Metamodel-based cycle time quantile estimation for real-time control of manufacturing systems

Abstract

Controlling the release of jobs in a manufacturing system is key to meet delivery and work in process targets in the system. The control strategy performance is highly dependent on the ability to predict completion time for each job to be loaded. Unfortunately, the distribution of job completion times is analytically intractable for realistic systems. While stochastic process models can provide approximations for job completion time, several challenges and limitations arise for this approach in multi-stage and multi-product manufacturing systems. On the other hand, simulation and data driven approaches require a large number of experiments to estimate the distribution of cycle times as a function of system parameters and/or the system state. As the complexity of the system increases, so does the simulation time, limiting the feasibility of real-time simulation-based cycle time characterization. In this paper, we explore the performance of a simulation metamodel approach to estimate quantiles of interest for the distribution of cycle time in a multi-station manufacturing system as a function of the work in process at each workstation, the operational state of the workstations, and the type of the released job. The metamodel is computationally inexpensive to evaluate, and so could be queried in real time when a loading decision is made and/or a loading sequence determined. We show that, unlike metamodels for steady state cycle time behavior as a function of system design, a relatively small simulation experiment design and simple polynomial regression models provide high fidelity for predicting job completion time quantiles, even with complicating aspects such as rework, job priority, and machine failures.

Keywords: production control, metamodel, cycle time, manufacturing.

1 Introduction

Real-time control has become increasingly difficult as manufacturing systems and their models become more complex, in terms of job variety, machine flexibility and machine reliability. We consider a *control policy* to be a set of rules governing when jobs are released to the manufacturing floor (and perhaps to which machine in a flexible system). The objective of a control policy is to maximize efficiency (machine utilization) while meeting job demand times. Our focus is on supporting control policies that make use of the job completion time distribution. The production scheduling literature calls this the *lead time* or *cycle time*, often interchangeably. When there is a distinction, lead time refers to the total time from customer order to order delivery, but since our

focus is on time spent in the manufacturing process, we follow Nadarajah and Kotz (2008) and use the term *manufacturing cycle time*, or *cycle time* for short. We consider a manufacturing system to be either of job shop or flow shop form, where jobs of a particular type progress through a series of steps at specific machines or workstations, where there is a single queue at each workstation.

In the semiconductor industry, it is particularly critical to develop methods that take into account the variability of workstation backlogs and modify the release decisions based on the system state (Gordon, 1993; Rose, 2001; Bekki et al., 2009; Durmusoglu and Aglan, 2017). In fact, the job release decision should be dependent on the level of congestion of the system, captured by the number of jobs at each workstation at any point in time. Cycle time estimation is important for the job, but also for the layers forming a single wafer. The more accurate the estimation, the better the synchronization among the different workstations. As a result, several policies have been developed for this setting such as *Layerwise CONWIP*, where the work in process is capped with respect to the wafer layers instead of the unfinished product and *CONLOAD* (Rose, 1999), which decides the release based on the load condition of the bottleneck machine (Prakash and Chin, 2015).

Rose (2001) proposed the *TOTAL CT* rule. This rule is based on the average cycle time estimate for each product, together with the associated *flow factor*, the latter defined as the ratio between the cycle time and the raw processing time. Each time a lot starts its processing (i.e., it is released into the system) the global variable *total cycle time* ($\mathrm{C_{TOT}}$) is increased by the average cycle time estimate of the product type being released. Each time a lot leaves a work center, the $\mathrm{C_{TOT}}$ is decreased by the raw processing time at this particular work center times the product flow factor. Lots are only released into the system if $\mathrm{C_{TOT}}$ is lower than some $\mathrm{C^u_{TOT}}$ threshold that is determined a priori by the user or through simulation studies (performed offline).

Maleck and Eckert (2017) considered *timelink* relations in a semiconductor manufacturing plant, where a timelink is defined as a set of time constraints between consecutive process steps, which can be important to guarantee product quality. In their procedure, the authors used a MIP model with rolling horizon for the construction of the optimal schedule for job release. A discrete event simulator was used to estimate the release dates and update the time constraints. In online/real-time applications, the use of simulation can become a critical component and the ability to develop predictive state based models for the variables of interest becomes a key factor of success.

The cycle time distribution is important beyond the manufacturing setting: it is fundamental for managing overall lead time for products in extended supply networks (Bureau et al., 2006; Maleck and Eckert, 2017). Lead time and cycle time are also critical measures for health care service providers. Online scheduling problems and real time optimization of patient acceptance (similar to job release) have been investigated in the health care literature. In particular, the decision on whether to accept walk-in patients in mixed pre-scheduled walk-in clinics was studied in He et al. (2019). The authors used hierarchical stochastic robust optimization for real-time decisions based on the estimated patient waiting time distribution. Time constraints were the focus in Kumar and Barton (2017), where the authors introduced the notion of controlled violations as the ability to monitor a running process and develop an approach based on constraint satisfaction

to determine the best schedule update for its completion to minimize the total penalty from time window violations.

From a management perspective, it is important to observe that cycle time is a random variable, and that repeated simulations of each scenario provide distributional information that can assist the decision maker. But simulations of complex systems are computationally expensive, and cannot support real-time decisions based on distributional information.

We propose a method for real-time evaluation of cycle time distribution information, in particular, cycle time quantiles. The method is based on offline simulation of a fraction of all possible manufacturing system states, fitting a metamodel to these results that can predict cycle time quantiles for any possible state. The rapid calculation enabled by the metamodel allows real-time support of a dynamic strategy for job release. A dynamically changing workload mix can result in shifting bottlenecks, problematic for some other state-based release rules, but represented by the metamodel through simulations with different states and potentially different bottlenecks. Our approach builds on related work in quantile metamodeling, design of simulation experiments, and simulation in production planning.

The next section provides more detail on related research in cycle time estimation and job release control policies. A formal problem description is provided in section 3, and our modeling approach is presented in section 4, together with the design of the simulation experiments for fitting the metamodel is discussed in the following section, including the common random numbers strategy. Section 5 presents numerical studies for a number of systems and assesses the effectiveness of our approach. Section 6 provides a summary of our approach, the implications of the numerical experiments, some potential applications to real-time job release decisions, and suggestions for further investigation.

Contribution While some preliminary findings were presented in Pedrielli and Barton (2019), here we develop a metamodeling strategy for cycle time quantile estimation for a broader class of manufacturing systems, and provide rationale for the good performance of low order polynomial metamodels. We present cycle time quantile metamodeling strategies that enable precise prediction to consider systems with rework, priorities, machine breakdowns, and job type in process at the time of the release decision. We evaluate the impact of common random number strategies in improving the precision of the metamodel. Our numerical results present metamodel fitting experiments consisting of a fraction of all possible states that produce accurate and precise cycle time estimations generally, thus making the proposed approach feasible for real time release policy problems.

2 Related Literature

As highlighted in the introduction, control of manufacturing systems is typically focused on measures of the work in process and the cycle times of jobs in the system. We focus on providing real-time assessment of the state-dependent cycle time distribution, to be used by a real-time manufacturing job release policy. Section 2.1 builds the justification for the need for such a model,

reviewing relevant cycle time based policies with a focus on state dependent hedging time. Subsequently, Section 2.2 introduces the literature in cycle time modeling with focus on manufacturing systems.

2.1 Real Time Control of Manufacturing Systems

Static manufacturing control policies were examined in many studies, including (Rose, 2001; Bekki et al., 2009). Dynamic release policies that determine release time based on the state of the manufacturing system at the time of the release decision are a recent development (Angius et al., 2018). In an early work, Gershwin (2000) proposed a family of *Control Point Policies* (CPP) as an extension of *Hedging Point Policies* (HPP) (Kimemia and Gershwin, 1983). An HPP policy seeks production rates balancing costs for holding inventory against failure to meet delivery demand, hedging against possible machine downtime. The main difference of CPP from HPP is that HPP focuses on cycle time, but CPP controls a proxy referred to as *hedging time*, a conservative measure of the cycle time to complete an individual job taking into account downtime risk. Within the family of dynamic release policies are token-based policies. These have been analytically investigated, and implemented in industrial settings, the most famous being Kanban, CONWIP, and base-stock policies (Liberopoulos, 2013). These policies have the same objective as CPP and HPP, but control the work in progress in different portions of the system instead of the release process. They are applied in single as well as multi-stage systems. The parameters of these policies are typically optimized in a static manner, i.e., while they are dependent on the specific system state, the parameters are set to optimize steady-state performance. Other real-time scheduling policies such as earliest due date or least slack, while dynamic, do not take into account the system state.

Of the methods above, CPP policies are the most closely related to our approach since they use cycle time, characterized as a random variable, and determine the hedging time from a quantile of the manufacturing cycle time probability distribution. When the difference between the job due date and the time at which the decision is taken is less than the hedging time, a job becomes *ready* to be released (control decision). In this paper, we will consider only state dependent policies, that typically require knowledge of the distribution of the cycle time.

State Dependent Hedging Time As mentioned an detailed in Gershwin (2000), we can define a time based CPP to decide, at each time step t, whether to: (i) load part i of type j into machine M_1 (the first machine in the line) and start processing; (ii) Do nothing.The decision is made based on the relevant parameters of the control point policy, which, as mentioned before, are referred to as *hedging times*, T_j for each part type j. In particular, CPP policies need to define whether a part is or not ready to be loaded into the system. In particular, for each job, at time t, the following release condition is proposed in (Colledani and Gershwin, 2017):

$$D_i - t \leq T_{R(i),S(t)} \tag{1}$$

Where D_i is due date for the i^{th} job and $T_{R(i),S(t)}$, where $R(i)$ is the part type associated to job i. The hedging time $T_{R(i),S(t)}$ is the $\gamma_{R(i)}$-quantile of job i cycle time. $\gamma_{R(i)}$ is a parameter of the policy. Differently, in (Angius et al., 2018), the cycle time is used to decide the release for products that are subjected to degradation during processing. This is the case in bio-pharmaceutical and food industries. In this case, the authors in (Angius et al., 2018) propose a policy that acts on the loading of parts at the first machine M_1 only if the second machine M_2 is failed. It defines a vector $\mathbf{b}_u$ of *critical buffer levels*. For each failure type that characterizes the second machine: (i) If the buffer level $b \leq b_j$ and machine M_2 is down in model j: load the part; (ii) If the buffer level $b > b_j$ and machine M_2 is down in model j: stop loading. In this paper, we will consider a different implementation of this family of policies considering again a quantile of each job cycle time. In particular, we want:

$$L_i > T_{R(i),S(t)} \tag{2}$$

where L_i is the lifetime of the i^{th} job. Similar to the hedging time policy using 1, we will set a quantile $\gamma^{'}_{R(i)}$.

2.2 Modeling of Manufacturing Cycle Times

An important literature has been devoted to the study of manufacturing cycle times, since cycle time represents an important measure of system performance. Cycle times are are affected by a number of factors such as capacity, loading, batching, and scheduling. And cycle time is important for modeling costs and defining control policies. As recognized by Karmarkar (1993), when cycle times are modeled, multiple elements need to be taken into account such as the queueing time in buffers, the processing time and the handling time (e.g., transportation time in case of large manufacturing systems such as semiconductor manufacturing).

There is extensive literature related to the study of the average cycle time. For synchronous production systems (i.e., manufacturing systems where the queue can practically be ignored due to the high level of coordination among different stages) static deterministic models have been proposed that estimate the average cycle time as $\widehat{E(C)} = \frac{bsize}{cap}\frac{1+a}{orate}$, where C is the random cycle time, *bsize* represents the batch size, *cap* is the system processing capacity in terms of number of jobs, *orate* is the average output rate and a is the safety factor (Sugimori et al., 1977). Such a simple model is justified by Little's law with an added safety factor in order to account for blocking and other phenomena in the manufacturing system. This formulation does not consider the stochastic dynamics of jobs moving between machines in the manufacturing system and it does not take into account time-specific buffer levels.

To address the issue of dynamics, much effort has been devoted to dynamic models able to capture the evolution of the system work in process by accounting for job-specific start times and finish times at each workstation, and sequencing relationships among the jobs within the system (Jaikumar, 1974; Geoffrion and Graves, 1976; Graves, 1980). These dynamic models are deterministic,

and so wait time is poorly modeled. But it can be argued that in complex manufacturing systems queue/buffer/wait time can be a significant component of the cycle time. For this issue, stochastic dynamic models have been developed for cycle time estimation. As recognized by Karmarkar (1993), this problem is highly complex; it is strongly state dependent and also depends on the specific job routings that occur. The complexity of the problem becomes similar to stochastic project scheduling, making real-time evaluations impractical for all but the simplest systems.

In the realm of continuous models, data driven approaches have been proposed for serial production lines, for the analysis of the cycle time distribution. Recently, Zou et al. (2017) proposed a dynamic model of the product work in process level across the manufacturing system based on continuous flow and calibrated with real-time sensor data from the manufacturing line. While the approach did not assume Bernoulli behavior, the continuous model approximation introduces error that must be accounted for. Further, the dynamics are assumed to be linear in the model, adding a further approximation error to the estimate of the cycle time. The approach focuses on estimation of the expectation instead of the distribution of cycle cycle time. Again, the distribution of the cycle time is not defined as a function of the state of the manufacturing system, but only on recent flow information.

