Distributed Data Filtering and Modeling for Fog Manufacturing

Yifu Li, Lening Wang, Xiaoyu Chen, and Ran Jin

***Abstract*— A Fog Manufacturing applies both Fog and Cloud Computing collaboratively and integrates manufacturing equipment and processes into an interconnected network through sensing, actuation, and computation nodes. Therefore, most of the computation services in manufacturing can be provided by using both Fog and Cloud. This is important for future manufacturing, as Fog Manufacturing will provide tremendous promise on reliable and responsive computation services, with the potential for privacy preservation to process data in local computation units. However, due to the relatively limited communication bandwidth to Cloud and computation capabilities of Fog nodes, a large amount of data from the manufacturing network will lead to information redundancy and significant computation time latency for data analysis. In this paper, we propose a distributed data filtering (i.e., subsampling) method to extract small but informative data subsets from the raw data, considering the similarity of the interconnected manufacturing processes. The filtered data from each manufacturing process will later be transmitted to Cloud for manufacturing process modeling. A simulation study and a real Fog Manufacturing testbed for an ingot growth manufacturing have been used to validate the proposed distributed data filtering and modeling method. The results indicate that the proposed distributed data filtering and modeling method will not only reduce the sample size and ensure the modeling performance but also improve the performance of the runtime metrics of the computation, such as time latency of the computation services and the communication bandwidth utilization, in Fog Manufacturing, compared with the centralized Cloud Computing.**

***Note to Practitioners*— This research was motivated by the data-driven modeling of an ingot growth manufacturing network with Fog Computing capability. Due to massive data generated from each furnace, analyzing all data will pose high computation cost at the Fog nodes or high communication workload to the Cloud. In this paper, we proposed a distributed data filtering and modeling method for the manufacturing network. Specifically, we filter data from each furnace at the Fog nodes and model the manufacturing network collectively at the Cloud. The proposed method adopts an automatic procedure to determine the optimal filtered sample size of data, such that the practitioners will not rely on trial-and-error to decide the filtered sample size. We also validate the proposed method with both a simulation study and a real Fog Manufacturing testbed for an ingot growth manufacturing**

***Index Terms*— cyber manufacturing process, data filtering, Fog Computing, Fog Manufacturing, manufacturing network modeling**

I. Introduction

THE Cloud Computing [1] has significantly improved manufacturing productivity and flexibility, ensured product quality, and reduced the manufacturing cost. Fog Computing extends the Cloud Computing paradigm close to the "Edge" of the manufacturing network [1], i.e., near the manufacturing equipment, processes and data [2], thus improving the performance of runtime metrics, such as time latency of the computation services and the communication bandwidth utilization [3]. In this paper, Fog is defined as low-cost computation devices, including Edge and other devices between Edge and Cloud [4]. A Fog Manufacturing process applies both Fog and Cloud Computing cooperatively collaboratively and integrates manufacturing equipment and processes into an interconnected network through sensing, actuation, and computation nodes [3]. Specifically, most of the computation services, such as process modeling, monitoring, diagnosis in manufacturing, are provided by using both Fog and Cloud to perform data analysis and facilitate decision-making in manufacturing. Such Fog Manufacturing [3] provides promise on reliable and responsive computation services, with the potential for privacy preservation to process data in local computation units. However, due to the relatively limited communication bandwidth to Cloud and computation capabilities of Fog nodes, a large amount of data from the manufacturing network lead to significant computation time latency for data analysis [5].

Taking an ingot growth manufacturing network as an example [2], there are multiple ingot growth furnaces equipped with sensors jointly collecting a large amount of *in situ* data [6]. Fig. 1 shows the facilities of the manufacturing. Specifically, the sensors on each furnace record more than thousands of multivariate observations per ingot. As more processes are connected in the manufacturing network, it is challenging to analyze all the data, since the Fog nodes have limited computation capabilities, while the transmission of all data to Cloud will pose significant time latency. Thus, the current computation network cannot provide computation services in a timely manner. As a result, decisions in prediction, change detection, root cause diagnosis, cannot be completed by deadlines, and the manufacturing process yields poor quality

