A digital twin framework for online optimization of supply chain business processes

Hector D. Perez[a], John M. Wassick[b], and Ignacio E. Grossmann[a]*

[a]*Carnegie Mellon University, Pittsburgh 15213, USA*
[b]*The Dow Chemical Company, Midland 48674, USA*
**grossmann@cmu.edu*

Abstract

An integrated framework for building a virtual replica of business transactional processes in supply chains is presented. The framework consists of three main components: 1) a systems identification module that uses a unifying abstraction layer to generate process models, 2) a simulation module to create and run discrete-event simulations, and 3) an optimization module to optimize the supply chain business process operations. These business processes are conceptually modeled as networks of queues where agents perform tasks on the orders (internal or external) that are flowing through the process. The modeling approach allows capturing the routing dynamics and stochasticity in both the task durations and order arrivals that are observed in practice. The digital replica of the business processes creates value by providing a flexible digital environment that can be used to evaluate operating and design policies, identify and mitigate bottlenecks, quote more accurate lead times to customers, and forecast the impact of system disturbances and their remediation. The models in the digital twin are generated with data from the historian database and are updated in real-time using the live process data. The framework allows deploying optimization models offline, or in simulated real-time in a feed-back loop in a stochastic simulation environment. As an integrated simulation and optimization environment, the framework bridges and extends the literature in business process simulation, business process optimization, and supply chain management. Examples are presented to illustrate how the proposed digital twin can generate value in a digital supply chain.

Keywords: Business process optimization, digital supply chain, order-to-cash, digital twin

1. Introduction

Supply chains are complex dynamical systems that integrate three main types of flows: material, financial, and informational. These flows are tightly interconnected within the different businesses in an enterprise. The recent widespread push towards establishing digital supply chains has made the relationship between these flows even more intimate and visible (Büyüközkan and Göçer, 2018). The digitalization of supply chains has become a key enabler to effectively manage these flows, providing significant value to enterprises that seek to reduce costs, increase customer satisfaction, and streamline operations. A report by the consulting firm *PwC Strategy&* states that digital supply chains can expect annual improvements of 4% in efficiency and 3% in revenue when compared to traditional supply chains (Schrauf and Berttram, 2016). While these percentages may not seem particularly high, their impact is significant when considering the volume of materials, services, and revenue that flow across a supply chain. A report by McKinsey & Company predicts significant reductions in lost sales (65-75%), logistics costs (15-30%), administrative costs (50-80%), inventories (35-75%), and forecasting error (30-50%) as a result of Supply Chain 4.0 (Alicke *et al.*, 2016). Clearly, efficient supply chain management, enabled by digitalization, is key

to supply chain growth and sustainability. It is in this effort towards Supply Chain 4.0 or Digital Supply Chain, that this work proposes a digital twin framework for supply chain business processes.

We first define the concept of supply chain business processes. A business process is the collection of connected events, actions, and decisions that add value to an enterprise and its customers (Dumas *et al.*, 2013). These processes involve agents (e.g., human actors, organizations, and automation software), physical objects (e.g., goods and documents), and intangible objects (digital footprint). Some examples of business processes common to supply chains are the order-to-cash, procure-to-pay, and issue-to-resolution. Effective management of these processes is essential to efficient operation and customer satisfaction. There is extensive literature in the area of business process management (BPM), to which the reader is referred to standard books and articles on the topic (van der Aalst, 2013; van der Aalst *et al.*, 2003; vom Brocke and Rosemann, 2015; Dumas *et al.*, 2013).

The next concept to define is that of a digital twin. A digital twin refers to a virtual environment that mimics physical objects or systems in real-time (Negri *et al.*, 2017; Stanford-Clark *et al.*, 2019). Digital twins can take different forms, from augmented reality powered by internet of things (IoT) in manufacturing (Zhu *et al.*, 2019) to simulation-based and data-driven decision support systems (DSS) in healthcare (Trotabas, 2019). Recent years have seen a big push toward the development and deployment of digital twins in many different industries. According to a recent study, the market for digital twins is expected to grow from $3 million to $48 million from 2020 to 2026 (58% annual growth) (Markets and Markets, 2020).

Despite the increased need for digital twins, existing literature on supply chain digital twins is scarce due to it being a relatively new concept that has been emphasized more in industry than in academia. Existing literature is for the most part conceptual and focused on the material flows and physical processes in supply chains. Ivanov and Dolgui (2019) present the concept of a digital twin for supply chain in the context of risk management. They discuss a new generation of tools that integrate various technologies to provide decision analysis, modeling, control, and learning systems (DAMCLS) for supply chains. At the core, these DAMCLS are digital twins that replicate the physical supply chains continuously in real-time in terms of inventory, production, logistics, and demand. These digital twins bring together simulation, optimization, data analytics, and artificial intelligence to identify disruptions, simulate their impact, optimize recovery policies, design resilient systems, and perform real-time performance control. Barykin, et al. (2020) describe software tools that can be used to build supply chain digital twins with various degrees of complexity where optimization is used to solve the supply chain design problem, and simulation is used to model the system dynamics. There are few examples in the literature of actual digital twins developed for supply chains. One of these is the modular logistics digital twin by Lee and Lee (2021), which combines building information modeling (BIM), internet of things (IoT), geographic information systems (GIS), to solve a vehicle routing problem (VRP) in the modular construction supply chain.

The key enablers of the proposed digital twin are discrete event simulation (DES) and mathematical programming (MP). DES has been extensively applied to queueing systems, manufacturing systems, and inventory systems (Goldsman *et al.*, 2015). In the context of business processes, Hlupic and De Vreede (2005) propose the use of DES to evaluate business process design alternatives and thus increase the success rate of business process re-engineering. van der Aalst (2010) discusses the difficulties in adopting business process simulation in practice and proposes the use of process mining in simulation. Wagner et

al. (2009) extend the traditional DES framework to include concepts from Business Process Management Notation (BPMN). Their proposed extensions include the use of activity constructs, agent-based activities, and logical gateways for event flow. They apply their framework to model the business process associated with a first-in-first-out (FIFO) service queue.

On the other hand, mathematical programming (MP) is one of the key drivers in supply chain management (SCM) (Laínez and Puigjaner, 2012; Shah, 2005), supply chain sustainability (Mota *et al.*, 2015), and enterprise-wide optimization (EWO) (Grossmann, 2012). Applications of MP in supply chain range from network design problems (Lara *et al.*, 2019) to scheduling (Guillén *et al.*, 2006) and vehicle routing problems (Subramanyam *et al.*, 2020). In terms of physical process operational scheduling, the process systems engineering (PSE) community has pioneered several mathematical modeling paradigms with applications in many industries (Harjunkoski *et al.*, 2014; Méndez *et al.*, 2006). Various methods have also been developed to account for uncertainty in scheduling. These include robust optimization (Lappas and Gounaris, 2016), stochastic programming (Balasubramanian and Grossmann, 2004a), and online or closed-loop scheduling (Gupta *et al.*, 2016; McAllister *et al.*, 2022). Although not applied in the context of supply chain processes, the computer science and information systems community has produced a handful of works in the area of workflow scheduling for cloud-based systems. These transactional scheduling approaches rely on both heuristics and mathematical programming (Cai *et al.*, 2016; Hoenisch *et al.*, 2016; Li *et al.*, 2018), but are limited to static deterministic systems.

In previous work, we have addressed the potential in applying scheduling models from the chemical process industry to improve the operations of supply chain business processes (Perez *et al.*, 2021a, 2021b). However, this work was limited to perfect information systems and ignored the stochastic nature of the transactional flows and future customer demands in the supply chain. In line with the growing need for digital twins, we extend our previous work to develop a framework for building and deploying virtual replicas of supply chain business processes. These twins model the system dynamics of the transactional processes, opening a varied range of tools to improve supply chain performance. These include, but are not limited to, performing Monte Carlo simulations for design evaluation, policy evaluation, policy improvement, disturbance management, fault detection, accurate order fulfillment quoting, and system forecasting. The digital twin further generates value by allowing for real-time and online (closed loop) business process optimization, where optimization events are triggered periodically or in response to system disturbances. The proposed framework thus brings together contributions from digital supply chain, discrete event simulation, business process management notation, queueing systems, and online optimization to extend the vision discussed by Ivanov and Dolgui in the context of physical supply chain processes to the information flows and transactional processes of the supply chain. Although the proposed framework is general for any supply chain business process, it is presented in the context of the order-to-cash (order fulfillment) process.

The paper is structured as follows. Section 2 describes the order-to-cash process being modelled with the digital twin. Section 3 details the digital twin framework. Section 3.3 illustrates the benefits of the digital twin with five different examples. Section 5 provides concluding remarks.

2. Problem Description

The order-to-cash (OTC) process is the business process that accounts for the steps that a customer order goes through, starting at the order creation and ending when the payment (cash) is received for the fulfilled order (Dumas *et al.*, 2013). In this process, a company has one or more customers $c \in C$, exhibiting the following characteristics,

- Customer segmentation: priority tier assigned to that customer by the company.
- Customer patience: the customer's willingness or likelihood of waiting for an order after its due date without cancelling it.

The different customers place orders $o \in O$, which belong to different order classes $i \in I$ and arrive at the company with stochastic interarrival times that are sampled from a given statistical distribution. Each order o has the following parameters,

- Release date: order arrival time, T_o^r.
- Early due date: earliest time the order will be accepted by the customer, T_o^e.
- Due date: latest time the order will be accepted by the customer and still be considered on-time, T_o^d.
- Lost sales date: time after which the order will no longer be accepted by the customer, T_o^{ls}, meaning the customer's waiting patience has been exceeded.
- Early fulfillment incentive: the profit bonus that the company receives when the order is fulfilled early. This can be thought of as a premium the customer pays to expedite the order fulfillment.
- Due date profit: the profit received when fulfilling the order on the due date.
- Fixed late delivery penalty: a fixed decrease in profit incurred by the company for a late delivery.
- Variable late delivery penalty: a variable (i.e., daily) penalty incurred after the due date has passed.
- Lost sale penalty: a fixed penalty incurred for not fulfilling the customer order. This penalty will cause the profit to drop to zero or to a negative value, indicating a loss of goodwill by the company.

For simplicity, order profit is modeled via piecewise linear functions as the one shown in **Figure 1**. However, the profit functions can be modelled with nonlinear expressions if required. With this modeling form, a delivery window between the early delivery date, T^e, and the due date, T^d, can be enforced with an incentive for early deliveries (if the early price, p^e, is greater than the price at the due date, p^d). Late orders can be penalized with a fixed penalty at T^d, such that the order profit drops to p^l, and a variable penalty, such that the order profit decreases linearly to p^f at T^{ls}, after which the order becomes a lost sale with a non-positive profit, p^{ls}. This approach models profit in a flexible way that allows the use of delivery windows, incentives, fixed penalties, and variable costs, and captures both order backlogging and lost sales, which are typically not considered together in supply chain models.

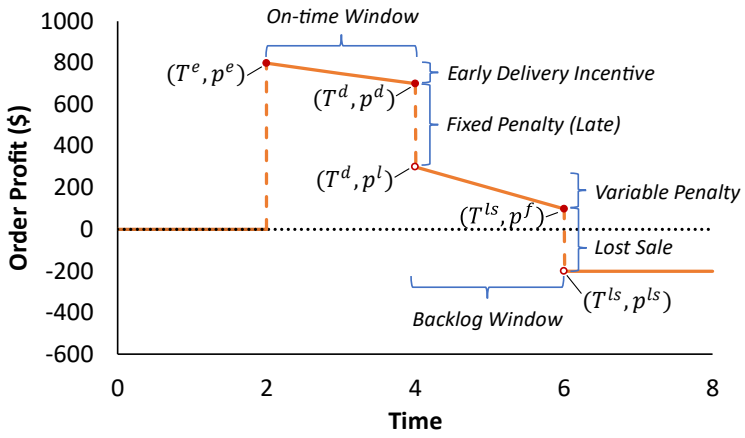

Figure 1. Piecewise linear function used to model order profit

A high-level view of the OTC process is given in **Figure 2**. As customer orders flow through the process, they undergo a series of transactions $l \in L$ (processing stages) that are executed by agents $a \in A$ (resources), which can be either human agents or automated processing steps. The duration (processing time) of each transaction is modeled as a random variable $\tau \sim G(\xi)$ that is sampled from a statistical distribution $G(\cdot)$ (e.g., Normal distribution) with known distribution parameters ξ (e.g., mean and variance). In practice, the task duration distributions are generated from and updated with real process data. Such distributions can be aggregated for the transaction or modelled specific to the order type (class), customer segment, and resource (agent) involved. Agents can be completely flexible such that they can perform any transaction on the orders in the system, or completely dedicated such that they can only perform a specific transaction on orders coming from specific customers. However, various levels of agent flexibility can exist between these two extremes. Once a transaction is completed on an order, the order will move on to the next step in the OTC process network.

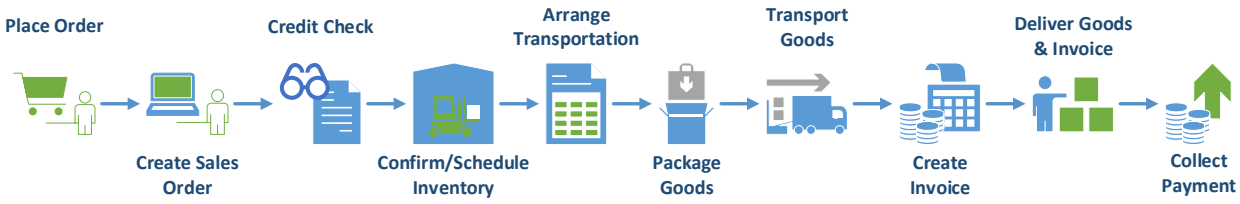

Figure 2. Overview of main processing steps (transactions) in the OTC process. Reproduced with permission Perez, H.D., Amaran, S., Erisen, E., Wassick, J.M., Grossmann, I.E., 2021. Optimization of extended business processes in digital supply chains using mathematical programming. Computers and Chemical Engineering 152, 107323. Copyright (2021) Elsevier, Ltd.

Although the OTC process is typically described as a serial (linear) network of steps, this is often not the case in practice. Real implementations of the OTC process can have various forms of complexity, such as parallel transactions, alternate paths, and recycle loops. Parallel transactions occur when the completion of a transaction triggers more than one transaction, which can be performed independently of each other. Alternate paths and recycle loops occur when the state of an order warrants an exception or deviation from the “standard” sequence of events, such as when an inconsistency must be corrected, or an order parameter modified.

From a process systems perspective, the OTC business process can be thought of as a flexible multi-stage batch plant, where customer orders are analogous to batches in the system. An alternative approach, which is the one used by the proposed digital twin is to model the OTC process as a network of queues, as shown in **Figure 3**. From this perspective each agent has its queue, where orders line up for transactions to be executed on them. Dedicated agents have single-class queues, whereas flexible agents have multi-class queues, meaning that they perform different types of tasks (classes) on the orders in their queue. Two types of queueing phenomena are observed: preemption and reneging. Preemption occurs when an agent is actively processing an order and a higher priority is assigned to an existing order in the queue or a higher priority order arrives at the queue. This can occur when a high priority customer places a rush order, or when a change in the system state warrants increasing the priority on an existing order. When this occurs, the service on the active order is interrupted so that the higher priority order takes precedence at that step. A major assumption of the queueing model used is that the amount of work done already on the interrupted order is stored so that when the order returns to service, its work can be resumed from where it left off. If needed, this assumption can be relaxed by adjusting the way that the task durations are modelled after a preemption event. Reneging occurs when an order has exceeded its lost sales date and the customer decides to cancel the order.