One simplifying approach has been the development of metamodels for the steady-state cycle time distribution as a function of system parameters. By metamodeling, we mean creating a simplified analytic model that approximates the input-output relationships generated by the simulation model (Blanning, 1974; Kleijnen, 2007). In our setting, manufacturing system simulations are run under a variety of system parameter settings, to produce cycle time values that are used to fit a functional approximation. The functional approximation (metamodel) then can be evaluated in real time or evaluated repeatedly and inexpensively for system design space exploration and optimization. For example, Bekki et al. (2014a) and Batur et al. (2018a) developed metamodels for steady-state cycle time quantiles as a function of throughput, machine MTBF and MTTR, load and unload time distributions, and (two) product mix. The models considered were quadratic, cubic and quadratic polynomial quantile regression, and Gaussian process models. The forecast errors ranged from several percent to several tens of percent, with increasing error for models with more predictor variables. Since the modeling was designed to forecast steady-state cycle time, state variables such as work in process by station and job type were not included as predictor variables. This setting is different from ours: we seek to provide a cycle time distribution forecast to assist job release decisions, rather than steady-state forecasts to explore the effect of alternate system process parameters. We assume that the manufacturing process is fixed in terms of stationary process parameters such as throughput, processing and loading time distributions, failure and repair distributions, and so forth. Our focus is on real-time rather than steady-state forecast, and the metamodel predictor variables are real-time state variables rather than process parameters. We show below that for our setting the metamodel structure is simpler and has higher fidelity.

While asymptotic distribution results for cycle time have been presented in the literature (Shantikumar and Sumita, 1988), the distributional analysis of cycle times as a function of a specific state

of a manufacturing systems has been limited to very special cases. A recent study modeled the state-dependent cycle time distribution, but assumed Bernoulli machines (Ju et al., 2016). In (Shi and Gershwin, 2012), and, more recently, in (Shi and Gershwin, 2016b) a method was presented to estimate the distribution of cycle time for the case of machines affected by single failure mode. The analysis was restricted to a two machine line, and was extended to three machines in (Shi and Gershwin, 2016a). Colledani et al. (Colledani and Gershwin, 2017) proposed a model for the steady state probability vector, and derived the time of absorption of a Markov Chain representing the dynamics of the system from the state when a job is placed in the buffer to the state when the job leaves the system. Nevertheless, the authors consider only Bernoulli machines, and analytical results are only available for 2-machine systems.

The estimation of the state-dependent cycle time quantile in the CPP paper by Angius et al. (2018) is the most closely related to our work. The simple structure of their model allows real-time evaluation. They employed an absorbing Markov chain model, but again under an assumption of Bernoulli machine operational state. As we also model, the hedging time values were dependent on the specific state of the system: the number of jobs in process or in buffers and the up/down state of the machines. The authors' approach was analytical, and the accuracy of the cycle time distribution characterization was limited by their restrictive assumptions. The discrete-time Markov Chain (DTMC) representation required equal and deterministic processing times and lead time constraints, in order to characterize the distribution of the manufacturing cycle time.

We estimate state-dependent cycle time distribution under less restrictive assumptions, using simulation and metamodeling. Our approach allows general processing time distributions, multiple job types, rework and job type-specific routing, and other features that can be coded in the discrete-event simulation model. We do not run the model to steady-state, but only until the newly released job is finished. A CPP-like job release strategy could be used with our more realistic cycle time quantile estimate: based on the machine state and work in process at each station, release a job of type i if the estimated cycle time quantile for that state does not exceed the cycle time constraint for that job. Our method allows real-time job release policies based on state-dependent cycle time quantiles, with high fidelity and for more general settings.

3 Problem Description and Notation

In this section, we develop the required notation to formally present the modeling approach that we have taken. To simplify exposition, we focus on a serial production line (flow shop) with multiple job types, each with different processing times at each machine and perhaps different priorities, and buffers used to decouple the machines. The system is shown schematically in Figure 1. This assumption could be relaxed to consider a job shop environment with different routings for different job types; there is limited testing of this through the rework scenario in Section 5.

Consider a manufacturing system producing $i = 1, \ldots, I$ different job types, through a sequence of manufacturing stages (i.e., machine or workstation) $j = 1, \ldots, J$, each machine has a number of

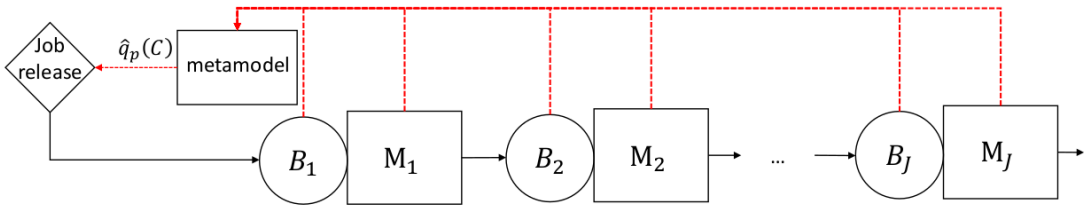


Figure 1: Job flow and information flow for a metamodel-enabled control point policy.

failure modes $k = 1, \ldots K_j, j = 1, \ldots, J$. The state of the system will be characterized by $\mathbf{s}$, composed by the state variables (w, p, r, o, m): $s = \{w_{ij}, p_{ij}, r_{kj}, o_{ij}, m_{kj}; i = 1, \ldots, I; j = 1 \ldots, J, k = 1, \ldots, K_j\}$. Specifically, w_{ij} represents the number of jobs of type i waiting at machine j at the time of the job release decision, p_{ij} the accumulated operating time for a job in process at j (zero or a positive value, at most one p_{ij} not zero for each j). The state components r_{kj} characterize the cumulative time spent in repair for failure mode k at machine j if the machine is currently under repair. Finally o_{ij} is a binary indicator that takes value one if the part of type i is loaded onto machine j, while m_{kj} is a binary indicator that is set to 1 if machine j is down under failure mode k. If $\sum_{k=1}^{K_j} r_{kj} = 0$, then machine j is either idle or processing at the time when the simulation starts. Also, unlike the accumulated operating time p_{ij}, we can have more than one $r_{kj} > 0$ for each j, that is, any machine can be affected by multiple failure types at the same time.

The repair time and the accumulated operating time are clearly dependent. If a machine is under repair, no matter from how many failures, the accumulated operating time will be 0, and no part will be loaded on the machine. We assume that if a job is in process and a failure affects the machine, then the job is scrapped and a new part starts from the first station as replacement job. It is important to notice that this can impact the cycle time of the target job in case the target job type has lower priority than the scrapped job type.

Based on the real-time system state s, the metamodel-forecast cycle time quantile is used in the job release decision. For the cases with failure, We assume bounded distributions for the processing time, failure times and repair times. This assumption is realistic since we are considering manufacturing systems where infinite processing or repair times are impossible. We define τ_{ij} as the random processing time for a job of type i on machine j, with distribution $F_{\tau_{ij}}(t)$, τ_{kj}^f as the random time between failures with associated probability $F_{\tau_{ij}^f}(t)$, and τ_{kj}^r as the random repair time with associated probability $F_{\tau_{ij}^r}(t)$. At the time of a release decision, the total number of jobs in the system, T, is $T = \sum_{i=1}^{I} \sum_{j=1}^{J} (w_{ij} + o_{ij})$. Let the random cycle time of the $(T+1)^{st}$ job be C_{T+1}. Then C_{T+1} is a complex function of the τ_{ij}, τ_{ij}^f, and τ_{ij}^r values for the jobs in process and waiting at each station, and of the failures affecting and that will affect each station following the job's release until its completion. Our objective is to characterize the distribution of C_{T+1}; more specifically to estimate one or more quantiles of this distribution.

4 Simulation Metamodel-Based Quantile Estimation

In this Section, we present a method using simulation and simulation metamodels to construct high quality state-specific estimates for cycle time quantiles. Section 4.1 builds the justification for such models, while section 4.2 shows our basic idea for the data driven quantile model using simple polynomial regression.

4.1 Model Justification

Simplified probability models for manufacturing systems within the broad class introduced in section 3 allow one to see that a low-order polynomial approximation may be adequate for estimating a completion time quantile for a job entering the system. Consider a two-machine tandem operation with all jobs of type 1, with a jobs waiting and in process at machine 1, and b jobs waiting and in process at machine 2 at time zero. Assume, for simplicity of exposition, that $a > 0$ and $b > 0$. Suppose $D_{1,k} \sim F_{\tau_{11}}(t)$ represents the processing time for the k^{th} job at machine 1, and $D_{2,k} \sim F_{\tau_{12}}(t)$ the processing time for the k^{th} job at machine 2. We arrange the indices so that earliest (oldest) job currently in the system has index 1 and the job being released has index $a+b+1$.

First, we consider a simple bound on cycle time of the job being released at time zero with index $k = T+1 = a+b+1$, based on the state vector $(w_1 + o_1 = a, w_2 + o_2 = b)$. Let the cycle time for job $a+b+1$ be C_{a+b+1}. Then we have a lower bound:

$$C_{a+b+1} \geq D'_{2,1} + \sum_{k=2}^{a+b+1} D_{2,k} \tag{3}$$

since all of the jobs in the system when job $a+b+1$ is released must complete at station 2 before job $a+b+1$ can start, and its completion time will add $D_{2,a+b+1}$ to the total. Note that $D'_{2,1}$ represents the remaining processing time for the job currently in process at station 2. If we were to assume exponential processing times then the term $D'_{2,1}$ would be removed and index for the sum would begin at 1 rather than 2. Completion time will be longer if station 2 observes any idle time.

Next, we give an alternative justification for the form (3).

Assuming that station 2 is never idle, the amount of time remaining for processing jobs ahead of job $a+b+1$ when job $a+b+1$ arrives at station 2 will be total processing time for jobs $1, 2, ..., a+b$ less the time of arrival of job $a+b+1$:

$$D'_{2,1} + \sum_{k=2}^{a+b+1} D_{2,k} - (D'_{1,b+1} + \sum_{k=b+2}^{a+b+1} D_{1,k}). \tag{4}$$

The completion time for job $a+b+1$ will be the sum of the remaining time at station 2 plus its arrival time to station 2, canceling the second sum in equation (4).

In semiconductor manufacturing, idle time for processing stations is avoided due to the high cost of the processing equipment. Under this assumption we can view the inequality (3) as tight,

and the cycle time distribution would be approach Gaussian with increasing $(a+b)$. When there are many stations, buffers and many job types, the equivalent of $(a+b)$ in this simple model will grow, the difference between using $D_{2,1}$ and $D'_{2,1}$ in (3) or (4) becomes negligible, and the sum in (3) approaches a Gaussian with mean $(a+b+1)\mu$ and variance $(a+b+1)\sigma^2$ where μ and σ are the mean and standard deviation, respectively, of the final station processing time distribution. With increasing $(a+b)$ the completion time p-quantile for the random variable C_{a+b+1} can be approximated by:

$$q_p(C_{a+b+1}) \simeq (a+b+1)\mu + q_p(\mathbb{Z})(\sqrt{a+b+1})\sigma \tag{5}$$

where $\mathbb{Z}$ is the standard normal random variable. The nonlinear relationship in (5) can be used as a metamodel directly for prediction of the completion time quantile as a function of a and b, or the nonlinear term can be replaced by a Taylor polynomial approximation in a and b about some typical value. This allows a second-order polynomial metamodel. The second order approximation to $\sqrt{a+b+1}$, about the nominal values a_0, b_0, is:

$$(a_0+b_0+1)^{\frac{1}{2}} + \frac{1}{2}\frac{(a-a_0)+(b-b_0)}{(a_0+b_0+1)^{\frac{1}{2}}} - \frac{1}{8}\frac{((a-a_0)+(b-b_0))^2}{(a_0+b_0+1)^{\frac{3}{2}}} \tag{6}$$

This model has leading multipliers that decrease in magnitude for each additional term, and denominators increasing for each additional term. Higher order terms have a rapidly decreasing contribution to the approximation. Note that the quadratic terms include the interaction ab. One can expect the polynomial approximation for a sequence of more than two stations to involve all two-factor products of state variables. Also, two-factor interactions may be more important in settings where the final station can become idle. In our computational studies, we do not enforce full utilization of the final station, and we observe that these interactions improve the metamodel prediction. The bound given by equation (3) can be extended to multiple job types and multiple stations, with the focus always on the processing times at the last station, again assuming it does not become idle. Consider the case with two job types, and, without loss of generality, an introduction of a job of type 1 when the system has a_i jobs of type i at station 1 and b_i jobs of type i at station 2. Assume that processing time for job type i at station j has mean μ_{ij} and standard deviation σ_{ij}. We simplify notation by writing $\sigma_{12}^2 = \sigma^2$ and $\sigma_{22}^2 = \gamma\sigma^2$. Then, the approximate bound given by equation (5) for a job of type 1, given a_1, b_1, a_2, b_2 jobs in the system, becomes

$$\begin{aligned} q_p(C_{1:a_1,b_1,a_2,b_2}) \simeq \quad & (a_1+b_1+1)\mu_{12} + (a_2+b_2)\mu_{22} \\ & + q_p(\mathbb{Z})(\sqrt{a_1+b_1+1+\gamma(a_2+b_2)})\sigma \end{aligned} \tag{7}$$

Taking a second order Taylor approximation of (7) will produce a result similar to (6). Note that the inclusion of the second order term implies the inclusion of pure quadratic and interaction terms in the regression metamodel. Adding rework to the system makes relatively simple changes to the mean and variance of processing times and so similar bounds remain. The situation is more complicated if: i) the downstream stations become idle (this can happen more easily with failures),

or ii) the dispatch rule allows priority based on job type. In our empirical study, the inclusion of interaction terms in the regression metamodel adequately captured these behaviors.

