UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

INVENTORY SCHEDULING FRAMEWORK TO FULFILL MULTI-PRODUCT ORDERS WITHIN AN INTERCONNECTED PRODUCTION NETWORK

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE

By

CHRISTOPHER BOURGEOIS
Norman, Oklahoma
2021

INVENTORY SCHEDULING FRAMEWORK TO FULFILL MULTI-PRODUCT ORDERS WITHIN AN INTERCONNECTED PRODUCTION NETWORK

A THESIS APPROVED FOR THE
SCHOOL OF INDUSTRIAL AND SYSTEMS ENGINEERING

BY THE COMMITTEE CONSISTING OF

Dr. Kash Barker, Chair

Dr. Andrés González

Dr. Yifu Li

ACKNOWLEDGMENTS

First, thank you to Leili Soltanisehat for her guidance over the past year. Leili provided the original formulation, Python implementation, and illustrative data that served as the foundation of this work. Between my boundless questions and spontaneous requests, she was an invaluable resource to me. This work would not have been possible without her.

Second, thank you to Dr. Kash Barker for encouraging me and my peers to attend graduate school and for serving as my thesis advisor. Kash was always available and offered a quick reply to any and all questions I had. In addition, thank you to Drs. Andrés González and Yifu Li for sharing their time and expertise with me by serving as members of my committee.

Lastly, thank you to the faculty and staff at the University of Oklahoma for exceeding my wildest dreams over the past five years. I am indebted to all of the educators that have played a role in my journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Production facilities are critical components of the global economy and supply chain. Firms are forced to balance several competing objectives—from reducing operating costs by managing inventory to increasing profits by satisfying customer demand. The act of scheduling material through an interconnected production network is analytically challenging; therefore, stakeholders require statistical insights in order to strengthen decision making. This inventory scheduling application allows decision makers to monitor the status and impact of numerous production parameters, while aiming to mitigate the propagation of schedule risk through the system. This work extends the material planning framework from Soltanisehat *et al*. (2021) that (i) jointly represents work center and material relationships and (ii) integrates stochastic methods to assess potential risk. Using network flow optimization and Monte Carlo simulation, this extension accommodates three primary considerations: (i) safety stock and other inventory restrictions, (ii) multi-product order satisfaction, and (iii) costs derived from holding inventory and delaying orders. An illustrative example examines the impact of six experimental scenarios on order fulfillment, work center utilization, and material inventory and production.

Keywords—Material requirements planning, Monte Carlo simulation, Network flow optimization, Production schedule risk

1 INTRODUCTION

An important function of production planning is inventory management and control. The objective of modern production systems is to satisfy customer orders on time and within budget; over the past decade, organizations have adopted several strategies (e.g., requiring inventory thresholds, increasing production capacity, optimizing for various planning horizons; Chowdhury *et al*., 2021) to assist in achieving these goals.

Recently, in the wake of the COVID-19 pandemic, production firms invested in formal scheduling processes (Mohammadi, 2020) and optimum inventory management procedures (Islam *et al*., 2020) to reduce the impact of internal and external risks. Supply chain stakeholders—ranging from businesses executives to governments officials to academics—are empowered to develop rigorous frameworks to model the intricacies of material planning in production systems, while simultaneously considering the potential schedule and demand impacts caused by adverse events. This area of research has an undeniable impact on the global economy and on society.

1.1 Literature survey

Orlicky (1975) introduces the groundbreaking material requirements planning (MRP) system that combines data from the master production schedule (MPS) and the bill of materials (BOM). He also defines two primary categories of inventory: manufacturing and distribution. Manufacturing inventory encompasses raw materials delivered from suppliers and work-in-process (WIP) materials within production, including components, subassemblies, and unfinished goods. Alternatively, distribution inventory comprises final products in transportation and storage. It is important to note that the original framework Orlicky (1975) proposes does not specifically incorporate an optimization function.

While previous strategies rely on heuristics to evaluate multi-level lot sizing requirements, Steinberg and Napier (1980) model the production system as a generalized network that considers multi-level material structures. The result of the model provides optimal inventory lot sizing recommendations that minimize cost flows through production. Steinberg and Napier (1980) suggest that the framework requires algorithm optimization or perhaps parameter simplification when applied to larger systems. Zhang and Chen (2001) discuss the effectiveness of push- and pull-based strategies in production planning and explore a hybrid approach known as constant work-in-process (CONWIP), which provides greater control of WIP inventory. An integer program determines optimal lot sizing requirements and material sequencing along a single serial production line. Zhang and Chen (2001) also recommend future research in algorithm efficiency and robustness. Arbib and Marielli (2005) propose an integer program that implements a cut-and-reuse policy within a cutting-stock production with skiving option. The model considers two competing objectives: minimizing purchasing costs by reusing material scraps and minimizing production costs by limiting inventory. The novelty of the framework once again necessitates improvements in computational efficiency.

Hnaien *et al*. (2008) study parametrization of lead times under uncertainty to address concerns over computational requirements. The recommended integer program utilizes a discrete time horizon to optimize single-level production systems. When applied to multi-level production facilities, the result serves as a preprocessing procedure to significantly increase and decrease the lower and upper bounds, respectively, of the objective function. This feature is based on the branch-and-bound technique to limit the size of inventory optimization problems. In contrast, Yenisey (2006) explores the

algorithmic benefits of modeling material transformation throughout production using a flow network approach. The model considers material lead times as well as the product structures defined by the BOM. Yenisey (2006) also discusses external lot-for-lot supplier policies and internal holding policies that buffer and limit in-house inventory. The deterministic program assumes that sufficient supply exists, that process operations never fail, and that demand is consistent.

Noori *et al*. (2008) adapt the network flow model Yenisey (2006) proposes by incorporating a second competing objective to minimize total completion time of the production system. The extended model also considers fuzzy logic to account for uncertainty in conjunction with expert judgements and weights. Noori *et al*. (2008) claim that multi-objective formulations of network flow problems in the context of inventory management and requirements planning have not been considered in previous research. Mula *et al*. (2008) propose a dual framework to manage inventory within the manufacturing environment that utilizes both linear and fuzzy approaches. The fuzzy program serves as a compliment, as opposed to a replacement, to the original linear program in order to provide greater insight to decisionmakers. Lin *et al*. (2009) revisit the network flow model Yenisey (2006) suggests by adapting the framework for manufacturers of thin-film transistor liquid crystal displays. The semiconductor industry presents unique challenges where material planning is critical. Products can vary greatly in quality, so most manufacturers use an alternative BOM that details the compatibility relationships and restrictions between different suppliers. Lin *et al*. (2009) also discuss the importance and complexity of supplier evaluation and selection.

Building upon decades of research, Soltanisehat *et al*. (2021) provide a stochastic framework for predicting supplier capacity, simulating material flow, and assessing schedule risk in an interconnected production facility. The system is represented as a directed network flow optimization problem that estimates process delays via Monte Carlo simulation. The framework expands upon previous literature by delivering two novel contributions: (i) the joint representation of material and work center interdependencies and (ii) the integration of stochastic methods to assess schedule risk.

First, while previous works notate network relationships in different manners, Soltanisehat *et al*. (2021) successfully model three production relationships simultaneously: (i) material transformation, (ii) work center precedence, and (iii) material flow through work centers. The optimization program utilizes the network relationships and the unit usage constraints from the BOM to model inventory flow through the production system. Second, the framework simulates schedule risk as a random discrete variable in order to identify critical work centers based on their unused capacity. This application allows stakeholders to monitor the status of each work center and its impact on the production system. Therefore, by allocating more resources to critical work centers (e.g., increasing the capacity, creating parallel production lines, assigning more human resources), stakeholders can mitigate the propagation of process time risk.

1.2 Research objectives

The goal of this research is to extend the recent contributions from Soltanisehat *et al*. (2021). First, the framework does not consider maximum inventory levels derived from physical storage limitations. The application also does not allow decision makers to set minimum inventory requirements. In reality, this is a common policy and necessary

inventory strategy to sustain production, prevent inventory stock-outs, and satisfy customer demand when firms experience a sudden reduction of material (Macchi *et al*., 2012). As complexity increases within a system, vulnerability to adverse events (e.g., machinery failure, product expiration) also increases. Buffer inventory is an essential element in maintaining resilient production systems (Alikhani *et al*., 2021). Goncalves *et al*. (2020) survey existing literature for operations research techniques to determine optimal safety stock policies. While several safety stock determination models exist, that is outside the scope of this work.

Second, Soltanisehat *et al*. (2021) assume that customer orders will only contain a single type of product. While this is common in many papers, the assumption does not reflect the needs of modern production facilities that face increasingly complex and customizable orders from customers. In this work, the model extension restricts partial deliveries; that is, a customer order cannot be fulfilled until the required number of units for each product type are available. Only some prior scheduling algorithms consider customer orders of multiple product types (Leung *et al*., 2006).

Third, the framework by Soltanisehat *et al*. (2021) assigns a different weight to each customer order and aims to maximize the number of orders fulfilled using this prioritization scheme. Order weights are also indexed by time in a descending manner such that customer orders are fulfilled as quickly as possible. In other applications, Méndez *et al*. (2000) address the delicate balance between production cost and customer satisfaction by minimizing order tardiness and WIP inventory. Wang and Lei (2015) also consider customer deadlines and penalize late orders based on their size. In order to capture the important economic context of the production environment, the extended model in this

work adopts a new objective function that minimizes cost while considering the competing goals of minimizing inventory and minimizing order delays.

The remaining sections are organized as follows: (§2) definition of the optimization and simulation framework that highlights new extensions; (§3) illustrative example that outlines data assumptions and six experimental scenarios, followed by results and visualizations; and (§4) concluding remarks that discuss limitations as well as future opportunities.

2 METHODOLOGY

The methodology presented in this section is a direct extension of the framework Soltanisehat *et al*. (2021) propose. As such, the production system is represented as a directed network that allows material to flow through a series of nodes and connected arcs. Several types of interdependencies exist in the production network—from predecessor and successor work center interactions to material unit usage requirements. Because of these work center and material relationships, process time delays easily propagate through the network and can greatly impact production costs, inventory levels, and customer satisfaction. The optimization model defined in this section aims to evaluate the potential impact of process time risk on inventory holding costs and order late fees. In order to develop a comprehensive schedule risk profile, the optimization framework relies on probabilistic risk parameters and, therefore, is evaluated using Monte Carlo simulation. The results of multiple simulation iterations can be aggregated to provide a more holistic perspective on material flow, work center criticality, and demand fulfillment.

2.1 Sets and indices

The production system is represented by a directed network denoted by $G = (N, A)$. This ordered pair represents the set of work centers N (e.g., suppliers, operations, inspections) connected using a set of directed arcs A. The subsets of virtual supply work centers $N_{=} \subseteq N$ and virtual demand work centers $N_{-} \subseteq N$ are defined to push orders of raw materials to suppliers and to pull final products through production, respectively. Set M is defined to include all of the material types (e.g., raw material, WIP inventory, final product) that travel from work center to work center along the directed arcs. Subset $M_{-} \subseteq M$ specifically represents the final products (i.e., distribution inventory; Orlicky, 1975).

Table 1: List of sets and indices

Set	Definition
N	Set of work centers indexed by i, j, or k
$N_=$	Subset of virtual supply work centers where $N_= \subseteq N$
N_-	Subset of virtual demand work centers where $N_- \subseteq N$
M	Set of materials indexed by m, p, or q
M_-	Subset of final products where $M_- \subseteq M$
O	Set of orders indexed by o
T	Set of time indexed by t or s
D	Set of relationships between materials and work centers indexed by (m, i, p, j) such that material m at work center i is related to material p at work center j where $m, p \in M$ and $i, j \in N: i \neq j$
R	Set of risks indexed by r
R_i	Subset of risks at work center $i \in N \backslash \{N_= \cup N_-\}$ where $R_i \subseteq R$

The arcs that connect work centers are created from the BOM, which defines (i) material relationships, (ii) work center relationships, and (iii) material and work center relationships. To track the changes in material between work centers, set D is constructed using the BOM and indexed by (m, i, p, j) such that material $m \in M$ at work center $i \in N$ is related to material $p \in M$ at work center $j \in N$. Namely, material $m \in M$ from work center $i \in N$ is sent to work center $j \in N$ to be transformed into material $p \in M$.