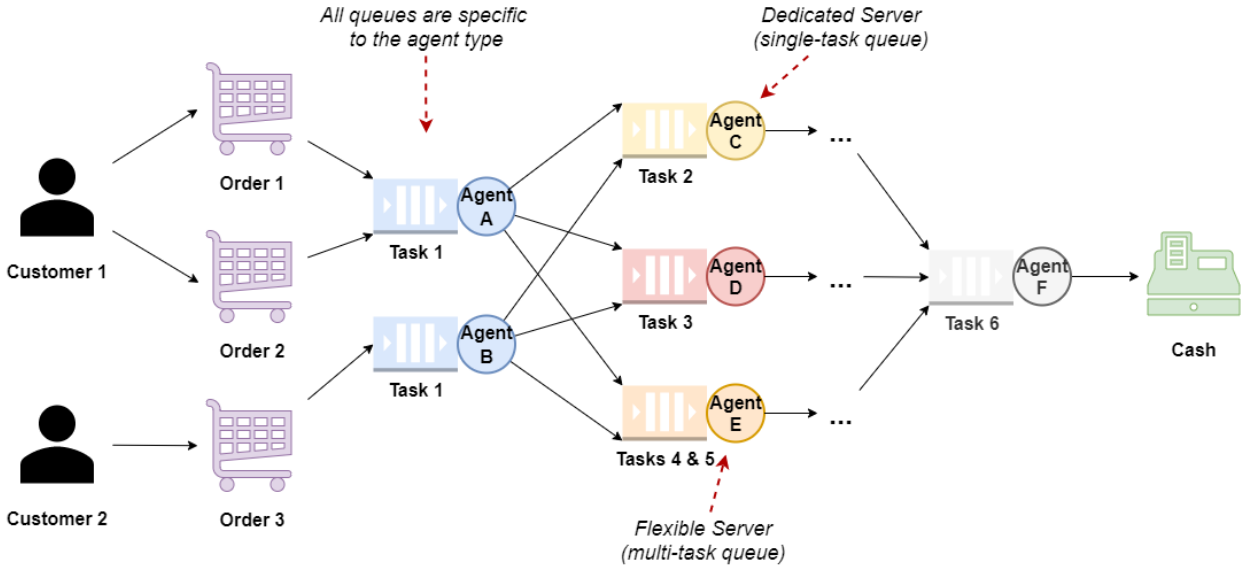

Figure 3. Sample queueing network representation of the supply chain order-to-cash business process

3. Digital Twin Framework

Now that the general characteristics of the OTC process being modelled have been described in **Section 2**, the following section takes a step back to introduce the concept of a digital twin for a supply chain business process such as the OTC process. This section describes both what can be accomplished with an OTC digital twin, as well as detailing the features and internals of the proposed twin. A conceptual view of this framework is depicted in **Figure 4**. The proposed framework relies on three main modules,

1. System identification module: used to model the supply chain business process as a graph with embedded metadata for the system parameters (see Section 3.2).

2. Simulation Engine: used to generate discrete event simulation models to simulate the system dynamics via Monte Carlo simulation (see Section 3.3). The simulation engine can be run independently of the optimization engine or in an integrated mode (see Section 3.5).
3. Optimization Engine: used to execute mixed-integer linear programming (MILP) models or heuristics to schedule requests (e.g., customer orders) in the business process (see Section 3.4). The optimization can be run offline or integrated with the simulation engine for online scheduling (see Section 3.5).

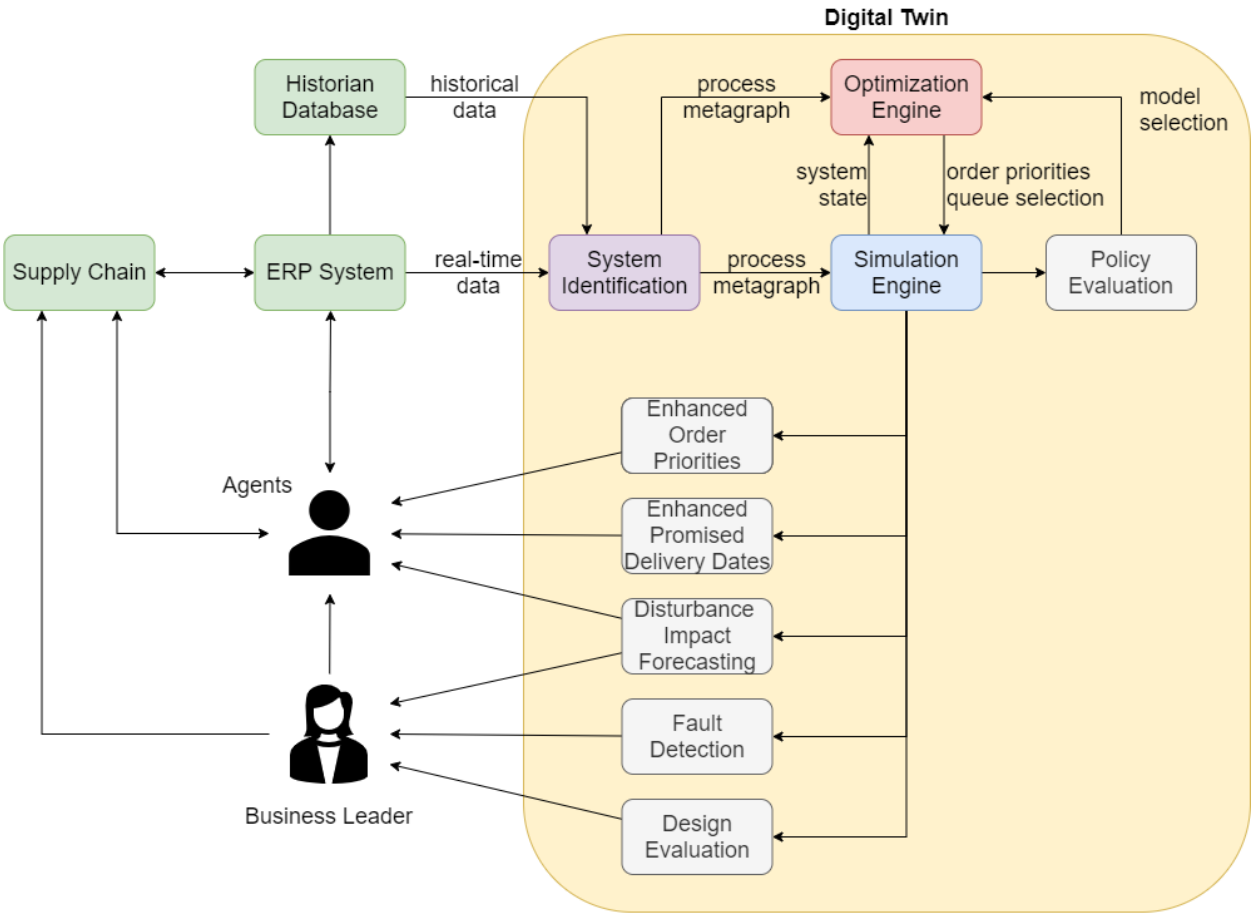

Figure 4. Supply chain business process digital twin framework

The following subsections give a high-level view of how the digital twin is used to generate value (Section 3.1), followed by a detailed description of the internals in each of the digital twin key components (system identification, simulation engine, and optimization engine).

3.1 Framework Overview

The digital twin uses both real-time data from the enterprise resource planning (ERP) systems and historical data from the historian database to create a model of the business process that mimics the behavior of the real system. This model is created by first creating a network abstraction model of the system, which is then used to generate the dynamic simulation model and the optimization methods in the simulation and optimization engines, respectively. These two engines are at the core of the digital

twin and enable various outcomes that are helpful to both the agents involved in executing the business process transactions, as well as the managers that oversee the business process. Some examples of what can be accomplished with the digital twin are,

1. Enhanced order priorities: agents can use the digital twin to get a ranking of the orders that are in their process queue. This queue prioritization can be done via heuristics or mathematical programming scheduling models in the optimization engine. In the latter case, the prioritization is based, not just on the orders in the specific agent's queue, but on the entire state of the system. Although not in the scope of the proposed framework, business rules commonly used in practice can be represented as heuristics or used to enhance existing heuristics.
2. Enhanced promised order delivery dates: since the simulation engine can mimic the current state of the orders in the supply chain, Monte Carlo simulation can be used to get an estimate of when an incoming customer order can be fulfilled realistically. This goes beyond using mean lead times, but actually using the digital replica of the system to get a fulfillment date with an appropriate statistical confidence level. This type of information is particularly relevant to the customer service representative so as to avoid quoting unrealistic fulfillment dates to their customers.
3. Forecasting the impact of a disturbance: when a disturbance occurs in the system, the digital twin can be used to run Monte Carlo simulations to estimate the impact of that disturbance on the system (e.g., on when the orders will be fulfilled and the load on the system queues). This information can be valuable to both agents and business leaders when evaluating proper mitigation strategies. The simulation engine itself can be used as an environment to evaluate alternative responses to supply chain disturbances.
4. Fault detection: bottlenecks can be identified in the supply chain by performing data analysis on the real-time and simulated data. By identifying bottlenecks and other system inefficiencies, design and operational modifications can be proposed to alleviate these stresses on the system.
5. Design evaluation: modifications to the supply chain business processes include changes to the business process structure and resource availability and configuration. These changes can be tested with the simulation engine with both Monte Carlo simulations and backtesting on historical data. An important design evaluation that can be addressed is where automation, such as robotic process automation (RPA) (van der Aalst *et al.*, 2018), can be applied for the greatest benefit.
6. Operational policy and business rule evaluation: optimization models and heuristics in the optimization engine can be fine-tuned and ranked to select the more appropriate method to assign order priorities and resources throughout the transactional network. Fixed business rules can also be evaluated. Since the digital twin is constantly updated with real-time data, current operating policies can be re-evaluated against the portfolio of policies to dynamically and adaptively select the best-suited policies.

The core elements of the proposed digital twin have been developed using the Julia programming language (Bezanson *et al.*, 2017) version 1.7, which allows leveraging existing high-performance libraries available within the Julia environment for network modeling, discrete event simulation, and mathematical programming. As an integrated platform, the framework reduces the need for creating interfaces between different programming languages, GUIs, and databases required to build and operate the digital twin. The digital twin framework is designed to model supply chain business processes of any complexity.

3.2 System Identification Module

To model the supply chain business process, a process graph with embedded metadata (metagraph) is used as an abstraction layer to capture the system structure, parameters, and dynamics. The *Graphs.jl* and *Metagraphs.jl* (Fairbanks *et al.*, 2021) open-source Julia language packages (versions 1.4 and 0.7, respectively) are used to efficiently create and modify the process metagraph. The value in having this abstraction layer lies in the way the metagraph is seamlessly used to build the simulation and optimization models in the core digital twin engines. As a unified system, any changes to the metagraph are automatically reflected in the simulation and optimization models. This eliminates the need to update multiple system models whenever a change is made or is being evaluated.

Figure 5 provides a sample of a business process metagraph, which consists of a task network that indicates the transactions that customer orders will flow through. The agents associated with each transaction, as well as the task duration probability distributions are embedded in the task node metadata. A single aggregated duration distribution can be stored for the transaction, or an array of distributions can be stored to model the durations for each agent, order type, and customer segment combination. Each distribution is either fit or bootstrapped from the historical database and is periodically updated with real-time data. Routing dynamics are modeled via logical gateway nodes (split AND, merge AND, split XOR, merge XOR, start recycle, and end recycle). Split/merge AND node pairs indicate parallel routes that orders will take. Split/merge XOR node pairs indicate alternate routes that can be taken by orders. Split XOR nodes represent decision steps in the network. The decision step is modeled via the statistical probabilities or likelihood of taking one of the paths. The decision outcome is obtained by sampling from a weighted roulette wheel (selector), where the weights for each slot are transition probabilities obtained from historical data for that step or specific to the order type and/or customer type. The selector probabilities are also updated with real-time data to improve the prediction accuracy. Start/end recycle node pairs are used as markers to identify loops in the network.

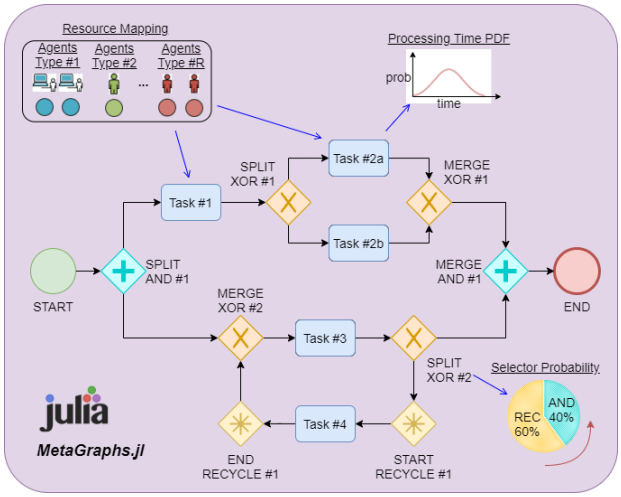

Figure 5. Sample business process metagraph

The example depicted in **Figure 5** can be used to illustrate the order flow in the process metagraph. In this sample system, customer orders are first sent to the Task #1 and Merge XOR #2 nodes. The Merge XOR #2 node checks that at least one of its predecessors has completed. Since this is a new order, the order

arrival process (Start node) has been completed, so the order moves on to Task #3. When Task #3 completes, the weighted roulette wheel in the Split XOR #2 node is spun with a 60% probability that the order will be recycled back to Task #3 and a 40% probability that the order will move on to the Merge AND #1 node. When Task #1 completes, it is sent to the Split XOR #1 node where another roulette wheel is spun to determine if the order will be routed to Task #2a or Task #2b. Once one of these has completed, the Merge XOR #1 node will allow the order to continue to the Merge AND #1 node. Once the order has gone through both parallel branches in the network, the Merge AND #1 node will enable the order to move to the End node, indicating the order has been fulfilled.

3.3 Simulation Engine

The process metagraph has all the information required to build the queueing network representation for the discrete-event simulation in the simulation engine. **Figure 6** shows the resulting queueing network obtained from the metagraph in **Figure 5** when dedicated agents are used. The simulation engine is powered by the *SimJulia.jl* (Andriessen and Lauwens, 2017; Lauwens, 2017) open-source Julia language package (version 0.8.2). The resulting agent-based business process discrete event simulation captures the network flow logic from the task network in the metagraph and has similar features to those described by Wagner, *et al.* (2009). In the simulation engine, the agents involved in the business process are modelled as servers with independent queues that receive and process customer orders. Any modifications to the metagraph are automatically transferred to the agent-based network simulation engine, which avoids having to modify two separate models.

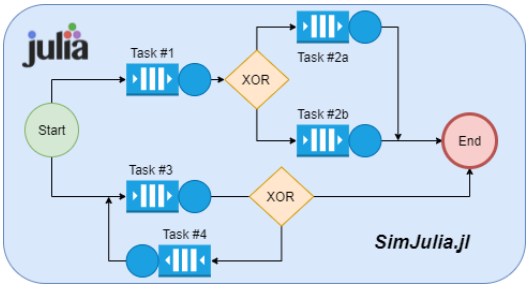

Figure 6. Sample mapping of the metagraph in **Figure 5** to a queueing network with dedicated agent queues in the simulation engine

3.4 Optimization Engine

From the process metagraph, mixed-integer linear programming (MILP) scheduling models or heuristic approaches can be built in the optimization engine. The mathematical programming functionality in the optimization engine is powered by the *JuMP.jl* (Dunning *et al.*, 2017) open-source Julia language package (version 0.21.10).

3.4.1 Mathematical Programming

The discrete-time State-task Network (STN) (Kondili *et al.*, 1993; Shah *et al.*, 1993) is used for the optimization engine within the framework. Although alternate modeling paradigms are available (e.g., Resource-task-network [RTN] and General Precedence models), this work focuses on the STN because previous results have shown that this modeling paradigm tends to exhibit reduced solution times for

larger instances of the order-to-cash process (Perez *et al.*, 2021b). As a discrete-time model, the STN model benefits from the tighter linear programming relaxations observed in discrete-time scheduling models (Harjunkoski *et al.*, 2014).

A sample graphical representation of a bipartite state-task network for the order-to-cash (OTC) process is given in **Figure 7**. The states in the OTC STN model are binary states that indicate if an order is waiting at a particular buffer in the queueing network (1 = order is present, 0 = order is not present). The exception is the sink state (S^{sink}), which can be integer-valued if there is more than one sink task (L^{sink}). In this case, the value of the order sink state indicates the number of sink tasks that have completed up to that point prior to the order being fulfilled. Thus, each state node has no more than one successor task and no more than one predecessor task, unless it is a sink state with more than one sink task. In this STN model, each task consumes one state from each of its upstream order states when the task is triggered, and generate one downstream order state when the task completes.