For our case of only three machines, buffer limits of three, and exponential service distributions, the Gaussian approximation in (7) is not very good. Then, metamodeling based empirical quantiles from multiple simulation replications is necessary. This produces higher fidelity, as shown in the computational studies. Because the simulation runs are short, the computational expense for these replications is not great.

4.2 Metamodel Construction

We consider a processing system like the one in Figure 1. The justification for a polynomial approximation suggests constructing a low-order polynomial metamodel for each cycle time quantile and each job type, as a function of the system state, i.e., the number of jobs per type waiting or in process at each machine, and elapsed operating time (or repair time if down) for each machine. We define the independent variables $x_{i,j}$ as a centered and scaled version of $w_{i,j}$, and $x'_{i,j}$ as a coded version of $p_{i,j}$. We will also add $x''_{k,j}$ as a centered version of the repair time state r_{kj}. Let us detail the covariate defined for the elapsed operating time and repair time. In particular, we have that for the operating time:

$$x'_{i,j} = \frac{\bar{\tau}_{ij} - p_{ij}}{\bar{\tau}_{ij} - \underline{\tau}_{ij}} \in [0,1] \tag{8}$$

where $\bar{\tau}_{ij}, \underline{\tau}_{ij}$ are the upper and lower limit of the random variable support, respectively. The basic idea is that the lower the elapsed time, relative to the support of the random variable, the higher the contribution to the quantile. We can further center the covariate between x^{ℓ}, and x^{u} (common for all covariates):

$$x'_{i,j} = x^{\ell} + x'_{i,j} \cdot \left(x^{u} - x^{\ell}\right).$$

Then, we define the independent variable vector:

$$z = \left(w_{11}, \ldots, w_{IJ}, x_{11}, \ldots, x_{IJ}, x'_{11}, \ldots, x'_{IJ}, x''_{11}, \ldots, x''_{KJ}\right). \tag{9}$$

Then, z can be thought of as a scaled version of s. The full second order metamodel (for one particular job type, say i, to be released, and one particular completion time quantile) will have the following form:

$$q_{ip}\left(C\left(z\right)\right) = \beta_{i0} + \sum_{k=1}^{2IJ+KJ} \beta_{ik} z_k + \sum_{k=1}^{2IJ+KJ} \sum_{k'=k}^{2IJ+KJ} \beta_{ikk'} z_k z_{k'}. \tag{10}$$

It is important to notice from equation (10) that the model form is identical, for each job type and each quantile. The simulation setup used to estimate specific metamodel coefficients will clearly differ: each job type requires a different simulation to be run (since job types have different process-

ing time distributions). Each quantile, while based on the same simulation runs, extracts different cycle time quantiles from them. We examined the need for third order polynomial models in our numerical experiments, but found quadratic models adequate, consistent with the justification in Section 4.1. Also, the beta values here are not "true" values in any sense, but coefficients for a numerical approximation based on the second order approximation in (6). A least squares fit makes sense, but unfortunately $q_{ip}(C(z))$ is not known and must be estimated via simulation, $\widehat{q}_{ip}(C(z))$. With this quantity on the left side, an error term, $\varepsilon(i,p,z)$ must be added to the right hand side. The (i,p,z) argument for ε is necessary because the error variance depends on these arguments, as shown in (Pedrielli and Barton, 2019). In that preliminary work, a variance stabilizing transformation did not improve model fit. We use unweighted least squares and untransformed quantile estimates. Weighted least squares can produce lower variance coefficient estimates, but would require added simulation effort to obtain adequate precision in the error variance, and was unnecessary to produce high fidelity in the models fitted in the numerical experiments that follow.

Next we provide an algorithmic description for metamodel construction.

Algorithm 1 Meta-model based cycle time estimation for real time control of manufacturing systems

Input: Number of job types $i = 1,\dots,I$, number of stations $j = 1,\dots,J$, p percentile to be used to define the corresponding quantile q_p; experiment design identifying the starting state vector z for each simulation run, a matrix $\mathcal{D}$ where the n^{th} row $(n = 1,\dots,N)$ provides the experiment condition z for the n^{th} set of replications; R number of replications for each experiment condition.

Output: $\hat{\beta}$, estimation of model (10) coefficients as a function of the system state. One set of coefficient estimates is generated for each job type, and statistical characterization of coefficient error and model prediction error.

1: **Run simulation experiments**: Run each of N simulation conditions R times, divided into B sets of R_b replications, from time 0 (time of job release) until the queue and the loaded job are cleared.
2: **Compute cycle times**: Record the cycle time for each condition and replication. This will result in $I \times N \times R$ cycle times.
3: **Estimate the quantile**: For each released job type $i = 1,\dots,I$, and condition $n = 1,\dots,N$ separate the R observations into B non-overlapping batches of size $R_b = \frac{R}{B}$, resulting in B sets of m clearing time values for each n, $\{C_{inbr}\}$. Compute Y_{in} from C_{inbr} as shown in equation (11).
4: **Model estimation**: Use as independent variable the initial system state as defined by design $\mathcal{D}$, augmented to the quadratic form. For each condition n, the dependent variable is the quantile estimated from the simulation replications Y_{in}.
5: **Estimation**: Estimate the coefficients, assess quality of fit and statistical significance for each of the the regression models.
6: **End**

Note that, in the numerical experiments that follow, we exclude cubic models, variance stabilizing transformations and quantile regression. These models did not improve predictive performance, consistent with the preliminary findings in (Pedrielli and Barton, 2019).

One realization of cycle time is obtained for each of the R replications of the simulation experiment (Step 4 of algorithm 1). We adopt a frequentist approach for the estimation of the cycle time quantile, using both the empirical quantile over all replications, and a batch-based average quantile, the latter providing an approximately normally distributed statistic appropriate for regression. Let B be the number of batches and $R_b = R/B$ be the number of replications per batch. Let C_{inbr} be the cycle time of the r^{th} replication in batch b for a job release of type i given system state described by row n of the experiment design. The expected quantile estimate is then obtained as:

$$Y_{in} = \frac{1}{B}\sum_{b=1}^{B} \tilde{Y}_{inb}, \tag{11}$$

where $\tilde{Y}_{inb}$ is the empirical p-quantile of $C_{inbr}, r = 1, ..., R_b$. In this notation, we have suppressed the dependence on p. Note that $\tilde{Y}_{inb}$ can be computed with respect to any particular value of p from the same set of C_{inbr} values, with no additional simulations required.

4.3 Experiment Design Strategy

The choice of experiment design depends on both the purpose for which the metamodel will be used, and the type of metamodel. These issues are discussed, for example, in (Barton, 2015; Kleijnen, 2007; Sanchez et al., 2018). There has been little written on the design of experiments for quantile regression, which historically has been applied in observational studies with randomly assigned experimental units (Bassett Jr and Koenker, 2017). Bekki et al. (2014b) employed Latin hypercubes for two-factor quantile regression modeling, and (Batur et al., 2018b) employed full-factorial designs for studies involving two and three factors and second through fourth order polynomial regression models.

The fidelity of the state-dependent completion time quantile metamodels developed here depend strongly on the experiment design, $\mathcal{D}$. In addition to buffer contents and priorities, in order to create the experiment design, we need to specify the maximum buffer contents (entities that can reside in queue) that will be simulated. While the metamodel can be used to forecast job completion time quantiles based on buffer contents beyond the limits used in the DOE, it is generally unwise to extrapolate metamodel results far beyond the design region. In order to estimate quantiles, for each starting state in the experiment design (indexed by n), R replications of the simulation are made; each terminates when the released job completes the final station, so these are relatively short, terminating simulations. In the numerical experiments below, R was set to 1000.

For even relatively simple systems, our system state design space has dimension in the tens or hundreds, making multilevel full-factorial designs impractical. From the approximation in (6) we expect at least some quadratic effect, and perhaps higher order effects as well. For that reason, rather than using three-level fractional factorial designs, a fractional design with all four buffer levels (0-3) was employed. For designs that are not too large, a library of orthogonal arrays is available (Sloane, N.J.A., 2007). Nearly orthogonal designs can be found in Cioppa and Lucas

(2007), and two-level fractional factorial designs such as in (Sanchez and Sanchez, 2005) can be used for four-level fractional designs by coding variables using pairs of columns from the two-level designs. The latter was used for the numerical experiments that follow.

Variables for Coding System State In addition to the selection of the set of run conditions for the experiment design, the way in which the system state is characterized has impact on the metamodel fidelity. We describe the parametrization strategies for each class of state variables. The buffer (queue) contents at a workstation could take on values 0 - 3, but these were centered and scaled to +/- 1 for metamodel fitting.

The workstation's machine status was coded as a continuous variable $t_j \in \left[T^{\ell}, T^{u}\right]$. A negative value, say $-t_j$ would indicate that, at the time of the job release, the machine at workstation j has been operating for p_{ij}. A positive value would indicate that the machine has been down and in repair for r_{kj} time units for any active failure $k = 1, \ldots, K_j$. A single quantitative variable can be used because a system is either operational (for a period) or down (for a period). These quantities are important for modeling non-exponential time to failure and time to repair distributions, and permit simulation of the next event (next failure or completion of current repair) based on a conditional distribution. Since different failure modes with different time-to-fail and different time-to-repair distributions may hold, indicator variables were created to indicate the kind of failure active at a particular machine (or none) at the instant of job release. In our numerical experiments, scenarios that do not include machine failures do not use this variable. An important aspect is that we can control the proportion of failed-to-operating experimental conditions by controlling the ratio T^{ℓ}/T^{u}. Favoring conditions with operating machines will lead to $T^{\ell} >> T^{u}$.

Finally, there was a variable for the type of job (if any) loaded on a machine at a particular workstation. The failure type and job in process indicators can be used to generate appropriate values for workstation machine status for the experiment design runs. Interestingly, to reduce the design dimension without compromising design quality, these three indicator variables for each station were combined into one four-level (in-process type = 0,1,2,3) variable to create the experiment design, and the value then used to initialize the experimental condition, but mapped into the three indicator variable values for metamodel fitting.

This parametrization minimized conflicts in setting the design parameters independently in creating a set of experiment runs. For example, a machine status variable greater than one would be combined with the indicated failure type for that run to set a (random) remaining repair time for that simulation run. A job in-process for that prescribed DOE run would then be discarded or assigned for rework in the simulation runs starting from that state.

Common Random Numbers We implemented common random numbers to generate the training data for our learning procedures. A critical aspect to the random numbers implementation was to guarantee that, under the different system configurations (e.g., benchmark, rework, priority), and the different experiment conditions within each system configuration, the random numbers were synchronized (Banks et al., 2013). Random numbers synchronization becomes particularly challenging when considering rework and priority. Indeed, a choice of different random numbers

across different system configurations may lead to the incorrect ranking of the clearing time in the different conditions. For example, it may result that the clearing time for a job of type 1 is larger under priority than under the benchmark case due to different random numbers used in the priority and benchmark runs. This can become an issue when estimating the model.

Our common random number strategy was to assign the *same* realizations of the processing times to the jobs across the experiment conditions *and* across cases (e.g., benchmark, priority, rework, and rework with priority). In order to achieve such synchronization, the processing times for each job were generated offline.

The run specification z_n identifies how many jobs are in the system. If no rework is possible, then it will suffice to "attach" to each job a vector of processing times. In particular, if a job i at time $t = 0$ is in station j, then $J - j + 1$ processing times will be associated to that job. In this way, given the same sequence of jobs, the same random numbers are used, thus generating the same processing time for each job across the different cases. In case rework is performed, then it is important to generate ahead of time the number of reworks that a job will undergo according to the specific rework policy. In the following section, we further detail our approach.

5 Numerical Experiments

In this section, we focus on the performance analysis of the proposed approach in terms of training and testing in regard to two families of problems: (i) serial production lines with random machines (section 5.1), and (ii) serial production lines with machine failures (section 5.2).

Concerning the performance evaluation, we considered the following for training and testing performance evaluation. (i) *Training Performance*: we look into the empirical distribution of the residuals resulting from the models estimation; (ii) *Testing Performance*: we look into the empirical distribution of the estimated prediction error $\widehat{q}_{ip}(C(z)) - q_{ip}^{mm}(C(z))$, where $q_{ip}^{mm}(C(z))$ were computed as in equation (10), but with regression estimated $\hat{\beta}$ values in place of the β's. The $\widehat{q}_{ip}(C(z))$ values were computed using the formula in equation (11) for 100 randomly chosen z states in place of the DOE-specified z_n values. Since $\widehat{q}_{ip}(C(z)) \equiv Y_{iz}$ still contain simulation error, the prediction error characterization is imperfect. For a particular state z_n, let N_{ij} be the *total* (including the jobs in process, if any) number of jobs of type i to be processed by the j^{th} machine. Then the cycle time of job of type of type i' given a state z_n specified by the DOE, can be approximated by the random variable $\tilde{C}_{i'n}$, defined as

$$\tilde{C}_{i'n} = \tau_{iJ} + \sum_{i=1}^{I}\sum_{j=1}^{J} \tau_{iJ} \cdot N_{ij} \sim \mathcal{N}\left(\mu_{i'J} + \sum_{i=1}^{I}\sum_{j=1}^{J} \mu_{iJ} \cdot N_{ij}, (\sigma_{iJ})^2 + \sum_{i=1}^{I}\sum_{j=1}^{J} (\sigma_{iJ})^2 \cdot N_{ij}\right), \forall i. \quad (12)$$

where $\mu_{iJ}, (\sigma_{iJ})^2$ are the mean and variance of the processing time of job i on station J, respectively. This is just an extension of the bound in equation (3) to the more complex system modeled here. The approximation gives simple quantile estimates, to compare with the metamodel-based quantile

estimates.