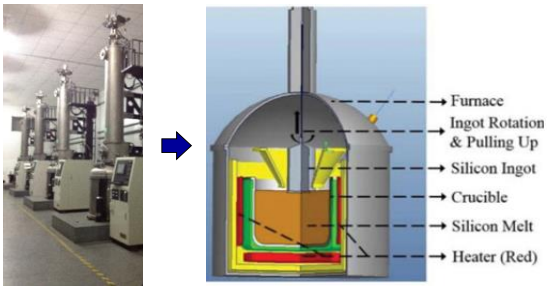


Fig. 1. A schematic of a crystal growth furnace network: the network of multiple furnaces (left) and the internal structure of each one (right) (redrawn with authors' permission) [2]

and production control performances. Second, data privacy is another issue when sending data to the Cloud. Even with the 5G communication when the time latency is no longer a challenge in the near future [7], sending all data to Cloud will release manufacturing data containing Intellectual Property and Know-How of the enterprises (e.g., process recipe, part design, and product quality). Third, the computation will become risky in the event of communication loss to Cloud, which cannot provide reliable computation results.

As a result, there is a need to generate small but informative data to provide affordable computational cost for Fog nodes and reduce the communication workload to Cloud. This game-changing opportunity lies in the similarity of the manufacturing processes, from the same ingot growth furnace configuration, comparable machine status, and maintenance, to similar recipe used in producing the same type of products. The data collected from such a "similar-but-non-identical" manufacturing process will have similar process input-output relationships and significant information redundancy [8]. Reducing the same size by depressing the information redundancy becomes a potential solution.

There are existing data filtering methods, also known as "subsampling", studied as a useful tool to extract small but informative data subsets in manufacturing. However, obtaining a suitable sample size for analysis can be a nontrivial question. A large amount of data will create communication and computational challenges, while a small data set will affect the data analysis performance. Therefore, existing methods often apply data filtering before analysis without knowing if the sample size filtered is appropriate for analysis. As a result, there is no guarantee that the filtering will result in a suitably sized data subset for data analysis. Furthermore, it is still an open question to utilize the similarities of the same types of processes (e.g., ingot growth) to reduce the sample size required for estimating the model of each process.

The objective of this research is to integrate the data filtering and modeling for Fog Manufacturing with similar-but-non-identical processes. Other data analysis, such as process monitoring, root cause diagnosis, and prognosis, are out of the scope of this paper. In order to tackle the knowledge gaps of the state-of-the-art, we proposed a distributed data filtering and modeling (DDFM) method. Specifically, the proposed DDFM method penalizes the sample size used for model estimation to reduce the computational cost. Additionally, the DDFM method penalizes heterogeneities between the individual model of each process and the network baseline model to incorporate the process similarity assumption. Here, the network baseline model reflects the commonalities in model structures among processes and warm-starts the model estimation of each process to reduce the data size required. To accelerate the computation of the proposed method, we propose a heuristic algorithm that distributes the filtering and modeling of each process in each Fog Node.

The DDFM method will effectively reduce the sample size and ensure computation performances because it automatically and adaptively determines the sample size extracted from each process with no prior knowledge required. Furthermore, the DDFM enables information sharing across the models of different processes in Fog Manufacturing in a sensible way for improved computation performance. The most informative samples from similar-but-non-identical processes will be identified to estimate a network baseline model.

A simulation study and a real Fog Manufacturing testbed for the ingot growth manufacturing have been used to validate the DDFM method. The benefit of the proposed method is two-fold. First, the proposed method is easy to be embedded in Fog Manufacturing for process modeling with a reduced computational cost. Second, compared with all benchmark filtering algorithms, the proposed method generated the most cost-effective modeling performance (i.e., the smallest modeling error with the same sample size filtered from the Fog Manufacturing).

The rest part of the paper is organized as follows. In Section II, we review the existing work of data filtering and modeling. In Section III, we introduce the DDFM method and its estimation algorithm. In Section IV, we perform the simulation study to evaluate the performance of the proposed method under varying simulation settings. In Section V, we perform the case study in Fog Manufacturing testbed. In Section VI, we summarize the paper and discuss future work.

