><td>Timeline 2</td><td colspan="2">0</td><td colspan="2">20</td><td colspan="2">40</td><td colspan="2">60</td><td colspan="2">80</td><td colspan="2">100</td></tr>
<tr><td>Timeline 3</td><td colspan="4">0</td><td colspan="4">40</td><td colspan="4">80</td></tr>
</table>

Figure 3.1. Example - Resource Dependent Timelines: Visual representation of 3 unique timelines, with a base timeline having time-step, $\Delta = 10$ hours. Annotations represent the start time of each period.

The representative timeline for resource group j is T^j. T^j is the timeline with the smallest bucket duration which is larger than the maximum process cycle time on that resource, δ_j, where:

$$\delta_j = \max_{\gamma \in \Gamma^j} \left(S_\gamma + M_\gamma R_\gamma \right)$$

M_γ is the maximum production batch size of process γ on any resource, e, within the resource group, j. The parameters Δ_j represents the duration of a time bucket for resource group j. The per-period capacity of resource j assigned to timeline T^j is: $C_{jt} = \Delta_j \ |E^j|$, where $|E^j|$ is the number of resources which belong to resource group, j.

The **integer** decision variables, $z_{\gamma t}$, is the number of setups which occur at resource group j during the bucket starting at time period t for process γ (considering the appropriate timeline, T^j). We assume that a maximum of 1 setup can be performed on each resource, e, per period, t, for a specific process, γ. We define the parameters, Z_γ, to represent the maximum value that the variable $z_{\gamma t}$ can take in any period. Considering an integer decision variable representation of $z_{\gamma t}$, with this assumption, Z_γ will be equal to the number of resources in the resource group associated with that process, $|E^j|$. Note that an alternative modeling approach is to represent $z_{\gamma t}$ as a binary decision variable. We also develop this alternative approach and compare the simulated performance of the planning models using an integer vs binary decision variable representation in the numerical implementation presented in Section 3.5.

We define a mapping for each time bucket in each timeline such that we know which time periods from the base timeline, T, occur in that bucket. This mapping enables the modeling of capacity consumption for multi-step production lots. For example, for the period which begins at

time, $t = 80$, for each of the timelines, the following sets represent the subset of time periods from the base timeline, T, which are found in that bucket:

Timeline	**Period, t**	**Set of corresponding base timeline period start times**
Timeline 1 (Base)	80	{80}
Timeline 2	80	{80, 90}
Timeline 3	80	{80, 90, 100, 110}

Table 3.2. Illustrative Example: Mapping of base timeline buckets which exist within the buckets are timelines with periods of larger duration

We define the variables, $m_{\gamma t}$ as the production lot of process γ during period t. As a reminder, γ, is a shorthand representation of the tuple, (i,j,b), which describes each process, such that a reference to γ inherently refers to the part i and resource group j which the process occurs. We have introduced the decision variable u_{it} as the quantity of part i to begin production during period t. However, the variable u_{it} has no reference to the processes, $\gamma \in \Gamma^i$ which are required to produce the part. We use the intra-route lead-time parameters, l_γ, and the timeline mappings, $s \in T_t^j$ to make the connections between u_{it} and $m_{\gamma t}$.

Example: Capacity Consumption with Resource-Dependent Timelines

Consider an example from the point of view of a resource group, j, which is responsible for the 3rd process, γ, in the routing of a part i. Let a production plan have:

1) A base timeline, T, with periods of duration, $\Delta = 10$,

2) A work order for the production of this part i to begin during time period 100: $u_{i,t=100} > 0$

3) a lead-time offset from the start of the production routing until it reaches process γ be $l_\gamma = 50$,

4) the resource group j be associated with Timeline 3 (with a time-step of 40 hours), such that: $\{0, 40, 80, 120, 160, ...\} \in T^j$.

We know that the production lot will reach process γ during the time period that begins at $t = 150$. However, the resource group, j, does not have a period that begins at time, $t = 150$. In order to account for the capacity requirements of this production lot from the resource group during this period, we use the timeline mapping scheme (see Table 3.2) to define the production plan for process γ at time t as:

$$m_{\gamma t} = \sum_{s \in T_t^j} u_{i,s-l_\gamma} \qquad \forall\, t \in T^j$$

where $i = I^\gamma$, $j = J^\gamma$. For each time period, t, this equation searches *backwards* in the planning horizon, using the smallest base time steps, to identify any production lots which will visit the resource group during period t.

Consider this equation with a value $t = 120$: the subset, $s \in T_t^j$ will include the base timeline periods, $\{120, 130, 140, 150\}$. For each value of s, we look at the value of the production decision variable, u_{it} from time $s - l_\gamma$ (in this case, $l_\gamma = 50$). We see that the set $s \in T_t^j$ contains $s = 150$, such that $s - l_\gamma = 150 - 50 = 100$. Therefore, we know that the production lot associated with $u_{i,t=100}$ will reach the resource group j that is associated with process γ during time period $t = 120$, on the timeline, T^j = Timeline 3. From this know that $m_{\gamma,t=120} = u_{i,t=100}$.

In the formulation described in Section 3.4, we define constraints which leverage this equation to identify all of the production lots which visit a resource group, j, in any period, t. The capacity consumption of this production lot can then be accounted for on the resource group associated with this process accordingly.

3.4 Mathematical Formulations

In the following, we present the integrated solution procedure developed to solve the production Planning and Execution Problem. The objective of the 3-phase proposed procedure is to provide a detailed, executable schedule for an extended planning horizon. The first optimization model we implement, the Planning Model (PM), generates a high-level production plan, then the Node Pegging Model (NM) associates production decisions to the external demands which they satisfy, then the Continuous-Time Scheduling Heuristic (CTSH, described in Section 2.7), translates the discrete-time production plan into a continuous-time executable schedule.

3.4.1 Planning Model

The PM addresses the following problem. A Make-To-Order firm must create a production plan to satisfy the sales orders, $o \in O$, in their demand backlog. The scheduling objective is to minimize the total penalty faced throughout the planning horizon. Note that the objective function

does not represent an actual cost, but rather a score for the production plan given the scheduler's preferences.

We introduce several new user-defined parameters which restrict the solution space of the production plan. To accommodate practical business requirements, we define the parameter, $\overline{\epsilon}$, as the amount of time before its due date that an order can be delivered. We allow and even incentive early delivery within this time window. We also define the parameter μ to restrict the minimum production lot size of any work order. This parameter is represented as a proportion, i.e. 0.10, of the maximum production lot size, M_γ. We introduce this parameter to ensure the work orders generated by the Planning Model are of acceptable size in practice. We discuss, evaluate, and justify the use of these parameters in Appendix B.

The required notations for the Planning Model is presented in Table 3.3.

Sets	Description
$i \in I$	Set of all parts
$o \in O$	Set of sales orders
$j \in J$	Set of constrained resource groups
$\gamma \in \Gamma$	Set of all processes
$\gamma \in \Gamma^j$	Set of processes that are processed by resource group j
$i \in I^j$	Set of all parts that require constrained resource j
$k \in K^i$	Set of parts which are direct parents to part i
$k \in \overline{K^i}$	Set of all parts which are ancestors to part i
$o \in O^i$	Set of orders that call for part i as final deliverable
$T \in \tau$	Set of all timelines used in the planning horizon
$t \in T$	Set of periods in the base timeline
$t \in T^j$	Set of periods in timeline specific to resource j
$s \in T_t^j$	Set of base time periods in time bucket, t, in the timeline associated with resource j
Parameters	**Description**
$\overline{T}$	Number of periods in the planning horizon
Δ	Duration of each period in the base timeline, T
$\overline{\epsilon}$	Amount of time prior to a line item's due date, D_o, a shipment can be delivered
Q_o	Quantity demanded in order o
D_o	Due Date of order o
$\underline{P_o}$	Per-period per-unit tardiness penalty for order o
$\overline{P_o}$	Per-unit penalty for not fulfilling order o during the planning horizon
ϵ_o	Per-period per-unit incentive for completing order o early
h_i	Per-unit per-period holding cost of part i
H_i	Per-unit penalty for under-production of part i
N_i	Units of part i needed to complete all orders
L_i	Lead time of part i
n_{ik}	Units of part i needed to produce a unit of part k
A_{it}	Scheduled receipt of part i in initial period t (includes initial inventory, i.e. $t = 0$)
C_{jt}	Capacity of constrained resource group j in period t
l_γ	Lead time offset for part i to begin process γ
R_γ	Per-unit processing rate of process γ
S_γ	Setup time of process γ
Z_γ	Maximum number of setups allowed per period for process γ
$M_{\gamma t}$	Maximum production output, per setup, of process γ in period t
Variables	**Description**
$m_{\gamma t}$	Production lot size of process γ in period t
r_i	Units of part i required but not completed by the end of the horizon
u_{it}	Production lot size of part i starting production at time t ; completed at time $t + L_i$
v_{it}	Inventory level of part i at time t
y_{ot}	Quantity of end item for order o shipped in period t
$z_{\gamma t}$	Number of setup activities for process γ in period t

Table 3.3. Set, parameter and variable notation for the Planning Model

Formulation **(PM)**

$$\min \quad \sum_{\substack{o\in O:\\ D_o\leq \overline{T}}} \bar{P}_o\big(Q_o - \sum_{t\in T} y_{ot}\big) \;+\; \sum_{o\in O}\sum_{\substack{t\in T:\\ t>D_o}} P_o\big(t-D_o\big)y_{ot}$$

$$- \sum_{o\in O}\sum_{\substack{t\in T:\\ t<D_o}} \epsilon_o\big(D_o - t\big)y_{ot} \;+\; \sum_{i\in I} H_i r_i \;+\; \sum_{i\in I}\sum_{t\in T} h_i v_{it}$$

s.t.

$$\sum_{t\in T} y_{ot} \;\leq\; Q_o \quad \forall \quad o\in O \tag{P1}$$

$$\sum_{\substack{t\in T:\\ t<D_o-\bar{\epsilon}}} y_{ot} \;=\; 0 \quad \forall \quad o\in O \tag{P2}$$

$$\sum_{t\in T} u_{it} \;\leq\; N_i \quad \forall \quad i\in I \tag{P3}$$

$$N_i - \sum_{t\in T}\Big(u_{it} + A_{it} + \sum_{k\in\overline{K^i}} n_{ik}A_{kt}\Big) \;\leq\; r_i \quad \forall \quad i\in I \tag{P4}$$

$$\sum_{s\in T_t^j} u_{i,s-l_\gamma} \;=\; m_{\gamma t} \quad \forall \quad j\in J, \gamma\in\Gamma^j, t\in T^j : i = I^\gamma \tag{C1}$$

$$M_{\gamma t}\, z_{\gamma t} \;\geq\; m_{\gamma t} \quad \forall \quad j\in J, \gamma\in\Gamma^j, t\in T^j \tag{C2}$$

$$\mu\, M_{\gamma t}\, z_{\gamma t} \;\leq\; m_{\gamma t} \quad \forall \quad j\in J, \gamma\in\Gamma^j, t\in T^j \tag{C3}$$

$$z_{\gamma t} \;\leq\; Z_{\gamma t} \quad \forall \quad \gamma\in\Gamma, t\in T \tag{C4}$$

$$\sum_{\gamma\in\Gamma^j}\Big(R_\gamma m_{\gamma t} + S_\gamma z_{\gamma t}\Big) \;\leq\; C_{jt} \quad \forall \quad j\in J, t\in T^j \tag{C5}$$

$$v_{i,t-\Delta} + A_{it} + u_{i,t-L_i} - \sum_{k\in K^i} n_{ik}\, u_{kt} - \sum_{o\in O^i} y_{ot} \;=\; v_{it} \quad \forall \quad i\in I, t\in T : t > L_i \tag{I1}$$

$$v_{i,t-\Delta} + A_{it} - \sum_{k\in K^i} n_{ik}\, u_{kt} - \sum_{o\in O^i} y_{ot} \;=\; v_{it} \quad \forall \quad i\in I, t\in T : t \leq L_i \tag{I2}$$

$$m, r, u, v, y \;\geq\; 0 \tag{NN}$$

$$z \;\in\; \mathbb{Z} \tag{INT}$$

The objective function minimizes the total penalty experienced throughout the planning horizon. This penalty is comprised of 5 scheduling objectives. (Obj1) calculates the penalties for the unsatisfied orders at the end of the planning period. (Obj2) calculates penalties for orders that are delivered past their deadline. (Obj3) calculates a reward for delivering orders earlier than their deadline within the delivery window. (Obj4) associates a penalty for incomplete production

requirements, which incentivizes the partial production of orders which the plan is not able to complete within the planning horizon. (Obj5) describes holding costs for inventory carried from period to period.

Constraints (P1) prevent over-delivery. Constraints (P2) prevent deliveries for any order, o, earlier than the user-defined delivery window. Constraints (P3) prevent over production. Constraints (P4) calculate the unsatisfied requirements of each part, i. Note that these constraints consider 3 methods for satisfying the production requirements of any part, i. The first is from the cumulative production of part i, in u_{it}. The second is the cumulative allocation from scheduled receipts or initial inventory of part i, in A_{it}. We refer to this as a *direct* allocation of the part, i. The third method considers *indirect* allocations of the part, i. This accounts for cases in which an assembly part, k, is received in the form of a scheduled receipt or existing inventory. Because of the existence of these ancestor parts, the production requirements of part i need to be adjusted accordingly.

Constraints (C1) calculate the total production of process γ in time period t. Constraints (C2) restrict the maximum output of each process γ. Constraints (C3) restrict the minimum output of each process γ. Constraints (C4) restrict the number of setups for process γ are performed. Note that Constraints (C2) and (C3) are only enforced when a setup activity is performed, i.e. $z_{\gamma t} > 0$. Constraints (C5) ensure capacity consumption does not exceed available capacity.

Constraints (I1) and (I2) are inventory balance constraints. Constraints (I1) describe the inventory level for time periods which are greater than the respective part i lead time, L_i. In these periods, the inventory v_{it} is found as the inventory level from the previous period, plus any scheduled receipts of that part, plus the completed production of that part (i.e. from production started L_i periods ago), minus the units needed to start production of any parent assemblies in that period, minus any units which are delivered to the end customer. For time periods which are less than the lead time of the respective parts, the inventory balance Constraints (I2) remain the same with the exception that no new production for the part can be completed, only scheduled receipts.

3.4.2 Node Pegging Model

The PM uses a general product structure to represent the Bill of Materials. A graphical representation of a general product structure displays each unique item, regardless of the finished good it is a component of, as a node in the graph and each unique parent-child relationship as an

edge. In this network, each part (node) can have multiple parent nodes. While this representation reduces the number of nodes required to represent the BOMs of the items a factory may produce, it provides little transparency as to which component/part in the production plan will satisfy each specific sales order. This information enables planners and schedulers to identify sales orders affected by delays in production as well as provide up-to-date lead time information based on the current state of production.

In Phase 2 of the solution procedure, the solution from the PM is used as input to the Node Pegging Model (NM). The purpose of NM model is to take the production plan and associate each unit of inventory to a specific production requirement of a sales order. This is accomplished by translating the general product structure representation of the BOMs into their assembly product structure representation. An assembly product structure representation captures each *instance* of a part in the BOM associated with each unique sales order as a node in the network. For example, if a part is a component of two different sub-assemblies in the finished good of a sales order, it will be represented in the network as two separate nodes, whereas in the general representation it would be displayed as just one. As a consequence, each *node* in the assembly network will have exactly one parent node (except finished goods which have none). Considering this product structure, the objective of the NM is to optimally allocate inventory to specific nodes such that each sales order can be delivered **exactly** as described in the PM solution (by the delivery variables, y_{ot}).

We define the set of all nodes that exist in the demand backlog as $f \in F$. A node represents a production requirement in the assembly structure of the finished good for a **specific** sales order. For example, even if two sales orders call for the same finished good, i, the production requirements for each sales order will be represented as multiple nodes in the NM. The set $f \in F^i : F^i \subseteq F$ defines all of the nodes which are representative of part i. An illustrative example of this property is shown in Figure 3.2.

In the NM, we **fix** the optimized decision variables from the solution from the PM, treating them as parameters. We introduce new notation for the *parameterized* decisions: U_{it} for production decisions, and V_{it} for inventory levels. The objective of the NM is to allocate the units from each of these parameterized decisions to the specific production requirement to the specific production requirements associated with each node $f \in F$ to satisfy each sales order. New variables, indexed in f, are defined for allocating production, u_{ft}, scheduled receipts, a_{ft}, and inventory levels v_{ft}.

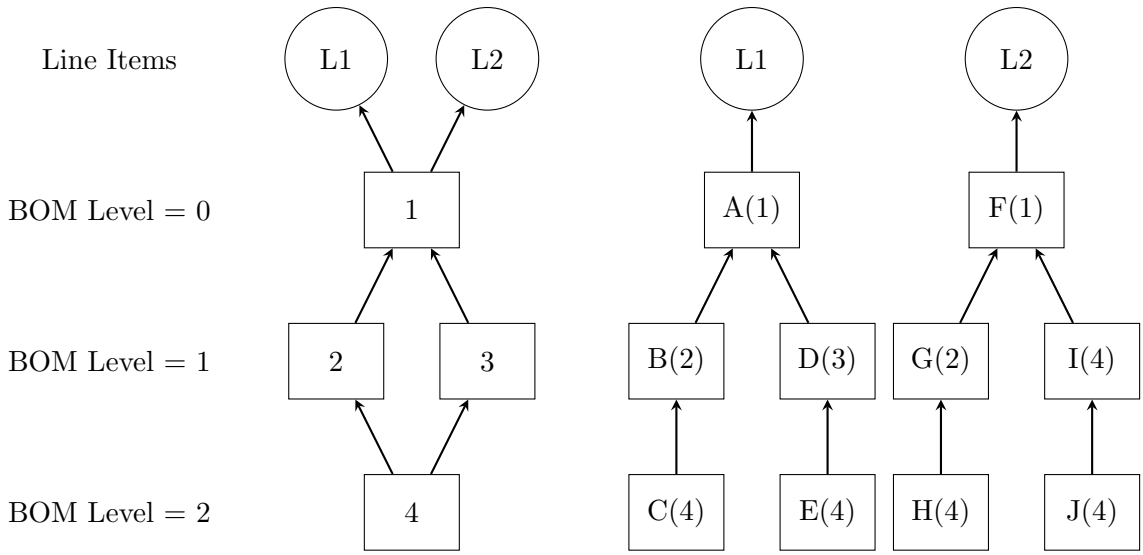


Figure 3.2. Example: General product structure (left) and Assembly product structures (right) representations of one end item (Part = 1) which satisfies two line items (Order 1 and Order 2). Circles represent line items, squares represent parts and nodes. Numerical values represent part numbers. Alphabetic values represent node ID. Note that network nodes are duplicated for each part that is found in multiple line items AND multiple assemblies in each line item, i.e. Part 4 is represented with Node C, Node E, Node H and Node J.

We define the one-to-one mapping $o \leftarrow O^f$ to specify the sales order, o, that node f satisfies, and the binary parameter Y_f to take a value of 1 if node f is the end item deliverable for the sales order O^f. The parameter N_f specifies the cumulative production requirement allocated to node f to complete the sales order it is associated with, and the variable r_f tracks the difference between the production requirements and total allocation dedicated to that node. The parameter H_f specifies the per-unit penalty of under production/production for node, f.

We also define a new set of **slack** variables that track the parts (in the system as given by the PM) that are not assigned to any sales orders. We denote this using a $*$ index over the planning model solution parameters, i.e. u^*_{it}. This inventory can be thought of as the safety stock on the shop floor.

	Notation	Description
Sets	$f \in F$	Set of all nodes
	$k \in K^f$	Single element set specifying the parent node to node f
	$k \in \overline{K^f}$	Set of all nodes which are ancestors to node f
	$f \in F^i$	Set of nodes which represent part i
	$o \in O^f$	Single element set specifying the sales order that node f satisfies
Parameters	H_f	penalty per unit of node f required but not completed by end of horizon
	L_f	lead time of node f
	N_f	Units of node f needed to complete the sales order it is associated with
	n_{fk}	Units of node f needed to build a unit of node k
	U_{it}	Parameterized lot production of part i from time period t (from PM)
	V_{it}	Parameterized inventory level of part i from time period t (from PM)
	Y_f	1 if node f is the end item for its associated sales order, otherwise 0
Variables	r_f	units of node f required but not completed by the end of the horizon
	a_{ft}	Allocation of scheduled receipts from time period t to node f
	u_{ft}	Allocation of lot production from time period t to node f
	v_{ft}	Allocation of inventory from time period t to node f
	a^*_{it}	Unallocated scheduled receipts of part i from time period t
	u^*_{it}	Unallocated lot production of part i from time period t
	v^*_{it}	Unallocated inventory of part i from time period t

Table 3.4. New and modified notation for the Node Pegging Model

Formulation **(NM)**

max
$$\sum_{f \in F} \Big(\sum_{t \in T} v_{ft} - H_f r_f \Big)$$

s.t.

$$\sum_{t \in T} \Big(u_{ft} + a_{ft} + \sum_{k \in \overline{K^f}} n_{fk} a_{kt} \Big) \leq N_f \quad \forall \quad f \in F \tag{1}$$

$$N_f - \sum_{t \in T} \Big(u_{ft} + a_{ft} + \sum_{k \in \overline{K^f}} n_{fk} a_{kt} \Big) \leq r_f \quad \forall \quad f \in F \tag{2}$$

$$a^*_{it} + \sum_{f \in F^i} a_{ft} = A_{it} \quad \forall \quad i \in I, t \in T \tag{3}$$

$$u^*_{it} + \sum_{f \in F^i} u_{ft} = U_{it} \quad \forall \quad i \in I, t \in T \tag{4}$$

$$v^*_{it} + \sum_{f \in F^i} v_{ft} = V_{it} \quad \forall \quad i \in I, t \in T \tag{5}$$

$$v_{f,t-1} + a_{ft} + u_{f,t-L_f} - \sum_{k \in K^f} n_{fk}\, u_{kt} - Y_f y_{ot} = v_{ft} \quad \forall \quad f \in F, o \in O^f, t \in T : t > L_f \tag{6}$$

$$v_{f,t-1} + a_{ft} - \sum_{k \in K^f} n_{fk}\, u_{kt} - Y_f y_{ot} = v_{ft} \quad \forall \quad f \in F, o \in O^f, t \in T : t \leq L_f \tag{7}$$

$$a, r, u, v \geq 0 \tag{NN}$$

The objective function of the NM is comprised of two components. The first component of the objective maximizes the total allocated inventory during the planning horizon to incentivize the allocation of inventory to sales orders as early as possible. The second objective component minimizes the total weighted under production penalty. This ensures the allocation of production requirements to the most important sales orders which are not delivered during the planning horizon.

Constraints 1 limit the allocation of units to any node to be less than the total number of units required to complete the production for the entire order. Constraints 2 calculate the under allocation of inventory to each node, relative to the cumulative allocation requirements. These constraints consider both direct and indirect allocations of scheduled receipts. Constraints 3-6 restrict the allocation of units across all nodes to be exactly equal to the available units of each part i, defined from the solution of the Planning Model. Constraints 3 determine the slack (which is unallocated inventory) remaining after the allocation of scheduled receipts. Constraints 4 determine the slack remaining after the allocation of scheduled production. Constraints 5 allocate the existing inventory to each node. Finally, Constraints 6-7 define the inventory dynamics, similar to how they are defined in the PM. Note that the parameterized delivery decisions, y_{ot}, from the solution of the Planning Model are the *driving force* that ensures the allocation of inventory across nodes throughout the planning horizon matches the PM solution.

3.4.3 Continuous-Time Scheduling Heuristic

The solutions of the PM and NM are used as input for Phase 3 of the solution procedure, the Continuous-Time Scheduling Heuristic (CTSH, see Section 2.7). The purpose of the CTSH is to transform the big-bucket lot-sizing production plan from the PM/NM solution into a continuous-time, assignment-based solution with enough detail to be executed on a shop floor and to evaluate the true performance of the proposed plan in practice. The CTSH algorithm has been developed as a myopic, forward-constructed scheduling heuristic that iteratively schedules all of the activities in the planning horizon.

3.5 Numerical Implementation

3.5.1 Design of Experiments

In this section, we evaluate the production planning framework developed in this chapter, when integrated with the Digital Twin framework presented in Chapter 2. Unless otherwise specified, all generated problem instances consider a factory which is comprised of 50 unique parts, 8 resource groups and 25 customer orders. Products are represented in a general product structure. Cyclical and multi-step routings are allowed. The production facility is structured into 3 production phases, to differentiate the characteristics of early, middle and final stage processes. Each with a target utilization of 0.80. Each part belongs to one of four part classes and each resource group belongs to one of six resource group classes (see Table 3.6 and 3.7). The Bill of Materials of any part can have a maximum of 4 offset levels, and each part can have up to 5 children. The numer of units per parent are in the range [1, 5] and the routing of each part can have up to 5 route steps. Each resource group is made up of up to 3 resources. Each customer order represents a demand for a single part (belonging to part class `FinishedGood`) for a quantity in the range [1,20]. The planning horizon spans 26 weeks, with 40 hours in a week. Three timelines are considered in the planning horizon with time steps of 10 hours, 20 hours and 40 hours, respectively.

Parameter	Notation	Default Value	Description
`prodStr`		General	Specification of Product Structure scheme
`resAssign`		Cyclical	Specification of Resource Assignment scheme
`routeComp`		Multi-Step	Specification of Routing Complexity scheme
`utilProf`		[80, 80, 80]	Specification of Utilization Profile scheme
`numI`	$\|I\|$	50	Number of unique items
`numJ`	$\|J\|$	8	Number of unique resource groups
`maxT`	$\|T\|$	26	Length of the planning horizon (weeks)
`numLI`	$\|O\|$	25	number of external demand orders
`numSR`	$\|A\|$	0	number of scheduled receipts
`maxRperRG`	$\|\overline{E}\|$	3	max number of resources per resource group
`maxLiQty`	$\overline{Q}$	20	Max qty. associated with any line item
`maxLev`	$\overline{\lambda}$	4	Max level in a BOM
`maxUPP`	$\overline{n}$	5	Max units-per-parent of any parent-child relationship

Table 3.5. User-defined parameters, notation, and values - `Factory` parameter model

In order to evaluate the solution quality and provide an analysis of the proposed algorithms, we used test instances generated using the algorithms introduced in Section 2.5. Each test instance,

`className`	Raw Material	MachinedPart	AssembledPart	FinishedGood
`classDensity`	0.1	0.3	0.3	0.3
[`minChild`, `maxChild`]	[0, 0]	[1, 3]	[1, 5]	[1, 5]
[`minParent`, `maxParent`]	[1, 10]	[1, 5]	[1, 3]	[0, 0]
[`minLevel`, `maxLevel`]	[3, 3]	[1, 2]	[1, 2]	[0, 0]
[`minRS`, `maxRS`]	[0, 0]	[1, 3]	[1, 3]	[1, 3]

Table 3.6. User-defined parameters, notation, and values - `PartClass` parameter model

`className`	RG1	RG2	RG3	RG4	RG5	RG6
`classDensity`	0.16	0.16	0.16	0.16	0.16	0.16
[`minLevel`, `maxLevel`]	[0, 3]	[0, 3]	[0, 1]	[0, 1]	[2, 3]	[2, 3]
`timeline`	10	10	20	20	40	40
`probSetup`	0.50	0.50	0.25	0.75	0.25	0.75
[`minS`, `maxS`]	[0, 0.5]	[0, 0.5]	[0, 0.5]	[0, 0.5]	[0, 0.5]	[0, 0.5]

Table 3.7. User-defined parameters, notation, and values - `ResourceGroupClass` parameter model

referred to as `base_factory`, is reproducible through the use of pseudo-random number generators. Each instance can also be copied then updated such that several *versions* of a `base_factory` can be evaluated and compared. For example, a `base_factory` can be updated for various coefficients weightings which influence coefficients such as the per-period holding cost of parts or per-period tardiness penalty of orders. The process of copying then updating a `base_factory` enables comparable results between different variations of each test instance. Once a `base_factory` has been updated, now named `factory`, the solution procedure of preparing parameters, solving the production Planning Model, the Node Pegging Model, the CTSH simulation and post-processing analysis' is followed in sequence. This process is summarized in Algorithm 8.

A total of 10 `base_factory` environments are generated and updated throughout the analysis' provided in each section. We limit runtimes of each problem to 3600 seconds, track the runtime required to reach significant MIP Gap milestones, and use an MIPGap termination condition of 1% to terminate near-optimal solutions. All computational experiments are performed using Gurobi v9.5.0 solver, on machines with 32 GB RAM and 8 cores.

In the following, we evaluate the proposed Planning and Pegging formulations and overall solution procedure against several benchmark solution methodologies. In Section 3.5.2, we evaluate the proposed solution procedure compared to the MRP-based heuristic presented in Section 2.6. In

Algorithm 8: Procedure for parameter calibration Design of Experiments

Input: `num_env` : Number of `base_factory` environments to run
Input: `grid` : Parameter grid with all variations of each `base_factory` to evaluate
Input: `UDP` : Collection of all required User-Defined Parameters

```
Initialize result

for env in range(num_env) do
    Let base_factory be the output of generator ← UDP

    for (trial_name, trial) in grid do
        Let key be the tuple (env, trial_name)
        Let factory be the output of update_factory ← (base_factory, trial)
        Let p be the output of preprocess ← factory
        Let pm be the output of the Planning Model ← (factory, p)
        Let nm be the output of the Pegging Model ← (factory, p, pm)
        Let sim be the output of the CTSH simulation ← (factory, pm, nm)
        Let kpi be the output of KPI_pipeline ← (sim, pm)
        Save key:kpi → result
```

Output: `result`

Section 3.5.3, we evaluate the implementation of the novel Resource-Dependent-Timeline methodology (see Section 3.3.1) versus an implementation of the Planning Model which relies on a single timeline. Finally, in Section 3.5.4, we evaluate the impact which the representation of the decision variable for setup activities has on the solutions generated by the Planning Model. We consider the use of integer, binary and continuous decision variables.

We refer the reader to Appendix B for an in-depth discussion and sensitivity analysis regarding the calibration of the user-defined parameters associated with the proposed solution procedure. These user-defined parameters are associated with: 1) the derivation of the cost coefficients in the objective function of the PM and NM and 2) the constraints and rules used in the PM and CTSH to reflect various common manufacturing practices.

3.5.2 MRP Heuristic Benchmarking

Table 3.8 provides an analysis on the performance of the proposed solution procedure and Planning/Pegging Models against the MRP-based production planning heuristic. Values shown in this table represent the *average* value of each metric across the 10 **shared** `base_factory` test instances. For example, values shown for the *25th pc.* of *Lateness* are calculated as the mean of

the 25th percentile of lateness of shipments from each test instance. The index, *Measure*, indicates the whether the delivery KPIs consider the delivery of the delivered *Units* or *Dollars*. The index, *Method*, indicates what solution method each row represents. PM shows the results from the production plan generated by the Planning Model, and MRP shows the results from the MRP-heuristic (see Section 2.6.1). The index *Source* indicates whether the delivery performance is given directly by the production plan (*Planned*) or calculated as output from the CTSH heuristic (*Actual)*. All values shown evaluate the delivery performances based on the weighted value of each unit in the demand backlog.

The multi-column, *% of Demanded*, represent the percentage of demands that were delivered during the planning horizon (*Delivered*) and delivered on time (*On Time*). Multi-column *Lateness (Hours)* describes the distribution of the lateness for shipments made during the planning horizon. Shipments which were delivered earlier than the associated order's due date will have a negative value for lateness.

As a reminder, the MRP-benchmark does not consider capacity and as a result is capable of completing and delivering all demands to the customer during the planning horizon in the production plan, while the Planning Model is only able to deliver, on average, 87.6% of demands (in *Dollars*) during the planning horizon. However, the Planning Model greatly outperforms the MRP heuristic when considering the continuous-time execution of the production plans. While the simulated execution of the production plan generated by the PM/NM is able to deliver on 84.2% of demands (in *Dollars*), the MRP plan is only able to deliver 27.0%.

Note that the MRP heuristic has much better delivery performance when considering the *% of Delivered Units*. This shows that while the simulated schedule of the MRP-based production plan is able to deliver 62% of units, that the units that were able to be shipped were a low-value, i.e. had less complex BOM structures, depending on the successful completion of less scheduling tasks. Further, the *Lateness* of deliveries in the simulated plan of the PM/NM outperforms the MRP heuristic.

Table 3.8. Comparison of proposed solution procedure vs MRP-based heuristic: Analysis of planned delivery performance based on each model and its corresponding assumptions, and actual delivery performance calculated via simulation using the CTSH

Measure	Source	method	% of Demanded Delivered	% of Demanded On Time	Lateness (Hours) Mean	Lateness (Hours) 25th pc.	Lateness (Hours) Median	Lateness (Hours) 75th pc.
Dollars	Planned	MRP	100.0 %	88.0 %	-29.8	-40.0	-40.0	-34.9
		PM	87.6 %	42.8 %	47.2	-44.0	-5.0	104.0
	Actual	MRP	27.0 %	5.4 %	263.3	56.0	282.0	439.0
		PM	84.2 %	21.8 %	124.6	11.0	94.0	223.0
Units	Planned	MRP	100.0 %	91.3 %	-31.4	-40.0	-40.0	-33.1
		PM	86.8 %	48.0 %	59.1	-57.0	-9.5	133.0
	Actual	MRP	62.9 %	21.3 %	163.8	0.0	130.5	298.0
		PM	83.2 %	33.0 %	100.2	-36.0	57.0	188.0

Table 3.9 presents the realized penalties in the production plan solution (*Plan*), simulated schedule (*Actual*), and the difference between the two (*Difference*). Values are shown as a percentage of the **Total Cost of Goods Sold**, i.e. the cumulative *value* of all demands in the planning horizon. The total cost of goods sold is calculated as: $\sum_{o \in O} Q_o V_{i \leftarrow I^o}$. As shown in Appendix B, the value of each part, V_i, is also used to derive the objective function coefficients in the PM.

Holding Cost shows the cost experienced for holding non-WIP inventory, *Under Production* is the cost for shortages in the production of items relative to the internal production requirements necessary to satisfy all external demands, *Tardiness Penalty* is the total penalty for lateness of all deliveries made during the planning horizon, and *Unfulfillment Penalty* is the total penalty experienced for units that were demanded during the planning horizon but not delivered. As shown, the MRP production plan greatly under-estimates the costs which are experienced in the simulated schedule. The MRP generated schedule also resulted in a *Total Penalty* more than twice as large as that of the schedule proposed by the PM/NM procedure.

Table 3.9. Comparison of proposed solution procedure vs MRP-based heuristic: Analysis of costs and penalties in the production plan and simulated schedule

Source	KPI method	Holding Cost	Under Production	Tardiness Penalty	Unfulfillment Penalty	Total Penalty
Plan	MRP	0.00 %	0.00 %	0.64 %	0.00 %	0.64 %
	PM	0.71 %	2.00 %	6.01 %	5.51 %	14.23 %
Actual	MRP	2.61 %	8.55 %	5.51 %	28.15 %	44.82 %
	PM	3.47 %	3.34 %	7.74 %	7.07 %	21.63 %
Difference	MRP	2.61 %	8.55 %	4.87 %	28.15 %	44.18 %
	PM	2.76 %	1.34 %	1.73 %	1.56 %	7.40 %

Table 3.10 provides a comparison of the *Planned* and *Actual* utilization rates across the 10 test scenarios for both the MRP-benchmark and proposed framework. The column, *Process* shows the percentage of available resource hours which are spent in the processing phase of an operation, and *Setup* shows the time spent in the setup phase of the planned tasks; *Total* is the sum of the two. It should be noted that the *Planned* utilization of the Planning Model much better estimates the *Actual* realized utilization, showing a 3.0% error, compared to a 10.4% error in the MRP plan. While the utilization rates resulting from both the MRP and PM are comparable in the simulated schedule, the proportion of time spent in *processing* in the simulated schedule is greater with the PM plan than the MRP plan. This is due to a larger proportion of planned production lots from the MRP plan being delayed beyond the limits of the planning horizon.

Table 3.10. Comparison of proposed solution procedure vs MRP-based heuristic: Analysis of resource utilization in production plan and simulated schedule

method	Planned Process	Planned Setup	Planned Total	Actual Process	Actual Setup	Actual Total
MRP	68.23 %	14.21 %	82.44 %	57.99 %	14.05 %	72.04 %
PM	65.46 %	13.38 %	77.81 %	60.55 %	14.26 %	74.81 %

Table 3.11 provides a summary into the number and size (quantity) of each **planned** delivery shipment and production work order. The column, *Count* is the number of unique shipments/work orders in the production plan. *Mean* and *Median* describe the quantity of units which are delivered in each shipment or produced in each work order, and *Total* is the total quantity of units shipped/produced in the production plan. All values shown are representative of the means across

all 10 tested scenarios. The MRP production plan generates a single shipment for each line item. The MRP plan also results in about 300 generated work orders compared to 423 from the PM plan. Also note that the MRP production plan has a greater number of units shipped and produced (column *Total*) compared to the PM production plan. This is a result of the MRP-based production plan incorrectly estimating what will be able to be produced during the planning horizon.

Table 3.11. Comparison of proposed solution procedure vs MRP-based heuristic: Analysis of generated shipments and work orders in the production plan

	Delivery Shipment Qty				Work Order Qty			
method	Count	Mean	Median	Total	Count	Mean	Median	Total
MRP	25.00	9.96	9.70	249.10	298.10	32.71	17.77	9750.00
PM	72.10	3.21	2.30	231.35	422.90	22.53	9.72	9526.90

Table 3.12 provides context to the **delays** which are experienced for both delivery *Shipments* and *Work Orders* in the simulated schedule, compared to the generated production plan. For both *Shipments* and *Work Orders*, *Count* describes the number of planned work orders (measured as an average across the 10 test scenarios) which are completed during the simulated planning horizon, *Mean* and *Median* are the mean and median lateness of task start times in the simulation relative to their **planned** start time. As shown, the proposed solution framework also greatly outperforms the MRP heuristic in generating a production plan that can be achieved in the CTSH. Note that the *Mean* and *Median* delays experienced for the *Work Orders* and *Shipments* throughout the planning horizon are much greater for the MRP plan than the PM/NM plan.

Table 3.12. Comparison of proposed solution procedure vs MRP-based heuristic: Analysis of delays in the simulated execution of work orders relative to the scheduled production plan

	Shipments			Work Orders		
method	Count	Mean	Median	Count	Mean	Median
MRP	17	190.4	143.9	279	159.3	98.2
PM	67	47.5	7.0	416	22.4	3.6

These results show that the simulated schedule of the MRP-based production plan, on average across the 10 tested instances, was able to complete 17 out of 25 planned shipments. However, as Table 3.8 showed, these deliveries consisted of low-value end items, requiring the completion of less

tasks. It is the high-value, complex end items which were most affected by the compounded delays experienced by the work orders associated with down-stream processes.

3.5.3 Single Timeline vs Resource Dependent Timeline

A notable contribution of the proposed Planning Model is the novel approach for the discretization of the planning horizon in different timescales. Specifically, several timelines are defined representing the same planning horizon, then each resource group is assigned to one of those timelines on the basis of the characteristics of the processes which they are responsible for. There are a user-defined number of timelines, τ. The timeline with the highest level of accuracy, i.e. with the smallest duration buckets, is referred to as the *base timeline*. The duration of each time period in the base timeline is Δ. Each timeline is related to each other in that the duration of the periods in each successive timeline is twice the duration of the previous one. For example, a problem which implements 3 unique timelines with a base-timeline period duration of 10 hours will result in a timeline with 10-hour, 20-hour, and 40-hour period durations respectively.

The benefit of this approach is in the accuracy of the production facility dynamics which this approach enables while not over-complicating the problem formulation. This benefit is exaggerated in problem settings which exhibit a large variation in the typical process cycle times (setup + processing time) across different resource groups.

In the following, we evaluate the performance of the proposed Resource-Dependent Timeline approach (RDT) with 3 timelines and a base time-step of 10 hours versus a Single Timeline (ST) approach with 1 timeline with periods of 40 hour duration. In order to enable an apples-to-apples evaluation, several parameters must be recalculated when translating the problem to a ST representation, specifically the maximum production lot size, $M_{\gamma t}$, and the maximum number of setups allowed per period, $Z_{\gamma t}$. These recalculations are for the resources that were associated with timelines that have shorter periods in the resource-dependent approach we used.

Table 3.13 shows the delivery performance of the proposed framework (with Resource-Dependent Timelines) compared to a Single-Timeline representation. As shown, the proposed RDT model outperforms the ST model in both the plan and simulated schedule. The RDT model is able to *Deliver* a higher proportion of demand in both the plan and simulation, as well as more demand *On Time*. Further, the RDT has better *Lateness* performance in the plan and schedule. Another aspect in

which RDT outperforms the ST model is in its estimation of performance in the production plan compared to the schedule simulation. For example, the RDT model planned to deliver 87.6% of the demand and was able to deliver 84.2% (a 3.4% difference) in the simulation, while the ST model planned for 86.9% and was able to deliver 74.6% (an 11.3% difference). A similar conclusion can be derived by comparing the difference in planned vs actual delivery KPIs for *Lateness*.

Table 3.13. Comparison of Resource-Dependent Timelines vs Single Timeline: Analysis of planned delivery performance based on each model and its corresponding assumptions, and actual delivery performance calculated via simulation using the CTSH

Source	Timeline	% of Demanded Delivered	% of Demanded On Time	Lateness (Hours) Mean	Lateness (Hours) 25th pc.	Lateness (Hours) Median	Lateness (Hours) 75th pc.
Actual	RDT	84.2 %	21.8 %	124.6	11.0	94.0	223.0
	ST	74.6 %	6.2 %	234.5	92.0	222.0	355.0
Planned	RDT	87.6 %	42.8 %	47.2	-44.0	-5.0	104.0
	ST	86.9 %	33.6 %	86.0	-31.0	43.0	161.0

Table 3.14 compares the accrued costs in the RDT model compared to the ST model for both the production plan and simulated schedule. In this analysis, the RDT model achieved a lower *Total Penalty* in both the production *Plan* and the schedule simulation (*Actual*). The RDT also provides a better estimate of the *Total Penalty* in the production plan that is realized in the simulation. This pattern is exhibited for all cost KPIs besides the *Holding Costs*. This is mainly due to the difference in the frequency with holding costs are accrued. In the RDT model, inventory is evaluated every 10 hours, while in the ST model, it is evaluated once every 40 hours, i.e. at the granularity of the base timeline, Δ.

Table 3.14. Comparison of Resource-Dependent Timelines vs Single Timeline: Analysis of costs and penalties in the production plan and simulated schedule

Source	KPI / Timeline	Holding Cost	Under Production	Tardiness Penalty	Unfulfillment Penalty	Total Penalty
Plan	RDT	0.71 %	2.00 %	6.01 %	5.51 %	14.23 %
	ST	0.58 %	3.67 %	7.71 %	6.24 %	18.20 %
Actual	RDT	3.47 %	3.34 %	7.74 %	7.07 %	21.63 %
	ST	2.69 %	5.67 %	11.92 %	10.29 %	30.57 %
Difference	RDT	2.76 %	1.34 %	1.73 %	1.56 %	7.40 %
	ST	2.11 %	2.00 %	4.21 %	4.05 %	12.37 %

Table 3.15 compares the overall utilization rates when implementing the RDT model and the ST model for both the production *Plan* and *Actual* schedule. While the *Planned* utilization rates are comparable between the two models, the RDT outperforms the ST in the utilization observed in the CTSH simulation. The RDT again provides a better representation (and estimation) of the planned utilization compared to the realized utilization showing a difference in utilization of 3% compared to 7.83% when using the ST model.

Table 3.15. Comparison of Resource-Dependent Timelines vs Single Timeline: Analysis of resource utilization in production plan and simulated schedule

	Planned			Actual		
Timeline	Process	Setup	Total	Process	Setup	Total
RDT	65.46 %	13.38 %	77.81 %	60.55 %	14.26 %	74.81 %
ST	63.03 %	14.50 %	76.82 %	55.59 %	13.40 %	68.99 %

Table 3.16 shows an analysis of size and count of *Shipments* and *Work Orders* generated in the production plans of the RDT and ST models. The RDT production plan generates a larger *Count* of shipments and work orders, mainly as a result of the ability to begin production of work orders and shipments at a finer time granularity (every 10 hours vs 40 hours). As a result, the *Mean* and *Median* lot sizes of shipments and work orders are also smaller in the RDT model.

Table 3.16. Comparison of Resource-Dependent Timelines vs Single Timeline: Analysis of generated shipments and work orders in the production plan

	Delivery Shipment Qty				Work Order Qty			
Timeline	Count	Mean	Median	Total	Count	Mean	Median	Total
RDT	72.10	3.21	2.30	231.35	422.90	22.53	9.72	9526.90
ST	48.90	4.24	3.21	207.41	234.60	38.68	17.44	9073.47

Table 3.17 provides a view into the delays *Shipments* and *Work Orders* in the simulated schedule compared to the production plan for both the RDT and ST models. As shown, the delays experienced in the simulation of the RDT production plan are much less than those in the ST production plan.

Table 3.17. Comparison of Resource-Dependent Timelines vs Single Timeline: Analysis of delays in the simulated execution of work orders relative to the scheduled production plan

	Shipments			Work Orders		
Timeline	Count	Mean	Median	Count	Mean	Median
RDT	67	47.5	7.0	416	22.4	3.6
ST	42	88.6	42.0	229	40.8	8.5

3.5.4 Representation of Setup Decision Variables

When considering the runtime required to identify high quality solutions, the definition of variable types is critical. All models presented to this point have leveraged integer variables for the setup activity of processes, $z_{\gamma t}$, as it seems to be the most accurate way to capture the reality of the production schedule. However, as alluded to in Section 3.3.1, it is possible to model the production planning problem using binary decision, where the decisions changes from "how many setups to perform in period t?", to "perform a setup activity during this period or not?". Additional steps must be taken to ensure that the calculation of the maximum production lot size, $M_{\gamma t}$, is correct so that resources aren't over-utilized. The assumptions taken in these *preprocessing* steps result in the possibility of several variations of a binary variable implementation.

For example, when implementing the binary variable representation of the setup activity, $M_{\gamma t}$ can be calculated assuming that capacity consumed by the associated setup activity is occurring on: 1) ONE resource within the resource group, 2) ALL resources within the resource group, or 3) SOME of the resources within the resource group. An implementation which assumes that a setup will take place on all resources represents the most conservative option. Specifically, this will result in the smallest maximum production lot size allowed for each setup activity. To account for the other options, we would adjust the setup time, S_{γ}, associated with any setup decision.

Regardless of the variation of implementation, we expect the resulting production plans leveraging binary setup decision variables to have a less accurate translation when converted into the executable schedule, resulting in more delays compared to the production plan which leverages integer decision variables for setup activities which better reflect the existence of multiple machines in the resource group and small batch sizes. We expect that the binary variation assuming one

resource setup will result in the worst solution performance and the assumption of a setup on all resources to perform the best.

Further, considering that the assumptions taken to construct the production planning model, i.e. lead time buffers, planning horizon discretization, etc., are already enough to muddy the results, we also evaluate the performance of a continuous variable representation of the setup variable. We expect that the runtimes to solve the model using continuous variables for setup decisions to be reduced drastically, but the resulting production plan to be even less accurate when converted into a continuous time schedule. We expect that the total setup time accounted for in the planning model will be greatly under-represented. In the following, we evaluate the solutions associated with each of the aforementioned representation methods.

Table 3.18 provides a view on the performance of delivery KPIs in both the production plan and simulated schedule, with respect to the type of decision variable (*varType*) used to represent production setup activities. Within the production *Plan*, as expected, the relaxed Continuous DV model is able to deliver the largest proportion of demands within the planning horizon, as well as the largest percentage of *On Time* deliveries. Both the Integer and Binary DV representations behave similarly for both the production plan (*Planned*) and schedule simulation (*Actual*). However, the Integer representation slightly outperforms the Binary when considering the *Lateness* of deliveries, while the Binary DV model slightly outperforms in the % *Delivered* and delivered *On Time*.

Surprisingly, the Continuous DV production plan is capable of delivering the most demands in the simulated schedule, while also delivering more demands *On Time* than the Integer DV model. However, the *Lateness* of deliveries in the simulated schedule of the Continuous DV model is significantly worse than in the Binary and Integer DV models.

Table 3.18. Comparison of Integer vs Binary vs Continuous decision variables for setups: Analysis of planned delivery performance based on each model and its corresponding assumptions, and actual delivery performance calculated via simulation using the CTSH

		% of Demanded		Lateness (Hours)			
Source	**varType**	Delivered	On Time	Mean	25th pc.	Median	75th pc.
Actual	Binary	80.4 %	25.2 %	89.4	-15.0	63.0	162.0
	Continuous	93.8 %	21.9 %	118.8	14.0	101.0	199.0
	Integer	79.8 %	19.7 %	83.1	-8.0	62.0	160.0
Planned	Binary	81.9 %	48.3 %	16.5	-62.0	-27.0	64.0
	Continuous	98.6 %	60.8 %	23.2	-60.0	-19.0	81.0
	Integer	81.4 %	47.0 %	8.3	-63.0	-32.0	36.0

Table 3.19 shows the costs associated with each DV model for both the production plan (*Plan*) and simulated schedule (*Actual*), and the difference between the two. Surprisingly, again, the Continuous DV model results in the lowest *Total Penalty* for the simulated schedule, while the Binary and Integer DV models perform similarly. Specifically, the Continuous DV model outperforms the Binary and Integer DV models in the cost accrued as an *Unfulfillment Penalty*. It should be noted that the Continuous DV estimation of costs in the production plan are the most inaccurate for all measures (besides *Holding Cost*). The padding associated with lead times and time buckets is put to use by the Continuous DV plan which allows more production to happen each period as it does not fully capture the setup times.