5.1 Serial Line with Random Processing Times

In this section, we present our empirical analysis. We considered a reference manufacturing system with three stations and three job types to be processed. We assumed a different distribution of the processing time for the different job types. In particular, the average processing time increases with the job type according to the following:

$$E\left[\tau_{ij}\right] = 0.5 + 0.5 \cdot (i-1) \tag{13}$$

where $\tau_{ij} \equiv \tau_i$ is the random processing time for the i^{th} job type. The processing time is exponentially distributed and identical across machines. Machine processing times are often modeled as exponential random variables, for their simple probability properties. Normal or lognormal distributions are also considered in machine scheduling. We use exponential processing time distributions, in part to align with prior work, and in part to provide a relatively difficult ($CV = 1$) case. We considered three job types and three serially connected workstations. We assume that no job release policy would consider a release for job type $i = 1, \ldots, I$ if more than three jobs of that type are waiting to be processed, and this holds true for any of the three workstations.

We ran $10,000$ replications for each experimental condition, using batches of 100 observations for the estimation of the quantiles.

5.1.1 Implementation Details

Experiment Design For the modelled system, the number of states to be considered for any job release decision is all possible combinations of buffer contents (0-3) for each of three job types at each of three stations, plus the job type (or none) of the job in process at each station (two possible values for each job type and each station). This results in approximately 16 million possible states for the modelled system.

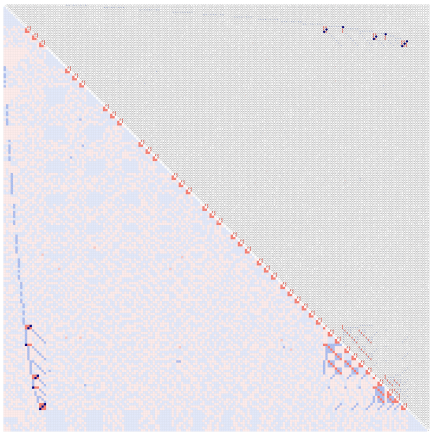


Figure 2: Correlation plot (corrgram) for the proposed design (Friendly, 2002).

The metamodel to be fitted included only linear, interaction and pure quadratic terms, the latter only for the buffer content variables, and interactions excluded for within-station in-process variables. The result was a total of 172 potential terms in the regression model.

We used a fractional factorial design with four levels for buffer variables coded from two columns each of a resolution V fractional factorial design from Sanchez and Sanchez (2005), with columns added consisting of repeated full factorial designs on the in-process model variables. The design was also augmented by additional runs for the completely

full state and for the completely empty state, giving a total of 518 runs. To check the quality of the design, we computed correlations across all model terms. Figure 2 shows the resulting coefficient estimate partial confounding to be very small, excluding within-station in-process job type factors, caused by the coding. These within-station in-process interactions were not included in the candidate model terms.

CRN for Benchmark and Priority Cases. Let us refer to $N_{ij}(z_n)$ as the number of jobs of type i to be processed by station j under the experiment condition z_n being tested. In case of systems with no priority and no rework, the job sequence is determined at the beginning of the simulation. In this case, we generate the processing time for each job at each station, both in buffer and in process, and also processing times for each of the I candidate job types about to be released. We generate a processing times for a number of jobs $N^* = I + \sum_{j=1}^{J}\sum_{i=1}^{I}\left(w_{ij}^{\max}+1\right)$, where $w_{ij}^{\max}$ is the maximum number of parts of type i allowed at station j. For each job, we will generate a number of processing times that depends on the job index. Specifically, if the job in in station j, then $J-j+1$ processing times will be associated to that job.

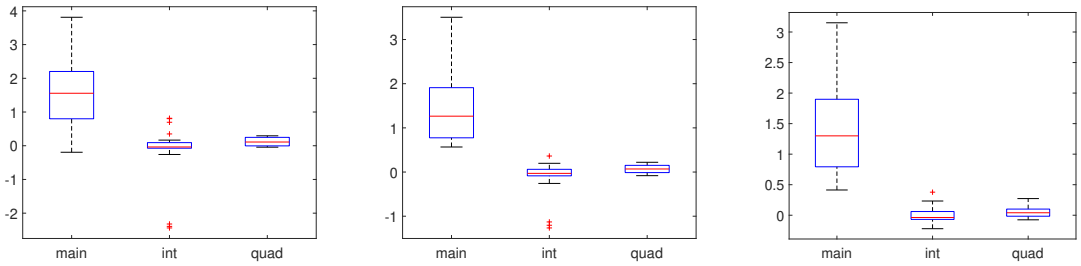
CRN for Rework Cases. Let L represent the $I \times J$ matrix containing the probability that a job of type $i = 1, 2, \ldots, I$ is reworked upon being processed at station $j = 1, \ldots, J$. If a job needs rework, it goes back in queue at the first station $j = 1$, thus undergoing the entire process one more time. In our test case, we set $L(2,2) = 0.2$, meaning that 20% of the jobs of type 2 get reworked after completing station 2. Intuitively, when a rework needs to be performed additional random processing times are required. Specifically, for each part of type 2 starting from station 1 or station 2, the rework count is generated as 1− a geometric random variable with parameter $1 - L(2,2)$. A number of processing times equal to twice the number of reworks (this is due to the fact that the rework implies processing from station 1 *and* station 2) is generated. These process times are appended to the list of processing times already assigned to the part.

In general, we observed that the impact of the CRN, while present, did not strongly influence the performance of the model. We observed an average decrease of the MSE (across part types and across system conditions and setups) associated to the quantile estimation of $\sim 0.1\%$. Clearly such reduction does not reflect in an important difference in terms of prediction error. In the following, we will show the detailed results, in terms of error distribution, that we obtained from the CRN experiment. Similar results held for the case where no common random numbers were adopted and the same conclusions can be drawn, so we omit those results in the interest of space.

5.1.2 Results for Serial Lines with Multiple Job Types

We used a stepwise regression technique to fit to the observed quantile response over the 518 experiment conditions, for each job type. The resulting models had all linear and quadratic terms included, as well as many two-factor interaction terms. A total of $(70, 69, 67)$ of 172 possible terms for a full quadratic model (with intercept) were included for the three part types, respectively. The resulting R^2 was larger than 0.99 for all the part types and Figure 3 shows a boxplot of coefficient magnitudes for linear, quadratic, and interaction terms for the metamodel for each job

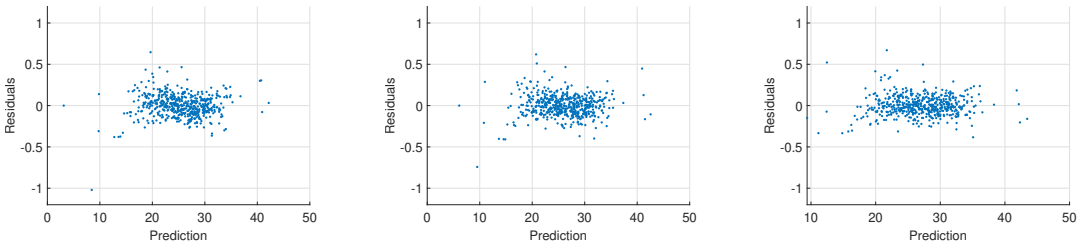
type. Because the variables were scaled, the magnitudes relate directly to the strength of the impact. We can observe the prevailing effect of the linear component, nevertheless, the interaction and quadratic components, while lower in terms of effect, all had associated practically 0 p-values.



(a) Job Type 1. (b) Job Type 2. (c) Job Type 3.

Figure 3: Boxplot for the model coefficients, 95% quantile, base model.

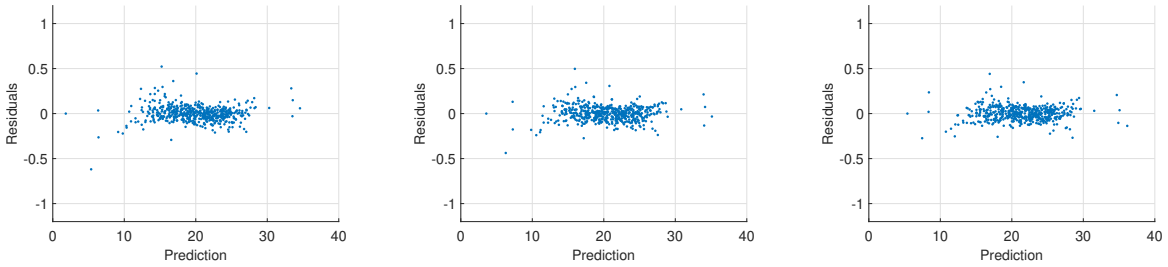
Figure 4 reports the residuals obtained as a result of model training for each job type. We can observe that the values are quite symmetrically distributed around 0. Most of the residuals lay within the $[-0.2, 0.2]$ band, and we observe no particular difference with respect to the job type. The model tends to overestimate the completion time for a small number of states corresponding to an empty or nearly empty system, especially for the "faster" part types (i.e., part type 1 and 2).



(a) Residuals Job Type 1, 95% quantile. (b) Residuals Job Type 2, 95% quantile. (c) Residuals Job Type 3, 95% quantile.

Figure 4: Residuals for the 95% quantile, base case.

We performed the analysis for several values of the quantile and we report the results for the 70% quantile below (Figure 5). As expected, the training performance of the state-based metamodel is better for smaller quantiles mainly due to the lower variance of the estimator.



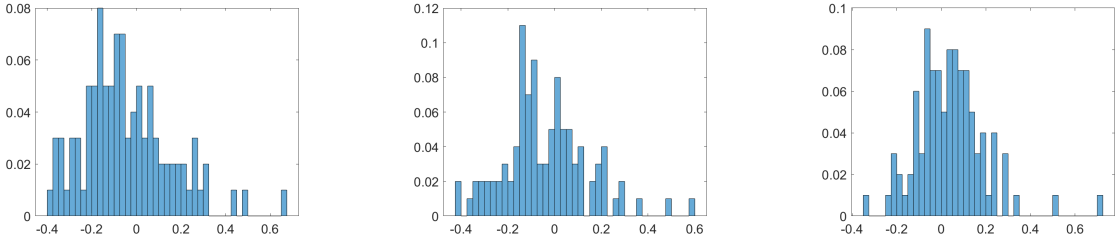
(a) Residuals Job Type 1, 70% quantile.

(b) Residuals Job Type 2, 70% quantile.

(c) Residuals Job Type 3, 70% quantile.

Figure 5: Residuals for the 70% quantile, base case.

Figure 6 shows the prediction error histograms for the different part type metamodels. Here error is defined as the observed quantile via simulation minus the metamodel-predicted quantile, $\widehat{q}_{ip}(C(z)) - q_{ip}^{mm}(C(z))$, at 100 randomly selected states not used in the experiment design. We see that most of the error values lay within the $[-0.2, 0.2]$ interval, similar to the residuals observed in the fitting process. Thus there is no evidence of overfitting of the metamodel.



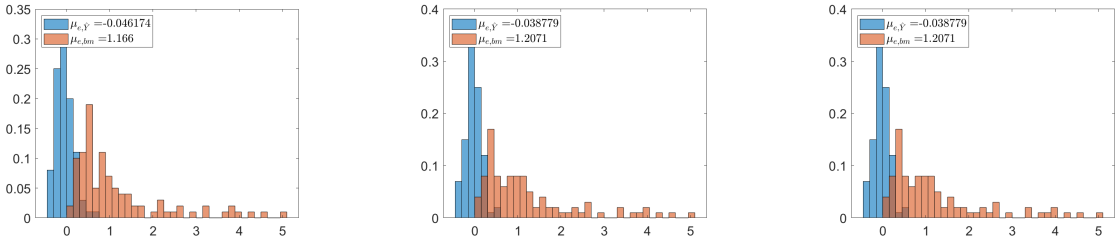
(a) Job type 1.

(b) Job type 2.

(c) Job type 3.

Figure 6: Metamodel quantile prediction error across 100 independent test conditions, for the 95% quantile, base case.

In order to benchmark our prediction, we use the quantile of the normal distribution bound in equation (3) to the more complex system modeled here. Figure 7 compares metamodel prediction error with the error resulting from the use of the normal approximation quantile predictor (following from equation (12)). As previously, the error is defined as the true value - predicted value (estimated via direct simulation of that state). While the state-based metamodel prediction error shows a concentration around 0, the error from the normal approximation appears to be skewed towards the positive values, and much more dispersed. These observations are also valid for the other types of systems. This makes sense, in the base system, the normal approximation arguably represents a lower bound to the cycle time.

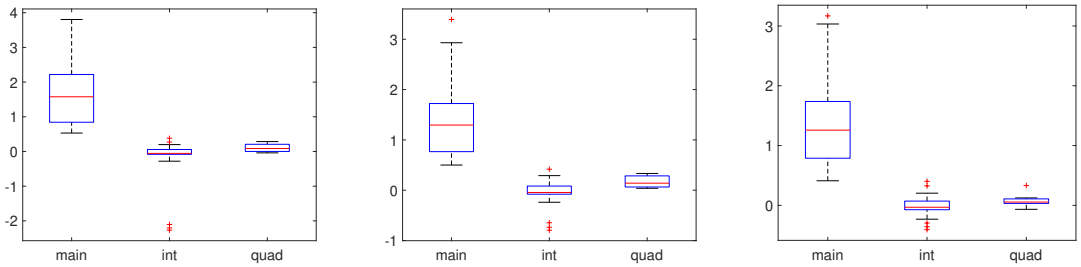


(a) Job type 1. (b) Job type 2. (c) Job type 3.

Figure 7: Metamodel quantile prediction error across 100 independent test conditions, for the 95% quantile compared with the normal approximation prediction error, base case.