Set O provides the customer orders that the production system aims to fulfill. The flow of material through work centers continues until the final customer order is fulfilled or until the time horizon expires, whichever event occurs first. Set T provides the discrete time horizon, and set R provides the potential time risks (e.g., machine failure, human error, material rework) that impact process times at work centers. The specific risks that can occur at non-virtual work center $i \in N \backslash \{N_= \cup N_-\}$ are represented by subset $R_i \subseteq R$. For a comprehensive list of sets and indices, see Table 1.

Table 2: List of parameters

Parameter	Definition
c_i	Capacity of work center $i \in N\backslash\{N_= \cup N_-\}$ where $c_i \in \mathbb{Z}^+$
e_i	Baseline process time of work center $i \in N\backslash\{N_= \cup N_-\}$ where $e_i \in \mathbb{Z}^+$
ε_i^r	Process time delay at work center $i \in N\backslash\{N_= \cup N_-\}$ due to risk $r \in R_i$ where $\tilde{\varepsilon}_i^r \in \mathbb{Z}^*$
a_i	Total process time of work center $i \in N\backslash\{N_= \cup N_-\}$ where $a_i \in \mathbb{Z}^+$
u^{mp}	Unit usage of material $m \in M$ to produce material $p \in M$ such that $(m, i, p, j) \in D$ and where $u^{mp} \in \mathbb{Z}^*$
b_0^m	Initial inventory of material $m \in M$ at time $t = 0$ where $b_0^m \in \mathbb{Z}^*$
$v_{\min}^m$	Minimum inventory of material $m \in M$ where $v_{\min}^m \in \mathbb{Z}^*$
$v_{\max}^m$	Maximum inventory of material $m \in M$ where $v_{\max}^m \in \mathbb{Z}^+$
h^m	Cost of holding material $m \in M$ where $h^m \in \mathbb{R}^*$
q_o^m	Quantity of final product $m \in M_-$ in order $o \in O$ where $q_o^m \in \mathbb{Z}^*$
τ_o	Deadline to fulfill order $o \in O$ where $\tau_o \in T$
f_o	Cost of delaying order $o \in O$ where $f_o \in \mathbb{R}^*$

2.2 Parameters

Every non-virtual work center $i \in N\backslash\{N_= \cup N_-\}$ has a defined capacity $c_i \in \mathbb{Z}^+$ and baseline process time $e_i \in \mathbb{Z}^+$. The actual process time $a_i \in \mathbb{Z}^+$ at work center $i \in N\backslash\{N_= \cup N_-\}$ considers the potential process delay $\tilde{\varepsilon}_i^r \in \mathbb{Z}^*$ due to each risk $r \in R_i$. Note that virtual work centers are not subject to any risks and do not have capacity limitations or process time considerations.

Considering the material transformations that occur during production, parameter $u^{mp} \in \mathbb{Z}^*$ represents the unit usage of material $m \in M$ necessary to produce material $p \in M$, as prescribed in the BOM, such that $(m, i, p, j) \in D$. Parameter $b_0^m \in \mathbb{Z}^*$ provides the initial inventory of each material $m \in M$ at time $t = 0$. Due to buffer policies and storage restrictions, this extension to the framework by Soltanisehat *et al.* (2021) considers that the inventory of each material $m \in M$ must fall between the minimum $v_{\min}^m \in \mathbb{Z}^*$ and maximum $v_{\max}^m \in \mathbb{Z}^+$ boundaries. This work also assumes that the production firm incurs holding cost $h^m \in \mathbb{R}^*$ for each unit of material $m \in M$ held during each time period.

Table 3: Discrete probability distribution of process time risk

$\tilde{\varepsilon}_i^r$	0	1	2	3	4	5
$P(\tilde{\varepsilon}_i^r)$	0.37	0.31	0.18	0.09	0.04	0.01

Another distinction from the original framework is that customer orders can contain multiple types of final products. Parameter $q_o^m \in \mathbb{Z}^*$ provides the quantity of final product $m \in M_-$ in order $o \in O$. Customer orders are also time-bound in this extension. The deadline to fulfill order $o \in O$ is given by $\tau_o \in T$. For each time period that order $o \in O$ is delayed, the production firm incurs late fee $f_o \in \mathbb{R}^*$. For a comprehensive list of parameters, see Table 2.

2.2.1 Risk calculation

In the original framework, the potential time extension $\tilde{\varepsilon}_i^r \in \mathbb{Z}^*$ for each risk $r \in R_i$ at work center $i \in N\backslash\{N_= \cup N_-\}$ is a probabilistic value generated randomly from the discrete probability distribution in Table 3. Thus, the input parameters of the optimization model will differ across each Monte Carlo simulation iteration. The total process time $a_i \in \mathbb{Z}^+$ is calculated using the formulation in equation (1) that aggregates the time impact for all sources of risk. In this instance, an arithmetic average of process delays for each risk type is calculated. Note that the total process time $a_i \in \mathbb{Z}^+$ is rounded up to the nearest integer because time is considered discrete.

$$a_i = \left\lceil e_i + \frac{1}{|R|} \sum_{r \in R_i} \tilde{\varepsilon}_i^r \right\rceil \qquad \forall\, i \in N\backslash\{N_= \cup N_-\} \qquad (1)$$

2.3 Decision variables

Five decision variables determine the results of the model. Variable $v_t^m \in \mathbb{Z}^*$ tracks the total inventory of each material $m \in M$ at time $t \in T$ during production. Material flow

Table 4: List of decision variables

Variable	Definition
v_t^m	Total inventory of material $m \in M$ at time $t \in T$ where $v_t^m \in \mathbb{Z}^*$
x_{ijt}^{mp}	Flow of material $m \in M$ to material $p \in M$ via work centers $i, j \in N$ at time $t \in T$ such that $(m, i, p, j) \in D$ and $x_{ijt}^{mp} \in \mathbb{Z}^*$
y_{it}^m	Amount of final product $m \in M\_$ arriving at virtual demand node $i \in N\_$ at time $t \in T$ where $y_{it}^m \in \mathbb{Z}^*$
z_{ot}	Fulfillment of order $o \in O$ at time $t \in T$ where $z_{ot} \in \{0,1\}$
d_{ot}	Delay of order $o \in O$ at time $t \in T$ where $d_{ot} \in \{0,1\}$

through the network is managed via variable $x_{ijt}^{mp} \in \mathbb{Z}^*$, where material $m \in M$ flows from work center $i \in N$ to work center $j \in N$ at time $t \in T$ to be transformed into material $p \in M$ such that $(m, i, p, j) \in D$. Variable $y_{it}^m \in \mathbb{Z}^*$ notes the amount of final product $m \in M\_$ that reaches virtual demand work center $i \in N\_$ at time $t \in T$. Binary variable $z_{ot} \in \{0,1\}$ indicates that order $o \in O$ is fulfilled (1) or unfulfilled (0) at time $t \in T$. Likewise, this extension defines binary variable $d_{ot} \in \{0,1\}$ to indicate that order $o \in O$ is delayed (1) or not delayed (0) at time $t \in T$. For a comprehensive list of decision variables, see Table 4.

2.4 Objective function

In contrast to Soltanisehat *et al*. (2021), the goal of this model is to minimize the total cost derived from holding inventory during production and delaying orders past their deadlines. Equation (2) provides the function that considers these competing objectives. At each time period, the production firm incurs the holding cost for each material and the late fee for each order delay.

$$\min_{v,x,y,z,d} \sum_{t \in T} \sum_{m \in M} h^m v_t^m + \sum_{t \in T} \sum_{o \in O} f_o d_{ot} \tag{2}$$

2.5 Constraints

Constraint (3) restricts the outflow from each work center to account for work center capacity limitations. This assumes that every material has a uniform impact on capacity. Note that virtual work centers do not have capacity limitations.

$$\sum_{m:(m,i,p,j)\in D} \sum_{p:(m,i,p,j)\in D} \sum_{j:(m,i,p,j)\in D} x_{ijt}^{mp} \leq c_i \qquad \begin{array}{l} \forall\, i \in N\backslash\{N_{=} \cup N_{-}\}: (m,i,p,j) \in D \\ \forall\, t \in T \end{array} \tag{3}$$

The next constraints restrict the total inventory of each material. Constraint (4) initializes the inventory of each material at the beginning of the time horizon. Constraints (5) and (6) are additions to the original framework and guarantee that the inventory of each material remains within its minimum and maximum bounds, respectively.

$$v_{t-1}^{m} = b_0^m \qquad \begin{array}{l} \forall\, m \in M \\ \forall\, t \in T: t = 1 \end{array} \tag{4}$$

$$v_t^m \geq v_{\min}^m \qquad \begin{array}{l} \forall\, m \in M \\ \forall\, t \in T \end{array} \tag{5}$$

$$v_t^m \leq v_{\max}^m \qquad \begin{array}{l} \forall\, m \in M \\ \forall\, t \in T \end{array} \tag{6}$$

Constraints (7) and (8) track inventory as material flows and transforms from work center to work center. Flow equilibrium is guaranteed by considering unit usage requirements, as prescribed by the BOM, in conjunction with inbound and outbound flows from each work center.

$$v_t^m = v_{t-1}^m - u^{mp} \sum_{q:(p,j,q,k)\in D} \sum_{k:(p,j,q,k)\in D} x_{jkt}^{pq} \qquad \begin{array}{l} \forall\, j \in N\backslash\{N_{-}\} \\ \forall\, m \in M\backslash\{M_{-}\}: (m,i,p,j) \in D \\ \forall\, t \in T: (t - a_j) \leq 0 \end{array} \tag{7}$$

$$v_t^m = \sum_{i:(m,i,p,j)\in D} x_{ij(t-a_j)}^{mp} + v_{t-1}^m - u^{mp} \sum_{q:(p,j,q,k)\in D} \sum_{k:(p,j,q,k)\in D} x_{jkt}^{pq} \qquad \begin{array}{l} \forall\, j \in N\backslash\{N_{-}\} \\ \forall\, m \in M\backslash\{M_{-}\}: (m,i,p,j) \in D \\ \forall\, t \in T: (t - a_j) \geq 1 \end{array} \tag{8}$$

Constraint (9) tracks the amount of final product that arrives at the virtual demand work center. Likewise, constraint (10) considers the quantity of each final product required

to fulfill customer orders. The value of z_{ot} is equal to 1 if sufficient final product arrives at the virtual demand work center, indicating that the order is fulfilled.

$$\sum_{m:(m,i,p,j)\in D} \sum_{i:(m,i,p,j)\in D} x_{ijt}^{mp} = y_{jt}^{p} \qquad \begin{matrix} \forall\, j \in N_{-} \\ \forall\, p \in M_{-} \\ \forall\, t \in T \end{matrix} \tag{9}$$

$$v_t^m = \sum_{i\in N_-} y_{it}^m + v_{t-1}^m - \sum_{o\in O} q_o^m z_{ot} \qquad \begin{matrix} \forall\, m \in M_{-} \\ \forall\, t \in T \end{matrix} \tag{10}$$

The concept of order deadlines and delays is an extension to the original framework from Soltanisehat *et al*. (2021). Constraint (11) restricts each order from being satisfied before its deadline, thus requiring the production firm to hold inventory. Constraint (12) guarantees that each order is only fulfilled once over the time horizon, if at all.

$$z_{ot} = 0 \qquad \begin{matrix} \forall\, o \in O \\ \forall\, t \in T: t < \tau_o \end{matrix} \tag{11}$$

$$\sum_{t\in T} z_{ot} \leq 1 \qquad \forall\, o \in O \tag{12}$$

The next two constraints track order delays for each time period. Constraint (13) states that the value of d_{ot} is equal to 0 when the time precedes the order deadline. Contrarily, constraint (14) states the value of d_{ot} is equal to 1 when the time is greater than or equal to the order deadline and the order was not fulfilled in previous periods.

$$d_{ot} = 0 \qquad \begin{matrix} \forall\, o \in O \\ \forall\, t \in T: t < \tau_o \end{matrix} \tag{13}$$

$$d_{ot} + \sum_{s=\tau_o}^{t} z_{os} = 1 \qquad \begin{matrix} \forall\, o \in O \\ \forall\, t \in T: t \geq \tau_o \end{matrix} \tag{14}$$

The final set of constraints (15) through (19) control the nature of non-negative integer variables v_t^m, x_{ijt}^{mp}, and y_{it}^m and binary variables z_{ot} and d_{ot}.

$$v_t^m \in \mathbb{Z}^* \qquad \begin{matrix} \forall\, m \in M \\ \forall\, t \in T \end{matrix} \tag{15}$$

$$x_{ijt}^{mp} \in \mathbb{Z}^* \qquad \forall\,(m,i,p,j) \in D,\ \forall\,t \in T \tag{16}$$

$$y_{it}^{m} \in \mathbb{Z}^* \qquad \forall\,i \in N_{\_},\ \forall\,m \in M_{\_},\ \forall\,t \in T \tag{17}$$

$$z_{ot} \in \{0,1\} \qquad \forall\,o \in O,\ \forall\,t \in T \tag{18}$$

$$d_{ot} \in \{0,1\} \qquad \forall\,o \in O,\ \forall\,t \in T \tag{19}$$

3 ILLUSTRATION

In order to assess the validity of the framework extensions, this section outlines an artificial example that immolates a real-world production system. The results discussed and visualized in this section describe how the behavior of the model adjusts to accommodate different input parameters. The discussion also considers how production stakeholders can use the results to inform their decision making.