Table 3.19. Comparison of Integer vs Binary vs Continuous decision variables for setups: Analysis of costs and penalties in the production plan and simulated schedule

	KPI	Holding Cost	Under Production	Tardiness Penalty	Unfulfillment Penalty	Total Penalty
Source	**varType**					
Plan	Binary	0.61 %	3.84 %	4.26 %	8.12 %	16.82 %
	Continuous	0.28 %	0.12 %	4.73 %	0.19 %	5.30 %
	Integer	0.60 %	4.30 %	4.10 %	8.50 %	17.50 %
Actual	Binary	3.47 %	4.78 %	5.93 %	8.60 %	22.77 %
	Continuous	3.07 %	4.28 %	7.71 %	1.99 %	17.04 %
	Integer	3.38 %	4.90 %	6.19 %	9.05 %	23.51 %
Difference	Binary	2.86 %	0.94 %	1.67 %	0.48 %	5.95 %
	Continuous	2.79 %	4.16 %	2.98 %	1.80 %	11.74 %
	Integer	2.78 %	0.60 %	2.09 %	0.55 %	6.01 %

Table 3.20 presents a comparison of the utilization rates achieved in the production plan and schedule simulation by *varType*. Note that the Continuous DV model greatly under-estimates the amount of time spent for *Setup* in the production *Plan*. However, when simulated, the resulting *Actual* utilization rate does not suffer (compared to the Binary and Integer DV models). This is due to the Continuous DV plan sequencing each work order such that it could be followed with minimal contradiction in the simulated schedule, besides being delayed by the underestimated setup time. Further, due to the conservative assumptions implemented, the setup time accounted for in the *Plan* is greatest for the Binary DV model. The Integer DV model provides the best estimation of *Total* utilization in the production *Plan*, with only a 0.77% difference against the *Actual:Total* utilization.

Table 3.20. Comparison of Integer vs Binary vs Continuous decision variables for setups: Analysis of resource utilization in production plan and simulated schedule

	Planned			Actual		
varType	Process	Setup	Total	Process	Setup	Total
Binary	66.41 %	10.01 %	75.93 %	62.46 %	12.76 %	75.22 %
Continuous	69.95 %	2.65 %	71.41 %	62.12 %	18.49 %	80.61 %
Integer	65.31 %	8.68 %	73.54 %	61.15 %	11.69 %	72.83 %

Table 3.21 provides an analysis on the *Count* and quantities associated with the generated *Shipments* and *Work Orders* in the production plans for each respective DV type. As expected, the Continuous DV model results in the largest number of generated shipments and work orders with the lowest *Mean* and *Median* quantities associated with each task.

Table 3.21. Comparison of Integer vs Binary vs Continuous decision variables for setups: Analysis of generated shipments and work orders in the production plan

	Delivery Shipment Qty				Work Order Qty			
varType	Count	Mean	Median	Total	Count	Mean	Median	Total
Binary	80.60	2.89	2.09	232.73	479.30	19.27	8.88	9235.14
Continuous	110.00	2.29	1.43	251.96	758.90	13.11	5.10	9951.45
Integer	74.70	3.09	2.20	230.84	445.60	20.78	9.47	9260.73

Table 3.22 shows the delays in the start times of delivery shipments and work orders during the scheduling simulation compared to the respective production plans. The Continuous DV production

plan results in the largest *Mean* and *Median* delays for both *Shipments* and *Work Orders*, while no clear superior model can be identified between the Integer DV and Binary DV plans.

Table 3.22. Comparison of Integer vs Binary vs Continuous decision variables for setups: Analysis of delays in the simulated execution of work orders relative to the scheduled production plan

	Shipments			Work Orders		
varType	Count	Mean	Median	Count	Mean	Median
Binary	76	39.4	3.5	474	20.7	3.1
Continuous	103	62.8	13.0	730	73.7	32.2
Integer	70	45.0	4.0	442	20.2	1.2

3.6 Practical Considerations and Model Extensions

To enable the proposed framework as a viable and effective scheduling tool, it is necessary to incorporate the existing state of the shop floor at the time in which a schedule is generated. It is also important to consider alternative scheduling objectives which incentivize good manufacturing practices and allow the user to focus on different performance indicators that best represent the shop priorities at a particular time. In the following, we present several extensions which address some of these considerations.

3.6.1 Incorporating Current Shop-Floor Conditions

Existing Orders and Work-In-Progress

We define two classes of work orders which may exist in the production environment at any given time of the factory. The first category of work orders are ones which have already been scheduled or are Work-In-Progress at the time the PM is run. We incorporate these work orders into the PM during the preprocessing phase of the procedure. We determine the time at which the work order will be completed, assuming no delays, and define the output of the work order a scheduled receipt. We account for the capacity requirements of these work orders by removing available capacity for each resource they visit in the remaining steps of their routings.

The second type of existing work order, $w \in W$, we consider calls for the production of a specific part and has a specified output quantity, q_w, but has NOT YET been scheduled. We account for these work orders by creating a new binary production variable x_{wt} which takes a value of 1 if work

order w is scheduled to begin production at time t, 0 otherwise. These types of work orders will be included in the optimization model similar to the standard production variable u_{it}; however, the quantity which is produced will be fixed at the value q_w and the work order scheduling decision is thus captured by a binary variable, x_{wt}. The existing work orders which call for the production of part i are denoted by $w \in W^i$: $W^i \subseteq W$. This subset mapping is used to link the production variables u_{it} with the work order production variables, x_{wt}, and is introduced to the formulation in each constraint which u_{it} is found.

Target Safety Stock Levels and Production Yields

In practice, it is common for firms to hold excess inventory of parts to hedge against the possibility of delayed production or quality conformance issues. We consider a decision-maker's preference for holding extra inventory of a part, i, by defining a target safety stock level, v_i^*, and a per-unit per-period cost, p_i, associated with inventory being below the target safety level. We define the non-negative variable, d_{it}, as the *difference* between the inventory level and the target safety stock level for each time period: $d_{it} \geq v_i^* - v_{it}$.

It should be noted that it would be possible to consider a time-dependent target safety stock which considers materials requirements congestion by adding a time-index to v_i^*, i.e. v_{it}^*. This consideration may be useful, as it provides an opportunity to incentivize the production of a part in preparation for an influx of demand. Consider the example where it is known that a part which consumes extensive resources will be required far into the planning horizon, but the resources it requires are known to be under utilized in the immediate portion of the horizon. By specifying a higher target safety stock level for this part in the near term, then reducing it once the parts have been produced, it provides the scheduler a method for guiding the optimizer.

To hedge against the possibility of quality issues and scrap in production, it is critical to include considerations of production yield in the optimization model. Consideration of production yield could be handled in preprocessing by incorporating a safety factor to the Units per Parents dataset, n_{ik}. For example, if it is known that production of a part is satisfactory 90% of the time, i.e. yield is 0.90, and an assembly requires 10 units of a specific produced components, then we'd represent the UPP (n_{ik}) as $\frac{10}{0.9} = 11.11$. This method of incorporating yield considers only the expected case

of yield. Safety stock will be coupled with this method in order to accommodate the uncertainty in the yield at any particular time.

3.6.2 Accounting for Further Shop-Floor Complexity

Alternative Routings and Part-Steps

The presented formulation is limited such that there is no flexibility in the schedule within the routing of a production lot. The only decision is when a production lot is started. The schedule of the subsequent processing steps to complete that work order are then fixed. However, the production plan could be improved by allowing flexibility within the routing by inserting idle time between route-steps. One method for introducing this flexibility into the planning model would be to define alternative routes for the work orders. We introduce a new index, ω, to denote each possible production routing, and the set, Ω, as the collection of all possible routes. Each route, ω, will produce as output a part i. We define the subset of all routes which produce part i as $\omega \in \Omega^i$. This subset of routes defines the set of alternative routes which are capable of producing part i and allows for the introduction of flexibility by allowing for unique definitions of lead time, now L_ω, and lead time offsets, now $l_{\omega j}$, during the production of part i. Production decisions, formally u_{it} would be denoted as, $u_{\omega t}$, defining the quantity of units which begin production in route ω during period t. The production of part i in each period would be equal to: $u_{it} = \sum_{\omega \in \Omega^i} u_{\omega t}$. Further, all instances of L_i would be replaced by L_ω, and the indices for processes, $\gamma \in \Gamma$ would need to be expanded from the tuple, ijb to $ijb\omega$. This alternative routing method could also allows for the flexibility of specifying secondary resources which are capable of completing certain processes. For example, if multiple resource groups can process a part, we would specify two alternative routes for the part, unique in the resource group that each routing specifies.

Another approach to allow for flexibility in the production of parts, would be to *split* the parts routing into part-steps. This would reduce the number of resources each part requires throughout its routing. New part indices would be created for each of the part-steps. The benefit of this approach would be that this allows for a route with potentially dozens of route-steps to be represented as a series of decisions with less complex routings, while introducing the ability to introduce idle time between part-steps. The drawback is that increased problem size and the departure from the common shop-floor practice of keeping the lot constant throughout its routing. Note that as the

production quantity at each step is decided separately in the part-step model, the optimal lot sizes will most likely vary.

Resources Sub-Groups

Production within factories requires the use of value-adding resources that transform raw materials into finished goods. We consider three classes of resources: machines, operators and tools. Machines are physical equipment which add value to parts, while remaining in a fixed location on the shop floor. Tools are also physical equipment, but they are not fixed to a specific location. Operators represent the human workforce who perform setup activities and run processes on machines. Completing a particular process may require a machine, a specific tool (e.g. a cast-iron die) and an operator to be available for potentially different amounts of time.

Due to the large size of modern-day aerospace factories, it is infeasible to represent each machine, operator, and tool as an independent entity in a medium-term planning model. The aggregation of similar resources into groups relieves some of the computational burden. Machines are grouped into work centers, operators into operator groups, and tools into tool groups. Capacity is aggregated across all individual entities within a group. The parameter $|E_j|$ defines the number of resources within a resource group, j, i.e. number of machines in a work center.

In practice, it is unlikely that all resources within a group can be aggregated into a homogeneous set. Consider the example in which several generations of a similar machine are grouped into a resource group. It is possible that some of the older generations may not be able to process all of the parts that the newer generation can. We introduce the concept of resource *subgroups* to account for this circumstance. By defining subgroups of resources, we allow the possibility to define a parts routing to require processing by a newer generation of machines within a resource subgroup, while also allowing that subgroup to be assigned the production of parts that can also be processed by the older generation. To add this consideration to the planning model we incorporate the following variables and parameters. We define the set $g \in G^j$ to describe the set of resource subgroups within the resource group j. The resource subgroups will also be represented as a resource group, $j \in J$. The variable c_{jt} is the capacity of resource group (or subgroup) j consumed in time period t. We replace the capacity constraint (C3) with the following:

$$\sum_{i \in I^j} \left(R_\gamma m_{\gamma t} + S_\gamma z_{\gamma t} \right) = c_{jt} \quad \forall \quad j \in J, t \in T^j \quad (G1)$$

$$c_{jt} + \sum_{g \in G^j} c_{gt} \leq C_{jt} \quad \forall \quad j \in J, t \in T^j \quad (G2)$$

The new Constraints (G1) calculate the capacity allocated of each resource group (or subgroup), $j \in J$, in time period t. Constraints (G2) ensure that the total capacity allocated for each resource group is less than the available capacity in that time period. The allocated capacity in this constraint is calculated as the sum of the capacity allocated from its own group index, j, plus the total allocated capacity of all of its subgroups, $g \in G^j$.

Delivery Requirements

We represent the delivery decision variable, y_{ot}, as a continuous variable in the PM. However, it may not be appropriate to deliver partial units of a finished good. In this case the variable y can be represented as an integer, rather than a continuous variable. The presented formulation also allows for partial deliveries. However, it may be required to deliver all items within an order at the same time, e.g. to reduce transportation costs. This can be included by defining y_{ot} as a binary variable and modifying the inventory dynamic constraints using the multiplier Q_o.

A further generalization in delivery requirements would consider the case of multi-product orders. We define an order, o, as a demand for a single product, i. However, in practice it is likely for customers to purchase multiple products. Sets would need to be defined for customer orders and for line items. A mapping signifying which line items belong to any order o would also be introduced, similar to the mapping O^i which defines the set of line items which call for the delivery of part i.

3.7 Conclusion

In this chapter, we introduce the Hierarchical Job Shop Schedule Planning and Execution Problem. The 3-phase solution procedure we develop includes: 1) the Planning Model (PM) to solve the Multi-Level Capacitated Lot-Sizing Problem (MLCLSP), 2) the Node Pegging Model (NM) to solve the Order-Lot Matching Problem (OLMP), 3) and the Continuous-Time Scheduling Heuristic (CTSH, presented in Section 2.7) to solve the continuous-time scheduling assignment problem.

Our main contributions to the literature consider the novel modelling techniques developed for the MLCLSP, as well as the development of the integrated solution procedure in a comprehensive framework. We focus on practical challenges that have not been addressed before in the modeling of these systems. This gap in the literature may be one of the root causes as to why companies continue to struggle in implementing optimization techniques for generating their production plans in practice.

Specifically, we develop a representation of the planning horizon in the MLCSLP with a set of Resource-Dependent Timelines, each with a unique discretization of the planning horizon, that dynamically associates each resource group with the timeline which best suits it on the basis of the typical processes it faces. We also consider multi-step, cyclical routings subject to positive lead times, observed for each step of the production routing.

Another major contribution to the literature is the application of the proposed Node Pegging Model (NM) following the generation of an optimized production plan. The purpose of the NM is to match production lots with the external demand line items their outputs will satisfy. This association provides planners visibility to the status of the shop with respect to the delivery performance they can expect to achieve. Further, this association allows for visibility into the "criticality" of each production lot, in that it is known which line items it is meant to satisfy are at risk of being delivered late. This information, although outside of the scope of this dissertation, can be leveraged to develop prioritization heuristics in the CTSH to minimize the costs which are realized in the simulated schedule.

We evaluate the proposed integrated solution procedure against several benchmark cases. Problem instances were generated using the algorithms described in Section 2.5. We first compare the Planning Model developed for the MLCLSP against an MRP-based heuristic (presented in Section 2.6). Then we compare the performance of the Resource-Dependent Timeline methodology against a Single Timeline representation of the same problem. We consider the performance of each test methodology in both the aggregated context the planning problems were solved in, as well as the executed plan in the unaggregated context the problem instances represent. We find that the proposed Planning Model greatly outperforms the MRP-based heuristic and that the Resource-Dependent Timeline methodology outperforms the Single Timeline methodology. The value of the proposed methodology is assessed based on the performance of the executable schedule.

We also evaluate several variations of the Planning Model, specifically considering alternative decision variable representations for setup activities. Surprisingly, we find that using a continuous variable for setup activities results in a production plan which is competitive with the production plans made using binary and integer variables. However, the plans which were made using continuous variables resulted in the most inaccurate estimation of realized penalties and the most delays once translated into an executable schedule.

We refer the reader to Appendix B for a presentation of the methods and experiments conducted to calibrate the user-defined parameters associated with Planning Model used throughout this numerical implementation. Future work will further extend the models to consider further practical aspects discussed in Appendix 3.6. These extensions include enhancing the Planning Model to consider: 1) sequencing decisions for existing work orders, with fixed production lot quantities, 2) alternative timed-routes for each part, 3) resource sub-groups within resource groups that exhibit slight, yet practically significant, differences from other sub-groups in the group, 4) target safety stocks, 5) production yield, 6) the introduction of part-steps as a means to break up long part routings into more manageable size, 7) more restrictive delivery constraints, such as enforcing all units in a line item to be shipped at the same time, etc.

Further, future experimentation is needed to evaluate the scalability of the proposed solution procedure. We identify three dimensions for which the problem will increase in difficulty: 1) the utilization of the facility, 2) the size of the problem, measured in the number of items, orders and resource groups which are considered, and 3) the complexity of the problem, measured in the depth of each Bill of Materials (the number of offset levels), the width of each BOM (the number of children each part can have), and the number of route-steps which are associated with each production routing. Given the results found in this chapter, these larger problems should also be evaluated with the use of continuous-variables for setup decisions, considering this implementation resulted in feasible, competitive production plans.

CHAPTER 4

DIRECT VS RELATIVE POSITIONAL SCHEDULING

4.1 Introduction

Machine scheduling problems are one of the most widely studied problem families in operations research. The roots of the problem go as far back as the late 19th and early 20th centuries when the industrial revolution began. With the introduction of factories and automation into our lives, the need for planning the shop floor arose. However, due to the difficulty of these problems, most of the literature on machine scheduling has focused on developing heuristics, testing meta-heuristics and exploring the effectiveness of dispatching rules. Even though the prominence of computers allows us to solve problems of larger scale today, especially when leveraging commercially available solvers such as Gurobi, the tradition of machine scheduling continued as it is, mostly avoiding exact methods. Therefore, comparison of different modeling techniques in exact approaches such as mixed integer programming (MIP) has received surprisingly little attention in machine scheduling literature.

In this chapter, we compare and evaluate the computational performance of two families of MIP models for solving different scheduling problems faced by Make-To-Order firms, such as Artaic - Innovative Mosaic, whose needs motivated this research. Artaic is a custom mosaic design studio and manufacturer in Boston, MA. They use robotic fabrication, which allows for fast, flexible and accurate assembly of unique tile work for their customers. They have two production stages. The first one is tile tubing, which groups different colored tiles into tubes that feed into the second stage: automated tile assembly. In this stage, the unique design is loaded up into a computer, which is translated into schematics for the robotic arm to build. A variety of robotic arms with different capabilities are available to perform the tile assembly. The product is then packaged and shipped to the customer. This process is summarized in Figure 4.1.

The second stage is the bottleneck of their production and can be modeled as an unrelated parallel machine scheduling problem. Explicitly including the first stage makes the problem a flexible

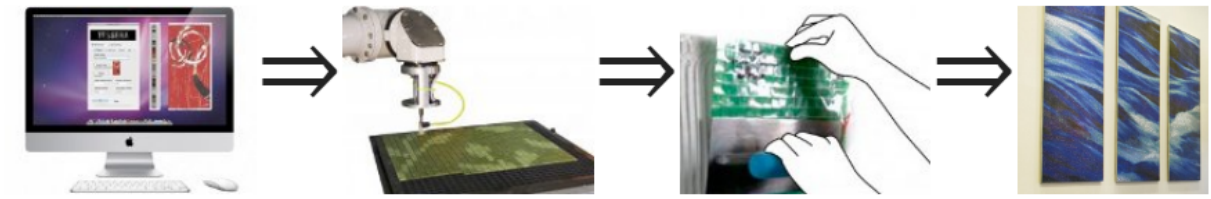


Figure 4.1. Example production system of Artaic

flow shop problem. We consider the scenario where the manufacturer hosts multiple generations of resources that are capable of processing the second stage operation. Specific to Artaic, newer generations of machines do not require a first-stage tubing operation. In this case, orders can either be processed in a single operation by a newer generation of machine, or as a two-stage process by an older generation of machine. In all problems, the scheduling objective of interest is to minimize a weighted combination of makespan (to increase shop utilization) and total weighted tardiness (to ensure customer satisfaction) subject to sequence-dependent setups and machine eligibility restrictions in which some machines are not capable of producing some product types.

MIP formulations for scheduling problems can be classified based on the type of binary decision variables used to define the production schedule. Three-classes of binary decision variables are found in scheduling literature. This includes formulations using direct-positional assignment variables, proposed by Wagner [181], relative-positional linear ordering variables, proposed by Manne [112], and time-indexed variables, proposed by Sousa and Wolsey [161] (based on the work of Bowman [28]).

Direct-positional decision variables control the assignment of jobs to specific positions in the processing sequence of resources. Binary variables are represented in the form, x_{ijk}, and take a value of 1 when job i is assigned to position k in the production sequence of resource j. Relative-positional decision variables control the relative ordering of jobs within the processing sequence. Binary variables takes the form x_{ijk} and take a value of 1 when job i precedes job k in the production sequence of resource j. Time-indexed formulations define production by deciding the time period in which jobs are completed (or started). Binary variables take the form x_{ijk} and take a value of 1 if job i is completed (or started) in time-period k by resource j. Time-indexed formulations represent time in discrete form, while direct- and relative-positional represent time in the continuous form.

In this study we limit our analysis to the direct- and relative-positional assignment formulation classes, given their capability to consider sequence-dependent setup activities such as those faced by our industrial partner.

The contributions of this chapter will be four-fold: 1) novel scheduling formulations are developed using direct-positional and relative-positional decision variables, for both the unrelated parallel machine and flexible flow shop problems, representing a generalization of the problem faced by our industrial partner. Note that we do not consider formulations with time-indexed decision variables in this chapter as this formulation is not suited for model sequence-dependent setup times, 2) a framework and data generation algorithm is developed for testing instances that result in a realistic machine scheduling problem with tight due dates, 3) the models will be used to solve randomly generated solutions of varying size to compare their relative solution quality and speed, 4) these families of models will be compared on the basis of problem size complexity (numbers of variables and constraints required).

The remainder of the chapter is organized as follows. In Section 4.2, we briefly summarize the literature. In Section 4.3, we present the mathematical models. In Section 4.4, we propose the computational studies which will be conducted to evaluate these formulations. In Section 4.5, we provide concluding remarks.

4.2 Literature Review

In our review, we focus on papers that develop and compare exact approaches for scheduling problems similar to the one we consider.

We start our review with the single machine scheduling problem. In this problem, n jobs must be sequenced for processing of a single operation on a single machine. Keha, Kowala and Fowler [86] compare the computational efficiencies and behaviors of four different MIP formulations for the single machine scheduling problem. Two relative-position formulations, one direct-position and one time-indexed formulation are proposed and considered for problems which optimize weighted completion time, maximum lateness, number of tardy jobs and weighted tardiness as their objective functions. They also consider release dates as a factor in their computational study. For the relative-position models, the authors compare formulations with binary decision variables which define whether a job, i, is completed right before another job, k, (see [12, 137]) or any time

before k, (see [48]). These models are referred to as having *Completion Time Variables* and *Linear Ordering Variables*, respectively. The authors find that the time-indexed approach results in models which are most difficult to solve, and that the direct positional formulation provides the best formulation. The authors cite this formulation's ability to solve the tested problem instances optimally within the time limits and the scalability of the LP relaxation of the formulation as key superiorities. The completion-time variable and direct-position formulations also provide the best approach for identifying feasible solutions quickly when considering job release date constraints. It is also noted that the time-indexed and linear ordering variables produce tighter bounds for the problem. However, the authors observe that it is harder to solve the LP relaxations for these formulations as the number of jobs increases. Note that the relative-positional formulation that we consider most resembles the completion-time variables in this study.

Parallel machine scheduling problems extend the single machine scheduling problem such that multiple resources are available to process jobs. The problem becomes one of scheduling n jobs to be processed by on one of m machines. Parallel machine problems can be classified by the relationships of each of the machines, including identical parallel machines, uniform parallel machines (also referred to as non-identical) or unrelated parallel machines. Identical parallel machines process each job at the same rate as each other, uniform machines at different yet consistent rates, and unrelated machines with job-dependent processing rates.

Unlu and Mason [177] model the parallel machine scheduling problem using four different mixed integer programs, two of which use relative positional variables (called Network model and Linear Ordering Model in the paper). One of the remaining two is direct positional and the other one is time-indexed. Even though they propose models for non-identical and unrelated parallel machine scheduling problems for a variety of different objectives, they consider only makespan and weighted completion time as the objective functions and only identical parallel machine in their computational study. They also consider ready-times (release dates) in the computational study. Extensive analysis of each objective function/number of parallel machines/existence of release dates combination, leads them to conclude that: 1) the time-indexed formulation is superior when job processing times are small, and 2) the feasible solutions found by the direct-positional formulation are very close to optimal in most test cases but struggles in producing sufficient lower bounds due to the disjunctive nature of the formulation. The authors suggest the incorporation of valid inequalities

can help improve this limitation. Further, the linear-ordering formulation is cited as giving good lower bounds for smaller problems, but its performance quickly degrades as the number of jobs increases.

Yu and Hung [197] also model the parallel machine scheduling problem using three different mixed integer programs, all of which use relative positional variables. The authors consider a formulation using immediate-precedence variables (F1), an enhanced version of the immediate-precedence formulation which also leverages binary machine-assignment variables (F2), and a linear-ordering formulation. The enhancements proposed in F2 separate the decisions associated with the decisions variables defined in F1 into 2 unique variables. Specifically, the decision variables in F1, x_{ijk}, takes a value of 1 if job i is processed immediately before job j on machine k, 0 otherwise. In F2, the variable x_{ik} takes a value of 1 if job i is assigned to machine k, and y_{ij} takes a value of 1 if job i precedes job j on the same machine. The objective function they consider is to minimize total tardiness, subject to job release date constraints. They find that hybrid direct-relative assignment formulation, F2, is superior based on time to optimality and optimality gaps on non-optimal solutions. Since they do not consider direct positional variables at all, their results are not directly comparable to ours.

The flow shop problem extends the parallel machine problem by considering jobs which require processing on multiple resources. In this problem, n jobs must go through o different processing stages. The sequence which each job visits the production stages is identical. Flow shop problems can be distinguished as either permutation flowshops or flexible flowshops. In a permutation flowshop scheduling problem (PFSP), each production stage is represented by a single machine, and the production sequence in which each machine processes the jobs must be the same. In a flexible flowshop (FFSP), the permutation flowshop problem is generalized so that at least one of the production stages is represented as a parallel machine production environment. The hybrid flexible flowshop (HFFSP) provides a further generalization in that jobs are allowed to skip stages. For instance, in our specific manufacturing setting, this would represent the case in which a job does not require the first stage (tubing operation).

Stafford, Tseng and Gupta [166] consider the PFSP. The authors distinguish between 2 classes of MIPs, referred to as the Wagner and Manne families, consisting of 3 and 5 unique models, respectively. In our terminology, the Wagner family of models correspond to direct-positional

variables and Manne family of models correspond to relative-positional variables. They consider the minimization of Makespan as their single objective function and do not consider setups in the formulations they evaluate. The authors find that the relative-positional variables result in models with more constraints and less binary variables than the direct position variable models. Both models require similar number of real variables. Even though the permutation flowshop differs from our tubing model, which is a flexible flowshop, their computational findings suggest that formulations with direct-positional variables dominate those with relative-positional variables.

Naderi and Gohari [122] consider the HFFSP. The authors note there is a very limited available literature on the topic of MIP development for this class of scheduling problems, let alone comparative studies. The authors develop four MIP models, including three novel formulations, with the objective of minimizing Makespan. The first model, (Model 1) uses immediate-predecessor variables, similar to the models proposed by Kis and Pisch [92] and Ruis et al. [146]. The authors also develop a hybrid assignment-predecessor formulation (Model 2). The assignment binary variable, Y_{jil}, defines if a job j is processed on a specific machine, l, at stage i, and the predecessor variable, X_{jikl}, defines if a job j is processed before a job k at stage i on machine l. Another direct-position model (Model 3) uses two assignment variables: one responsible for assigning each job to a specific machine at each stage, and one responsible for assigning each job a position in the queue. The fourth model (Model 4) defines a decision variable, X_{jik}, which takes a value of 1 if a job j is processed after job k at stage i and another, Y_{jil}, if job j is processed at stage i on machine l. The relative-position in this variable differs from the previous hybrid assignment-predecessor formulation in that a machine index is not used in this formulation. The authors find that Model 4 requires the least binary variables, while Model 1 requires the most binary variables, yet the least constraints. Models 2,3 and 4 are shown to solve small-sized problems (up to 12 jobs, 4 stages) with similar performance, while outperforming Model 1. Literature on comparisons of MIP exact-method solutions for the FFSP and HFFSP is scarce. For a detailed review on the literature of the hybrid flow shop problem we refer the reader to Ruiz et al. [147].

Our research adds to the literature as one of the few studies which provides a computational comparison of various representations of the HFFSP. Note that all of the studies above consider only one production environment setting (i.e. single machine or flow shop or flexible job shop) while our study encompasses multiple settings to understand the robustness of the formulation choice

across problem families. Our study also considers sequence-dependent setups as a complicating factor, which is not considered in most of the comparison literature.

4.3 Mathematical Models

In this section, we develop direct- and relative-positional scheduling formulations for a variety of practical settings rooted in the needs of our industrial partner. We begin with the unrelated parallel machine problem. We first formulate the direct-positional formulation, then the relative-positional formulation. In Section 4.3.2, we extend each of the aforementioned formulations for a special case of the two-stage flexible flow-shop problem. In all models, we assume deterministic processing and setup times, and no machine breakdowns. We do not allow preemption, job-splitting or outsourcing.

4.3.1 Unrelated Parallel Machines Setting

In the unrelated parallel machine problem, the decision-maker must schedule $|I|$ jobs, $i \in I$, to be completed by one of $|J|$ unrelated parallel machines, $j \in J$. A_{ij} defines whether machine j is capable of processing job i, and P_{ij} defines the time required for machine j to process job i. Each job is also defined by a due date, d_i, a per-period penalty for tardiness, c_i, and an earliest allowable starting time, s_i. The objective of the scheduler is to minimize the sum of the Makespan of the backlog, C_{max}, and the total weighted tardiness of all jobs.

4.3.1.1 Direct-Positional Formulation

In the direct-positional assignment models, we define the set, $k \in K$, as the set of possible positions a job can be assigned to on each machine, where $K = \{1..|I|\}$. The direct-position binary decision variable, x_{ijk}, takes a value of 1 if job i is assigned to be completed in position k on machine j. The variable t_{jk} defines the completion time of position k on machine j. In the following we present the notation and formulation of the **D**irect-positional **P**arallel machine problem (**DP**).

	Notation	Description
Sets	$j \in J$	Set of machines, $\{1..\|J\|\}$
	$i \in I$	Set of all jobs, $\{1..\|I\|\}$
	$k \in K$	Set of positions on the machines, $\{1..\|I\|\}$
Parameters	$\|I\|$	Number of jobs
	$\|J\|$	Number of machines
	M	A very large number
	c_i	Penalty for tardiness of job i
	d_i	Due date of job i
	s_i	Earliest allowable start time of job i
	A_{ij}	1 if job i can be processed by machine j, 0 otherwise
	P_{ij}	Processing time of job i on machine j
Variables	C_{max}	Shop Makespan
	D_i	Tardiness of job i
	t_{jk}	Completion time of position k on machine j
	x_{ijk}	1 if job i is assigned to position k on machine j, 0 otherwise

Table 4.1. Notation for direct-positional unrelated parallel machine problem (DP)

Formulation **(DP)**

$$\min \quad C_{max} + \sum_{i \in I} c_i \, D_i$$

s.t.

$$t_{jk} \leq C_{max} \quad \forall \; j \in J, k \in K \tag{1}$$

$$t_{j,k-1} + \sum_{i \in I} P_{ij} x_{ijk} \leq t_{jk} \quad \forall \; j \in J, k \in K \tag{2}$$

$$(s_i + P_{ij}) \, x_{ijk} \leq t_{jk} \quad \forall \; i \in I, j \in J, k \in K \tag{3}$$

$$d_i + M(1 - x_{ijk}) + D_i \geq t_{jk} \quad \forall \; i \in I, j \in J, k \in K \tag{4}$$

$$t_{j0} = 0 \quad \forall \; j \in J \tag{5}$$

$$\sum_{i \in I} x_{ijk} \leq 1 \quad \forall \; j \in J, k \in K \tag{6}$$

$$\sum_{j \in J} \sum_{k \in K} x_{ijk} = 1 \quad \forall \; i \in I \tag{7}$$

$$\sum_{k \in K} x_{ijk} \leq A_{ij} \quad \forall \; i \in I, j \in J \tag{8}$$

$$\sum_{i \in I} x_{ijk} \geq \sum_{i \in I} x_{i,j,k+1} \quad \forall \; j \in J, k \in K \tag{9}$$

As mentioned, the scheduler's objective is to minimize a weighted combination of makespan and tardiness. Constraints 1 ensure that the makespan is a time after all jobs have been processed. Constraints 2 require the completion time of the job processed in position k on machine j to be

after the completion time of the job in the previous position, $k-1$, plus its own processing time. If job i is assigned to position k on machine j, Constraints 3 ensures that its completion time is after its earliest allowed start time, s_i, plus processing time. Constraints 4 calculate the tardiness of each job i; notice that the big M term makes the Constraints relevant only when job i is assigned to position k on machine j. Constraints 5 initialize the completion time at position $k = 0$ as 0. Constraints 6 ensure at most one job is assigned to position k on machine j. Constraints 7 ensure each job is assigned to exactly one position on a machine. Constraints 8 guarantee that job i is not assigned to any machines that are not capable of processing it. Constraints 9 require earlier positions to be filled first.

Direct-Positional, with Sequence-Dependent Setups

We consider the problem faced by MTO firms which offer Mass Customizable products for their customers. Although each job is unique, we assume that each job i, can be characterized as one of $|U|$ specific product types, $u \in U$. In the case of Artaic, product types are defined by the type of tile used for each mosaic. These tiles are unique in their surface area and material.

The binary parameter, q_{iu}, takes a value of 1 if job i is a product of type u. f_{juv} is the sequence-dependent setup time required on machine j to changeover from a product of type u to a product of type v. The binary decision variable, y_{jkuv}, takes a value of 1 if a setup from product type u to product type v occurs on machine j at position k.

We also extend the definition of the job set, I, to include an artificial job, $i = 0$. We refer to this extended job set as I_0. This artificial job is represented as an artificial product type, $u = 0$. We require that position $k = 0$ in each machine's processing sequence be occupied by this artificial job, $i = 0$, whose processing time is set to 0. The processing time associated with this artificial job is 0. This method enables the representation of a sequence-dependent setup time associated for the first job in each machine's processing sequence.

	Notation	Description
Sets	$i \in I_0$	Set of all jobs, including artificial job, $i = 0$, $\{0, 1, .., \lvert I \rvert\}$
	$u \in U$	Set of product types, $\{0, 1, .., \lvert U \rvert\}$
Parameters	$\lvert U \rvert$	Number of product types
	f_{juv}	Setup time to changeover from product type $u \rightarrow v$ on machine j
	q_{iu}	1 if job i is product type u, 0 otherwise
Variables	y_{jkuv}	1 if position k in the sequence of machine j requires a changeover from product type u to v.

Table 4.2. New notation for direct-positional unrelated parallel machine problem with setups (DPwS)

Formulation **(DPwS)**

$$\min \quad C_{max} + \sum_{i \in I} c_i \, D_i$$

Constraints 1, 3-9 from DP

$$x_{0j0} = 1 \quad \forall \; j \in J \tag{10}$$

$$x_{0jk} = 0 \quad \forall \; j \in J, k \in K \tag{11}$$

$$t_{j,k-1} + \sum_{i \in I} \left(P_{ij} x_{ijk} \right) + \sum_{u \in U} \sum_{v \in U} f_{juv} y_{jkuv} \leq t_{jk} \quad \forall \; j \in J, k \in K \tag{12}$$

$$\sum_{i \in I_0} (q_{iu} x_{ijk-1}) + \sum_{l \in I} (q_{lv} x_{ljk}) - 1 \leq y_{jkuv} \quad \forall \; j \in J, k \in K, u \in U, v \in U \tag{13}$$

Constraints 10 require that an artificial job, $i = 0$, occupies position, $k = 0$, on each machine j. Constraints 11 ensure that artificial job, $i = 0$, is not assigned to any other positions. Constraints 12 replace Constraints 2 from the DP model. Constraints 12 require the completion time of the job processed in position k on machine j to be after the completion time of the job in the previous position, $k-1$, plus its own processing time and the setup time of changing over from product type u to type v. Constraints 13 ensure that the setup variables, y_{jkuv}, takes a value of 1 if job i, of type u, is in position $k-1$ on machine j, and job l, of type v, is in position k; note that the extended job set I_0 is used in this constraint.

4.3.1.2 Relative-Positional Formulation

In the **R**elative-positional unrelated **P**arallel machine problem (**RP**), we define the binary decision variable, x_{ijk}, to take a value of 1 if job k immediately follows job i in the processing sequence of machine j. We introduce the binary variable y_{ij}, which takes a value of 1 when job i

is assigned to be completed by machine j. We redefine the completion time variable to represent the completion time of each job on each machine, t_{ij}.

We also define several extensions to the sets of jobs, I, and machines, J. Specifically, the subset $I^j \subseteq I$, is the set of jobs that can be processed by machine j. Similarly, we define $J^i \subseteq J$ as the subset of machines that can process job i. Both of these methods are used to reduce the size of the problem by avoiding defining constraints which are known to consider infeasible job-machine assignments.

In the RP formulation we establish another artificial job, $i = |I| + 1$. The subsets I_0^j and I_1^j are defined as $I^j \cup \{0\}$ and $I^j \cup \{|I|+1\}$, respectively. These artificial jobs are used to create sequencing precedence relationships for the first and last jobs which are scheduled on each machine, j.

	Notation	**Description**
Sets	$j \in J^i$	Set of machines that can process job i
	$i \in I^j$	Set of jobs that can be processed on machine j
	$i \in I_0^j$	Set of jobs that can be processed on machine j, with artificial job, $j = 0$
	$i \in I_1^j$	Set of jobs that can be processed on machine j, with artificial job, $i = \|I\| + 1$
Variables	t_{ij}	Completion time of job i on machine j
	x_{ijk}	1 if job k follows job i on machine j, 0 otherwise
	y_{ij}	1 if job i is assigned to machine j, 0 otherwise

Table 4.3. New and modified notation for relative-positional unrelated parallel machine problem (RP)

Formulation **(RP)**

$$\min \quad C_{max} + \sum_{i \in I} c_i \, D_i$$

s.t.

$$\begin{aligned}
t_{ij} &\leq C_{max} && \forall \quad i \in I, j \in J && (1)\\
t_{ij} + P_{kj} - M(1 - x_{ijk}) &\leq t_{kj} && \forall \quad j \in J, i \in I_0^j, k \in I^j : i \neq k && (2)\\
(s_i + P_{ij}) y_{ij} &\leq t_{ij} && \forall \quad i \in I, j \in J^i && (3)\\
d_i + D_i + M(1 - y_{ij}) &\geq t_{ij} && \forall \quad i \in I, j \in J^i && (4)\\
\sum_{\substack{k \in I_1^j: \\ i \neq k}} x_{ijk} &= y_{ij} && \forall \quad j \in J, i \in I_0^j && (5)\\
\sum_{\substack{i \in I_0^j: \\ i \neq k}} x_{ijk} &= y_{kj} && \forall \quad j \in J, k \in I_1^j && (6)\\
\sum_{j \in J} y_{ij} &= 1 && \forall \quad i \in I && (7)\\
y_{ij} &\leq A_{ij} && \forall \quad i \in I, j \in J && (8)\\
y_{ij} &= 1 && \forall \quad j \in J, i \in \{0, |I| + 1\} && (9)\\
x_{ijk} &= 0 && \forall \quad j \in J, i \in I, k = 0 && (10)\\
x_{ijk} &= 0 && \forall \quad j \in J, i = |I| + 1, k \in I && (11)\\
t_{ij} &= 0 && \forall \quad j \in J, i = 0 && (15)
\end{aligned}$$

Constraints 1 ensure that the makespan is a time after all jobs have been processed. Constraints 2 require the completion time of each job to be after the completion time of the job processed before it, plus its own processing time. If job i is processed by machine j, Constraints 3 require the completion time of each job to be after the earliest allowable start time, s_i, plus its own processing time. Constraints 4 calculate the tardiness of each job i; notice that the big M term makes the constraints relevant only when job i is processed on machine j. Constraints 5 require that each job, k, has a predecessor, i in the processing sequence of machine j (the last job precedes artificial job $|I|+1$). Constraints 6 require that each job, i, has a successor, k, in the processing sequence of machine j (the first job in the sequence succeeds artificial job 0). Constraints 7 ensure that both artificial jobs, $i = 0$ and $i = |I| + 1$, exist in the processing sequence of all machines. Constraints 8 ensure each real job, $i \in I$, is assigned exactly once. Constraints 9 ensure that jobs are only assigned to machines which are able to process them. Constraints 10 ensure that artificial job, 0,

is the first in the sequence of each machine by not allowing it to follow any other jobs Constraints 11 ensure that artificial job, $|I|+1$, is the last in the sequence of each machine by not allowing it to precedes any other jobs Constraints 12 initialize the completion time of artificial job, $i=0$, completion time, as 0, for each machine, j.

Relative-Positional, with Sequence-Dependent Setups

No new notation is required to extend the relative-positional formulation to the case with sequence-dependent setups, as shown below:

Formulation **(RPwS)**

$$\min \quad C_{max} + \sum_{i \in I} c_i \, D_i$$

Constraints 1, 3-11 from RP

$$t_{ij} + P_{kj} + \sum_{u \in U} \sum_{v \in U} q_{iu} \, q_{kv} \, f_{juv} - M(1 - x_{ijk}) \quad \leq \quad t_{kj} \quad \forall \quad j \in J, i \in I_0^j, k \in I^j : i \neq k \quad (13)$$

$$s_k + P_{kj} + \sum_{u \in U} \sum_{v \in U} q_{iu} \, q_{kv} \, f_{juv} - M(1 - x_{ijk}) \quad \leq \quad t_{kj} \quad \forall \quad j \in J, i \in I_0^j, k \in I^j : i \neq k \quad (14)$$

Constraints 13 replace Constraints 2 from the RP to consider relevant setup times. These constraints require the completion time of each job to be after the completion time of the job processed before it, plus its own processing time and changeover time. Notice that the i index in these constraints include the artificial job, $i \in I_0^j$. Constraints 14 replace Constraints 3 from RP. These constraints require the completion time of each job to be after its earliest allowable start time, plus its own processing time and changeover setup time. Note that Constraints 14 are only needed if we assume the setup can't be started until s_k. If the setup activity for job k is allowed to start before s_k, then Constraints 3 would hold.

4.3.2 Flexible Flow Shop Setting

The Flexible Flow Shop we consider is made up of two distinct stages. Both stages are represented as unrelated parallel machine settings. We refer to first-stage machines as J_1 and second-stage machines as J_2. Only a subset of the machines in the second stage require an operation from a first stage machine. We define this subset of machines as $J_2^b \subseteq J_2$. The subset $J_2^a \subseteq J_2$ is the

subset of second-stage resources that do not require a first-stage operation. The subsets J_2^a and J_2^b are mutually exclusive and collectively exhaustive of the set J_2, i.e. $J_2^a \cap J_2^b = \emptyset$ and $J_2^a \cup J_2^b = J_2$. Note that formulations presented below are capable of representing both the two-stage flow shop problem (when $J_2^a = \emptyset$, $J_2^b \neq \emptyset$) and the flexible flow shop problem (when $J_2^a \neq \emptyset$, $J_2^b \neq \emptyset$). Figure 4.2 shows an example the shop floor environment we consider:

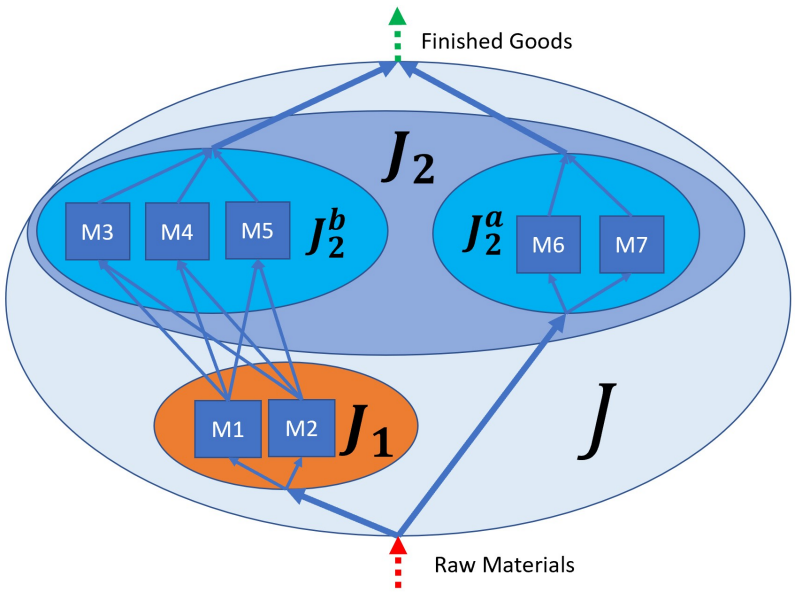


Figure 4.2. Example production system for the novel Flexible Flow Shop problem

4.3.2.1 Direct-Positional Formulation

For the **D**irect-positional **F**lexible flow shop problem (**DF**), we introduce a new variable, ρ_{ij}, to represent the earliest possible starting time of job i on any second-stage machine. This variable is the maximum value of the release date of each job, s_i, and the time job i finishes processing on a first-stage machine. In the table below, we present the new and modified notation for the sets, parameters and variables required for the DF problem:

	Notation	Description
Sets	$j \in J$	Set of all resources, $\{1..\lvert J_1\rvert + \lvert J_2^a\rvert + \lvert J_2^b\rvert\}$
	$j \in J_1$	Set of first-stage resources
	$j \in J_2$	Set of second-stage resources
	$j \in J_2^a$	Set of second-stage resources that **don't** require a first-stage operation
	$j \in J_2^b$	Set of second-stage resources that require a first-stage operation
Parameters	$\lvert J_1\rvert$	Number of first-stage resources
	$\lvert J_2^a\rvert$	Number of second-stage resources that **don't** require a first-stage operation
	$\lvert J_2^b\rvert$	Number of second-stage resources that require a first-stage operation
Variables	ρ_{ij}	Earliest possible start time of job i on second-stage machine j
	t_{jk}	Completion time of position k on machine j
	x_{ijk}	1 if job i is assigned to position k on machine j, 0 otherwise

Table 4.4. New and modified notation for direct-positional flexible flow shop problem (DF)

Formulation **(DF)**

$$\min \quad C_{max} + \sum_{i \in I} c_i \, D_i$$

s.t.

$$t_{jk} \leq C_{max} \quad \forall \; j \in J, k \in K \tag{1}$$

$$t_{jk-1} + \sum_{i \in I} P_{ij} x_{ijk} \leq t_{jk} \quad \forall \; j \in J, k \in K \tag{2}$$

$$(\rho_{ij} + P_{ij}) \, x_{ijk} \leq t_{jk} \quad \forall \; i \in I, j \in J, k \in K \tag{3}$$

$$s_i \leq \rho_{ij} \quad \forall \; i \in I, j \in J \tag{4}$$

$$t_{hk} - M(1 - x_{hjk}) \leq \rho_{ij} \quad \forall \; h \in J_1, j \in J_2^b, i \in I, k \in K \tag{5}$$

$$d_i + M(1 - x_{ijk}) + D_i \geq t_{jk} \quad \forall \; j \in J_2, i \in I, k \in K \tag{6}$$

$$\sum_{j \in J_2} \sum_{k \in K} x_{ijk} = 1 \quad \forall \; i \in I \tag{7}$$

$$\sum_{h \in J_1} \sum_{k \in K} x_{ihk} = \sum_{j \in J_2^b} \sum_{k \in K} x_{ijk} \quad \forall \; i \in I \tag{8}$$

$$\sum_{k \in K} x_{ijk} \leq A_{ij} \quad \forall \; i \in I, j \in J \tag{9}$$

$$t_{jk} = 0 \quad \forall \; j \in J, k = 0 \tag{10}$$

$$\sum_{i \in I} x_{ijk} \leq 1 \quad \forall \; j \in J, k \in K \tag{11}$$

$$\sum_{i \in I} x_{ijk+1} \leq \sum_{i \in I} x_{ijk} \quad \forall \; j \in J, k \in K : k \neq |I| \tag{12}$$

Constraints 1 ensure that the makespan is a time after all jobs have been processed. Constraints 2 require the completion time of the job processed in position k on machine j to be after the completion time of the job in the previous position, $k-1$, plus its own processing time. If job i is assigned to position k on machine j, Constraints 3 ensure that its completion time is after the job-machine pair's earliest allowed start time, ρ_{ij}, plus processing time. Constraints 4 require the earliest allowed start times of each job-machine pair, ρ_{ij}, to be greater than the earliest allowed start time of that job, s_i. If job i is assigned to a second-stage machine that requires a first-stage operation, $j \in J_2^b$, Constraints 5 ensure that ρ_{ij} is greater than the completion time of that first-stage operation for that job. Constraints 6 calculate the tardiness of each job i; notice that only second-stage machines are required for this constraint.

Constraints 7 ensure each job is assigned to exactly one position on a second-stage machine, $j \in J_2$. If job i is processed by a machine in J_2^b, Constraints 8 requires it to also be assigned to exactly one position on a first-stage resource, $h \in J_1$. Constraints 9 ensure that each job is only assigned to a machine that is capable of processing it. Constraints 10 initialize the completion time at position $k = 0$ as 0 for each resource. Constraints 11 ensure at most one job is assigned to position k on machine j. Constraints 12 require earlier positions to be filled first.

The extension of the flexible flow shop, direct-positional assignment model to include sequence-dependent setups does not require any new notation. The formulation of these additional constraints is identical to the extension of the DP model to the DPwS model, see Section 4.3.1.1.

4.3.2.2 Relative-Positional Formulation

In the following, we extend the relative-positional scheduling model from the parallel machine problem to the flexible flow shop problem. No new notation is required for the formulation of the relative-positional flexible flow shop model.