5.1.3 Results for Serial Lines with Multiple Job Types and rework

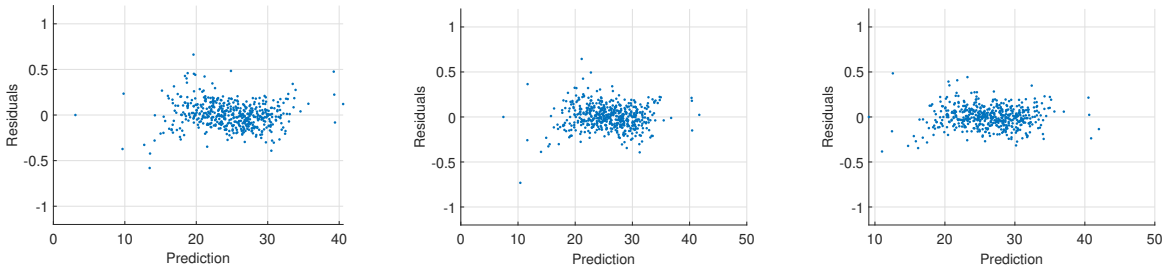
A total of $(63, 67, 73)$ of 172 possible terms for a full quadratic model (with intercept) were included for the three part types, respectively. The resulting R^2 was larger than 0.99 for all the part types and Figure 8 shows the boxplot of coefficients and similar conclusion can be drawn compared to the benchmark case.



(a) Job Type 1. (b) Job Type 2. (c) Job Type 3.

Figure 8: Boxplot for the model coefficients, 95% quantile, rework case.

Figure 9-10, give similar results to the benchmark case: "Serial Lines with Multiple Job Types." The residuals largely concentrate within $[-0.2, 0.2]$. The largest residuals again correspond to overestimation for nearly empty states, particularly for type 2 completion time estimation, where type 2 is the job type subject to rework (Figure 9b).



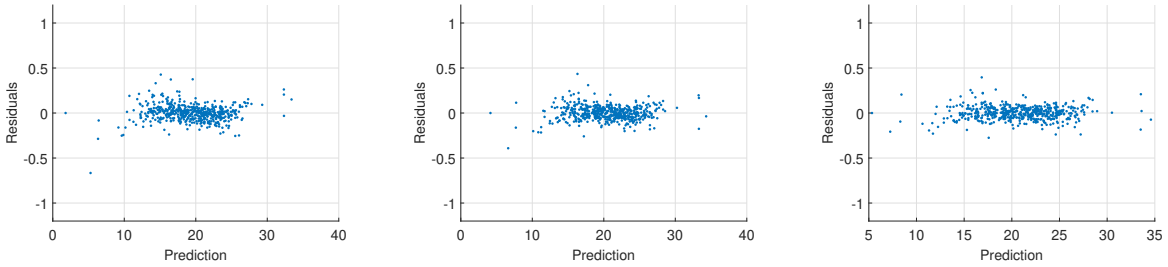
(a) Residuals Job Type 1, 95% quantile.

(b) Residuals Job Type 2, 95% quantile.

(c) Residuals Job Type 3, 95% quantile.

Figure 9: Residuals for the 95% quantile, rework case.

Also for the rework case, we observe that the residuals for the 70% quantile metamodel concentrate more around 0 line compared with residuals from the the 95% quantile estimation metamodel.



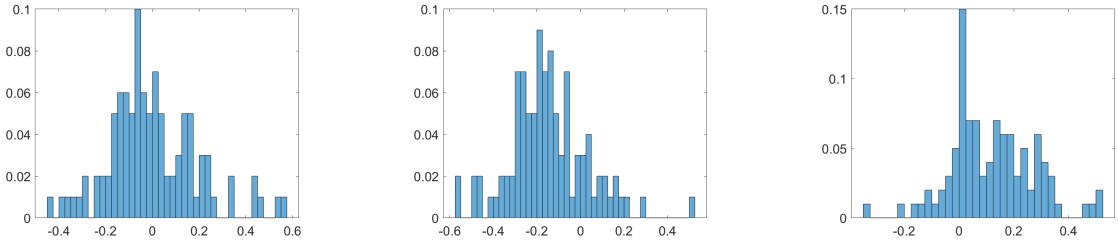
(a) Residuals Job Type 1, 70% quantile.

(b) Residuals Job Type 2, 70% quantile.

(c) Residuals Job Type 3, 70% quantile.

Figure 10: Residuals for the 70% quantile, rework case.

Figures 11-12 show results similar to the benchmark case. However, we observe a slight tendency towards overestimation for type 2 (rework job) and underestimation for job type 3 (the slowest job), an expected effect of the priority on the model fidelity.



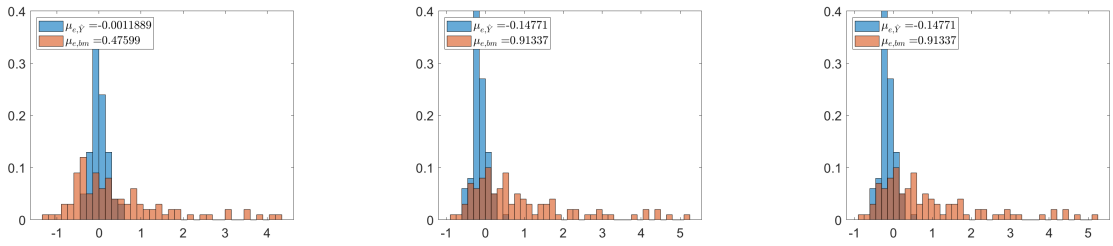
(a) Job type 1.

(b) Job type 2.

(c) Job type 3.

Figure 11: Metamodel quantile prediction error across 100 independent test conditions, for the 95% quantile, rework case.

Figure 12, shows that the metamodel quantile predictor has lower error compared to the normal approximation.

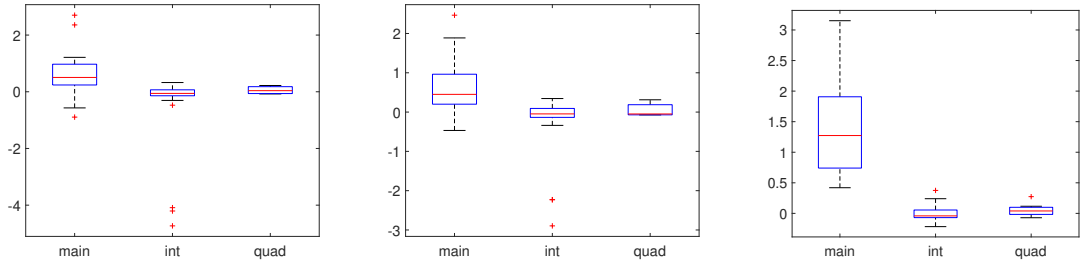


(a) Job type 1. (b) Job type 2. (c) Job type 3.

Figure 12: Metamodel quantile prediction error across 100 independent test conditions, for the 95% quantile compared with the normal approximation, rework case.

5.1.4 Results for Serial Lines with Multiple Job Types and Priority

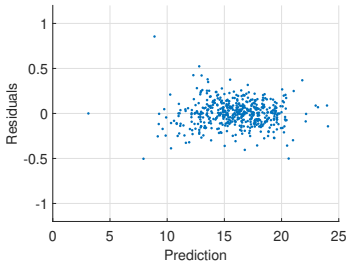
A total of $(75, 59, 68)$ of 172 possible terms for a full quadratic model (with intercept) were included for the three part types, respectively. The resulting R^2 was larger than 0.99 for all the part types and Figure 13 shows the boxplot of coefficients and similar conclusion can be drawn compared to the previous cases. However, the increase in difficulty in modeling the priority case is also reflected by a large number of extreme coefficient values in the negative side. This makes sense since the priority shortens the processing times for the first and second job type.



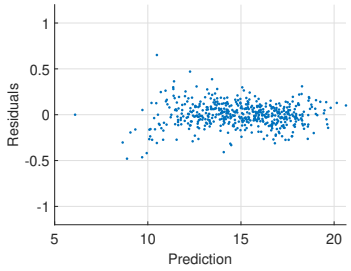
(a) Job Type 1. (b) Job Type 2. (c) Job Type 3.

Figure 13: Boxplot for the model coefficients, 95% quantile, priority case.

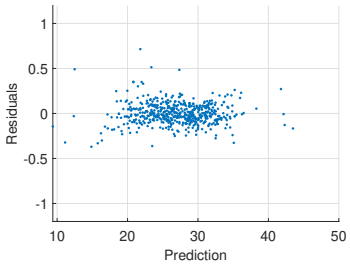
Priority naturally decreases the completion time, for jobs with higher priority. This appears to add some dispersion to the metamodel residuals (Figure 14). This is expected as, in general, the case of priority service discipline is the hardest to model. Nonetheless, we observe a concentration of the residuals within an interval a little larger than $[-0.2, 0.2]$.



(a) Residuals Job Type 1, 95% quantile.

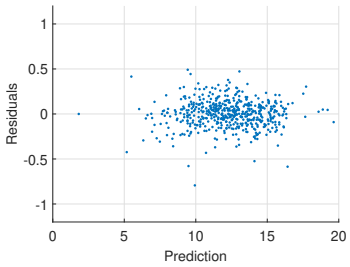


(b) Residuals Job Type 2, 95% quantile.

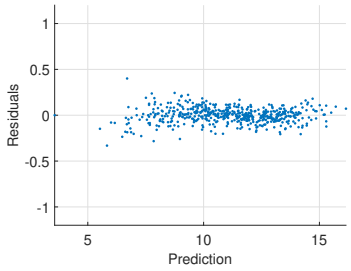


(c) Residuals Job Type 3, 95% quantile.

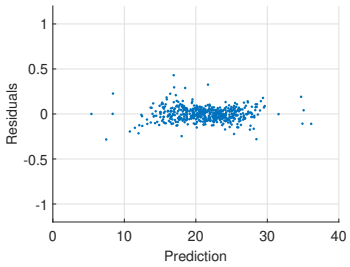
Figure 14: Residuals for the 95% quantile, priority case.



(a) Residuals Job Type 1, 70% quantile.



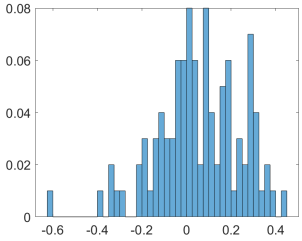
(b) Residuals Job Type 2, 70% quantile.



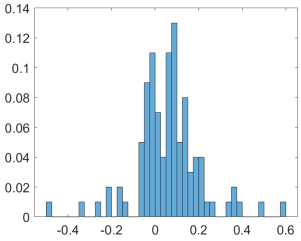
(c) Residuals Job Type 3, 70% quantile.

Figure 15: Metamodel residuals for the 70% quantile, priority case.

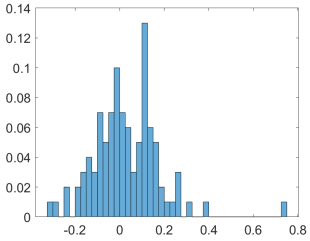
Figure 16 shows good predictive performance. Figure 17 shows that the priority case results in the largest error for the normal approximation quantile predictor, as predictable, the cycle time of type 1 and 2 (higher priority) are overestimated by the approximate model, while the lowest priority job type (type 3) has the cycle time largely underestimated.



(a) Job type 1.

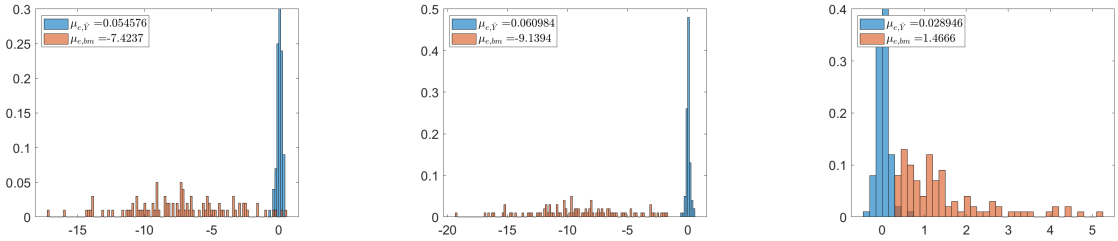


(b) Job type 2.



(c) Job type 3.

Figure 16: Absolute quantile prediction error across 100 independent test conditions, for the 95% quantile, priority case.

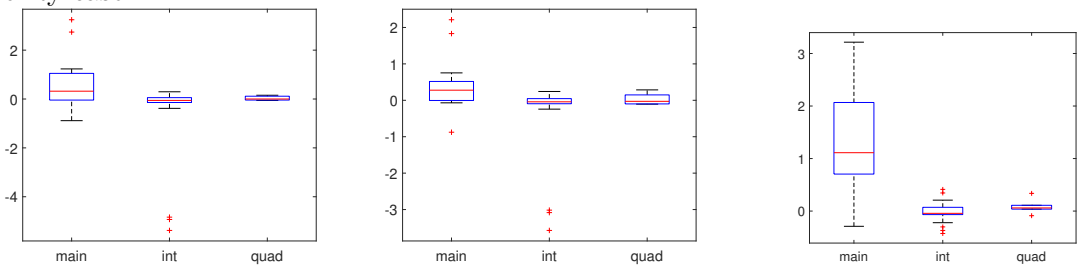


(a) Job type 1. (b) Job type 2. (c) Job type 3.

Figure 17: Absolute quantile prediction error across 100 independent test conditions, for the 95% quantile compared with the normal approximation, priority case.