II. Literature Review

Data analytics in manufacturing, such as process modeling, quality prediction, process monitoring, and process control, have been widely adopted. To meet the demands of various data analytics tasks, people have integrated a variety of data analytics algorithms as computational service platforms. Existing works on computational services include but not limited to the Berkeley data analysis system (BDAS) [9], Cloud-based industrial automation [10], machine learning-based natural language processing (NLP) services [11], healthcare services [12], etc.

Among the data analytics algorithms used for computation services, regression is one of the most popular methods to model manufacturing processes or networks. Along this direction, the state-of-the-art approaches include Ridge Regression [13], LASSO [14], Elastic Net [15], sure independence screening [16], etc. However, these methods mainly concentrate on stand-alone processes, which are inefficient when there are multiple same types of manufacturing processes connected in a network. Therefore, as the scope scales up to modeling the manufacturing network, we often assume that those connected processes have similar-but-non-identical model structures and parameters [17]. Effectively capturing such similarities in modeling often leads to better modeling performance. To model those processes, researchers proposed transferred learning [18] with applications in wafer defect-recognition [19], wastewater treatment [20], and multi-task learning (MTL) [21], with applications in MRI data analysis [22] and thermal field estimation for grain storage [23]. As the sample size grows, however, the existing modeling methods often create significant computational and data transformation challenges since it requires to aggregate all information from each process together. Such challenges

become greater as more processes are involved.

Significant efforts in modeling methods are to study how to incorporate data reduction for modeling when the sample size needs to be reduced before analysis. The most generally purposed sampling methods (not only for modeling) are uniform sampling, which extracts observations with a uniformly distributed probability. The most popular uniform sampling methods include random sampling [24-26] and stratified sampling [27-29]. When a regression model is designated as the down-stream task following sampling or filtering, people study how to better preserve the information in the design matrix (i.e., input variable for regression) for model estimation with the least variation. The research along this direction includes information-based uniform sampling for regression [30], leveraging-based sampling for regression [31-33], etc. Among these works, the state-of-the-art method is information-based optimal subdata selection (IBOSS) [8], which has low computational intensity and extract information-rich data subset for modeling. In the experimental study, IBOSS outperforms the past benchmark filtering methods on modeling performance. The idea of IBOSS is similar to the optimal experimental design [34], which extracts the observations with extreme values to maximize the information preserved.

The major drawback of the existing filtering methods for modeling is that they are always performed before the modeling step. As a result, the pre-determined data subset sizes in the filtering step are weakly associated with modeling requirements. When using these filtering methods for modeling, the filtered data subset might not always yield satisfactory modeling performance, which should be addressed.

III. The Proposed Data Filtering Method

A. Distributed Data Filtering and Modeling

We proposed the DDFM method for modeling a manufacturing network in Fog Manufacturing that can automatically determine the filtering ratio, i.e., the percentage of observations filtered from the raw data, based on modeling performance. We assume that the K processes studied are similar-but-non-identical, so they hold similar process input-output relationships and result in similar model parameters. The commonalities among these processes, referred to as a network baseline model, can be estimated to serve as warm-start when estimating the individual model for each system to reduce the data usage in Fog Manufacturing. Denote predictors in observation i as $\boldsymbol{x}_i^k \in \mathbb{R}^{p+1}$ and the response as $y_i^k \in \mathbb{R}^1$ for Process $k = 1, \dots, K$, then we assume $\boldsymbol{x}_1^k$ and y_i^k follow a linear relationship as

$$y_i^k = \boldsymbol{x}_i^{k\prime}\boldsymbol{\beta}^k + \epsilon^k,\ i = 1, \dots, n_k, \tag{1}$$

where $\boldsymbol{\beta}^k \in \mathbb{R}^{p+1}$ denote the model parameters for process k, and the residual $\epsilon^k \sim Normal(0, \sigma_k^2)$. We validated the normality assumptions in Figure A-1 of the Appendix and similarities among processes in Figure A-2 of the Appendix.