Formulation **(RF)**

$$\min \quad C_{max} + \sum_{i \in I} c_i \, D_i$$

s.t.

$$\begin{aligned}
t_{ij} &\leq C_{max} && \forall \quad i \in I, j \in J && (1)\\
t_{ij} + P_{kj} - M(1 - x_{ijk}) &\leq t_{kj} && \forall \quad j \in J, i \in I_0^j, k \in I^j : i \neq k && (2)\\
(\rho_{ij} + P_{ij}) \, y_{ij} &\leq t_{ij} && \forall \quad i \in I, j \in J^i && (3)\\
s_i &\leq \rho_{ij} && \forall \quad i \in I, j \in J && (4)\\
t_{ih} - M(1 - y_{ih}) &\leq \rho_{ij} && \forall \quad i \in I, h \in J_1, j \in J_2^b && (5)\\
d_i + D_i + M(1 - y_{ij}) &\geq t_{ij} && \forall \quad j \in J_2, i \in I^j && (6)\\
\sum_{j \in J_2} y_{ij} &= 1 && \forall \quad i \in I && (7)\\
\sum_{j \in J_1} y_{ij} &= \sum_{j \in J_2^b} y_{ij} && \forall \quad i \in I && (8)\\
y_{ij} &\leq A_{ij} && \forall \quad i \in I, j \in J && (9)\\
\sum_{i \in I_1^j \, : \, i \neq k} x_{ijk} &= y_{ij} && \forall \quad j \in J, i \in I_0^j && (10)\\
\sum_{i \in I_0^j \, : \, i \neq k} x_{ijk} &= y_{kj} && \forall \quad j \in J, k \in I_1^j && (11)\\
y_{ij} &= 1 && \forall \quad j \in J, i = \{0, |I| + 1\} && (12)\\
x_{ijk} &= 0 && \forall \quad j \in J, i \in I, k = 0 && (13)\\
x_{ijk} &= 0 && \forall \quad j \in J, i = |I| + 1, k \in I && (14)\\
t_{ij} &= 0 && \forall \quad j \in J, i = 0 && (15)
\end{aligned}$$

Constraints 1 ensure that the makespan is a time after all jobs have been processed. Constraints 2 require the completion time of each job to be after the completion time of the job processed before it, plus its own processing time. Constraints 3 require the completion time of each job to be after the earliest allowable start time, ρ_{ij}, plus its own processing time. Constraints 4 ensure ρ_{ij}, is greater than the release time of each job, s_i. If job i is processed by a machine that requires a first-stage operation ($j \in J_2^b$), Constraints 5 ensure that ρ_{ij} is greater than the completion time of that jobs first-stage operation. Constraints 6 calculate the tardiness of each job i. Constraints 7 ensure each job is assigned exactly once for the second-stage operation. If job i is processed by a machine that requires a first-stage operation ($j \in J_2^b$), Constraints 8 ensures that job is also

assigned for processing on a first-stage machine, J_1. Constraints 9 ensure that jobs are only assigned to machines which are capable of processing them. Constraints 10 require that each job, i, has a successor, k, in the processing sequence of machine j. Constraints 11 require that each job, k, has a predecessor, i, in the processing sequence of machine j. Constraint 12 require both artificial jobs, ($i = 0$ and $i = |I| + 1$), to exist in the processing sequence of all machines. Constraints 13 ensure that artificial job, 0, is the first in the sequence of each machine. Constraints 14 ensure that artificial job, $|I| + 1$, is the last in the sequence of each machine. Constraints 15 initialize the completion time of artificial job, $i = 0$, completion time, as 0, for each machine, j.

The extension of the flexible flow shop, relative-positional assignment model to include sequence-dependent setups does not require any new notations. The formulation of these additional constraints is identical to the extension of the RP model to the RPwS model, see Section 4.3.1.2.

4.4 Numerical Implementation

4.4.1 Data Generation Procedure

The size and complexity of each randomly generated problem instance is dependent on the user-defined parameters shown in Table 4.5.

Notation	Description
$\|I\|$	Number of jobs
$\|J_1\|$	Number of first-stage machines
$\|J_2^a\|$	Number of second-stage machines that DO NOT require a first-stage operation
$\|J_2^b\|$	Number of second-stage machines that require a first-stage operation machines
$\|U\|$	Number of product types
$\underline{Q}/\overline{Q}$	Minimum/Maximum size of each job
$\underline{P}/\overline{P}$	Minimum/Maximum processing rate for each machine-product type pair
$\underline{F}/\overline{F}$	Minimum/Maximum time for sequence-dependent setups between product types
$\underline{C}/\overline{C}$	Minimum/Maximum per-unit tardiness penalty
RDD	Due Date Range (must be between [0,1])
TF	Tardiness Factor (must be between [0,1])
$ELIG$	Minimum proportion of product types each machine is capable of processing, and vice versa

Table 4.5. User-defined parameters for data generation procedure of instances for evaluating direct- vs relative-positional formulations

Each order, i, is defined by two physical attributes: job size, Q_i (measured in sq. units, i.e. ft^2, m^2, or quantity) and a product type, U_i. The size of each job, Q_i, is drawn from an integer uniform distribution $\mathtt{U}[\underline{Q}, \overline{Q}]$. The product type of each job is randomly selected from the set,

$U_i = \{1, 2, ..., |U|\}$. As a reminder, the parameter q_{iu} takes a value of 1 if job i is a product of type u, 0 otherwise. The processing rate of a unit of product type u on machine j, p_{uj}, is drawn from a uniform distribution $\mathtt{U}[\underline{P}, \overline{P}]$. The processing time of each job, i, is $P_{ij} = p_{uj}\ Q_i$

The binary parameter a_{uj} describes the capability of machine j to process product type u. We implement an eligibility condition, $ELIG$, which ensures that each material can be processed by at least some proportion of the machines, and that each machine is capable of processing at least some proportion of the materials. Smaller values of $ELIG$ will result in settings with more eligibility restrictions. The procedure of defining the processing rate and eligibility parameters is shown in Algorithm 9.

The sequence-dependent setup time on machine j from a product type u to another product type v, f_{juv}, is drawn from $U[\underline{F}, \overline{F}]$. The due date, d_i, of each job is drawn from the integer uniform distribution:

$$d_i = U\left[\max\left(0, \overline{L}\Big(1 - TF - \frac{RDD}{2}\Big)\right), \overline{L}\Big(1 - TF + \frac{RDD}{2}\Big)\right]$$

where $\overline{L}$ is an estimate for the Makespan of the demand backlog, RDD is the due date range and TF is the tardiness factor (representing a rough estimate of the proportion of jobs that might be expected to be tardy in an arbitrary sequence [9, 163]). The value of $\overline{L}$ is calculated as:

$$\overline{L} = \frac{1}{|J^2|} \sum_{i \in I} \min_{j \in J_2} (P_{ij} + \overline{f_{ij}})$$

where $\overline{f_{ij}}$ is the mean of all possible sequence-dependent changeover times that could be applicable prior to processing job i on machine j. This method of due date generation is common in the literature [2, 9, 136, 154]. Note that we only consider second-stage machines in the calculation of $\overline{L}$ (we assume that this is the bottleneck operation). The per period tardiness penalty, c_i, is drawn from an integer uniform distribution $U[\underline{C}, \overline{C}]$. We allow the earliest starting time of each job to be 0, $s_i = 0$.

Algorithm 9: Data generation procedure for Direct-vs-Relative formulation comparison

Input: See Table 4.5

```
# Generate parameters relating machine-productType pairs
```

for $j \in J, u \in U$ **do**
  Let $p_{uj} \leftarrow \mathtt{U}[\underline{P}, \overline{P}]$ be the processing rate of product type u on machine j
  Let $a_{uj} \leftarrow \{0, 1\}$ be 1 if machine j can process product type u
  for $v \in U$ **do**
    Let $f_{juv} \leftarrow \mathtt{U}[\underline{F}, \overline{F}]$ be the sequenced-dependent setup time from product type $u \leftarrow v$ on machine j

```
# Ensure that ELIG conditions are valid for each resource AND product type
```

for $j \in J$ **do**
  while $\sum_{u \in U} a_{uj} < \left\lceil |J| * ELIG \right\rceil$ **do**
    Randomly switch an ineligible machine, j, to eligible for product type u

for $u \in U$ **do**
  while $\sum_{j \in J} a_{uj} < \left\lceil |U| * ELIG \right\rceil$ **do**
    Randomly switch an ineligible machine, j, to eligible for product type u

```
# Initialize Job Parameters
```

for $i \in I$ **do**
  Let $U_i \in_R U$ be the the randomly selected product type of job i
  Let $q_{iu} = 1$ for product type $u = U_i$ and $q_{iu} = 0$ for all other product types, $u \neq U_i$
  Let $Q_i \in_R \mathtt{U}[\underline{Q}, \overline{Q}]$ be the size of job i, measured in units
  Let $c_i \leftarrow \mathtt{U}[\underline{C}, \overline{C}]$ be the penalty rate for job i

```
# Generate processing times and machine-job eligibility parameters
```

for $i \in I, j \in J, u \in U$ **do**
  if *job i is product type u ($q_{iu} = 1$)* **then**
    Set $A_{ij} = a_{uj}$
    if *machine j can process product type u ($a_{uj} = 1$)* **then**
      Set processing time $P_{ij} = p_{uj} Q_i$
    else
      Set processing time $P_{ij} >> \overline{P}$ (such that this P_{ij} will never be used)

```
# Generate due dates for each Job
```

Let $\overline{L}$ be the estimated MakeSpan of the shop for the generated problem

for $i \in I$ **do**
  Let $d_i \leftarrow \mathtt{U}\left[\max\left(0, \overline{L}\left(1 - TF - \frac{RDD}{2}\right)\right), \overline{L}\left(1 - TF + \frac{RDD}{2}\right)\right]$ be the due date of job i

4.4.2 Design of Experiments

We propose the following Design of Experiments to evaluate and compare the direct-positional and relative-positional formulations presented in this chapter. We will consider several production settings, specified by the number of resources found in each machine subset, $|J_1|, |J_2^a|, |J_2^b|$. The settings we will consider are shown in Table 4.6.

		Manufacturing Environment			
Description	Notation	Single Machine	Parallel Machine	Two Stage	Flex Flow
First-Stage Resources	$\|J_1\|$	0	0	1	1
Second-Stage Resources	$\|J_2^b\|$	0	0	3	2
Multi-Stage Resources	$\|J_2^a\|$	1	3	0	1

Table 4.6. Design of Experiments - manufacturing environment settings

The production settings listed include the following scheduling problems: the single machine scheduling problem ($|J_1| = 0$, $|J_2^a| = 1$, $|J_2^b| = 0$), the unrelated parallel machine problem ($|J_1| = 0$, $|J_2^a| > 1$, $|J_2^b| = 0$), the two-stage flexible flow shop problem ($|J_1| > 0$, $|J_2^a| = 0$, $|J_2^b| > 0$), and the novel hybrid flexible flow shop problem ($|J_1| > 0$, $|J_2^a| > 0$, $|J_2^b| > 0$). For each production setting, listed in Table 4.6, we generate 10 random instances for each problem setting for each combination of the following parameter values shown in Table 4.7. All instances use shared values for the parameters shown in Table 4.8.

Description	**Parameter Grid**		
Number of Jobs	10	20	30
With Setups	True	False	-

Table 4.7. Design of Experiments - variable parameter settings

For each problem instance, we will solve the scheduling problem using both the direct- and relative-positional models. A total of 24 unique problem settings will be considered (4 production environments, 6 parameter combinations) for both the direct-positional and relative-positional scheduling formulations. 10 generated scenarios will be tested for each production setting for a total of 480 problem instances solved. It should be noted that for each generated instance of a problem setting, that both the direct- and relative-positional formulations will solve the **same** problem, enabling the direct comparison of the two formulations.

Description	Notation	Value
Number of Product Types	$\|U\|$	5
Min/Max size/qty of each job	$\underline{Q}/\overline{Q}$	1/5
Min/Max processing rates	$\underline{P}/\overline{P}$	1/5
Min/Max time for sequence-dependent setups	$\underline{F}/\overline{F}$	1/10
Min/Max per-unit tardiness penalty	$\underline{C}/\overline{C}$	1/5
Range of Due Date	RDD	0.5
Tardiness Factor	TF	0.25
Eligibility Factor	$ELIG$	0.5

Table 4.8. Design of Experiments - common parameter settings

Each problem instance will be solved using the Gurobi commercial solver, v9.5, on an AWS cloud machine with 32 GB RAM and 8 cores (x5.4xlarge machine). We limit the runtime of instances to 1 hour and evaluate the formulations based on the following criteria: number of problem instances which solve to optimality, solution time to optimality (in the case that the models solve within the time limit), optimality gap (for instances that do not solve to optimality within the time limit), solution quality (measured in the objective function value, Makespan, and value of the best lower bound) and model size/complexity (measured in the number of variables, binary variables, constraints and non-zero parameters required in each formulation).

4.4.3 Results

Throughout this section, we use shorthand notation to describe each production setting and environment in the DoE. Specifically we use a four-component acronym describing:

1. The manufacturing environment (Single Machine = S, Parallel Machine = P, Two-Stage = T, Flex Flow = F)

2. The number of jobs scheduled (10, 20, or 30)

3. The formulation implemented (Direct positional assignment = D, Relative positional assignment = R)

4. Whether or not the problem includes sequence-dependent setups (Without Setups = ”-”, With Setups = ”+”)

For example, when discussing a parallel machine manufacturing environment scheduling 20 jobs, with consideration of setups, with the direct positional assignment, we refer to the problems as: P20D+. Similarly, F30R- would describe a 30 job flex flow shop scheduling problem without setups using the relative positional assignment model. We also use the value "x" as a wildcard symbol. For example, P20xx, refers to all of the following settings: the direct-position formulation parallel machine problem with 20 jobs without setups, the direct-position formulation parallel machine problem with 20 jobs with setups, the relative-position formulation parallel machine problem with 20 jobs without setups, and the relative-position formulation parallel machine problem with 20 jobs with setups,

In the following, we first focus on the solutions related to the problem setting, P20xx, where we analyze each particular instance in detail. We then discuss the aggregate KPIs of solutions across all tested problem settings, followed by a discussion on model size/complexity. We conclude the section with a brief discussion into the behaviors of the solutions when: 1) the 1 hour time limit is increased to a 5 hour time limit, 2) a heuristic lower bound constraint is enforced.

Table 4.9 shows the outcome of each individual scenario tested for the P20xx production setting. Note that the result for the *Direct*- vs *Relative*-positional model is shown in the columns of this table. We re-emphasize that each *Trial* tested for the production setting is **identical** for the direct and relative models, allowing for an apples-to-apples comparison of the results. The KPIs in this table include the *Runtime*, in seconds, of each trial, considering a maximum time limit of 3600 seconds. A problem which is solved to 0% optimality gap prior to 3600 seconds will show the computational runtime required to reach proven optimality. The *Objective/Bound* columns show the value of the objective function and lower bound for each problem at the time of solver termination (whether it be terminated due to "completion" or due to the enforced time limit). Finally, *MIP Gap (%)* described the MIPGap achieved at the time of solver termination.

Table 4.9. Computational runtimes, Objective Function Values, Lower Bounds and MIPGaps for each test instance - Parallel Machine setting with 20 jobs

MFG Env	Num. Jobs	With Setups	KPI / Model / Trial	Runtime Direct	Runtime Relative	obj/bound Direct	obj/bound Relative	MIP Gap (%) Direct	MIP Gap (%) Relative
Parallel Machine	20	No	0	3600	3600	363/166	364/64	54.3 %	82.2 %
			1	1	3600	130/130	130/60	0.0 %	53.7 %
			2	3600	3600	167/140	171/53	16.4 %	68.5 %
			3	2255	3600	157/157	160/53	0.0 %	66.5 %
			4	3600	3600	606/213	606/65	64.8 %	89.2 %
			5	3600	3600	323/174	323/69	45.9 %	78.6 %
			6	1	3600	84/84	84/35	0.0 %	58.3 %
			7	3600	3600	427/218	427/92	48.8 %	78.4 %
			8	3600	3600	152/146	152/48	4.3 %	68.3 %
			9	3600	3600	237/166	248/53	29.9 %	78.5 %
		Yes	0	783	3600	140/140	140/48	0.0 %	65.2 %
			1	3600	3600	385/190	385/73	50.6 %	80.9 %
			2	3600	3600	528/186	528/54	64.8 %	89.6 %
			3	3600	3600	539/218	539/57	59.5 %	89.3 %
			4	3600	3600	500/192	504/59	61.6 %	88.3 %
			5	3600	3600	309/177	305/66	42.8 %	78.1 %
			6	3600	3600	175/161	175/54	8.4 %	69.3 %
			7	3600	3600	348/175	348/69	49.6 %	80.1 %
			8	3600	3600	357/170	343/54	52.2 %	84.2 %
			9	3600	3600	626/210	631/65	66.4 %	89.6 %

Table 4.10 provides an aggregated view of the results for each production setting across the 10 trials. *Trials to Optimality* shows the number of trials (out of 10) reached proven optimality within the 1 hour time limit. *Mean MIPGap (%)* and *Median MIPGap (%)* provide the mean and median MIPGap achieved across all 10 trials.

Table 4.10. Aggregation of MIPGaps achieved - Parallel Machine setting with 20 jobs

MFG Env	Num. Jobs	KPI / Model With Setups	Trials to Optimality (Out of 10) Direct	Trials to Optimality (Out of 10) Relative	Mean MIPGap (%) Direct	Mean MIPGap (%) Relative	Median MIPGap (%) Direct	Median MIPGap (%) Relative
Parallel Machine	20	No	3	0	26.4 %	72.2 %	23.2 %	73.5 %
		Yes	1	0	45.6 %	81.4 %	51.4 %	82.5 %

Figure 4.3 provides context to the MIPGap of each individual trial during the progression of the solver throughout the maximum time limit. Each path drawn in the figure shows the MIPGap of the trial over time as the solver identifies new solutions and makes cuts in the MIP Tree.

Each quadrant in the image is dedicated for a specific combination of the model formulation (direct or relative) and with or without setups. Note, that progression paths of the shared *Trial* (color coded) across each facet row are comparable. For example, when considering problems without setups, P20x-, the instances which are the most difficult to solve for P20R- are also the most difficult to solve for P20D- (i.e. trial 4). Similarly, the problems which are the least difficult are common across the formulations. A common behavior shared by all trials which are not solved to optimality is that the achieved MIPGap quickly reached an asymptote which is barely improved as time goes on.

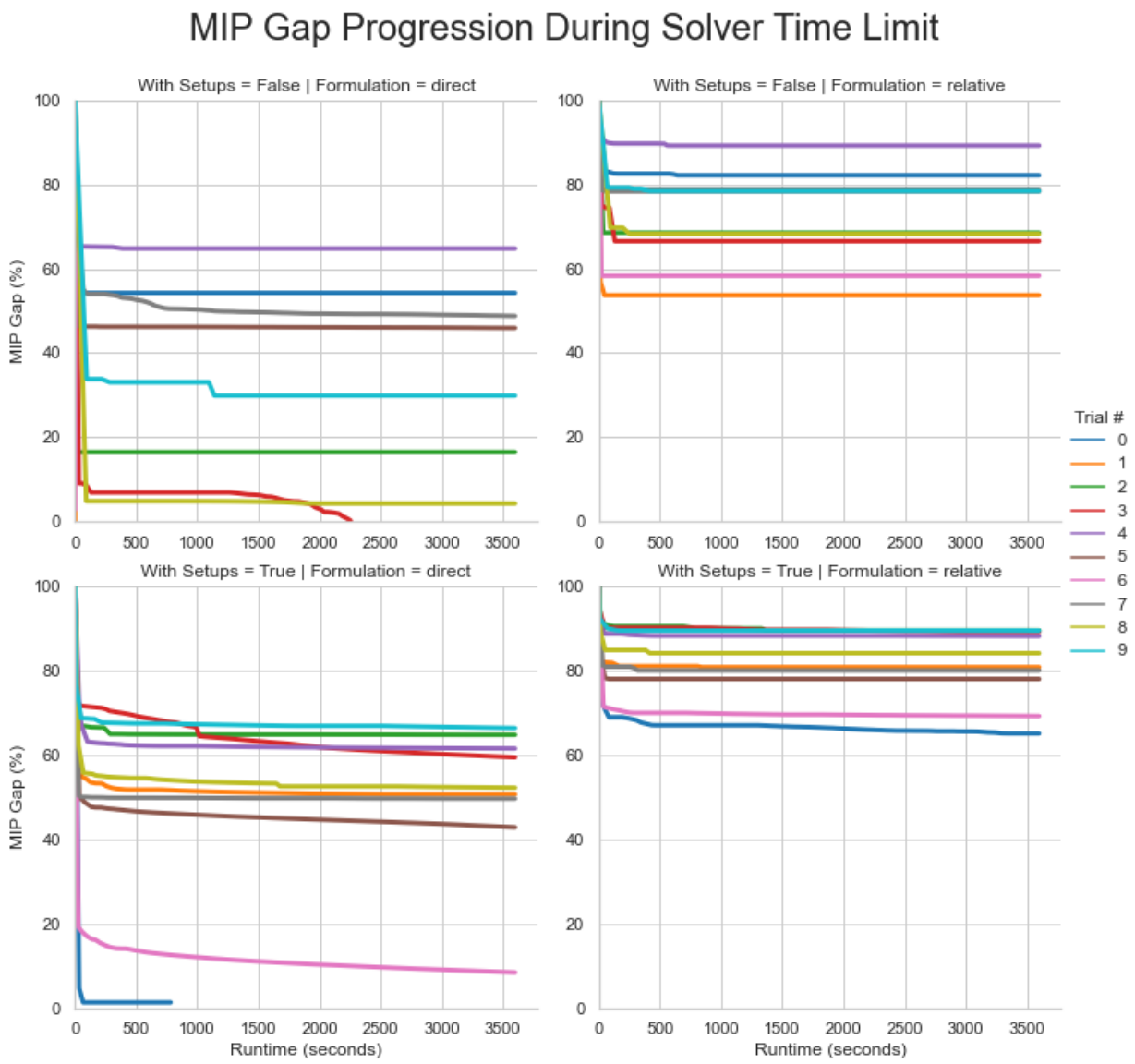


Figure 4.3. MIP Progression during the solver search of each trial - Parallel Machine setting with 20 jobs

Table 4.11 summarizes the mean and median MIPGap's achieved across all tested problem settings, similar to Table 4.10. The direct-positional formulation solved all tested problems with 10 jobs, x10Dx, to optimality, while the relative-positional formulation was only capable of this for the Single and Parallel Machine manufacturing environments. Further, while the direct-positional formulation was capable of solving several problem instances with more than 10 jobs in manufacturing environments besides the Single Machine problem, the relative positional formulation was

not. Note that the Flex Flow manufacturing environment shows achieved MIPGaps which are lower than the Two Stage manufacturing environment. It should be noted that for instances in which both the direct- and relative-positional formulations are capable of reaching optimality, the direct-positional formulation was able to do so in less time. Further, for settings in which both instances with- and without setups are solved to optimality, problems which do not consider setups are able to do so requiring less computational runtime, as expected.

Table 4.11. Aggregation of MIPGaps achieved - All settings

		KPI	Trials to Optimality (Out of 10)		Mean MIPGap (%)		Median MIPGap (%)	
		Model	Direct	Relative	Direct	Relative	Direct	Relative
MFG Env	Num. Jobs	With Setups						
Single Machine	10	No	10	10	0.0 %	0.0 %	0.0 %	0.0 %
		Yes	10	10	0.0 %	0.0 %	0.0 %	0.0 %
	20	No	10	0	0.0 %	88.4 %	0.0 %	88.1 %
		Yes	0	0	21.3 %	88.7 %	23.3 %	88.4 %
	30	No	10	0	0.0 %	91.5 %	0.0 %	91.6 %
		Yes	4	0	12.5 %	92.3 %	11.2 %	91.7 %
Parallel Machine	10	No	10	10	0.0 %	0.0 %	0.0 %	0.0 %
		Yes	10	10	0.0 %	0.0 %	0.0 %	0.0 %
	20	No	3	0	26.4 %	72.2 %	23.2 %	73.5 %
		Yes	1	0	45.6 %	81.4 %	51.4 %	82.5 %
	30	No	3	0	30.3 %	81.4 %	30.7 %	82.5 %
		Yes	0	0	41.2 %	85.8 %	45.4 %	86.9 %
Two Stage	10	No	10	3	0.0 %	34.9 %	0.0 %	38.2 %
		Yes	10	3	0.0 %	29.3 %	0.0 %	30.5 %
	20	No	0	0	38.2 %	78.0 %	37.6 %	77.0 %
		Yes	0	0	21.7 %	69.4 %	14.7 %	68.3 %
	30	No	0	0	52.7 %	89.2 %	69.5 %	92.8 %
		Yes	0	0	16.1 %	80.9 %	10.9 %	82.3 %
Flex Flow	10	No	10	8	0.0 %	10.8 %	0.0 %	0.0 %
		Yes	10	9	0.0 %	7.2 %	0.0 %	0.0 %
	20	No	2	0	17.4 %	70.0 %	10.3 %	66.7 %
		Yes	0	0	25.8 %	73.7 %	13.8 %	72.8 %
	30	No	0	0	38.3 %	84.3 %	44.5 %	87.6 %
		Yes	0	0	21.4 %	79.1 %	9.7 %	77.3 %

Figure 4.4 depicts the MIPGap achieved in each of the 10 individual problems aggregated in Table 4.11. Each quadrant in the Figure represents a different manufacturing environment, and each column within the quadrant separates problems by the number of jobs which are considered.

The formulation and consideration of setups are encoded in the color of each point, where each point represents the MIPGap achieved during the time limit. Problems which are solved to optimality are shown where the value of *MIP Gap (%)* is 0.

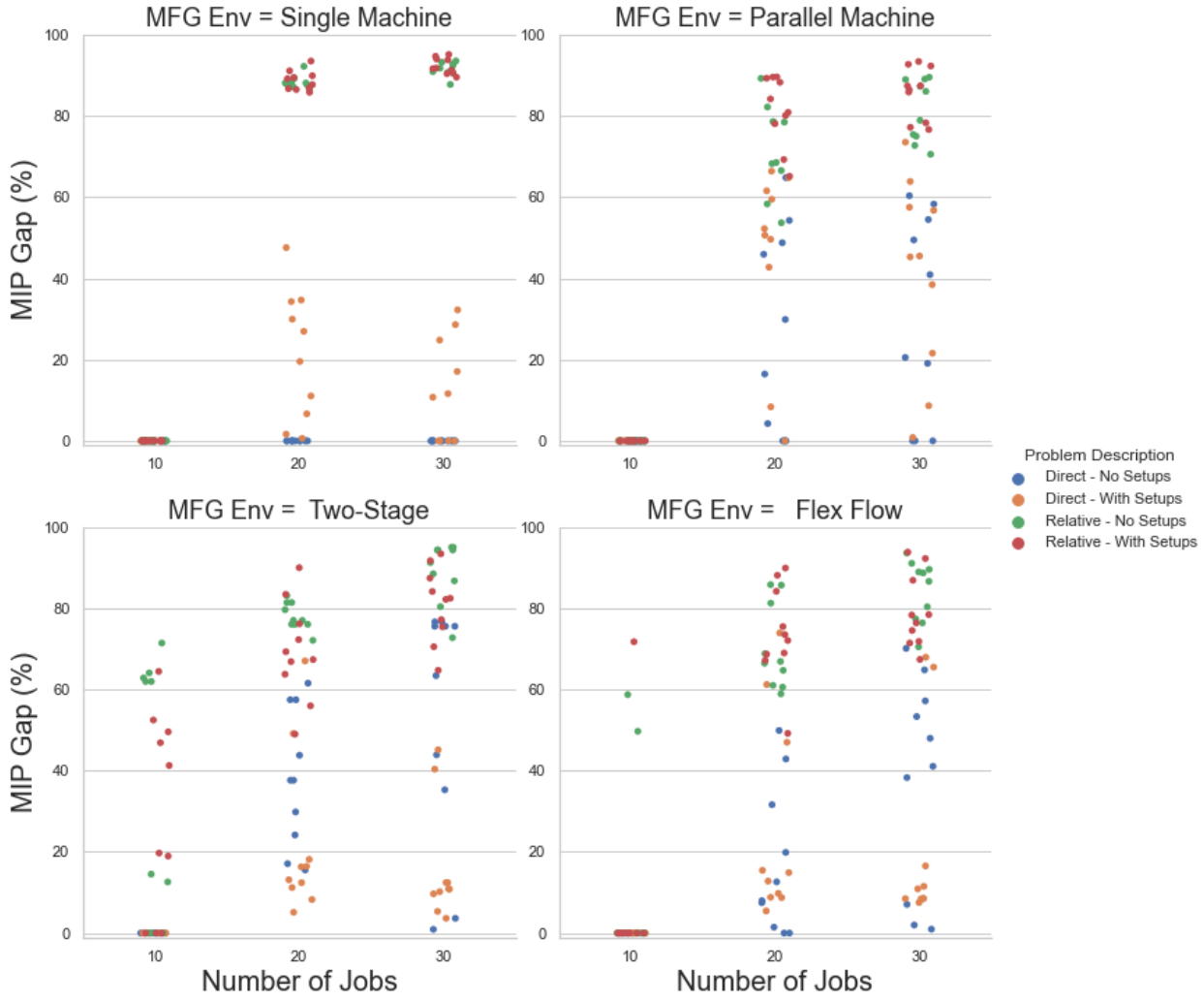


Figure 4.4. Depiction of achieved MIP Gap within hour time limit for each individual solve. Each facet represents a unique manufacturing environment, each facet column represents the number of jobs considered in each problem, and the formulation of each trial is color-coded.

Figure 4.5 and Figure 4.6 depicts the *difference* across each individual problem which are comparable (i.e. xxDx and xxRx) for the achieved MIPGap and Objective Function Value, respectively. The values shown in Figure 4.5 show the *absolute* difference of the MIP Gaps between comparable solutions, while Figure 4.6 shows the *percentage* difference of the objective function value of the relative-positional model compared to the direct-positional model, where values greater than 0%

show the relative-positional model's solution to be worse (larger objective function value) than the direct-positional model.

These results show that the direct-positional formulation outperforms the relative-positional formulation in the achieved MIPGap for all tested problem instances in all problem settings. This disparity seems to increase as the number of the jobs in each problem increases. Figure 4.6 shows that there are only several instances tested where the relative-positional formulation is able to identify a superior solution compared to the direct-positional formulation. It should be noted that the difference in the objective function values in these instances are not as extreme as the cases in which the direct-positional model outperforms the relative-positional model. For example, in the *Two Stage* manufacturing environment, the achieved objective function value for solutions of the relative-positional model are $> 80\%$ larger than the solutions found using the direct-positional formulation. From these two figures, while the objective function values from solutions achieved by the relative-positional formulation are in many cases comparable to the direct-positional solutions, the lower bounds in the problems are significantly inferior.

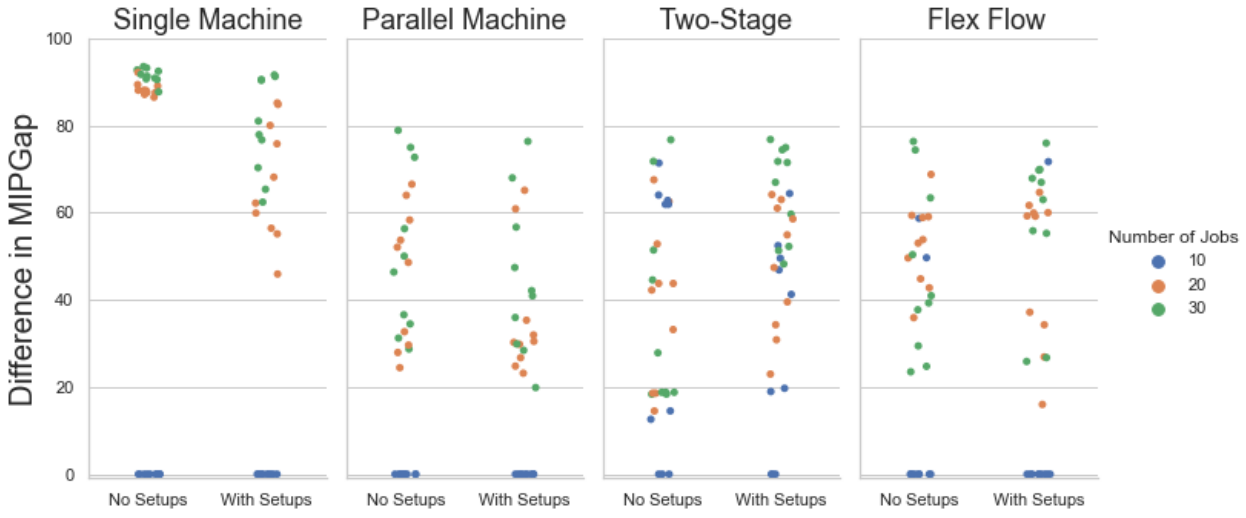


Figure 4.5. Depiction of the differences in MIP Gap within hour time limit for scenario for the direct- and relative-positional formulations, respectively

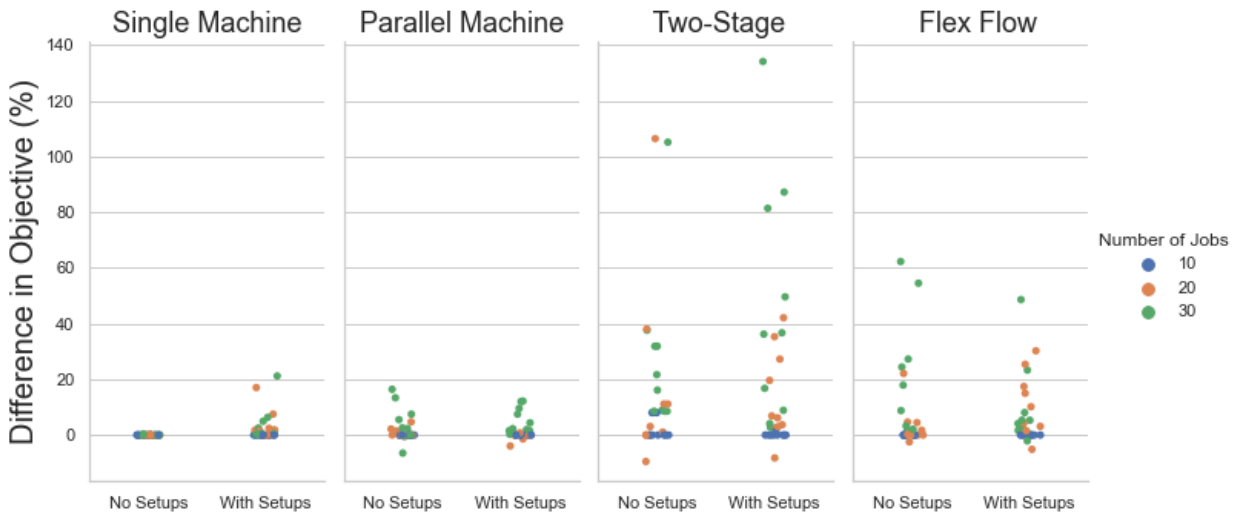


Figure 4.6. Depiction of the percentage difference in Objective Function Value within hour time limit for scenario for the direct- and relative-positional formulations, respectively

Table 4.12 provides context to the performance of the schedules generated for each problem setting, and how the quality of these solutions differ between the direct- and relative-positional formulations. As shown, the relative-positional model is able to find solutions which are nearly as good as the direct-positional model, however, the proven lower bound of these solutions is much worse. The achieved Makespan of the two formulations are also similar in all problem contexts with the direct-positional formulation strictly outperforming the relative-positional formulation, with the exception of P20R+ and T10R+, in which the relative formulation slightly outperforms the direct formulation.

Table 4.12. Aggregation of achieved Objective Function Values, Lower Bounds and MakeSpans - All settings

MFG Env	Num. Jobs	KPI Model With Setups	Obj. Func. Direct	Obj. Func. Relative	Bound Direct	Bound Relative	Makespan Direct	Makespan Relative
Single Machine	10	No	213	213	213	213	160	160
		Yes	251	251	251	251	177	177
	20	No	180	180	180	21	160	160
		Yes	215	224	162	24	164	165
	30	No	179	179	179	15	160	160
		Yes	206	214	175	16	163	164
Parallel Machine	10	No	360	360	360	360	181	181
		Yes	438	438	438	438	194	194
	20	No	265	267	160	60	161	162
		Yes	391	390	182	60	192	191
	30	No	282	293	167	46	171	173
		Yes	332	347	159	40	172	177
Two Stage	10	No	389	396	389	235	195	196
		Yes	340	340	340	227	183	182
	20	No	302	332	175	73	174	177
		Yes	194	234	130	56	146	151
	30	No	558	672	180	50	196	202
		Yes	163	251	129	38	143	148
Flex Flow	10	No	460	460	460	390	198	198
		Yes	346	346	346	319	172	172
	20	No	211	218	158	56	165	166
		Yes	234	257	136	52	153	158
	30	No	314	369	160	47	167	177
		Yes	211	234	132	36	146	154

Table 4.13 shows how the size of each MIP model grows as the number of jobs grows in each problem setting, specifically in the number of binary variables, constraints, and non-zero coefficients required to construct the model. Figures 4.7 and 4.8 represent this data in a grouped-bar chart, where each facet column represents a manufacturing environment, bars are grouped by the number of jobs in the problem setting, and the formulation considered is color-coded.

Note that the number of binary variables does NOT increase for the relative-positional model when including setups. These results also show that the number of binary variables required for both the *Two Stage* and *Flex Flow* manufacturing settings are identical, while the Flex Flow environment requires slightly less constraints and non-zeros. Clearly, the direct-positional formulation,

considering sequence-dependent setups, is the most susceptible to computational memory scalability limitations as the problem size grows.

Table 4.13. Evaluation of direct- and relative-positional assignment formulation model sizes - All settings

		KPI	Binary Variables		Constraints		Nonzeros	
MFG Env	Num. Jobs	Model With Setups	Direct	Relative	Direct	Relative	Direct	Relative
Single Machine	10	No	132	156	170	215	931	677
		Yes	564	156	641	315	6822	977
	20	No	462	506	540	625	3661	2347
		Yes	1254	506	1681	1025	25642	3547
	30	No	992	1056	1110	1235	8191	5017
		Yes	2144	1056	3121	2135	56462	7717
Parallel Machine	10	No	396	468	490	448	2793	1299
		Yes	1692	468	1903	625	20466	1845
	20	No	1386	1518	1580	1300	10983	4650
		Yes	3762	1518	5003	1903	76926	6339
	30	No	2976	3168	3270	2335	24573	8945
		Yes	6432	3168	9303	3989	169386	14055
Two Stage	10	No	528	624	860	724	4624	2278
		Yes	2256	624	2744	992	28188	3054
	20	No	1848	2024	2920	1987	18244	7320
		Yes	5016	2024	7484	3185	106168	10958
	30	No	3968	4224	6180	3580	40864	14054
		Yes	8576	4224	14224	6654	233948	23966
Flex Flow	10	No	528	624	760	700	4224	2170
		Yes	2256	624	2644	1011	27788	3128
	20	No	1848	2024	2520	2055	16644	7666
		Yes	5016	2024	7084	3179	104568	10934
	30	No	3968	4224	5280	3778	37264	15041
		Yes	8576	4224	13324	6247	230348	22350

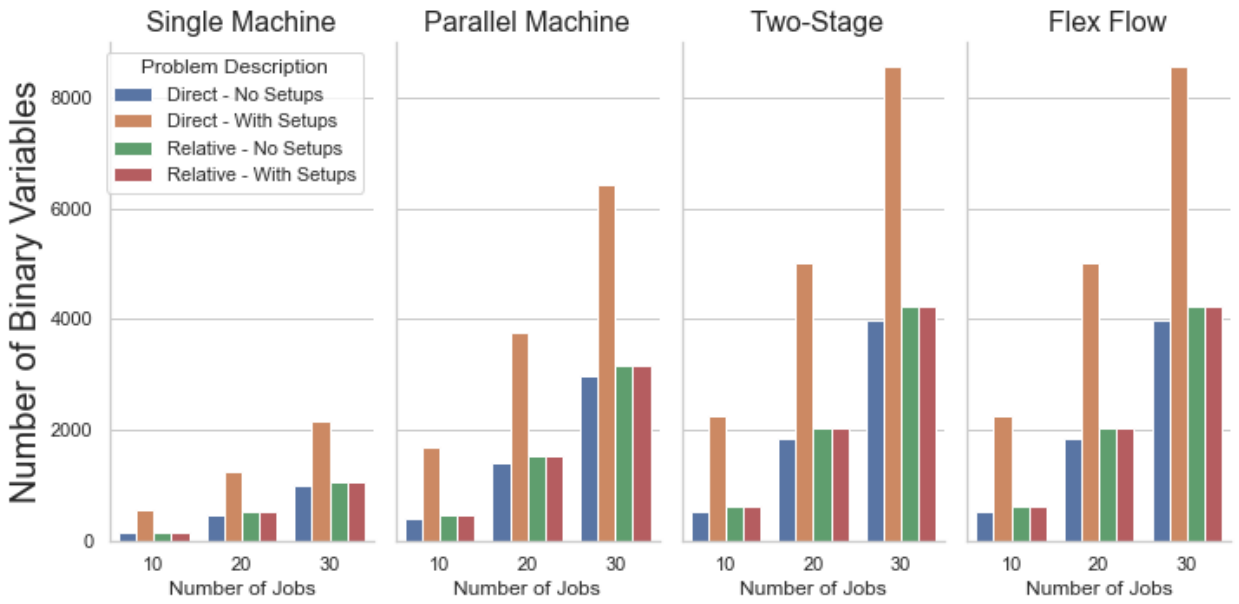


Figure 4.7. Number of binary variables required for Direct- and Relative-positional formulations

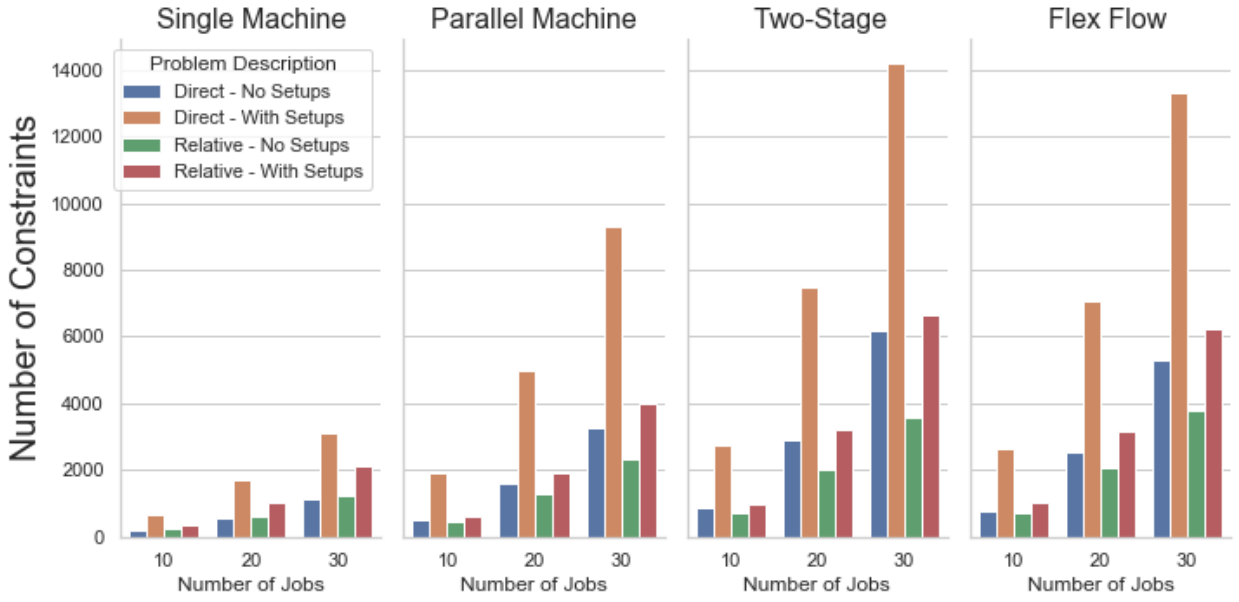


Figure 4.8. Number of constraints required for Direct- and Relative-positional formulations

4.4.3.1 Extension: Implementation with Increased Time Limits

We acknowledge that for a majority of trials tested in the DoE, a 1 hour time limit was insufficient for identifying the optimal solution for each respective problem. To explore the solver

capabilities further, tested 10 additional trials for the P20xx problem setting, and allowed a maximum time limit of 5 hours. Table 4.14 shows details of each individual run from this experiment. It is clear that the plateauing behavior of the MIP Gap persists well beyond the 1 hour time limit for all problem settings within P20xx. This is re-emphasized in Figure 4.9 which shows the solver progression for the MIPGap of each solve throughout the duration of the 5 hour time limit. Note that only 2 of the 40 tested problems required more than 1 hour, yet less than 5 hours to reach a proven optimal solution.

Table 4.14. Recorded instances of computational runtimes, achieved Objective Function Values, Lower Bounds and MIPGaps within 5 hour time limit - Parallel Machine setting with 20 jobs

			KPI	Runtime		obj/bound		MIP Gap (%)	
			Model	Direct	Relative	Direct	Relative	Direct	Relative
MFG Env	Num. Jobs	With Setups	Trial						
Parallel Machine	20	No	0	339	18000	174/174	174/59	0.0 %	65.8 %
			1	18000	18000	649/215	649/78	66.8 %	87.9 %
			2	29	18001	146/146	146/89	0.0 %	38.8 %
			3	8393	18000	183/183	183/50	0.0 %	72.7 %
			4	22	18001	116/116	116/44	0.0 %	61.5 %
			5	27	18000	143/143	143/59	0.0 %	58.3 %
			6	261	18000	145/145	145/69	0.0 %	52.4 %
			7	9	18000	156/156	156/48	0.0 %	69.2 %
			8	7340	18001	169/169	204/55	0.0 %	73.1 %
			9	18001	18000	252/164	252/64	35.1 %	74.4 %
		Yes	0	18001	18000	578/229	578/71	60.4 %	87.6 %
			1	18001	18000	370/176	373/67	52.5 %	82.0 %
			2	18001	18001	457/265	472/82	42.0 %	82.5 %
			3	18001	18000	222/172	228/52	22.4 %	77.3 %
			4	18001	18000	524/232	524/74	55.7 %	85.8 %
			5	18001	18000	383/183	383/63	52.2 %	83.4 %
			6	18001	18000	517/197	516/63	61.9 %	87.7 %
			7	18001	18000	349/171	367/53	51.0 %	85.5 %
			8	18001	18000	485/201	495/58	58.5 %	88.1 %
			9	18002	18000	228/184	228/62	19.4 %	72.6 %

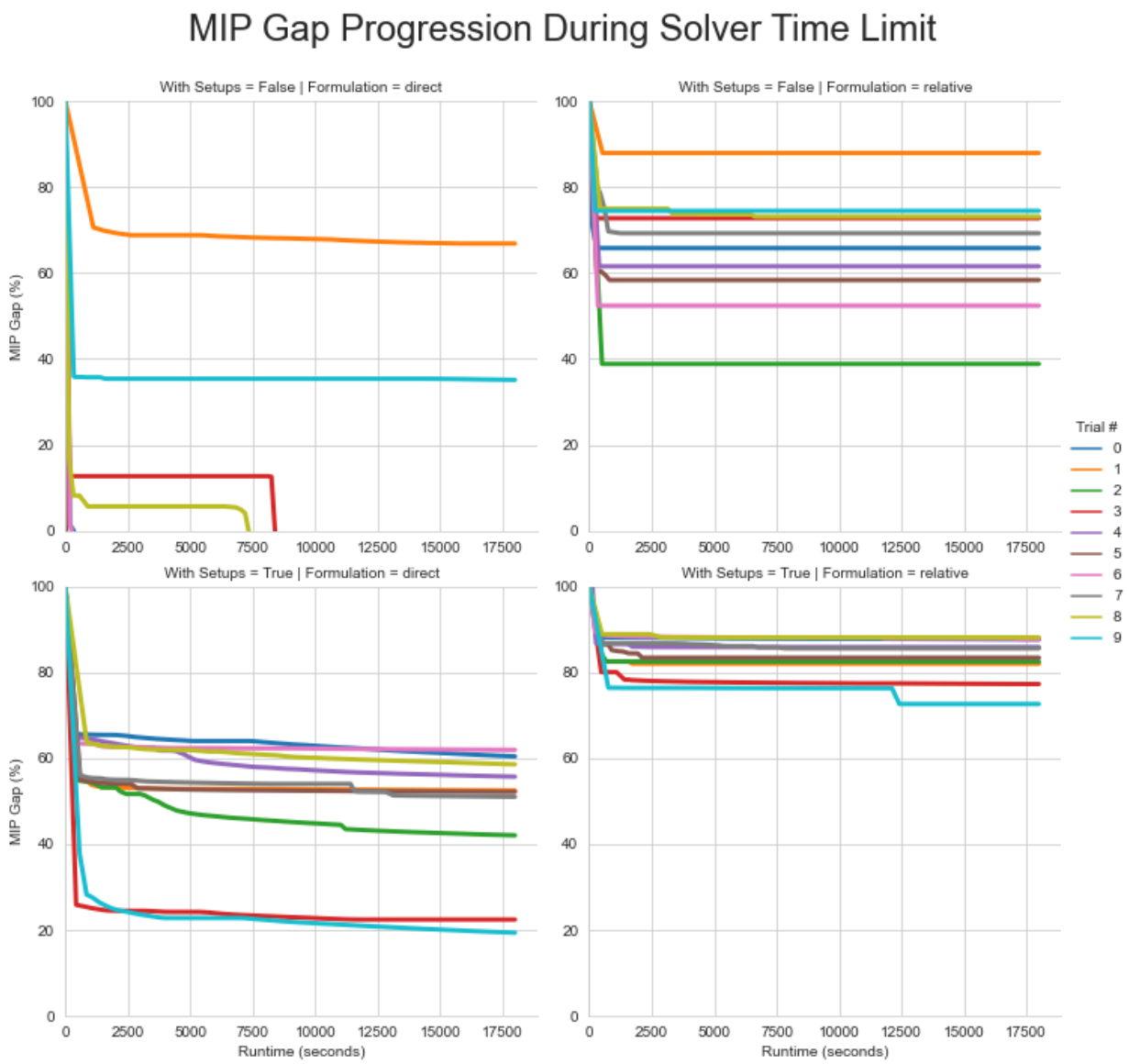


Figure 4.9. MIP Gap progression - allowing for 5 hour maximum run time limit

4.4.3.2 Extension: Implementation of Lower Bounds

In the following we evaluate: 1) the performance of the formulations presented in this chapter compared to heuristic benchmarks for finding upper and lower bounds to the tested problems, and 2) the impact of adding constraints restricting the search space in the MIP tree to solutions with values greater than the heuristic lower bound. Specifically, we present these results considering the P20xx problem setting.