5.1.5 Results for Serial Lines with Multiple Job Types, Rework and Priority

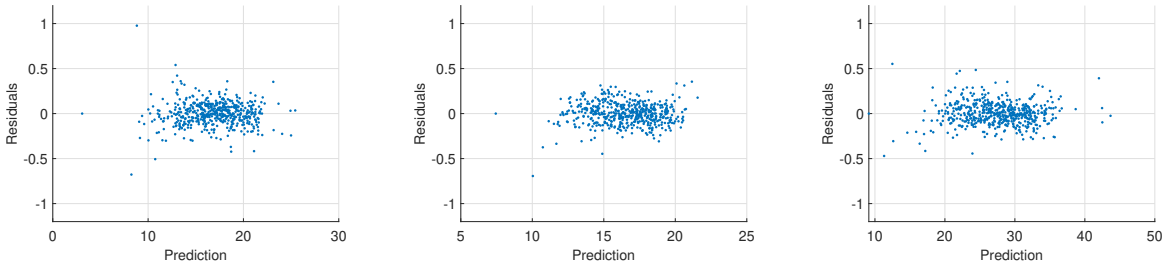
A total of $(75, 67, 74)$ of 172 possible terms for a full quadratic model (with intercept) were included for the three part types, respectively. The resulting R^2 was larger than 0.99 for all the part types and Figure 18 shows the boxplot of coefficients and similar conclusion can be drawn compared to the priority case.



(a) Job Type 1. (b) Job Type 2. (c) Job Type 3.

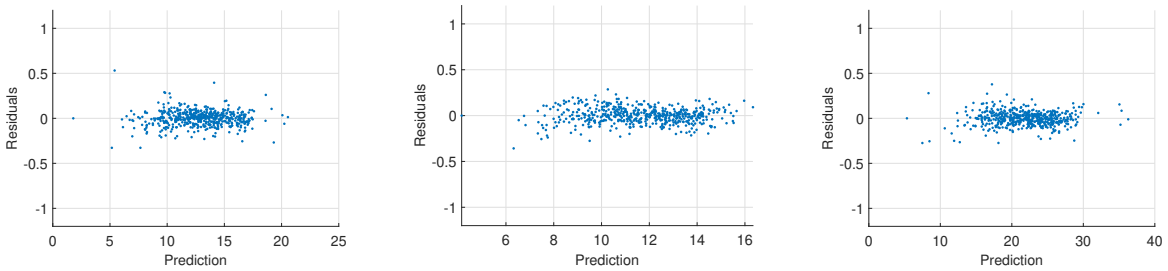
Figure 18: Boxplot for the model coefficients, 95% quantile, rework and priority case.

The final scenario included both rework and priority mechanisms. Figures 19-20 show similar results to the priority case without rework.



(a) Residuals Job Type 1, 95% quantile. (b) Residuals Job Type 2, 95% quantile. (c) Residuals Job Type 3, 95% quantile.

Figure 19: Metamodel residuals for the 95% quantile, rework and priority case.



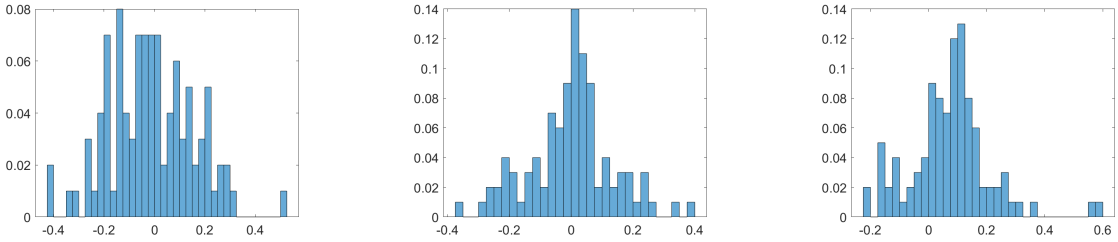
(a) Residuals Job Type 1, 70% quantile.

(b) Residuals Job Type 2, 70% quantile.

(c) Residuals Job Type 3, 70% quantile.

Figure 20: Metamodel residuals for the 70% quantile, rework and priority case.

Figures 21-22 show very similar performance to the priority example. Again, the normal approximation quantile predictor performs poorly when a priority service discipline is present, but a quadratic state-based metamodel still performs well.

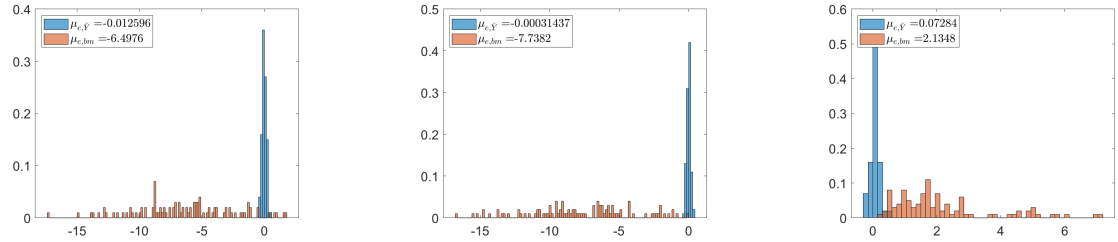


(a) Job type 1.

(b) Job type 2.

(c) Job type 3.

Figure 21: Absolute quantile prediction error across 100 independent test conditions, for the 95% quantile, rework and priority case.



(a) Job type 1.

(b) Job type 2.

(c) Job type 3.

Figure 22: Absolute quantile prediction error across 100 independent test conditions, for the 95% quantile compared with the normal approximation, rework and priority case.

5.2 Serial Line with Multiple Failure Modes and Random Processing Times

For the second set of experiments, we considered the case of machines potentially affected by failures. We considered uniformly distributed processing times, and the parameters are reported in Table 1. Concerning the failures, we considered two failure types, and we also considered both exponential as well as uniform time between failure and time to repair. For the two failure types we considered a first class of frequent, but fast to repair failures and one class of rare failures that require long repair times. Specifically, we considered the first failure mode (frequent) to be uniform

with time between failures to be uniform with bounds $[15, 25]$, and uniform repair time with limits $[2.5, 3.5]$. For the rare mode, we considered an exponential time between failures with mean 60 and time to repair exponential with mean 35. In this set of experiments we do not consider the common random number implementation. Also, we used a small input design with 112 conditions necessary to estimate first order, second order and a subset of the interactions.

Table 1: Parameters for uniform processing times

Part Type	Machine	p^ℓ	p^u
1	1	5	7.5
	2	7.5	10
	3	7.5	8.5
2	1	4.5	6.75
	2	6.75	9
	3	6.75	7.65
3	1	5.5	8.25
	2	8.25	11
	3	8.25	9.35

5.2.1 Implementation Details

In order to reduce the computational time requirements, we replicated each experiment 1000 times and used a batch size of 100 outcomes to produce our quantile estimations.

Design of Experiment. For the cases with failure we considered a slightly different design to account for the possibility of the machines to be failed at the simulation start. The factors are described in Table 2. We can observe we have 9 factors for the queue level, 3 factors for the time, 6 factors for the failure mode and 3 factors for the part type loaded onto the machine resulting in 21 factors.In the model, we considered first order effect for the 9 factors of queue level (9 terms); the interaction of for these factors was also considered (81 terms); the interaction between time factor and part on machine factor (9 terms); the interaction between time and failure (18 terms); finally we included the quadratic effect of the queue and time (12 terms). This results in a model with potentially 129 terms. Among the new factors we defined $t_j, j = 1, \ldots, J$ to account for the accumulated operating time (refer to section 4.3 for details). Specifically, the value of t_j was used to determine p_{ij} or r_{kj} needed for the model (eqn.(8)). We set $T^\ell = -8, T^u = 2$ with the idea to favour conditions with positive accumulated processing time as compared to scenarios with failed machines. Since in the simulation environment we generate processing times and repair times, for negative t_j, the value $-100T_{ij}/8\%$ was used as the percentage of the operating time already accumulated by the job being processed. If, instead, $t_j > 0$ then $100T_{ij}/2\%$ was used as the percentage of the repair time already accumulated by the machine being maintained. The design we used for this experiment contained 112 conditions.

Table 2: Factors used for the model for line with failures

Name	Type	Lower bound	Upper bound	Description
Q_{11}	discrete	0	3	Level of job type 1 queue machine 1
⋮	⋮	⋮	⋮	⋮
Q_{33}	discrete	0	3	Level of job type 3 queue machine 3
T_1	continuous	-8	2	relative time machine has been operating (if t<0) or under repair (t>0)
T_2	continuous	-8	2	relative time machine has been operating (if t<0) or under repair (t>0)
T_3	continuous	-8	2	relative time machine has been operating (if t<0) or under repair (t>0)
F_{11}	binary	0	1	if failure mode 1 is active on machine 1
F_{21}	binary	0	1	if failure mode 2 is active on machine 1
⋮	⋮	⋮	⋮	⋮
F_{23}	binary	0	1	if failure mode 2 is active on machine 3
O_1	discrete	0	3	job type loaded on machine 1
O_2	discrete	0	3	job type loaded on machine 2
O_3	discrete	0	3	job type loaded on machine 3

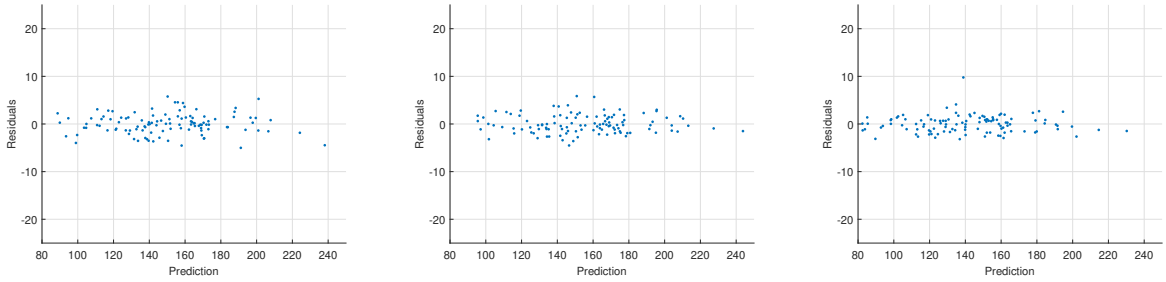
5.2.2 Results

Below we provide the results following the same rationale as in the previous sections, and we focus on residuals, absolute error distribution and comparison against the benchmark for the 95% quantile. The models generally had large R^2-adjusted. Also for the failure case, we will be looking into the behavior of the residuals resulting from model training, the distribution of the absolute error in testing, and compare the performance with the normal approximation.

Table 3: R^2 and R^2-adjusted (%) coefficients associated to the models for the 95% quantile.

Job Type	Case	R^2	R^2-adj
1	bm	99.49	99.36
	rw	99.33	99.14
	pt	96.02	95.45
	rwpt	95.68	95.00
2	bm	99.59	99.48
	rw	98.49	98.22
	pt	88.83	87.84
	rwpt	88.30	86.88
3	bm	99.64	99.57
	rw	99.58	99.59
	pt	99.53	99.40
	rwpt	98.79	98.43

5.2.3 Results for Serial Lines with Multiple Job Types



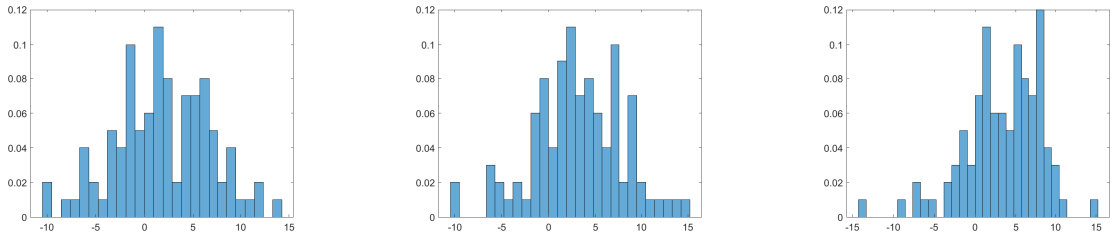
(a) Residuals Job Type 1, 95% quantile.

(b) Residuals Job Type 2, 95% quantile.

(c) Residuals Job Type 3, 95% quantile.

Figure 23: Metamodel residuals for the 95% quantile, base case.

While, from Figure 23 we observe how the residuals, while being centered around 0, appear larger than the cases without failure. Nonetheless, no particular pattern in the data can be observed.



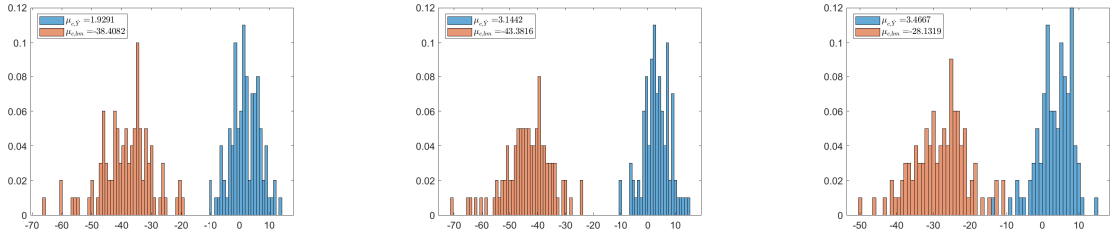
(a) Job type 1.

(b) Job type 2.

(c) Job type 3.

Figure 24: Absolute quantile prediction error across 100 independent test conditions, for the 95% quantile, benchmark case.

Observing Figure 24, we notice how while most of the error in within the $[-5, 5]$ band, considering failure definitely impacts the quality of the estimation.