3.1 Description and assumptions

In this example, consider a time horizon where $|T| = 50$ and each time period is one hour. The production system is comprised of 19 suppliers, 9 operations, and 5 inspections. These work centers comprise the set of non-virtual nodes $N \backslash \{N_= \cup N_-\}$. Each supplier has a material capacity of 45, while each operation and inspection have a material capacity of 30. All work centers have a baseline processing time of one hour and are subject to uniform sources of risk such that $|R_i| = 12$. The exact impact each risk has on the process time at each work center is determined during each Monte Carlo simulation iteration. In order to push and pull material through the network, the example also considers one virtual supply node and one virtual demand node such that $|N_=| = 1$ and $|N_-| = 1$, respectively. Note that virtual work centers do not have capacity limitations or process time requirements.

This example includes three different final products. Each product is produced from 53 raw materials that contribute to 19 distinct production stages (e.g., components, subassemblies) and 60 inspections. The first product is completely unique and does not share any materials with the other two products. In comparison, the second and third products are very similar and share the same raw materials (see Table 5). The production

Table 5: Number of materials attributed to each final product

	Raw	Processed	Inspected
Exclusive to first product	53	19	60
Exclusive to second product	–	19	7
Exclusive to third product	–	19	7
Shared between second and third products	53	–	53
Total across all three products	106	57	127

Table 6: Discrete probability distribution of unit usage

u^{mp}	1	2	3	4
$P(u^{mp})$	0.94	0.02	0.01	0.03

Table 7: Quantity of final products attributed to each order

Order o	First product q_o^1	Second product q_o^2	Third product q_o^3	Deadline τ_o
1	9	–	–	10
2	2	7	–	20
3	1	5	3	30
4	–	–	9	40

facility cannot store more than 10 units of each material. According to its buffer policy, the production firm requires at least two units of each inspected material to remain in inventory at all times. At the beginning of this time horizon, the initial inventory of each inspected material is eight.

In total, the materials required to produce the final products travel along 63 interconnecting paths that account for 7 to 12 material transformations each. The unit usage of each material transformation follows the discrete probability distribution in Table 6. The set of relationships between work centers and materials represents the flow of material from the virtual supply node to suppliers, to operation and inspection work centers, and to the virtual demand node such that $|D| = 353$. This example also considers the four customer orders with varying quantities and deadlines in Table 7. The number of

Table 8: Inventory holding cost attributed to each scenario

Scenario	Raw	Processed	Inspected	Final
A	\$0.50	\$0.50	\$0.50	\$0.50
B	\$0.25	\$0.50	\$0.75	\$1.00

Table 9: Order late fee attributed to each scenario

Scenario	First order f_1	Second order f_2	Third order f_3	Fourth order f_4
I	\$5.00	\$5.00	\$5.00	\$5.00
II	\$2.50	\$5.00	\$7.50	\$10.00
III	\$10.00	\$7.50	\$5.00	\$2.50

Table 10: Run time of ten simulation iterations

Scenario	A-I	A-II	A-III	B-I	B-II	B-III
Minutes	60	65	65	30	42	44

materials required in each order and of each product type is consistent (9 and 12, respectively).

3.2 Financial scenarios

Inventory holding costs and order delay fees are the significant financial parameters that affect the objective function. Table 8 presents two different financial scenarios with different inventory holding costs. Scenario A assumes that holding cost is uniform across all materials, while Scenario B assumes that holding cost increases as materials are processed, inspected, and completed. Similarly, Table 9 defines three different financial scenarios for order late fees. Scenario I assumes that the late fee attributed to each customer order is the same. In comparison, Scenarios II and III prioritize orders with later and earlier deadlines, respectively. The combination of different holding costs and late fees create six distinct cost environments (A-I, A-II, A-III, B-I, B-II, and B-III). By evaluating the interactions between

different cost schemes, stakeholders are able to identify critical materials and work centers while also evaluating pricing strategies.

3.3 Experimental results

The framework defined in §2 is programmed in Python using Gurobi. In this implementation, the optimization model executes within a 1% optimality gap each iteration. Using the example data and financial environments previously defined, the optimization is solved 10 times for each experimental scenario (60 iterations total). The program run time for each scenario is given in Table 10. In general, the holding cost and late fee scheme that is most computationally efficient is Scenario B-I.

3.3.1 Order fulfillment

Figure 1 visualizes the time at which each order is met after 10 simulation iterations. The sequence in which orders are most frequently satisfied is Order 2, Order 1, Order 3, and Order 4. While Orders 2 and 4 are usually fulfilled by their respective deadlines, it is common for Orders 1 and 3 to be several time periods late. Specifically, Order 1 is always met at least 10 hours after its deadline, regardless of experimental scenario.

When comparing the two holding cost structures, in general, Scenario A is more likely to satisfy orders in a timely manner. Because there is no difference in holding cost between different material types, the optimization model does not have an incentive to strategically schedule material transformations during production. In Scenario B, the distribution of order fulfillment is more widespread.

Similarly, Scenario I also has a large time distribution. In Scenario II, as later orders become more important, there is the greatest amount of crossover between fulfillment

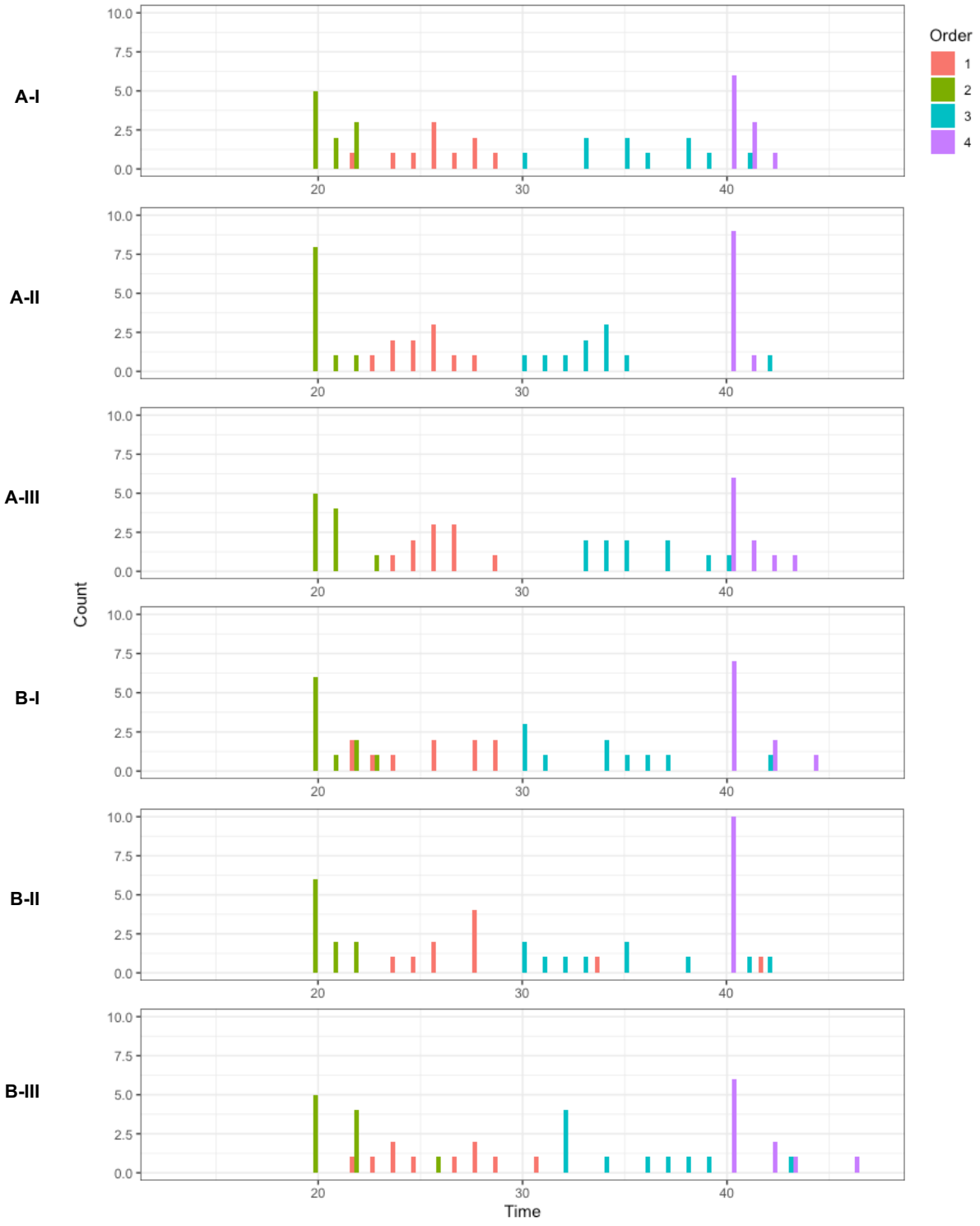

Figure 1: Timeline of order fulfillment after 10 simulation iterations

times. For instance, in Scenario B-II, Order 1 is sometimes fulfilled after Order 4, resulting in a delay greater than 30 hours. That said, the most frequent number of delays occurs in Scenario III. This is likely due to the fact that the late fee is higher for earlier orders and that orders cannot be satisfied before their deadlines.

3.3.2 Work center utilization

When scheduling the flow of material through production, decision makers can easily identify work centers that are operating close to their capacity. Stakeholders can increase the capacity, create parallel production lines, or assign more resources as necessary in order to minimize the propagation of schedule risk. Figures 2, 3 and 4 visualize the maximum utilization of each supplier, operation, and inspection, respectively, from all simulation iterations during the time horizon. In general, the results across each scenario combination are similar.

Initially, two suppliers experience a drastic increase in demand (see Figure 2). At the tenth hour, numerous suppliers reach capacity; this level of demand is sustained for the majority of the time horizon. The remaining suppliers have utilizations less than 0.25. Note the significant difference between holding cost scenarios. In Scenario A, production appears to continue as more raw materials are ordered from suppliers, which all reach full utilization by the end of the time horizon. In Scenario B, production phases out once the final orders are satisfied.