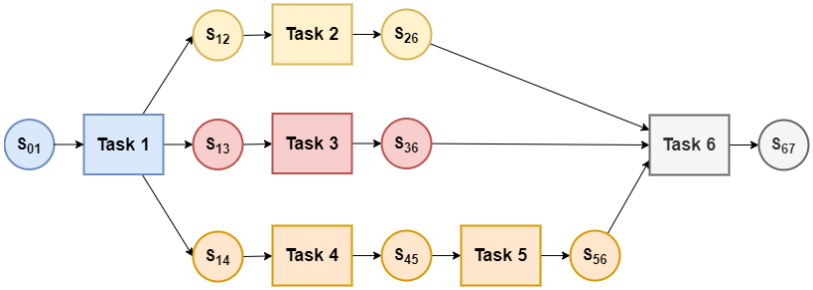

Figure 7. Sample state-task network for the system shown in **Figure 3**. Circular nodes are order states and rectangular nodes are tasks.

As a discrete-time model, a fixed time grid with uniform slots of size ΔT is used to model time. Since temporal discretization introduces round-off errors, these are handled in a conservative fashion to avoid infeasibilities in the real system. The mean (expected) task durations $\overline{\tau}_{i,l,a} = \mathrm{E}[\tau_{i,l,a}]$ for each order class i at task l processed by agent a are mapped to mean durations $\overline{\tau}_{o,l,a}$ for order o using the class of each order. These durations are then rounded up to the nearest multiple of ΔT $\left(\tau'_{o,l,a} = \lceil \overline{\tau}_{o,l,a}/\Delta T \rceil\right)$. The scheduling horizon is taken to be the latest lost sales date ($H = \max_{o \in O} T_o^{ls}$) and is rounded up to the nearest multiple of ΔT ($\underline{h} = \lceil H/\Delta T \rceil + 1$) to ensure that the latest lost sales date is included. Using the latest lost sales date as the horizon results in an adaptive rolling horizon approach. This approach avoids the end-of-horizon effects observed when a fixed or shrinking horizon is used (Lima *et al.*, 2011), in which case the model assumes that no new orders can be fulfilled beyond the horizon. If a fixed or shrinking horizon approach is desired, the floor operator (round down) is used on the discretized horizon to guarantee that the horizon is not exceeded. The release, early, due, and lost sales dates for each order are rounded up and down to ensure feasibility: $\overline{t_o^r} = \lceil T_o^r/\Delta T \rceil + 1$, $\overline{t_o^e} = \lceil T_o^e/\Delta T \rceil + 1$, $\underline{t_o^d} = \lfloor T_o^d/\Delta T \rfloor + 1$, and $\underline{t_o^{ls}} = \lfloor T_o^{ls}/\Delta T \rfloor + 1$, where $\lceil x \rceil$ is the ceiling operator and $\lfloor x \rfloor$ is the floor operator. *Note*: t is a time index that indicates the time point (or time period), rather than the absolute time, with the $+1$ in each expressions indicating that the first time point ($t = 1$) is the beginning of the scheduling horizon (zeroth time, $T = 0$).

Assignment Constraints: The unit allocation constraint in Eq. (1) proposed by Shah *et al.* (1993) is used to ensure that each agent can process no more than one order at a time during the duration of a task, $[t - \tau'_{o,l,a} + 1,\ t]$. $W_{o,l,a,t}$ is a binary variable that denotes that task l is triggered at time point t on order

o by agent a. The $(x)^+$ notation in the inner summation is equivalent to the positive part of x, or $\max(x, 0)$, and is used to ensure that the summation is properly defined when $t < \tau'_{o,l,a}$. It should be noted that the STN representation in **Figure 7** must be duplicated for each order in the system, and Eq. (1) is the linking constraint that joins each of the customer order subnetworks.

$$\sum_{o\in O}\sum_{l\in L_a}\sum_{t'=(t-\tau'_{o,l,a})^+ +1}^{t} W_{o,l,a,t'} \leq 1 \qquad \forall a \in A, t \in TP \tag{1}$$

Order State Balance: A balance is performed for each order state at each time point. This balance is described by Eq. (2), in which the order availability at state s and time point t is equal to the initial condition, plus the order state value in the previous time point, minus any consumption occurring when its succeeding task (L_s^{succ}) is triggered, plus any generation occurring when a preceding task (L_s^{pred}) is completed, plus the arrival of a new order if it is a source state ($s \in S^{src}$), minus the delivery of an order if it is the sink state ($s \in S^{sink}$). The binary parameter $E_{o,t}$ indicates when a new order arrives, such that $E_{o,t} = 1$ when $t = \overline{t_o^r}$, and $E_{o,t} = 0$ at all other time points. $D_{o,t}$ is a binary variable that indicates if order o was fulfilled (delivered) at time point t. Since order states cannot be negative, the coefficient $\left|L^{sink}\right|$, ensures that the order can only be fulfilled after all sink tasks have completed.

$$\begin{aligned} OS_{o,s,t} = OS_{o,s,t}^{init} + OS_{o,s,t-1}\big|_{t>1} - \sum_{l\in L_s^{succ}}\sum_{a\in A_l} W_{o,l,a,t} + \sum_{l\in L_s^{pred}}\sum_{a\in A_l} W_{o,l,a,t-\tau'_{o,l,a}}\Big|_{t>\tau'_{o,l,a}} \\ + E_{o,t}\big|_{s\in S^{src}} - \left(\left|L^{sink}\right| \cdot D_{o,t}\right)\big|_{s\in S^{sink}} \\ \forall o \in O, s \in S, t \in TP \end{aligned} \tag{2}$$

The initialization parameter $OS_{o,s,t}^{init}$ is only relevant when optimizing an on-going order-to-cash process, as is the case when the optimization engine is run within the simulation engine. $OS_{o,s,t}^{init}$ must be defined for both queued orders and active (on-going) tasks. Queued order states are initialized using Eq. (3), where $Y_{o,l}^0$ is a binary parameter that is 1 if task l was already completed on order o by time point 0. Thus, the initialization will only have a positive value if at least one predecessor task $l \in L_s^{pred}$ has finished, but the successor task $l \in L_s^{succ}$ has not begun. For source states $s \in S^{src}$, L_s^{pred} represents the order arrival event. Active task state initialization is defined based on if preemption is allowed or not. When preemption is not allowed, Eq. (4) is used, where $Y_{o,l,\omega}^{np}$ is a binary parameter that is 1 if and only if task l began on order o at $t = \omega - \tau'_{o,l,a} < 0$ by agent a, and is expected to end at $t = \omega$. An additional constraint must be added in this case that fixes all task triggering variables for that agent to zero up to the period prior to when the task is expected to end ($W_{o,l,a,t} = 0 \;\; \forall o \in O, l \in L_a, t \in [1, \omega - 1]$). This ensures that the busy agent is not scheduled for a task while it is still busy. The initialization for an agent that does not support preemption is illustrated in **Figure 8**. When task preemption is allowed, Eq. (5) is used, where $Y_{o,s}^p$ is a binary parameter that is 1 if and only if the successor task $l \in L_s^{succ}$ is active at time 0. In the case of an active task that is preemptible, the mean processing time for order o at task $l \in L_s^{succ}$ is replaced by the expected remaining processing time given the current time-in-service ($\tau'_{o,l,a} = \left\lceil \left(\mathrm{E}\left[\tau_{o,l,a} | T \geq \delta\right] - \delta\right)/\Delta T \right\rceil \;\; \forall a \in A_l$, where δ is the time-in-service). All initialization calculations are done in a pre-processing step when building the scheduling model.

$$OS_{o,s,0}^{init} = \sum_{l \in L_s^{pred}} Y_{o,l}^{0} - \sum_{l \in L_s^{succ}} Y_{o,l}^{0} \qquad \forall o \in O, s \in S \tag{3}$$

$$OS_{o,s,\omega}^{init} = \sum_{l \in L_s^{pred}} Y_{o,l,\omega}^{np} \qquad \forall o \in O, s \in S \tag{4}$$

$$OS_{o,s,1}^{init} = Y_{o,s}^{p} \qquad \forall o \in O, s \in S \tag{5}$$

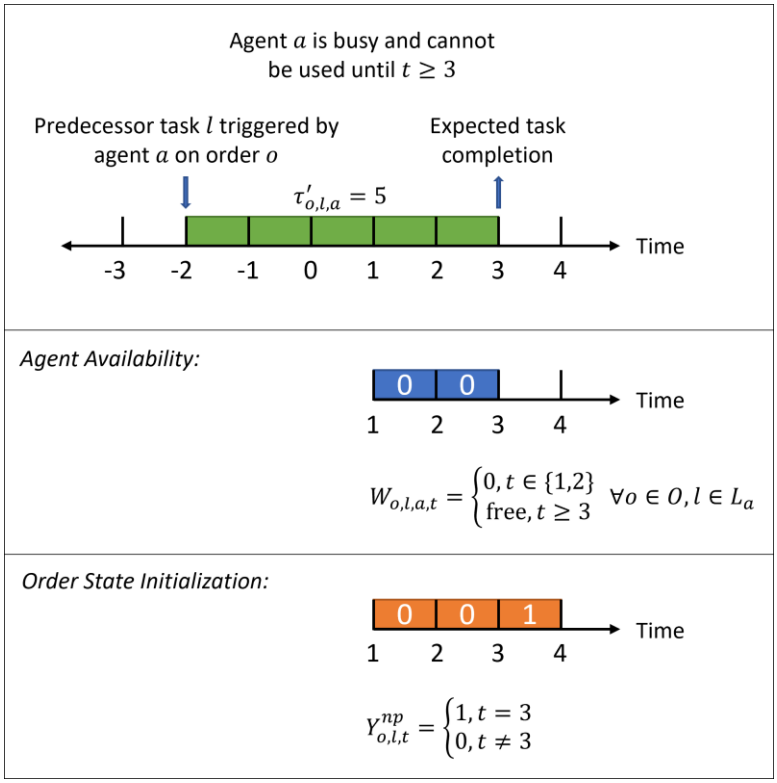

Figure 8. Sample initialization when an agent that does not support preemption begins a transaction at $t = -2$.

Order Fulfillment: The fulfillment of an order can be categorized as either on-time, backlogged, or lost sale. An order is fulfilled on-time when it is delivered within the allowed window ($[\overline{t_o^e}, \underline{t_o^d}]$) as given in Eq. (6). An order is backlogged when it is delivered after the due date $\underline{t_o^d}$ and by the final delivery date $\underline{t_o^{ls}}$ as given in Eq. (7). After this point, the order becomes a lost sale as given in Eq. (8).

$$OT_o = \sum_{t=\overline{t_o^e}}^{\underline{t_o^d}} D_{o,t} \qquad \forall o \in O \tag{6}$$

$$BL_o = \sum_{t=\underline{t_o^d}+1}^{\underline{t_o^{ls}}} D_{o,t} \qquad \forall o \in O \tag{7}$$

$$LS_o = 1 - OT_o - BL_o \qquad \forall o \in O \qquad (8)$$

Order Profit: The time that an order is fulfilled t_o^{max} is dictated by the delivery variable $D_{o,t}$ as given in Eq. (9). This fulfillment date is then used to determine which regime the order profit z_o belongs to, as visualized in **Figure 1** and enforced by the disjunction in (10). The disjunction is reformulated into mixed-integer constraints using the convex hull reformulation (Grossmann and Trespalacios, 2013), as shown in Eqs. (11)-(12). For simplicity, Boolean and binary variables are used interchangeably in the disjunction and the reformulated MILP constraints. Since t_o^{max} is governed by Eq. (9), the constraints for t_o^{max} in the disjunctions do not need to be included in the reformulation. However, since t_o^{max} must be disaggregated, the upper bound constraints are included. A disaggregated variable for t_o^{max} does not need to be defined for the lost sales case since the aggregated variable will automatically be set to zero if the order is neither on-time nor backlogged. The upper bound constraints on the disaggregated z_o are redundant and not included.

$$D_{o,t} \cdot t \leq t_o^{max} \qquad \forall o \in O, t \in \left\{\overline{t_o^e}, \dots, |TP|\right\} \qquad (9)$$

$$\begin{bmatrix} OT_o \\ \overline{t_o^e} \leq t_o^{max} \leq \underline{t_o^d} \\ z_o \leq p_o^e - m_{1,o} \cdot \left(t_o^{max} - \overline{t_o^e}\right) \end{bmatrix} \vee \begin{bmatrix} BL_o \\ \underline{t_o^d} < t_o^{max} \leq \underline{t_o^{ls}} \\ z_o \leq p_o^l - m_{2,o} \cdot \left(t_o^{max} - \underline{t_o^d}\right) \end{bmatrix} \vee \begin{bmatrix} LS_o \\ t_o^{max} = 0 \\ z_o \leq p_o^{ls} \end{bmatrix} \qquad \forall o \in O \qquad (10)$$

$$\left.\begin{aligned} t_o^{max} &= t_o^{max,1} + t_o^{max,2} \\ t_o^{max,1} &\leq \underline{t_o^d} \cdot OT_o \\ t_o^{max,2} &\leq \underline{t_o^{ls}} \cdot BL_o \end{aligned}\right\} \qquad \forall o \in O \qquad (11)$$

$$\left.\begin{aligned} z_o &= z_{o,1} + z_{o,2} + z_{o,3} \\ z_{o,1} + m_{1,o} \cdot t_o^{max,1} &\leq \left(p_o^e + m_{1,o} \cdot \overline{t_o^e}\right) \cdot OT_o \\ z_{o,2} + m_{2,o} \cdot t_o^{max,2} &\leq \left(p_o^l + m_{2,o} \cdot \underline{t_o^d}\right) \cdot BL_o \\ z_{o,3} &\leq p_o^{ls} \cdot LS_o \end{aligned}\right\} \qquad \forall o \in O \qquad (12)$$

Objective Function: The objective is to maximize profit, which is the cumulative sales profit for the orders in the system. It is assumed that operating costs (e.g., labor costs) are fixed and taken into account in linear functions used to quantify z_o. The objective function given in (13) is used.

$$\max \sum_{o \in O} z_o \qquad (13)$$

Variable Domains: The domains of the variables are given below,

$$0 \leq t_o^{max} \leq \underline{t_o^{ls}} \qquad \forall o \in O \qquad (14)$$

$$0 \leq t_o^{max,1} \leq \underline{t_o^d} \qquad \forall o \in O \qquad (15)$$

$$0 \leq t_o^{max,2} \leq \underline{t_o^{ls}} \qquad \forall o \in O \quad (16)$$

$$p_o^{ls} \leq z_o \leq p_o^e \qquad \forall o \in O \quad (17)$$

$$0 \leq z_{o,1} \leq p_o^e \qquad \forall o \in O \quad (18)$$

$$0 \leq z_{o,2} \leq p_o^l \qquad \forall o \in O \quad (19)$$

$$p_o^{ls} \leq z_{o,3} \leq 0 \qquad \forall o \in O \quad (20)$$

$$OT_o, BL_o, LS_o \in \{0,1\} \qquad \forall o \in O \quad (21)$$

$$D_{o,t} \in \{0,1\} \qquad \forall o \in O, t \in TP \quad (22)$$

$$OS_{o,s,t} \in \{0,1\} \qquad \forall o \in O, s \in S \backslash S^{sink}, t \in TP \quad (23)$$

$$OS_{o,s,t} \in \left\{0, \dots, \left|L^{sink}\right|\right\} \quad \forall o \in O, s \in S^{sink} t \in TP \quad (24)$$

$$W_{o,l,a,t} \in \{0,1\} \qquad \forall o \in O, l \in L, a \in A_l, t \in TP \quad (25)$$

The discrete-time STN model is given by Eqs. (1),(2),(6)-(9),(11)-(25).

3.4.2 Heuristics

Another option to schedule customer orders in the business process is to use heuristics. There are two classes of decisions that need to be made when operating the supply chain business processes: 1) what queue should an order be assigned to for a particular task (if there is more than one)?, and 2) what priority or rank should the order be given in the queue that it is assigned to?

Queue Assignment: the following heuristics are available when there is more than one agent able to perform a transaction on an order,

- Join-shortest-queue (JSQ): the order is assigned to the agent with the least number of orders in its queue with ties broken randomly.
- Join-fastest-queue (JFQ): the order is assigned to the agent with the shortest expected response time (shortest time the order is expected to wait until the transaction is completed on it). Any ties are broken at random.
- Join-any-queue (JAQ): the order is assigned randomly to one of the agents capable of performing that task.
- Join-designated-queue (JDQ): the order is assigned to an agent that processes orders with certain characteristics (e.g., from the same customer or customer segment).