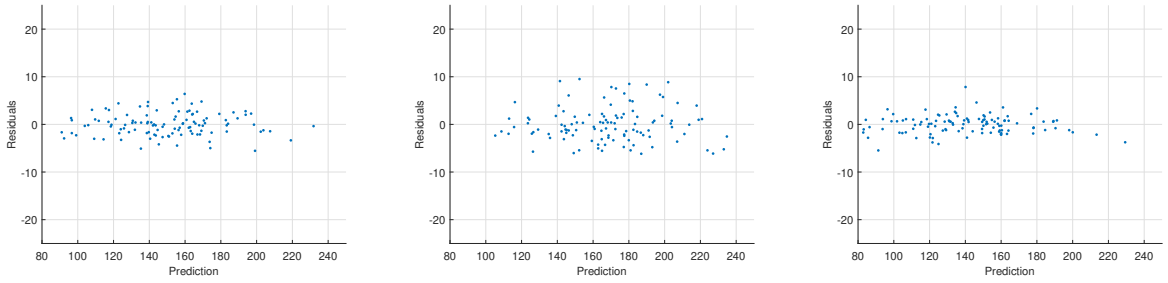
(a) Job type 1. (b) Job type 2. (c) Job type 3.

Figure 25: Absolute quantile prediction error across 100 independent test conditions, for the 95% quantile compared with the normal approximation, benchmark case.

Nonetheless, as shown in Figure 25 it is clear that our model is still an order of magnitude better than the normal approximation.

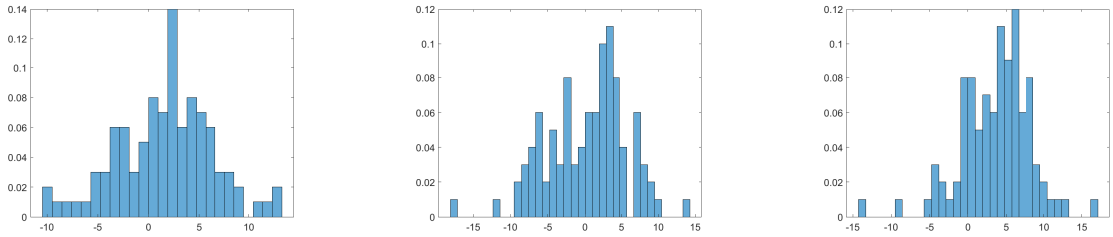
5.2.4 Results for Serial Lines with Multiple Job Types and rework

From the results, we can see that rework, affecting job type 2 in particular, greatly damages the performance of the normal approximation while our approach is quite consistent with error mostly in the $[-5, 5]$ band.



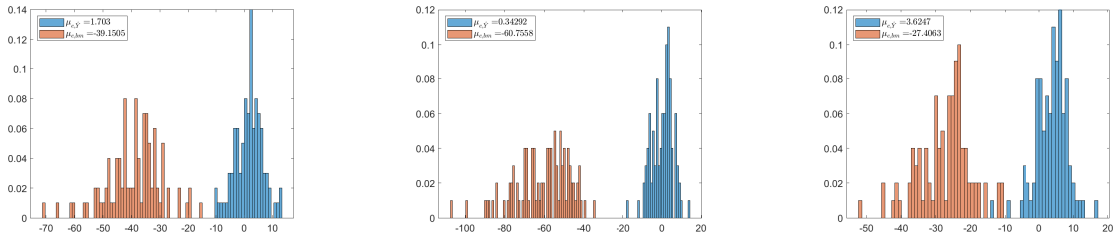
(a) Residuals Job Type 1, 95% quantile. (b) Residuals Job Type 2, 95% quantile. (c) Residuals Job Type 3, 95% quantile.

Figure 26: Metamodel residuals for the 95% quantile, rework case.



(a) Job type 1. (b) Job type 2. (c) Job type 3.

Figure 27: Absolute quantile prediction error across 100 independent test conditions, for the 95% quantile, benchmark case.

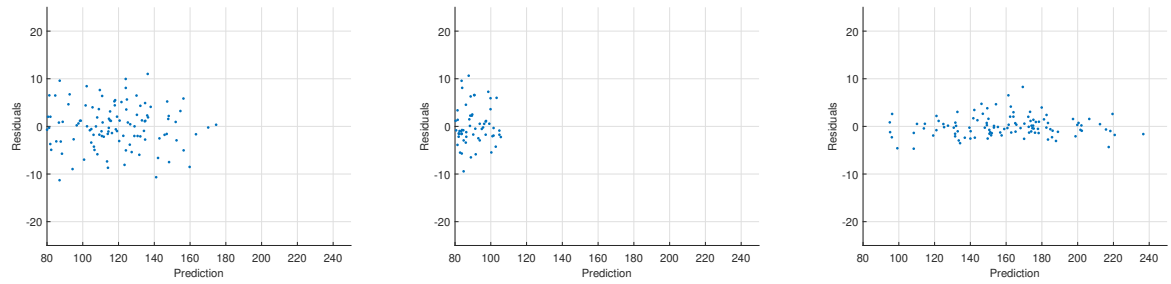


(a) Job type 1. (b) Job type 2. (c) Job type 3.

Figure 28: Absolute quantile prediction error across 100 independent test conditions, for the 95% quantile compared with the normal approximation, rework case.

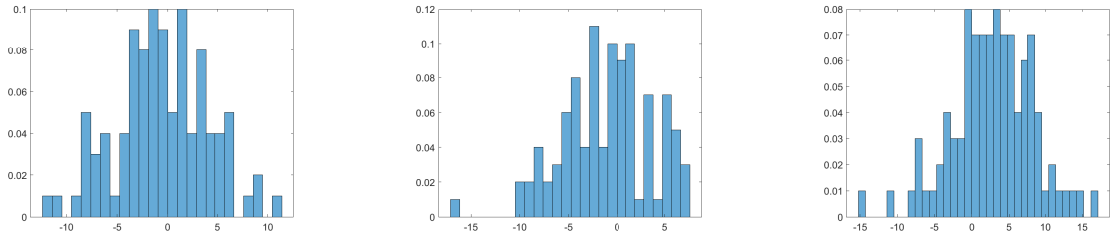
5.2.5 Results for Serial Lines with Multiple Job Types and priority

Priority, as in the non-failure case, is very critical, and also in this case we clearly see how the highest (type 2) and lowest (type 3) priority jobs are affected. Also in this case, the performance of the normal approximation is further deteriorated.



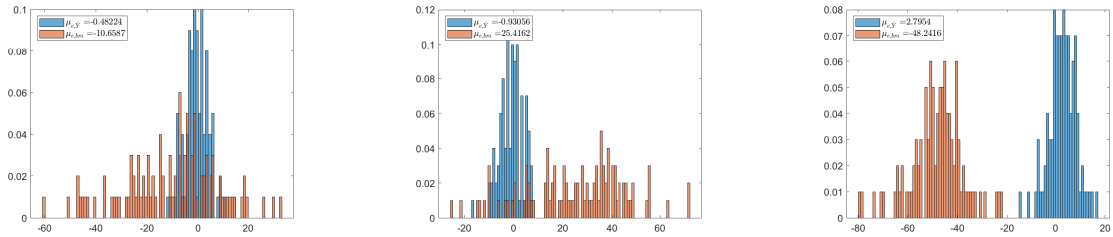
(a) Residuals Job Type 1, 95% quantile. (b) Residuals Job Type 2, 95% quantile. (c) Residuals Job Type 3, 95% quantile.

Figure 29: Metamodel residuals for the 95% quantile, priority case.



(a) Job type 1. (b) Job type 2. (c) Job type 3.

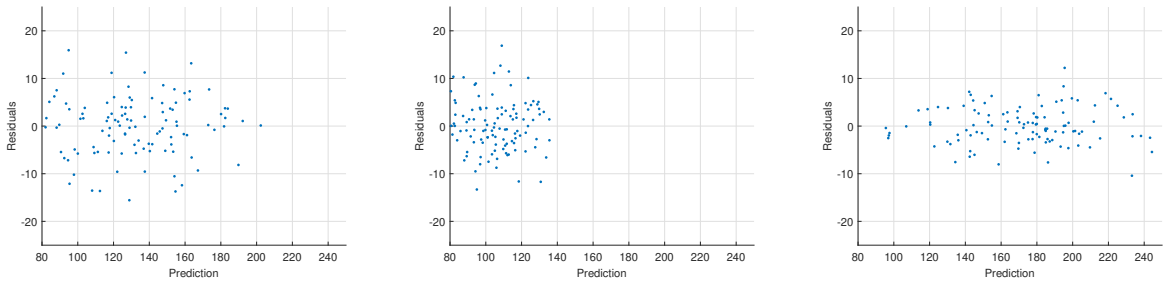
Figure 30: Absolute quantile prediction error across 100 independent test conditions, for the 95% quantile, priority case.



(a) Job type 1. (b) Job type 2. (c) Job type 3.

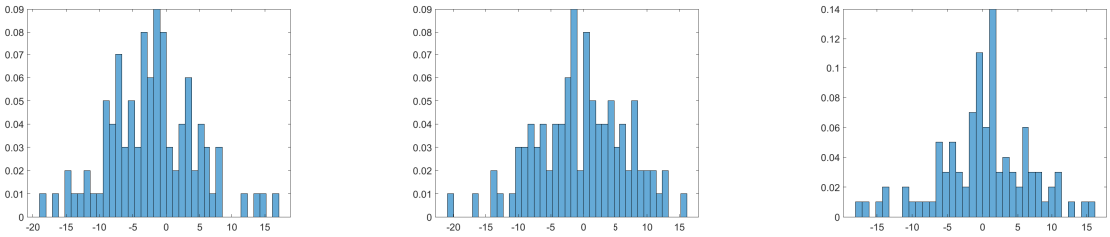
Figure 31: Absolute quantile prediction error across 100 independent test conditions, for the 95% quantile compared with the normal approximation, priority case.

5.2.6 Results for Serial Lines with Multiple Job Types Rework and Priority



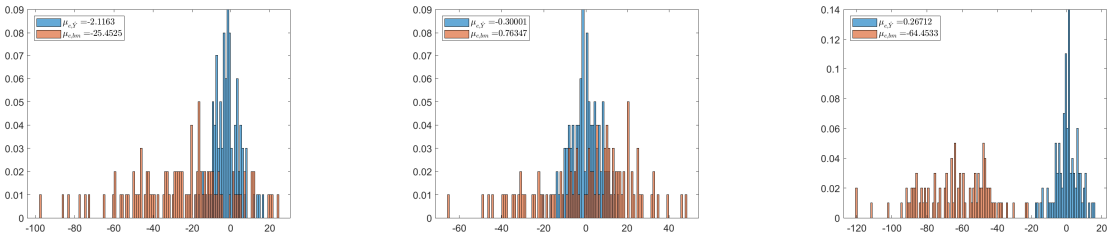
(a) Residuals Job Type 1, 95% quantile. (b) Residuals Job Type 2, 95% quantile. (c) Residuals Job Type 3, 95% quantile.

Figure 32: Metamodel residuals for the 95% quantile, priority and rework case.



(a) Job type 1. (b) Job type 2. (c) Job type 3.

Figure 33: Absolute quantile prediction error across 100 independent test conditions, for the 95% quantile, priority and rework case.



(a) Job type 1. (b) Job type 2. (c) Job type 3.

Figure 34: Absolute quantile prediction error across 100 independent test conditions, for the 95% quantile compared with the normal approximation, priority and rework case.

5.3 Summary Observations

Below, we show the relative error $|q_{ip}^{mm}(C(z)) - \widehat{q}_{ip}(C(z))|/\widehat{q}_{ip}(C(z))$ averaged over 100 random states z for the case with and without failures.

Table 4: Average absolute relative prediction error ($|q_{ip}^{mm}(C(z)) - \widehat{q}_{ip}(C(z))|/\widehat{q}_{ip}(C(z))$) across 100 random conditions ("bm" refers to the benchmark, "rw" rework, "pt" priority, and "rwpt" priority and rework system, respectively).

Job Type	Case	No Failure				Failure			
		0.7	0.8	0.9	0.95	0.7	0.8	0.9	0.95
1	bm	0.004	0.005	0.006	0.007	0.036	0.035	0.032	0.029
	rw	0.004	0.005	0.006	0.006	0.034	0.035	0.031	0.028
	pt	0.013	0.009	0.009	0.010	0.045	0.042	0.037	0.031
	rwpt	0.008	0.008	0.008	0.009	0.051	0.051	0.042	0.042
2	bm	0.004	0.005	0.005	0.006	0.036	0.035	0.030	0.031
	rw	0.006	0.007	0.006	0.008	0.030	0.027	0.024	0.026
	pt	0.007	0.008	0.009	0.009	0.062	0.059	0.047	0.044
	rwpt	0.008	0.007	0.007	0.006	0.056	0.060	0.060	0.052
3	bm	0.004	0.004	0.005	0.005	0.037	0.036	0.035	0.036
	rw	0.004	0.004	0.005	0.006	0.035	0.036	0.035	0.036
	pt	0.004	0.004	0.005	0.005	0.032	0.034	0.033	0.032
	rwpt	0.004	0.004	0.004	0.004	0.029	0.028	0.027	0.029

Table 4 shows that the relative error achieved for the state-based metamodel quantile estimates to be far less than the 4%-15% relative error in previous metamodel quantile estimation procedures (Bekki et al., 2014a,b; Chen and Kim, 2013) (nonetheless we should highlight that the failure contributes in terms of an order of magnitude in the relative error). The state-based metamodel for cycle time benefits from being fitted to short, simple terminating simulations. It does not need to estimate steady state distributions that require long simulation times for systems with high utilization (as expected in a manufacturing setting).