Considering the initial state of each operation, while there are a few work centers that range from 0.25 to 0.75 utilization, only one work center is in danger of reaching maximum capacity (see Figure 3). After the first 10 hours, utilization consistently remains

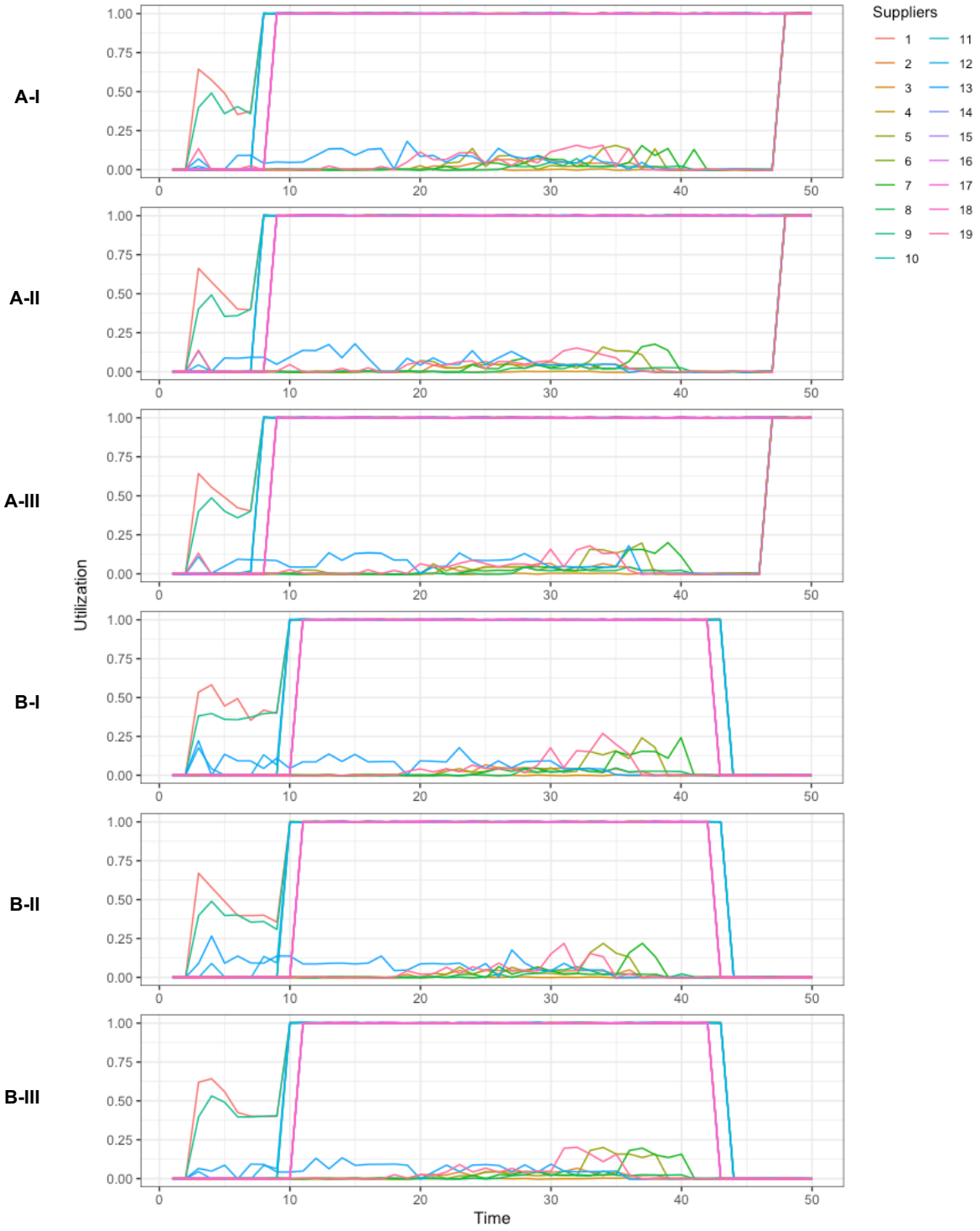

Figure 2: Maximum supplier utilization across all iterations

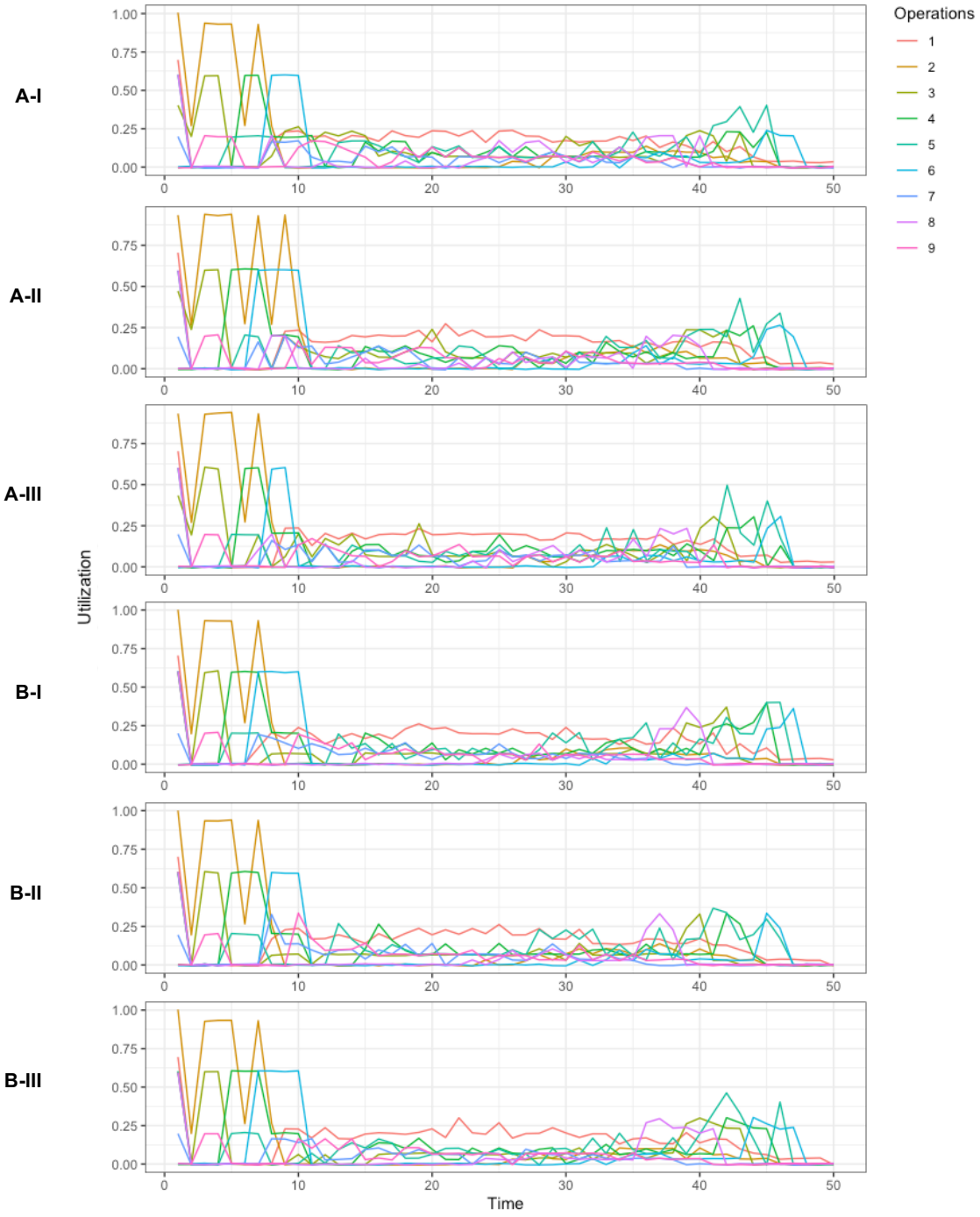

Figure 3: Maximum operation utilization across all iterations

Figure 4: Maximum inspection utilization across all iterations

beneath 0.25 for all operations. Towards the end of the time horizon, a handful of operations approaches 0.50 utilization.

For inspections, there is one work center that reaches capacity within the first few hours and remains at capacity until Order 4 is fulfilled (see Figure 4). In Scenario A, the other inspections range from 0.00 to 0.50 utilization. In Scenario B, the other inspections are vacant until approximately the tenth hour. Then, inspection utilization grows until the end of the time horizon. This tendency is expected due to the increased cost of holding inspected material.

3.3.3 Material inventory and production

Figures 5 and 6 summarize how inventory fluctuates for each material type. Overall, there are no noticeable differences between the late fee schemes. Contrarily, holding cost greatly affects the inventory strategy that the optimization model adopts. In Scenario A, inspected materials comprise the largest proportion of inventory. Processed materials significantly dwindle from hour 25 until the end of the time horizon. In preparation for customer orders, the model often holds 6 units of final product. In contrast, Scenario B capitalizes on the low holding cost of raw and processed materials, which covers the majority of inventory. Inspected materials from initial inventory are quickly transformed into other materials to minimize holding costs. Because they have the greatest holding costs, final products are finished just-in-time to fulfill customer orders. In all scenario combinations, materials are unlikely to reach the maximum holding capacity of 10.

Lastly, Figures 7 and 8 consider the cumulative production of each final product. Most notably, Scenario A introduces the greatest variation between runs, whereas Scenario B standardizes appropriate inventory levels over time. Scenarios II and III also lead to

greater deviation in inventory because the late fee attributed to each order is not the same; therefore, the model prioritizes certain final products over others. The product with the greatest variation between simulation iterations is Product 3 (likely due to the fact that this product is only required for Orders 3 and 4, which have the furthest deadlines).