The heuristic developed for calculating the lower bounds in the Pxxx problem setting is:

$$LowerBound = \frac{1}{|J|} \sum_{i \in I} \left(\min_{j \in J}(P_{ij} + f^*_{ij}) + \max\left(0, \min_{j \in J} c_i(P_{ij} - d_i)\right) \right)$$

where f^*_{ij} is the minimum sequence-dependent setup time possible for job i on machine j. This lower bound is calculated by finding: the machine able to process each job the quickest (to account for the Makespan component of the objective function) plus the minimum possible tardiness penalty for each job, assuming that each job begins processing at time 0 on the machine which is able to process it the quickest. This value is divided by the number of parallel machines which are available. Note that f^*_{ij} takes a value of 0 for all jobs and machines in problem settings which do not consider setups.

The heuristic developed for calculating the upper bounds in the Pxxx problem setting is found by successively assigning each job (in order of Earliest Due Date) to the machine which is able to *complete* it the earliest, considering sequence-dependent setup times, if applicable. The upper bound value is found as the sum of the MakeSpan plus resulting tardiness penalties from this dispatching policy. We acknowledge that both the upper and lower bounds are not optimal, can be improved, and will be a focus in future work.

Table 4.15 provides context to the quality of solutions found by the direct- and relative-positional formulations compared to the heuristic upper and lower bounds for each tested problem instance. Note that both the direct- and relative-positional formulations are capable of finding superior solutions compared to the heuristic dispatching policy. However, in many of the instances shown, the best lower bound found using relative-positional formulation is inferior to the heuristic lower bound.

Table 4.15. Evaluation of achieved Objective Function Values and Lower Bounds of MIP solver and Heuristic Upper/Lower Bounds - Parallel Machine setting with 20 jobs

With Setups	Model KPI / Trial	Heuristic Bounds Upper	Heuristic Bounds Lower	Direct Obj. Func.	Direct Lower Bound	Relative Obj. Func.	Relative Lower Bound
No	0	219	143	146	146	172	79
	1	2186	126	1043	194	1061	73
	2	144	106	137	137	137	70
	3	211	94	126	126	126	44
	4	277	134	144	144	144	72
	5	1667	118	508	268	508	64
	6	196	81	124	124	124	39
	7	181	128	148	148	148	59
	8	1560	131	569	191	573	61
	9	231	105	144	144	147	60
Yes	0	320	69	160	160	171	53
	1	530	81	169	169	169	58
	2	2970	95	551	230	532	70
	3	1800	83	642	199	670	68
	4	1296	107	602	200	633	56
	5	4030	77	1169	214	1190	66
	6	506	66	164	145	164	45
	7	778	92	284	151	284	53
	8	2023	94	350	180	350	72
	9	1241	119	398	184	392	58

Table 4.16 illustrates the impacts of implementing a constraint to the formulations which restricts the search space to solutions which have an objective function value greater than the heuristic lower bound. Specifically, the following modifications are made to the formulation:

Formulation Modifications

$$\begin{aligned} \min \quad & Z \\ \text{s.t.} \quad & \text{All Original Constraints} \\ & C_{max} + \sum_{i \in I} c_i \, D_i = Z \quad \forall \; j \in J \quad (LB1) \\ & Z \geq LB \quad \forall \; j \in J \quad (LB2) \end{aligned}$$

The column, *Uses LB*, signifies whether or not the formulation modifications are implemented to each respective formulation. We observe that incorporating the heuristic lower bound results in an improvement to the MIPGap realized for the relative-positional formulation, especially when

sequence-dependent setups are NOT considered. We also note that the instances solved to optimality by the direct-positional formulation, not considering sequence-dependent setups, tend to take longer when the lower bound constraints are included. However, these runtimes are slightly improved when setups are considered. Figure 4.10 depicts the progression of the MIPGap of each problem instance during the course of the time limit.

Table 4.16. Impact of implementing heuristic lower bound constraint: Recorded instances of achieved Objective Function Values, Lower Bounds and MIPGaps within 1 hour time limit - Parallel Machine setting with 20 jobs

With Setups	Trial	KPI / Model / Uses LB	MIP Gap (%) Direct	MIP Gap (%) Relative	Runtime Direct	Runtime Relative	obj/bound Direct	obj/bound Relative
No	0	N	0.0 %	54.3 %	4	3600	145/145	171/78
		Y	0.0 %	1.8 %	7	3600	145/145	145/142
	1	N	81.4 %	93.1 %	3600	3600	1043/194	1060/73
		Y	81.2 %	89.3 %	3600	3600	1042/196	1176/125
	2	N	0.0 %	48.5 %	7	3600	136/136	136/70
		Y	0.0 %	22.2 %	9	3600	136/136	136/106
	3	N	0.0 %	65.0 %	8	3600	125/125	125/44
		Y	0.0 %	26.8 %	8	3600	125/125	129/94
	4	N	0.0 %	50.0 %	18	3600	143/143	143/71
		Y	0.0 %	6.7 %	79	3600	143/143	143/134
	5	N	47.4 %	87.5 %	3600	3600	508/267	508/63
		Y	48.3 %	76.9 %	3601	3600	508/263	508/117
	6	N	0.0 %	68.4 %	2	3600	123/123	123/39
		Y	0.0 %	34.2 %	5	3600	123/123	123/81
	7	N	0.0 %	60.0 %	20	3600	147/147	147/59
		Y	0.0 %	13.3 %	54	3600	147/147	147/128
	8	N	66.5 %	89.3 %	3600	3600	569/190	573/61
		Y	67.0 %	77.6 %	3600	3600	578/190	584/131
	9	N	0.0 %	59.2 %	7	3600	144/144	146/60
		Y	0.0 %	27.1 %	14	3600	144/144	143/105
Yes	0	N	0.0 %	68.7 %	2362	3600	160/160	170/53
		Y	0.0 %	58.0 %	1853	3600	159/159	164/69
	1	N	0.0 %	65.7 %	1496	3600	169/169	169/58
		Y	0.0 %	52.3 %	331	3600	169/169	169/80
	2	N	58.2 %	86.8 %	3600	3600	550/230	532/70
		Y	56.4 %	83.2 %	3600	3600	550/240	565/95
	3	N	69.0 %	89.8 %	3600	3600	642/198	670/68
		Y	69.3 %	87.3 %	3601	3600	642/197	652/82
	4	N	66.8 %	91.2 %	3600	3600	602/200	633/55
		Y	66.9 %	82.2 %	3600	3600	604/199	602/107
	5	N	81.7 %	94.5 %	3601	3600	1169/214	1190/66
		Y	82.4 %	93.5 %	3600	3600	1195/209	1186/77
	6	N	11.5 %	72.6 %	3600	3600	164/145	164/45
		Y	2.2 %	59.7 %	3600	3600	164/160	164/66
	7	N	46.8 %	81.4 %	3600	3600	283/151	283/52
		Y	47.2 %	67.4 %	3600	3600	283/149	283/92
	8	N	48.5 %	79.5 %	3600	3600	350/180	350/71
		Y	49.3 %	74.5 %	3600	3600	350/177	368/93
	9	N	53.7 %	85.2 %	3600	3600	398/184	392/57
		Y	53.8 %	70.5 %	3600	3600	398/184	402/118

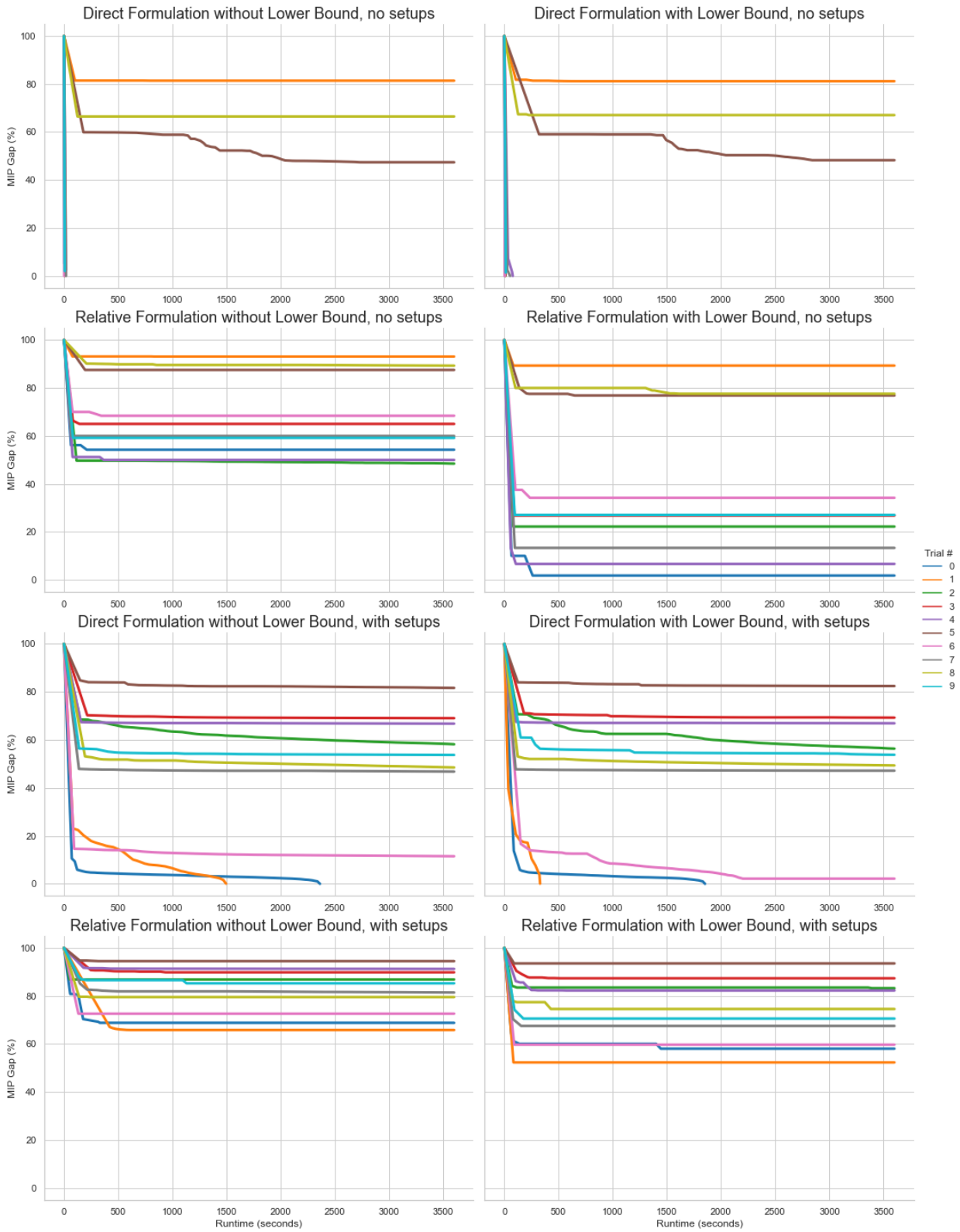


Figure 4.10. MIP Gap progression - implementing lower bound constraint

4.5 Conclusion

In this chapter, we present and compare several formulations for the unrelated parallel machine and flexible flow shop scheduling problems. The scheduling problem we consider is motivated by the production system of Artaic - Innovative Mosaic, a mosaic manufacturer located in Boston, MA. This Make-To-Order firm employs several generations of automated tile-placing robots to fabricate mass-customized murals. A subset of these robots require an additional preprocessing operation which prepares the tiles which make up the murals for the robots.

The scheduling formulations we develop are capable of considering sequence-dependent setups and vary in the manner in which scheduling decisions are represented. These representations include a direct-positional assignment formulation and a relative-positional assignment formulation, with an objective of minimizing shop Makespan and weighted tardiness. We also consider a novel representation of the flexible flow Shop problem where only a subset of the resources in the second production phase require an operation in the first phase of production.

Based on the results of our computational experiments, implemented using the Gurobi v9.5.0 solver with a Python interface, we conclude that the direct-positional formulation dominates the relative-positional formulation in all tested problem settings. While the relative-positional formulation seems to be a promising alternative for problems considering sequence-dependent setups, considering that the number of variables and constraints does not increase compared to a problem with no setups, the direct-positional formulation still provides superior solutions. However, limitations exist in the formulations presented, as they are only capable of identifying optimal solutions for smaller problems. Future work in this research stream will be dedicated to the enhancement of the formulations by improving the performance of the models in identifying lower bounds, i.e. by tightening the value of the big-M parameter, defining valid inequality constraints, and calibrating the parameters of Gurobi's solver.

CHAPTER 5

SCHEDULING WITH CONTINGENT DEMAND

5.1 Introduction

To remain competitive, Make-To-Order firms must be able to provide short lead-times, low prices and customizable products which suit the needs of their customers. Make-to-Order firms typically operate in a Direct-to-Consumer/Business-to-Customer environment, where the demand generation process is initiated when a potential customer submits an enquiry for a specific product. The firm follows up by submitting a quotation to that customer detailing their capabilities, including the price and lead-time they are able to offer. Assuming the potential customer goes out for quotation to multiple competing MTO firms, the firm which offers the most attractive combination of price, leadtime, quality and incentives (such as compensation for late deliveries), will be awarded the demand.

A trade-off exists in the quotation decisions a firm makes in response to new customer requests. If a firm quotes too aggressively (i.e. low prices and short lead times), they are likely to win a lot of bids, but will be unlikely to satisfy the terms offered in the quote, resulting in tardiness penalties. If they bid too conservatively, they are unlikely to win any bids, resulting in the loss of potential revenues and the under utilization of their *perishable* resources. The firm must strike a balance between competitiveness and attainability when submitting quotes to new customers.

The time between the delivery of a quotation and the agreement on order terms between the firm and the customer is referred to as the *tendering period*, a consideration which has received shockingly little attention in scheduling literature. Customer demand during this period is considered a *contingent demand* and acts as a critical source of uncertainty for MTO firms. During this time it is unknown whether the firm will need to reserve resource capacities for the order, making the estimation of product lead times difficult. When a new customer enquiry arrives, the firm may have several contingent orders in their backlog. This presents a challenge for the firm, as they must offer lead time quotations to these new customers without full information of their current backlog.

Ideally, the duration of the tendering period should be reduced as much as possible. However, due to the nature of MTO manufacturing, in which products tend to be expensive and require extensive planning, the tendering period may be long.

The purpose of this chapter is to develop a generalized decision support system (DSS) framework for the scheduling problem, subject to contingent demand, which underlies the application of various Revenue Management problems, such as: the Order Acceptance (OA), the Dynamic Pricing (DP), the Due Date Setting (DDS), and the Simultaneous Pricing and Due Date Setting problems. The execution of these Revenue Management strategies depends on the underlying scheduling problem comprising the production of the firm's realized demand. Sub-optimal scheduling practices can lead to inaccurate estimates of expected lead times and as a result, loss of revenue due to either overly aggressive or conservative quotations. We consider the problem where a decision-maker is tasked with scheduling a set of jobs, with the requirement that they must *commit/freeze* the most immediate portion of the schedule, say the next two weeks, to allow other decision-makers in the firm to act with certainty when making decisions such as human resource planning, preventative maintenance planning, raw material procurement, delivery planning, etc.

Although the existing literature has addressed applications encompassing disruptive uncertainties such as job cancellations or the introduction of new orders, the work dedicated to specifically applying scheduling under the uncertainty of contingent demand is limited. Our objective is to identify a framework which finds a balance in the trade-off of solution fidelity and computational efficiency. As will be described, the scheduling problem subject to contingent demand quickly becomes intractable as the number of contingent jobs increases, so finding a framework which can handle large-sized problems will provide a beneficial contribution to the existing literature.

The rest of the paper is organized as follows. In Section 5.2, we provide a review on scheduling under uncertainty and the considerations of contingent demand. In Section 5.3, we provide the problem statement and present a simple example to illustrate the impacts which contingent demands may have on scheduling decisions. In Section 5.4, we develop several stochastic programs for solving the scheduling problem with contingent demand, each with varying levels of solution fidelity. In Section 5.5, we present a computational analysis of the aforementioned formulations and provide our results. In Section 5.6, we propose further directions of research, and in Section 5.7 we conclude the chapter.

5.2 Literature Review

Much of the existing literature in machine scheduling addresses static scheduling, where all resource and job characteristics are assumed to be deterministic and known prior to generating a schedule [178]. Solutions obtained using these "classic" scheduling approaches quickly become infeasible and cause production disturbances in the face of disruptions or deviations from their deterministic expectations [64, 103]. Practitioners often view the ignorance of uncertainty and the dynamic elements of the scheduling process as a major source of the gap between scheduling theory and practice [148]. This shortcoming has begun to be addressed in recent decades and lately has been of primary concern in the scheduling literature.

The purpose of this literature review is two-fold: First we provide a high-level overview of the modeling dimensions associated with defining the framework of the scheduling problem we consider. Second, we provide review on the limited existing literature which directly address the impacts of contingent demand. Supplemental reviews are also presented as Appendices in this dissertation.

5.2.1 Overview of Scheduling Under Uncertainty

Many classification schemes have been developed to address the types of uncertainties faced by manufacturing and service firms. Pistikopoulos [131] developed a classification scheme where uncertainties are labeled as model-inherent, process-inherent, external, or disruptive uncertainties.

Uncertainties derived from contingent demands can be classified as a disruptive uncertainty. Disruptive uncertainties can be broken into two subcategories: resource-related disruptions and job-related disruptions. Examples of resource-related disruptions include random machine breakdowns, the sudden unavailability of suppliers, or a sudden loss of personnel. Job-related disruptions include dynamic release dates, order cancellations, due-date changes, arrival of urgent jobs, and changes in job specifications which have a direct impact on the processing times.

The representation of these uncertainties within a decision-making framework acts as a defining characteristic in scheduling models. Generally, uncertainties are represented using scenario-based methods or probabilistic-based methods. In scenario-based representations, uncertainty is modeled in a finite set of *scenarios*, which can be defined through the discretization of continuous probability distributions or using simulation-based methods. For example, Basset et al. [16] consider uncertainties in processing rates, process yields, and resource availability using a discrete set of

scenarios that were generated using a Monte Carlo simulation. They determine a schedule for each instance and aggregate the solutions across all scenarios to identify optimal scheduling policies. Scenario-based representations provide a straightforward way to incorporate uncertainty. However, the problem inevitably grows in complexity as the size of the considered scenario set grows.

Approaches that address scheduling under uncertainty are classified as reactive, proactive or as hybrid of the two. Reactive scheduling approaches (also referred to as dynamic or online scheduling) revise an existing schedule in real-time throughout its execution as unexpected events occur. Research objectives in this field concern the identification of optimal rescheduling policies and methods given a scheduling environment. The fundamental research questions are: *when* should a reschedule be triggered? and *how* should the reschedule be implemented?

It should be noted that strictly reactive scheduling approaches do not directly consider the uncertainties while generating schedules, but rather identify the optimal policies which should be implemented while reacting to the realization of those uncertainties [148]. While this does accommodate for considerable flexibility in the schedule to compensate for unforeseen system disturbances, these strategies lack the global perspective provided by proactive scheduling approaches. For this reason, these approaches are typically reserved for problem settings characterized by extremely disruptive and/or frequently realized uncertainties.

Proactive scheduling considers future disruptions while generating schedules in an effort to hedge against potential disruptions and parametric uncertainties. This approach is more synonymous with robust or stochastic optimization and makes use of historical data and forecasting techniques to derive scheduling decisions [102]. The implementation of proactive scheduling approaches typically occurs during the generation of a *preschedule*. Although a preschedule is unlikely to be executed, it serves as the basis for planning supporting activities, establishing commitments with employees, suppliers and customers, and making other revenue management decisions such as order acceptance/rejection or new customer pricing and/or due date quotation [102]. Scheduling objectives in proactive approaches incorporate the preferences of the firm and can be classified based on the following distinctions: preference of schedule stability vs performance, objectives based on reward vs regret, and attitude towards risk (i.e. risk-neutral vs risk-averse).

We refer the reader to Appendix D for a review of these solution approach methodologies and representations of proactive scheduling objectives.

5.2.2 Scheduling and Revenue Management with Contingent Demand

The impacts of contingent demands remain an under-researched topic in the field of scheduling. We have found that all studies which directly address the uncertainty of contingent demand, consider it in the context of Revenue Management problems in which models are developed to make decisions such as order acceptance/rejection or determining price and/or lead time quotations.

Contingent demands exist between the time a firm submits a quotation to a potential customer and the time which the customer makes a decision on whether to accept or reject the offer. There also exists the potential for a negotiation period in which the customer returns to the firm with requests for adjustments to the terms described in the quotation. Each contingent order has 2 potential outcomes: it is either accepted by the customer and becomes realized as actual demand, or it is rejected, resulting in a total of 2^N possible demand scenarios, where N is the number of contingent jobs. Each demand scenario will also result in a different optimal production schedule.

It is possible to estimate the probability that a contingent demand will become realized as a function of the quotation offered to the customer using historical data. For example, the *probability of acceptance* of a job, i, can be estimated using statistical models, such as Berkson's S-shaped binary choice logit model [22]:

$$a_i = \left[1 + \beta_0 \ \exp\Big(-\sum_{j=1}^{A} \beta_j x_j\Big)\right]^{-1}$$

where A is a vector of the distinguishing attributes of a bid, x are the quoted value of each attribute, and β coefficients are estimated empirically from the firms historical bids. The probability of any scenario, $s \in S$, occurring in a contingent demand backlog can then be calculated as the product of the probabilities of each job's outcome in that scenario:

$$q_s = \prod_{i \in I^C} \Big(a_i \ b_{is} + \overline{a_i} \ (1 - b_{is})\Big)$$

where q_s is the probability of scenario s occurring, b_{is} is a parameter taking a value of 1 if job i is accepted by the customer in scenario s, I^C is the set of contingent jobs, and the probability that a job i will be rejected is: $\overline{a_i} = 1 - a_i$.

Easton and Moodie [50] are cited as the first to consider the implications of contingent demand. They consider a price and lead time quotation model, leveraging a form of Berkson's logit model to maximize the expected revenue of a new arriving customer. The authors consider a single machine environment where all jobs are processed in the order in which they arrive as confirmed orders. The expected contribution of each job enquiry is calculated as the expected revenue of the offered price (considering the probability of acceptance) minus the expected tardiness of the order as a result of offering an overly competitive leadtime. The authors follow up this research in Moodie [118] and Moodie and Bobrowski [119] to evaluate various strategies for negotiating price and due dates.

Watanapa and Techanitisawad [187] extend this bidding model to account for multiple customer classes, each with unique parameters for willingness to pay and sensitivity to lead time. They consider a time-critical rush order class and a class for regular orders. In their model, they implement an Earliest Due Date sequencing heuristic for rush orders and a FCFS sequencing rule for regular orders. Watanapa and Techanitisawad [186] extend this work by proposing a genetic algorithm to search for near-optimal sequences of the demand backlog. They conduct a simulation study on a finite horizon and find a significant benefit in the average marginal revenue associated with the order processing sequence. Liu and Liu [104] extend the models of Watanapa and Techanitisawad by optimizing the scheduling sequence to minimize the weighted lateness of the backlog. The authors develop and evaluate several meta-heuristics to solve the sequential bidding and scheduling problems, including a proposed hybrid meta-heuristic method that combines simulated annealing and tabu search. The authors consider a single machine scheduling problem with an objective of minimizing the total weighted earliness and tardiness cost.

Zhou and Zhou [202] also extend the quoting of Easton and Moodie to include service level constraints. The authors find that a consideration of service level constraints (defining the percentage of orders which will be delivered before the quoted lead time) has a beneficial impact on the long-term expected marginal revenue. Cakravastia et al. [34, 35] extend the quoting model of Easton of Moodie to consider multiple resources and multiple jobs per order, each potentially with unique routings. The authors develop a manufacturing planning model which is responsible for determining the shop schedule and lot sizes and conclude that the incorporation of the manufacturing planning model can help the firm avoid overly optimistic bidding.

Lu and Liu [109] consider the Dynamic Pricing and Scheduling problem. They explore the benefits of price and scheduling coordination compared to partial coordination and blanket quoting schemes and find that coordination benefits the manufacturer for a wide range of parameter settings. The authors consider a single machine manufacturing environment and apply Smith's [160] *Shortest Weighted Processing Time* dispatching rule for scenario-based scheduling decisions. The authors also investigate the validity of assuming that the firm only has several price points available for quotation. They show that the expected profit loss of having limited discrete set of price points decreases rapidly as the number of price points increases.

Ebadian et al. [51] consider an extension of the Simultaneous Pricing, Due-Date Setting and Scheduling Problem (SDPSP) subject to contingent demand to consider a set of suppliers and subcontractors that are able to provide raw materials and outsourcing capacities to complete the accepted workload. Baykasoglu et al. [17] develop a model for the SPDSP that incorporates novel price increment/reduction functions to represent back-and-forth negotiations between the firm and customers. This model is applied to the real-life problem of a bridal gown company and is shown to provide superior results compared to models without a negotiations mechanism.

Sujan et al. [168] also consider an extension of the SPDSP that enables a multi-round negotiation phase to the quoting process. The authors compare two strategies, namely opportunistic (aggressive) and liberalistic (conservative), to react to customer counteroffers. The authors define terms such as *aspiration level* and *limit level* to characterize the firm's willingness to offer competitive pricing and lead times (represented as a value curve depicting the trade-off between offered price and lead time), as well as the term *customer preferred line* to represent customer counter offers. The authors demonstrate through a numerical analysis the working mechanisms of the proposed model. In Sujan et al. [167], the SPDSP is extended to provide multiple quotations for each arriving quote at the same aspiration level. This logic is beneficial when customers have a tendency towards price- or time-sensitivity which is unknown to the firm. Sujan [134], extends the SPDSP further to generate quotes for multiple orders simultaneously, rather than sequentially. The author shows that dealing with quotes simultaneously increases the expected contribution and probability of acceptance of new orders. Throughout the development of these models, the authors computational experiments are limited in scope, considering less than 5 work centers and less than 10 jobs (3 of which are contingent).

This study contributes to the literature in the following ways: 1) we evaluate the impacts of contingent demand strictly in the context of scheduling, which to our knowledge, has never been addressed before in the literature, 2) we develop several novel unrelated-parallel machine scheduling formulations which promote schedule stability by fixing the most immediate portions of the schedule across all demand backlog scenarios, including a novel time-indexed formulation which considers a heterogeneous discretization of the planning horizon to enable the implementation of large-scale contingent demand scheduling and revenue management decisions, 3) we provide a computational comparison of the various formulations to solve larger-scale scheduling problems under contingent demand.

5.3 Problem Description

In this section, we present the contingent demand scheduling problem that is at the heart of the generation of robust price and due date quotes under contingent demand. In this problem, $|I|$ jobs must be processed for a single operation by one of $|J|$ unrelated parallel machines, $j \in J$.

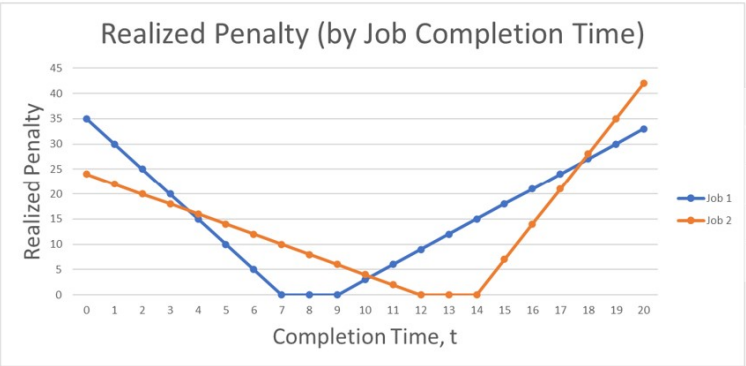
Each job, $i \in I$, belongs to either the accepted job set, I^A, or the contingent job set, I^C. All jobs in the accepted job set, $i \in I^A$, have been already committed to and are known to be a part of the eventual realized schedule. The contingent jobs, $i \in I^C$, describe outstanding orders which have unresolved quotations offered to customers. Uncertainty in this problem derives from the observed realization of contingent demand from the job set, I^C. The possible demand scenarios, $s \in S$, are defined by the jobs which exist as actual demand. The integer parameter, $|S|$, defines the number of scenarios which are considered. The binary parameter, b_{is}, describes whether or not job i is accepted by the customer in scenario s. Only jobs that are confirmed in scenario s, i.e. $b_{is} = 1$, are scheduled. Each contingent job is also characterized by its probability of acceptance a_i. Each scenario, s, has a probability of occurrence, q_s, calculated as the joint probability of the contingent jobs that are accepted/rejected in that scenario.

Each job which is accepted in a scenario must be scheduled to one of k positions in the schedule. The set of all scheduling positions is $k \in K$. As mentioned, we consider a requirement in which the most immediate portion of the schedule $k \in K^F \subseteq K$, must be frozen, i.e. the same across all scenarios $s \in S$. The integer parameter, f, defines the number of positions in the schedule which must be fixed on each resource. Only accepted jobs, $i \in I^A$, are allowed to be assigned to this

portion of the schedule, as contingent jobs are not certain to be realized. We refer to the portion of the schedule which is not fixed as K^{NF}, where $K^F \cap K^{NF} = \emptyset$ and $K^F \cup K^{NF} = K$.

The scenario-independent binary decision variable x_{ijk}, defined for each $k \in K^F$ and $i \in I^A$, takes the value 1, if job i is scheduled on resource j in the frozen position k, 0 otherwise. The scenario-dependent binary decision variable y^s_{ijk}, defined for each $k \in K^{NF}$ and $i \in I$, takes a the value 1, if job i is scheduled in position k on resource j in scenario s.

The binary parameters, A_{ij}, take a value of 1 if job i can be processed by resource j (0 otherwise), and P_{ij} is the processing time of job i on resource j. Each job i is also characterized by the following parameters: a due date, d_i, an earliest acceptable delivery date, e_i, a per-period holding cost, h_i, for jobs completed prior to their earliest acceptable due date and a per-period tardiness cost, c_i, for jobs completed after their due date. This penalty structure is referred to as a *due-window* penalty structure, and is representative of the problem faced by our industrial partner.



Parameter	Notation	Job 1	Job 2
Earliest Delivery Date	e_i	7	12
Desired Delivery Date	d_i	9	14
Holding Cost	h_i	5	2
Lateness Penalty	c_i	3	7

Figure 5.1. Example of due-window penalty structure

The objective of the scheduler is to minimize the expected costs (sum earliness and tardiness penalties) associated with the backlog across a set of contingent demand scenarios. There is no preemption, job splitting, or outsourcing, and processing rates are deterministic. There may be a setup time associated with a job, independent of its size, but since jobs are assigned to a single resource without preemption, this time can simply be included to its overall processing time. It is assumed that setups are not sequence-dependent, which differs from the problem considered in Chapter 4.

We refer the reader to Appendix E for a presentation and discussion of an example problem for the setting we describe above. In this Appendix we show that: 1) scheduling decisions when

optimizing expected costs are sensitive to the probabilities associated with each scenario, 2) job-resource eligibility constraints can have major influence in this problem and 3) different attitudes towards risk (neutral vs averse) can result in unique optimal scheduling decisions.

5.4 Formulations

In the following, we present five formulations for this problem, differentiating in the definition of the planning horizon and scheduling assignment decision variables. First, we adapt the direct-positional scheduling formulation from Chapter 4 to incorporate the scenario-based problem required for considering contingent demands. We then present formulations for four cases of a time-indexed representation of the scheduling problem. Each time-indexed formulation differs in the discretization method of time used when defining the scheduling horizon. First, we formulate the case in which the discretization of time is limited such that each job requires at least one time period to be completed by any resource (small-bucket). Then we formulate the case where each time period is at least as large as the maximum processing time required by any resource to complete any job (big-bucket). This formulation allows for multiple jobs to be assigned and completed within the same time period. A special case in which all jobs are identical, allowing for the discretization of time equal to the processing time on each resource, is also presented. Finally, we formulate a hybrid-bucket model which combines aspects of both the small- and big-bucket formulations. Specifically, this formulation considers small-buckets for the frozen portion of the schedule and big-buckets for the scenario-dependent portion of the schedule.

5.4.1 Direct-Positional Assignment

In the direct-positional assignment formulation (D), decision variables, x_{ijk} and y^s_{ijk}, take a value of 1 if job i is assigned in the kth position in the processing sequence of machine j in scenario s (if applicable). The frozen portion of the schedule, K^F, is defined such that the first f jobs assigned in the sequence of each resource must be the same across all considered scenarios. The variables, t_{jk} and T^s_{jk} are the completion time of position k on resource j (for fixed and non-fixed assignments respectively). The realized earliness associated with these completion times are calculated as ϵ_i and E^s_i. Similarly, δ_i and D^s_i, are calculated as the tardiness associated with completion times later than the desired due date d_i. Realized holding costs and tardiness penalties are calculated as the

product of these earliness/tardiness variables and the per-period holding cost rate, h_i, and tardiness rates, c_i, respectively. Note that all variables associated with the frozen portion of the schedule are scenario-independent. The formulation of the direct-positional contingent backlog scheduling model is presented below:

	Notation	**Description**
Sets	$j \in J$	Set of machines, $\{1..\|J\|\}$
	$i \in I$	Set of all jobs, $\{1..\|I\|\}$
	$i \in I^A$	Set of accepted jobs, $I^A \subseteq J$
	$i \in I^C$	Set of contingent jobs, $I^C \subseteq I$
	$k \in K$	Set of all positions, $\{0..\|I\|\}$
	$k \in K^F$	Set of fixed positions, $\{0..f-1\}$
	$k \in K^{NF}$	Set of non-fixed positions, $\{f..\|I\|\}$
	$s \in S$	Set of scenarios, $\{1..\|S\|\}$
Parameters	f	Number of fixed positions
	$\|I\|$	Number of jobs
	$\|J\|$	Number of machines
	$\|S\|$	Number of scenarios
	M	A really large number
	d_i	Desired delivery date of job i
	e_i	Earliest acceptable delivery date of job i
	h_i	Per-period holding cost of job i
	c_i	Per-period tardiness penalty of job i
	A_{ij}	1 if resource j can process job i, 0 otherwise
	P_{ij}	Processing time of job i on resource j
	b_{is}	1 if job i is a confirmed order in scenario s, 0 otherwise
Variables	$t_{jk},\ T^s_{jk}$	Completion time of position k on resource j, in scenario s (if applicable)
	$\delta_i,\ D^s_i$	Tardiness associated with job i, in scenario s (if applicable)
	$\epsilon_i,\ E^s_i$	Earliness associated with job i, in scenario s (if applicable)
	x_{ijk}	1 if job i is assigned to position k on resource j, 0 otherwise
	y^s_{ijk}	1 if job i is assigned to position k on resource j in scenario s, 0 otherwise

Table 5.1. Notation for the direct-positional formulation

Formulation **(D)**

$$\min \quad \sum_{i \in I^A} \Big(h_i \, \epsilon_i + c_i \, \delta_i \Big) + \sum_{s \in S} \sum_{i \in I} q_s \Big(h_i \, E_i^s + c_i \, D_i^s \Big)$$

s.t.

$$\sum_{j \in J} \Big(\sum_{k \in K^F} x_{ijk} + \sum_{k \in K^{NF}} y_{ijk}^s \Big) = 1 \qquad \forall \quad i \in I^A, s \in S \tag{1}$$

$$\sum_{j \in J} \sum_{k \in K^{NF}} y_{ijk}^s = b_{is} \qquad \forall \quad i \in I^C, s \in S \tag{2}$$

$$\sum_{i \in I^A} x_{ijk} + \sum_{i \in I} y_{ijk}^s \leq 1 \qquad \forall \quad j \in J, k \in K, s \in S \tag{3}$$

$$\sum_{k \in K^F} x_{ijk} + \sum_{k \in K^{NF}} y_{ijk}^s \leq A_{ij} \qquad \forall \quad i \in I, j \in J, s \in S \tag{4}$$

$$t_{jk} = 0 \qquad \forall \quad j \in J, k = 0 \tag{5}$$

$$t_{j,k-1} + \sum_{i \in I^A} P_{ij} \, x_{ijk} \leq t_{jk} \qquad \forall \quad j \in J, k \in K^F \tag{6}$$

$$t_{jk} + \sum_{i \in I} P_{ij} \, y_{ijk}^s \leq T_{j,k+1}^s \qquad \forall \quad j \in J, k = f, s \in S \tag{7}$$

$$T_{j,k-1}^s + \sum_{i \in I} P_{ij} \, y_{ijk}^s \leq T_{jk}^s \qquad \forall \quad j \in J, k \in K^{NF}, s \in S : k > f \tag{8}$$

$$t_{jk} - d_i - M\big(1 - x_{ijk}\big) \leq \delta_i \qquad \forall \quad i \in I^A, j \in J, k \in K^F \tag{9}$$

$$T_{jk}^s - d_i - M\big(1 - y_{ijk}^s\big) \leq D_i^s \qquad \forall \quad i \in I, j \in J, k \in K^{NF}, s \in S \tag{10}$$

$$e_i - t_{jk} - M\big(1 - x_{ijk}\big) \leq \epsilon_i \qquad \forall \quad i \in I^A, j \in J, k \in K^F \tag{11}$$

$$e_i - T_{jk}^s - M\big(1 - y_{ijk}^s\big) \leq E_i^s \qquad \forall \quad i \in I, j \in J, k \in K^{NF}, s \in S \tag{12}$$

$$\sum_{i \in I^A} x_{ijk+1} \leq \sum_{i \in I^A} x_{ijk} \qquad \forall \quad j \in J, k \in K^F \tag{13}$$

$$\sum_{i \in I} y_{ij,k+1}^s \leq \sum_{i \in I} y_{ijk}^s \qquad \forall \quad j \in J, k \in K^{NF}, s \in S \tag{14}$$

The objective function minimizes the expected penalty associated with the schedule across the scenario set. This value is calculated as the sum of penalties associated with fixed-positions assignments plus the penalties associated with non-fixed positional assignments. Note that the scenario-dependent penalties are weighted by the probability that that scenario will occur, q_s.

Constraints 1 require each accepted job to be scheduled exactly once for each scenario. Constraints 2 require each contingent job to be scheduled exactly once for only the scenarios in which it exists in the backlog, i.e. when $b_{is} = 1$. Constraints 3 ensure that at most one job can be

assigned to each position. Constraints 4 ensure that jobs can only be assigned to machines which are capable of processing them. Constraints 5 initialize the completion time at position $k = 0$ on each resource as 0. Constraints 6-8 require the completion time of the job processed in position k on resource j to be after the completion time of the job in the previous position, plus its own processing time. Constraints 6 consider all fixed positions, $k \in K^F$. Constraints 7 consider the first non-fixed position, $k = f$, (which follows a fixed position) for each scenario. Constraints 8 consider all other non-fixed positions, $k \in K^{NF} : k \neq f$. Constraints 9-10 calculate the tardiness of jobs for fixed and non-fixed assignments; notice the big M term makes the constraints relevant only when job i is assigned to position k on resource j in scenario s (if applicable). Constraints 11-12 calculate the earliness of fixed and non-fixed assignments. Constraints 13-14 require earlier positions to be filled first.

The direct-positional formulation provides the most accurate representation of the schedule, considering the representation of the planning horizon in continuous time. However, the completion times and penalties associated with each assignment are dependent variables and must be calculated on-the-fly. This leads to excessive computational expense which may not be justified considering the use-case, i.e. as the underlying framework of a dynamic and/or due date quotation model. In practice, including our motivational use-case, it is common to provide customers with lead time quotes on the scale of weeks, not to the minute, as this continuous-time direct-positional model would provide. For this reason, a time-indexed scheduling formulation may be beneficial, as a trade-off in solution fidelity may be justified to achieve reasonable solution times for much larger problem instances. Unlike the formulation above, the discretization of time allows for the parameterization of the completion time and penalties associated with each scheduling decision, beforehand in preprocessing. Further, while the direct-positional formulation requires constraints including a big-M term (Constraints 9-12), the following time-indexed formulations do not.

5.4.2 Small-Bucket Time-Indexed Assignment

The small-bucket (S) time-indexed formulation redefines x_{ijk} and y^s_{ijk} as the assignment of job i to be *started* on resource j at the beginning of period k. The definition of the frozen period of the planning horizon, K^F, is redefined to freeze the first f **periods** in the planning horizon, in

contrast to the direct-positional formulation, which freezes the first f **positions** in the sequence of each resource.

The processing time required for machine j to complete job i is P_{ij} periods. Processing times are represented as an integer value, making it is possible to quickly calculate the completion time associated with any scheduling assignment. For example, if a job is assigned to be started at position k, the completion time of that job will be $t_{ijk} = k + P_{ij}$. Notice that t_{ijk} is no longer scenario-dependent, as it was in the direct-positional formulation. The associated penalty of a job completed at time t_{ijk} can also be calculated as a parameter:

$$R_{ijk} = h_i \ \max\left(0, e_i - t_{ijk}\right) + c_i \ \max\left(0, t_{ijk} - d_i\right)$$

The parameterization of both the completion time, t, and observed penalty, R, of any assignment decision allows for the removal of all dependent variables which are required to solve the direct-positional scheduling problem, e.g. tardiness of a job, D_i^s. Another benefit of this parameterization is that this allows for the ability to capture complex cost structures associated with any assignment. Although outside of the scope of this dissertation, the calculation of R_{ijk} can be enhanced to consider additional, sequence-independent setup costs (associated with resource j and job i), overtime costs (associated with period k), and more complex penalty structures (i.e. tiered), all without requiring any changes to the formulation.

To ensure that each resource processes at most 1 job at any time, we must define constraints that restrict the assignment of any other jobs for each period between the starting time, k and completion time $k + P_{ij} - 1$, of a job, i, which is assigned to that resource, j. To enable this, we define two new parameter sets, U_{ijk} and V_{ijk}. U_{ijk} defines the number of fixed scheduling periods, $k \in K^F$, which are consumed by job i on machine j when started in period k, and V_{ijk} defines the number of non-fixed scheduling periods, $k \in K^{NF}$, which are consumed by that assignment.

U_{ijk} is calculated as the minimum of: 1) the processing time of job i on machine j, P_{ij}, and 2) the number of frozen periods between the assignment period k and beginning of the non-frozen portion of the schedule, $f - k$. V_{ijk} is calculated as the maximum of: 1) 0, and 2) the sum of the starting time, k, plus the processing time, P_{ij} minus the time period in which the non-frozen period begins, f. Consider an example in which a job, i, requires 10 processing periods on a machine, j,

but is scheduled to begin processing 3 periods before the end of the frozen period of the schedule, $k = f - 3$. We define U_{ijk} and V_{ijk} in order to ensure: 1) allocation constraints for the frozen period are only created for periods which are within the fixed period, i.e. $k \in K^F$, and 2) that no other jobs are scheduled to begin in the first 7 periods of the non-frozen period on that machine. In this example, $P_{ij} = 10$, $U_{ijk} = 3$, and $V_{ijk} = 7$.

	Notation	Description
Sets	$k \in K$	Set of all time periods, $\{0..\|K\|-1\}$
	$k \in K^F$	Set of all fixed periods, $\{0..f-1\}$
	$k \in K^{NF}$	Set of all non-fixed periods, $\{f..\|K\|\}$
Parameters	f	Number of fixed time periods, $\|K^F\|$
	$\|K\|$	Number of time periods
	P_{ij}	Time periods required for resource j to process job i
	t_{ijk}	Completion time of job i on resource j when started in time period k
	R_{ijk}	Penalty associated with scheduling job i to be started by resource j at time k
	U_{ijk}	Number of fixed periods allocated to processing job i on resource j
	V_{ijk}	Number of non-fixed periods allocated to processing job i on resource j
Variables	x_{ijk}	1 if job i is assigned to begin processing on resource j in period k
	y^s_{ijk}	1 if job i is assigned to begin processing on resource j in period k in scenario s

Table 5.2. New and modified notation for small-bucket time-indexed formulation

Formulation **(S)**

$$\min \quad \sum_{j\in J}\left(\sum_{k\in K^F}\sum_{i\in I^A} R_{ijk}\, x_{ijk} + \sum_{k\in K^{NF}}\sum_{s\in S}\sum_{i\in I} q_s\, R_{ijk}\, y^s_{ijk}\right)$$

s.t.

$$\sum_{j\in J}\Big(\sum_{k\in K^F} x_{ijk} + \sum_{k\in K^{NF}} y^s_{ijk}\Big) = 1 \quad \forall \quad i\in I^A, s\in S \tag{1}$$

$$\sum_{j\in J}\sum_{k\in K^{NF}} y^s_{ijk} = b_{is} \quad \forall \quad i\in I^C, s\in S \tag{2}$$

$$\sum_{k\in K^F} x_{ijk} + \sum_{k\in K^{NF}} y^s_{ijk} \le A_{ij} \quad \forall \quad i\in I, j\in J, s\in S \tag{3}$$

$$1 - \sum_{l\in I^A: l\neq i} x_{lj,k+u} \ge x_{ijk} \quad \forall \quad i\in I^A, j\in J, k\in K^F, u\in\{0..U_{ijk}-1\} \tag{4}$$

$$1 - \sum_{l\in I: l\neq i} y^s_{lj,f+u} \ge x_{ijk} \quad \forall \quad i\in I^A, j\in J, k\in K^F, s\in S, u\in\{0..V_{ijk}-1\} \tag{5}$$

$$1 - \sum_{l\in I: l\neq i} y^s_{lj,k+u} \ge y^s_{ijk} \quad \forall \quad i\in I, j\in J, k\in K^{NF}, s\in S, u\in\{0..P_{ij}-1\} \tag{6}$$

Constraints 1 ensure that each accepted job is only scheduled to one position. Constraints 2 ensure that each contingent job is scheduled once for each scenario, if it exists in that scenario, i.e. when $b_{is} = 1$. Constraints 3 ensure that jobs can only be assigned to machines which are capable of processing them. Constraints 4-6 ensure that machines process no more than one job in any period. Constraints 4 considers fixed periods. Constraints 5 consider non-fixed periods which may be allocated to complete jobs started during the fixed portion of the schedule. Constraints 6 consider non-fixed periods.

5.4.3 Big-Bucket Time-Indexed Assignment

Although the time-indexed formulation allows for the parameterization of the dependent variables, in contrast with the direct-positional formulation, the number of binary decision variables required to represent the problem increases drastically. We present the big-bucket time-index formulation (B) to account for this issue. In the big-bucket formulation, periods are of long duration, i.e. a day or week, such that multiple jobs can be completed within a single period. Let C_{jk} be the processing time available on machine j in period k. This parameter also allows the flexibility for the capacity of each period to be unequal, i.e. in the case that a planned maintenance reduces the availability of a machine in period k. Capacity constraints are introduced such that the sum of the processing times of all the jobs assigned to any machine are less than the capacity of that machine in period k. The schedule here is less accurate as it does not specify the precise sequence of jobs processed within this larger time period they are assigned to.

The decision variables x_{ijk} and y^s_{ijk} are now defined as the assignment of job i to be *entirely processed* by machine j during period k, in scenario s (if applicable). As a result, the completion time of any assignment is no longer job-dependent, and is only indexed in j and k: t_{jk}. It is assumed that all jobs assigned to be completed during period k will be delivered at the end of the period, i.e. the end of the day/week. This assumption is justified so long as the time granularity of customer deadlines (e_i, d_i) are measured in increments at least as large as the scheduling period length. The penalty associated with a scheduling assignment is now calculated as:

$$R_{ijk} = h_i \ \max(0, e_i - t_{jk}) + c_i \ \max(0, t_{jk} - d_i)$$

The big-bucket formulation is presented below:

	Notation	Description
Parameters	C_{jk}	Available processing time of resource j in time period k
Variables	x_{ijk}	1 if job i is to be processed on resource j during time period k
	y^s_{ijk}	1 if job i is to be processed on resource j during time period k in scenario s

Table 5.3. New and modified notation for the big-bucket time-indexed formulation

Formulation **(B)**

$$\min \quad \sum_{j\in J}\left(\sum_{k\in K^F}\sum_{i\in I^A} R_{ijk}\, x_{ijk} + \sum_{k\in K^{NF}}\sum_{s\in S}\sum_{i\in I} q_s\, R_{ijk}\, y^s_{ijk}\right)$$

s.t.

$$\sum_{j\in J}\Big(\sum_{k\in K^F} x_{ijk} + \sum_{k\in K^{NF}} y^s_{ijk}\Big) = 1 \quad \forall \quad i\in I^A, s\in S \tag{1}$$

$$\sum_{j\in J}\sum_{k\in K^{NF}} y^s_{ijk} = b_{is} \quad \forall \quad i\in I^C, s\in S \tag{2}$$

$$\sum_{k\in K^F} x_{ijk} + \sum_{k\in K^{NF}} y^s_{ijk} \le A_{ij} \quad \forall \quad i\in I, j\in J, s\in S \tag{3}$$

$$\sum_{i\in I^A} P_{ij}\, x_{ijk} \le C_{jk} \quad \forall \quad j\in J, k\in K^F \tag{4}$$

$$\sum_{i\in I} P_{ij}\, y^s_{ijk} \le C_{jk} \quad \forall \quad j\in J, k\in K^{NF}, s\in S \tag{5}$$

Constraints 1 ensure that each accepted job is only scheduled to one position. Constraints 2 ensure that each contingent job is scheduled once for each scenario, if it exists in that scenario, i.e. $b_{is} = 1$. Constraints 3 ensure that jobs can only be assigned to machines which are capable of processing them. Constraints 4 and 5 limit the work that can be assigned to a machine on a time slot so that capacity is not exceeded. Note that Constraints 4 only considers jobs from the accepted job set, and is not dependant on the scenario $s \in S$.