Let w_i be the indicator variable that equals 1 if observation $(\boldsymbol{x}_i^k, y_i^k)$ is filtered and 0 otherwise, the filtering for all K processes can be formulated as

$$\begin{aligned} &\min_{w_i^k, \boldsymbol{\beta}^k} \sum_{k=1}^{K}\sum_{i=1}^{n_k} w_i^k (y_i^k - \boldsymbol{x}_i^{k\prime}\boldsymbol{\beta}^k)^2 \\ &s.t. \frac{\sum_{i=1}^{n_k} w_i^k}{n_k} \le r^k \text{ for } k = 1, \dots, K, \end{aligned} \tag{2}$$

where $0 \le r^k \le 1$ is the filtering ratio for Process k. However, several challenges exist when applying Problem (2) for modeling with filtered data. The first one is that practitioners have to manually define $r^k, k = 1, \dots, K$ based on their subjective judgment. Furthermore, the above formulation does not consider the similarities of the models estimated for different manufacturing processes. Enforcing the similarities for model parameter estimation often leads to improved modeling performance [35].

To address the above challenges, we propose to estimate model parameters based on filtered data with automatically determined filtering ratio r^k for different processes in the network. To enforce model similarities among processes, we propose to incorporate commonalities of the model parameters shared across processes into the estimation. We denote such commonalities as the network baseline model with parameters $\boldsymbol{\beta}' \in \mathbb{R}^{p+1}$. Then, we enforce the similarities between the model parameter vector of each process $\boldsymbol{\beta}^k$ and $\boldsymbol{\beta}'$, so that $\boldsymbol{\beta}'$ warm-starts estimation of $\boldsymbol{\beta}^k$ to reduce the sample size required. In summary, we refer to the proposed method as the distributed data filtering and modeling (DDFM) with the objective function presented in Problem (3). The estimation for the network baseline model $\boldsymbol{\beta}'$ and the model for each process, $\boldsymbol{\beta}^1, \dots, \boldsymbol{\beta}^K$ can be written as:

$$\begin{aligned} \underset{\boldsymbol{\beta}', \boldsymbol{\beta}^1, \dots, \boldsymbol{\beta}^K}{\arg\min} &\sum_{k=1}^{K}\sum_{i=1}^{n_k} w_i^k (y_i^k - \boldsymbol{x}_i^{k\prime}\boldsymbol{\beta}^k)^2 \\ &+ \lambda_1 \sum_{k=1}^{K} \|\boldsymbol{\beta}^k - \boldsymbol{\beta}'\|_1 \\ &+ \lambda_2 \sum_{k=1}^{K}\sum_{i=1}^{n_k} w_i^k \end{aligned} \tag{3}$$

where $|\cdot|_1$ is the l_1 norm, λ_1 and λ_2 are the tunable hyperparameters to balance the weight of three terms in Problem (3). Incorporating the l_1-norm enforces the difference between the network baseline model parameters ($\boldsymbol{\beta}'$) and individual process model parameters ($\boldsymbol{\beta}^k$).

Although Problem (3) is well-defined, several issues remain challenging. The first one is that obtaining an exact solution of Problem (3) can be computationally expensive or infeasible, mainly due to the presence of determining all binary variables of w_i^k [36]. Furthermore, the tuning process of λ_1 and λ_2 can also become computationally expensive (e.g., through cross-validation). To address these challenges, we proposed a scalable heuristic, suitable for distributed computation in Fog Manufacturing, for the DDFM method.

B. The Heuristic Algorithm for Solving DDFM

The heuristic algorithm contains two steps: 1) estimating the network baseline model parameter $\boldsymbol{\beta}'$; and 2) estimating the individual model parameter $\boldsymbol{\beta}^k$ for each process. In both steps, the heuristic adopts information-based optimal subdata selection (IBOSS) [8], the state-of-the-art model-based filtering for modeling method, to reduce the computational complexity. For Process k, IBOSS proceeds as follows: For each variable, it selects $2s$ observations with s smallest and s largest values of the underlying variable, then it removes the previously selected samples and moves to the next variable. IBOSS repeats such a selection process for all p variables (without the intercept term). If there are $\hat{n}^k$ amount of observations to be extracted from Process k, then $r = \frac{\hat{n}^k}{2p}$.