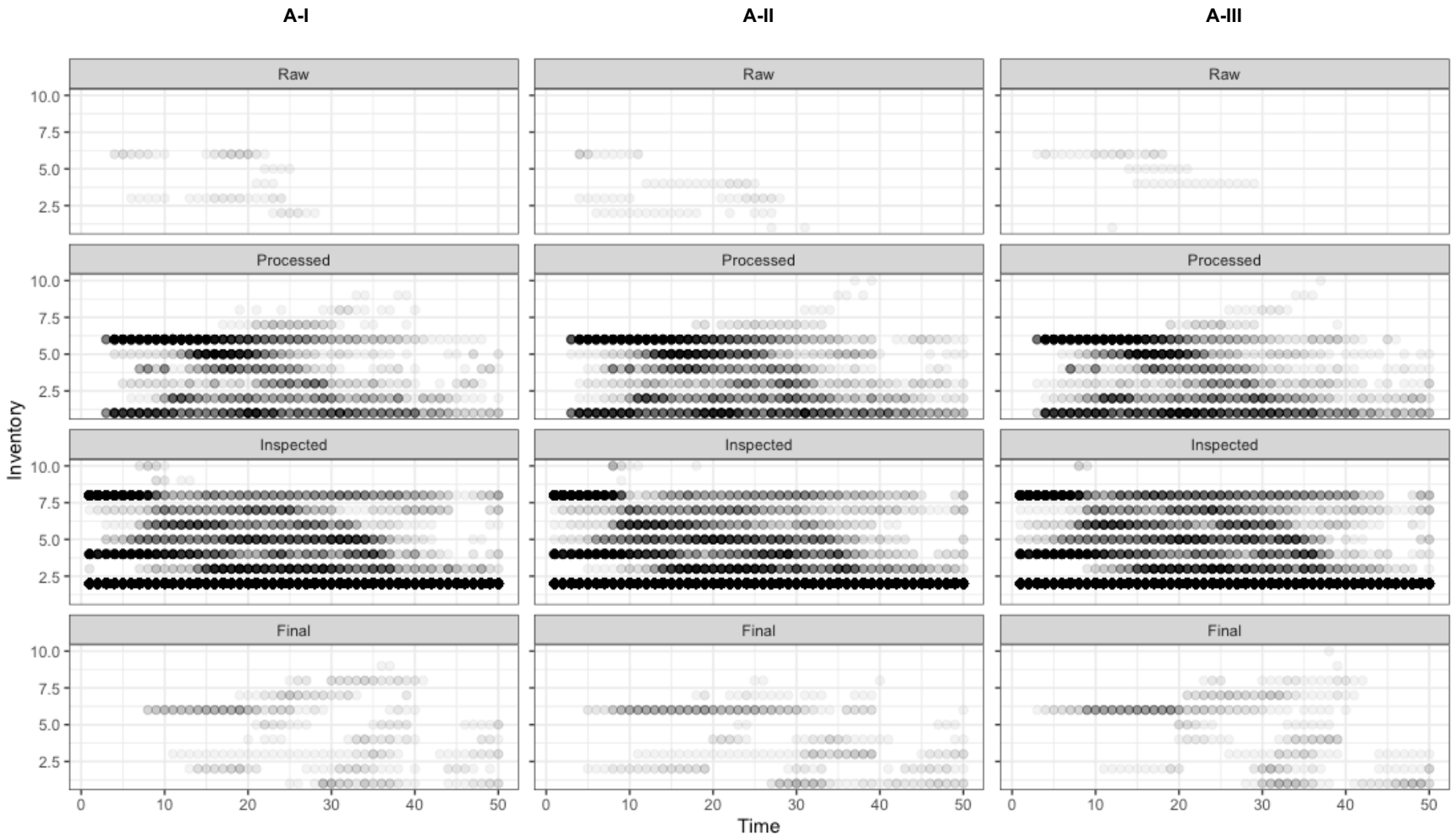

Figure 5: Inventory held during each iteration of Scenario A

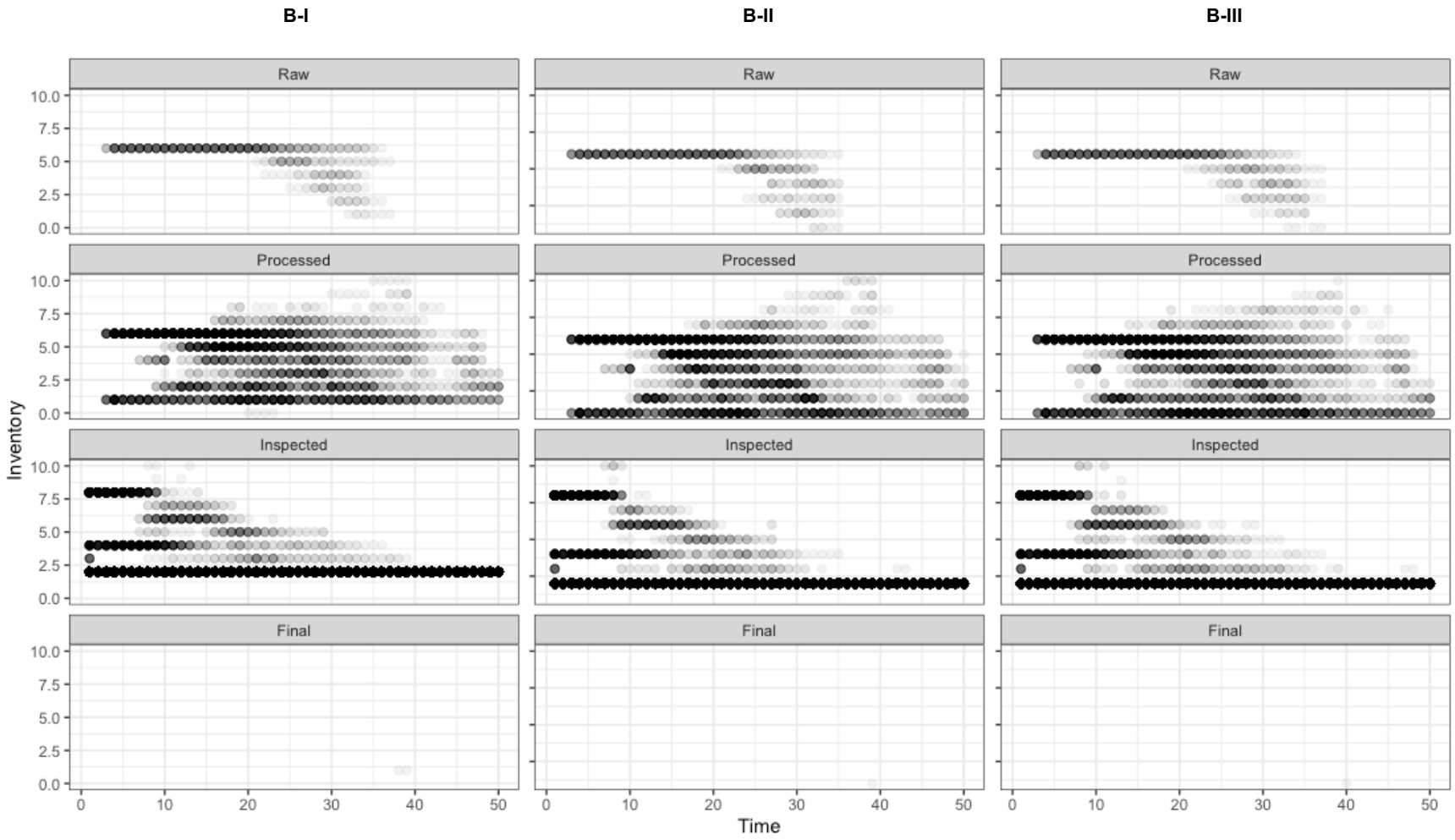

Figure 6: Inventory held during each iteration of Scenario B

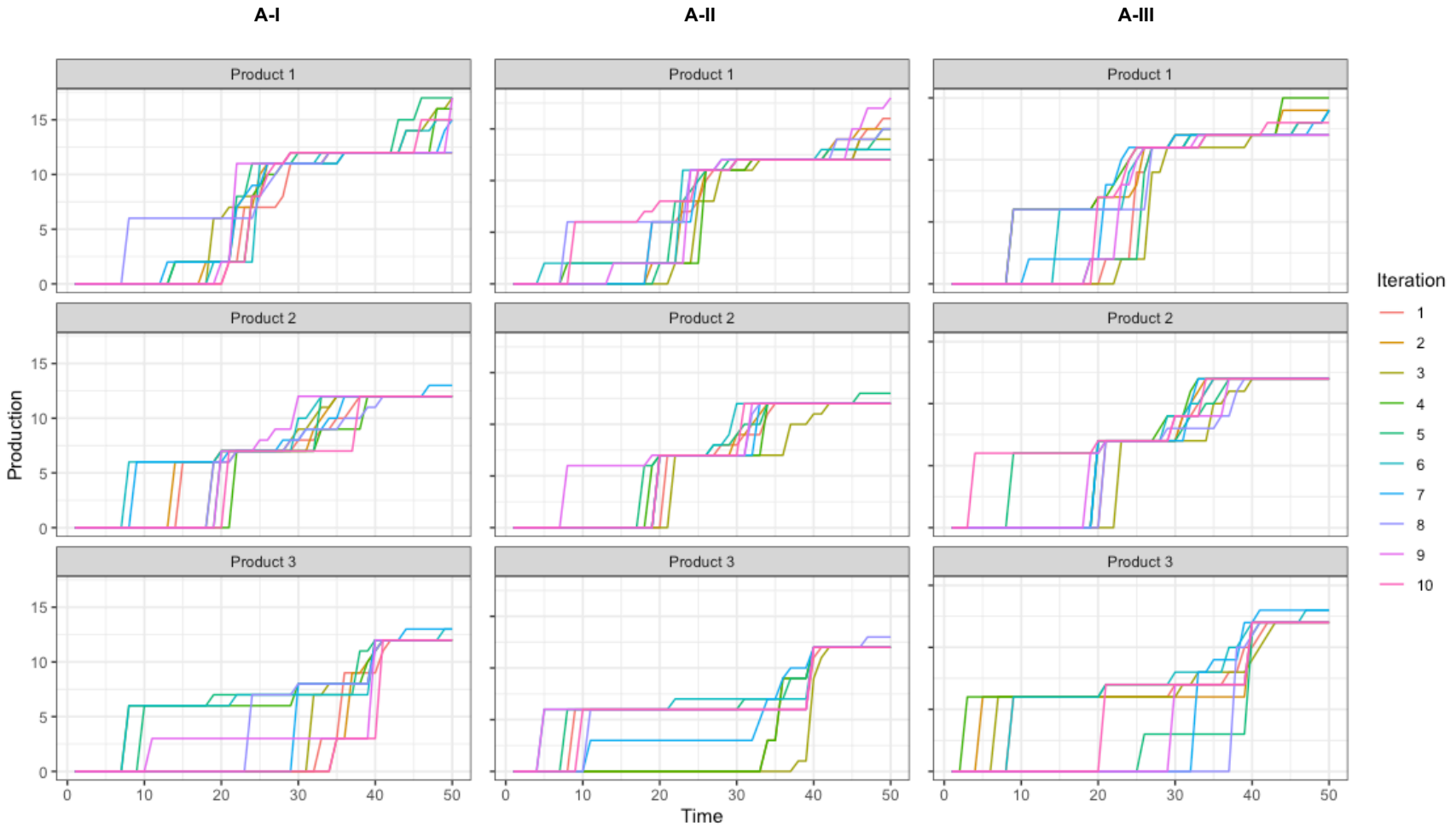

Figure 7: Cumulative production for each final product in Scenario A

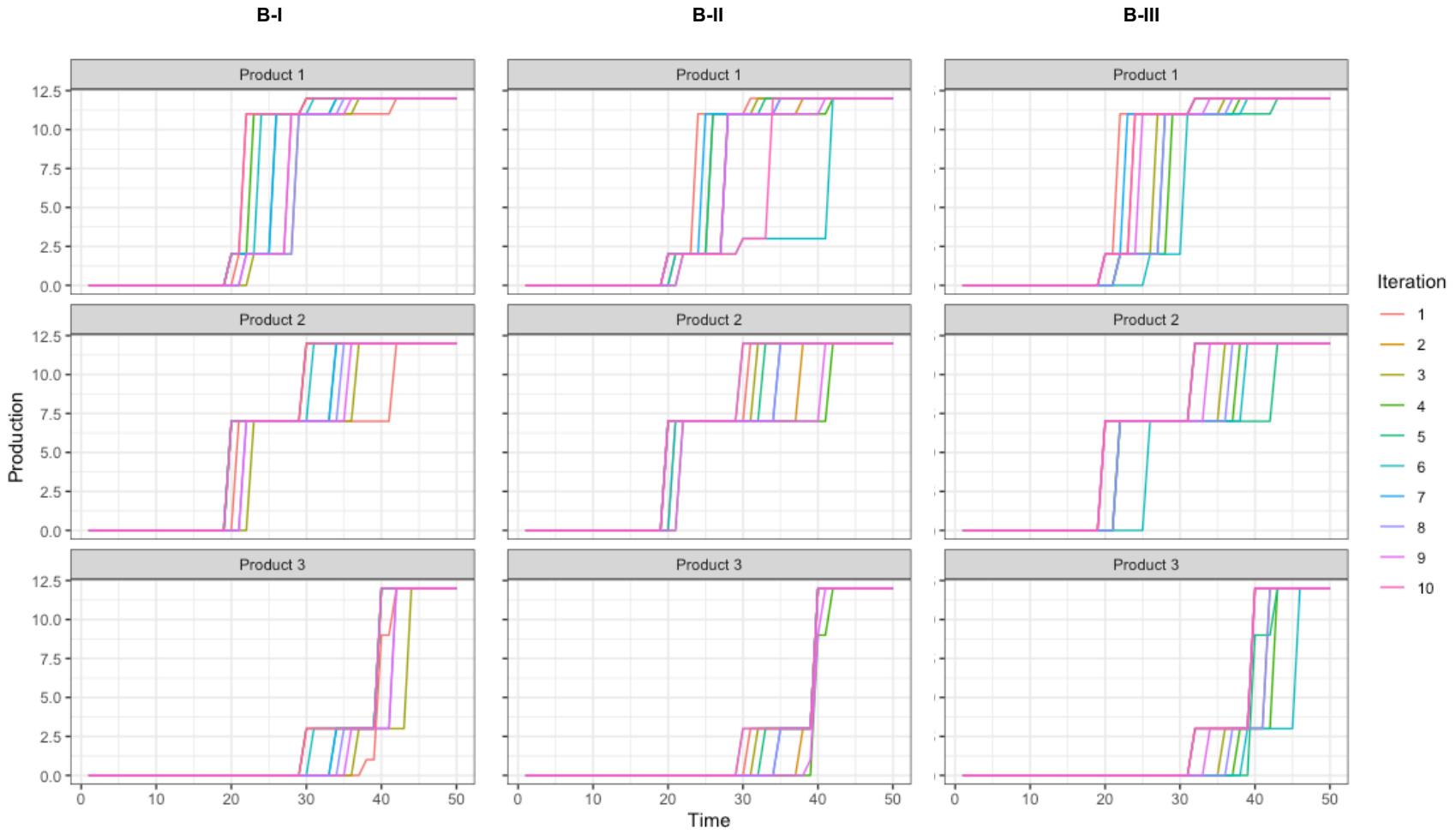

Figure 8: Cumulative production for each final product in Scenario B

4 CONCLUSION

This work presents several extensions to the production inventory formulation from Soltanisehat *et al*. (2021) that (i) combines interdependencies between work centers and materials and (ii) estimates the probabilistic impact of schedule risk. Expanding upon the network flow optimization and Monte Carlo simulation, this extension presents three modifications to the original framework: (i) the addition of inventory buffer and storage parameters, (ii) restrictions that allow multi-product customer orders, and (iii) the minimization of holding costs and order late fees.