Order Priority: Several heuristics are available to assign priorities to orders when they arrive at an agent's queue,

- Highest-profit (HP): the order is assigned a priority proportional to the profit it would have if it was fulfilled at the time of arrival at the queue. This value is obtained from the profit function for that order (see **Figure 1**). This priority heuristic is dynamic in that it is updated each time the order is preempted, or moves downstream in the network to the next queue.

- Soonest-due-date (SDD): the order is assigned a priority that is inversely proportional to the order due date, meaning that orders that are sooner are given a higher priority. This priority is static in that it does not change as the order moves through the network.
- Shortest-expected-remaining-processing-time (SERPT): the order is assigned a priority that is inversely proportional to the expected remaining processing time, meaning that orders that are expected to complete the transaction sooner are assigned a higher priority. Whenever a new order arrives at the queue, its expected remaining processing time is equal to its expected processing time. However, its priority is reevaluated each time a new order enters the queue and whenever it is preempted. Thus, it is a localized dynamic policy for each queue.
- Highest-priority-customer-segment (HPCS): the order is assigned a priority proportional to the customer segmentation priority.
- First-in-first-out (FIFO): the priority is solely based on order of arrival. No preemption occurs.

3.5 Simulation-Optimization Integration

Each of the engines in the digital twin can be used separately by using the simulation engine to run Monte Carlo simulations or using the optimization engine to run offline optimizations. However, additional value is generated by integrating the two engines. In this case, the user can decide to run Monte Carlo simulations using the heuristics described in Section 3.4.2, or the user can leverage the strengths of mathematical programming by embedding MILP optimization routines in the simulation engine. In the latter case, the optimization is embedded as an event in the discrete event simulation that is triggered periodically at fixed intervals or in response to a system condition (e.g., when a new order arrives or after a disturbance). When an optimization event is triggered, a snapshot of the system state and the process metagraph are used to generate the STN model, which is then optimized to schedule the active orders and assign the resources (agents) involved in each transaction. The resulting schedule obtained from the model is then translated into order priorities and queue assignments throughout the network, which are passed to the discrete event simulation where the system queues are updated accordingly. Since the optimization is an event in the simulation, the computational time spent building the MILP model and performing the scheduling optimization is accounted for in the duration of the optimization event. The simulation does not pause for the optimization to occur, but keeps marching forward in time, allowing for a more realistic implementation of optimization in the business process environment. Thus, the optimization frequency and solution time plays an important role in the simulation-optimization integration. If an optimization event takes too long relative to the process time scales, the system may have deviated significantly from the state used to generate the model and the solution found may no longer be relevant.

4. Examples

For the examples below, the following heuristics are defined based on those presented in Section 3.4.2,

- Heuristic P: Highest-profit prioritization with join-fastest-queue assignment.
- Heuristic D: Soonest-due-date prioritization with join-fastest-queue assignment.
- Heuristic S: Shortest-expected-remaining-processing-time prioritization with join-fastest-queue assignment.
- Heuristic F1: First-in-first-out with join-fastest-queue assignment.

- Heuristic F2: First-in-first-out with join-shortest-queue assignment.
- Heuristic F3: First-in-first-out with join-any-queue assignment.
- Heuristic PD: Priority is obtained by averaging the priorities from Heuristics P and D; join-fastest-queue is used.
- Heuristic PS: Priority is obtained by averaging the priorities from Heuristics P and S; join-fastest-queue is used.
- Heuristic PDS: Priority is obtained by averaging the priorities from Heuristics P, D, and S; join-fastest-queue is used.

4.1 Example 1: benefit of business process scheduling

Example 1 is an overly simplified example to illustrate, by inspection, the benefit of using optimization models to schedule orders in the order fulfillment process. **Figure 9** shows the layout of the order fulfillment network, which has three stages: create order, produce material, and ship order. The first stage has one customer service representative (CSR) processing orders at a rate of 1 h/order. The manufacturing stage has two plant sites (A and B), which produce product at a rate of 3 h/order. The shipping stage has one third party logistics agent (3PL) that takes 2 h to ship orders. The STN model is compared against each of the heuristics for a system with five orders that enter the system at the zeroth hour and have different due dates and profit functions as shown in **Figure 10**. In this toy example, a 14-hour scheduling horizon is used, and the MILP model (710 binary variables, 35 continuous variables, and 1,885 constraints) outperforms the heuristics by finding the optimal schedule in 0.2 s of CPU time. The STN model yields a system profit of $8.81 thousand and fulfills four out of five orders on time as shown in **Table 1**. This is a 2-8% improvement in profit and an improvement in on-time fulfillment of up to 100% in some cases. **Figure 11** shows the optimal schedule obtained with the best heuristic (top) and the STN model (bottom). Although the best heuristic (Heuristics PD and PDS) uses both the order profit and the due date to assign priorities, it underperforms relative to the STN model which has a complete mathematical model of the system that accounts for not only order profit and due dates, but also for processing times and resource availability. As a result, the STN model is able to fulfill the first order on-time, whereas Heuristics PD and PDS fulfill it 5 days late.

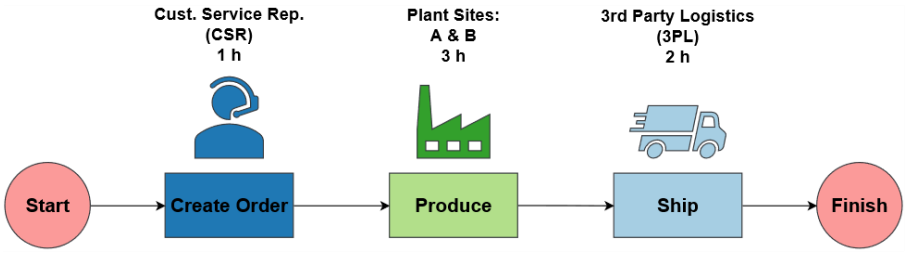

Figure 9. Network layout and process parameters for Example 1

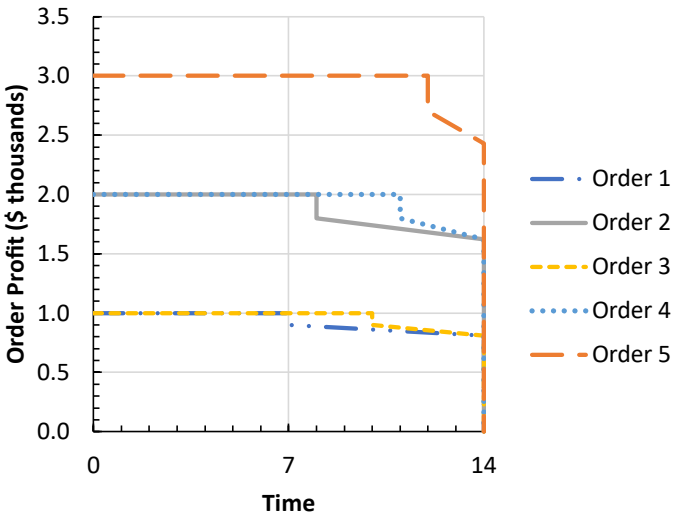

Figure 10. Order profit functions for each of the five orders in Example 1

Table 1. Scheduling results for Example 1. Cells in red indicate a late delivery for that order with respect to the order due date. Cells in green indicate an on-time delivery for that order.

			Fulfillment Date							
Order ID	**On-time Profit**	**Due Date**	**Heuristic P**	**Heuristic D[a]**	**Heuristic F2**	**Heuristic F3**	**Heuristic PD[b]**	**Heuristic PS**	**MILP**	
1	$1,000	7	14	6	6	6	12	14	6	
2	$2,000	8	10	8	8	8	8	10	8	
3	$1,000	10	12	10	10	12	14	12	14	
4	$2,000	11	8	12	14	10	10	8	10	
5	$3,000	12	6	14	12	14	6	6	12	
			$8,405	$8,170	$8,620	$8,285	$8,646	$8,405	$8,810	**Profit**
			40%	60%	80%	60%	60%	40%	80%	**On-time**
			100%	100%	100%	100%	100%	100%	100%	**Fulfilled**

[a]The same outcome is observed for Heuristics S and F1
[b]The same outcome is observed for Heuristic PDS

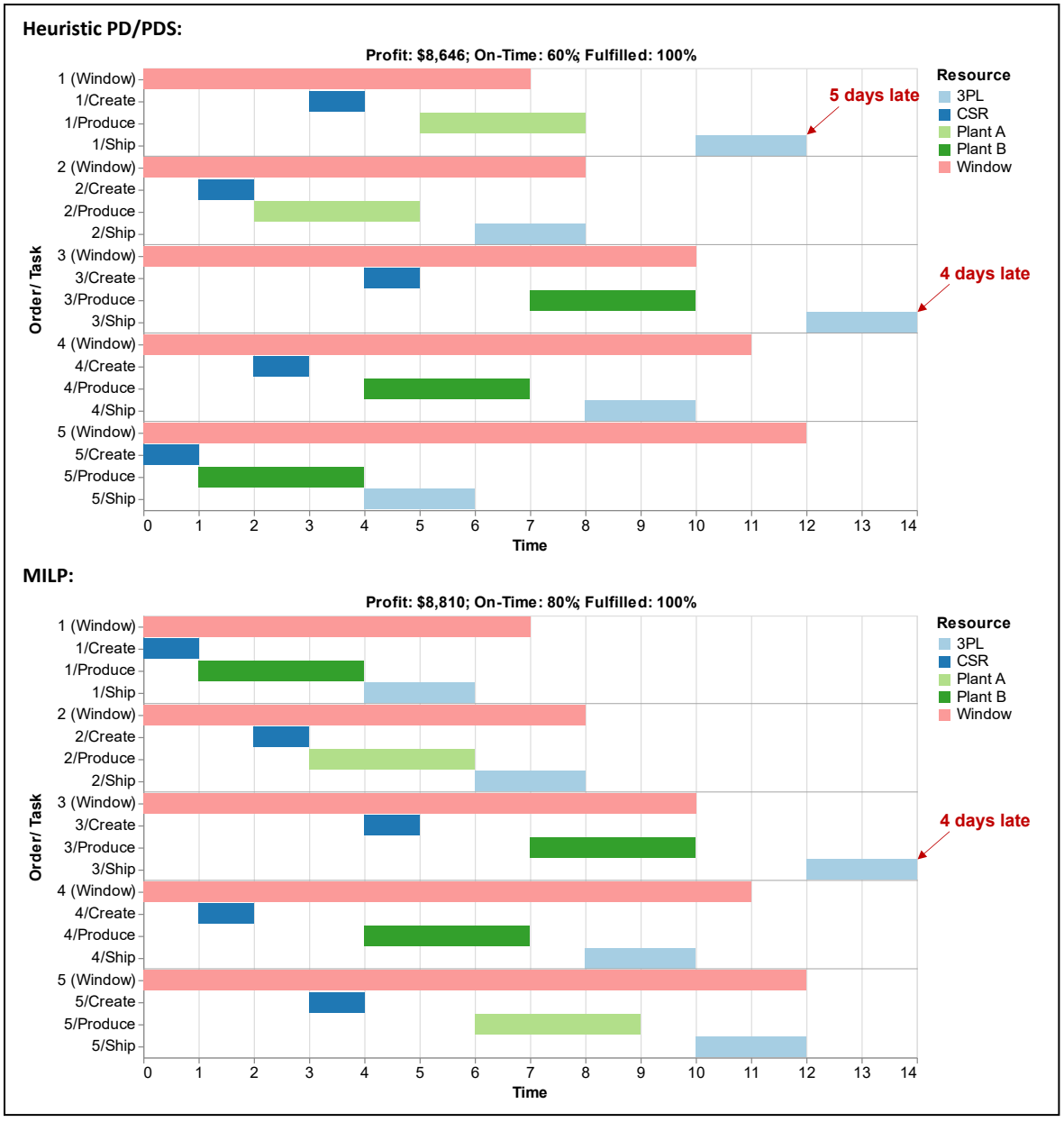

Figure 11. Gantt chart for the best heuristic solution (Heuristic PD and PDS; top) and the optimal schedule (MILP; bottom) in Example 1

4.2 Example 2: benefit of business process scheduling when under order control

Example 2 illustrates the impact of system congestion on the different scheduling approaches. This is done by increasing the number of orders in the system shown in Example 1 from one order to 40 orders. The profit functions for the orders are generated at random with the following characteristics,

- All orders arrive at the beginning of the scheduling horizon (T = 0 hours),
- The order due dates are sampled uniformly from the range between T = 6 hours and T = 20 hours to account for a minimum lead time of 6 hours,

- The lost sales dates are sampled uniformly from the range between the order due date and T = 20 hours,
- The order fulfillment profit is sampled uniformly from the range between $0K and $1K,
- No early delivery incentive is used,
- A 10% profit decrease occurs at the order due date,
- An additional 10% profit decrease occurs by the order lost sales date,
- A profit of $0 is assigned beyond the lost sales date.

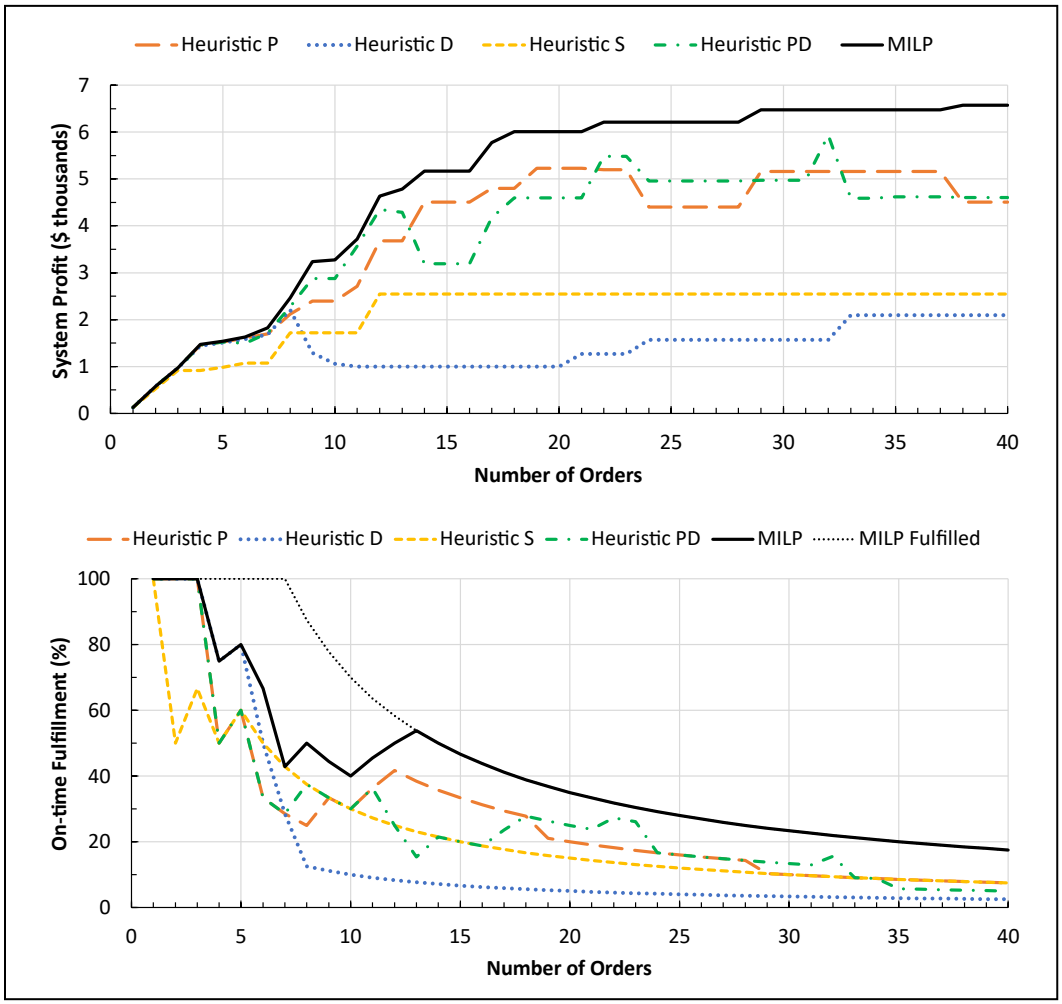

Figure 12. System profit (top) and on-time fulfillment rate (bottom) as a function of the number of orders during the first 20 hours of simulation. Heuristics F1, F2 and F3 are not shown since they have a similar profit profile as Heuristic S. Heuristic PDS is not shown since it has a similar profit profile as Heuristic PD. Heuristic PS is not shown since it has a similar profit profile as Heuristic P.