The big-bucket model works at a higher level of aggregation, which is attractive because it significantly reduces the number of variables. This leads to shorter solution times while providing sufficient solution fidelity to allow for revenue management decision making. The appropriate size of the time bin may be a shift, a day, or a week depending on the magnitude of processing times and the accuracy requirements in the particular application.

In settings where job sizes vary wildly, large jobs may need to be broken into smaller units that fit into the bin size. If a job i is broken into n_i smaller jobs, $i_1, i_2, \ldots, i_{n_i}$, then the penalties in

the objective function will only be associated with the completion time of the last job, i_n, and new constraints must be added to ensure that the earlier jobs are completed before that last one.

5.4.4 Identical Job Time-Indexed Assignment

The identical-job formulation (I) is a special case of the time-indexed formulation when all jobs have the same resource requirements (i.e., $P_{ij} = P_j \quad \forall \quad i \in I$). This results in a much simpler formulation that only includes assignment constraints. In this simplified formulation, because each machine will be able to process all jobs $i \in I$ at the same rate, the length of time which constitutes a period on each machine can be standardized to be equal to P_j. Therefore, in this special case formulation of the time-indexed scheduling problem, x_{ijk} and y^s_{ijk} are defined as the assignment of job i to be *fully processed* during period k on machine j.

Formulation **(I)**

$$\min \quad \sum_{j\in J}\left(\sum_{k\in K^F}\sum_{i\in I^A} R_{ijk}\, x_{ijk} + \sum_{k\in K^{NF}}\sum_{s\in S}\sum_{i\in I} q_s\, R_{ijk}\, y^s_{ijk}\right)$$

s.t.

$$\sum_{j\in J}\Big(\sum_{k\in K^F} x_{ijk} + \sum_{k\in K^{NF}} y^s_{ijk}\Big) = 1 \quad \forall \quad i \in I^A, s \in S \tag{1}$$

$$\sum_{j\in J}\sum_{k\in K^{NF}} y^s_{ijk} = b_{is} \quad \forall \quad i \in I^C, s \in S \tag{2}$$

$$\sum_{i\in I^A} x_{ijk} + \sum_{i\in I} y^s_{ijk} \leq 1 \quad \forall \quad j \in J, k \in K, s \in S \tag{3}$$

$$\sum_{k\in K^F} x_{ijk} + \sum_{k\in K^{NF}} y^s_{ijk} \leq A_{ij} \quad \forall \quad i \in I, j \in J, s \in S \tag{4}$$

As a reminder, Constraints 1 ensure that each accepted job is only scheduled to one position. Constraints 2 ensure that each contingent job is scheduled once for each scenario, if it exists in that scenario, i.e. $b_{is} = 1$. Constraints 3 limit the number of jobs that can be assigned to each position. Constraints 4 ensure that jobs can only be assigned to machines which are capable of processing them.

Consider an application in which job splitting is allowed. Each job $i \in I$ can be split into equally sized units of *work content* such that the processing requirements of any unit of work content is identical. Further, in the case in which partial deliveries is allowed, the completion time of any assignment can be calculated as $t_{jk} = k\, P_j$, similar to when all jobs are of the same size.

5.4.5 Hybrid-Bucket Time-Indexed Assignment

As mentioned, a trade-off exists between the small- and big-bucket time-indexed formulations presented above. The small-bucket formulation provides high fidelity scheduling solutions, but is subject to extensive memory requirements, while the opposite stands for the big-bucket formulation. However, to a scheduling manager, it may be ideal to incorporate the benefits of both formulations. Consider the use case in which the decision maker must decide a schedule for only the upcoming two weeks, but would like to make robust decisions by accounting for all of the potential contingent jobs. We present a hybrid-bucket formulation which encodes the frozen portion of the schedule, K^F, using the small-bucket time-indexed formulation while representing the non-frozen schedule, K^{NF}, using the big-bucket formulation. The resulting solution will provide a detailed schedule for the frozen period while providing a loose production plan for the remaining backlog. This implementation will also reduce the number of binary decision variables required to represent the solution.

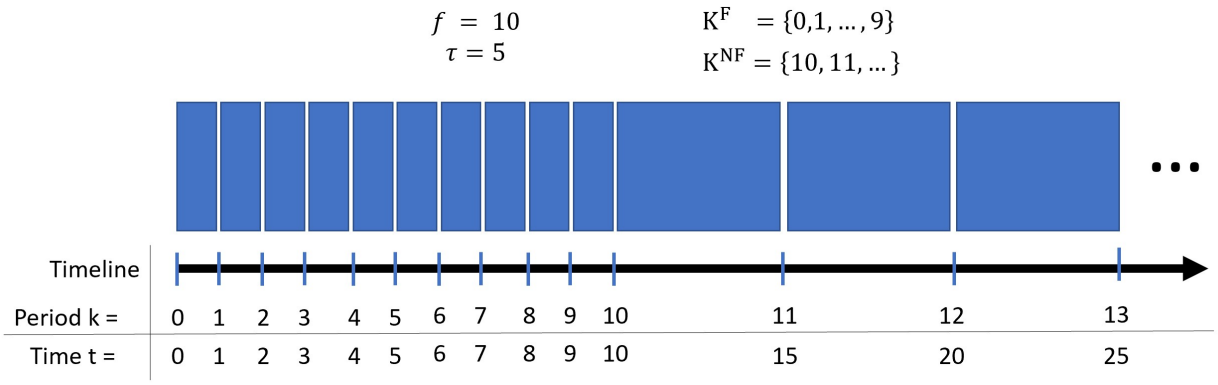


Figure 5.2. Example of hybrid-bucket time-index scheduling timeline

The planning horizon, $k \in K$, is now composed of periods of heterogeneous duration. A proportional factor, τ, relating the duration of small- and big-buckets is required, such that each bucket in K^{NF} can be divided into τ small-bucket periods. For example, if each $k \in K^F$ represents one hour, each $k \in K^{NF}$ represents one week, and each week is 40 working hours, then a conversion factor $\tau = 40\ \dfrac{hours}{week}$ is used to generate the planning horizon $K^F \subseteq K$. As a reminder, the planning horizon $k \in K$ is zero-indexed, such that when there are f periods in the frozen period K^F, the final period in the frozen period is $k = f - 1$, and the first non-frozen time period is $k = f$. The

completion time of each job assignment is calculated using the following:

$$t_{ijk} = \begin{cases} k + P_{ij} & \text{if } k \in K^F \\ f + \tau(1 + k - f) & \text{if } k \in K^{NF} \end{cases}$$

Because the binary variable x_{ijk} defines when a job in the frozen period is started, there exists the chance that a job started during the frozen period will be completed during the non-frozen portion of the schedule. In this case, capacity from the first non-fixed big-bucket bin must be adjusted to account for this overflow. The hybrid-time formulation is presented below:

Formulation **(H)**

$$\min \quad \sum_{j \in J} \left(\sum_{k \in K^F} \sum_{i \in I^A} R_{ijk}\, x_{ijk} + \sum_{k \in K^{NF}} \sum_{s \in S} \sum_{i \in I} q_s\, R_{ijk}\, y^s_{ijk} \right)$$

s.t.

$$\sum_{j \in J} \Big(\sum_{k \in K^F} x_{ijk} + \sum_{k \in K^{NF}} y^s_{ijk} \Big) = 1 \quad \forall \; i \in I^A, s \in S \tag{1}$$

$$\sum_{j \in J} \sum_{k \in K^{NF}} y^s_{ijk} = b_{is} \quad \forall \; i \in I^C, s \in S \tag{2}$$

$$\sum_{i \in I^A} x_{ijk} \leq 1 \quad \forall \; j \in J, k \in K \tag{3}$$

$$\sum_{k \in K^F} x_{ijk} + \sum_{k \in K^{NF}} y^s_{ijk} \leq A_{ij} \quad \forall \; i \in I, j \in J, s \in S \tag{4}$$

$$1 - \sum_{l \in I^A : l \neq i} x_{lj,k+u} \geq x_{ijk} \quad \forall \; i \in I^A, j \in J, k \in K^F, u \in \{0..U_{ijk} - 1\} \tag{5}$$

$$\sum_{i \in I} P_{ij}\, y^s_{ijf} + \sum_{i \in I^A} \sum_{k \in K^F} V_{ijk}\, x_{ijk} \leq C_{jf} \quad \forall \; j \in J, s \in S \tag{6}$$

$$\sum_{i \in I} P_{ij}\, y^s_{ijk} \leq C_{jk} \quad \forall \; j \in J, k \in K^{NF}, s \in S \tag{7}$$

Constraints 1 require each accepted job to be scheduled exactly once for each scenario. Constraints 2 require each contingent job to be scheduled exactly once for only the scenarios in which it exists in the backlog, i.e. $b_{is} = 1$. Constraints 3 ensure that at most one job can be assigned to each position. Constraints 4 ensure that jobs can only be assigned to machines which are capable of processing them. Constraints 5 ensure that no two jobs in the frozen period are processed

simultaneously. Constraints 6 require the total processing time allocated for the first non-fixed big-bucket period to be less than the available capacity in this period. The allocated processing time for this period is the sum of all non-fixed assignments in that period plus all of the carry-over work content from jobs that were assigned (but not completed) during the fixed portion of the schedule. Constraints 7 require the total processing time allocated for all other big-bucket periods is to be less than the available capacity.

5.5 Numerical Implementation

In this section we first extend the data generation framework, presented in Chapter 4, for the contingent demand scheduling problem described above. We then provide a computational analysis of the formulations to highlight the benefits and disadvantages of each formulation.

5.5.1 Data Generation Procedure

The data generation procedure for the contingent demand scheduling problem extends that found in Section 4.4.1. We refer the reader to this section for a description of the user-defined parameters associated with generating each problem instance and the data generation procedures which result in the generation of the following parameters and coefficients:

Notation	Description
α_i	Size of job i
d_i	Due date of job i
c_i	Per-period tardiness penalty of job i
A_{ij}	1 if job i can be processed by resource j
P_{ij}	Processing time of job i by resource j

Table 5.4. Parameters with generation procedures described in Section 4.4.1

Several additional user-defined parameters must be established to guide the generation of the parameters and coefficients associated with the contingent demand scheduling problem. These user-defined parameters are shown in Table 5.5. The following procedure extends that presented in Section 4.4.1.

Each job, $i \in I$, is randomly assigned to either the accepted job set, I^A, or contingent job set, I^C, such that the number of jobs in each set matches the user-specified values. Each job in the

Notation	Description
$\lvert I^A \rvert$	Number of jobs in the accepted job set, I^A
$\lvert I^C \rvert$	Number of jobs in the contingent job set, I^C
$\lvert S \rvert$	Number of scenarios to be considered
f	Number of periods to include in the frozen portion of the schedule
e	Length of due-window for each job i
H	Proportion of job size, α_i to per-period holding cost, h_i
Ω	Set of potential probability of acceptances, a_i any contingent job can have
ω	Probability of each value of Ω being selected for each contingent job

Table 5.5. Additional user-defined parameters, extending Table 4.5, required for the data generation procedure of the Contingent Demand Scheduling Problem

accepted job set, $i \in I^A$, has a probability of acceptance $a_i = 1$. The probability of acceptance, a_i, of each contingent job $i \in I^C$ is randomly selected from the set Ω with probability ω.

The due date characteristics for each job include an earliest acceptable delivery date and a desired delivery date, notated e_i and d_i, respectively. The generation procedure for the desired delivery date, d_i, of each job, i, has been defined in Section 4.4.1. The earliest acceptable delivery date, e_i, is generated as $e_i = d_i - e$. Any job which is completed between the due-window, defined by $[e_i, d_i]$, experiences no penalty. If a job is completed earlier that e_i, it experiences an earliness penalty, h_i. The per-period holding cost, h_i, is proportional to its size: $h_i = H\alpha_i$.

5.5.2 Computational Results

In this section, we investigate the computational performance of the contingent scheduling models presented in Section 5.4. All computational experiments are performed using Gurobi v9.5, on an AWS cloud machine with 32 GB RAM and 8 cores (x5.4xlarge machine). The CPU time limit for each run on each problem instance is 3600 seconds, and an MIPGap termination condition of 0.1% was used to terminate near-optimal solutions.

The objective of this computational comparison is to identify which contingent demand scheduling formulation is most capable of providing high fidelity solutions in reasonable time. We consider various instances of the contingent scheduling problem with a requirement of fixing the first week of production across all demand realization scenarios. Instances vary in the number of certain jobs, $|I^A|$, the number of contingent jobs $|I^C|$, and the number of randomly generated scenarios that are considered $|S|$. Each problem instance is defined by a set of 3 unrelated parallel machines

with eligibility restrictions based on the product type of each job. Table 5.6 describes the defining parameters which are shared across all problem instances. 10 instances of each problem setting are generated and solved by each formulation.

Description	**Notation**	Value
Number of Product Types	$\|U\|$	5
Number of Machines	$\|J\|$	3
Length of frozen period (hours)	f	40
Min/Max size/qty of each job	$\underline{Q}/\overline{Q}$	1/5
Min/Max processing rates	$\underline{P}/\overline{P}$	1/5
Min/Max time for sequence-dependent setups	$\underline{F}/\overline{F}$	1/10
Min/Max per-unit tardiness penalty	$\underline{C}/\overline{C}$	1/5
Range of Due Date	RDD	0.5
Tardiness Factor	TF	0.5
Eligibility Factor	$ELIG$	0.5
Set of Probability of Acceptance	Ω	{0.1, 0.25, 0.5, 0.75, 0.9}
Set of Probability of POA selection	ω	{0.1, 0.2, 0.4, 0.2, 0.1}
Due Window Size	e	20
Holding Cost Rate	H	2

Table 5.6. Design of Experiments: common parameter settings for all problem instances

5.5.2.1 Small Problem Instances

In this subsection, we focus on smaller problem sizes and evaluate the performance of all the presented formulations. We consider problems with 10 and 20 accepted jobs and up to 4 contingent jobs (including a baseline case where no contingent jobs exist). For all instances, all possible demand backlogs are considered, considering their probability of occurrence, i.e. $|S| = 2^{|I^C|}$. In all results presented, the following naming convention is implemented: D = direct-positional formulation, S = small-bucket time-indexed formulation, B = big-bucket time-indexed formulation, and H = hybrid time-indexed formulation.

Table 5.7 shows a computational comparison of the number of binary variables and constraints required to construct the MIP model of each formulation, with respect to the number of accepted and contingent jobs. The values in this table depict the mean of observed values across the 10 randomly generated instances of each problem setting. Clearly, the small-bucket formulation is most susceptible to problem intractability, with respect to computational memory, as the problem size grows. The size of the hybrid-bucket formulation grows quicker as the number of accepted jobs

increases compared to the number of contingent jobs. This is due to the frozen period assignment variables, represented with small-bucket time periods, only being associated with accepted jobs. However, the number of constraints required to represent the problem for the Hybrid formulation (as well as the big-bucket formulation) remain relatively inelastic as the number of contingent jobs/scenarios increases. While the number of constraints and variables is much greater in the hybrid formulation than the big-bucket formulation, it ensures solutions of higher fidelity, especially for the frozen period of the planning horizon.

Table 5.7. Evaluation of formulation model sizes - small problem settings

		KPI	Binary Variables				Constraints			
		Model	D	S	B	H	D	S	B	H
Num. Acc.	Num. Cont.	Num. Scen.								
	0	1	300	2808	300	1500	724	68307	70	21564
	1	2	660	5135	693	2046	1567	124920	151	21040
10	2	4	1512	11707	1620	3168	3555	311390	327	21553
	3	8	3510	24274	3783	5616	8183	626292	707	21491
	4	16	8148	57053	8778	11088	18839	1450204	1523	21334
	0	1	1200	11484	1200	3600	2654	252248	140	35740
	1	2	2520	21483	2583	5166	5547	457991	291	34680
20	2	4	5412	44352	5610	8448	11875	954553	607	34917
	3	8	11730	97594	12213	15456	25663	2125419	1267	35534
	4	16	25488	206323	26568	30528	55599	4229176	2643	35979

The results of the solution run times is shown in Table 5.8. *Count* shows the number of trials (out of 10) which reach the optimality termination condition within the maximum time limit of 1 hour, and *Mean Runtime* and *Median Runtime* are the mean/median runtime required for those trials to reach that condition. Only the direct-positional formulation and small-bucket time indexed formulation are unable to reach the optimality condition in any of the problems which were attempted. Table 5.9 reports the MIPGaps which were achieved for trials which were not able to reach the optimality termination condition during the maximum time limit. Note that for all problems that did not reach termination for the direct-positional formulation, the MIPGap remained at 100%. This shows that the lower bound did not become positive (¿0) at any point.

Table 5.8. Aggregation of runtimes for trials which reached termination condition - small problem settings

Num. Acc.	Num. Cont.	KPI Model Num. Scen.	Count D	Count S	Count B	Count H	Mean Runtime D	Mean Runtime S	Mean Runtime B	Mean Runtime H	Median Runtime D	Median Runtime S	Median Runtime B	Median Runtime H
10	0	1	10	10	10	10	0.0	2.1	0.0	0.7	0.0	2.2	0.0	0.7
	1	2	10	10	10	10	0.1	2.4	0.0	0.6	0.0	2.0	0.0	0.6
	2	4	10	10	10	10	1.1	9.4	0.0	0.8	0.5	7.4	0.0	0.7
	3	8	10	10	10	10	8.4	45.0	0.0	1.0	4.5	19.8	0.0	1.0
	4	16	10	10	10	10	112.8	194.6	0.1	2.0	27.3	65.9	0.1	2.0
20	0	1	9	10	10	10	27.5	15.8	0.0	3.2	2.6	11.1	0.0	3.1
	1	2	10	10	10	10	201.6	40.3	0.1	4.0	51.5	22.2	0.1	3.4
	2	4	5	10	10	10	1023.6	54.1	0.2	3.8	112.0	50.5	0.2	3.6
	3	8	3	10	10	10	682.0	165.9	0.6	6.6	313.4	128.1	0.4	6.7
	4	16	1	6	10	10	2025.6	646.7	4.2	16.1	2025.6	653.8	3.5	15.6

Table 5.9. Aggregation of achieved MIPGaps - small problem settings

Num. Acc.	Num. Cont.	KPI Model Num. Scen.	Count D	Count S	Count B	Count H	Mean MIPGap (%) D	Mean MIPGap (%) S	Mean MIPGap (%) B	Mean MIPGap (%) H	Median MIPGap (%) D	Median MIPGap (%) S	Median MIPGap (%) B	Median MIPGap (%) H
10	0	1	0	0	0	0	-	-	-	-	-	-	-	-
	1	2	0	0	0	0	-	-	-	-	-	-	-	-
	2	4	0	0	0	0	-	-	-	-	-	-	-	-
	3	8	0	0	0	0	-	-	-	-	-	-	-	-
	4	16	0	0	0	0	-	-	-	-	-	-	-	-
20	0	1	1	0	0	0	100.0	-	-	-	100.0	-	-	-
	1	2	0	0	0	0	-	-	-	-	-	-	-	-
	2	4	5	0	0	0	100.0	-	-	-	100.0	-	-	-
	3	8	7	0	0	0	100.0	-	-	-	100.0	-	-	-
	4	16	9	4	0	0	100.0	67.5	-	-	100.0	74.3	-	-

5.5.2.2 Large Problem Instances

This section considers larger problem instance sizes for the Binned and Hybrid formulations, which are shown to be capable of solving much larger problem instances. In addressing larger sets of contingent jobs, which entail an exponential number of potential backlog realizations, we now generate a limited, random set of scenarios to evaluate the formulations. Scenarios are randomly generated using a Monte Carlo simulation in which each contingent job is included in a scenario,

i.e. $b_{is} = 1$, with a probability a_i. Each scenario has a probability $q_s = \frac{1}{|S|}$ of occurring, where $|S|$ is the user-defined number of scenarios to generate.

It is shown in Table 5.10 reports the number of binary variables and constraints required to represent the scheduling problem. It should be noted that problem sizes can be reduced by only considering a subset of the accepted jobs to be permitted for assignment within the frozen portion of the schedule. For example, if a group of accepted jobs have due dates far into the future, they would not need to be considered for scheduling in the immediate portion of the schedule. This additional restriction would reduce the size of the Hybrid Time formulations drastically, as the greatest number of variables and constraints describe the frozen portion of the schedule (e.g. 40 hourly periods per week in the frozen period vs 1 in the non-frozen portion of the schedule).

Table 5.10. Evaluation of formulation model sizes - large problem settings

		KPI	Binary Variables		Constraints	
		Model	B	H	B	H
Num. Acc.	Num. Cont.	Num. Scen.				
20	10	25	65340	71100	5178	34975
		50	130590	138600	10353	39587
		100	261090	273600	20703	49992
	20	25	117120	124800	6928	31573
		50	234120	244800	13853	38533
		100	468120	484800	27703	53416
30	10	25	117120	124800	6928	44375
		50	234120	244800	13853	50849
		100	468120	484800	27703	64728
	20	25	183900	193500	8678	39493
		50	367650	381000	17353	48572
		100	735150	756000	34703	67253

Tables 5.11 and 5.12 report the performance of the big-bucket and Hybrid formulations for larger problem instances. Both formulations are capable of solving a majority of the tested instances, and for instances which do not reach the termination condition, the MIPGap reaches a near-optimal solution.

Table 5.11. Aggregation of runtimes for trials which reached termination condition - large problem settings

Num. Acc.	Num. Cont.	KPI Model Num. Scen.	Count B	Count H	Mean Runtime B	Mean Runtime H	Median Runtime B	Median Runtime H
20	10	25	10	10	3.5	18.5	3.1	18.5
		50	10	10	19.0	77.5	11.1	57.4
		100	10	10	53.5	147.5	31.0	90.2
	20	25	10	10	5.7	52.2	4.2	29.8
		50	10	10	21.7	606.1	14.6	164.8
		100	10	9	78.2	906.6	67.0	521.3
30	10	25	10	10	10.5	31.9	6.7	27.0
		50	10	10	34.3	193.8	24.2	102.9
		100	9	7	74.0	600.8	87.8	232.7
	20	25	10	10	10.3	251.3	9.3	53.3
		50	10	7	64.0	400.2	40.4	293.1
		100	9	7	137.4	1078.2	128.9	1266.7

Table 5.12. Aggregation of achieved MIPGaps - large problem settings

Num. Acc.	Num. Cont.	KPI Model Num. Scen.	Count B	Count H	Mean MIPGap (%) B	Mean MIPGap (%) H	Median MIPGap (%) B	Median MIPGap (%) H
20	10	25	0	0	-	-	-	-
		50	0	0	-	-	-	-
		100	0	0	-	-	-	-
	20	25	0	0	-	-	-	-
		50	0	0	-	-	-	-
		100	0	1	-	1.5	-	1.5
30	10	25	0	0	-	-	-	-
		50	0	0	-	-	-	-
		100	1	3	1.8	1.4	1.8	1.3
	20	25	0	0	-	-	-	-
		50	0	3	-	1.2	-	1.2
		100	1	3	1.2	1.7	1.2	1.6

From these results, the Hybrid time-indexed formulation is superior relative the direct-positional, small-bucket and big-bucket formulations. The Hybrid formulation provides a higher fidelity solution than the big-bucket formulation, a model which requires less computational burden than the direct-positional formulation, and a problem structure which requires less computational memory

than the small-bucket formulation. It is also shown that the Hybrid model is capable of solving large problems in reasonable time for the unrelated parallel machine scheduling problem subject to strict job-machine eligibility restrictions with weighted earliness and tardiness penalties.

5.6 Future Research Directions

Future work in this research stream can be dedicated to: 1) enhancing the computational results associated with the contingent demand scheduling problem including testing under a wider variety of problem settings, 2) evaluating the benefits of considering contingent demand in the scheduling of the frozen period, and 3) extending the framework to consider the Order Acceptance & Scheduling Problem, Dynamic Pricing Problem, Due Date Setting Problem, and the Simultaneous Pricing, Due Date Setting and Scheduling Problem, all considering contingent demand. We provide an brief overview of these extensions in Appendix F.

5.7 Conclusion

In this chapter, we develop and compare several scheduling formulations which act as the underlying framework for revenue management applications for MTO firms which are subject to contingent demands. Specifically, we consider the unrelated parallel machine scheduling problem, subject to contingent demand, with a requirement for freezing the most immediate portion of the schedule to promote shop stability. We develop novel, scenario-based formulations to consider the uncertainties that derive from a contingent demand backlog. The formulations developed include a direct-positional, a small-bucket time-indexed, a large-bucket time-indexed, and a hybrid-bucket time-indexed scheduling model. We also present a special case of the formulation which can be achieved when all jobs are represented as identical.

We show through a computational experiment that the hybrid-bucket formulation provides a promising trade-off between solution fidelity and computational requirements. The continuous-time direct-positional formulation provides the highest fidelity solution at the cost of extensive solution runtimes. The small-bucket time-indexed formulation also provides high fidelity solution at the cost of extensive computational memory requirements. The big-bucket formulation is shown to provide solutions quickly but at the cost of low-fidelity solutions. We also discuss future research directions and extensions to the formulations presented in Section F.0.3.

CHAPTER 6
CONCLUSION

In this dissertation, we developed and applied a Digital Twin framework for manufacturing systems as a decision support system to aid ii production planning and scheduling problems faced by Make-To-Order (MTO) firms. Part 1 of this dissertation addressed a generalized class of Production Planning and Execution problems faced by MTO firms exhibiting a job shop manufacturing system. Part 2 focused on a class of scheduling problems faced by manufacturers whose production system is dominated by a single operation.

In Chapter 2, we introduced the reader to the Digital Twin (DT) concept and developed the framework which is used throughout the dissertation. We presented the DT framework as a collection of modular software packages, developed using *Python*, and described the object-oriented methodologies implemented to represent the generalized manufacturing systems we consider in our applications. We also developed algorithms for generating large-scale, realistic production facilities representative of Tier II manufacturers in the aerospace supply chain. Further, we presented several heuristics capable of creating a production plan and executable schedule for the scenarios generated by the DT. To validate the framework, we presented an illustrative example of a randomly generated production facility, highlighting the importance of the advanced analytic and visualization capabilities of the framework. Future research directions include: the enhancement of the algorithms presented in this chapter to be applicable to other specific use-cases, the expansion of the framework to enable the ingestion of real-world data from existing data architectures, and the integration of the presented analytic functionalities and visualizations into a centralized, interactive, web-based platform.

In Chapter 3, we addressed the Multi-Level Capacitated Lot Sizing Problem (MLCLSP). We developed an integrated solution procedure that is capable of: 1) creating a long-term production plan, 2) matching each production lot with the sales order they satisfy, and 3) translating the lot-sizing production plan into an executable schedule which can be followed on the shop floor. In this

implementation, we incorporated many practical considerations which have been overlooked in the existing literature. Our main contributions include the formulation of novel modelling techniques for: 1) discretizing the planning horizon for each resource, dynamically, on the basis of the processes they typically face, and 2) considering intra-route lead times for cyclical production routings. We evaluated the proposed solution procedure against several benchmark cases, and found that the formulations developed in this chapter are superior in finding high-quality solutions in reasonable computational time. We emphasize the importance of evaluating the performance of a production plan in the unaggregated context which it will be executed in. We also presented a methodology for calibrating the framework and evaluating various scheduling policies in Appendix B. Future research directions include the evaluation of the scalability of the formulations presented in this chapter, the development of more use-case specific generalizations which will enable the practical use of the framework, and the enhancement of the framework through the introduction of tighter formulations. Further, enhancements to the forward-constructed, myopic, Continuous-Time Scheduling Heuristic can be developed to incorporate *improvement* heuristics to the scheduling procedure.

In Chapter 4, we developed direct-positional and relative-positional formulations for the unrelated parallel machine and flexible flow shop scheduling problems, subject to sequence-dependent setups and machine eligibility restrictions. We provided a computational comparison of the two exact-method optimization modeling techniques when applied to randomly generated problem instances from the proposed Digital Twin framework. We observed that the direct-positional formulation dominated the relative-positional formulation in all tested production settings. However, future work is required to enhance the performance of the tested formulations to tighten the models, specifically through the calibration of the big-M parameters presented in this chapter.

In Chapter 5, we extended the scheduling formulations of Chapter 4 to consider uncertainties derived from the contingent demand paradigm faced by MTO firms which offer mass customized products. In this context, we considered a practical scheduling objective of freezing the first-most portion of the schedule to promote shop stability. We first extended the unrelated parallel machine direct-positional formulation from Chapter 4 to a scenario-based formulation. We also developed several variations of time-indexed formulations for the scheduling problem, differentiated by the duration of each period in the discretized planning horizon. We showed, through a computational evaluation, that the direct-positional and small-bucket formulation faced challenges in the

computational expense and memory, respectively, required to solve the generated problems, while the big-bucket formulation provided a low-fidelity solution. We developed a Hybrid-Bucket time-indexed formulation and showed that it provides a promising trade-off between solution fidelity and computational efficiency. We also formulated an extension to the scheduling problem to address the Order Acceptance & Scheduling problem (considering risk-neutral and risk-averse objectives), and presented a methodology for quantifying the value of considering a scenario-based scheduling model when subject to a contingent demand backlog.

In conclusion, we emphasize the generality and flexibility which applying a Digital Twin framework to traditional production planning and scheduling problems provides researchers and practitioners. The value of frameworks like this are exhibited in their ability to allow developers to apply their algorithms and optimization formulations to a wide-range of use-cases with minimal modeling efforts. Further, Digital Twin frameworks allow researchers to develop applications, considering a stream of real-time data, without affecting the actual physical systems they represent. This provides them the test-bed for the realistic evaluation of competing strategies, simultaneously, to derive the insights and policies that will enable decision-makers to adapt in a global market with increased competition and an increasingly fragile supply chain. While current data architectures implemented in practice today are not yet capable of providing the cross-functional data transparency required to deploy the Digital Twin concept in real-world manufacturing systems, we show that, with the use of random data generators, these frameworks of the future can and should be developed today.

APPENDIX A

SOLUTION APPROACHES FOR LARGE JOB SHOP PROBLEMS

In this Appendix, we provide an overview of the solution approaches which have been implemented to solve large instances of the Job Shop Problem. In Chapter 3, we implement exact-method algorithms for the Planning and Pegging Models. We plan to investigate potential implementations of the approaches below as we continue this study.

In this review, we consider the classification scheme developed by Buschkul et al. [32]. As a reminder, they classify solution approaches for the Lot Sizing Problem into five groups: mathematical programming heuristics, Lagrangean heuristics, decomposition and aggregation heuristics, meta-heuristics and problem specific greedy heuristics. In the following we summarize several of these approaches and identify examples in recent literature which apply them to problems similar to the once we consider. We focus primarily on examples in literature that implement approaches using meta-heuristics and greedy heuristics. Not presented in this review are approaches which consider Lagrangian Heuristics.

A.1 Integer Programming Solution Approaches

A.1.1 Branch-and-Bound Heuristics

The branch-and-bound (B&B) algorithm, developed in the 1950s by Land and Doig [96], is an enumerate search technique for solving combinatorial optimization problems to optimality. B&B algorithms are characterized mainly by 1) a branching rule dictating how solutions are partitioned into groups and 2) a bounding rule which computes the lower/upper bound of the cost/reward of any solution within the partitioned subset. Whether the lower or upper bound is used depends on if the optimization problem is a minimization problem or maximization problem, respectively. Other characteristics in B&B algorithms include dominance rules which can eliminate subsets of possible solutions from consideration and additional bounding rules that can be evaluated to produce and upper bound for minimization problems or lower bound for maximization problems.

B&B algorithms first began to be used in scheduling in the mid 1960's, see [107, 81], using a *forward sequencing* branch rule. In a forward sequencing branch rule, the first branch in the search tree considers all possible jobs that can be assigned to the first position, the second branching considers all jobs that can take the second position, and so on. Bounding schemes usually involved solving relaxed versions of the original problem [135]. Backward sequencing branching rules were also of particular interest, specifically in single machine problems to minimize total tardiness and total weighted tardiness, see [54]. The use of job-based bounding rules and machine-based bounding rules along with various mathematical programs is found in the literature [135]. Examples of programs used to define bounds include the polyhedral approach and the Lagrangian relaxation approach. Both of these examples were developed to solve the Traveling Salesman Problem and later implemented for the scheduling problem, see [75]. Because branch-and-bound algorithms are enumerative, they quickly become an impractical solution method as the search space grows exponentially as the scheduling problem grows.

A.1.2 Valid Inequalities

Valid inequalities attempt to supplement the overall solution process through the addition of constraints which result in a tighter model. Using expert system knowledge, it is possible to reduce the size of the problem by cutting off potentially large portions of the search space. If the inequalities are generated dynamically to cut off the current non-integer solution, it is called the cutting plane method. Valid inequalities can also be introduced to the B&B algorithm. A distinction is made as to whether the inequalities are introduced during the B&B procedure (Branch-and-Cut), or prior to the start of the procedure (Cut-and-Branch).

A.1.3 Fix-and-Relax Heuristics

The Fix-and-Relax attempts to solve the problem by partitioning the binary variables into three groups. While one subset of variables is solved to optimality, another group is fixed (to the values from the last iteration) and another is relaxed from its binary condition. Heuristics which are developed differ in the definition of these subsets. Simplifications of this heuristic, such as fix-and-optimize, only partition the binary variables into two groups.

Helber and Sahling [74] use the fix-and-optimize heuristic introduced by Sahling et al. [150] to solve MLCLSP with positive lead times. They compare their algorithm to previous algorithms by Tempelmeier and Derstroff [172] and Stadtler [164] and conclude that it outperforms them. This paper is also relevant for our purposes because HJSP deals with positive lead times as well.

A.2 Decomposition and Aggregation

Decomposition and aggregation solution approaches aim to reduce the complexity of LSP problems by either breaking the problem into separable partitions or by ignoring the detailed structure of the problem, respectively. Decomposition approaches work by solving several sub-problems then coordinating the solutions, while aggregation approaches solve a less granular problem then solving a more detailed version of the problem. Characteristics of the LSP that can be decomposed or aggregated can be time-based, item-based or resource-based.

Item-based decompositions are concerned with multi-product production systems. This approach aims to solve for the LSP for subgroups of items, then coordinate a solution. Item-based aggregations solve a simplified versions of the LSP by representing sub-groups of items as a single item. See example, Boctor and Poulin [27]. Resource-based decompositions and aggregations consider a similar approach.

A.2.1 Time-Based Decomposition

Stadtler [164] proposes a time-oriented decomposition heuristic to solve the MLCLSP with general product structures, multiple constrained resources and setup times. The heuristic depends on an internally rolling lot-sizing window and the lot-sizing decisions are made sequentially. This approach decomposes the problem into submodels, which are represented by the "Simple Plant Location" model formulation. They show their approach to provide better solutions than another heuristic by Tempelmeier and Derstroff [172], as well as the ability to solve larger problem sizes.

Wu et al. [190] extend the work in Sahling et al. [150] by allowing backlogging in addition to setup carry-overs. They propose mathematical models and a new heuristic, which they call a progressive time-oriented decomposition heuristic (PTH) to solve test instances from the literature. They conclude that their heuristic is superior to previous applications and a commercial solver. This paper is the most relevant one in the literature to our study since our execution model also considers

partial sequencing of the items and backlogging. The differences between this paper and ours will be discussed below, along with the rest of the literature.

A.3 Meta-heuristics approaches

Meta-heuristics describe high level procedures designed to find near-optimal solutions for large-scale optimization problems efficiently. These approaches are problem-specific that utilize expert knowledge of the system and implement heuristics such as dispatching rules for determining optimal lot-sizes. In the following we present several commonly found meta-heuristics including Variable Neighborhood Search, Simulated Annealing, Tabu Search, Genetic Algorithms and Ant Colony Optimization techniques. It should be noted that aspects of multiple meta-heuristics can be combined to create hybrid meta-heuristics.

All meta-heuristics are initialized with a starting solution and work to improve them by iteratively searching the solution space. Searching principles that guide the *movement* from one solution to the next include diversification and intensification. Diversification serves the purpose of altering the solution as to avoid becoming trapped in a local optimum, while intensification acts to improve the solution by following the optimal gradient planes. Solutions employed by meta heuristics are either represented directly or indirectly. Direct representations retain the variables define in the mathematical formulation of the problem, i.e. binary variables for setup decisions, continuous variables for lot sizes, etc. Indirect representations encode scheduling solutions, i.e. representing scheduling decisions by selecting a specific heuristic. In this case, the schedule must be inferred from the indirect representation.

In the following, we briefly describe some popular meta-heuristics developed for the MLCLSP and its extensions. We do not conduct a full review, as the development of meta-heuristics for the proposed framework is a part of the proposed work of this dissertation.

A.3.1 Variable Neighborhood Search

Local search algorithms work by starting at an initial solution for the scheduling problem and iteratively tries to *move* to a better solution by searching and evaluating a *neighborhood* solution or solutions. In the following, several methods of generating neighborhood solutions will be presented in a multiple machine system. Solution neighborhoods are generate by creating variations of the

current schedule using one of three methods: shifting, insertion, and swapping. In a shift, a job is randomly chosen from on of the machines schedule and "shifted" to a different position of the same machine. In an insertion, a job is randomly chosen from one machine's schedule and "inserted" into a random position from another randomly selected machine's schedule. In a swap, two random jobs from two random machine's schedules are swapped. These variations represent a single one-step neighbor of an existing solution. A neighborhood can made up of any number of neighbors using any or all of the methods mentioned above. Within each iteration of a local search algorithm, it is possible to consider only a single neighbor of the current solution, a subset of the neighbors within the neighborhood of the current solution, or the entire neighborhood simultaneously. Due to the combinatorial complexity of the scheduling problem, generating and evaluating an entire solution neighborhood may be infeasible. It is also possible to consider multi-step neighbors (neighbors of a neighbor) of the current solution.

The VNS is a local search based meta-heuristic which systematically changes the neighbourhood structure of local search and which uses a random shaking routine to create a new starting solution for a local search every time when it reaches a local optimum.

Chen and Chu[41] model a supply chain planning problem as an MLCLSP. They develop a heuristic approach to solve this problem based on Lagrangian Relaxation (LR) and local search. Their approach only relaxes the binary setup constraints and forces them to take the value of 1 if the corresponding continuous variable is non-zero. They solve the relaxed linear problem (LP) using the simplex method and update the Lagrange multipliers using a surrogate subgradient method. A feasible solution is obtained at each step and improved by a local search by changing two setup variables at a time. They take advantage of a special structure of the LSP and improve upon the local search, reducing computation time. They use numerical experiments to show the effectiveness of their approach by comparing their solutions to those obtained by a commercial solver. They solve small sized problems (10 items, 6 periods) to 1% optimality gap within a second and their algorithm finds solutions to medium sized problems (60 items, 12 periods) within 360 seconds. The objective function value obtained by the algorithm is 10.5% better than the solution found by the commercial solver when it terminates after 10000 seconds.

Chen [40] develops a fix-and-optimize and VNS approach for the MLCLSP with and without setup carryover. The author proposes a framework which first implements the fix-and-optimize

algorithm approach, then feeds the solution to a VNS algorithm which improves the solution by diversifying the search space. Given a starting solution, the VNS defines a series of neighborhood solutions by "shaking" the solution. The author defines the solution in terms of the "setup plan", arguing that the solution to the MLCLSP can be inferred from the resulting values of the setup variables. A swapping heuristic which *switches* setup activity binary variables ($0 \leftarrow 1 || 1 \leftarrow 0$) is used to generate neighbors and the fix-and-optimize heuristic is implemented to find a local optimum. A neighbor generated using this process is only considered if it produces a feasible starting solution for the fix-and-optimize search. The process is completed until a maximum iteration stopping condition is met. The author shows that the proposed meta-heuristic outperforms the fix-and-optimize approach of Helber and Sahling [74] while arguing that their method is more general and can be applied to other MIP problems.

Other recent examples which use the VNS meta-heuristic and variants include Hindi et al. [77], Li et al. [98], and Xiao et al. [191].

A.3.2 Simulated Annealing

Simulated annealing (SA), first proposed by Kirpatrick et al. [91], accepts worse solutions with some probability, $e^{-\delta/T}$ defined by a temperature parameter, T, and δ as the amount which the objective value of the neighborhood solution is worse than the current solution. Similar to it's inspiration, the value (temperature) of T is iteratively lowered, comparable to that of an annealing metal. As the temperature lowers, worse solutions are chosen with a lower probability. This iterative process is repeated until a maximum number of conditions is reached. This algorithm is also defined by meta-parameters which control the starting temperature of T and the "cooling scheme" of T.

Torkaman, Ghomi and Karimi [175] consider the MLCLSP with sequence-dependent setups and re-manufacturing activities. The authors develop a hybrid meta-heuristic with an SA to solve the problem using a genetic algorithm to find an initial solution. Solutions are represented as two strings of length $2NT$ where N is the number of products and T is the number of periods. The first string represents the sequence of products and their relevant processes as integers, and the second row details whether the processes of a product would be executed in each period. Neighbors of the solution are created through 1) a sequence swapping procedure between two

products in a randomly selected period t, or 2) a switch of a production decision variable of a random product/process/period. The meta-heuristic was compared to four rolling horizon based fix-and-optimize heuristics, and was found to solve perform significantly faster for smaller problem sizes and perform significantly better for large instances.

Other recent examples which use the Simulated Annealing meta-heuristic and variants include Ozdamar and Barbarosoglu [128], Ramezanian and Saidi [139], and Sifaleras, Konstantaras and Mladenovic [156].

A.3.3 Tabu Search

Tabu search (TS), introduced by Glover [61] takes an approach in which the best solution in the neighbourhood of the current solution is chosen, regardless of the objective value. A "tabu list" keeps track of and forbids recently visited solutions to ensure that the search algorithm does not fall into a cycle. The search is stopped once a predetermined number of tabu solutions is reached, or a predetermined number of steps have been taken without finding a best performing solution. At the end of the search, the solution with the best performance is chosen.

Hindi [76] applies a tabu search algorithm for the single-level capacitated lot sizing problem, using a starting solution generated using a non-optimal exact method procedure. The tabu search is used to improve the solution, represented as a setup schedule. The authors use generate new solutions by randomly switching the value of setup decisions in random periods, and show that this algorithm is capable of finding near optimal solutions. The authors note that it exploring complete neighborhoods, although adequate for small problems, is not necessary for larger problems. Rather, several neighbors should be evaluated before moving to the next neighborhood. The author refers to this as search a *restricted* neighborhood and shows that the heuristic is capable of producing high quality solutions in short computation times with modest memory requirements.

Other recent examples which use the Tabu Search meta-heuristic and variants include Berretta, Franca and Armentano [23], Gopalakrishnan et al. [63], Raza, Akgunduz and Chen [142], and Romero et al. [145].

A.3.4 Genetic Algorithms

Genetic algorithms (GA) take a different approach to local search. Rather than iteratively improving a single solution, GA aims to improve a "population" of solutions. In each iteration, pairs of solutions are chosen from the current population and are used to create pairs of offspring. These offspring solutions are created as either a "crossover", of the parents where segments of either parent solutions are swapped. Then each of the offspring undergo a "mutation" where elements of the solution are randomly changed (similar to selecting a random neighbor from a neighborhood like those evaluated in SA or TS). The offspring created in this iteration replace the worst performing solutions from the population. The algorithm continues until a predetermined number of "generations" (iterations) is reached, a predetermined number of iterations have gone without a new optimum candidate solution, or a predetermined objective value is reached by one of the solutions.

Recent examples which use the Genetic Algorithms meta-heuristic and variants include Badri et al. [11], Pitakaso [132], Toledo, Oliveira and Franca [173], Toledo et al. [174], and Xie and Dong [192],

A.3.5 Ant Colony Optimization

Many meta-heuristics which have been developed take inspiration from nature. An example of this is the Ant Colony Optimization (ACO) algorithm, takes inspiration from the foraging behaviors of ants. The ACO algorithm implements mechanisms to guide optimal solution search similar to how ants releasing pheromones to mark favorable paths that should be followed by others. The ACO algorithm is initialized by creating a *pheromon value* for each decision variable. For each iteration, a population of *ants* traverses through a graphical representation of the solution space. In the case of the MLCLSP, this could be represented as a forward-constructed sequence of operations. The choice of the path which each ant traverses is determined using a stochastic mechanism, weighted by the pheromone value of each reachable vertex. Once each of the ants has completed traversal, a local search optimization can be performed on each initial solution to identify local optimum. The pheromone values of each decision variable are then updated by decreasing the pheromone values of decision variables associated with poor solutions and increasing the values associated with good solutions. This process is repeated until some stopping condition are met.

Hajipour et al. [69] develop a hybrid ACO algorithm for the capacitated lot sizing problem for a multi-level multi-product capacitated production system with general product structures and time-varying costs and capacity. The authors do not consider lead time, and do not allow backlogging. A production network consisting of 7 products, each with 3 levels, considering 6 time periods and a single resource is considered for the computational experiment. The authors develop an ACO with a novel heuristic they call the shifting technique and compare the algorithm with others such as the TS, SA and GA algorithms as well as an exact solution obtained using a Lagrangian relaxation heuristic. Results show that the proposed ACO algorithm outperforms the TS and SA algorithms, and provides comparable results to the GA. It is noted that the effectiveness of the algorithm decreases as problem size increases.

Other recent examples which use Ant Colony Optimization and variants include Almeder [4], Pitakaso et al. [133], Homberger and Gehring [78], and Buer, Homberger and Gehring [31].

A.4 Greedy Heuristics

Greedy heuristics are characterized by intuitive algorithms which iteratively construct or improve an existing solution making decisions based on feasibility conditions and priority indices [32]. Constructive greedy heuristics work by iterating through the time periods (either forwards or backwards) in the planning horizon and adding to the schedule. These heuristics are *myopic* and do not consider the implications that current decisions have in the future. Priority indices that influence lot sizes and setup decisions often make use of metrics calculated based on the solution of the uncapacitated solution for the lot-sizing problem, such as [46, 66, 159, 158]. We plan to conduct a full literature review in the topic of greedy heuristics for the MLCLSP for this dissertation, focusing on priority indices used for determining lot sequencing, as we look to enhance the greedy heuristic proposed in this framework.

APPENDIX B

CALIBRATION OF THE PRODUCTION PLANNING AND EXECUTION FRAMEWORK

In this appendix, we evaluate the proposed mathematical formulations developed in Chapter 3 when integrated with the Digital Twin framework presented in Chapter 2. Our objective in these evaluations is to provide context to the sensitivities of the proposed Planning and Execution solution procedure subject to: 1) the parameters used to generate cost/reward coefficients in the Planning Model, and 2) the implementation of manufacturing policies representing user preferences in the solution provided by the framework. For a description of the experimental design implemented for the analysis in this section, see Section 3.5.1.

The remainder of this appendix is structured as follows. In Section B.1, we present the methodology for defining and calibrating the user-defined parameters used to generate the cost coefficients in the objective function of the proposed Planning Model (See Section 3.4). In Section B.2, we evaluate the impacts of implementing various manufacturing scheduling policies in the proposed Planning Model and CTSH algorithm. In Section B.3, we conclude on our findings.

B.1 Parametric Calibration

The objective of this analysis is to quantify the sensitivity of the proposed production Planning Model in relation to the value of different parameters which are used to derive the cost coefficients. We evaluate the following user-defined parameters: 1) the per-unit per-period tardiness penalty, P, 2) the per-unit per-period reward for early deliveries, ϵ, 3) the per-unit penalty for the unfulfillment of requested external demands during the planning horizon, $\overline{P}$, 4) the per-unit per-period holding cost, h, and 5) the per-unit penalty for under-production of parts relative to the total production requirements required to satisfy all external demands, H.

These parameters have been strategically defined to be **intuitive** to a scheduler/planner who would be an end-user of the proposed framework. It should be noted that several of these param-

eters are highly related, e.g. the per-period per-unit lateness penalty, P, and per-unit penalty for unfulfilled orders, $\overline{P}$. It is critical for the relationships between related parameters to be logical such that the solutions derived from the Planning Model are viable. In the following, we introduce each of these parameters and present their relationships with the objective function coefficients they impact.

The tardiness penalty parameter, P, dictates the per-unit per-period cost for tardiness. This parameter is specified as a percentage of the *Value*, V_i, of the end item, $i \leftarrow I^o$, associated with each line item, o. This percentage represents the cost accrued each week for delivering an end-item late, relative to the due date of the associated line item, D_o. This penalty is realized per-period, Δ, which may not be weekly. In such cases, the penalty will be prorated according to the number of hours-per-week, HPW.