The illustrative example in §3 validates the efficacy and applicability of the model to decision makers. The results and discussion of six potential scenarios demonstrate the potential analytical benefits for existing production systems. Overall, the framework invites stakeholders to monitor the status and impact of numerous production parameters in the hopes of mitigating the propagation of schedule risk.

4.1 Future opportunities

In future work, this framework should be applied in a real-world setting to an existing production system. A production firm can provide a more accurate network of work centers and materials with precise relationships and interdependencies, as well as more cost parameters related to production, purchasing, pricing, etc. that can contribute to the objective function. Access to sales information will also deliver more accurate demand forecasting.

Considering risk estimation, future work should also improve upon the assumption that risk follows a discrete probability distribution by incorporating a more robust

prediction model. In reality, different risks likely follow different distributions. Moreover, the framework could incorporate a risk weighting scheme to improve aggregation.

Finally, future research should explore the computational challenges that impact run time. Increasing the time horizon greatly affects the number of variables and constraints. Future iterations of this framework should explore and justify long-term planning.

REFERENCES

Alikhani, R., Torabi, S. A. & Altay, N. (2021). Retail supply chain network design with concurrent resilience capabilities. *International Journal of Production Economics*, *234*, Article 108042.

Arbib, C. & Marinelli, F. (2005). Integrating process optimization and inventory planning in cutting-stock with skiving option: An optimization model and its application. *European Journal of Operational Research*, *163*(3), 617-630.

Chowdhury, P., Paul, S. K., Kaisar, S. & Moktadir, M. A. (2021). COVID-19 pandemic related supply chain studies: A systematic review. *Transportation Research Part E: Logistics and Transportation Review*, *148*, Article 102271.

Goncalves, J. N. C., Carvalho, M. S. & Cortez, P. (2020). Operations research models and methods for safety stock determination: A review. *Operations Research Perspectives*, *7*, Article 100164.

Hnaien, F., Dolgui, A. & Ould Louly, M. A. (2008). Planned lead time optimization in material requirement planning environment for multilevel production systems. *Journal of Systems Science and Systems Engineering*, *17*(2), 132-155.

Islam, M. T., Azeem, A., Jabir, M., Paul, A. & Paul, S. K. (2020). An inventory model for a three-stage supply chain with random capacities considering disruptions and supplier reliability. *Annals of Operations Research*. Advance online publication.

Leung, J. Y. T., Li, H. & Pinedo, M. (2006). Scheduling orders for multiple product types with due date related objectives. *European Journal of Operational Research*, *168*(2), 370-389.

Lin, J. T., Chen, T. L. & Lin, Y. T. (2009). Critical material planning for TFT-LCD production industry. *International Journal of Production Economics*, *122*(2), 639-655.

Macchi, M., Kristjanpoller, F., Garetti, M., Arata, A. & Fumagalli, L. (2012). Introducing buffer inventories in the RBD analysis of process production systems. *Reliability Engineering & System Safety*, *104*, 84-95.

Méndez, C. A., Henning, G. P. & Cerdá, J. (2000). Optimal scheduling of batch plants satisfying multiple product orders with different due dates. *Computers & Chemical Engineering*, *24*(9), 2223-2245.

Mohammadi, M. (2020). Designing an integrated reliable model for stochastic lot-sizing and scheduling problem in hazardous materials supply chain under disruption and demand uncertainty. *Journal of Cleaner Production*, *274*, Article 122621.

Mula, J., Poler, R. & Garcia-Sabater, J. P. (2008). Capacity and material requirement planning modelling by comparing deterministic and fuzzy models. *International Journal of Production Research*, *46*(20), 5589-5606.

Noori, S., Feylizadeh, M. R., Bagherpour, M., Zorriassatine, F. & Parkin, R. M. (2008). Optimization of material requirement planning by fuzzy multi-objective linear programming. *Journal of Engineering Manufacture*, *222*(7), 887-900.

Orlicky, J. (1975). *Material Requirements Planning*. McGraw-Hill.

Soltanisehat, L., Ghorbani-Renani, N., Barker, K. & González, A. D. (2021). *Assessing production scheduling risk from interconnected sources: Application to pandemic health equipment*. In progress.

Steinberg, E. & Napier, H. A. (1980). Optimal multi-level lot sizing for requirements planning systems. *Management Science*, *26*(12), 1258-1271.

Wang, G. & Lei, L. (2015). Integrated operations scheduling with delivery deadlines. *Computers & Industrial Engineering*, *85*, 177-185.

Yenisey, M. M. (2006). A flow-network approach for equilibrium of material requirements planning. *International Journal of Production Economics*, *102*(2), 317-332.

Zhang, W. & Chen, M. (2001). A mathematical programming model for production planning using CONWIP. *International Journal of Production Research*, *39*(12), 2723-2734.