A simulation horizon of 20 hours is used. Since the minimum lead time for an order is six hours, it becomes impossible to fulfill all orders on-time with the existing resources as the number of orders is increased. **Figure 12** shows the system revenue and on-time fulfillment rate as the number of orders is increased from 1 to 40. Heuristics S and F1-F3 can no longer fulfill all orders after three orders. The other heuristics fail after five or six orders, and the MILP after seven orders. The gap between the system revenue obtained

by the MILP model and the heuristics begins to become more pronounced at this point when the system is under order control. When the system has 40 orders, the gap between MILP model and the heuristics has grown to approximately 40% relative to the best heuristic solution (Heuristic PD). In general, any heuristic that does not use order profit information performs poorly, as expected. Decreases in the profit profiles on some of the heuristics occur when a new order enters the system and receives a higher priority, but cannot be fulfilled due to its earlier lost sale date. However, since it received a higher priority, it tied up resources that could not be used by other orders that has previously been fulfilled. Once again, this highlights the benefit of a model-based approach that captures the entire profit profiles, processing times, and resource utilizations.

4.3 Example 3: using the digital twin to test impact of online optimization (stochastic environment)

The previous examples were deterministic because all orders entered the system at the beginning of the simulation horizon and the task durations were fixed and known. Example 3 now shows the performance of the MILP model in a more complex and realistic order fulfillment process that is simulated with uncertainty in the order parameters (arrival times, due dates, and profits) and process parameters (task durations). **Figure 13** represents the order fulfillment process network, with parallel tasks and multiple agents. The network contains two sets of parallel transactions. The first parallel branching occurs downstream of the *Credit Check* step, such that the *Schedule Shipment* and *Reserve Inventory* steps can be executed independently of each other, but are prerequisites to the *Check-in Shipment* step. The second set of parallel tasks occurs at the end of the process, where the *Create Invoice* and *Goods Issue* steps must both be completed for an order to be considered fulfilled. In terms of resource availability, each task has one agent assigned to it, except for the *Reserve Inventory* or *Load Goods* transactions, which have two agents assigned. At each transaction, three different Gaussian distributions are used to model the event duration. The mean task durations (μ) for each of these distributions is indicated next to each task in **Figure 13**. Orders are assigned to one of these distributions at each transaction based on the order class and customer segment. The orders in the system are generated at random with the following characteristics,

- The earliest due date for each order is its release date,
- The due date for each order is sampled uniformly from the range between 2 hours after its release date (this is the soonest an order can be processed at expectation) and 15 hours after its release date,
- The lost sales date for each order is sampled uniformly from the range between its due date and 5 hours after its due date,
- The profit for each order at its due date is sampled from the *Normal* distribution with mean \$1K and standard deviation \$0.1K,
- A 10% early fulfillment incentive is used on all orders,
- A 30% profit decrease at the due date occurs on all orders,
- An additional 50% profit decrease by the lost sales date occurs on all orders,
- A profit of \$0 is applied to all orders beyond the lost sales date.

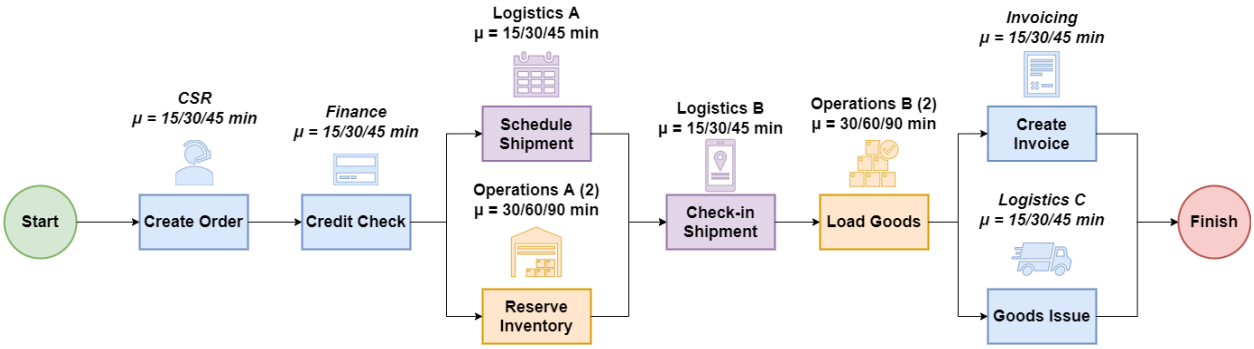

Figure 13. Order fulfillment task network for Example 3

The system is modeled across a 24-hour horizon, in which 47 customer orders enter the system at random with interarrival times sampled from the *Exponential* distribution with mean interarrival times of 35 min. The system being modelled is found to be stable from a queueing perspective since the mean arrival rate is lower than the expected task durations, which results in a mean utilization factor that is less than unity at expectation. Furthermore, since orders will eventually be forced to leave as lost sales if not fulfilled, the number of orders in the system is bounded, which ensures that the network queues remain stable. On average, the number of orders in the system does not exceed 30 orders, as shown in **Figure 14**. As can be observed, the number of orders at the end of the simulation horizon does not drop to zero. This is done to provide a more realistic simulation where orders continue arriving throughout the simulation horizon, resulting in a number of orders still in the system by the end of the simulation. Thus, the simulation model does not experience end-of-horizon effects that are seen in other studies.

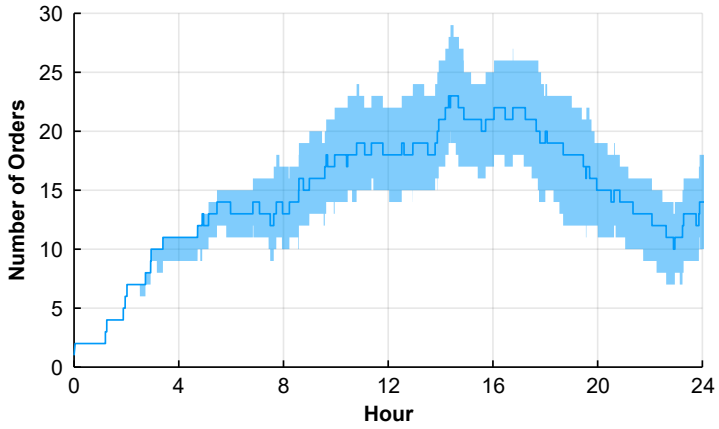

Figure 14. Number of orders in the system when run using Heuristic F3

In Example 3, the heuristics and MILP approaches are compared under three main cases with varying degrees of uncertainty. For the first case, the coefficient of variation (CoV, ratio between the standard deviation and the mean) of the task durations is set at 10%. The CoV is increased to 30% in the second case and to 90% in the final case. For each case, 30 random simulations are performed with a simulation horizon of 24 hours. The nine heuristics and the MILP model are run against each other to dynamically schedule the system orders in each of these 90 simulation instances. The MILP model (STN model) uses a 15-min resolution, and is triggered each time a new order arrives in the system (every 35 min on average). The models are solved with a 1% optimality gap tolerance and a 10 min time limit. In practice, the

termination criteria will likely need to be fine-tuned to determine the criteria values that will allow finding good feasible solutions with minimal delays relative to the simulation timescales. It should be noted that the MILP model uses a rolling horizon approach (Lima *et al.*, 2011) that is adaptive in the sense that it does not use a fixed look-ahead horizon, but instead uses a custom look-ahead horizon at each run that is equal to the last lost sales date for the current orders in the system. For comparison purposes, a second version of the MILP model is used in which a shrinking scheduling horizon (Balasubramanian and Grossmann, 2004b) is used, meaning that the STN model uses the remaining simulation horizon for its scheduling horizon when triggered. The two solution approaches are illustrated in **Figure 15**. It is expected that the shrinking horizon solution approach will achieve a higher performance at the expense of artificial end-of-horizon effects. Of the two modeling approaches, the adaptive rolling horizon approach is the one to be used in practice.

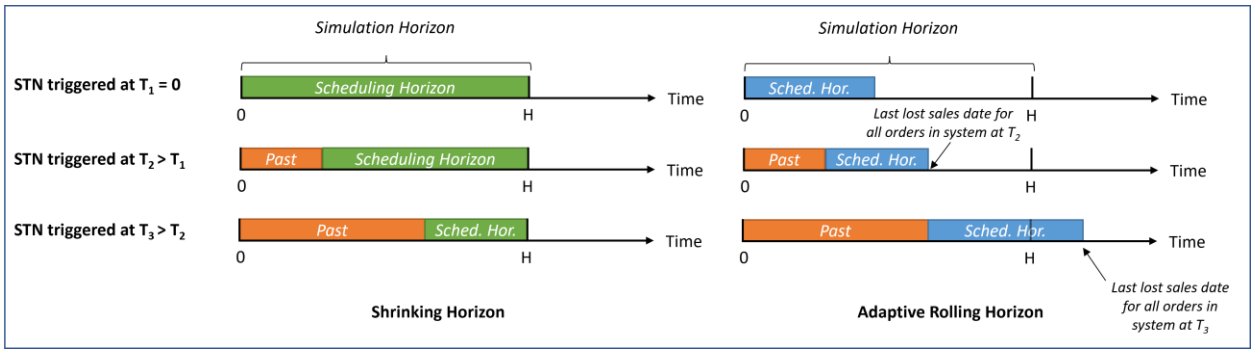

Figure 15. Illustrative comparison between the shrinking horizon solution approach (left) and the adaptive rolling horizon approach (right).

The simulation results in **Figure 16** and **Table A1** (see **Appendix**) show that the deterministic Rolling MILP model outperforms the heuristics even in the stochastic simulation environment. The mean profit from the Rolling MILP approach is statistically different from all the heuristics in each CoV case with a confidence level of at least 99.6%, based on a correlated (paired) Student's t-test. The mean profit attained by the Rolling MILP model is greater than that obtained with each heuristic by at least 3% and at most 70%. Since the Rolling MILP model is deterministic, the model error increases with the coefficient of variation of the task durations, resulting in a decrease in the mean system profit. However, a performance decrease is also observed in the heuristics. On a run-by-run basis, the Rolling MILP model was observed to yield a system profit that ranged from approximately 13% below to 79% above the profit of the other heuristics in the 10% and 30% CoV cases, and approximately 30% below to 170% above in the 90% CoV case. Out of the 810 simulation instances (90 instances for each of the 9 heuristics), the Rolling MILP was outperformed by a heuristic in 13% of the instances.

Table A2 shows the mean relative performance of each of the heuristics on a run-by-run basis, relative to the Rolling MILP model. The worst relative performance by a heuristic occurs for Heuristic D in the 90% CoV case, which has a profit that is 40% below the Rolling MILP profit on average. The best relative performance by a heuristic is for Heuristic PD in the 30% CoV case, which has a profit that is only 3% below the Rolling MILP profit on average. The Shrinking MILP model attains a profit that is 7-14% higher on average. Despite having a higher apparent profit, the Shrinking MILP model neglects that the system continues beyond the 24-hour simulation horizon. As a result, no schedule is generated at the end of the simulation and any remaining orders are erroneously assumed to be lost sales. On the other hand, the

Rolling MILP model generates a schedule that extends beyond the 24-hour simulation horizon in the last optimization runs, which is the desired optimization mode since orders are still present in the system. With the final schedule generated near the end of the simulation horizon, the Rolling MILP model is expected to attain an average profit of \$40.2 ± 0.8 thousand for the 10% CoV case, \$37.9 ± 0.5 thousand for the 30% CoV case, and \$27.6 ± 1.6 thousand for the 90% CoV case, which exceeds the Shrinking MILP average profit by 20-27%.

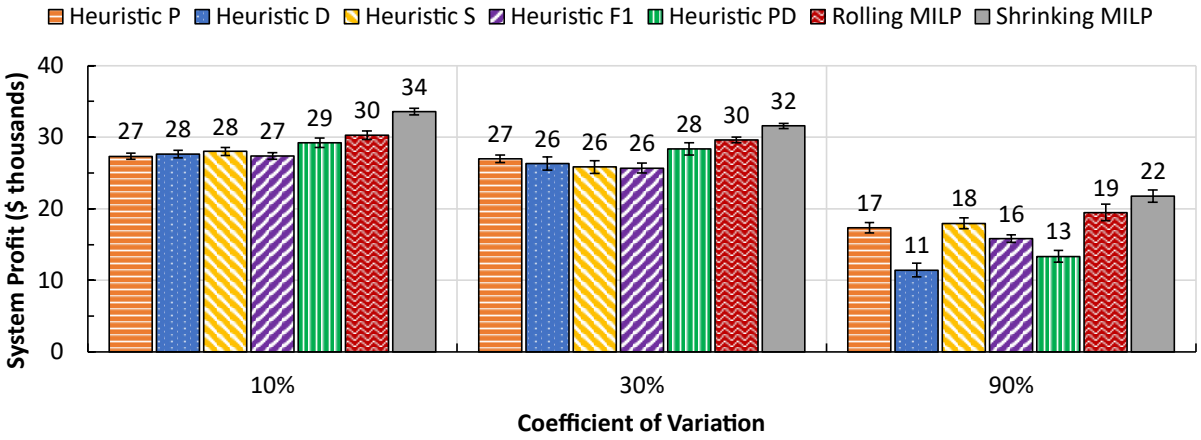

Figure 16. Mean system profit for each of the methods used in Example 3 in each of the CoV cases with 95% confidence intervals.

The Rolling MILP model also takes the lead in the on-time fulfillment rates as shown in **Figure 17** and **Table A3**, with a mean on-time fulfillment rate exceeding the best heuristic by 4-8 points on average. The mean on-time fulfillment rate in the Rolling MILP model is statistically greater than in each of the heuristics with a confidence of at least 98.9% based on a correlated Student's t-test. The gap between the Rolling MILP model and the best heuristic decreases as the system uncertainty increases. In terms of total order fulfillment, the Rolling MILP model only surpasses all heuristics in the 90% CoV case, whereas the Shrinking MILP model is comparable to or superior to all the heuristics in each case on average. The Shrinking MILP model exhibits higher fulfillment rates, relative to the Rolling MILP model because the Rolling MILP model schedules orders beyond the 24-hour simulation horizon; however, their fulfillment is not accounted for in the simulation metric.

Regarding system preemptions, which are shown in **Figure 18** and **Table A4**, the Rolling MILP model introduces more preemptions than every other heuristic on average, except for Heuristic PS in the 90% CoV case and Heuristic PDS in the 30% and 90% CoV cases. The Shrinking MILP model presents fewer preemptions than the Rolling MILP model due to the end-of-horizon effects. Towards the end of the simulation horizon, the Shrinking MILP model reduces the preemption frequency since it assumes that the system ends at T = 24 hours. Once again, these end-of-horizon effects are not desired, supporting the use of the Rolling MILP model, which extends to T = 36 hours in the last optimization run, over the Shrinking horizon approach.