The demand unfulfillment parameter, $\overline{P}$, is used for the per-unit penalty for demands which are not satisfied by the end of the planning horizon. The coefficient, associated with a line item o, $\overline{P_o}$, is defined in relation to the per-period tardiness penalty coefficient, P_o. Specifically, it is defined as a multiple of the realized tardiness penalty that would be assessed to that end item if it was delivered at the end of the planning horizon, i.e. when $t = |T|$. For example, when $\overline{P}$ is 1.25, the penalty realized for not delivering an end item will be 1.25 times the penalty of delivering the end item at the end of the planning horizon, $P_o * (|T| - D_o)$. Intuitively, when the value of $\overline{P}$ is greater than 1, the Planning Model experiences a greater penalty for not delivering an end-item than it does for delivering it anytime during the planning horizon. Therefore, a larger value of $\overline{P}$ provides a larger incentive for delivering as many end items as possible.

The parameter, ϵ, is associated with a per-unit per-period reward for early deliveries. The coefficients derived from ϵ are also defined in relation to the tardiness penalty parameter, P. This reward is defined as a proportion of the penalty for late deliveries. For example, when $\epsilon = 1\%$, the Planning Model would receive a per-period reward which is $1/100^{th}$ the penalty for delivering one period late. Because the reward realized in the proposed Planning Model is *weighted* by the value of each end item, it is important that this reward incentive is not so large that a reward for delivering a high-value end items early would lead to the under-prioritization of the on-time delivery of lower-value end items.

The parameter, h, is associated with the per-unit per-period holding cost for each part, i. This parameter is defined as a percentage of the part's value, V_i, experienced as a cost, per year. For example, when $h = 20\%$, it is assumed that the cost of holding a part, i, for one year would be 20% of the value of that part. Similar to the derivation of the coefficients for tardiness, holding cost coefficients are accrued per period, such that the yearly cost, which the user-defined parameter is defined, would need to be prorated based on the length of each period, Δ.

The parameter H dictates the per-unit penalty associated with the under-production of each part, i, relative to the production requirements, N_i, necessary to be able to complete all end items associated with all line items during the planning horizon. This provides incentive to begin production of end items, even if they are not delivered during the planning horizon. This penalty is defined relative to the holding cost of each part, h_i, such that the penalty for NOT producing a unit of the part, i, outweighs the penalty for producing the part and holding it in inventory. Specifically, the parameter, H, is defined as a multiple of the cost of holding a part for the entire duration of the planning horizon, i.e. $h_i * |T|$.

These user-defined parameters are summarized in Table B.1 where *DoE Values* are the values for the parameters which are evaluated in the following sensitivity analysis. *Obj. Coefficient* is the association between each user-defined *Parameter* and the part/order-specific coefficients in the Planning Model. Bolded values represent the **default** values of each parameter. These values are fixed in the sensitivity analysis when the focus is on another parameter.

Parameter	**Description**	**DoE Values**	**Obj. Coefficient**
P	Delay penalty (% per week)	1%, **2.5%**, 5%	$P_o = P * V_{i \leftarrow I^o} * \frac{\Delta}{HPW}$
$\overline{P}$	Unfulfilled order penalty factor	1, **1.25**, 1.5	$\overline{P_o} = \overline{P} * P_o * (\lvert T \rvert - D_o)$
ϵ	Earliness incentive (% of P)	0.1%, **1%**, 10%	$\epsilon_o = \epsilon * P_o$
h	Holding cost rate (% per year)	10%, **20%**, 30%	$h_i = h * V_i * \frac{\Delta}{HPY}$
H	Underproduction penalty factor	0.5, **1**, 2	$H_i = H * h_i * \lvert T \rvert$

Table B.1. Subset of relevant parameters for parametric calibration (default values bolded)... Assume 40 work hours per week, Δ is time discretization factor, $|T|$ is length of planning horizon, V_i is value of part i, and D_o is the due date of order o

In the following sensitivity analysis, we evaluate the impact of these parameters on Key Performance Indicators (KPI) related to the shipments made during the planning horizon (evaluated in

the basis of units and dollars in each shipment), the realized costs from the resulting production plan, resource utilization, and the number and size of each dynamically generated shipment and work order in the production plan.

B.1.1 Tardiness Penalty

Table B.2 provides an view of the performance of the Planning Model as the parameter for per-unit per-period tardiness changes. Values shown in this table represent the *average* value of each metric across the 10 **shared** `base_factory` test instances. For example, values shown for the *25th pc.* of *Lateness* is calculated as the mean of the 25th percentile of lateness of shipments from each test instance. The index, *Measure*, indicates whether the evaluations consider the delivery performance is aggregated against the shipment of dollars or units. P is the independent variable which is used to update each `base_factory`.

The multi-column, *% of Demanded*, represents the percentage of demands that were delivered during the planning horizon (*Delivered*) and delivered on time (*On Time*). Multi-column *Lateness (Hours)* describes the distribution of the lateness for shipments made during the planning horizon. Shipments which were delivered earlier than the associated order's due date will have a negative value for lateness. Table B.2 shows that as P increases, so does the percentage of demands which are shipped to the customer *On Time*. Further, as P increases, the lateness of shipments decreases. Note that the total percentage of *Delivered* Dollars and Units initially increases then decreases as P increases. This is due to the over-prioritization of delivering high-value demands on time rather than shipping low-value demands at all.

Table B.2. Parametric Sensitivity Analysis: Analysis of planned delivery performance based on each model and its corresponding assumptions with respect to changes in the tardiness penalty parameter, P

		% of Demanded		Lateness (Hours)			
Measure	**P**	Delivered	On Time	Mean	25th pc.	Median	75th pc.
Dollars	1%	87.4 %	41.5 %	50.2	-43.0	-6.0	98.0
	2.5%	87.6 %	42.8 %	47.2	-44.0	-5.0	104.0
	5%	87.4 %	45.3 %	45.4	-45.0	-11.0	95.0
Units	1%	86.6 %	47.1 %	60.9	-67.8	-5.0	130.0
	2.5%	86.8 %	48.0 %	59.1	-57.0	-9.5	133.0
	5%	86.1 %	50.0 %	52.5	-62.0	-13.0	115.2

Table B.3 provides an analysis of the realized penalties in the production plan solution with respect to different inputs for P. Values are shown as a percentage of the **Total Cost of Goods Sold**, i.e. the cumulative *value* of all demands in the planning horizon. The total cost of goods sold is calculated as: $\sum_{o \in O} Q_o V_{i \leftarrow I^o}$. As a reminder, the value of each part, V_i, is also used to derive the objective function coefficients in the production planning model.

Holding Cost shows the cost experienced for holding non-WIP inventory, *Under Production* is the cost for shortages in the production of items relative to the internal production requirements necessary to satisfy all external demands, *Tardiness Penalty* is the total penalty for lateness of all deliveries made during the planning horizon, and *Unfulfillment Penalty* is the total penalty experienced for units that were demanded during the planning horizon but not delivered. Note that the experience penalty in ALL categories increase as the parameter P increases. It is trivial that the *Tardiness Penalty* will increase. The *Unfulfillment Penalty* increases since the coefficients generated for the unfulfilment penalty, $\overline{P_o}$, of each order is also derived from P.

Table B.3. Parametric Sensitivity Analysis: Analysis of costs and penalties in the production plan with respect to changes in the tardiness penalty parameter, P

KPI **P**	Holding Cost	Under Production	Tardiness Penalty	Unfulfillment Penalty	Total Penalty
1%	0.67 %	1.96 %	2.47 %	2.19 %	7.29 %
2.5%	0.71 %	2.00 %	6.01 %	5.51 %	14.23 %
5%	0.73 %	2.05 %	11.00 %	10.81 %	24.59 %

Table B.4 shows the sensitivity of the **planned** utilization on the shop floor as P increases. The column, *Process* shows the percentage of available resource hours which are spent in the processing phase of an operation, and *Setup* shows the time spent in the setup phase of the planned tasks; *Total* is the sum of the two. The values shown in this table aggregates the resource hours across all resources groups, $j \in J$, throughout the duration of the planning horizon, $t \in T$. Specifically, the values for *Setup* and *Process* (for a test instance, ω) are calculated as:

$$Setup_\omega = \sum_{j \in J} \sum_{\gamma \in \Gamma^j} \sum_{t \in T} \frac{S_\gamma \, z_{\gamma t}}{C_{jt}}$$

$$Process_{\omega} = \sum_{j \in J} \sum_{\gamma \in \Gamma^j} \sum_{t \in T} \frac{R_\gamma \, m_{\gamma t}}{C_{jt}}$$

respectively, where: Γ^j is the set of all processes which can be processed on resource group j, S_γ is the setup time for process γ, $z_{\gamma t}$ is the number of setups that are planned to be executed during time period t, C_{jt} is the capacity of resource group j in period t, R_γ is the processing rate per-unit for process γ, and $m_{\gamma t}$ is the production lot size (in units) for process γ in period t. The values shown in Table B.4 represent the mean utilization in across the 10 tested scenarios, $\omega \in \Omega$. These results show that the utilization rates in the production plan are relatively inelastic in response to an increase in P, however, a slight decrease in utilization rates can be observed. Specifically, as P increases, the time spent in processing decreases while the time spent in setup increases.

Table B.4. Parametric Sensitivity Analysis: Analysis of resource utilization in production plans with respect to changes in the tardiness penalty parameter, P

P	Process	Setup	Total
1%	65.65 %	13.36 %	77.98 %
2.5%	65.46 %	13.38 %	77.81 %
5%	65.40 %	13.43 %	77.80 %

B.1.2 Unfulfillment Penalty

Table B.5shows the delivery KPIs as $\overline{P}$ increases. *% of Demanded: Delivered* shows that as $\overline{P}$ increases, so does the total percentage of demands which are delivered during the planning horizon. However, as $\overline{P}$ increases, the percentage of units shipped *On Time* decreases. This trend is less clear when evaluating *On Time* dollars. For both dollars and units, the mean lateness of shipments increases with $\overline{P}$. This is due to the prioritization of shipping lower-value demands during the planning horizon over shipping higher-value demands on time.

Table B.5. Parametric Sensitivity Analysis: Analysis of planned delivery performance based on each model and its corresponding assumptions with respect to changes in the unfulfillment penalty parameter, $\overline{P}$

Measure	$\overline{P}$	% of Demanded Delivered	% of Demanded On Time	Lateness (Hours) Mean	Lateness (Hours) 25th pc.	Lateness (Hours) Median	Lateness (Hours) 75th pc.
Dollars	1.00	85.7 %	43.6 %	40.1	-45.0	-8.0	93.0
	1.25	87.6 %	42.8 %	47.2	-44.0	-5.0	104.0
	1.50	87.7 %	43.9 %	47.3	-47.0	-4.0	102.0
Units	1.00	85.0 %	49.2 %	46.1	-61.0	-9.5	109.5
	1.25	86.8 %	48.0 %	59.1	-57.0	-9.5	133.0
	1.50	87.3 %	47.8 %	60.1	-58.5	-5.0	132.2

Table B.6 shows the breakout of the objective function implemented in the production plan. Note that $\overline{P}$ has little influence on the experienced *Holding Cost* and *Under Production* penalty. Similarly, the planned utilization of resources is not significantly influenced as $\overline{P}$ increases. The *Tardiness Penalty* increases when $\overline{P}$ increases from 1.00 to 1.25, but only slightly from 1.25 to 1.50. This initial increase in the *Tardiness Penalty* is due to the introduction of an "additional" penalty for non-delivered units when $\overline{P} > 1$ resulting in the prioritization of the shipment of "late" deliveries. As expected, the penalty for *Unfulfillment* increases as $\overline{P}$ increases.

Table B.6. Parametric Sensitivity Analysis: Analysis of costs and penalties in the production plan with respect to changes in the unfulfillment penalty parameter, $\overline{P}$

KPI / $\overline{P}$	Holding Cost	Under Production	Tardiness Penalty	Unfulfillment Penalty	Total Penalty
1.00	0.71 %	1.97 %	5.39 %	4.89 %	12.96 %
1.25	0.71 %	2.00 %	6.01 %	5.51 %	14.23 %
1.50	0.71 %	1.99 %	5.99 %	6.50 %	15.18 %

B.1.3 Earliness Incentive

Table B.7 shows the delivery KPIs as ϵ increases. As ϵ increases, the *Lateness* decreases, more-so in the *Measure* of Dollars than in Units. This is due to the definition of the objective function in the planning model being weighted in the *value* of the finished goods associated with each customer order. The percentage of *Dollars* which are delivered *On Time* also decreases as ϵ increases. This is a result of a prioritization in delivering high-value finished goods as early as possible in cases

where the per-period incentive of an early high-value end item is greater than per-period penalty of delivering a low-value end item late (relative to their respective requested due dates). It should be noted that changes to the parameter ϵ has little to no influence on the expected costs associated with the production plan or the expected utilization of resources in the test scenarios. This is due to the intention of the parameter ϵ: to provide a *small* incentive for early deliveries, as to front-load production activities in the production plan, when possible.

Table B.7. Parametric Sensitivity Analysis: Analysis of planned delivery performance based on each model and its corresponding assumptions with respect to changes in the earliness reward parameter, ϵ

		% of Demanded		Lateness (Hours)			
		Delivered	On Time	Mean	25th pc.	Median	75th pc.
Measure	ϵ						
Dollars	0.1%	86.9 %	43.5 %	48.1	-45.0	-4.0	98.0
	1%	87.6 %	42.8 %	47.2	-44.0	-5.0	104.0
	10%	86.8 %	42.5 %	46.5	-50.0	-9.0	93.0
Units	0.1%	85.7 %	48.3 %	55.2	-63.0	-6.0	116.5
	1%	86.8 %	48.0 %	59.1	-57.0	-9.5	133.0
	10%	86.6 %	48.5 %	56.4	-64.2	-12.0	133.5

B.1.4 Holding Cost

Table B.8 shows the delivery KPIs as the parameter **h** increases. As **h** increases, the delivery KPIs worsen, due to a prioritization in the minimization of holding costs rather than timely deliveries. This behavior can be seen in the decrease in the percentage of *Delivered* and *On Time* demands as **h** increases. Further, the *Lateness* of shipments also increases as **h** increases.

Table B.8. Parametric Sensitivity Analysis: Analysis of planned delivery performance based on each model and its corresponding assumptions with respect to changes in the holding cost parameter, h

		% of Demanded		Lateness (Hours)			
		Delivered	On Time	Mean	25th pc.	Median	75th pc.
Measure	**h**						
Dollars	10%	87.7 %	44.9 %	45.7	-47.0	-12.0	93.0
	20%	87.6 %	42.8 %	47.2	-44.0	-5.0	104.0
	30%	87.1 %	42.1 %	48.5	-46.0	1.0	98.0
Units	10%	87.2 %	49.6 %	54.9	-60.0	-11.0	122.2
	20%	86.8 %	48.0 %	59.1	-57.0	-9.5	133.0
	30%	86.2 %	45.8 %	66.0	-56.0	4.0	140.5

The influence which the parameter **h** has on the observed objective function costs is shown in Table B.9. Note that the cost of BOTH *Holding Costs* and *Under Production* increase linearly as **h** increases. As a reminder, the coefficients associated with under production are also derived from the parameter **h**. The expected cost of *Unfulfillment Penalties* also increases with **h**, validating the trends seen in the percentage of delivered demands in Table B.8.

Table B.9. Parametric Sensitivity Analysis: Analysis of costs and penalties in the production plan with respect to changes in the holding cost parameter, h

KPI **h**	Holding Cost	Under Production	Tardiness Penalty	Unfulfillment Penalty	Total Penalty
10%	0.37 %	1.01 %	5.85 %	5.46 %	12.70 %
20%	0.71 %	2.00 %	6.01 %	5.51 %	14.23 %
30%	1.04 %	2.97 %	5.96 %	5.56 %	15.53 %

Table B.10 provides a summary into the number and size (quantity) of each planned delivery shipment and production work order. The column, *Count* is the number of unique shipments/work orders in the production plan. *Mean* and *Median* describe the quantity of units which are delivered in each shipment or produced in each work order, and *Total* is the total quantity of units shipped/produced in the production plan. All values shown are representative of the means across all 10 tested scenarios. It is shown that as **h** increases, both the *Count* of unique shipments and work orders increases, while the *Mean* quantity per shipment/work order decreases. This tendency emphasizes a production plan strategy in which more inventory exists as Work-In-Progress inventory, rather than in storage (where holding costs are accrued). Further, the column *Work Order Qty:Total* shows a decrease in production output as **h** increases, validating the results shown in Table B.9.

Table B.10. Parameteric Sensitivity Analysis: Analysis of generated shipments and work orders in the production plan with respect to changes in the holding cost parameter, h

	Delivery Shipment Qty				Work Order Qty			
h	Count	Mean	Median	Total	Count	Mean	Median	Total
10%	69.40	3.29	2.28	228.61	417.00	22.90	10.24	9547.77
20%	72.10	3.21	2.30	231.35	422.90	22.53	9.72	9526.90
30%	73.30	3.11	2.33	228.10	435.40	21.78	9.32	9481.42

Another impact observed as a result of an increase in the count of production work orders is shown in Table B.11. Note that although the total production output decreases during the planning horizon, utilization slightly increases. This is a result of the increase of time which resources are spent in setup, due to an increase in the number processing cycles in the production plan.

Table B.11. Parametric Sensitivity Analysis: Analysis of resource utilization in production plan with respect to changes in the holding cost parameter, h

h	Process	Setup	Total
10%	65.52 %	13.10 %	77.61 %
20%	65.46 %	13.38 %	77.81 %
30%	65.60 %	13.63 %	78.21 %

B.1.5 Under Production

Table B.12 shows the sensitivity of the production plan's delivery performance as a function of **H**. Similar to the influence of the holding cost parameter, **h**, an emphasis in non-delivery penalties results in a degradation in delivery performance. As **H** increases, the percentage of *Units* which are delivered *On Time* decreases. However, the total percentage of *Units Delivered* increases. These trends are less clear when considering the delivery KPIs measured in *Dollars*. The *Mean Lateness* of shipments also worsen as **H** increases.

Table B.12. Parametric Sensitivity Analysis: Analysis of planned delivery performance based on each model and its corresponding assumptions with respect to changes in the under-production penalty parameter, H

		% of Demanded		Lateness (Hours)			
Measure	**H**	Delivered	On Time	Mean	25th pc.	Median	75th pc.
Dollars	0.5	87.5 %	45.2 %	43.7	-47.0	-12.0	94.0
	1.0	87.6 %	42.8 %	47.2	-44.0	-5.0	104.0
	2.0	87.3 %	43.1 %	48.4	-46.0	-5.0	96.0
Units	0.5	86.3 %	49.3 %	51.5	-62.0	-11.0	109.5
	1.0	86.8 %	48.0 %	59.1	-57.0	-9.5	133.0
	2.0	87.0 %	47.5 %	64.8	-58.2	-4.0	135.0

As expected, the expected penalties associated with *Under Production* in the objective function increases with the parameter **H**, as shown in Table B.13. Similarly, the realized *Holding Cost* also

increases. The penalties associated with *Tardiness* and *Unfulfillment* are not clearly affected by an increase in **H**.

Table B.13. Parametric Sensitivity Analysis: Analysis of costs and penalties in the production plan with respect to changes in the under-production penalty parameter, H

KPI **H**	Holding Cost	Under Production	Tardiness Penalty	Unfulfillment Penalty	Total Penalty
0.5	0.67 %	1.05 %	5.78 %	5.57 %	13.06 %
1.0	0.71 %	2.00 %	6.01 %	5.51 %	14.23 %
2.0	0.74 %	3.91 %	5.97 %	5.55 %	16.16 %

Overall, the planned utilization (Table B.14) also increases with **H**, primarily due to an increase of time spent in *Process*. This is in contrast to the increase in utilization as a result of an increase **h**, where the increase in utilization was due to the increase of time spent in *Setup*.

Table B.14. Parametric Sensitivity Analysis: Analysis of resource utilization in production plan with respect to changes in the under-production penalty parameter, H

H	Process	Setup	Total
0.5	65.04 %	13.29 %	77.18 %
1.0	65.46 %	13.38 %	77.81 %
2.0	65.83 %	13.33 %	78.17 %

B.2 The Effects of Practical Manufacturing Policies

In this section, we evaluate the impacts of implementing several practical manufacturing policies which guide the production plan towards a schedule which meets the objectives of a scheduler/planner. These manufacturing policies are implemented through the introduction of new parameters and constraints which restrict the solution space of the Planning Model. Specifically, we evaluate two policies in the Planning Module and one policy in the Continuous-Time Scheduling Heuristic (CTSH). In addition, we introduce two additional policies related to the CTSH which can be evaluated as a part of future research efforts.

The policies evaluated in this section include a hard restriction for the earliest time for which an shipment associated with a line item can be delivered (Section B.2.1), a hard restriction in the minimum quantity associated with a work order for any part, (Section B.2.2), and a relaxation in

the CTSH which allows for the assignment of a generated work order to occur prior to its planned start time as defined in the production plan (Section B.2.3).

B.2.1 Earliest Allowable Shipments

We have referenced that the Planning Model is allowed to prioritize early deliveries for orders due far in the future over orders which may already be overdue. This behavior will occur if the reward for delivering a large-value order early outweighs the penalty for delivering a small-value order late. In order to restrict this, we introduce a new parameter, $\overline{\epsilon}$, defined as the earliest allowable delivery of any shipment, relative to the due date of that customer order, D_o. A complimentary set of constraints ensure that no orders can be delivered earlier than this point of time, $D_o - \overline{\epsilon}$:

$$\sum_{\substack{t \in T^1: \\ t < D_o - \overline{\epsilon}}} y_{ot} \quad = \quad 0 \qquad \forall \quad o \in O$$

Table B.15 shows the delivery KPIs achieved in both the production plan (*Source:Planned*) and simulated schedule execution (*Source:Actual*), as the restriction on the earliest allowable delivery time, $\overline{\epsilon}$ increases. In this sense, the Planning Model is most restricted when $\overline{\epsilon} = 0$. This view only shows the delivery KPIs, weighted in *Dollars*. As $\overline{\epsilon}$ increases, the percentage of demands *Delivered* and delivered *On Time* initially increases before decreasing, signifying that some additionally flexibility is beneficial to the performance of the production plan, but too much results in poorer performance. However, *Lateness* of shipments strictly decrease as $\overline{\epsilon}$ increases.

Table B.15. Manufacturing Policy Sensitivity Analysis: Analysis of performance in deliveries made in production plan and simulated schedule with respect to changes in the restriction of earliest allowable delivery, $\overline{\epsilon}$

		% of Demanded		Lateness (Hours)			
Source	$\overline{\epsilon}$	Delivered	On Time	Mean	25th pc.	Median	75th pc.
Actual	0	77.5 %	11.9 %	127.6	30.0	119.0	212.0
	80	84.2 %	21.8 %	124.6	11.0	94.0	223.0
	160	76.9 %	17.4 %	90.5	-15.0	74.0	188.0
Planned	0	80.9 %	36.7 %	48.9	-28.0	2.0	79.0
	80	87.6 %	42.8 %	47.2	-44.0	-5.0	104.0
	160	80.1 %	39.3 %	27.1	-67.0	-24.0	93.0

Table B.16 shows a breakout of the objective function costs incurred in the production plan as $\overline{\epsilon}$ increases. Similar to the results shown in B.15, some flexibility allotment in the framework does improve the performance of the plan, but excess flexibility results in a deterioration in performance.

Table B.16. Manufacturing Policy Sensitivity Analysis: Analysis of costs and penalties in the simulated schedule with respect to changes in the restriction of earliest allowable delivery, $\overline{\epsilon}$

KPI $\overline{\epsilon}$	Holding Cost	Under Production	Tardiness Penalty	Unfulfillment Penalty	Total Penalty
0	0.73 %	4.40 %	5.32 %	8.74 %	19.19 %
80	0.71 %	2.00 %	6.01 %	5.51 %	14.23 %
160	0.71 %	4.52 %	5.60 %	9.05 %	19.87 %

Table B.17 provides an analysis on the quantities associated with each delivery shipment and work order as $\overline{\epsilon}$ increases. As $\overline{\epsilon}$ increases, so does the *Count* of shipments, while the *Mean* quantity of units shipped in each delivery decreases. Again, the results in this table show improved KPIs when some flexibility is introduced for $\overline{\epsilon}$ before worsening, i.e. *Work Order Qty: Total*.

Table B.17. Manufacturing Policy Sensitivity Analysis: Analysis of generated shipments and work orders in the production plan with respect to changes in the restriction of earliest allowable delivery, $\overline{\epsilon}$

	Delivery Shipment Qty				Work Order Qty			
$\overline{\epsilon}$	Count	Mean	Median	Total	Count	Mean	Median	Total
0	64.30	3.54	2.17	227.61	433.30	21.27	9.78	9216.79
80	72.10	3.21	2.30	231.35	422.90	22.53	9.72	9526.90
160	71.10	2.90	2.11	205.84	412.00	21.90	10.19	9023.76

B.2.2 Minimum Production Lot Quantities

Another potentially undesirable characteristic of a solution from the production Planning Model is the recommendation of production lots with very small quantities. For example, a production plan solution may call for 10 work orders for the same part, each for a quantity of 1, when in practice a scheduler would likely prefer to release 1 work order for a quantity of 10 units. This decision will result in a potentially larger holding cost, but represents a more practical production plan. We present and evaluate a method which introduces a new set of constraints which, similar to the constraints which limit the maximum quantity associated with a production lot, will restrict

the minimum quantity associated with a production lot. A new coefficient, $0 < \mu < 1$, is introduced to ensure that the minimum production lot quantity is larger than a proportion of the maximum allowable quantity, $M_{\gamma t}$. These constraints are shown below, along with the original constraints which limits the maximum lot size constraint:

$$\begin{aligned} M_{\gamma t}\, z_{\gamma t} &\geq m_{\gamma t} \quad \forall \quad j \in J, (\gamma) \in \Gamma^j, t \in T^j \qquad (MaxProd) \\ \mu\, M_{\gamma t}\, z_{\gamma t} &\leq m_{\gamma t} \quad \forall \quad j \in J, (\gamma) \in \Gamma^j, t \in T^j \qquad (MinProd) \end{aligned}$$

where $m_{\gamma t}$ is production lot size for a process γ during period t, and $z_{\gamma t}$ is the number of setups planned to occur during period t for process γ.

Table B.18 shows the delivery KPIs (in dollars) of the production plan as the parameter μ increases. Note that the percentage of demands *Delivered* and delivered *On Time* worsen as μ increases, as a result of the restrictions which this parameter represents. As μ increases, and reaches a larger value of 25%, the performance of the production plan drops significantly.

This caused by production infeasilibilities which stem from the simplicity of the wide-sweeping constraint, such as the one described above. Consider the example of a multi-step routing containing 2 route steps and two different resource groups where the process associated with the first route-step has a maximum lot size of 100 units while the second route step has a maximum lot size of 10 units. Considering the constraints, and a value of $\mu = 25\%$, the production of this routing would be considered infeasible. The first route-step would require a production variable, u_{it}, to be between [25, 100], while the second would require [2.5, 10]. To avoid this type of infeasibility, these constraints would need to consider a minimum production lot size based on the *constraining* route-step, with respect to the maximum production lot size.

Table B.18. Manufacutring Policy Sensitivity Analysis: Analysis of performance in deliveries made in production plan and simulated schedule with respect to changes in the restriction of minimum production lot sizes, μ

Source	μ	% of Demanded Delivered	% of Demanded On Time	Lateness (Hours) Mean	Lateness (Hours) 25th pc.	Lateness (Hours) Median	Lateness (Hours) 75th pc.
Actual	0.00	88.9 %	20.1 %	168.6	31.0	144.0	279.0
	0.10	84.2 %	21.8 %	124.6	11.0	94.0	223.0
	0.25	43.1 %	15.1 %	57.4	-44.0	43.0	152.0
Planned	0.00	95.6 %	47.6 %	78.4	-31.0	18.0	143.0
	0.10	87.6 %	42.8 %	47.2	-44.0	-5.0	104.0
	0.25	48.8 %	28.4 %	-7.8	-83.0	-54.0	37.0

B.2.3 The Early Release of Dynamically Generated Work Orders

The proposed algorithm for the Continuous-Time Scheduling Heuristic (CTSH), see Section 2.7, contains several user-defined parameters which dictate the logic which translates the discretized production plan into an executable schedule. It should be noted that the procedure for evaluating policies in teh CTSH is slightly modified. Specifically, because the CTSH algorithm is implemented following the generation of a production plan, the updating of a `base_factory` into a `factory` associated with a specific `trial` occurs later in the solution procedure, i.e. following the solve of the Node Pegging Model. This provides a fair evaluation, as the production plan associated with the test instance in each trial will be identical. The updated Design of Experiments procedure is shown in Algorithm 10.

In the production plan, production decision variables, u_{it}, describes the quantity of a part, i, which should begin its production routing during period t. During the simulation of the schedule execution these production lots, (`Work Orders`) are not *released* to the shop floor, i.e. allowed to be scheduled, until their planned start times, t.

However, due to the lead time buffers inserted during the calculation of the parameters (all lead times are calculated assuming that a full production lot is run each cycle), there will be cases in which the processing times of work orders with a lot quantity less than the maximum lot size that will be less than the *planned lead time* of that work order. This leads to cases in which resources may be left idle, i.e. when they complete a production lot quicker than anticipated. Further, resource

Algorithm 10: Procedure for CTSH policy calibration Design of Experiments

Input: `num_trials`, `grid`, `UDP`

```
let result be an empty dictionary

for num_trial in range(num_trials) do
    1. Let factory be the output of generator ← this_UDP
    2. Let p be the output of preprocess ← factory
    3. Let pm be the output of planning_model ← (factory, p)
    4. Let nm be the output of pegging_model ← (factory, p, pm)

    for (trial_name, trial) in grid do
        Let key be (num_trial, trial_name)
        Update factory attributes, factory.CTSH_SETTINGS ← trial
        5. Let sim be the output of CTSH ← (factory, pm, nm)
        6. Let kpi be the output of KPI_pipeline ← (sim, pm)
        Save key:kpi → result
```

Output: `result`

idleness in the CTSH may arise due to inventory infeasibilities of down-stream work orders due to delays at upstream resources.

For these reasons, it may be beneficial to schedule some work orders prior to their planned start time such that the continuous-time schedule can take advantage of resource idleness. This will help ensure a high utilization rate throughout the shop while potentially resulting in a reduction in the delays experienced in down-stream processes.

A trade-off exists when allowing an early release of the planned work orders. When a work order is released too early, uncontrolled deviations from the production plan can occur. For example, work orders scheduled earlier than their planned start time may be allowed to consume component inventory which was planned to be used to complete another work order which may not be ready to be started (due to shortages of other component inventories). If a work order is scheduled out of sequence and consumes inventory that another work order required, there can be unforeseen delays as the latter work order would need to wait until more component inventory is available.

To evaluate this trade-off, we introduce a new parameter, $\mathbf{REL} \geq 0$, which defines the duration of time (in working hours) prior to each work order's planned start time that it can be released to the shop floor, i.e. moved from $W^0 \rightarrow W^1$. For example, a value of $\mathbf{REL} = 40$ represents a policy

in which a dynamically generated work order can be assigned to be started by a resource up to 40 working hours prior to its planned start time.

Table B.19 describes the delivery performance of a production plan **after** it has been simulated in continuous time, using the CTSH algorithm. As **REL** increases, the delivery KPIs initially improve before worsening. For example, the percentage of *Dollars* and *Units* which are *Delivered* and delivered *On Time* initially increase, then start to decrease as **REL** increases. The *Lateness* of delivered *Units*, show that an increase in **REL** is beneficial, while the *Lateness* of delivered *Dollars* shows an initial improvement, then deteriorates, as **REL** increases.

Table B.19. Manufacturing Policy Sensitivity Analysis: Analysis of performance in deliveries made in simulated schedule with respect to changes to the release date of generated work orders in the CTSH

		% of Demanded		Lateness (Hours)			
Measure	**REL**	Delivered	On Time	Mean	25th pc.	Median	75th pc.
Dollars	0	81.2 %	14.1 %	146.5	37.0	108.0	222.0
	40	84.2 %	21.8 %	124.6	11.0	94.0	223.0
	80	83.0 %	25.0 %	109.8	-3.0	93.0	198.0
	160	83.0 %	20.0 %	139.6	-9.0	133.0	269.0
Units	0	82.0 %	22.8 %	126.5	-3.2	77.0	206.8
	40	83.2 %	33.0 %	100.2	-36.0	57.0	188.0
	80	83.7 %	36.0 %	80.3	-65.5	48.0	178.5
	160	83.9 %	35.7 %	68.4	-114.0	37.0	191.8

Table B.20 shows a breakout of the realized costs following the simulation of the production plan. Note the deterioration of the performance of the simulated continuous-time schedule compared to the production plan. Overall, the *Total Penalty* of the simulated schedule initially decreases as **REL** increases from 0 to 40 hours, but then begins to increase. Each component of the costs described in the table experience an initial improvement as **REL** increases before reaching an inflection point; however, these inflection points vary for each KPI.

Table B.20. Manufacturing Policy Sensitivity Analysis: Analysis of costs and penalties in the simulated schedule with respect to changes to the release date of generated work orders in the CTSH

Source	KPI **REL**	Holding Cost	Under Production	Tardiness Penalty	Unfulfillment Penalty	Total Penalty
Plan	0	0.72 %	2.00 %	6.00 %	5.51 %	14.21 %
Actual	0	3.39 %	3.43 %	8.06 %	8.11 %	22.98 %
	40	3.47 %	3.34 %	7.74 %	7.07 %	21.63 %
	80	3.55 %	3.58 %	7.20 %	7.34 %	21.66 %
	160	3.26 %	3.70 %	9.60 %	7.75 %	24.30 %

Table B.21 shows the realized utilization following the continuous-time simulation of the production plan. It is shown that as **REL** increases, so does utilization. This increase in utilization is apparent for both *Setup* and *Process* activities. This trend contradicts the patterns shown before, and signifies that the cause of the deteriorating performance is due to infeasibilities (specifically due to lack of available component inventory) found in the simulation. These infeasibilities are a result of the cannibalization of inventory by work orders which are released earlier and are scheduled prior to other work orders which had originally been planned to be completed first.

Table B.21. Manufacturing Policy Sensitivity Analysis: Analysis of resource utilization in simulated schedule with respect to changes to the release date of generated work orders in the CTSH

REL	Process	Setup	Total
0	59.70 %	13.98 %	73.68 %
40	60.55 %	14.26 %	74.81 %
80	60.84 %	14.43 %	75.27 %
160	61.00 %	14.43 %	75.44 %

Table B.22 provides context to the *delays* which are experienced for both delivery *Shipments* and *Work Orders* in the simulated schedule, compared to the generated production plan. For each respective task: *Count* describes the number of planned tasks (measured as an average across the 10 test scenarios) which are completed during the simulated horizon, *Mean* and *Median* are the mean and median lateness of tasks in the simulation relative to their planned **start time**.

As **REL** increases, both the *Mean* and *Median* lateness of *Work Orders* decreases, while increasing for *Shipments*. This is due to a bullwhip effect in the production system; while upstream

work orders, which are not dependent on the completion of other work orders, are not affected by the unexpected infeasibilities caused by releasing work orders early, downstream work orders and end item shipments are subject to the compounded infeasibilities and delays of all the work orders they depend on. As a result, these work orders and shipments experience much more frequent and extreme delays.

Table B.22. Manufacturing Policy Sensitivity Analysis: Analysis of delays in the simulated execution of work orders relative to the scheduled production plan with respect to changes to the release date of generated work orders in the CTSH

	Shipments			Work Orders		
REL	Count	Mean	Median	Count	Mean	Median
0	66	41.2	6.5	416	47.3	29.5
40	67	47.5	7.0	416	22.4	3.6
80	68	66.0	25.5	416	3.9	-8.5
160	68	127.8	86.5	417	-23.2	-37.9

B.3 Discussion

In this appendix, we present several numerical analysis' to justify the selection of the user-defined parameters which dictate the calculation of cost coefficients and bounds for constraints in the Planning Model (PM) for the Multi-Level Capacitated Lot-Sizing Problem.

Considering the parametric calibration of the PM objective function cost coefficients, we emphasize the definition of parameters to be intuitive to users. We evaluate parameters related to the per-unit per-period tardiness penalties, per-unit penalty for demand unfulfillment, per-unit per-period reward early deliveries, per-unit per-period holding costs and per-unit penalty for under-production. We find that in defining related parameters in relation to each other, i.e. tardiness and unfulfillment penalties, has beneficial qualities in the calculation of cost coefficients.

We then evaluate the impacts which several practical manufacturing policies have on the performance of the Planning Model and CTSH. For the Planning Model, we define parameters which restrict the earliest time a shipment can be made to satisfy an order (relative to the due date of that order). We also define a parameter which restricts the minimum size of a production lot (relative to the maximum allowable size of that production lot). We find that the inclusion of these restrictions improve the production plan relative to not enforcing any restrictions. We also find that enforcing

policies which are too restrictive are also detrimental to the performance of the production plan. For the CTSH algorithm, we define a parameter which relaxes the rules of the heuristic to allow work orders from the production plan to be scheduled earlier than their planned start time. Again, we find that relaxing the restrictions in the CTSH, by allowing work orders to be scheduled earlier than stated in the plan, is beneficial. However, these benefits are only observed to a certain point, as large deviations from the production plan leads to unforeseen infeasibilities which cause delays in the schedule and leads to poorer performance.

Parameter	**Description**	**Value**
P	Delay penalty (% per week)	2.5%
$\overline{P}$	Unfulfilled order penalty factor	1.25
ϵ	Earliness incentive (% of P)	1%
h	Holding cost rate (% per year)	20%
H	Underproduction penalty factor	1
$\overline{\epsilon}$	Earliest allowable delivery (in hours)	80
μ	Minimum production lot size (% of $M_{\gamma t}$)	10%
REL	Release time of generated work orders (in hours)	40

Table B.23. Calibrated parameters and their default values as implemented in Section 3.5

We identify several additional parameters which can be calibrated in future research efforts. Specifically, related to the parameters associated with the implementation of the CTSH, the prioritization metric used for released work orders competing for resources is of interest. As implemented, existing work orders are prioritized by their planned start time, however this can be improved by leveraging information regarding the order-pegged demands which each work order is matched with. Prioritizing work orders based on metrics based on the current expected tardiness of the orders associated with each work order, the critical ratio associated with the remaining lead times of the parts produced by the work order (relative to order-pegged due dates), or by the number of dependent sales order could be promising prioritization metrics.

Further, an evaluation of the impacts of different enforcement methods of paper-routing practices should be explored. In the current CTSH, the only batches within the production lot are allowed to travel from work center to work center (*within the routing of the part*) without waiting for all batches to be completed. They are however, held as Work-In-Progress inventory at the end of the production routing until all batch complete all route-steps. An alternate practice would

restrict the batches from moving onto a next route-step until all batches had finished the current route-step. Further, we could explore the resulting performance of the schedule if no paper routing measures were enforced.

APPENDIX C

GRAPHICAL REPRESENTATIONS OF ASSEMBLIES

Product structures can be either *single-level* or *multi-level*. A single-level product describes a product which is produced in a single step. In practice, these products are simple, i.e. transformation of a raw material into a final product via forging or casting. We consider multi-level production systems in this study. Multi-level products have been represented in several ways, namely: serial, parallel, assembly, or generalized, and represent the parent-component relationships of the products defined in the applied models. A *serial* product is one with a single final product output with multiple processing steps. The output from one step in production is the sole input to the next step, resulting in a representation where each part has exactly one parent and one component. An *assembly* product structure extends the parent-component relationship such that each part has exactly one parent but can have multiple components. A *general* product structure relaxes this assumption such that each part can have multiple parents and components. See Figure C.1 for a visual representation of each of these product structures.

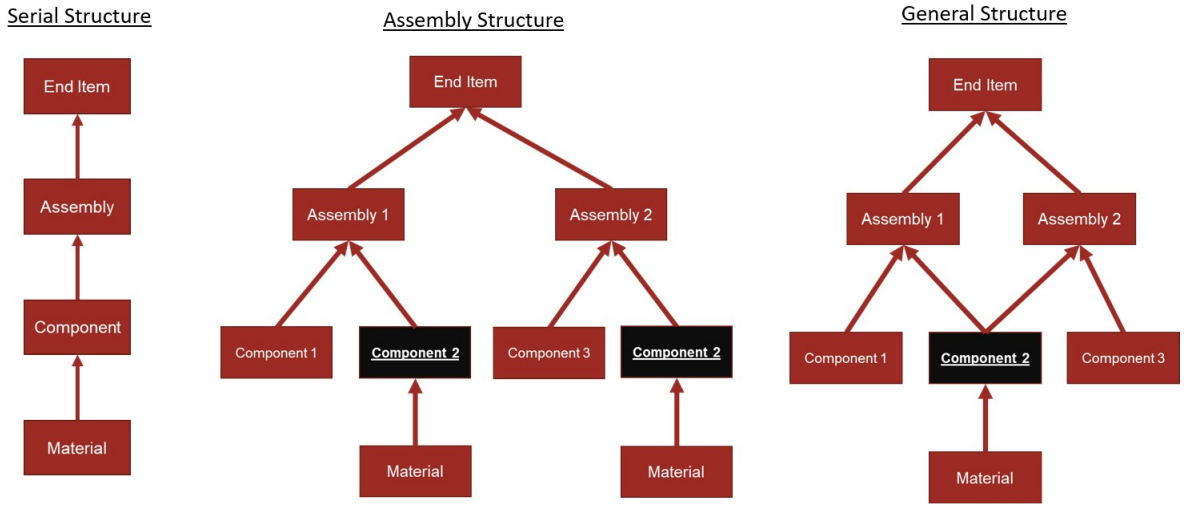


Figure C.1. Product Structures

C.1 Representation as a Graph and Implementation using Networkx

A directed graph is an obvious choice for representing BOM (and supply chain) networks, as edges are defined with a restricted direction. That means that each edge has a source and a target (materials can only flow from the source to the target). This makes sense as in the BOM data, a part with an offset level of 5 will always flow to a part with an offset level of 4. The opposite is never true. Using directed graphs allow for much more detailed analysis of the networks. We leverage the Python open-source library *Networkx* in the implementation of all network-based encodings of the BOM data. "NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks."

In the following, two types of networks that can be made using the BOM data will be introduced and discussed. Each of them has its benefits and disadvantages, but both are generated from the same data. These two type of networks will be referred to as *partID* networks and *nodeID* networks. These networks are constructed using a node-link data structure and tree data structure, respectively. In each network structure, each node is assigned a number of attributes such as costs, lead times, network-based centrality values, etc. Each edge is also assigned time attributes such as processing rate, move time, expected queue time, as well as relational attributes such as units per parent. These attributes are leveraged to create parameters and subset mappings which otherwise would not be possible.

A partID network is a representation of the BOM where each unique part number in the BOM is associated with a single node in the graph. In this network, the unique part numbers are used to name each of the nodes in the network. Each parent-child relation ship in the BOM data is represented as an edge. This type of network structure is represented using a node-link data structure. Note that each node can be the origin of **multiple** edges. This means that this node (part number) is the child of each edge's target node. In other words, each node can have multiple parents. This representation of the BOM data is essentially a network displaying the flow of materials throughout the BOM.

In graph theory, a tree is defined as a network in which no closed loops exist (acyclic). This can be compared to a cyclic graph which contains a path from at least one node back to itself. Other characteristics of trees include:

- Any tree with N nodes will have exactly N – 1 edges
- Points of connection are known as forks and the segments as branches
- Final segments are called tree leaves

Trees can take either the form of a rooted tree or a free tree. The difference between the two is the presence or not of a "root", respectively. The tree's root acts as the focal node in the graph, while every other node has a unique parent on the path towards the root of the tree. In this structure, parent-child relationships between any two parts/assemblies are connected by a directed edge, with the direction of flow moving from the child part to the parent assembly. Using this logic, for each finsihed good BOM, all of the raw materials and vendor supplied parts would be represented as tree leaves, and the top-level finished good deliverable would act as the root of the tree.

We use a nodeID network to represent BOM data for unique line item in the BOM, which becomes the root node of the tree. In this network, a nodeID indexing system is used to name each of the nodes in the network. This means that nodes with a shared part number will be duplicated each time it is found in the BOM data. Each parent-child relationship in the BOM data is represented as an edge. This type of network structure is represented using a tree data structure.

Notice how each node has exactly one edge originating from itself. This means that each node (Node ID) is the child of exactly one parent node. In other words, each node is restricted to having a single parent. This representation of the BOM data is essentially a network displaying the exact structure of the BOM.

C.2 Algorithms and Notation of BOM networks

The creation of tree-based representations leverages the implementation of a novel "NodeID" indexing system. This indexing system essentially creates a unique ID for each line item in each BOM, which defines that line item's exact location within the BOM. Having a NodeID for each location in the bill of materials allows for the encoding of a lot of valuable information. The style of the ID allows for the traversal within the BOM with only the use of the NodeID indexing system.

The Node ID works as follows. At the top level each finished good line item deliverable is assigned a unique number. For each of these orders, all of the first offset level assemblies (the

children of the top-level assembly) would be assigned a number respective to its location within its own BOM. This new number would be appended to the first number. For example, for line item "1", the first assembly will be assigned the ID, "1.1", the second, "1.2", then "1.3", and so on. For line item "2", this would be "2.1", "2.2", "2.3", etc. This process would then be repeated for each of these assemblies in each engine. For assembly "1.1", the sub-assemblies within it would be named "1.1.1", then "1.1.2", etc. For assembly "1.2", this would be "1.2.1", "1.2.2", etc. For assembly "4.4", this would be "4.4.1", "4.4.2". This process would continue until each component in the entire BOM network is associated with each of the ordered line items assigned an ID. An example of a part number at a lower network level could look like: "3.10.2.17.4", which calls the 4th child of the 17th child of the 2nd child of the 10th child of the finished good which is delivered on line item 3.

It is important to consider that this Node ID is dependant on the order of the line items within the BOM. However, because this ID is only used as an index, the exact values of each component within the ID is arbitrary and contains no value other than being consistent within each BOM and being unique across all of the factory's end item deliverables. This NodeID indexing system is valuable as it allows for quick traversal throughout a BOM and provides full visibility in the system. In the example provided above, this Node ID (3.10.2.17.4) defines the index of a line item (3) as well as the path which that part traverses on its path to the top level engine. Specifically NodeID, 3.10.2.17 is known to be the parent of 3.10.2.17.4, and 3.10.2 is the parent of 3.10.2.17, and so on. By manipulating the NodeID string, it is possible to identify all of the assemblies which any component flows through on its path to the deliverable. A similar procedure can be used to search for any assemblies components. For example, because the Node ID 3.10.2.17.4 starts with 3.10.2, then you are able to conclude that 3.10.2.17.4 is a component of 3.10.2, as well as being exactly 2 offset levels below it, since 3.10.2.17.4 is two offset levels (5) greater than 3.10.2 (3). This example also shows that each Node ID encodes the information of what offset level that line item is. Further, it is possible to find all of the components of 3.10.2 by finding all of the Node ID's in the BOM which start with 3.10.2.

C.3 Discussion on Benefits and Disadvantages

As mentioned earlier, each of these network structures has its benefits and disadvantages. The type of network which should be used depends on the nature of the problem as there are cases in which a partID network is usable while a nodeID network isn't. The same could be said the other way around. These advantages will be presented through use case examples.

When thinking about a partID network which accurately models the flow of materials throughout the BOM, some obvious use cases come to mind. One of the benefits of this structure is that it is great for any type of "bottom-up" analysis of BOM networks. For example, consider a case where someone knows that a certain component-level part number needs to be recalled due to some sort of uncovered manufacturing issues. Using this partID network, it is possible to specify a sub network as that given part number and all the parts it flows through on its path to the top level finished goods. In this case, all that needs to be done is identify the nodes that are "reachable" from the originating node. Because the network is modeled as a Directed graph, there are only a limited number of nodes it can reach. When this calculation is done for each node in the network, you have essentially defined a centrality measure of the network which defines the vulnerability of the network to any part in the BOM. Measures such as these can be leveraged when implementing heuristics requiring the prioritization of the production of parts or activities which are competing over shared resources.

One of the other advantages of a partID network is that because many of the parts which show up many times in a BOM are merged together when creating the network, the size of the network is much smaller. Having smaller networks is obviously beneficial as the smaller the network, the faster any function is that is applied to it, including the proposed optimization model.

There are some disadvantages however to using a partID network. All of these issues arise as the consequences of aggregating the multiple locations of parts (if they do show up multiple times) into a single node. In order to use the partID network structure, these data points must be aggregated somehow. In doing this, information is lost, while it is not while using the nodeID network as each location is represented as a node.

In many ways the advantages of the nodeID network is the complement to the advantages of the partID network structure. For example, this network type is great for "top-down" analysis like analyzing the contents of a top level assembly. Because of the nested nature of the tree

data structures, it is incredibly easy to partition the tree network just by specifying any node as the "root" of a sub-tree. Also, as mentioned earlier, this nodeID network is the exact structural representation of the BOM data file. No data needs to be aggregated in order to build the network, and as a result there is no lost information. However, this results in the network being larger, in some cases nearly twice as large as the partID network counterpart (for any finished good).

C.4 Extensions to Graph Implementations

Although only two types of networks have been described here, there is potential for many others. Many of these can be used to further research the impacts of events and entities on supply chain networks. For example, consider a network structure in which the vendors are described as nodes in the network. Another potential extension would be one in which each of the nodes of this supply chain network was positioned on a geospatial mapping to represent its location on the actual globe. This could also be done with suppliers, along with encoding transportation times/costs along edges across the map. From here, the supply chain could be optimized, simulated as well as analyzed under what-if scenarios.