APPENDIX A: Python implementation of model

```
# General Libraries
import os
import time as tt

# Optimization Libraries
import csv
import pandas as pd
from gurobipy import *

# Monte Carlo Libraries
import pandas as pd
import numpy as np
import seaborn as sns
import itertools
import pylab
import numpy
import random
import time as tt

def ScheduleRisk(Time_Horizon, Simulation_Runs):

    # Define and read the path of folder
    os.getcwd()

    # Creating the new path for the outputs inside the current directory
    OutputDir = 'Outputs'
    if not os.path.exists(OutputDir):
        os.makedirs(OutputDir)

    # ------------------------------------------------------------
    # Define the mathematical model sets, indices, and parameters
    # ------------------------------------------------------------

    # Track the computational time
    start = tt.time()

    # Set the time horizon of the mathematical model
    timeN = list()
    Time = Time_Horizon
    for t in range(1, Time + 1):
        timeN.append(t)

    # Define sets, indices and parameters associated with work centers
    WorkStations= open('work_center.csv', 'r')
    csv_WorkStations = csv.reader(WorkStations)

    mydict_workstation_capacity = {}
    mydict_operation_capacity = {}
    mydict_operation_time = {}
    mydict_inspection_capacity = {}
    mydict_inspection_time = {}
    mydict_VS_time = {}
    mydict_demand_time = {}
    mydict_Supplier_capacity = {}
    mydict_Supplier_time = {}
    mydict_Allnode_time = {}

    for row in csv_WorkStations:
        if 'O' in row[1]:
            mydict_operation_capacity[(row[0])] = int(row[3])
            mydict_operation_time[(row[0])] = int(row[2])
        if 'I' in row[1]:
            mydict_inspection_capacity[(row[0])] = int(row[3])
            mydict_inspection_time[(row[0])] = int(row[2])
        if 'V' in row[1]:
            mydict_VS_time[(row[0])] = int(row[2])
        if 'D' in row[1]:
            mydict_demand_time[(row[0])] = int(row[2])
        if 'S' in row[1]:
            mydict_Supplier_capacity[(row[0])] = int(row[3])
```

```
        mydict_Supplier_time[(row[0])] = int(row[2])
    mydict_Allnode_time[(row[0])] = int(row[2])
    mydict_workstation_capacity[(row[0])] = int(row[3])

Inode, inspection_cap_value = multidict(mydict_inspection_capacity)
Inode, inspection_time_value = multidict(mydict_inspection_time)

Onode, operation_cap_value = multidict(mydict_operation_capacity)
Onode, operation_time_value = multidict(mydict_operation_time)

VSnode, VS_time = multidict(mydict_VS_time)

Dnode, demand_time_value = multidict(mydict_demand_time)

Snode, Supplier_cap_value = multidict(mydict_Supplier_capacity)
Snode, Supplier_time_value = multidict(mydict_Supplier_time)

Anode, time = multidict(mydict_Allnode_time)

# Define sets, indices and parameters associated with materials
inventory = open('material.csv', 'r')
csv_inventory = csv.reader(inventory)

mydict_Omaterial_inventory = {}
mydict_Imaterial_inventory = {}
mydict_VS_inventory = {}
mydict_Fmaterial_inventory = {}
mydict_Rmaterial_inventory = {}
mydict_Amaterial_inventory = {}

mydict_Omaterial_min = {}
mydict_Imaterial_min = {}
mydict_VS_min = {}
mydict_Fmaterial_min = {}
mydict_Rmaterial_min = {}
mydict_Amaterial_min = {}

mydict_Omaterial_max = {}
mydict_Imaterial_max = {}
mydict_VS_max = {}
mydict_Fmaterial_max = {}
mydict_Rmaterial_max = {}
mydict_Amaterial_max = {}

mydict_Omaterial_cost = {}
mydict_Imaterial_cost = {}
mydict_VS_cost = {}
mydict_Fmaterial_cost = {}
mydict_Rmaterial_cost = {}
mydict_Amaterial_cost = {}

for row in csv_inventory:
    if 'O' in row[1]:
        mydict_Omaterial_inventory[(row[0])] = int(row[2])
        mydict_Omaterial_min[(row[0])] = int(row[3])
        mydict_Omaterial_max[(row[0])] = int(row[4])
        mydict_Omaterial_cost[(row[0])] = float(row[5])
    if 'I' in row[1]:
        mydict_Imaterial_inventory[(row[0])] = int(row[2])
        mydict_Imaterial_min[(row[0])] = int(row[3])
        mydict_Imaterial_max[(row[0])] = int(row[4])
        mydict_Imaterial_cost[(row[0])] = float(row[5])
    if 'V' in row[1]:
        mydict_VS_inventory[(row[0])] = int(row[2])
        mydict_VS_min[(row[0])] = int(row[3])
        mydict_VS_max[(row[0])] = int(row[4])
        mydict_VS_cost[(row[0])] = float(row[5])
    if 'D' in row[1]:
        mydict_Fmaterial_inventory[(row[0])] = int(row[2])
        mydict_Fmaterial_min[(row[0])] = int(row[3])
        mydict_Fmaterial_max[(row[0])] = int(row[4])
        mydict_Fmaterial_cost[(row[0])] = float(row[5])
    if 'S' in row[1]:
```

```
        mydict_Rmaterial_inventory[(row[0])] = int(row[2])
        mydict_Rmaterial_min[(row[0])] = int(row[3])
        mydict_Rmaterial_max[(row[0])] = int(row[4])
        mydict_Rmaterial_cost[(row[0])] = float(row[5])
    mydict_Amaterial_inventory[(row[0])] = int(row[2])
    mydict_Amaterial_min[(row[0])] = int(row[3])
    mydict_Amaterial_max[(row[0])] = int(row[4])
    mydict_Amaterial_cost[(row[0])] = float(row[5])

Omaterial, OInventory = multidict(mydict_Omaterial_inventory)
Imaterial, IInventory = multidict(mydict_Imaterial_inventory)
VSmaterial, VSInventory = multidict(mydict_VS_inventory)
Fmaterial, FInventory = multidict(mydict_Fmaterial_inventory)
Rmaterial, RInventory = multidict(mydict_Rmaterial_inventory)
Amaterial, AInventory = multidict(mydict_Amaterial_inventory)

Omaterial, Omin = multidict(mydict_Omaterial_min)
Imaterial, Imin = multidict(mydict_Imaterial_min)
VSmaterial, VSmin = multidict(mydict_VS_min)
Fmaterial, Fmin = multidict(mydict_Fmaterial_min)
Rmaterial, Rmin = multidict(mydict_Rmaterial_min)
Amaterial, Amin = multidict(mydict_Amaterial_min)

Omaterial, Omax = multidict(mydict_Omaterial_max)
Imaterial, Imax = multidict(mydict_Imaterial_max)
VSmaterial, VSmax = multidict(mydict_VS_max)
Fmaterial, Fmax = multidict(mydict_Fmaterial_max)
Rmaterial, Rmax = multidict(mydict_Rmaterial_max)
Amaterial, Amax = multidict(mydict_Amaterial_max)

Omaterial, Ocost = multidict(mydict_Omaterial_cost)
Imaterial, Icost = multidict(mydict_Imaterial_cost)
VSmaterial, VScost = multidict(mydict_VS_cost)
Fmaterial, Fcost = multidict(mydict_Fmaterial_cost)
Rmaterial, Rcost = multidict(mydict_Rmaterial_cost)
Amaterial, Acost = multidict(mydict_Amaterial_cost)

# Define sets, indices and parameters associated with orders
order = open('order.csv', 'r')
csv_order = csv.reader(order)

mydict_order_deadline = {}
mydict_order_fee = {}

for row in csv_order:
    mydict_order_deadline[(row[0])] = int(row[1])
    mydict_order_fee[(row[0])] = float(row[2])

order, deadline = multidict(mydict_order_deadline)
order, fee = multidict(mydict_order_fee)

# Define sets, indices and parameters associated with order relations
order_rel = open('order_rel.csv', 'r')
csv_order_rel = csv.reader(order_rel)

mydict_order_s = {}

for row in csv_order_rel:
    mydict_order_s[(row[0], row[1])] = int(row[2])

order_rel, size = multidict(mydict_order_s)

# Define sets, indices and parameters associated with the material and work center relations
interdependecy_relations = open('relation.csv','r')
csv_interdependecy_relations = csv.reader(interdependecy_relations)

mydict_interdependecy_relations = {}
mydict_Workstation_relations = {}
mydict_material_relations = {}

for row in csv_interdependecy_relations:
    mydict_interdependecy_relations[row[0], row[1], row[2], row[3]] = row[4]
    mydict_Workstation_relations[row[1], row[3]] = int(row[4])
```

```
        mydict_material_relations[(row[0], row[2])] = int(row[4])

    relations, value1 = multidict(mydict_interdependecy_relations)
    Wrelations, value2 = multidict(mydict_Workstation_relations)
    Mrelations, quantity = multidict(mydict_material_relations)

    # ----------------------------------------------------------
    # Network flow optimization model
    # ----------------------------------------------------------

    def Mathematical_Model (work_center_risk):

        # Set up the mathematical model and name it as HW_M
        HW_M = Model()
        HW_M.setParam('MIPGap', 0.01)

        # Constraints (15),(16),(17),(18),(19): manage decision variables
        Y = HW_M.addVars(Dnode, Fmaterial, timeN, vtype=GRB.INTEGER, lb=0, name="Y")
        X = HW_M.addVars(relations, timeN, vtype=GRB.INTEGER, lb=0, name='X')
        V = HW_M.addVars(Amaterial, timeN, vtype=GRB.INTEGER, lb=0, name='V')
        Z = HW_M.addVars(order, timeN, vtype=GRB.BINARY, name='Z')
        D = HW_M.addVars(order, timeN, vtype=GRB.BINARY, name='D')

        # Constraints (3): work center capacity
        for t in timeN:
            for m,i,p,j in relations:
                if i not in Dnode and i not in VSnode:
                        HW_M.addConstr(quicksum(X[m, i, p, j, t] for m, i, p, j in
                                                relations.select('*', i, '*', '*')) <=
                                       (mydict_workstation_capacity[(i)]))

        # Constraints (4),(7),(8): Flow balance and unit usage
        for t in timeN:
            for m, i, p, j in relations:
                if j not in Dnode:
                    for p, j, q, k in relations:
                        if (m,p) in Mrelations:
                            if t == 1:
                                if (t - int(mydict_Allnode_time [(j)]) -work_center_risk[j]<= 0
):
                                    HW_M.addConstr(mydict_Amaterial_inventory[m]-
((mydict_material_relations[(m,p)]) * quicksum(X[p,j,q,k,t] for p,j,q,k in
relations.select(p,j,'*','*'))) == V[m,t] ,name="(3)(6) Flow equilibrium at t=1 by considering
unit usage")
                                else:
                                    HW_M.addConstr(quicksum(X[m,i,p,j,t-int(mydict_Allnode_time
[(j)])-work_center_risk[j]] for m,i,p,j in relations.select(m,'*',p, j)) +
mydict_Amaterial_inventory[m]- ((mydict_material_relations[(m,p)]) * quicksum(X[p,j,q,k,t] for
p,j,q,k in relations.select(p,j,'*','*'))) == V[m,t] ,name="(3)(7) Flow equilibrium at t=1 by
considering unit usage")
                            else:
                                if (t - int(mydict_Allnode_time [(j)]) -work_center_risk[j]<=0 ):
                                    HW_M.addConstr(V[m, t - 1] - ((mydict_material_relations[(m,
p)]) * quicksum(X[p, j, q, k, t] for p, j, q, k in relations.select(p, j, '*', '*'))) == V[m,
t],name="(-)(6) Flow equilibrium at t>1 by considering unit usage")
                                else:
                                    HW_M.addConstr(quicksum(X[m, i, p, j, t -
int(mydict_Allnode_time[(j)])-work_center_risk[j]] for m, i, p, j in relations.select(m, '*', p,
j)) + V[m, t - 1] - ((mydict_material_relations[(m, p)]) * quicksum(X[p, j, q, k, t] for p, j, q,
k in relations.select(p, j, '*', '*'))) == V[m, t], name="(-)(7) Flow equilibrium at t>1 by
considering unit usage")

        # Constraints (9): Deliver final product to demand nodes
        for t in timeN:
            for m, i, p, j in relations:
                if j in Dnode:
                    HW_M.addConstr(quicksum(X[m, i, p, j,t] for m, i, p, j in
relations.select('*', '*', p, j)) == Y[j, p,t], name="(8) Final product reaches virtual demand
work center")

        # Constraints (10): Satisfy orders by considering sizes
        for t in timeN:
            for p in Fmaterial:
```

```
                if t == 1:
                    HW_M.addConstr(quicksum(Y[j,p,t] for j in Dnode) +
mydict_Amaterial_inventory[p] - quicksum(mydict_order_s[p,o]* Z[o,t] for p,o in
order_rel.select(p, '*')) == V[p,t], name="(9) fulfill orders at t=1 by considering size")
                else:
                    HW_M.addConstr(quicksum(Y[j,p,t] for j in Dnode) + V[p, t-1] -
quicksum(mydict_order_s[p,o]* Z[o,t] for p,o in order_rel.select(p, '*')) == V[p,t], name="(9)
fulfill orders at t>1 by considering size")

        # Constraints (12): Ensure each order only met once
        for o in order:
            HW_M.addConstr(quicksum(Z[o,t] for t in timeN) <= 1, name="(10) fulfill order once")

        # Constraints (11),(13),(14): Track order delays
        for t in timeN:
            for o in order:
                if t < mydict_order_deadline[o]:
                    HW_M.addConstr(Z[o,t] == 0, name="(new) hold inventory until deadline")
                    HW_M.addConstr(D[o,t] == 0, name="(11) order delay before deadline")
                else:
                    HW_M.addConstr(D[o,t] + quicksum(Z[o,s] for s in
timeN[(mydict_order_deadline[o]-1):t]) == 1, name="(12) order delay after deadline")

        # Constraints (5): Minimum inventory constraints
        for t in timeN:
            for m in Amaterial:
                HW_M.addConstr(V[m,t] >= mydict_Amaterial_min[m], name="(4) min inventory")

        # Constraints (6): Maximum inventory constraints
        for t in timeN:
            for m in Amaterial:
                HW_M.addConstr(V[m,t] <= mydict_Amaterial_max[m], name="(5) max inventory")

        # Equation (2): Objective function that calculates total holding and delay costs
        OBJ = quicksum(mydict_Amaterial_cost[m]*V[m,t] for m in Amaterial for t in timeN) +
quicksum(mydict_order_fee[o]*D[o,t] for o in order for t in timeN)

        # Solve the mathematical model and get the results
        HW_M.setObjective(OBJ, GRB.MINIMIZE)
        HW_M.optimize()
        flow = HW_M.getAttr('X', X)
        final_product = HW_M.getAttr('X', Y)
        met_order = HW_M.getAttr('X', Z)
        delay_order = HW_M.getAttr('X', D)
        inventory = HW_M.getAttr('X', V)

        # Add material index to order
        met_order_rel = {}
        for t in timeN:
            for m,o in order_rel:
                met_order_rel[(m,o,t)] = met_order[(o,t)]

        # Return the mathematical model outputs
        return met_order_rel, final_product, flow, inventory, delay_order

    # ------------------------------------------------------------
    # Monte Carlo simulation
    # ------------------------------------------------------------

    def Risk(filename):
        with open(filename,'r') as riskfile:
            csv_Risk = csv.DictReader(open(filename))
            Risk_Measure = {}
            NO_Risk_Measure={}
            for row in csv_Risk:
                R_key = row.pop('workcenter')
                Risk_Measure[R_key] = row

            for key ,value in Risk_Measure.items():
                for k, v in value.items():
                        a=random.uniform(0, 1)
                        if  0.0 <= a <= 0.01:
                            Risk_Measure[key][k]= 5* int(Risk_Measure[key][k])
```

```
                        elif 0.01 <= a <= 0.05:
                            Risk_Measure[key][k]= 4* int(Risk_Measure[key][k])
                        elif 0.05 <= a <= 0.14:
                            Risk_Measure[key][k]= 3* int(Risk_Measure[key][k])
                        elif 0.14 <= a <= 0.32:
                            Risk_Measure[key][k]= 2* int(Risk_Measure[key][k])
                        elif 0.32 <= a <= 0.63:
                            Risk_Measure[key][k]= 1* int(Risk_Measure[key][k])
                        elif 0.63 <= a <= 1:
                            Risk_Measure[key][k]= 0 * int(Risk_Measure[key][k])

            Risk_value= {}
            for key ,value in Risk_Measure.items():
                Risk_value[key]=0
                b = 0
                for k, v in value.items():
                   b = (b + int(Risk_Measure[key][k]))
                Risk_value[key]= math.ceil(b/len(value))

            return Risk_value

    # ------------------------------------------------------------------

    def MonteCarlo (Simulation_num, filename):

        counter = 1
        while counter <= Simulation_num:
            ws_risk = Risk(filename)

met_order_count,final_product_count,unused_capacity_count,flow_count,inventory_count,delay_order_
count=Mathematical_Model(ws_risk)

            # Z df
            if counter ==1:
                met_order_df = pd.DataFrame.from_dict(met_order_count,
orient='index').reset_index()
                met_order_df.columns = ['index','met'+ str(counter)]
                relations2, value2 =multidict(met_order_count)
                relations2 = [list(elem) for elem in relations2]
                Met_Order= pd.DataFrame(relations2, columns=['material', 'order', 'time'])
                Met_Order['met'+ str(counter)]=met_order_df['met'+ str(counter)]
                MetOrder_df=Met_Order
            else:
                met_order_df = pd.DataFrame.from_dict(met_order_count,
orient='index').reset_index()
                met_order_df.columns = ['index','met'+ str(counter)]
                relations2, value2 =multidict(met_order_count)
                relations2 = [list(elem) for elem in relations2]
                Met_Order= pd.DataFrame(relations2, columns=['material', 'order', 'time'])
                Met_Order['met'+ str(counter)]=met_order_df['met'+ str(counter)]
                MetOrder_df= (pd.merge(MetOrder_df, Met_Order, on=['material','order', 'time']))

            # D df
            if counter ==1:
                del_order_df = pd.DataFrame.from_dict(delay_order_count,
orient='index').reset_index()
                del_order_df.columns = ['index','del'+ str(counter)]
                relations4, value4 =multidict(delay_order_count)
                relations4 = [list(elem) for elem in relations4]
                Del_Order= pd.DataFrame(relations4, columns=['order', 'time'])
                Del_Order['del'+ str(counter)]=del_order_df['del'+ str(counter)]
                DelOrder_df=Del_Order
            else:
                del_order_df = pd.DataFrame.from_dict(delay_order_count,
orient='index').reset_index()
                del_order_df.columns = ['index','del'+ str(counter)]
                relations4, value4 =multidict(delay_order_count)
                relations4 = [list(elem) for elem in relations4]
                Del_Order= pd.DataFrame(relations4, columns=['order', 'time'])
                Del_Order['del'+ str(counter)]=del_order_df['del'+ str(counter)]
                DelOrder_df=(pd.merge(DelOrder_df, Del_Order, on=['order', 'time']))

            # V df
```

```
            if counter ==1:
                inv_df = pd.DataFrame.from_dict(inventory_count, orient='index').reset_index()
                inv_df.columns = ['index','inv'+ str(counter)]
                relations5, value5 =multidict(inventory_count)
                relations5 = [list(elem) for elem in relations5]
                Inv = pd.DataFrame(relations5, columns=['material', 'time'])
                Inv['inv'+ str(counter)]=inv_df['inv'+ str(counter)]
                Inv_df=Inv
            else:
                inv_df = pd.DataFrame.from_dict(inventory_count, orient='index').reset_index()
                inv_df.columns = ['index','inv'+ str(counter)]
                relations5, value5 =multidict(inventory_count)
                relations5 = [list(elem) for elem in relations5]
                Inv = pd.DataFrame(relations5, columns=['material', 'time'])
                Inv['inv'+ str(counter)]=inv_df['inv'+ str(counter)]
                Inv_df=(pd.merge(Inv_df, Inv, on=['material', 'time']))

            # Y df
            if counter ==1:
                fin_df = pd.DataFrame.from_dict(final_product_count,
orient='index').reset_index()
                fin_df.columns = ['index','fin'+ str(counter)]
                relations6, value6 =multidict(final_product_count)
                relations6 = [list(elem) for elem in relations6]
                Fin = pd.DataFrame(relations6, columns=['work center', 'material', 'time'])
                Fin['fin'+ str(counter)]=fin_df['fin'+ str(counter)]
                Fin_df=Fin
            else:
                fin_df = pd.DataFrame.from_dict(final_product_count,
orient='index').reset_index()
                fin_df.columns = ['index','fin'+ str(counter)]
                relations6, value6 =multidict(final_product_count)
                relations6 = [list(elem) for elem in relations6]
                Fin = pd.DataFrame(relations6, columns=['work center', 'material', 'time'])
                Fin['fin'+ str(counter)]=fin_df['fin'+ str(counter)]
                Fin_df=(pd.merge(Fin_df, Fin, on=['work center', 'material', 'time']))

            # X df
            if counter ==1:
                flow_df = pd.DataFrame.from_dict(flow_count, orient='index').reset_index()
                flow_df.columns = ['index','flo'+ str(counter)]
                relations7, value7 =multidict(flow_count)
                relations7 = [list(elem) for elem in relations7]
                Flow = pd.DataFrame(relations7, columns=['m', 'i', 'p', 'j', 'time'])
                Flow['flo'+ str(counter)]=flow_df['flo'+ str(counter)]
                Flow_df=Flow
            else:
                flow_df = pd.DataFrame.from_dict(flow_count, orient='index').reset_index()
                flow_df.columns = ['index','flo'+ str(counter)]
                relations7, value7 =multidict(flow_count)
                relations7 = [list(elem) for elem in relations7]
                Flow = pd.DataFrame(relations7, columns=['m', 'i', 'p', 'j', 'time'])
                Flow['flo'+ str(counter)]=flow_df['flo'+ str(counter)]
                Flow_df=(pd.merge(Flow_df, Flow, on=['m', 'i', 'p', 'j', 'time']))

            # Save to spreadsheet
            sheets = {'X':Flow_df,
                      'Y':Fin_df,
                      'V':Inv_df,
                      'Z':MetOrder_df,
                      'D':DelOrder_df}
            writer = pd.ExcelWriter('final_output.xlsx', engine='xlsxwriter')
            for name in sheets.keys():
                sheets[name].to_excel(writer, sheet_name=name, index=False)
            writer.save()

        return

    # ---------------------------------------------------------------------

    MonteCarlo(Simulation_Runs,'Risk.csv')

    end = tt.time()
```

```
    a=end-start
    print ('Computational time:',a)

################################################################################

# Run the schedule risk assessment programm
ScheduleRisk(50,10)
```