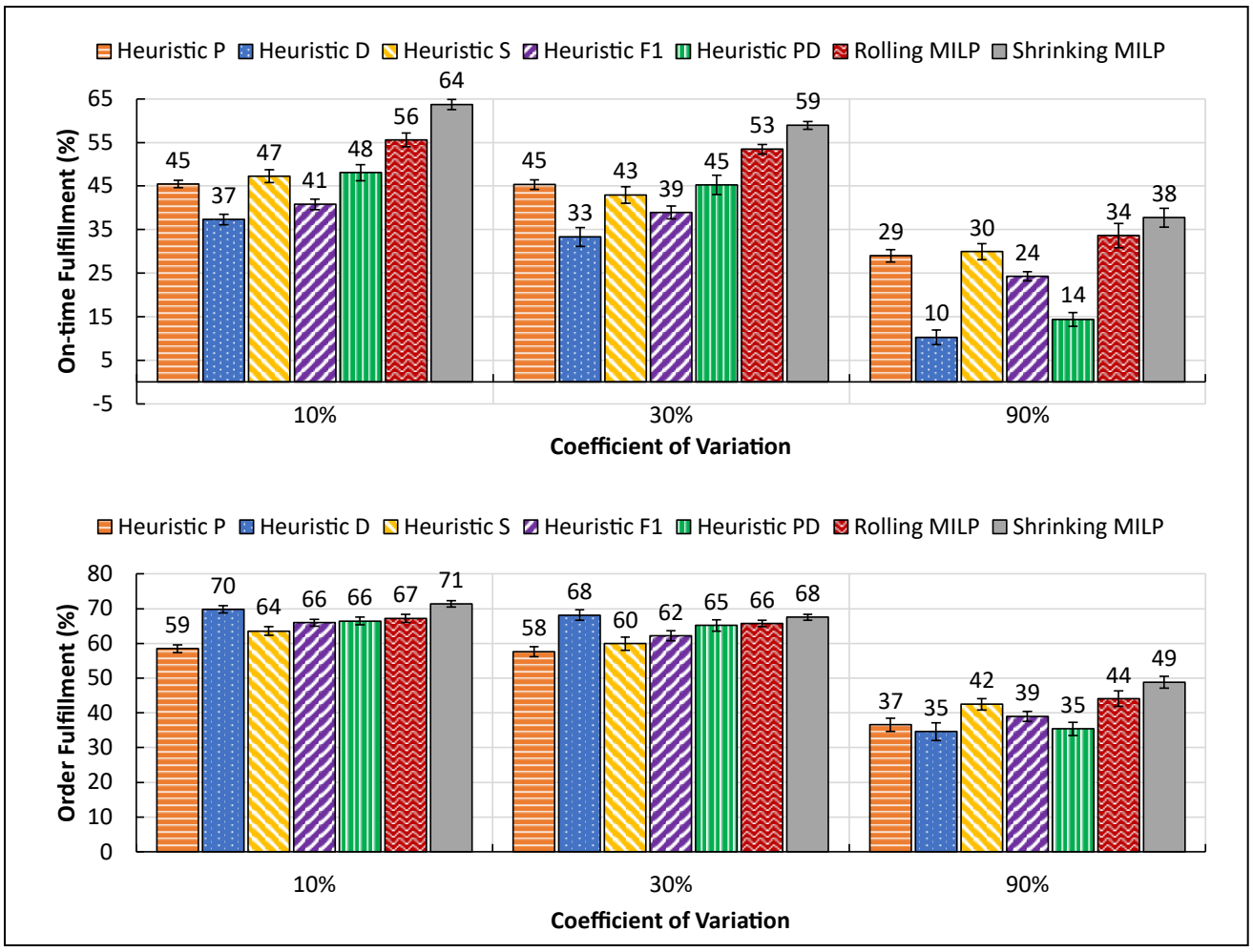

Figure 17. Mean on-time order fulfillment (top) and order fulfillment (bottom) for each of the methods used in Example 3 in each of the CoV cases with 95% confidence intervals.

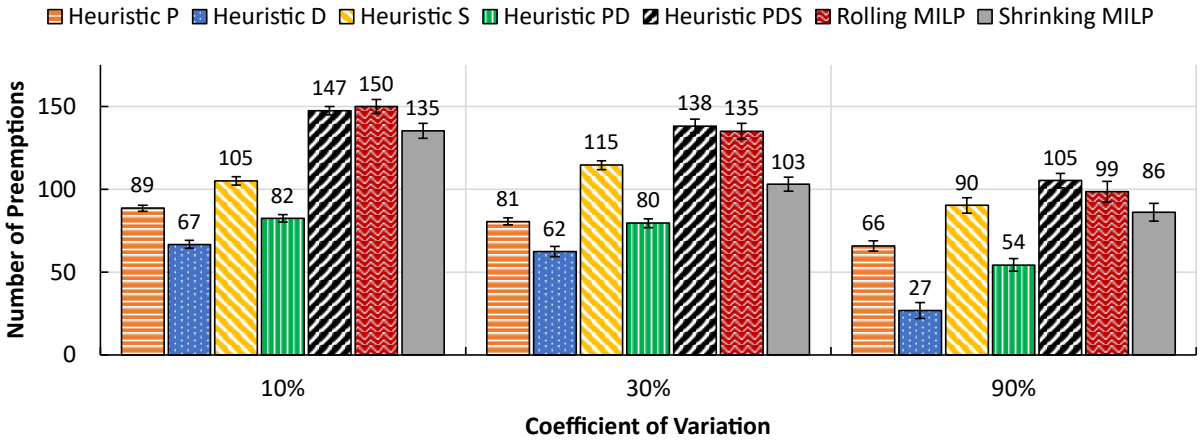

Figure 18. Mean number of preemptions for each method used in Example 3 in each of the CoV cases with 95% confidence intervals

A closer look at the preemptions indicates that the Rolling MILP model tends to preempt the *Reserve Inventory* and *Load Goods* steps more often than the other heuristics (at least 86% higher than the next highest heuristic) as shown in **Figure 19** and **Table A5**. This is due in part to the fact that these are the only two transactions with more than one agent. As a result, the optimizer exploits the differences in the expected task durations between the two agents to its advantage. This occurs in at most 17% of the preemptions in the *Reserve Inventory* step and 12% of the preemptions in the *Load Goods* step. The increased number of preemptions in these transactions is also likely due to the fact they are the most time-consuming steps at expectation. Thus, adjustments in the scheduling of orders at these steps is likely to have a greater impact on the system performance. It should be noted that since the transactions in the business process are for the most part executed by human agents, there is a cost to interrupting the jobs done by these agents, especially if it involves reassigning the job to a different agent. Various approaches can be taken to mitigate the occurrence of preemptions in the STN model if the specific business requires it. Some approaches to accomplish this include,

- Adding an upper bound to the number of preemptions allowed,
- Penalizing preemptions in the objective function,
- Increasing the remaining service time for a preempted transaction to account for inefficiencies resulting from the preemption,
- Making agent reassignments forbidden when more than one agent is available to perform a transaction,
- Making preemptions forbidden altogether for certain steps where preempting is not allowed in practice (e.g., a load goods step may not allow preemptions in some businesses since there is a significant setup time that occurs to start such an operation, and that setup time would be required when a preempted load goods step is restarted).

Each of these approaches comes at the expense of a lower profit margin, but may be required to realistically model the business process.

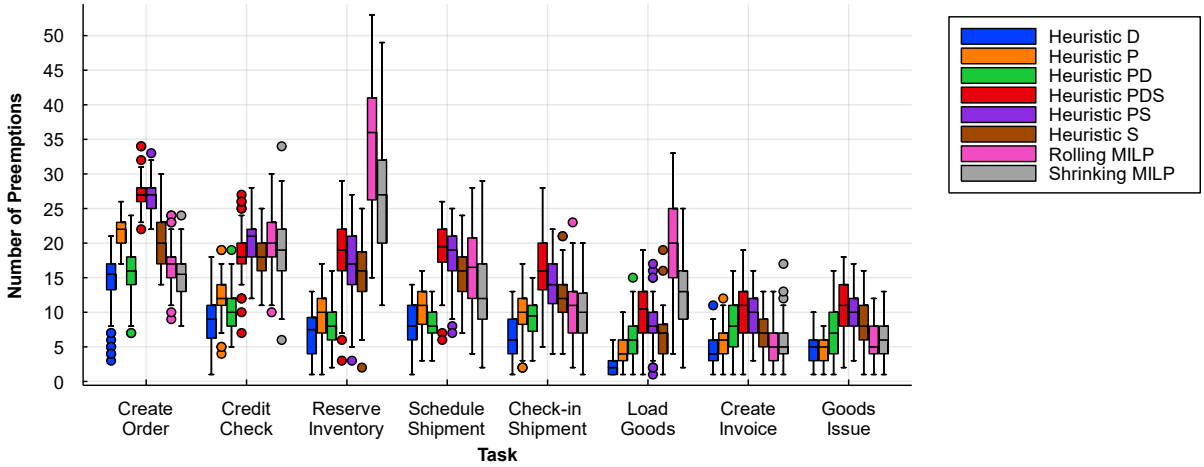

Figure 19. Boxplot of the number of preemptions by task in Example 3

Upon comparing the heuristics among themselves, it is observed that the heuristics that assign orders based on the *join-fastest-queue* heuristic (all except Heuristics F2 and F3) have similar performance when the uncertainty is low (10% and 30% CoV). Their performance is superior to those that use other queue

assignment heuristics by at least 10% on average. Interestingly, despite having a comparable profit to the other heuristics, Heuristic D has the lowest on-time fulfillment rate in the low uncertainty cases. However, because it has the highest total fulfillment rate, it makes up the difference in terms of profit. Once the CoV is increased to 90%, some clear differences are observed between the heuristics. Heuristics D and PD show poor performance. Although Heuristic P benefits from scheduling the orders with the highest profit, Heuristic S benefits from prioritizing orders with lower expected service time, ensuring that more orders are fulfilled both on-time and overall.

The simulation outputs can also be used to determine vulnerabilities in the supply chain business process. Analyzing the utilization factors (ratio of arrival rates to service rates) for each of the agents in the system shows that the *Customer Service* agent has a utilization factor that is greater than 90% on average regardless of the scheduling policy used as shown in **Figure 20**. This does not provide much capacity for errors or other contingencies. With these results, a business leader could then determine if additional resources should be allocated at certain transactions in the business process.

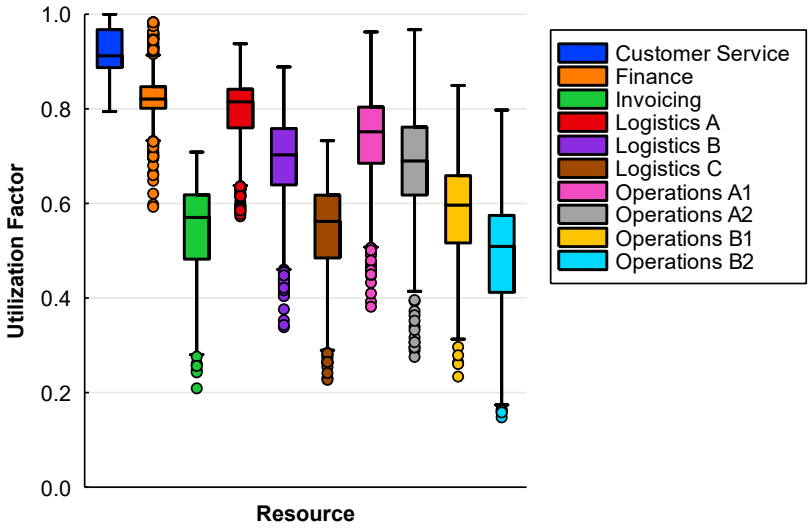

Figure 20. Utilization factor boxplot for each of the resources in Example 3

4.4 Example 4: using the digital twin to for disturbance mitigation

Example 4 illustrates how the digital twin can be used to assess the impact of system disturbances and evaluate potential mitigation strategies prior to their implementation in the real business process. The system in Example 3 is modified to assess the impact of losing a resource at the *Load Goods* step. Three mitigation strategies are evaluated,

- Disruption Approach: do nothing. The system continues to run with a single resource for the *Load Goods* step.
- Restore Approach: hire an additional resource comparable to the one that was lost two days after the disturbance occurred.
- Invest Approach: hire two additional resources comparable to the one that was lost two days after the disturbance occurred.

Prior to the disturbance, the simulation engine was initialized with two days of simulation time. When the disturbance occurs on day two, five random instances of customer orders are generated as described in Example 3. For each of these instances, 30 random simulations are run with a 30% CoV for the task durations. **Figure 21** shows the number of enqueued orders at the *Load Goods* step, as well as the total number of orders in the system and the cumulative number of orders lost due to exceeding the fulfillment window. The *Baseline* results show the performance if no disturbance had occurred. The simulation results show that if an additional resource is hired after two days (*Restore Approach*), the system rapidly stabilizes to its baseline queue length levels within three hours. On the other hand, if no mitigation is performed (*Disturbance Approach*), the load on the system increases, as indicated by the gap in the total system orders with respect to the baseline. The gap in the mean queue length at the warehouse with respect to the baseline is much more pronounced with the mean queue length in the perturbed system being more than twice that of the baseline on average. A drop is observed in the mean number of orders at the end of day four for the perturbed system. However, this drop is not due to any increased system performance, but due to an increase in lost orders. The perturbed system subsequently rebounds back to its previous queue length levels after day five. The negative impact of the disturbance is made clear in the cumulative lost sales plot. When no mitigation is performed, the increase in lost sales is more pronounced. When the disturbance is mitigated, the slope on the mean lost sales is comparable to that of the baseline system, with an offset of approximately 25 orders. The Invest Approach is not shown in the plots because it follows the same profiles as the *Restore Approach*.

Table 2 reports the mean profit, on-time fulfillment, and total fulfillment for each of the scenarios. Of the mitigation strategies presented, the Invest Approach results in the smallest gap with respect to the baseline, with a decrease in 10-20% in each performance metric. This is attributed to the fact that this approach allows the *Load Goods* step to operate with a 17% lower load on average. However, it involves hiring an additional agent, which may not be best financial decision since the difference between the Invest Approach and the Restore Approach is quite small (less than 2%).

Table 2. Mean performance for each of the approaches with 95% confidence intervals

	Approach			
Parameter	**Baseline**	**Disruption**	**Restore**	**Invest**
System Profit ($K)	174 ± 0.4	124 ± 0.5	147 ± 0.4	148 ± 0.5
On-time fulfillment (%)	51.3 ± 0.2	33.3 ± 0.2	40.3 ± 0.2	41.1 ± 0.2
Order fulfillment (%)	74.7 ± 0.2	55.1 ± 0.2	65 ± 0.2	65.6 ± 0.2

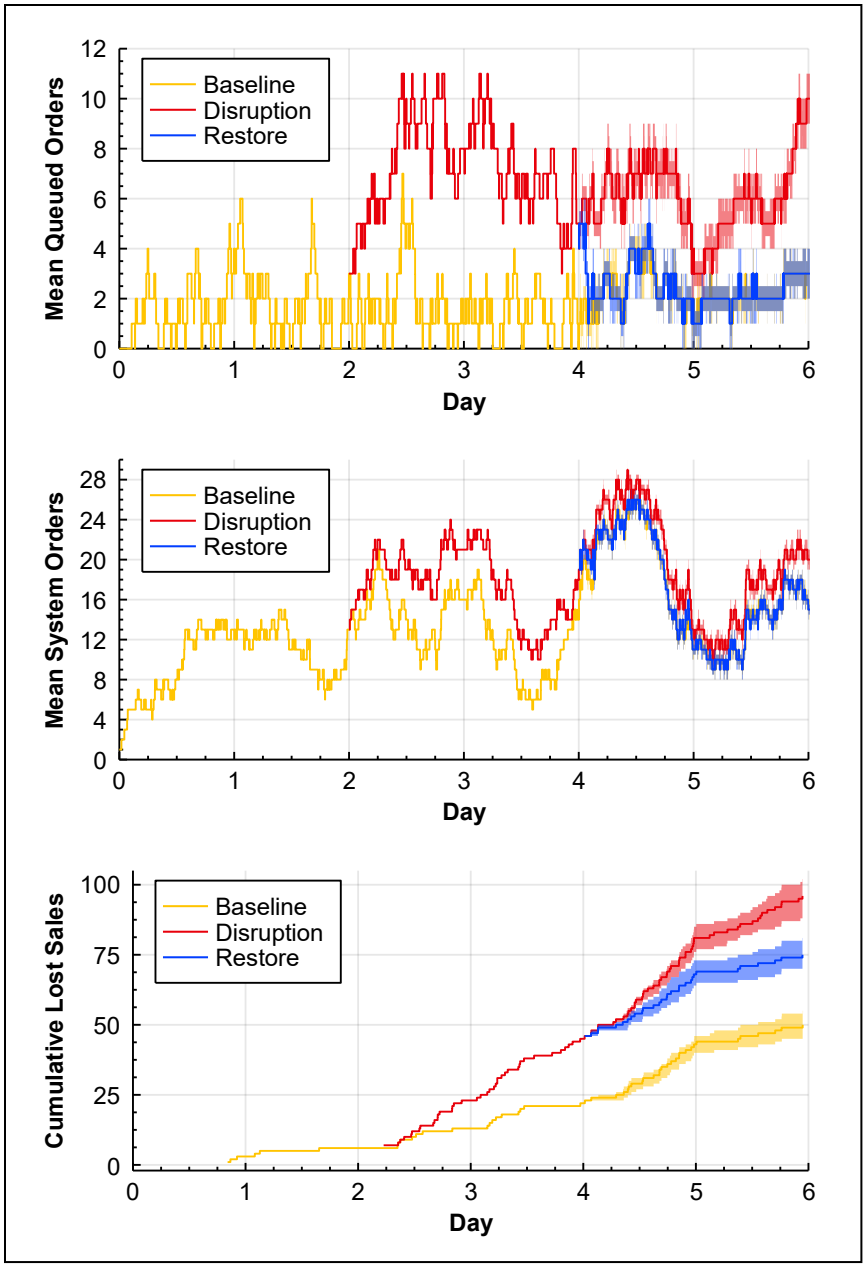

Figure 21. Mean number of orders queued at the *Load Goods* step (top; with 99% confidence intervals), mean number of orders in the system (middle; with 99% confidence intervals), and cumulative lost sales (bottom; with maximum and minimum bands) for Example 4

4.5 Example 5: using the digital twin to provide better promised delivery dates to customers

Example 5 illustrates how the digital twin can be used to provide more accurate quoted delivery dates to customers. The following modifications are made to the system in Example 3,

- The mean interarrival time for customer orders is 45 min.