APPENDIX D

APPROACHES FOR SCHEDULING UNDER UNCERTAINTY

In this Appendix, we provide a brief literature review pertaining to approaches to addressing scheduling subject to uncertainty. Specifically, we extend the literature review of Chapter 5 by providing a more in-depth discussion of the different approaches to scheduling under uncertainty. Further, considering specifically proactive scheduling approaches, we provide a review of scheduling objective functions which have been presented in the literature, with a focus on robust scheduling objectives. This literature review was conducted in preparation of the development of Revenue Management applications for the scheduling frameworks, subject to uncertainties derived from the existence of contingent demands, as presented in Chapter 5.

Approaches that address scheduling under uncertainty are classified as reactive, proactive or as hybrid of the two. Reactive scheduling (also referred to as dynamic or online scheduling) approaches revise an existing schedule in real-time throughout its execution as unexpected events occur. Vieira et al. [179] develop a framework to describe rescheduling in manufacturing systems and identify rescheduling environments, rescheduling strategies, rescheduling policies and rescheduling methods as the key modeling dimensions in literature. In their framework they define the following: the rescheduling *environment* identifies the set of jobs that need to be scheduled, making a distinction between static and dynamic environments. A static environment has a set of finite jobs where a dynamic environment has an infinite set of jobs. The rescheduling strategy describes whether or not production schedules are generated. Dynamic strategies result in the identification of an optimal disputing rule to implement for a set of jobs, rather than generating a schedule. A predictive-reactive strategy schedule generates and maintains a schedule. Under the umbrella of predictive-reactive strategies, rescheduling policies defines when to update the existing schedule. Finally, the rescheduling method specifies how to reschedule. The authors make note that there may be multiple rescheduling policies and/or methods that can be used depending on the current conditions of the manufacturing system.

Research objectives in this field concerns the identification of optimal rescheduling policies and methods given a scheduling environment. This is achieved by answering research questions: *when* should a rescheduling activity be triggered and *how* should the reschedule be implemented.

Rescheduling triggers are classified as either periodic or event-driven (or a hybrid of the two). In a periodic policy, the rescheduling problem is decomposed into a series of static problems, solvable using classical scheduling procedures. A periodic policy is defined by a *rescheduling frequency*, specifying how often the schedule should be revised. The *rescheduling period* acts as the inverse to the scheduling frequency, and defines the amount of time between consecutive reschedules [179]. The rescheduling period is typically a fixed duration throughout the scheduling horizon, i.e. trigger a reschedule at the beginning of each shift/day/week. Event-driven rescheduling is triggered in response to a disruptive event or the realization of an uncertain parameter.

Muhlemann et al. [121] investigate how the frequency of rescheduling in a periodic rescheduling policy affects the performance of the system where machine breakdowns may occur randomly and processing times exhibit variability. At the beginning of each rescheduling period, a static schedule for the existing jobs is generated using a dispatching rule. The authors observed a deterioration in schedule performance as rescheduling frequency increases. Vieira et al. [178] extend the work of Muhlemann et al. to a parallel machine system subject to sequence dependent set-ups and find that reducing the rescheduling frequency reduces the number of observed set ups. Sabuncuoglu and Karabuk [149] consider event-driven rescheduling policies in multi-resource flexible manufacturing system with random machine breakdowns and processing times. They find that it is not always beneficial to reschedule in response to every machine breakdown, citing that the benefits of more frequent scheduling become marginal after a certain number of revisions. Ramen et al. [138] study a similar problem, but implement a rescheduling methods after a certain number of random events. Both come to the conclusion that event-driven rescheduling is better than periodic rescheduling.

Rescheduling methods are classified as either full- or partial-reschedules. In full-rescheduling, any time a rescheduling trigger occurs, all of the jobs are rescheduled using the most up-to-date information available to the scheduler. Partial rescheduling, unlike total rescheduling, aim to ensure continuity in the existing schedule by emphasizing schedule stability.

Several partial rescheduling methods include right-shifting, match-up rescheduling or partial repair rescheduling. Right shifting rescheduling is a simple dispatching method which *right shifts*

jobs that are due to be scheduled after a disruptive event occurs. For example, in the case of a machine breakdown, the right-shift method would recommend executing the existing schedule sequences, offset by the machine repair time. Another method proposed by Bean et al. [19], called match-up rescheduling, aims to adjust the upcoming portion of the schedule in reaction to a disruptive event. The distinctive quality of match-up rescheduling is that the post-disruption schedule at some point will "match-up" with the pre-disruption schedule.

Sawik [152] implements a mixed-integer linear programming solution framework in a dynamic, make-to-order manufacturing flexible flow shop environment, comparing the performance of full-rescheduling method compared to a partial-repair to the existing schedule. The scheduling objective is to dynamically assign/reassign customer orders with various due dates to planning periods with limited capacities while minimizing the number of tardy orders and inventory costs. Disruptive events include customer modifications that increase/decrease processing times, order cancellations or changes to the requested due dates. Sawik compares a full reschedule policy to a restrictive partial rescheduling policy which only reschedules a subset of jobs awaiting raw materials. The author concludes that partial rescheduling reduces CPU-time in finding solutions while maintaining near optimal performance compared to full rescheduling. Yan-hai et al. [194] consider the flow shop rescheduling problem and implement a full-rescheduling policy after the arrival of randomly generated rush orders. They develop a novel Ant Colony Optimization meta-heuristic and use a weighted mean flow time as the objective function where the weights for newly arriving rush orders are much greater than the original jobs. The authors suggest the proposed algorithm to be implementable and effective while considering other types of disruptive uncertainties such as order cancellations, material shortages and machine breakdowns.

A common theme in reactive scheduling emphasizes the problem-specificity of each environment. To overcome this issue, it is recommended that multiple rescheduling policies and methods should be tested to identify the best performing implementation. For literature reviews in reactive scheduling approaches we refer the reader to Chaari et al. [39], Ouelhadj et al. [127], Sabuncuoglu et al. [148].

It should be noted that strictly reactive scheduling approaches do not directly consider the uncertainties it faces while generating schedules, but rather define the optimal policies which should be implemented while reacting to the realization of those uncertainties [148]. While this does accommodate for considerable flexibility in the schedule to compensate for unforeseen system dis-

turbances, these strategies lack the global perspective provided by proactive scheduling approaches. For this reason, these approaches are typically reserved for problem settings which are subject to highly perturbed environments characterized by extremely disruptive and/or frequent uncertainties.

Proactive scheduling considers future disruptions while generating schedules in an effort to hedge against potential disruptions and parametric uncertainties. This approach is more synonymous with robust or stochastic optimization and makes use of historical data and forecasting techniques to derive scheduling decisions [102]. The implementation of proactive scheduling approaches typically occurs during the generation of a *preschedule*. Although a preschedule is unlikely to be executed, it serves as the basis for planning supporting activities, establishing commitments with employees, suppliers and customers, and making other revenue management decisions such as order acceptance/rejection or new customer pricing and/or due date quotation [102]. Scheduling objectives in proactive approaches incorporate the preferences of the firm and can be classified based on the following distinctions: preference of schedule stability vs performance, objectives based on reward vs regret, and attitude towards risk (i.e. risk-neutral vs risk-averse).

D.1 Solution vs Model Robustness

A schedule is considered *solution robust* if it's performance remains close to optimal for all scenarios. It is considered *model robust* if its schedule remains almost feasible for all scenarios. Solution robustness objectives typically minimize the deviation of performance metrics across the uncertainty set. Model robustness represents a stable schedule, requiring minimal adjustments deviation from the original preschedule and the eventual executed schedule [97].

Schedule stability objectives can be achieved through the introduction of flexibility into the solution and are classified as either temporal or sequential. Temporal flexiblities are incorporated through the insertion of idle times to act as a buffer against adverse realizations of uncertain parameters or events. Sequential flexibility is achieved through the characterization of group assignments, in which families of schedules are grouped into permutable sequences. In the face of disruptions, the sequence of execution within the ordered group can be modified, without adjusting the original solutions. By switching from one solution to another when disruptions occur, the scheduler can control the performance degradation. The preschedule in this approach is optimized based on the worst-case outcome for each job within its own group assignment.

Mehta and Uzsoy [113] study a single-machine environment subject to random machine breakdowns with job release times. The scheduling model's objective KPI is the minimization of maximum lateness, with the robust objective of minimizing the deviation of job completion times between the planned schedule and the realized schedule. The authors develop a two-stage solution to the problem where the first step determines an optimal sequence to $Lmax$ problem, and the second step inserts idle buffer times in the schedule to minimize worst-case completion time deviations. It is shown that the insertion of idle times in the preschedule provides significant resilience to disruptive events without significant deterioration of the primary performance measure. Gao and Fox [58] consider the single machine scheduling problem subject to random machine breakdowns and down times with the objective to minimize the sum of job tardiness cost, inventory costs and idleness costs. The authors develop a temporal protection policy that adds a buffer to job processing times and release times (to a sequence created using the Jackson Algorithm to minimize maximum lateness) and show that this approach reduces the observed interruptions to the original schedule, and reduces costs by up to 60-80% compared to a schedule with no temporal protections. Lambrechts et al. [95] consider the use of time slack-based techniques in the resource-constrained project scheduling problem, subject to random machine breakdowns and repair times. Given an initial project schedule, explicit idle time is inserted in front of the starting times of activities to improve robustness. This buffering method absorbs potential disruptions with an objective to minimize the expected instability costs without exceeding project deadlines. In cases which disruptions cause infeasibilities, the authors propose a reactive rescheduling procedure to repair the schedule.

Artigues et al. [6] take a different approach in introducing flexibility to a robust scheduling problem. The authors consider a single machine problem as a sub-problem of a job shop environment with job release dates and deadlines. An ordered group assignment technique is proposed in which families of schedules are grouped into permutable sequences allowing for flexibility in the face of disruptions. A polynomial time algorithm is developed to evaluate the worst case completion time for any operation within each assignment group. The authors maximize flexibility in the schedule by minimizing the number of assignment groups and maximizing the number of characterized sequences, subject to precedence constraints. Briand et al. [30] consider the single machine scheduling problem with uncertain release dates, due dates and processing rates. It is assumed that the relative order of these parameters are known, i.e. job j is known to be release before job

k. The authors show through the use of the dominance theorem, proposed by Erschler et al. [55], that a set of dominant sequences can be characterized and evaluated for a worst-case performance while avoiding the complete enumeration of the sequences.

D.2 Reward vs Regret

Solutions which consider *reward* aim to optimize for the performance of the realized schedule. Solutions which consider *regret*, or sub-optimiality, optimize for the difference between the realized performance and the optimal solution of the realized scenario, given perfect information.

Goren and Sabuncuoglo [64] study a single machine manufacturing environment subject to processing time variability and random machine breakdown using known probability distributions to characterize uncertainty. The authors consider two robustness measures (expected total flow time and expected total tardiness), and three stability measures (sum of the squared differences and absolute differences of the job completion times and the sum of the variances of the realized completion times). They identify special cases for which the measures can be easily optimized. For example, if no machine breakdowns are present, the authors show that sequencing the jobs according to a non-decreasing order of job processing time variance is optimal when considering the sum of squared differences and absolute differences of job completion times. In non-special cases, the problem is analytically intractable for all measures. To overcome this, the authors develop a simulation-enabled beam search branch-and-bound heuristic and find that this method requires reasonable computational times, and cite an extension to multi-machine environments as a direction of further research.

D.3 Risk-neutral vs risk-averse

Risk neutral objectives optimize for the expected realized reward or regret, while risk averse objectives hedge against the worst-cases of reward or regret. Risk-averse objectives rely on robust optimization techniques. General robust optimization models were first developed as early as the 1970s. It was not until the mid-1990s following a series of breakthroughs in theory and methodology was it that these models began to be applied to Operations Management areas such as inventory management, production planning, revenue management, etc. [102, 110]. Risk-averse tendencies are

most commonly represented through optimization of the worst-case performance of the preschedule, i.e. minimizing the maximum penalty or maximizing the minimum revenue.

Dainels and Kouvelis [45] are cited as one of the first applications of robust optimization to the scheduling problem. They implement a scenario-based approach in a single-machine environment subject to uncertain processing times. The scheduling objective was to determine the schedule which minimizes worst-case degradation (min-max regret) of total flow time of all jobs. The authors show that only a finite number of scenarios are needed to determine the worst-case absolute deviation of a sequence, even in the case that processing time distributions are unknown. Specifically, only the extreme points (upper and lower bounds) of processing times of each job are required to calculate the worst-case deviation, which can be accomplished in polynomial time. The authors develop a branch and bound algorithm, and several relaxation heuristics that utilize the worst-case evaluation procedure, and compare their solutions to the Shortest Expected Processing Time (SEPT) solution. They show that the SEPT heuristic performs poorly when considering the min-max regret robust objective compared to their proposed algorithms.

Lu et al. [108] study the robust single machine scheduling problem (RSMSP) subject to uncertain processing times and uncertain sequence dependent setup times, where uncertain parameters are represented by interval data. The authors reformulate the problem as a robust traveling salesman problem (RTSP) to obtain a robust sequence that minimizes the worst-case regret of makespan. The authors leverage a property of the RTSP reformulation to identify worst-case scenarios, and implement a simulated-annealing based local search algorithm to find optimal sequences for practically sized problems. An analysis of the comparison of implementing a worst-case robust schedule compared to the optimal expected-value sequence shows that the larger the data uncertainty is, the greater regret for adopting the optimal expected-value sequence in the worst-case scenario.

Yue et al. [198] study the RSMSP with uncertain job due dates where the decision maker objective is to minimize the worst-case maximum tardiness over all jobs. Taking a scenario-based representation of an interval uncertainty, the authors show that only n scenarios are required to identify the worst-case scenario, where n is the number of jobs. For each sequence, x, the authors define the finite set of worst case scenarios $U = \{d^i, i = 1, 2.., n\}$, where in scenario d^i, the due date of job $j = i$ is the lower bound in it's interval, $\underline{d_j}$, and all other jobs $j \neq i$ is the upper bound, $\bar{d_j}$. Using this property, a solvable MILP is developed and a robust dominance heuristic rule is proposed

to solve the problem. This heuristic method is shown to be feasible for large-scale problems (up to 500 jobs), and is applied to in the context of an MTO manufacturer of an iron-making process.

Wu et al. [189] consider a two-machine flow-shop environment with the objective to minimize makespan, subject to scenario-dependent processing times. The robustness criterion of interest in this study is absolute robustness, i.e. determining the permutation which minimizes the maximum makespan associated with any scenario. Each operation stage is composed of a single machine, and no idle time is allowed in the first stage. A branch-and-bound algorithm is developed and compared to 12 heuristics and 12 variants of a cloud-based simulated annealing algorithm. However, the proposed solution methods were only tested in a problem with 2 scenarios and only up to 12 jobs.

Silva et al. [157] investigate algorithms that solve the RSMSP with processing time uncertainties that take any value in an interval set. The robustness criterion minimizes worst-case total tardiness of any scenario. A controlling parameter, γ, which defines the size of the uncertainty interval allows for a flexible definition of decision-maker risk-aversion. In cases where a decision-maker is more risk-averse, a larger uncertainty set is chosen. The authors compare two solution methodologies: a combination of an MILP formulation with row-and-column generation algorithms, and a robust branch-and-bound algorithm which leverages dominance rules concerning precedence relations between jobs to define additional constraints in an effort to reduce the number of sequence-position variables. The authors show through computational results that the additional dominance-rule constraints were "fundamental" for a good performance of both the branch-and-bound and MILP formulation algorithms.

One of the downsides to risk-averse objectives which optimize on the worst-case is that over-conservatism dominates decision making. Other objectives functions such as beta-robustness, Pareto robustly optimal solutions, Conditional-Value-at-Risk, Value-at-Risk address this by removing emphasis from the worst-case solution. Each of these alternative objective functions will be described below.

Wang et al. [182, 183] discuss the job-shop scheduling problem with makespan as the performance criterion subject to uncertain processing times. Uncertainty is represented as a set of discrete scenario sets with no knowledge on the probability of each scenario. In [182] the authors establish a new robust optimization model which considers a set of bad scenarios rather than only the worst-case scenario. A reference standard, T, used to evaluate the solutions rather than considering the

schedule performance. The threshold-based bad-scenario set is defined as the set of scenarios, $\lambda \in \Lambda$ in which a schedule performs worse than T. The authors define the penalty of bad scenarios as the square of the degree of performance degradation past T, $[C(s, \lambda) - T]^2$, where C is the makespan of schedule s in scenario λ. No penalty is associated with scenarios that observe a makespan less than T. The robust objective is then to minimize the sum of penalties across all scenarios. The authors develop a combined simulated-annealing and tabu-search local search algorithm leveraging a problem-specific neighborhood construct uniting multiple single-scenario neighborhoods. In [183] extend this problem to case in which there is no known reference standard, T, arguing the case in which a specified standard performance level is too high, that no penalty will be encountered leading to sub-optimal performance. To solve this problem, a two-stage framework is developed which simultaneously minimizes the bad-scenario threshold, T, and the measured penalty of the resulting penalty on the bad-scenario set. The proposed framework defines a reasonable value for T as any value in the interval $[EC^*, WC^*]$, where EC^* is optimal expected-case performance among all possible scenarios, and WC^* as the optimal worst-case performance among all possible scenarios.

Daniels and Carrillo [44] study a single machine environment with a total flow time performance measure and uncertain processing times. Uncertainty is represented within a set of discrete processing time scenarios with a known probability of realizing each scenario. The authors define a general β-robustness measure that describes the likelihood, $\beta \in [0, 1)$, of achieving a system performance level no worse than a specified target level, T. A β-Robust Scheduling Problem (β-RSP) is formulated which identifies the scheduling sequence which maximizes the likelihood of achieving a flow time performance no greater than T. Compared to robust scheduling models which optimize considering only the worst-case scenario, the β-robust scheduling model can be tailored to reflect the level of risk which an individual is willing to accept. The authors also formulate the β-Robust Scheduling Problem with Variance Reduction (β-RSPVR), in which the uncertainty in processing time can be reduced through the application of an additional limited resource.

Kasperski et al. [84] study the identical parallel machine scheduling problem to minimize makespan with uncertain processing times, represented as a bounded set of discrete scenarios, similar to [83]. This paper generalizes the robustness criterion from a worst case optimization by using the *ordered weighted averaging aggregation* operator (OWA), proposed by Yager et al. [193]. This operator allows a decision maker to specify their attitude towards risk, specifically by assessing

a weighting, α, defining the importance of the performance criteria in both the worst-case scenario and best-case scenario:

$$OWA_{\omega} = (1-\alpha)\,\underline{C}_{max} + \alpha\,\bar{C}_{max} = H_{\alpha}$$

This representation is a special case of the OWA operator, known as the Hurwicz criterion, and acts as a compromise between the best (optimistic) and worst (pessimistic) cases. Another special case of the Hurwicz criterion exists when $\alpha = 1$, in which case H_{α} is equivalent to absolute robustness. The authors proposed an MIP-based approach that provides an optimal schedule to this problem. It is also shown that the generalized OWA can be solved in pseudo-polynomial time, however no algorithm is provided.

A critical limitation of the implementations OWA operator, including the Hurwicz criterion, is that these do not take into account probabilistic information as to the likelihood of each scenario. When a probability distribution of a discrete scenario set is known, or can be estimated, stochastic scheduling models can be considered. Typically, the expected solution performance of a schedule is optimized under this assumption, representing long-run performance optimization. However other criterion exist which allow the incorporation of risk-averse behaviors, such as *value at risk* (VaR) and *conditional value at risk* (CVaR). Both criterion allow the decision maker to specify an attitude towards risk, α. For the VaR, this represents optimization of the outcome (scheduling KPI) in the α-quantile scenario given a fixed scheduling solution. Compared to the minimization of the maximum performance measure over all scenarios, as in the worst-case robust optimization, VaR minimizes the maximum performance measure over the subset of scenarios with an aggregate probability of at least α. In other words, the objective becomes identify the sequence with the smallest possible upper bound on the random performance measure that will be exceeded with at most a α probability [9].

A limitation of VaR is that it does not consider the outcome in the top $(1-\alpha)$ percentile outcomes, a.k.a. the tail of the distribution. In the case that performance degradation in those unaccounted scenarios are extreme, the decision maker is vulnerable to highly sub-optimal realized performance. The ideal user of the VaR criterion is a decision-maker whom is concerned with

strictly concerned with the frequency of undesirable outcomes. CVaR, introduced by Rockafellar and Uryasev [144], addresses these tail-region scenarios and aims to optimize the expected value of the random performance measure of the scenarios which fall in the top $(1 - \alpha)$ upper bound. This criterion is advantageous as it simultaneously optimizes both the variance and the expectation of the performance measure. Another superior quality of the CVaR is that when the underlying deterministic decision making problem is represented as an LP, then so can the CVaR. However, because the VaR is non-convex and non-smooth, binary variables need to be introduced. This however is not of critical importance in the scheduling problem, as a majority of the underlying deterministic scheduling decision variables are binary.

Sarin et al. [151] and Atakan et al. [9] are among the first to propose the implementation of VaR and CVaR in a scheduling problem, specifically with total weighted tardiness as the scheduling objective. Sarin et al. formulate a scenario-based MIP to minimize CVaR for total weighted tardiness in the single-machine scheduling problems with uncertain processing times. The authors implement a local search algorithm capable of solving problems of moderate size (15 jobs, 400 scenarios) in 360 seconds and large problems (100 jobs, 800 scenarios) in 9000 seconds. Solutions derived using the CVaR of total weighted tardiness are shown to greatly reduce performance variance while marginally increasing the expected performance (compared to solutions optimizing the expected total weighted tardiness). The formulation of an application in an identical parallel machine environment is also provided. Atakan et al. develop a generic risk-averse stochastic programming model using the VaR of total weighted tardiness with uncertain processing times. They propose a Lagrangian relaxation-based scenario decomposition method to obtain lower bounds and a stabilised cut generation algorithm to solve the Lagrangian dual problem. A computation analysis shows the effectiveness of the proposed algorithm in salving large-scale problems (30 jobs, 500 scenarios). Kasperski et al. [85] considers the specific class of RSMSPs discussed by Sarin et al. and Atakan et al., and provide a number of positive and negative complexity results.

APPENDIX E

EXAMPLE OF CONTINGENT DEMAND SCHEDULING

In the following, we present a simplified instance of a scheduling problem, subject to contingent demands, as described in Chapter 5. In this example, the backlog consists of 3 jobs: $\{i_1, i_2, i_3\} \in I$, referred to as Job 1, Job 2 and Job 3 respectively. Job 1 and 2 have already been accepted by the customer $\{i_1, i_2\} \in I^A$, and both have a probability of acceptance $a_i = 1$. Job 3 is contingent, i.e. $\{i_3\} \in I^C$, and has a probability of acceptance of $a_3 = 0.75$. The desired delivery dates of jobs 1,2 and 3 are time periods 2,3 and 5, respectively. Job 1 and Job 2 are made up of material A and Job 3 is made up of material B. For simplicity, all jobs have an identical tardiness penalty $c_i = 1$, size $\alpha_j = 1$, and earliest acceptable delivery date, $e_i = 1$ (see Table E.1). There are two unrelated parallel machines, $\{j_1, j_2\} \in J$, referred to as Resource 1 and Resource 2 respectively, available to process the 3 jobs. The processing time required for each machine to complete one unit of materials A and B are shown in the Table E.2.

Job Name	Material	Job Set	d_i	s_i	a_i	c_i	e_i	α_i
i_1	A	I^A	2	0	1	1	1	1
i_2	A	I^A	3	0	1	1	1	1
i_3	B	I^C	5	1	0.75	1	1	1

Table E.1. Contingent scheduling example - job characteristics

	Machine	
Material	M1	M2
A	2	3
B	-	4

Table E.2. Contingent scheduling example - machine characteristics

The realized outcome of Job 3 will not be known until after the first time period has concluded. Because there is only one contingent job in the backlog, there exists only two scenarios. Let Scenario

1 be the scenario where Job 3 is accepted by the customer and Scenario 2 be the scenario where Job 3 is declined by the customer. Regardless of the outcome of Job 3, the scheduler must decide on the most immediate portion of the schedule now. If the scheduler waits to make their decision, then they will experience resource under-utilization of a perishable capacity, leading to non-optimal performance.

The scheduler is faced with the decision of whether or not to hedge against the possibility that Job 3 is eventually accepted by the customer. It is trivial that Job 1 has priority and should be scheduled to be completed first by Resource 1. This leaves the scheduler with the decision to either schedule Job 2 to be completed after Job 1 on Resource 1 or to be started immediately on Resource 2. Lets call these decisions Plan A and Plan B, respectively. Job 3 can only be processed by Resource 2, and will be started at time period 1 if the scheduler goes by Plan A, or will be started in time period 3 (after Job 2 is completed) in Plan B. Table E.3 shows the associated completion times and profits of each job in each realized outcome.

Metric	Completion Time				Penalty			
Scenario	1	1	2	2	1	1	2	2
Plan	A	B	A	B	A	B	A	B
Job 1	2	2	2	2	0	0	0	0
Job 2	4	3	4	3	1	0	1	0
Job 3	5	7	-	-	0	2	-	-
Total					1	2	1	0

Table E.3. Contingent scheduling example - schedule realizations

We know that because the probability of acceptance of Job 3 is $a_3 = 0.75$, therefore the probability that Scenario 1 occurs is $q_1 = 0.75$, and Scenario 2 is $q_2 = 0.25$. We can evaluate and solve for the optimal decision by comparing the expected penalty (or profit) of each plan:

$$E[P_A] = q_1 P_{A,1} + q_2 P_{A,2} = 0.75 * 1 + 0.25 * 1 = 1$$

$$E[P_B] = q_1 P_{B,1} + q_2 P_{B,2} = 0.75 * 2 + 0.25 * 0 = 1.5$$

From this assessment, it appears that enacting Plan A is optimal for the decision maker. Taking a risk and scheduling Job 2 in a locally sub-optimal position results in the globally optimal decision.

However, this assessment changes if the probability of acceptance of Job 3 changes from $a_3 = 0.75$ to $a_3 = 0.25$. The evaluation now becomes:

$$E[P_A] = q_1 P_{A,1} + q_2 P_{A,2} = 0.25 * 1 + 0.75 * 1 = 1$$
$$E[P_B] = q_1 P_{B,1} + q_2 P_{B,2} = 0.25 * 2 + 0.75 * 0 = 0.5$$

In this case, Plan B is optimal. However, implementing this assessment requires an accurate approximation of each contingent jobs' probability of acceptance. In the case that this is not available, it is possible to implement robust decision making assessments to determine the optimal decision. For example, when considering a robust decision making objective of minimizing the maximum realized penalty across all scenarios, the assessment becomes:

$$P^* = argmax(WC[P_A],\ WC[P_B])$$
$$WC[P_A] = \max(P_{A,1},\ P_{A,2}) = \max(1, 1) = 1$$
$$WC[P_B] = \max(P_{B,1},\ P_{B,2}) = \max(2, 0) = 2$$

Again, the optimal decision is Plan A. Notice in this assessment, estimating the probability of acceptance of the contingent jobs is not required.

Through this example we have shown several things: 1) the presence of contingent demand introduces complexities in decision making which require a scenario-based formulation, 2) the optimal decision in the optimization of the expected value is sensitive to the probabilities associated with each scenario, 3) this particular problem is relevant in use-cases where eligibility constraints of machines/jobs are a major factor and 4) different decision making objectives can result in unique optimal decisions.

APPENDIX F

FUTURE RESEARCH DIRECTIONS FOR THE CONTINGENT SCHEDULING PROBLEM

F.0.1 Extension to the Order Acceptance & Scheduling Problem

In the following, we develop the formulation of the OA&S problem, subject to contingent demand, and describe the computational experiments we would implement to evaluate the effectiveness of the proposed Revenue Management framework. We develop the OA&S problem as the first step in the sequential development of series of Revenue Management applications culminating in the Simultaneous Pricing, Due-Date Setting and Scheduling problem (SPDSP). The OA&S problem (also referred to as Scheduling with Rejection problem) extends the contingent demand scheduling problem presented in this chapter by allowing the decision maker the ability to chose which jobs to allow into the demand backlog.

In the problem we consider, we define a new subset of jobs, I^N, as the set of jobs from *new* customers which have not been offered quotes for their requested product enquiries. We assume that each new customer has a desired price point and due-window, and is unwilling to award their demand to the firm if these requirements are not agreed to. We also assume, that if the firm offers a quotation bid to the new customers which meet their requests, the customer will accept the bid with a probability of 1, i.e. $a_i = 1$. Given these assumptions, the decision-maker is able to decide if this new job should be added to the demand backlog, considering the existing contingent demand.

We define the binary decision variable, z_i, to take a value of 1 if the decision maker chooses to offer a quotation to the incoming order, i.e. *accept*, and 0 otherwise. All new jobs, $i \in I^N$, which are accepted by the firm are treated similarly to the jobs from the accepted job set, $i \in I^A$, since it is known that the customer will also accept the terms of the quotation offered. We also define the set, $I^{AN} = I^A \cup I^N$, as the union of the accepted jobs and new arriving jobs. Finally, the parameters, R_{ijk}, are modified to consider the realized *profit* of each job:

$$R_{ijk} = r_i - h_i \ \max\big(0, e_i - t_{ijk}\big) + c_i \ \max\big(0, t_{ijk} - d_i\big)$$

where r_i is the revenue associated with each job, i. The objective of the Order Acceptance and Scheduling problem becomes one to maximize the expected profit across considered scenarios. In this problem, profit is calculated as the sum of revenues minus the penalties associated with scheduling the jobs in the backlog. The formulation of the hybrid-bucket Order Acceptance and Scheduling problem becomes:

Formulation **(OA)**

$$\max \quad \sum_{j\in J}\left(\sum_{k\in K^F}\sum_{i\in I^{AN}} R_{ijk}\, x_{ijk} + \sum_{k\in K^{NF}}\sum_{s\in S}\sum_{i\in I} q_s\, R_{ijk}\, y^s_{ijk}\right)$$

s.t.

$$\sum_{j\in J}\Big(\sum_{k\in K^F} x_{ijk} + \sum_{k\in K^{NF}} y^s_{ijk}\Big) = 1 \quad \forall \ i\in I^A, s\in S \tag{1}$$

$$\sum_{j\in J}\Big(\sum_{k\in K^F} x_{ijk} + \sum_{k\in K^{NF}} y^s_{ijk}\Big) = z_i \quad \forall \ i\in I^N, s\in S \tag{2}$$

$$\sum_{j\in J}\sum_{k\in K^{NF}} y^s_{ijk} = b_{is} \quad \forall \ i\in I^C, s\in S \tag{3}$$

$$\sum_{i\in I^{AN}} x_{ijk} \le 1 \quad \forall \ j\in J, k\in K \tag{4}$$

$$\sum_{k\in K^F} x_{ijk} + \sum_{k\in K^{NF}} y^s_{ijk} \le A_{ij} \quad \forall \ i\in I, j\in J, s\in S \tag{5}$$

$$1 - \sum_{l\in J^{AN}} x_{ij,k+u} \ge x_{ijk} \quad \forall \ i\in I^{AN}, j\in J, k\in K^F, u\in\{0..U_{ijk}\} \tag{6}$$

$$\sum_{i\in I} P_{ij}\, y^s_{ijf} + \sum_{i\in I^{AN}}\sum_{k\in K^F} V_{ijk}\, x_{ijk} \le C_{jf} \quad \forall \ j\in J, s\in S \tag{7}$$

$$\sum_{i\in I} P_{ij}\, y^s_{ijk} \le C_{jk} \quad \forall \ j\in J, k\in K^{NF}, s\in S \tag{8}$$

Constraints 1 require each accepted job to be scheduled exactly once for each scenario. Constraints 2 require that each new job, $i \in I^N$, is scheduled exactly once if it is accepted by the firm, i.e. $z_i = 1$, and not at all if it is rejected. Constraints 3 require each contingent job to be scheduled exactly once for only the scenarios which it exists in the backlog, i.e. $b_{is} = 1$. Constraints 4 limit the number of jobs $i \in I^{AN}$ that are assigned to any timeslot within the frozen portion of the schedule. Constraints 5 ensure that jobs can only be assigned to resources which are capable of processing

them. Constraints 6 ensure that no two jobs in the frozen period are processed simultaneously. Constraints 7 require the total processing time allocated for the first non-fixed big-bucket period to be less than the available capacity in this period. Constraints 8 require the total processing time allocated for all other big-bucket periods is less than the available capacity.

The formulation for the OA&S problem can also be extended to consider robust scheduling objectives. The formulations presented to this point have aimed to optimize the expected cost/reward, representing a risk-neutral approach. However, there has been growing interest in academia and practice in making robust decisions which are risk-averse which hedge against the likelihood of unfavorable scenarios. The objective for the robust optimization of revenue aims to maximize the revenue associated with the worst-case scenario across the scenario set, S. We have shown in the example from Appendix E, that the optimal scheduling solution can differ when considering a robust optimization objective compared to an expected-case objective.

The objective function of the expected-case formulations can be decomposed into two segments: the scenario-independent and scenario-dependent segments. Note that the scenario-dependent portion of the objective function is a weighted (by the probability of each scenario, q_s) sum of the revenue associated with each scenario. This assumes that a probability of occurrence can be estimated for each scenario, which in some cases may be too difficult to estimate accurately. The robust adaptation replaces this portion of the objective function (and any consideration of the probability of each scenario, q_s), with a new variable Z.

Formulation **(RH-OAS)**

$$\max \quad \left(\sum_{j \in J} \sum_{k \in K^F} \sum_{i \in I^{AN}} R_{ijk}\ x_{ijk} \right) + Z$$

s.t.

Constraints 1-8 from OA

$$Z \ \leq \ \sum_{i \in I} \sum_{j \in J} \sum_{k \in K^{NF}} R_{ijk}\ y^s_{ijk} \quad \forall \quad s \in S \quad (9)$$

The calculation of the scenario-dependent revenue is added as a new constraint, Constraints 9, which restrict Z to be less than the minimum revenue associated with the non-fixed portion any scenario, s. The objective function now represents the sum of the revenue associated with the

frozen portion of the schedule plus the minimum revenue of the non-fixed portion of the schedule, measured across the scenario set S. All other constraints in the problem remain unchanged.

One of the downsides to risk-averse objectives which optimize on the worst-case is that over-conservatism dominates decision making. Other objectives functions such as beta-robustness, Pareto robustly optimal solutions, Conditional-Value-at-Risk, Value-at-Risk address this by removing emphasis from the worst-case solution. We propose the following objective function that allows for the specification of the risk-averseness of the decision maker. In this formulation, the parameter $\epsilon \in [0, 1)$, defines the decision makers attitude towards risk. When ϵ takes a value of 1, this objective becomes equivalent to the risk-neutral stochastic optimization problem, and when $\epsilon = 0$, the objective is equivalent to the worst-case optimization problem.

$$\max \sum_{j \in J} \sum_{k \in K^F} \sum_{i \in I^{AN}} R_{ijk}\ x_{ijk} + \epsilon \sum_{j \in J} \sum_{k \in K^{NF}} \sum_{s \in S} \sum_{i \in I} q_s\ R_{ijk}\ y^s_{ijk} + (1 - \epsilon) Z \tag{eRH-OAS}$$

F.0.2 Evaluation of the Order Acceptance & Scheduling Problem

To strengthen our findings, we propose to provide an analysis which evaluates the Value of the Stochastic Solution (VSS) and the Expected Value of Perfect Information (EVPI) for the Order Acceptance & Scheduling formulations. The VSS defines the expected value which is gained by using the scenario-based Hybrid formulation for the underlying scheduling problem compared to using a deterministic counterpart to the problem. In this experiment, we evaluate the OA&S formulation as a two-stage problem. In the first stage, the decision maker makes job acceptance decisions and creates the schedule for the frozen portion of the schedule. In the second stage, given the decisions made for the frozen period, the remaining jobs are scheduled to the non-fixed portion of the schedule, for each possible scenario. Each scenario in the second-stage is solved for individually, so they can be treated as a deterministic scheduling problem. The second-stage revenues are used to generate a distribution which can be used to evaluate the effectiveness of the scheduling formulation in the first-stage problem.

The stochastic model differs from the deterministic model in how it considers contingent jobs while making the first-stage scheduling decisions. The non-deterministic implementations solve for the first stage schedule while optimizing for the expected, worst-case or epsilon-robust revenue, considering the individual demand scenarios subject to the contingent jobs. However, in the de-

terministic problem, uncertainty is not considered. Given that only accepted jobs, $i \in I^A$ can be scheduled in the frozen portion of the schedule, no contingent jobs are considered in the first stage problem of the deterministic problem. Therefore, the formulation of the first stage problem becomes:

Formulation **(D1)**

$$\max \quad \sum_{j \in J} \left(\sum_{k \in K} \sum_{i \in I^A} R_{ijk}\, x_{ijk} \right)$$

s.t.

$$\sum_{j \in J} \sum_{k \in K} x_{ijk} = 1 \quad \forall \quad i \in I^A \tag{1}$$

$$\sum_{i \in I^A} x_{ijk} \leq 1 \quad \forall \quad j \in J, k \in K \tag{2}$$

$$\sum_{k \in K} x_{ijk} \leq A_{ij} \quad \forall \quad i \in I^A, j \in J \tag{3}$$

$$1 - \sum_{l \in I^A} x_{lj,k+u} \geq x_{ijk} \quad \forall \quad i \in I^A, j \in J, k \in K, u \in \{0..P_{ij}\} \tag{4}$$

Note that in this formulation, a schedule is created for the entire planning horizon, $k \in K$. However, only the schedule for the fixed portion of the horizon, $k \in K^F$, will be considered as the first-stage solution. It is assumed that the decision maker gains knowledge of the realization of each contingent job, $i \in I^C$, as soon as the schedule for the frozen period is decided. Once the contingent jobs are *realized*, the newly accepted subset of jobs can be added to I^A and the declined subset can be discarded. The deterministic problem can be resolved for the jobs which were not scheduled in the first stage using the deterministic counterpart of the big-bucket formulation. The revenue of the solution for that scenario can be found as the sum of the revenues from the first and second stage problems. We will refer to the revenue associated with a single contingent demand realization following a deterministic scheduling of the first stage problem as π_D^s. The second stage problem will be solved for many scenarios to define a distribution of the expected revenue, given the scheduling decisions made during the first-stage problem.

We find the VSS by finding the difference in the expected revenue of the two solution methods:

$$VSS = \pi_S - E[\pi_D] = \pi_S - \frac{1}{|S|} \sum_{s \in S} \pi_D^s$$

where π_S is the objective function optimized by the stochastic formulation. $E[\pi_D]$ is the expected revenue associated with the conditions defined in the deterministic counterpart to the problem. This value is found as the mean of the optimal schedules for the second-stage problem of the deterministic counterpart for each scenario, s, in the scenario set, S, plus the revenue from the first stage.

The Expected Value of Perfect Information (EVPI) of the stochastic formulation can be calculated by finding the difference of the objective function found using the Hybrid formulation and the weighted objective function of the deterministic model solved for each scenario with perfect information. In this case, perfect information constitutes knowledge of which contingent jobs will be accepted, prior to making any of the first stage decision. The value of EVPI is found as:

$$EVPI = \sum_{s \in S} q_s \pi^s_{D*} - \pi_S$$

where π^s_{D*}, also referred to as the *Wait-and-see* value, is the value of the objective function of the deterministic problem for the full schedule, given perfect information.

F.0.3 Extensions to Additional Revenue Management Problems

Future research directions outside the scope of this dissertation include the extension of the proposed models to other various Revenue Management algorithms. This would include the Dynamic Pricing and Scheduling Problem, Due-Date Setting and Scheduling Problem, and the Simultaneous Pricing, Due-Date Setting and Scheduling Problem (SPDSP).

Several challenges must be addressed in the formulation of these problems. First, we must define a method of specifying what quotation to offer to a new incoming customer. Lu and Liu [109] have shown that the offering a quotation from a discrete, finite set of potential offers, i.e. a *quotation menu*, will result in a minimal expected loss of profit, compared to considering a continuous offering. The decision variable in the OA&S model, z_i, which takes a value of 1 if the firm chooses to offer a quotation for order i, can be extended. We propose the extension of the variable to be z^o_i, to take a value of 1 if quotation offer, o, is offered to incoming order, i, and 0 otherwise. The offer o is selected from quotation menu, $o \in O$.

We must also characterize customer reactions to quotations through the use of a customer value function. Easton and Moodie [49], use Berkson's S-shaped binary choice logit model [22], and recommend the estimation of parameters using historical records. Other customer value functions have been used in the literature. We provide a short review and discussion the value functions found in the literature in Appendix G. The estimation of the customer probability acceptance of a quote can be used to create a profit function for a given incoming job, which can then be used in the formulation framework we have presented in this chapter.

APPENDIX G

CUSTOMER VALUE FUNCTIONS IN REVENUE MANAGEMENT

In recent decades, the development of the Internet and information technologies along with intensified global competition and higher customer expectations have put manufacturers and service providers under increasing external pressures. Customers expect lower prices, shorter lead times and reliable delivery dates, while sellers prefer to quote a higher price and longer delivery times [72]. Companies that operate as direct sellers to their customers typically receive enquiries from their customers with limited information on their preferences regarding aspects such as the desired product, order size, price and lead time. In order to gain these orders, the firm must respond by preparing a quotation bid, including the terms they are willing to offer. Customers, likely to have requested quotes from several firms, will commit to a purchase with the firm that offers the most attractive combination of price, lead time and quality [50]. It is important that firms do not promise overly optimistic prices or lead times as this will result in excess demand and lead to production congestion which will increase the risk of facing tardiness penalties. On the other hand, if the firm quotes to cautiously, with longer lead times and higher prices, the customer will select a competitors offer. For that reason, proposal bids should be prepared in such a way as to increase the likelihood of customer acceptance while remaining attainable and profitable [167]. The aim of this exercise is to review the literature pertaining to the methods in which these selling firms generate optimal proposals, specifically when simultaneously quoting price and lead times. Focus will be dedicated to understanding how customer preferences are modeled as well as how the influences of uncertainty in the form of contingent demand plays a role in these decisions.

For years, product and service prices were rarely adjusted, mainly because of the high costs and extensive manual efforts required to do so. However, the rise of information technologies and data sciences have drastically transformed the capabilities of firms to make these adjustments [62]. Sellers are now able to implement automated pricing mechanisms that take into account customer behaviors based on historical transactions. Prior to this, research in the field of joint pricing and

lead time quoting was limited to static cases [26]. In these problems, an optimal price and lead time was solved to satisfy steady-state queuing systems where the objective was to maximize average expected profits, given some processing capability. This optimal price and lead time would be used for every customer, regardless of their preferences and lead to sub-optimal utility for both customers and suppliers [200]. Typically, in these problems customers were assumed to be homogeneous and the demand faced by the firm was modeled either as a linear function price and lead time [141, 70], or as a threshold function where a customer would place an order only if the price was less than some value [3, 170]. Solution methods in these models typically could be described as a closed-form solution [101]. In terms of application, these formulations tend to be limited to production settings which invoke fixed sequencing rules such as EDD or FCFS [129, 188].

In recent years however, research has been dedicated to the formulation of dynamic quoting algorithms where each customer would receive a unique quote. This follows literature and industry trends of implementing revenue management principles in these type of direct seller environments. Broadly speaking, revenue management has been described as the science involving sophisticated information technology systems leveraging historical data and current congestion that enables firms facing the constraints of fixed perishable capacity the ability to identify optimal policies and make decisions with the objective of maximizing profits over some time horizon [38, 42]. In other words, revenue management has been described as "selling the right product to the right customer at the right time at the right place at the right price" [26]. During the quotation process, the objective of the offering firm is to provide the arriving customer a combination of price, lead time, etc. that maximizes its expected return. Generally, the expected return that a firm stands to gain is composed of the sum of the revenue gained from successful bids minus the variable cost of production and any penalties associated with the early or tardy completion of those orders. As the performance of the firm is dependent on its existing commitments, the joint price and lead-time quotation problem of new jobs is typically solved in conjunction with the scheduling or sequencing problem of the existing jobs [50, 134, 106]. This problem, when considering scheduling decisions is also referred to as the simultaneous pricing, due-date setting and scheduling problem (SPDSP). Due to the NP-hardness of the scheduling problem alone, it is common for scheduling decision in this problem to be determined through the selection of some sequencing rule such as EDD or SWPT policies [105, 187].

One of the major modeling dimensions which determines the solutions in each model is the choice of modeling customer decisions. Specifically, in order to develop a reliable decision-making model, there must be a method to describe the behavior of the customer. For example, these customer behaviors could describe the likelihood that a customer will accept a proposal offered by the firm, or could describe the negotiation tactics of the customer. Regarding the likelihood that a customer will accept a proposal, behaviors can be modeled as either deterministic or probabilistic functions. An example of a deterministic customer behavior could be where a customer will accept a bid with a probability of 1 if the price (or lead time) is below some threshold value [170]. If the offered price if above that threshold then the customer would reject the offer. This threshold parameter could be a shared threshold among all customer agents in the model or could be unique for each customer class [87] or unique value for each customer [10]. Probabilistic behaviors are much more common in the literature, as this represents a more realistic representation of the firms understanding of customer behaviors. The customer behavior, in these cases, can be modeled as a utility function where the probability that a customer accepts an offer for the firm is a function of the offered price/lead time. Several types of functions are used to describe different customer preferences and their sensitivity to discrepancies between those preferences and the firms' offer. The most commonly used of these is to model demand as a linear combination of price and lead time [3, 88, 105], such as shown below:

$P(p, l) = 1 - \alpha * p - \beta * l$ Where p,l represent the offered price and lead time, α, β represent sensitivity parameters for price and lead-time, respectively, of the customer, and P(p,l) represents the probability the customer will accept the offer. In the equation above, the probability of customer acceptance is linearly decreasing in both price and lead time. Using this formulation, it is possible to describe a customer as price-sensitive if: $\alpha > \beta$ And time-sensitive otherwise. The values of these parameters can be established through historical data, or with knowledge regarding the customers preferences. Further, it is possible to establish different customer classes in the system through modification of these sensitivity parameters.

Aside from threshold and linear descriptions of customer behaviors, more complex, non-linear functions have been implemented to reflect more realistic customer behaviors. For example, the use of convex [56], concave [68, 195, 8], or convex-concave [50, 7, 125, 35] (also referred to as an S-logit curve) probability functions can be used to describe the sensitivities of customers to price, lead-time

or both. To explain the impact of these function curves, consider a time-sensitive customer, with no preference on price. This assumption reduces the probability of acceptance calculation to be a function of a single input, i.e. P(l). In this case, the following function shapes of this customer's behavior would have the following implications:

- Convex curve – the customer has a preferred delivery date at which if offered they will accept with probability 1. As the quoted LT increases, the customer at first is relatively insensitive, to incremental increase, aka will not affect their probability of acceptance much, but becomes more sensitive as the LT continues to rise

- Concave curve – the customer has a preferred delivery date at which if offered will accept with probability 1. As the quoted LT increases above that preferred delivery date, the customer is highly sensitive to incremental increases in, but becomes less and less sensitive as the LT continues to rise

- Convex-concave curve – the customer has a preferred LT at which if offered will accept with probability 1. As the quoted LT increases above that preferred LT, the customer at first is relatively insensitive to incremental increases in LT, but becomes more sensitive as the LT continues to rise. As the LT continues to increase, the customer becomes less sensitive to those incremental increases

This example was simplified to only consider the probability of acceptance as a function of quoted lead time, but this is also applicable for lead time sensitive and price-and-lead time sensitive customers. The example below shows what a function for the convex-concave probability of acceptance curve of a price-and-time sensitive customer could look like, where each β parameter describes the customers sensitivity to varying inputs [50]: $P(p, l) = [1 + \beta_0 \exp(\beta_1 * p - \beta_2 * l)]^{-1}$

Similar to how the threshold and linear approximations of customer behaviors depend on parameterization, the curved representations of customer behaviors can be leveraged to describe unique customer classes, as shown in [57]. Under these assumptions, customers which are considered risk-averse could be modeled using a concave utility curve, risk-insensitive as linear utility curve and risk-seeking as convex utility curve. Other non-linear customer utility functions include log-linear [36], and exponential functions [18]. Further, discrete probabilities for any quote can be established

using any customer utility function [106]. There are many examples of explicitly defining differences between customer classes in literature. Some examples include: patient vs impatient customers [7], price-sensitive vs lead time sensitive [53, 125, 187], guaranteed delivery-date vs best effort delivery date [123] as well as distinctions between customer channels such as online customers vs in-store customers [80], etc. Within each of these groupings individual customers can be modeled to have unique or identical utility functions.

Aside from describing customer behaviors, the matter of how the quoting process is modeled is critical. Some distinctions include offering the customer a single take-it-or-leave-it quote [52] compared to offering a quote menu to the customer consisting of multiple price-lead time pairs which the customer can choose from [7, 38]. Offering a quote menu has the advantage of pandering to multiple customer classes when the customers utility function is unknown [123]. Several streams of research also explore the benefits of delaying the quotation response compared to responding immediately [87]. The reasoning being that during the delay between customer enquiry and quotation delivery, better potential customers may enter the queue. Liu et. al. [105] compare the performance of a firm who must quote several arriving customers. The firm can either quote all of the customer enquiries simultaneously or sequentially. In the simultaneous case, the firm would quote all new jobs at the same time. Comparatively, in the sequential case, the firm would quote each job in the pool awaiting quotes one at a time, where the result of each quote would be known before quoting the next. In this problem, the objective was to maximize the expected contribution of all the new jobs. The case of sequential quoting led to less uncertainty and therefore an optimal performance [105].