APPENDIX B: Visualization of results in R

```
# LIBRARIES
library(readxl)
library(tidyverse)
library(ggplot2)


# INPUT
file_N = "/Users/chrismbourgeois/Documents/GitHub/thesis/work_center.csv"
file_M = "/Users/chrismbourgeois/Documents/GitHub/thesis/material.csv"
file_results <- "A-I.xlsx"


# READ DATA FILES
N_raw <- read_csv(file_N, col_names=FALSE)
M_raw <- read_csv(file_M, col_names=FALSE)
X_raw <- read_excel(file_results, sheet="X")
Y_raw <- read_excel(file_results, sheet="Y")
V_raw <- read_excel(file_results, sheet="V")
Z_raw <- read_excel(file_results, sheet="Z")
D_raw <- read_excel(file_results, sheet="D")


# ORDER FULFILLMENT
Z_raw[Z_raw$material=="P1-A-60220_F",] %>%
  group_by(order,time) %>%
  summarise(count = sum(met1,met2,met3,met4,met5,
                        met6,met7,met8,met9,met10)) %>%
  ggplot(aes(time,count)) + theme_bw() +
  geom_bar(aes(fill=order), position="dodge", stat="identity") +
  xlim(13,47) + ylim(0,10) +
  xlab("Time") + ylab("Count") +
  scale_fill_discrete(name="Order", labels=c("1","2","3","4"))


# WORK CENTER CAPACITY
N_cap_all <- X_raw %>%
  group_by(i,time) %>%
  summarise(run1 = sum(flo1),
            run2 = sum(flo2),
            run3 = sum(flo3),
            run4 = sum(flo4),
            run5 = sum(flo5),
            run6 = sum(flo6),
            run7 = sum(flo7),
            run8 = sum(flo8),
            run9 = sum(flo9),
            run10 = sum(flo10),
            max = max(run1,run2,run3,run4,run5,
                      run6,run7,run8,run9,run10)) %>%
  select(-run1,-run2,-run3,-run4,-run5,
         -run6,-run7,-run8,-run9,-run10)
N_cap_all <- N_cap_all %>%
  inner_join(N_raw[1:2], by=c("i"="X1"))

# Suppliers
N_cap_all[N_cap_all$X2=="S",] %>%
  ggplot() + theme_bw() +
  geom_line(aes(x=time, y=jitter(max/45), color=i)) +
  xlab("Time") + ylab("Utilization") +
  scale_color_discrete(name="Suppliers",
                       labels=c("1","2","3","4","5",
                                "6","7","8","9","10",
                                "11","12","13","14","15",
                                "16","17","18","19")) +
  guides(color=guide_legend(ncol=2))

# Operations
N_cap_all[N_cap_all$X2=="O",] %>%
  ggplot() + theme_bw() +
  geom_line(aes(x=time, y=jitter(max/30), color=i)) +
```

```
  xlab("Time") + ylab("Utilization") +
  scale_color_discrete(name="Operations",
                       labels=c("1","2","3","4","5","6","7","8","9"))

# Inspections
N_cap_all[N_cap_all$X2=="I",] %>%
  ggplot() + theme_bw() +
  geom_line(aes(x=time, y=jitter(max/30), color=i)) +
  xlab("Time") + ylab("Utilization") +
  scale_color_discrete(name="Inspections",
                       labels=c("1","2","3","4","5"))


# PRODUCTION RATE
Y_cum <- Y_raw %>%
  pivot_longer(
    cols = starts_with("fin"),
    names_to = "run",
    values_to = "fin")
Y_cum$cum <- 0

mat = "P1-A-60220_F"
Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin1",]$cum <-
  cumsum(Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin1",]$fin)
Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin2",]$cum <-
  cumsum(Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin2",]$fin)
Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin3",]$cum <-
  cumsum(Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin3",]$fin)
Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin4",]$cum <-
  cumsum(Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin4",]$fin)
Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin5",]$cum <-
  cumsum(Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin5",]$fin)
Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin6",]$cum <-
  cumsum(Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin6",]$fin)
Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin7",]$cum <-
  cumsum(Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin7",]$fin)
Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin8",]$cum <-
  cumsum(Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin8",]$fin)
Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin9",]$cum <-
  cumsum(Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin9",]$fin)
Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin10",]$cum <-
  cumsum(Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin10",]$fin)

mat = "P2-A-60220_F"
Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin1",]$cum <-
  cumsum(Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin1",]$fin)
Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin2",]$cum <-
  cumsum(Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin2",]$fin)
Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin3",]$cum <-
  cumsum(Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin3",]$fin)
Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin4",]$cum <-
  cumsum(Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin4",]$fin)
Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin5",]$cum <-
  cumsum(Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin5",]$fin)
Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin6",]$cum <-
  cumsum(Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin6",]$fin)
Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin7",]$cum <-
  cumsum(Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin7",]$fin)
Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin8",]$cum <-
  cumsum(Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin8",]$fin)
Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin9",]$cum <-
  cumsum(Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin9",]$fin)
Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin10",]$cum <-
  cumsum(Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin10",]$fin)

mat = "P3-A-60220_F"
Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin1",]$cum <-
  cumsum(Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin1",]$fin)
Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin2",]$cum <-
  cumsum(Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin2",]$fin)
Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin3",]$cum <-
  cumsum(Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin3",]$fin)
Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin4",]$cum <-
```

```
  cumsum(Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin4",]$fin)
Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin5",]$cum <-
  cumsum(Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin5",]$fin)
Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin6",]$cum <-
  cumsum(Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin6",]$fin)
Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin7",]$cum <-
  cumsum(Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin7",]$fin)
Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin8",]$cum <-
  cumsum(Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin8",]$fin)
Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin9",]$cum <-
  cumsum(Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin9",]$fin)
Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin10",]$cum <-
  cumsum(Y_cum[Y_cum[,2]==mat & Y_cum[,4]=="fin10",]$fin)

Y_cum[Y_cum[,2]=="P1-A-60220_F",]$material <- "Product 1"
Y_cum[Y_cum[,2]=="P2-A-60220_F",]$material <- "Product 2"
Y_cum[Y_cum[,2]=="P3-A-60220_F",]$material <- "Product 3"

Y_cum %>%
  ggplot(aes(x=time, y=cum, color=run)) + geom_line() +
  facet_wrap(vars(material), nrow=3, ncol=1) +
  theme_bw() + xlab("Time") + ylab("Production") +
  scale_color_discrete(name="Iteration",
                       labels=c("1","2","3","4","5",
                                "6","7","8","9","10"))


# MATERIAL INVENTORY
V_inv <- V_raw %>%
  pivot_longer(
    cols = starts_with("inv"),
    names_to = "run",
    values_to = "inv") %>%
  inner_join(M_raw[1:2],
             by=c("material"="X1"))

V_inv[V_inv$X2 == "S",]$X2 <- "Raw"
V_inv[V_inv$X2 == "O",]$X2 <- "Processed"
V_inv[V_inv$X2 == "I",]$X2 <- "Inspected"
V_inv[V_inv$X2 == "D",]$X2 <- "Final"

V_inv$X2 <- factor(V_inv$X2,
                   levels=c("V","Raw","Processed",
                            "Inspected","Final"),
                   ordered=TRUE)

V_inv[V_inv$inv != 0 & V_inv$X2 != "V",] %>% ggplot() +
  geom_point(aes(x=time, y=inv),
             alpha=0.05, size=2) +
  theme_bw() + xlab("Time") + ylab("Inventory") +
  facet_wrap(vars(X2), nrow=4, ncol=1)
```