When estimating $\boldsymbol{\beta}'$, the heuristic uses data-shared Lasso [35], which proposes that any $\boldsymbol{\beta}^k$ can be generated as $\boldsymbol{\beta}^k = \boldsymbol{\beta}' + \Delta_k$. Such a formulation makes $\boldsymbol{\beta}'$ the ideal estimator for the network baseline model. Denote the filtered data set from each process as ($\hat{X}^k \in \mathbb{R}^{\hat{n}^k \times (p+1)}, \hat{\boldsymbol{y}}^k \in \mathbb{R}^{\hat{n}^k}$) from the raw data ($X^k \in \mathbb{R}^{n_k \times (p+1)}, \boldsymbol{y}^k \in \mathbb{R}^{n_k}$), $\boldsymbol{\beta}'$ is estimated as part of $\boldsymbol{\beta} = (\boldsymbol{\beta}', \boldsymbol{\beta}^1, \dots, \boldsymbol{\beta}^K)$ as

$$\boldsymbol{\beta} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \frac{1}{2} \left\| \hat{y} - \hat{Z}\boldsymbol{\beta} \right\|_2^2 + \|\boldsymbol{\beta}\|_1, \qquad (4)$$

where $\hat{Z} = \begin{pmatrix} \hat{X}^1 & \frac{1}{K}\hat{X}^1 & \dots & 0 \\ \vdots & & & \\ \hat{X}^K & 0 & \dots & \frac{1}{K}\hat{X}^K \end{pmatrix}$, $\hat{\boldsymbol{y}} = \left(\hat{\boldsymbol{y}}^{1\prime}, \dots, \hat{\boldsymbol{y}}^{K\prime}\right)'$, where is the data subset filtered from Process k. As data-shared Lasso adopts l_1-norm-based regularization, it only requires a small amount of data from each process to estimate $\boldsymbol{\beta}'$ [35]. To estimate $\boldsymbol{\beta}'$ with minimal computational complexity, we adopt IBOSS with $r = 1$ for $(\hat{X}^k, \hat{\boldsymbol{y}}^k)$, where $k = 1, \dots, K$.

Given $\boldsymbol{\beta}'$, the heuristic applies IBOSS to estimate $\boldsymbol{\beta}^k$ using Problem (5) as

$$\boldsymbol{\beta}^k = \underset{\boldsymbol{\beta}^k}{\operatorname{argmin}} \left\| \boldsymbol{y}^k - \hat{X}^k\boldsymbol{\beta}^k \right\|_2^2 + \|\boldsymbol{\beta}' - \boldsymbol{\beta}^k\|_1, \qquad (5)$$

with iteratively increased values of s for IBOSS, until the change of $\boldsymbol{\beta}^k$ between two consecutive iterations drops below a pre-defined threshold (e.g., $(\boldsymbol{\beta}_s^k\text{-}\boldsymbol{\beta}_{s-1}^k)/\boldsymbol{\beta}_{s-1}^k < 1e^{-3}$ between iteration s and $s-1$).

We summarized the heuristic in Algorithm 1. Here we observe that the estimation of $\boldsymbol{\beta}^k$ in Step 2 can be easily distributed to each individual process with the dedicated Fog nodes. Therefore, we can implement the proposed DDFM in the Fog Manufacturing, which will be illustrated in detail in Section V.

ALGORITHM 1
PSEUDOCODE FOR THE DISTRIBUTED FILTERING AND MODELING HEURISTIC

Initialize $\boldsymbol{\beta}'$:
Step 1 (For $\boldsymbol{\beta}'$ of the network):
$(\hat{X}^1, \hat{\boldsymbol{y}}^1), \dots, (\hat{X}^K, \hat{\boldsymbol{y}}^K)$
$= IBOSS(X^1, \boldsymbol{y}^1, 1), \dots, IBOSS(X^K, \boldsymbol{y}^K, 1)$
$\hat{Z} = \begin{pmatrix} \hat{X}^1 & \frac{1}{K}\hat{X}^1 & \dots & 0 \\ \vdots & & & \\ \hat{X}^K & 0 & \dots & \frac{1}{K}\hat{X}^K \end{pmatrix}, \hat{\boldsymbol{y}} = \left(\hat{\boldsymbol{y}}^{1\prime}, \dots, \hat{\boldsymbol{y}}^{K\prime}\right)'$
$\boldsymbol{\beta} = (\boldsymbol{\beta}', \boldsymbol{\beta}^1, \dots, \boldsymbol{\beta}^K) = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \frac{1}{2} \left\| \hat{y} - \hat{Z}\boldsymbol{\beta} \right\|_2^2 + \|\boldsymbol{\beta}\|_1$
Step 2 (For Process $k = 1, \dots, K$):
$s = 1$
Do Until $(\boldsymbol{\beta}_s^k\text{-}\boldsymbol{\beta}_{s-1}^k)/\boldsymbol{\beta}_{s-1}^k < 1e^{-3}$:
$\hat{X}^k, \hat{\boldsymbol{y}}^k = IBOSS(X^k, \boldsymbol{y}^k, s)$
$\boldsymbol{\beta}^k = \underset{\boldsymbol{\beta}^k}{\operatorname{argmin}} \left\| \boldsymbol{y}^k - \hat{X}^k\boldsymbol{\beta}^k \right\|_2^2 + \|\boldsymbol{\beta}' - \boldsymbol{\beta}^k\|_1$
$s = s + 1$