In practice, it is common for customers and suppliers to negotiate during the demand generation process. Rather than just accepting or rejecting a firm quoted offer, the customer can provide a counter-offer at which point the firm can reassess and requote the customer. This negotiation process has been modeled and solved in several different methods including using game theory models such as Nash equilibrium or Stackleberg games [195], through simulation-based optimization [130, 18] or closed-form optimization [170]. The optimization problems tend to focus on identifying an optimal quote as the point at which the distance between the customer's utility function and the firm's utility function are at a minimum. Pan et. al. [130] explore a case of an intra-supply chain interaction between an upstream supplier and downstream customer. In this scenario, it

is of the best interest of both parties to identify an agreement that is mutually optimal. In a typical customer-supplier relationship, successful negotiation is also beneficial for both parties, and from the perspective of the firm could result in additional future demand in the form of a repeat customer.

It is important to consider however, that in practice, this negotiation process can take weeks, and often times can result in an unsuccessful bid. During this period of uncertainty, the potential project is considered as a contingent demand [50], and is a source of uncertainty that must be accounted for when quoting arriving customers. The literature on the joint price and lead time quotation problem while considering contingent demand is relatively adolescent, as few papers explore this problem setting. Easton and Moodie [50] are recognized as the first attempt to consider contingent demand in the pricing and lead time quotation of new customers in a direct seller environment. However, their model as well as many that follow it [187, 104] do not attempt to solve the problem in a practical generalized form. Instead they either implement scheduling heuristics such as EDD or FCFS to reduce the complexity of the problem, model simple single-machine processing environment or simply acknowledge that the problem is too complex to solve in medium-large settings [167].

G.1 Discussion

In order to develop a practical SPDSP model which considers a more generalized production setting while providing quality solutions in reasonable time, several challenges must be addressed. Firstly, in the joint price and lead-time quotation problem which considers contingent demand, the firm faces an exponentially growing number of possible scheduling scenarios, with respect to the number of existing contingent jobs at the time of quoting. While it is feasible to consider all of these scenarios when the number of contingent jobs is low, the problem quickly becomes too complex to solve in reasonable time. Essentially in order to optimally develop a new joint quotation for an arriving customer, all of these scheduling scenarios would need to be considered. However, as it is infeasible to solve this problem in reasonable time, methods or heuristics need to be implemented to reduce the number of scenarios that will be used to make quotation decisions, while remaining robust. Easton and Moodie [50] have shown that if the probability of each of the customer accepting their offered quotation can be estimated, then the likelihood that any one of these scheduling scenarios to be realized can be calculated. Further, our research indicates

that a majority of these possible scenarios are highly unlikely, assuming that at least some of the contingent jobs' probability of acceptance skew towards either being accepted or rejected.

Another method to reduce the complexity of the joint quoting problem would be to simplify the underlying scheduling problem which dictates the outcome of each scenario. This method had been implemented by Easton and Moodie [50] by assuming a FCFS sequencing policy as well as others who solve similar problems [187]. The current scheduling model used in my research has been a discrete-time mixed-integer linear programming assignment problem, where many attributes of the problem are parameterized before-hand. In this scheduling model, each job in each realization is assigned to be completed by one of several unrelated parallel machines. However, the granularity of the scheduling model may not be necessary considering the granularity of joint quotations in practice are made on a weekly scope. For example, typical lead time quotations would be made as an estimation of weeks, rather than an exact time. For this reason, it may be possible to evaluate each tested scenario through a different problem. Albana et. al. [3] evaluated current backlogs through a bin-packing problem where the size of the bins was dependent on the capacity available and the jobs were assigned within weekly buckets. In this sense, a new job could be quoted a price and lead time by finding which of these weekly buckets to add the new job to. It is unclear whether this application would greatly reduce the processing time required when considering contingent backlogs, but by reducing the granularity of decision-making it may result in a scheduling policy that is less sensitive to changes in the backlog across realizations.

Aside from the challenge of complexity reduction, the other main challenge in developing the SPDSP model will be in the form of clearly defining the problem at hand. This includes explicitly defining customer classes as well as the behaviors of each of the customer, in the form of utility functions. To be able to optimize quoting decisions, these customers behaviors and preferences must be determined. This will include decisions such as what utility function structure to implement, whether it be a linear, S-logit, discretized, etc. function. Continuing with the concept of reducing the granularity of decision-making, it may be beneficial to implement a discretized probability of acceptance function where each pair of price and lead-time would have an expected probability of acceptance for each customer. This would allow for a reduction in the decision space, as the decision would be among a single quote within a quote menu rather than optimizing across a continuous space. While the solution will likely not be globally optimal, the solutions will be at the scope of

practice. For example, when the firm quotes lead times on the scale of weeks, there is no practical difference solving for an optimal customer reaction to a lead time in the scale of hours, say the difference between quoting a lead time 6.2 weeks and 6.3 weeks. By reducing this decision space, the complexity of determining the expected associated marginal costs of this potential job is reduced, as less forward-looking scenarios need to be solved for.

In terms of defining customer classes, some possible options include time- vs price- vs price-and-time-sensitive customers, as well as risk-averse vs -neutral vs -tolerant behaviors within those aforementioned classes. Another aspect of customer definition which needs to be determined is whether the firm will have perfect information on customer preferences, say their desired lead-time and price or how sensitive they are to discrepancies from these preferences. Aside from determining customer classes and customer behaviors, a quoting process must also be explicitly defined. Aspects such as determining whether customers should be offered a single quote or a quote menu and whether the firm should offer a quote to the customer immediately or to delay their response will depend on the definition of the customers in the model. For example, if perfect information on customer preferences is known, it would be beneficial to offer a single quote, however if it is unclear, offering a quote menu may be beneficial. This will be an area of focus in this research. Other considerations such as customer response times, i.e. the period of time which the job remains contingent, and whether negotiations should be included will also need to be decided. Sujan [167] defines a methodology for negotiating quotes with customers subject to contingent demand. A major contribution of this work was Sujan's S-logit model of customer's behavior during negotiations and the probability that a customer would accept a firm's follow up quotation, given the customers counter-offer. As a continuation, terms such as how many and how long each negotiation period lasts are needed in order to be implemented.

G.2 Conclusion

In recent years, the problem of joint price and lead time quotation has been an increasing focus following industry trends of the rise of make-to-order manufacturers, service providers and direct selling firms. Prior to this, much of the literature in this stream had been focused considering a homogeneous customer group and a seller determining an optimal uniform price and lead-time to offer all of their customers. The demand the firm faces would be found as a function of that price

and lead time. However, the typical problem setting now has adapted to determine a dynamic optimal quote for each arriving customer considering the firms current backlog. Critical aspects in developing these models include defining customer utility functions, the firm's objective and associated costs and constraints they are liable to. Customer behaviors are most commonly portrayed by either a linear or convex-concave function which describes the probability that customer will accept a firm offer, given that customers preferences and an input price and lead-time. Others have expanded this utility function to include other considerations such as competitive firms [59], product quality [187], rebate values for late deliveries [1], etc. The firm decides what price and lead-time to offer the customer such that either: its expected revenue across its entire backlog is maximized or the expected marginal contribution of the current arriving customer/s are maximized. This revenue is measures as a sum of the incoming revenue minus operating costs such as for capacity installation [141, 8], expediting/outsourcing [38], or holding and tardiness penalties [100]. Due to the integrated nature of demand generation and production, these joint quotation problems typically consider other decision-making problems such as scheduling [167], procurement [35] and order acceptance/rejection decisions [18]. Other decision-making models have also incorporated negotiation sub-models to depict realistic interactions between suppliers and customers [130, 195], while others consider the trade-offs between offering a single price-lead time quote or multiple pairs as a quote menu [123], as well as exploring whether it is optimal to respond to customer enquiries immediately or to delay the response to wait for other potential customers [87]. Major limitations in this stream of research include the inability to optimally solve for medium-large sized problem sets due to the complexity of the problem as well as implementing an abundance of assumptions which restrict the practical capabilities of their models. Future work in this field include expanding the use-cases of these models to more practical settings.

BIBLIOGRAPHY

[1] Afèche, Philipp, Baron, Opher, and Kerner, Yoav. Pricing time-sensitive services based on realized performance. *Manufacturing & Service Operations Management 15*, 3 (2013), 492–506.

[2] Afzalirad, Mojtaba, and Rezaeian, Javad. A realistic variant of bi-objective unrelated parallel machine scheduling problem: Nsga-ii and moaco approaches. *Applied Soft Computing 50* (2017), 109–123.

[3] Albana, Abduh Sayid, Frein, Yannick, and Hammami, Ramzi. Effect of a lead time-dependent cost on lead time quotation, pricing, and capacity decisions in a stochastic make-to-order system with endogenous demand. *International Journal of Production Economics 203* (2018), 83–95.

[4] Almeder, Christian. A hybrid optimization approach for multi-level capacitated lot-sizing problems. *European Journal of Operational Research 200*, 2 (2010), 599–606.

[5] Almeder, Christian, Klabjan, Diego, Traxler, Renate, and Almada-Lobo, Bernardo. Lead time considerations for the multi-level capacitated lot-sizing problem. *European Journal of Operational Research 241*, 3 (2015), 727–738.

[6] Artigues, Christian, Billaut, Jean-Charles, and Esswein, Carl. Maximization of solution flexibility for robust shop scheduling. *European Journal of Operational Research 165*, 2 (2005), 314–328.

[7] Ata, B, and Olsen, Tava Lennon. Congestion-based leadtime quotation and pricing for revenue maximization with heterogeneous customers. Tech. rep., Citeseer, 2008.

[8] Ata, Barış, and Olsen, Tava Lennon. Near-optimal dynamic lead-time quotation and scheduling under convex-concave customer delay costs. *Operations Research 57*, 3 (2009), 753–768.

[9] Atakan, Semih, Bülbül, Kerem, and Noyan, Nilay. Minimizing value-at-risk in single-machine scheduling. *Annals of Operations Research 248*, 1-2 (2017), 25–73.

[10] Aviv, Yossi, and Pazgal, Amit. A partially observed markov decision process for dynamic pricing. *Management science 51*, 9 (2005), 1400–1416.

[11] Badri, Huda Muhamad, Khamis, Nor Kamaliana, and Ghazali, Mariyam Jameelah. Integration of lot sizing and scheduling models to minimize production cost and time in the automotive industry. *International Journal of Industrial Optimization 1*, 1 (2020), 1–14.

[12] Balas, Egon. On the facial structure of scheduling polyhedra. In *Mathematical Programming Essays in Honor of George B. Dantzig Part I*. Springer, 1985, pp. 179–218.

[13] Bao, Jinsong, Guo, Dongsheng, Li, Jie, and Zhang, Jie. The modelling and operations for the digital twin in the context of manufacturing. *Enterprise Information Systems 13*, 4 (2019), 534–556.

[14] Baruffaldi, Giulia, Accorsi, Riccardo, and Manzini, Riccardo. Warehouse management system customization and information availability in 3pl companies: A decision-support tool. *Industrial Management & Data Systems* (2019).

[15] Barykin, Sergey Yevgenievich, Bochkarev, Andrey Aleksandrovich, Kalinina, Olga Vladimirovna, and Yadykin, Vladimir Konstantinovich. Concept for a supply chain digital twin. *International Journal of Mathematical, Engineering and Management Sciences 5*, 6 (2020), 1498–1515.

[16] Bassett, Matthew H, Pekny, Joseph F, and Reklaitis, Gintaras V. Using detailed scheduling to obtain realistic operating policies for a batch processing facility. *Industrial & engineering chemistry research 36*, 5 (1997), 1717–1726.

[17] Baykasoğlu, Adil, Subulan, Kemal, Güçdemir, Hülya, Dudaklı, Nurhan, and Eren Akyol, Derya. Revenue management for make-to-order manufacturing systems with a real-life application. *The Engineering Economist* (2019), 1–39.

[18] Baykasoğlu, Adil, Subulan, Kemal, Güçdemir, Hülya, Dudaklı, Nurhan, and Eren Akyol, Derya. Revenue management for make-to-order manufacturing systems with a real-life application. *The Engineering Economist 65*, 1 (2020), 27–65.

[19] Bean, James C, Birge, John R, Mittenthal, John, and Noon, Charles E. Matchup scheduling with multiple resources, release dates and disruptions. *Operations research 39*, 3 (1991), 470–483.

[20] Beasley, John E. Or-library: distributing test problems by electronic mail. *Journal of the operational research society 41*, 11 (1990), 1069–1072.

[21] Beraudy, Sébastien, Absi, Nabil, and Dauzère-Pérès, Stéphane. Timed route approaches for large multi-product multi-step capacitated production planning problems. *European Journal of Operational Research 300*, 2 (2022), 602–614.

[22] Berkson, Joseph. A statistically precise and relatively simple method of estimating the bioassay with quantal response, based on the logistic function. *Journal of the American Statistical Association 48*, 263 (1953), 565–599.

[23] Berretta, Regina, França, Paulo M, and Armentano, Vinícius A. Metaheuristic approaches for the multilevel resource-constrained lot-sizing problem with setup and lead times. *Asia-Pacific Journal of Operational Research 22*, 02 (2005), 261–286.

[24] Bertrand, JWM, Wortmann, Johan C, and Wijngaard, Juan. *Production control: a structural and design oriented approach.* Elsevier Science Inc., 1990.

[25] Billington, Peter J, McClain, John O, and Thomas, L Joseph. Mathematical programming approaches to capacity-constrained mrp systems: review, formulation and problem reduction. *Management Science 29*, 10 (1983), 1126–1141.

[26] Bitran, Gabriel, and Caldentey, René. An overview of pricing models for revenue management. *Manufacturing & Service Operations Management 5*, 3 (2003), 203–229.

[27] Boctor*, Fayez F, and Poulin, Patrice. Heuristics for the n-product, m-stage, economic lot sizing and scheduling problem with dynamic demand. *International journal of production research 43*, 13 (2005), 2809–2828.

[28] Bowman, Edward H. The schedule-sequencing problem. *Operations research 7*, 5 (1959), 621–624.

[29] Brandimarte, Paolo. Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations research 41*, 3 (1993), 157–183.

[30] Briand, Cyril, La, H Trung, and Erschler, Jacques. A robust approach for the single machine scheduling problem. *Journal of Scheduling 10*, 3 (2007), 209–221.

[31] Buer, Tobias, Homberger, Jörg, and Gehring, Hermann. A collaborative ant colony metaheuristic for distributed multi-level uncapacitated lot-sizing. *International Journal of Production Research 51*, 17 (2013), 5253–5270.

[32] Buschkühl, Lisbeth, Sahling, Florian, Helber, Stefan, and Tempelmeier, Horst. Dynamic capacitated lot-sizing problems: a classification and review of solution approaches. *Or Spectrum 32*, 2 (2010), 231–261.

[33] Busse, Anselm, Gerlach, Benno, Lengeling, Joel Cedric, Poschmann, Peter, Werner, Johannes, and Zarnitz, Simon. Towards digital twins of multimodal supply chains. *Logistics 5*, 2 (2021), 25.

[34] Cakravastia, Andi, and Nakamura, Nobuto. Model for negotiating the price and due date for a single order with multiple suppliers in a make-to-order environment. *International journal of production research 40*, 14 (2002), 3425–3440.

[35] Cakravastia, Andi, and Takahashi, Katsuhiko. Integrated bidding and manufacturing planning decisions with contingent orders in a make-to-order environment. *Journal of Japan Industrial Management Association 54*, 5 (2003), 291–301.

[36] Carvalho, Alexandre X, and Puterman, ML. Dynamic pricing and reinforcement learning. In *Proceedings of the International Joint Conference on Neural Networks, 2003.* (2003), vol. 4, IEEE, pp. 2916–2921.

[37] Caserta, Marco, Ramirez, Adriana, and Voß, Stefan. A math-heuristic for the multi-level capacitated lot sizing problem with carryover. In *European Conference on the Applications of Evolutionary Computation* (2010), Springer, pp. 462–471.

[38] Çelik, Sabri, and Maglaras, Costis. Dynamic pricing and lead-time quotation for a multiclass make-to-order queue. *Management Science 54*, 6 (2008), 1132–1146.

[39] Chaari, Tarek, Chaabane, Sondes, Aissani, Nassima, and Trentesaux, Damien. Scheduling under uncertainty: Survey and research directions. In *2014 International conference on advanced logistics and transport (ICALT)* (2014), IEEE, pp. 229–234.

[40] Chen, Haoxun. Fix-and-optimize and variable neighborhood search approaches for multi-level capacitated lot sizing problems. *Omega 56* (2015), 25–36.

[41] Chen, Haoxun, and Chu, Chengbin. Supply chain planning with order/setup costs and capacity constraints a new lagrangian relaxation approach. In *2003 IEEE international conference on robotics and automation (Cat. no. 03CH37422)* (2003), vol. 2, IEEE, pp. 1743–1748.

[42] Cheraghi, S Hossein, Dadashzadeh, Mohammad, and Venkitachalam, Prakash. Revenue management in manufacturing: a research landscape. *Journal of Business & Economics Research (JBER) 8*, 2 (2010).

[43] Cullinan, Charles, Sutton, Steve G, and Arnold, Vicky. Technology monoculture: Erp systems,"techno-process diversity" and the threat to the information technology ecosystem. In *Advances in Accounting Behavioral Research*. Emerald Group Publishing Limited, 2010.

[44] Daniels, Richard L, and Carrillo, Janice E. β-robust scheduling for single-machine systems with uncertain processing times. *IIE transactions 29*, 11 (1997), 977–985.

[45] Daniels, Richard L, and Kouvelis, Panagiotis. Robust scheduling to hedge against processing time uncertainty in single-stage production. *Management Science 41*, 2 (1995), 363–376.

[46] DeMatteis, John J. An economic lot-sizing technique, i: The part-period algorithm. *IBM systems Journal 7*, 1 (1968), 30–38.

[47] Ducrée, Jens, Gravitt, Max, Walshe, Ray, Bartling, Sönke, Etzrodt, Martin, and Harrington, Tomás. Open platform concept for blockchain-enabled crowdsourcing of technology development and supply chains. *Frontiers in Blockchain* (2020), 47.

[48] Dyer, Martin E, and Wolsey, Laurence A. Formulating the single machine sequencing problem with release dates as a mixed integer program. *Discrete Applied Mathematics 26*, 2-3 (1990), 255–270.

[49] Eastman, Willard Lawrence. *Linear programming with pattern constraints: a thesis.* PhD thesis, Harvard University, 1958.

[50] Easton, Fred F, and Moodie, Douglas R. Pricing and lead time decisions for make-to-order firms with contingent orders. *European Journal of operational research 116*, 2 (1999), 305–318.

[51] Ebadian, M, Rabbani, M, Jolai, F, Torabi, SA, and Tavakkoli-Moghaddam, R. A new decision-making structure for the order entry stage in make-to-order environments. *International Journal of Production Economics 111*, 2 (2008), 351–367.

[52] ElHafsi, Mohsen. An operational decision model for lead-time and price quotation in congested manufacturing systems. *European Journal of Operational Research 126*, 2 (2000), 355–370.

[53] Elhafsi, Mohsen, and Rolland, Erik. Negotiating price/delivery date in a stochastic manufacturing environment. *IIE transactions 31*, 3 (1999), 255–270.

[54] Elmaghraby, Salah E. The one-machine sequencing problem with delay costs. *Journal of Industrial Engineering 19* (1968), 105–108.

[55] Erschler, JGCF, Fontan, Gérard, Mercé, Colette, and Roubellat, François. A new dominance concept in scheduling n jobs on a single machine with ready times and due dates. *Operations Research 31*, 1 (1983), 114–127.

[56] Feng, Jiejian, Liu, Liming, and Liu, Xiaoming. An optimal policy for joint dynamic price and lead-time quotation. *Operations research 59*, 6 (2011), 1523–1527.

[57] Feng, Jiejian, and Zhang, Michael. Dynamic quotation of leadtime and price for a make-to-order system with multiple customer classes and perfect information on customer preferences. *European Journal of Operational Research 258*, 1 (2017), 334–342.

[58] Gao, Hong, Fox, Mark S, Chiang, Whay-Yu, and Hikita, Shiro. Building robust schedules—an empirical study of single machine scheduling with uncertainty. *Soumis à Management Science* (1995).

[59] Garmdare, Hamid Sattari, Lotfi, MM, and Honarvar, Mahboobeh. Integrated model for pricing, delivery time setting, and scheduling in make-to-order environments. *Journal of Industrial Engineering International 14*, 1 (2018), 55–64.

[60] Gerlach, B, Zarnitz, S, and Straube, F. Digital twins in logistics and supply chain management–conceptual clarification, use cases and potentials of "digital supply chain twins". *Unpublished work*.

[61] Glover, Fred. Future paths for integer programming and links to ar tifi cial intelli g en ce. *Computers operations research 13*, 5 (1986), 533–549.

[62] Gönsch, Jochen, Klein, Robert, Neugebauer, Michael, and Steinhardt, Claudius. Dynamic pricing with strategic customers. *Journal of Business Economics 83*, 5 (2013), 505–549.

[63] Gopalakrishnan, Mohan, Ding, Ke, Bourjolly, Jean-Marie, and Mohan, Srimathy. A tabu-search heuristic for the capacitated lot-sizing problem with set-up carryover. *Management science 47*, 6 (2001), 851–863.

[64] Goren, Selcuk, and Sabuncuoglu, Ihsan. Optimization of schedule robustness and stability under random machine breakdowns and processing time variability. *IIE Transactions 42*, 3 (2009), 203–220.

[65] Grieves, Michael. Digital twin: manufacturing excellence through virtual factory replication. *White paper 1* (2014), 1–7.

[66] Groff, Gene K. A lot sizing rule for time-phased component demand. *Production and Inventory Management 20*, 1 (1979), 47–53.

[67] Guo, Jiapeng, Zhao, Ning, Sun, Lin, and Zhang, Saipeng. Modular based flexible digital twin for factory design. *Journal of Ambient Intelligence and Humanized Computing 10*, 3 (2019), 1189–1200.

[68] Hafızoğlu, A Baykal, Gel, Esma S, and Keskinocak, Pınar. Price and lead time quotation for contract and spot customers. *Operations Research 64*, 2 (2016), 406–415.

[69] Hajipour, Vahid, Fattahi, Parviz, and Nobari, Arash. A hybrid ant colony optimization algorithm to optimize capacitated lot-sizing problem. *Journal of Industrial and Systems Engineering 7*, 1 (2014), 1–20.

[70] Hall, Joseph M, Kopalle, Praveen K, and Pyke, David F. Static and dynamic pricing of excess capacity in a make-to-order environment. *Production and Operations Management 18*, 4 (2009), 411–425.

[71] Hall, Nicholas G, and Posner, Marc E. Generating experimental data for computational testing with machine scheduling applications. *Operations Research 49*, 6 (2001), 854–865.

[72] Hao, Juan, Yu, Jian Jun, and Wu, Mian Can. Dynamic joint decision on price and delivery date in mto manufacturer based on agent. In *Advanced Materials Research* (2014), vol. 860, Trans Tech Publ, pp. 2812–2816.

[73] Harris, Ford W. How many parts to make at once. *Operations research 38*, 6 (1990), 947–950.

[74] Helber, Stefan, and Sahling, Florian. A fix-and-optimize approach for the multi-level capacitated lot sizing problem. *International Journal of Production Economics 123*, 2 (2010), 247–256.

[75] Held, Michael, and Karp, Richard M. The traveling-salesman problem and minimum spanning trees: Part ii. *Mathematical programming 1*, 1 (1971), 6–25.

[76] Hindi, Khalil S. Solving the clsp by a tabu search heuristic. *Journal of the Operational Research Society 47*, 1 (1996), 151–161.

[77] Hindi, Khalil S, Fleszar, Krzysztof, and Charalambous, Christoforos. An effective heuristic for the clsp with set-up times. *Journal of the Operational Research Society 54*, 5 (2003), 490–498.

[78] Homberger, Jörg, and Gehring, Hermann. An ant colony optimization approach for the multi-level unconstrained lot-sizing problem. In *2009 42nd Hawaii International Conference on System Sciences* (2009), IEEE, pp. 1–7.

[79] Hopp, Wallace J, and Spearman, Mark L. *Factory physics*. Waveland Press, 2011.

[80] Hua, Guowei, Wang, Shouyang, and Cheng, TC Edwin. Price and lead time decisions in dual-channel supply chains. *European journal of operational research 205*, 1 (2010), 113–126.

[81] Ignall, Edward, and Schrage, Linus. Application of the branch and bound technique to some flow-shop scheduling problems. *Operations research 13*, 3 (1965), 400–412.

[82] Karimi, Behrooz, Ghomi, SMT Fatemi, and Wilson, JM. The capacitated lot sizing problem: a review of models and algorithms. *Omega 31*, 5 (2003), 365–378.

[83] Kasperski, Adam, Kurpisz, Adam, and Zieliński, Paweł. Approximating a two-machine flow shop scheduling under discrete scenario uncertainty. *European Journal of Operational Research 217*, 1 (2012), 36–43.

[84] Kasperski, Adam, Kurpisz, Adam, and Zieliński, Paweł. Parallel machine scheduling under uncertainty. In *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems* (2012), Springer, pp. 74–83.

[85] Kasperski, Adam, and Zieliński, Paweł. Risk-averse single machine scheduling: complexity and approximation. *Journal of Scheduling 22*, 5 (2019), 567–580.

[86] Keha, Ahmet B, Khowala, Ketan, and Fowler, John W. Mixed integer programming formulations for single machine scheduling problems. *Computers & Industrial Engineering 56*, 1 (2009), 357–367.

[87] Keskinocak, Pinar, Ravi, R, and Tayur, Sridhar. Scheduling and reliable lead-time quotation for orders with availability intervals and lead-time sensitive revenues. *Management science 47*, 2 (2001), 264–279.

[88] Khouja, Moutaz. A joint optimal pricing, rebate value, and lot sizing model. *European Journal of Operational Research 174*, 2 (2006), 706–723.

[89] Kim, Jae-Gon, Bang, June-Young, Jun, Hong-Bae, and Shin, Jong-Ho. Dominance conditions for optimal order-lot matching in the make-to-order production system. *Processes 8*, 2 (2020), 255.

[90] Kim, Jae-Gon, and Lim, SK. Order-lot pegging for minimizing total tardiness in semiconductor wafer fabrication process. *Journal of the Operational Research Society 63*, 9 (2012), 1258–1270.

[91] Kirkpatrick, Scott, Gelatt, C Daniel, and Vecchi, Mario P. Optimization by simulated annealing. *science 220*, 4598 (1983), 671–680.

[92] Kis, Tamás, and Pesch, Erwin. A review of exact solution methods for the non-preemptive multiprocessor flowshop problem. *European Journal of Operational Research 164*, 3 (2005), 592–608.

[93] Koker, Ekin. Three essays on data-driven optimization for scheduling in manufacturing and healthcare. *University of Massachusetts Amherst* (2019).

[94] Kunath, Martin, and Winkler, Herwig. Integrating the digital twin of the manufacturing system into a decision support system for improving the order management process. *Procedia Cirp 72* (2018), 225–231.

[95] Lambrechts, Olivier, Demeulemeester, Erik, and Herroelen, Willy. Time slack-based techniques for robust project scheduling subject to resource uncertainty. *Annals of Operations Research 186*, 1 (2011), 443–464.

[96] Land, Ailsa H, and Doig, Alison G. An automatic method for solving discrete programming problems. In *50 Years of Integer Programming 1958-2008*. Springer, 2010, pp. 105–132.

[97] Leus, Roel, and Herroelen, Willy. The complexity of machine scheduling for stability with a single disrupted job. *Operations Research Letters 33*, 2 (2005), 151–156.

[98] Li, Liuxi, Song, Shiji, Wu, Cheng, and Wang, Rui. Fix-and-optimize and variable neighborhood search approaches for stochastic multi-item capacitated lot-sizing problems. *Mathematical Problems in Engineering 2017* (2017).

[99] Li, Luning, Aslam, Sohaib, Wileman, Andrew, and Perinpanayagam, Suresh. Digital twin in aerospace industry: A gentle introduction. *IEEE Access* (2021).

[100] Li, Xueping, Wang, Jiao, and Sawhney, Rapinder. Reinforcement learning for joint pricing, lead-time and scheduling decisions in make-to-order systems. *European Journal of Operational Research 221*, 1 (2012), 99–109.

[101] Li, Yina, Lin, Qiang, and Ye, Fei. Pricing and promised delivery lead time decisions with a risk-averse agent. *International Journal of Production Research 52*, 12 (2014), 3518–3537.

[102] Li, Zukui, and Ierapetritou, Marianthi. Process scheduling under uncertainty: Review and challenges. *Computers & Chemical Engineering 32*, 4-5 (2008), 715–727.

[103] Lin, Xiaoxia, Janak, Stacy L, and Floudas, Christodoulos A. A new robust optimization approach for scheduling under uncertainty:: I. bounded uncertainty. *Computers & chemical engineering 28*, 6-7 (2004), 1069–1085.

[104] Liu, Shu-Chu, and Liu, Chun-Hui. Simultaneous pricing, due-date setting and scheduling for make-to-order firms with contingent orders. *Journal of the Chinese Institute of Industrial Engineers 26*, 1 (2009), 32–43.

[105] Liu, Zhixin, Lu, Liang, and Qi, Xiangtong. Simultaneous and sequential price quotations for uncertain order inquiries with production scheduling cost. *IIE Transactions 44*, 10 (2012), 820–833.

[106] Liu, Zhixin, Lu, Liang, and Qi, Xiangtong. Price quotation for orders with different due dates. *International Journal of Production Economics 220* (2020), 107448.

[107] Lomnicki, ZA. A "branch-and-bound" algorithm for the exact solution of the three-machine scheduling problem. *Journal of the operational research society 16*, 1 (1965), 89–100.

[108] Lu, Chung-Cheng, Lin, Shih-Wei, and Ying, Kuo-Ching. Minimizing worst-case regret of makespan on a single machine with uncertain processing and setup times. *Applied Soft Computing 23* (2014), 144–151.

[109] Lu, Liang, Liu, Zhixin, and Qi, Xiangtong. Coordinated price quotation and production scheduling for uncertain order inquiries. *IIE transactions 45*, 12 (2013), 1293–1308.

[110] Lu, Mengshi, and Shen, Zuo-Jun Max. A review of robust operations management under model uncertainty. *Production and Operations Management* (2020).

[111] Lupeikiene, Audrone, Dzemyda, Gintautas, Kiss, Ferenc, and Caplinskas, Albertas. Advanced planning and scheduling systems: modeling and implementation challenges. *Informatica 25*, 4 (2014), 581–616.

[112] Manne, Alan S. Programming of economic lot sizes. *Management science 4*, 2 (1958), 115–135.

[113] Mehta, Sanjay V. Predictable scheduling of a single machine subject to breakdowns. *International Journal of Computer Integrated Manufacturing 12*, 1 (1999), 15–38.

[114] Mestry, Siddharth, Damodaran, Purushothaman, and Chen, Chin-Sheng. A branch and price solution approach for order acceptance and capacity planning in make-to-order operations. *European Journal of Operational Research 211*, 3 (2011), 480–495.

[115] Modarres, Mohammad, and Nazemi, Jamshid. Yield management in manufacturing: A conceptual model & research propositions. *Available at SSRN 2305232* (2005).

[116] Mohammadi, Mohammad, Ghomi, SMT Fatemi, Karimi, Behrooz, and Torabi, S Ali. Rolling-horizon and fix-and-relax heuristics for the multi-product multi-level capacitated lotsizing problem with sequence-dependent setups. *Journal of Intelligent Manufacturing 21*, 4 (2010), 501–510.

[117] Mönch, Lars, Shen, Liji, and Fowler, John W. Heuristics for order-lot pegging in multi-fab settings. In *2020 Winter Simulation Conference (WSC)* (2020), IEEE, pp. 1742–1752.

[118] Moodie, Douglas R. Demand management: The evaluation of price and due date negotiation strategies using simulation. *Production and Operations Management 8*, 2 (1999), 151–162.

[119] Moodie, DR. Due date demand management: negotiating the trade-off between price and delivery. *International Journal of Production Research 37*, 5 (1999), 997–1021.

[120] Mourtzis, Dimitris. Challenges and future perspectives for the life cycle of manufacturing networks in the mass customisation era. *Logistics Research 9*, 1 (2016), 2.

[121] Muhlemann, AP, Lockett, AG, and Farn, C-K. Job shop scheduling heuristics and frequency of scheduling. *THE INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH 20*, 2 (1982), 227–241.

[122] Naderi, B, Gohari, Sheida, and Yazdani, M. Hybrid flexible flowshop problems: Models and solution methods. *Applied mathematical modelling 38*, 24 (2014), 5767–5780.

[123] Nault, Barrie R, Yang, Wensheng, Tu, Yiliu, et al. Dynamic price quotation in a responsive supply chain for one-of-a-kind production. *International Journal of Production Economics 139*, 1 (2012), 275–287.

[124] Novák, Petr, Vyskočil, Jiří, and Wally, Bernhard. The digital twin as a core component for industry 4.0 smart production planning. *IFAC-PapersOnLine 53*, 2 (2020), 10803–10809.

[125] Öner-Közen, Miray, and Ehm, Hans. Dynamic price and lead time quotation under semiconductor industry related challenges. In *2018 Winter Simulation Conference (WSC)* (2018), IEEE, pp. 3386–3396.

[126] Otto, Alena, Otto, Christian, and Scholl, Armin. Systematic data generation and test design for solution algorithms on the example of salbpgen for assembly line balancing. *European Journal of Operational Research 228*, 1 (2013), 33–45.

[127] Ouelhadj, Djamila, and Petrovic, Sanja. A survey of dynamic scheduling in manufacturing systems. *Journal of scheduling 12*, 4 (2009), 417.

[128] Özdamar, Linet, and Barbarosoglu, Gülay. An integrated lagrangean relaxation-simulated annealing approach to the multi-level multi-item capacitated lot sizing problem. *International Journal of production economics 68*, 3 (2000), 319–331.

[129] Palaka, Kondalarao, Erlebacher, Steven, and Kropp, Dean H. Lead-time setting, capacity utilization, and pricing decisions under lead-time dependent demand. *IIE transactions 30*, 2 (1998), 151–163.

[130] Pan, An, and Choi, Tsan-Ming. An agent-based negotiation model on price and delivery date in a fashion supply chain. *Annals of Operations Research 242*, 2 (2016), 529–557.

[131] Pistikopoulos, EN. Uncertainty in process design and operations. *Computers & Chemical Engineering 19* (1995), 553–563.

[132] Pitakaso, Rapeepan, Almeder, Christian, Doerner, Karl F, and Hartl, Richard F. Combining population-based and exact methods for multi-level capacitated lot-sizing problems. *International Journal of Production Research 44*, 22 (2006), 4755–4771.

[133] Pitakaso, Rapeepan, Almeder, Christian, Doerner, Karl F, and Hartl, Richard F. A max-min ant system for unconstrained multi-level lot-sizing problems. *Computers & Operations Research 34*, 9 (2007), 2533–2552.

[134] Piya, Sujan. Dealing with customers enquiries simultaneously under contingent situation. *International Journal of Industrial Engineering Computations 6*, 3 (2015), 391–404.

[135] Potts, Chris N, and Strusevich, Vitaly A. Fifty years of scheduling: a survey of milestones. *Journal of the Operational Research Society 60*, 1 (2009), S41–S68.

[136] Potts, Chris N, and Van Wassenhove, Luk N. A decomposition algorithm for the single machine total tardiness problem. *Operations Research Letters 1*, 5 (1982), 177–181.

[137] Queyranne, Maurice, and Wang, Yaoguang. Single-machine scheduling polyhedra with precedence constraints. *Mathematics of Operations Research 16*, 1 (1991), 1–20.

[138] Raman, Narayan, Rachamadugu, Ram V, and Talbot, F Brian. Real-time scheduling of an automated manufacturing center. *European Journal of Operational Research 40*, 2 (1989), 222–242.

[139] Ramezanian, Reza, and Saidi-Mehrabad, Mohammad. Hybrid simulated annealing and mip-based heuristics for stochastic lot-sizing and scheduling problem in capacitated multi-stage production system. *Applied Mathematical Modelling 37*, 7 (2013), 5134–5147.

[140] Ramezanian, Reza, Saidi-Mehrabad, Mohammad, and Fattahi, Parviz. Mip formulation and heuristics for multi-stage capacitated lot-sizing and scheduling problem with availability constraints. *Journal of Manufacturing Systems 32*, 2 (2013), 392–401.

[141] Ray, Saibal, and Jewkes, Elizabeth M. Customer lead time management when both demand and price are lead time sensitive. *European Journal of operational research 153*, 3 (2004), 769–781.

[142] Raza, Syed Asif, Akgunduz, Ali, and Chen, MY. A tabu search algorithm for solving economic lot scheduling problem. *Journal of Heuristics 12*, 6 (2006), 413.

[143] Robinson Jr, E Powell, and Lawrence, F Barry. Coordinated capacitated lot-sizing problem with dynamic demand: A lagrangian heuristic. *Decision Sciences 35*, 1 (2004), 25–53.

[144] Rockafellar, R Tyrrell, and Wets, Roger J-B. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of operations research 16*, 1 (1991), 119–147.

[145] Romero-Conrado, Alfonso R, Coronado-Hernandez, Jairo R, Rius-Sorolla, Gregorio, and García-Sabater, José P. A tabu list-based algorithm for capacitated multilevel lot-sizing with alternate bills of materials and co-production environments. *Applied Sciences 9*, 7 (2019), 1464.

[146] Ruiz, Ruben, Şerifoğlu, Funda Sivrikaya, and Urlings, Thijs. Modeling realistic hybrid flexible flowshop scheduling problems. *Computers & Operations Research 35*, 4 (2008), 1151–1175.

[147] Ruiz, Rubén, and Vázquez-Rodríguez, José Antonio. The hybrid flow shop scheduling problem. *European journal of operational research 205*, 1 (2010), 1–18.

[148] Sabuncuoglu, Ihsan, and Goren, Selcuk. Hedging production schedules against uncertainty in manufacturing environment with a review of robustness and stability research. *International Journal of Computer Integrated Manufacturing 22*, 2 (2009), 138–157.

[149] Sabuncuoglu, Ihsan, and Karabuk, Suleyman. Rescheduling frequency in an fms with uncertain processing times and unreliable machines. *Journal of Manufacturing Systems 18*, 4 (1999), 268–283.

[150] Sahling, Florian, Buschkühl, Lisbeth, Tempelmeier, Horst, and Helber, Stefan. Solving a multi-level capacitated lot sizing problem with multi-period setup carry-over via a fix-and-optimize heuristic. *Computers & Operations Research 36*, 9 (2009), 2546–2553.

[151] Sarin, Subhash C, Sherali, Hanif D, and Liao, Lingrui. Minimizing conditional-value-at-risk for stochastic scheduling problems. *Journal of Scheduling 17*, 1 (2014), 5–15.

[152] Sawik, Tadeusz. Integer programming approach to production scheduling for make-to-order manufacturing. *Mathematical and Computer Modelling 41*, 1 (2005), 99–118.

[153] Schluse, Michael, and Rossmann, Juergen. From simulation to experimentable digital twins: Simulation-based development and operation of complex technical systems. In *2016 IEEE International Symposium on Systems Engineering (ISSE)* (2016), IEEE, pp. 1–6.

[154] Şen, Halil, and Bülbül, Kerem. A strong preemptive relaxation for weighted tardiness and earliness/tardiness problems on unrelated parallel machines. *INFORMS Journal on Computing 27*, 1 (2015), 135–150.

[155] Shi, Li, Yan, Liang, Shuqing, Bai, Cuobo, Zhuang, and Yuanchong, Cao. Research on intelligent assembly modes of aerospace products based on digital twin. In *Journal of Physics: Conference Series* (2021), vol. 1756, IOP Publishing, p. 012011.

[156] Sifaleras, Angelo, Konstantaras, Ioannis, and Mladenović, Nenad. Variable neighborhood search for the economic lot sizing problem with product returns and recovery. *International Journal of Production Economics 160* (2015), 133–143.

[157] Silva, Marco, Poss, Michael, and Maculan, Nelson. Solution algorithms for minimizing the total tardiness with budgeted processing time uncertainty. *European Journal of Operational Research 283*, 1 (2020), 70–82.

[158] Silver, Edward A. A heuristic for selecting lot size quantities for the case of a deterministic time-varying demand rate and discrete opportunities for replenishment. *Prod. Inventory Manage. 2* (1973), 64–74.

[159] Silver, Edward A, and Meal, Harlan C. A simple modification of the eoq for the case of a varying demand rate. *Production and inventory management 10*, 4 (1969), 52–65.

[160] Smith, Wayne E, et al. Various optimizers for single-stage production. *Naval Research Logistics Quarterly 3*, 1-2 (1956), 59–66.

[161] Sousa, Jorge P, and Wolsey, Laurence A. A time indexed formulation of non-preemptive single machine scheduling problems. *Mathematical programming 54*, 1 (1992), 353–367.

[162] Spitter, JM, Hurkens, Cor AJ, De Kok, AG, Lenstra, Jan Karel, and Negenman, Ebbe G. Linear programming models with planned lead times for supply chain operations planning. *European Journal of operational research 163*, 3 (2005), 706–720.

[163] Srinivasan, V. A hybrid algorithm for the one machine sequencing problem to minimize total tardiness. *Naval Research Logistics Quarterly 18*, 3 (1971), 317–327.

[164] Stadtler, Hartmut. Multilevel lot sizing with setup times and multiple constrained resources: Internally rolling schedules with lot-sizing windows. *Operations Research 51*, 3 (2003), 487–502.

[165] Stadtler, Hartmut, and Sürie, Christopher. Description of mlclsp test instances. *Technische Universität Darmstadt, Tech. Rep* (2000).

[166] Stafford, EF, Tseng, Fan T, and Gupta, Jatinder ND. Comparative evaluation of milp flowshop models. *Journal of the Operational Research Society 56*, 1 (2005), 88–101.

[167] Sujan, Piya, Katsuhiko, Takahashi, and Katsumi, Morikawa. Simultaneous quotations with capacity planning under contingent orders. *International Journal of Operations and Quantitative Management 21*, 2 (2015), 99–125.

[168] Sujan, Piya, Takahashi, Katsuhiko, and Morikawa, Katsumi. Decision model for order acceptance (oa) in a make-to-order (mto) production system: a negotiation based approach. *Proceedings of the 9th Asia Pasific Industrial Engineering & Management Systems Conference* (2008).

[169] Sun, Yang, Fowler, John W, and Shunk, Dan L. Policies for allocating product lots to customer orders in semiconductor manufacturing supply chains. *Production Planning and Control 22*, 1 (2011), 69–80.

[170] Tang, Kwei, and Tang, Jen. Time-based pricing and leadtime policies for a build-to-order manufacturer. *Production and Operations Management 11*, 3 (2002), 374–392.

[171] Tao, Fei, and Zhang, Meng. Digital twin shop-floor: a new shop-floor paradigm towards smart manufacturing. *Ieee Access 5* (2017), 20418–20427.

[172] Tempelmeier, Horst, and Derstroff, Matthias. A lagrangean-based heuristic for dynamic multilevel multiitem constrained lotsizing with setup times. *Management Science 42*, 5 (1996), 738–757.

[173] Toledo, Claudio Fabiano Motta, de Oliveira, Renato Resende Ribeiro, and França, Paulo Morelato. A hybrid heuristic approach to solve the multi level capacitated lot sizing problem. In *2011 IEEE Congress of Evolutionary Computation (CEC)* (2011), Ieee, pp. 1194–1201.

[174] Toledo, Claudio Fabiano Motta, De Oliveira, Renato Resende Ribeiro, and França, Paulo Morelato. A hybrid multi-population genetic algorithm applied to solve the multi-level capacitated lot sizing problem with backlogging. *Computers & Operations Research 40*, 4 (2013), 910–919.

[175] Torkaman, S, Ghomi, SMT Fatemi, and Karimi, Behrooz. Hybrid simulated annealing and genetic approach for solving a multi-stage production planning with sequence-dependent setups in a closed-loop supply chain. *Applied Soft Computing 71* (2018), 1085–1104.

[176] Tuegel, Eric J, Ingraffea, Anthony R, Eason, Thomas G, and Spottswood, S Michael. Reengineering aircraft structural life prediction using a digital twin. *International Journal of Aerospace Engineering 2011* (2011).

[177] Unlu, Yasin, and Mason, Scott J. Evaluation of mixed integer programming formulations for non-preemptive parallel machine scheduling problems. *Computers & Industrial Engineering 58*, 4 (2010), 785–800.

[178] Vieira, Guilherme E, Herrmann, Jeffrey W, and Lin, Edward. Predicting the performance of rescheduling strategies for parallel machine systems. *Journal of manufacturing Systems 19*, 4 (2000), 256–266.

[179] Vieira, Guilherme E, Herrmann, Jeffrey W, and Lin, Edward. Rescheduling manufacturing systems: a framework of strategies, policies, and methods. *Journal of scheduling 6*, 1 (2003), 39–62.

[180] Wagelmans, Albert, Van Hoesel, Stan, and Kolen, Antoon. Economic lot sizing: An o (n log n) algorithm that runs in linear time in the wagner-whitin case. *Operations Research 40*, 1-supplement-1 (1992), S145–S156.

[181] Wagner, Harvey M, and Whitin, Thomson M. Dynamic version of the economic lot size model. *Management science 5*, 1 (1958), 89–96.

[182] Wang, Bing, Wang, Xiaozhi, Lan, Fengming, and Pan, Quanke. A hybrid local-search algorithm for robust job-shop scheduling under scenarios. *Applied Soft Computing 62* (2018), 259–271.

[183] Wang, Bing, Wang, Xiaozhi, and Xie, Hanxin. Bad-scenario-set robust scheduling for a job shop to hedge against processing time uncertainty. *International Journal of Production Research 57*, 10 (2019), 3168–3185.

[184] Wang, Lu, Deng, Tianhu, Shen, Zuo-Jun Max, Hu, Hao, and Qi, Yongzhi. Digital twin-driven smart supply chain. *Frontiers of Engineering Management* (2022), 1–15.

[185] Wang, Yunrui, and Wu, Zhengli. Model construction of planning and scheduling system based on digital twin. *The International Journal of Advanced Manufacturing Technology 109*, 7 (2020), 2189–2203.

[186] Watanapa, Bunthit, and Techanitisawad, Anulark. A genetic algorithm for the multi-class contingent bidding model. *OR Spectrum 27*, 4 (2005), 525–549.

[187] Watanapa, Bunthit, and Techanitisawad, Anulark. Simultaneous price and due date settings for multiple customer classes. *European Journal of Operational Research 166*, 2 (2005), 351–368.

[188] Webster, Scott. Dynamic pricing and lead-time policies for make-to-order systems. *Decision Sciences 33*, 4 (2002), 579–600.

[189] Wu, Chin-Chia, Gupta, Jatinder ND, Cheng, Shuenn-Ren, Lin, Bertrand MT, Yip, Siu-Hung, and Lin, Win-Chin. Robust scheduling for a two-stage assembly shop with scenario-dependent processing times. *International Journal of Production Research* (2020), 1–16.

[190] Wu, Tao, Akartunalı, Kerem, Song, Jie, and Shi, Leyuan. Mixed integer programming in production planning with backlogging and setup carryover: modeling and algorithms. *Discrete Event Dynamic Systems 23*, 2 (2013), 211–239.

[191] Xiao, Yiyong, Zhang, Renqian, Zhao, Qiuhong, Kaku, Ikou, and Xu, Yuchun. A variable neighborhood search with an effective local search for uncapacitated multilevel lot-sizing problems. *European Journal of Operational Research 235*, 1 (2014), 102–114.

[192] Xie, Jinxing, and Dong, Jiefang. Heuristic genetic algorithms for general capacitated lot-sizing problems. *Computers & Mathematics with applications 44*, 1-2 (2002), 263–276.

[193] Yager, Ronald R. On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Transactions on systems, Man, and Cybernetics 18*, 1 (1988), 183–190.

[194] Yan-hai, Hu, Jun-qi, Yan, Fei-fan, Ye, and Jun-he, Yu. Flow shop rescheduling problem under rush orders. *Journal of Zhejiang University-SCIENCE A 6*, 10 (2005), 1040–1046.

[195] Ye, Taofeng, Sun, Hao, and Li, Zhengyi. Coordination of pricing and leadtime quotation under leadtime uncertainty. *Computers & Industrial Engineering 102* (2016), 147–159.

[196] Yu, Haifei, Han, Songjian, Yang, Dongsheng, Wang, Zhiyong, and Feng, Wei. Job shop scheduling based on digital twin technology: A survey and an intelligent platform. *Complexity 2021* (2021).

[197] Yu, Shan-Hao, and Hung, Yi-Feng. Comparisons of three mixed integer programming models for parallel machine scheduling. In *2016 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)* (2016), IEEE, pp. 917–921.

[198] Yue, Fan, Song, Shiji, Jia, Peng, Wu, Guangping, and Zhao, Han. Robust single machine scheduling problem with uncertain job due dates for industrial mass production. *Journal of Systems Engineering and Electronics 31*, 2 (2020), 350–358.

[199] Zhang, Fuqiang, Bai, Junyan, Yang, Dongyu, and Wang, Qiang. Digital twin data-driven proactive job-shop scheduling strategy towards asymmetric manufacturing execution decision. *Scientific Reports 12*, 1 (2022), 1–19.

[200] Zhao, Xuying, Stecke, Kathryn E, and Prasad, Ashutosh. Lead time and price quotation mode selection: Uniform or differentiated? *Production and Operations Management 21*, 1 (2012), 177–193.

[201] Zhou, Guanghui, Zhang, Chao, Li, Zhi, Ding, Kai, and Wang, Chuang. Knowledge-driven digital twin manufacturing cell towards intelligent manufacturing. *International Journal of Production Research 58*, 4 (2020), 1034–1051.

[202] Zhou, Hua, and Zhou, Shuiyin. Price and due date setting under service level constraints with contingent orders. In *2009 16th International Conference on Industrial Engineering and Engineering Management* (2009), IEEE, pp. 2111–2115.