- The order profit is sampled from the uniform distribution with a minimum of $0K and a maximum of $1K. Orders are split among three profit classes: low ($0.00K - $0.33K), medium ($0.33K - $0.67K), and high ($0.67 - $1.00K).
- Order profits are constant, and no penalties are assessed.
- Due dates and lost sales dates are removed.

The system operating with highest profit priority (Heuristic P) is observed to be stable as indicated by the number of orders in the system in the historian database displayed in **Figure 22**. 400 2-week periods are superimposed in **Figure 22**, showing that the number of orders in the system varies between 0 and 40, with less than 25 orders in most cases. The 800 weeks of historical data are used to determine the lead time distributions for orders based on their profit class, as shown in **Figure 23**. Without the advantages of a digital twin, businesses would likely need to use the lead time distributions for each order class, or the aggregated lead time distribution, to quote promised delivery dates to customers. However, improved lead time estimates can be obtained by leveraging the capabilities of the digital twin to forecast system performance when a customer places a new order.

In Example 5, a new order enters the system after there are already seven other orders in the system at various locations in the order-to-cash network. Of the existing orders, five are high-valued, one is medium-valued, and another low-valued. The simulation engine is then initialized with the current system state and 10 Monte Carlo simulations are subsequently run to estimate the fulfillment date for that order based on the current system state and the scheduling policy being used (Heuristic P). Each Monte Carlo instance is performed by running 20 random simulations with a simulation horizon of 36 hours, giving a total of 7,200 hours of simulation per instance and 72,000 hours overall. The high performant features of the Julia language are leveraged in this example to speed up the simulation runs by using the out-of-the box multi-threading features to run the simulations in parallel.

For illustrative purposes, Example 5 is executed three times, one for each class, by assuming a different value for the new order ($0.25K, $0.50K, and $0.75K). The estimated mean lead times obtained by both Monte Carlo simulation are then compared with those from the historical lead time distributions in **Table 3**. In this example, the expected lead times obtained from the Monte Carlo simulations are lower than both the aggregated mean lead time and the segmented mean lead times for each value range. Although a conservative approach could be taken by using the historical lead times, this is not the best approach since it will result in higher inventory levels if customers do not accept early shipments. It should be noted that the digital twin produces distributions for the new order's expected lead time, which have much less variability (lower CoV) than the historical lead time distributions. Clearly, using a high-fidelity simulation engine can facilitate providing lead time quotes that are more accurate because they are based on the current system state and system forecasts generated via Monte Carlo simulation.

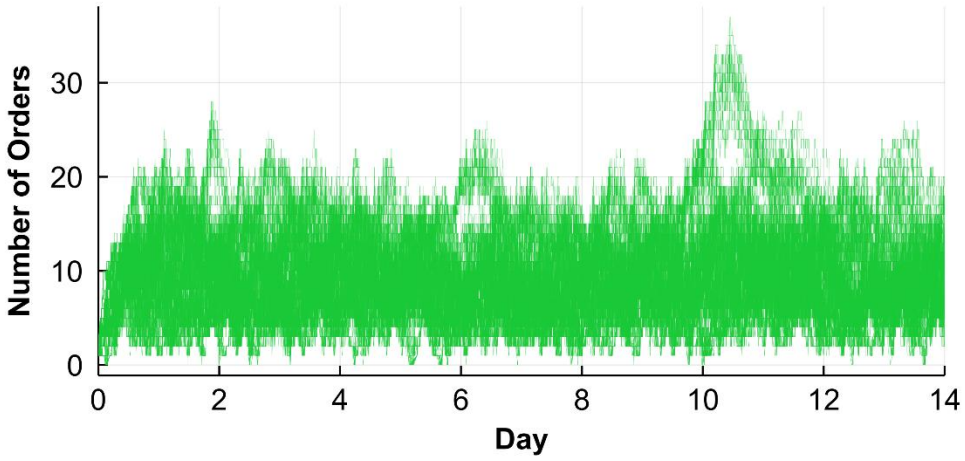

Figure 22. 800 weeks of historical orders in the system superimposed over a 2-week period

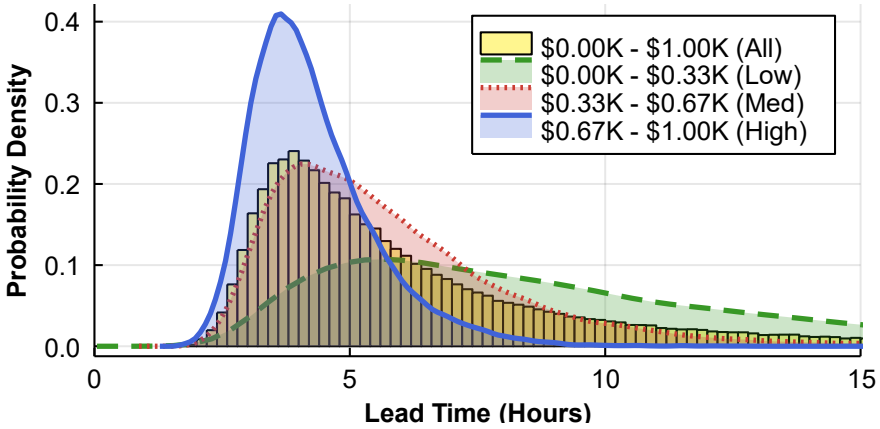

Figure 23. Lead time distributions for orders in the historian database

Table 3. Comparison between the simulated lead time for the new order and the expected lead times from historical data. 95% confidence intervals are reported for the mean lead times.

		Aggregated Historical Data		**Segmented Historical Data**		**Digital Twin Simulation**	
Order Price	**Order Class**	*Mean (hr)*	*Deviation (hr)*	*Mean (hr)*	*Deviation (hr)*	*Mean (hr)*	*Deviation (hr)*
$0.25K	Low	6.7 ± 3.4	4.7	10 ± 0.1	6.4	6.6 ± 0.1	0.8
$0.50K	Medium	6.7 ± 3.4	4.7	5.8 ± 0	2.5	4.1 ± 0.1	0.6
$0.75K	High	6.7 ± 3.4	4.7	4.3 ± 0	1.2	3.7 ± 0.1	0.6

5. Conclusions

A framework for a supply chain business process digital twin is presented, which leverages the highly performant packages from the open-sourced Julia ecosystem. The use of a task-network abstraction layer

enhanced with business process management notation (BPMN) and process metadata is used to readily generate simulation and optimization models for the business processes being studied. The internals of the digital twin's simulation and optimization engines are described with five examples to illustrate the benefits of using a digital twin to assign priorities to customer orders, increase system revenue and on-time fulfillment, evaluate disturbance mitigation approaches, and quote accurate order lead times.

The STN scheduling paradigm from the chemical process industry is applied to the supply chain business processes. A novel approach for integrating optimization models within simulation engines is presented, in which optimization events are embedded in the stochastic discrete event simulation engine of the digital twin. This approach provides a more realistic modeling by allowing for the MILP model build times and solution times to be taken into account in the simulation. In the examples shown, the STN model surpasses each of the heuristics despite being a deterministic model. This confirms the benefit of using mathematical models that capture the system features (e.g., order prices, backlog penalties, processing times, due dates, and lost sales dates) and rely on this holistic view to assign orders to queues and prioritize the orders in each of the queues.

The simulation results indicate that the performance of the heuristics is problem-specific, highlighting the benefit of using the digital twin to select and fine tune the best operating policy from the portfolio of scheduling policies. Overall, there is great promise in integrating simulation with optimization in the context of a digital twin to improve the performance of supply chains in the digital era.

Future work includes adding business rules that are used in practice into the digital twin. These can be used to set order priorities and queue assignments, as well as dictate the outcomes of decisions at split XOR nodes. Next steps on the mathematical modeling side include extending the STN formulation to capture the recycle loops and XOR logical gates observed in industrial supply chain business processes. Another area of further development is in integrating financial and material flows in the supply chain digital twin to obtain a holistic supply chain virtual replica.

6. Abbreviations

BIM	Building information modeling
BPMN	Business process management notation
CoV	Coefficient of variation
DAMCLS	Decision analysis, modeling, control, and learning systems
DES	Discrete-event simulation
DSC	Digital supply chain
DSS	Decision support system
EWO	Enterprise-wide optimization
FIFO	First-in-first-out
GIS	Geographic information system
HPCS	Highest-priority-customer-segment
HP	Highest-priority
IoT	Internet of things
JAQ	Join-any-queue
JDQ	Join-designated-queue
JFQ	Join-fastest-queue

JSQ	Join-shortest-queue
MILP	Mixed-integer linear programming
MP	Mathematical programming
OTC	Order-to-cash
PSE	Process systems engineering
RPA	Robotic process automation
SCM	Supply chain management
SDD	Shortest-due-date
SERPT	Shortest-expected-remaining-processing-time
STN	State-task network
VRP	Vehicle routing problem
XOR	Exclusive OR operator

7. Nomenclature

	Description
Sets	
$a \in A$	Agents
$a \in A_l$	Agents assigned to stage l
$o \in O$	Customer orders
$l \in L$	Tasks
$l \in L_a$	Tasks performed by agent a
$l \in L^{src}$	Source tasks (have no predecessors)
$l \in L^{sink}$	Sink tasks (have no successors)
$l \in L_s^{pred}$	Tasks preceding state s
$l \in L_s^{succ}$	Tasks succeeding state s
$s \in S$	Order states
$s \in S^{src}$	Source order states (predecessors to L^{src})
$s \in S^{sink}$	Singleton final order state
$t \in TP$	Discrete time points (or time periods)
$T \in TV$	Time
Parameters	
$E_{o,t}$	Binary parameter indicating if order o enters the system at time point t.
Δt	Temporal resolution.
h	Scheduling horizon.
$\underline{h}$	Rounded (floor) horizon
$OS_{o,s,t}^{init}$	Initialization parameter for order o indicating the arrival of an order state at state s at time point t
$p_o^e, p_o^d, p_o^l, p_o^f, p_o^{ls}$	Early, due, late, final, and lost sales profits for order o.
$T_o^r, T_o^e, T_o^d, T_o^{ls}$	Release, early, due, and lost sales dates for order o.
$\overline{t_o^r}, \overline{t_o^e}, \underline{t_o^d}, \underline{t_o^{ls}}$	Rounded release, early, due, and lost sales dates for order o in terms time points.
τ	Processing time random variable.
$\overline{\tau}_{i,l,a}$	Expected processing time for order class i in stage l by agent a.
$\overline{\tau}_{o,l,a}$	Expected processing time for order o in stage l by agent a.
$\tau'_{o,l,a}$	Rounded expected processing times (ceiling) in terms of time points.
$Y_{o,l}^0$	Binary parameter indicating if task l was already completed on order o in the past (by time point 0).

$Y_{o,l,\omega}^{np}$	Binary parameter indicating if task l by agent a on order o is expected to end at $t = \omega$.
$Y_{o,s}^{p}$	Binary parameter indicating if the successor task $l \in L_s^{succ}$ is active at time 0
Continuous Variables	
$OS_{o,s,t}$	Order state level for order o at state s and time point t.
t_o^{max}	Fulfillment date for order o.
$t_o^{max,1}, t_o^{max,2}$	Disaggregated fulfillment date for order o.
z_o	Profit for order o.
$z_{o,1}, z_{o,2}, z_{o,3}$	Disaggregated profit for order o.
Binary Variables	
BL_o	Binary indicating if order o is backlogged.
$D_{o,t}$	Binary indicating if order o is delivered at time point t.
OT_o	Binary indicating if order o is fulfilled on-time.
LS_o	Binary indicating if order o is a lost sale.
$W_{o,l,a,t}$	Binary indicating if task l is triggered on order o by agent a at time point t.

8. Acknowledgements

The authors gratefully acknowledge the support and funding from Dow, Inc. and the Center for Advanced Process Decision-making at Carnegie Mellon University.

9. References

van der Aalst, W.M.P. (2010), "Business process simulation revisited", *Lecture Notes in Business Information Processing*, Vol. 63 LNBIP, Springer, Berlin, pp. 1–14.

van der Aalst, W.M.P. (2013), "Business Process Management: A Comprehensive Survey", *ISRN Software Engineering*, Vol. 2013, pp. 1–37.

van der Aalst, W.M.P., Bichler, M. and Heinzl, A. (2018), "Robotic Process Automation", *Business and Information Systems Engineering*, Springer Fachmedien Wiesbaden, Vol. 60 No. 4, pp. 269–272.

van der Aalst, W.M.P., ter Hofstede, A.H.M. and Weske, M. (2003), "Business process management: A survey", in van der Aalst, W.M.P. and Weske, M. (Eds.), *International Conference on Business Process Management*, Springer, Berlin, Heidelberg, Eindhoven, The Netherlands, pp. 1–12.

Alicke, K., Rachor, J. and Seyfert, A. (2016), *Supply Chain 4.0 – the next-Generation Digital Supply Chain*, available at: https://www.mckinsey.com/business-functions/operations/our-insights/supply-chain-40--the-next-generation-digital-supply-chain (accessed 25 January 2022).

Andriessen, N. and Lauwens, B. (2017), "SimJulia: The Good, the Bad and the Ugly", in L'Ecuyer, P., Cools, R., Schmid, W.Ch., Dimov, I., Müller-Gronbach, T. and Lécot, C. (Eds.), *11th International Conference on Monte Carlo Methods and Applications*, Montreal, p. 122.

Balasubramanian, J. and Grossmann, I.E. (2004a), "Approximation to Multistage Stochastic Optimization in Multiperiod Batch Plant Scheduling under Demand Uncertainty", available at:https://doi.org/10.1021/IE030308.

Balasubramanian, J. and Grossmann, I.E. (2004b), “Approximation to Multistage Stochastic Optimization in Multiperiod Batch Plant Scheduling under Demand Uncertainty”, *Industrial & Engineering Chemistry Research*, Vol. 43 No. 14, pp. 3695–3713.

Barykin, S.Y., Bochkarev, A.A., Kalinina, O.V. and Yadykin, V.K. (2020), “Concept for a Supply Chain Digital Twin”, *International Journal of Mathematical, Engineering and Management Sciences*, Vol. 5 No. 6, pp. 1498–1515.

Bezanson, J., Edelman, A., Karpinski, S. and Shah, V.B. (2017), “Julia: A fresh approach to numerical computing”, *SIAM Review*, SIAM, Vol. 59 No. 1, pp. 65–98.

vom Brocke, J. and Rosemann, M. (2015), “Handbook on business process management 1: Introduction, methods, and information systems”, *Handbook on Business Process Management 1: Introduction, Methods, and Information Systems*, Springer Berlin Heidelberg, pp. 1–727.

Büyüközkan, G. and Göçer, F. (2018), “Digital Supply Chain: Literature review and a proposed framework for future research”, *Computers in Industry*, Elsevier B.V., Vol. 97, pp. 157–177.

Cai, Z., Li, X. and Gupta, J.N.D. (2016), “Heuristics for Provisioning Services to Workflows in XaaS Clouds”, *IEEE Transactions on Services Computing*, Institute of Electrical and Electronics Engineers, Vol. 9 No. 2, pp. 250–263.

Dumas, M., la Rosa, M., Mendling, J. and Reijers, H.A. (2013), *Fundamentals of Business Process Management*, *Fundamentals of Business Process Management*, Springer, Berlin, Heidelberg, available at:https://doi.org/10.1007/978-3-642-33143-5.