IV. SIMULATION

The objective of the simulation is to evaluate the modeling performance of the DDFM method in various simulation settings and compare the performance with a few benchmark methods. In particular, we considered four settings by varying the sample size of the raw data in each process (n^k), the process similarities of the underlying model parameters (τ), the sparsity of model parameters (ρ), and the signal-to-noise ratios (γ), summarized in TABLE I. Inspired by Wang, et al. [8], for each process, we simulated input variables as $X^k \in \mathbb{R}^{n^k \times (p+1)}$, with the first p variables following $Normal(\boldsymbol{\mu}, \Sigma)$, and the additional intercept variable as all one. Each element in $\boldsymbol{\mu} \in \mathbb{R}^p$ follows $Uniform([0, 1))$ and the regularized covariance matrix $\Sigma = [\sigma_{ij}]$ with $\sigma_{ii} = 1$ and $\sigma_{ij} = 0.3$ when $i \neq j$.

To generate the response variable $\boldsymbol{y}^k \in \mathbb{R}^{n^k}$ for Process k, we adopt a linear model $\boldsymbol{y}^k = X^k\boldsymbol{\beta} + \boldsymbol{\epsilon}^k$, where $\boldsymbol{\beta}^k \in \mathbb{R}^{p+1}$, $\boldsymbol{\epsilon}^k \in \mathbb{R}^{n^k}$ as the residual term with each element following $N(0, \sigma_k^2)$ in independent and identically distributed. We used the similarity parameter τ to control the similarity of the model parameters among different processes. Specifically, we first simulated a network baseline model parameters $\boldsymbol{\beta}' \in \mathbb{R}^{p+1}$, with each element following $Normal(0,1)$. Then $\boldsymbol{\beta}^k = \boldsymbol{\beta}' + \frac{1}{\tau}\boldsymbol{s}$, where $\boldsymbol{s} = \left(sign(\beta_0')c_0, \dots, sign\left(\beta_p'\right)c_p\right)'$, $c_i \sim Uniform([0, 1))$, and $sign(\cdot)$ returns 1 when the value is larger than 0 and returns -1 otherwise.

For each Process k, we set ρ^k as the model sparsity, representing the percentage of significant variables in the model

TABLE I
SIMULATION SETTING VARIABLES AND THEIR VALUES II

Setting Variables	Low	High
Sample Size of Raw Data (n^k)	5000	10000
Process Similarity (τ)	5	10
Model Sparsity (ρ^k)	0.3	0.7
Signal-to-Noise Ratio (γ)	3	7

TABLE IIIIV
MEAN AND STANDARD ERRORS (WITHIN PARENTHESIS) OF MODEL PARAMETER ESTIMATION ERROR OVER 100 SIMULATION REPLICATIONS