6 Conclusions

We have presented a state-based polynomial metamodel to predict the completion time of jobs for serial production systems. The empirical analysis gave practical guidelines to develop a design of experiment that minimizes confounding, and a way to implement common random numbers in the attempt to reduce the variance associated to the observations.

There are many potential directions for further development of state-based metamodels for real-time cycle time quantile prediction. Given the satisfactory performance in cases with priority, future research should be performed exploring the more general context of preemptive policies. The satisfactory performance in rework suggests that the modeling approach is effective for both flow shop and job shop systems. Another important aspect for future investigation is the applicability of the method in practical settings. The experiments in small settings indicate that a full quadratic

polynomial metamodel provides very good fit for the 256-state system in (Pedrielli and Barton, 2019), and for the larger example explored here, with in some cases more than two million states.

Note that, while the number of states grows exponentially with system size $s = |\{\text{state variables}\}|$, the number of terms in the regression grows only polynomially, as $(s+1)(s+2)/2$. The behavior observed in our computational studies suggests that the experiment design need be only a small multiple, say on the order of 2-3 times the number of terms in the regression. For large numbers of terms in the quadratic metamodel, the degrees of freedom for error would be sufficient for designs only slightly larger than the number of model terms, but lack of fit would be more difficult to assess. However, our experience has shown quadratic regression metamodels for cycle time quantiles to provide high fidelity, with R^2 values greater than .99, and so lack of fit may not me a major concern.

While the growth in metamodel and experiment size is small relative to the growth in the number of states, it can still be substantial in a practical setting with a large number of machines. For example, for three part types, up to three in buffers of each type at each station, and ten machining stations, the number of metamodel state variables (assuming our coding for the in-process job status) is 60, and so there would be 1,891 terms in the full quadratic regression metamodel, requiring 2,000 or more conditions in the DOE for full estimation of the DOE. But it is important to remember that these are relatively short simulations, only running until the newly released job completes. Further, in larger systems such as this, some station-station interactions may be unimportant (e.g., station 1, part type 1 buffer contents x station 10 part type 1 buffer contents). In that case, a reduced design confounding some two-factor interactions might still provide good predictive ability across all possible states.

For very large systems, very large experiment designs for fitting quadratic models will be required. The literature on large experimental designs falls in two categories; nearly orthogonal Latin hypercube (NOLH) arrays, and large fractional factorial designs. Hernandez et al. (2012) propose a mixed integer program formulation that can be applied to construct NOLH designs of arbitrary size, and so a NOLH of 2,000 runs is feasible, though technically challenging. Other NOLH methods for large designs (Cioppa and Lucas, 2007; Lin et al., 2009) might also be employed, but Latin hypercube designs do not provide optimal structure for simple quadratic regression models - they tend to be far from D-optimal. It may be more appropriate to augment a highly fractionated factorial design with runs chosen allow efficient estimation of pure quadratic terms. In addition to the designs mentioned in (Sanchez and Sanchez, 2005), algorithms for construction of large fractional factorial designs are presented by El-Boujdaini and Driouchi (2020); Xu (2009). In particular, Table 15 in (Xu, 2009) provides a fractional factorial design for up to 160 variables in 512 runs. For our application, it is most likely that the number of runs needed would be required to exceed the number of quadratic model terms to avoid confounding.

Cycle time quantile forecasts based on the real-time state of the system are critical to hedging and other real-time job release control policies. Unlike prediction of steady-state cycle time as a function of system design, state dependent metamodels have simple and interpretable structure

and can achieve high fidelity efficiently. The combinatorial explosion of possible system states does not inhibit the use of a state-based metamodeling strategy. In the numerical study presented here, the fraction of all possible states that required simulation was approximately 0.0003, about three hundredths of one percent. The resulting predictive error was typically within ±4% of the mean processing time of a single job at a single station, a very small error. State-based metamodeling for cycle time quantile prediction provides a new way to enable real-time hedging and control policies.

References

Angius, A., M. Colledani, and A. Horvath. 2018. "Lead-Time-Oriented Production Control Policies in Two-Machine Production Lines". *IISE Transactions* 50 (3): 178–190.

Banks, J., J. S. Carson, B. L. Nelson, and D. M. Nicol. 2013. *Discrete-event system simulation: Pearson new international edition*. Pearson Higher Ed.

Barton, R. R. 2015. "Tutorial: Simulation Metamodeling". In *Proceedings of the 2015 Winter Simulation Conference*, 1765–1779. Piscataway, New Jersey: IEEE Press.

Bassett Jr, G. W., and R. Koenker. 2017. "A Quantile Regression Memoir". In *Handbook of Quantile Regression*, 23–26. Chapman and Hall/CRC.

Batur, D., J. M. Bekki, and X. Chen. 2018a, January. "Quantile regression metamodeling: Toward improved responsiveness in the high-tech electronics manufacturing industry". *European Journal of Operational Research* 264 (1): 212–224.

Batur, D., J. M. Bekki, and X. Chen. 2018b. "Quantile Regression Metamodeling: Toward Improved Responsiveness in the High-Tech Electronics Manufacturing Industry". *European Journal of Operational Research* 264 (1): 212–224.

Bekki, J. M., X. Chen, and D. Batur. 2014a. "Steady-state Quantile Parameter Estimation: An Empirical Comparison of Stochastic Kriging and Quantile Regression". In *Proceedings of the 2014 Winter Simulation Conference*, WSC '14, 3880–3891. Piscataway, NJ, USA: IEEE Press.

Bekki, J. M., X. Chen, and D. Batur. 2014b. "Steady-State Quantile Parameter Estimation: an Empirical Comparison of Stochastic Kriging and Quantile Regression". In *Proceedings of the 2014 winter simulation conference*, 3880–3891. Piscataway, New Jersey: IEEE Press.

Bekki, J. M., J. W. Fowler, G. T. Mackulak, and B. L. Nelson. 2009. "Indirect cycle time quantile estimation using the Cornish–Fisher expansion". *IIE Transactions* 42 (1): 31–44.

Blanning, R. W. 1974, August. "The Sources and Uses of Sensitivity Information". *Interfaces* 4 (4): 32–38.

Bureau, M., S. Dauzère-Pérès, and Y. Mati. 2006. "Scheduling challenges and approaches in semiconductor manufacturing". *IFAC Proceedings Volumes* 39 (3): 739–744.

Chen, X., and K.-K. Kim. 2013. "Building Metamodels for Quantile-Based Measures Using Sectioning". In *2013 Winter Simulations Conference (WSC)*, 521–532. Piscataway, New Jersey: IEEE Press.

Cioppa, T. M., and T. W. Lucas. 2007, February. "Efficient Nearly Orthogonal and Space-Filling Latin Hypercubes". *Technometrics* 49 (1): 45–55. Publisher: Taylor & Francis.

Colledani, M., and S. B. Gershwin. 2017. "Dynamic Lead Time Based Control Point Policy for Multi-Stage Manufacturing Systems". In *Proceedings of the 11-th Conference on Sotchastic Models of Manufacturing and Service Operations SMMSO 2017*, 19–26. Milan, Italy: Institute of Industrial Technology and Automation.

Durmusoglu, M. B., and C. Aglan. 2017. "CONWIP and Hybrid CONWIP Production Control Systems: A Literature Review". In *Production Management*, 123–140. Productivity Press.

El-Boujdaini, B., and D. Driouchi. 2020, January. "An Improved Algorithm for Constructing Large Fractional Factorial Designs". *International Journal of Innovative Technology and Exploring Engineering* 9 (3): 30–33.

Friendly, M. 2002, January. "Corrgrams: Exploratory Displays for Correlation Matrices". *The American Statistician* 56 (4): 316–324.

Geoffrion, A. M., and G. W. Graves. 1976. "Scheduling parallel production lines with changeover costs: Practical application of a quadratic assignment/LP approach". *Operations Research* 24 (4): 595–610.

Gershwin, S. B. 2000. "Design and Operation of Manufacturing Systems: the Control-Point Policy". *IIE Transactions* 32 (10): 891–906.

Gordon, V. 1993. "A note on optimal assignment of slack due-dates in single-machine scheduling". *European Journal of Operational Research* 70 (3): 311–315.

Graves, S. C. 1980. "The multi-product production cycling problem". *AIIE transactions* 12 (3): 233–240.

He, S., M. Sim, and M. Zhang. 2019. "Data-driven patient scheduling in emergency departments: A hybrid robust-stochastic approach". *Management Science*.

Hernandez, A. S., T. W. Lucas, and M. Carlyle. 2012, November. "Constructing nearly orthogonal latin hypercubes for any nonsaturated run-variable combination". *ACM Transactions on Modeling and Computer Simulation* 22 (4): 20:1–20:17.

Jaikumar, R. 1974. "An operational optimization procedure for production scheduling". *Computers & Operations Research* 1 (2): 191–200.

Ju, F., J. Li, and J. A. Horst. 2016. "Transient analysis of serial production lines with perishable products: Bernoulli reliability model". *IEEE Transactions on automatic control* 62 (2): 694–707.

Karmarkar, U. S. 1993. "Manufacturing lead times, order release and capacity loading". *Handbooks in operations research and management science* 4:287–329.

Kimemia, J., and S. B. Gershwin. 1983. "An Algorithm for the Computer Control of a Flexible Manufacturing System". *AIIE Transactions* 15 (4): 353–362.

Kleijnen, J. P. 2007. *Design and Analysis of Simulation Experiments*. 2nd ed., Springer, New York, NY.

Kumar, A., and R. R. Barton. 2017. "Controlled violation of temporal process constraints–models, algorithms and results". *Information Systems* 64:410–424.

Liberopoulos, G. 2013. "Production Release Control: Paced, WIP-Based or Demand-Driven? Revisiting the Push/Pull and Make-to-Order/Make-to-Stock Distinctions". In *Handbook of Stochastic Models and Analysis of Manufacturing System Operations*, 211–247. Springer, New York, NY.

Lin, C. D., R. Mukerjee, and B. Tang. 2009, March. "Construction of orthogonal and nearly orthogonal Latin hypercubes". *Biometrika* 96 (1): 243–247. Publisher: Oxford Academic.

Maleck, C., and T. Eckert. 2017. "A comparison of control methods for production areas with time constraints and tool interruptions in semiconductor manufacturing". In *2017 40th International Spring Seminar on Electronics Technology (ISSE)*, 1–6. IEEE.

Nadarajah, S., and S. Kotz. 2008, June. "The cycle time distribution". *International Journal of Production Research* 46:3133–3141.

Pedrielli, G., and R. R. Barton. 2019. "Metamodel-Based Quantile Estimation for Hedging Control of Manufacturing Systems". In *Proceedings of the 2019 Winter Simulation Conference*, edited by N. Mustafee, K.-H. Bae, S. Lazarova-Molnar, M. Rabe, C. Szabo, P. Haas, and Y.-J. Son, 452–463. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Prakash, J., and J. F. Chin. 2015. "Modified CONWIP systems: a review and classification". *Production Planning & Control* 26 (4): 296–307.

Rose, O. 1999. "CONLOAD-A new lot release rule for semiconductor wafer fabs". In *WSC'99. 1999 Winter Simulation Conference Proceedings.'Simulation-A Bridge to the Future'(Cat. No. 99CH37038)*, Volume 1, 850–855. IEEE.

Rose, O. 2001. "CONWIP-like lot release for a wafer fabrication facility with dynamic load changes". *Proceedings of the SMOMS* 1:41–46.

Sanchez, S. M., and P. J. Sanchez. 2005, October. "Very large fractional factorial and central composite designs". *ACM Transactions on Modeling and Computer Simulation* 15 (4): 362–377.

Sanchez, S. M., P. J. Sánchez, and H. Wan. 2018. "Work Smarter, not Harder: a Tutorial on Designing and Conducting Simulation Experiments". In *2018 Winter Simulation Conference (WSC)*, 237–251. Piscataway, New Jersey: IEEE Press.

Shantikumar, J. G., and U. Sumita. 1988. "Approximations for the time spent in a dynamic job shop with applications to due-date assignment". *The International Journal of Production Research* 26 (8): 1329–1352.

Shi, C., and S. B. Gershwin. 2012. "Part waiting time distribution in a two-machine line". *IFAC Proceedings Volumes* 45 (6): 457–462.

Shi, C., and S. B. Gershwin. 2016a. "Lead time distribution of three-machine two-buffer lines with unreliable machines and finite buffers".

Shi, C., and S. B. Gershwin. 2016b. "Part sojourn time distribution in a two-machine line". *European Journal of Operational Research* 248 (1): 146–158.

Sloane, N.J.A. 2007. "$OA(68, 2^{67})$. A Library of Orthogonal Arrays. http://neilsloane.com/oadir/index.html. Accessed on 06/18/20".

Sugimori, Y., K. Kusunoki, F. Cho, and S. UchikawaA. 1977. "Toyota production system and kanban system materialization of just-in-time and respect-for-human system". *The International Journal of Production Research* 15 (6): 553–564.

Xu, H. 2009. "Algorithmic Construction of Efficient Fractional Factorial Designs With Large Run Sizes". *Technometrics* 51 (3): 262–277. Publisher: [Taylor & Francis, Ltd., American Statistical Association, American Society for Quality].

Zou, J., Q. Chang, J. Arinez, G. Xiao, and Y. Lei. 2017. "Dynamic production system diagnosis and prognosis using model-based data-driven method". *Expert Systems with Applications* 80:200–209.