Dunning, I., Huchette, J. and Lubin, M. (2017), “JuMP: A Modeling Language for Mathematical Optimization”, *SIAM Review*, Vol. 59 No. 2, pp. 295–320.

Fairbanks, J., Besançon, M., Schölly, S., Hoffiman, J., Eubank, N. and Karpinski, S. (2021), “JuliaGraphs/Graphs.jl: an optimized graphs package for the Julia programming language”, Julia Language, available at: https://github.com/JuliaGraphs/Graphs.jl/.

Goldsman, D., Goldsman, P., Goldsman, D. and Goldsman, P. (2015), “Discrete-Event Simulation”, Springer, London, pp. 103–109.

Grossmann, I.E. (2012), “Advances in mathematical programming models for enterprise-wide optimization”, *Computers and Chemical Engineering*, Vol. 47, pp. 2–18.

Grossmann, I.E. and Trespalacios, F. (2013), “Systematic modeling of discrete-continuous optimization models through generalized disjunctive programming”, *AIChE Journal*, Vol. 59 No. 9, pp. 3276–3295.

Guillén, G., Badell, M., Espuña, A. and Puigjaner, L. (2006), “Simultaneous optimization of process operations and financial decisions to enhance the integrated planning/scheduling of chemical supply chains”, *Computers & Chemical Engineering*, Pergamon, Vol. 30 No. 3, pp. 421–436.

Gupta, D., Maravelias, C.T. and Wassick, J.M. (2016), “From rescheduling to online scheduling”, *Chemical Engineering Research and Design*, Elsevier, Vol. 116, pp. 83–97.

Harjunkoski, I., Maravelias, C.T., Bongers, P., Castro, P.M., Engell, S., Grossmann, I.E., Hooker, J., *et al.* (2014), "Scope for industrial applications of production scheduling models and solution methods", *Computers and Chemical Engineering*, Elsevier Ltd, Vol. 62, pp. 161–193.

Hlupic, V. and de Vreede, G.J. (2005), "Business process modelling using discrete-event simulation: current opportunities and future challenges", *International Journal of Simulation and Process Modelling*, Vol. 1 No. 1–2, pp. 72–81.

Hoenisch, P., Schuller, D., Schulte, S., Hochreiner, C. and Dustdar, S. (2016), "Optimization of Complex Elastic Processes", *IEEE Transactions on Services Computing*, Institute of Electrical and Electronics Engineers, Vol. 9 No. 5, pp. 700–713.

Ivanov, D. and Dolgui, A. (2019), "New disruption risk management perspectives in supply chains: digital twins, the ripple effect, and resileanness", *IFAC-PapersOnLine*, Elsevier, Vol. 52 No. 13, pp. 337–342.

Kondili, E., Pantelides, C.C. and Sargent, R.W.H. (1993), "A general algorithm for short-term scheduling of batch operations-I. MILP formulation", *Computers and Chemical Engineering*, Vol. 17 No. 2, pp. 211–227.

Laínez, J.M. and Puigjaner, L. (2012), "Prospective and perspective review in integrated supply chain modelling for the chemical process industry", *Current Opinion in Chemical Engineering*, Vol. 1 No. 4, pp. 430–445.

Lappas, N.H. and Gounaris, C.E. (2016), "Multi-stage adjustable robust optimization for process scheduling under uncertainty", *AIChE Journal*, John Wiley & Sons, Ltd, Vol. 62 No. 5, pp. 1646–1667.

Lara, C.L., Bernal, D.E., Li, C. and Grossmann, I.E. (2019), "Global optimization algorithm for multi-period design and planning of centralized and distributed manufacturing networks", *Computers and Chemical Engineering*, Elsevier Ltd, Vol. 127, pp. 295–310.

Lauwens, B. (2017), "ResumableFunctions: C# sharp style generators for Julia.", *The Journal of Open Source Software*, The Open Journal, Vol. 2 No. 18, p. 400.

Lee, D. and Lee, S. (2021), "Digital Twin for Supply Chain Coordination in Modular Construction", *Applied Sciences 2021, Vol. 11, Page 5909*, Multidisciplinary Digital Publishing Institute, Vol. 11 No. 13, p. 5909.

Li, X., Qian, L. and Ruiz, R. (2018), "Cloud Workflow Scheduling with Deadlines and Time Slot Availability", *IEEE Transactions on Services Computing*, Institute of Electrical and Electronics Engineers, Vol. 11 No. 2, pp. 329–340.

Lima, R.M., Grossmann, I.E. and Jiao, Y. (2011), "Long-term scheduling of a single-unit multi-product continuous process to manufacture high performance glass", *Computers and Chemical Engineering*, Vol. 35 No. 3, pp. 554–574.

Markets and Markets. (2020), *Digital Twin Market by Technology, Type (Product, Process, and System), Application (Predictive Maintenance), Industry (Aerospace & Defense, Automotive & Transportation, Healthcare), and Geography - Global Forecast to 2026*, available at:

https://www.marketsandmarkets.com/Market-Reports/digital-twin-market-225269522.html (accessed 25 January 2022).

McAllister, R.D., Rawlings, J.B. and Maravelias, C.T. (2022), "The Inherent Robustness of Closed-Loop Scheduling", *Computers & Chemical Engineering*, Pergamon, p. 107678.

Méndez, C.A., Cerdá, J., Grossmann, I.E., Harjunkoski, I. and Fahl, M. (2006), "State-of-the-art review of optimization methods for short-term scheduling of batch processes", *Computers and Chemical Engineering*.

Mota, B., Gomes, M.I., Carvalho, A. and Barbosa-Povoa, A.P. (2015), "Towards supply chain sustainability: Economic, environmental and social design and planning", *Journal of Cleaner Production*, Elsevier Ltd, Vol. 105, pp. 14–27.

Negri, E., Fumagalli, L. and Macchi, M. (2017), "A Review of the Roles of Digital Twin in CPS-based Production Systems", *Procedia Manufacturing*, Elsevier, Vol. 11, pp. 939–948.

Perez, H.D., Amaran, S., Erisen, E., Wassick, J.M. and Grossmann, I.E. (2021a), "A Digital Twin Framework for Business Transactional Processes in Supply Chains", *31st European Symposium on Computer Aided Process Engineering*, Istambul, pp. 1755–1760.

Perez, H.D., Amaran, S., Erisen, E., Wassick, J.M. and Grossmann, I.E. (2021b), "Optimization of extended business processes in digital supply chains using mathematical programming", *Computers and Chemical Engineering*, Elsevier Ltd, Vol. 152, p. 107323.

Schrauf, S. and Berttram, P. (2016), *Industry 4.0: How Digitization Makes the Supply Chain More Efficient, Agile, and Customer-Focused*, available at: https://www.strategyand.pwc.com/gx/en/insights/2016/digitization-more-efficient.html (accessed 25 January 2022).

Shah, N. (2005), "Process industry supply chains: Advances and challenges", *Computers and Chemical Engineering*, Vol. 29 No. 6 SPEC. ISS., pp. 1225–1235.

Shah, N., Pantelides, C.C. and Sargent, R.W.H. (1993), "A general algorithm for short-term scheduling of batch operations-II. Computational issues", *Computers and Chemical Engineering*, Pergamon, Vol. 17 No. 2, pp. 229–244.

Stanford-Clark, A., Frank-Schultz, E. and Harris, M. (2019), "What are digital twins?", *IBM Developer*, 14 November, available at: https://developer.ibm.com/articles/what-are-digital-twins/ (accessed 25 January 2022).

Subramanyam, A., Mufalli, F., Laínez-Aguirre, J.M., Pinto, J.M. and Gounaris, C.E. (2020), "Robust Multiperiod Vehicle Routing Under Customer Order Uncertainty", *Https://Doi.Org/10.1287/Opre.2020.2009*, INFORMS, Vol. 69 No. 1, pp. 30–60.

Trotabas, G. (2019), "The Digital Twin in healthcare: What it is and why it matters", 10 April, available at: https://www.linkedin.com/pulse/digital-twin-healthcare-what-why-matters-ghada-trotabas/ (accessed 25 January 2022).

Wagner, G., Nicolae, O. and Werner, J. (2009), "Extending discrete event simulation by adding an activity concept for business process modeling and simulation", *Proceedings - Winter Simulation Conference*, pp. 2951–2962.

Zhu, Z., Liu, C. and Xu, X. (2019), "Visualisation of the Digital Twin data in manufacturing by using Augmented Reality", *Procedia CIRP*, Elsevier, Vol. 81, pp. 898–903.

10. Appendix

Tabular results from Example 3 are shown in **Tables A1 – A5**.

Table A1. Mean system profit ($ thousands) for each method used in Example 3 in each of the CoV cases with 95% confidence intervals

	Coefficient of Variation		
Method	**10%**	**30%**	**90%**
Heuristic P	27.3 ± 0.4	27.0 ± 0.6	17.3 ± 0.7
Heuristic D	27.7 ± 0.5	26.3 ± 0.9	11.4 ± 1.0
Heuristic S	28.0 ± 0.6	25.8 ± 0.9	18.0 ± 0.8
Heuristic F1	27.4 ± 0.5	25.7 ± 0.7	15.8 ± 0.5
Heuristic F2	24.3 ± 0.3	23.1 ± 0.5	14.7 ± 0.8
Heuristic F3	21.6 ± 0.6	20.9 ± 0.7	12.8 ± 0.7
Heuristic PD	29.2 ± 0.6	28.4 ± 0.9	13.4 ± 0.8
Heuristic PS	29.1 ± 0.5	27.6 ± 0.9	17.8 ± 0.7
Heuristic PDS	29.0 ± 0.7	27.7 ± 0.8	17.1 ± 0.8
Rolling MILP	30.3 ± 0.6	29.6 ± 0.4	19.5 ± 1.2
Shrinking MILP	33.6 ± 0.4	31.6 ± 0.4	21.8 ± 0.9

Table A2. Mean scenario performance ratio for system revenue (relative to the Rolling MILP model) for each method used in Example 3 in each of the CoV cases with 95% confidence intervals

	Coefficient of Variation		
Method	**10%**	**30%**	**90%**
Heuristic P	0.90 ± 0.02	0.91 ± 0.02	0.91 ± 0.06
Heuristic D	0.91 ± 0.02	0.89 ± 0.03	0.59 ± 0.04
Heuristic S	0.93 ± 0.03	0.87 ± 0.03	0.94 ± 0.07
Heuristic F1	0.91 ± 0.02	0.87 ± 0.02	0.83 ± 0.05
Heuristic F2	0.80 ± 0.02	0.78 ± 0.02	0.77 ± 0.06
Heuristic F3	0.71 ± 0.03	0.71 ± 0.03	0.67 ± 0.04
Heuristic PD	0.97 ± 0.03	0.96 ± 0.03	0.70 ± 0.05
Heuristic PS	0.96 ± 0.02	0.93 ± 0.03	0.94 ± 0.06
Heuristic PDS	0.96 ± 0.03	0.94 ± 0.03	0.90 ± 0.07
Rolling MILP	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
Shrinking MILP	1.11 ± 0.03	1.07 ± 0.02	1.14 ± 0.05

Table A3. Mean on-time fulfillment rate (%) and total fulfillment rate (%) for each method used in Example 3 in each of the CoV cases with 95% confidence intervals

	Coefficient of Variation					
	10%		30%		90%	
Method	On-time	Total	On-time	Total	On-time	Total
Heuristic P	45.5 ± 0.8	58.5 ± 1.1	45.3 ± 1.1	57.6 ± 1.5	29.0 ± 1.4	36.6 ± 1.9
Heuristic D	37.3 ± 1.2	69.8 ± 1.1	33.3 ± 2.2	68.2 ± 1.5	10.3 ± 1.7	34.6 ± 2.5
Heuristic S	47.2 ± 1.5	63.5 ± 1.2	43.0 ± 1.9	59.9 ± 1.9	29.9 ± 1.8	42.5 ± 1.6
Heuristic F1	40.8 ± 1.2	66.0 ± 1.0	38.9 ± 1.5	62.2 ± 1.5	24.3 ± 1.1	38.9 ± 1.5
Heuristic F2	37.4 ± 0.8	59.3 ± 0.7	34.8 ± 1.0	56.8 ± 1.1	21.6 ± 1.7	37.0 ± 2.0
Heuristic F3	32.1 ± 1.4	53.6 ± 1.5	31.4 ± 1.5	52.0 ± 1.9	18.3 ± 1.5	32.8 ± 1.6
Heuristic PD	48.1 ± 1.8	66.5 ± 1.2	45.2 ± 2.2	65.2 ± 1.6	14.4 ± 1.6	35.4 ± 1.9
Heuristic PS	50.3 ± 1.2	63.8 ± 1.1	47.4 ± 1.9	61.0 ± 1.7	29.6 ± 1.5	40.7 ± 1.6
Heuristic PDS	47.4 ± 2.1	66.0 ± 1.3	45.9 ± 2.0	62.9 ± 1.5	27.1 ± 1.6	40.6 ± 1.8
Rolling MILP	55.6 ± 1.5	67.2 ± 1.2	53.5 ± 1.1	65.7 ± 0.9	33.6 ± 2.7	44.0 ± 2.2
Shrinking MILP	63.8 ± 1.2	71.4 ± 0.9	58.9 ± 0.9	67.5 ± 0.9	37.7 ± 2.2	48.9 ± 1.7

Table A4. Mean number of preemptions for each method used in Example 3 in each of the CoV cases with 95% confidence intervals

	Coefficient of Variation		
Method	10%	30%	90%
Heuristic P	89 ± 2	81 ± 2	66 ± 3
Heuristic D	67 ± 2	62 ± 3	27 ± 5
Heuristic S	105 ± 3	115 ± 3	90 ± 5
Heuristic F1/F2/F3	0 ± 0	0 ± 0	0 ± 0
Heuristic PD	82 ± 2	80 ± 3	54 ± 4
Heuristic PS	135 ± 3	130 ± 4	107 ± 4
Heuristic PDS	147 ± 3	138 ± 4	105 ± 4
Rolling MILP	150 ± 4	135 ± 5	99 ± 6
Shrinking MILP	135 ± 4	103 ± 4	86 ± 5

Table A5. Mean preemptions by task for each method used in Example 3 with 95% confidence intervals

Method	Create Order	Credit Check	Reserve Inventory	Schedule Shipment	Check-in Shipment	Load Goods	Create Invoice	Goods Issue
Heuristic P	21.3 ± 0.4	12.4 ± 0.6	9.6 ± 0.7	10.6 ± 0.6	10.1 ± 0.7	4.4 ± 0.5	5.8 ± 0.5	4.6 ± 0.4
Heuristic D	14.5 ± 0.8	8.6 ± 0.8	6.8 ± 0.7	7.7 ± 0.8	6.3 ± 0.6	2.6 ± 0.4	4.4 ± 0.5	4.4 ± 0.5
Heuristic S	20.6 ± 0.8	18.0 ± 0.6	15.6 ± 1.0	15.8 ± 0.7	11.8 ± 0.8	6.5 ± 0.7	6.7 ± 0.6	8.5 ± 0.7
Heuristic PD	15.5 ± 0.7	10.3 ± 0.6	7.8 ± 0.6	8.2 ± 0.5	9.4 ± 0.6	6.0 ± 0.6	7.7 ± 0.8	7.3 ± 0.8
Heuristic PS	27.0 ± 0.5	20.1 ± 0.7	16.9 ± 1.0	18.8 ± 0.8	14.1 ± 0.8	8.4 ± 0.7	9.4 ± 0.6	9.8 ± 0.7
Heuristic PDS	27.0 ± 0.4	18.5 ± 0.7	18.5 ± 1.1	19.1 ± 0.8	16.2 ± 1.0	10.2 ± 1.0	10.2 ± 0.9	10.8 ± 0.9
Rolling MILP	16.9 ± 0.6	20.1 ± 0.9	34.4 ± 2.0	16.3 ± 1.2	10.3 ± 0.9	19.1 ± 1.3	5.1 ± 0.6	5.8 ± 0.6
Shrinking MILP	15.4 ± 0.6	18.6 ± 1.0	26.9 ± 1.7	13.1 ± 1.2	9.6 ± 0.9	12.8 ± 1.0	5.6 ± 0.6	6.1 ± 0.6