n^k	$\tilde{n}^k$	τ	ρ	γ	Proposed	IBOSS	Random	Stratified
5000	86.32	5	0.3	3	**0.062 (0.002)**	0.171 (0.006)	0.195 (0.007)	0.195 (0.007)
5000	86.68	5	0.3	7	**0.051 (0.001)**	0.142 (0.006)	0.167 (0.007)	0.163 (0.007)
5000	84.8	5	0.7	3	**0.105 (0.002)**	0.271 (0.007)	0.313 (0.009)	0.310 (0.008)
5000	85.64	5	0.7	7	**0.082 (0.002)**	0.207 (0.006)	0.244 (0.007)	0.241 (0.007)
5000	82.12	10	0.3	3	**0.048 (0.002)**	0.180 (0.007)	0.205 (0.008)	0.206 (0.009)
5000	81.64	10	0.3	7	**0.038 (0.001)**	0.140 (0.005)	0.163 (0.006)	0.162 (0.006)
5000	80.92	10	0.7	3	**0.086 (0.002)**	0.261 (0.007)	0.297 (0.007)	0.304 (0.008)
5000	81.6	10	0.7	7	**0.065 (0.002)**	0.205 (0.006)	0.240 (0.006)	0.244 (0.007)
10000	85.68	5	0.3	3	**0.061 (0.002)**	0.170 (0.006)	0.201 (0.007)	0.199 (0.007)
10000	85.56	5	0.3	7	**0.050 (0.001)**	0.138 (0.005)	0.167 (0.007)	0.164 (0.006)
10000	83.76	5	0.7	3	**0.098 (0.002)**	0.256 (0.007)	0.309 (0.008)	0.311 (0.009)
10000	84.52	5	0.7	7	**0.083 (0.002)**	0.204 (0.006)	0.243 (0.007)	0.242 (0.007)
10000	81.08	10	0.3	3	**0.051 (0.002)**	0.178 (0.007)	0.198 (0.007)	0.204 (0.008)
10000	82	10	0.3	7	**0.036 (0.001)**	0.130 (0.005)	0.150 (0.005)	0.148 (0.006)
10000	81.36	10	0.7	3	**0.084 (0.002)**	0.250 (0.006)	0.299 (0.007)	0.299 (0.007)
10000	81.8	10	0.7	7	**0.063 (0.002)**	0.195 (0.006)	0.229 (0.007)	0.225 (0.007)

(non-zero elements in $\boldsymbol{\beta}^k$). All processes share the same variable significance pattern as the connected processes are similar in manufacturing. When generating $\boldsymbol{\epsilon}^k$, we control the signal-to-noise ratio as $\gamma = \frac{var(X^k\boldsymbol{\beta}^k)}{var(\epsilon^k)}$. In this simulation, $p = 20$, which is similar to the case study. The values of the simulation setting variables are summarized in TABLE I.

We compared the modeling performance of the proposed DDFM method with data randomly partitioned into training and testing sets in 90%-10% split for each process over 100 replications. Two performance measures on model parameter estimation error and prediction error on the testing sets are adopted. The first performance measure is the root-mean-squared errors between the simulated underlying true parameters and estimated for all processes. The second performance measure is root-mean-squared prediction error (RMSPE) when the trained model predicts the $\boldsymbol{y}^k$ of the 10% testing data in each process.

We considered several benchmark methods, which first extract data from one process at a time, and then estimate their model parameters with LASSO regression based on the filtered dataset [14]. The proposed DDFM method can automatically determine the number of observations preserved during filtering. Therefore, to ensure a fair performance comparison, each benchmark method will preserve the same sample size. For example, if the proposed method extracts N observations in total from K processes, then each benchmark method extracts $\frac{N}{K}$ observations to generate as the filtered dataset. The benchmark methods are IBOSS [8], random sampling [25], which extracts observations randomly from the full data set, and stratified sampling [28], which partitions data into the equally spaced and non-overlapping windows, and then extracts the first observation in each window.

We summarize the results of model parameter estimation error and response prediction accuracy in TABLE II and TABLE III with n^k and $\tilde{n}^k$ represents the full and the average filtered data size for each process. The best results in each simulation setting are highlighted in bold. Both tables show that the proposed method significantly outperformed all the benchmark methods in all simulation settings. It is because these benchmark methods do not consider the model similarities among different processes. For the impact of each setting, we can observe that the increase of sample size (n^k) did not lead to a significant difference in terms of the modeling performance. When processes are more similar to each other (τ grew larger), the modeling errors were smaller for the proposed method, since it explicitly enforces the similarities of model parameters estimated for different processes. As we increased the model sparsity (ρ) and generated a more complex model, the errors grew larger for all methods. The proposed method, however, yielded superior performance on both levels of ρ's. Similarily, the less noise in the model residuals due to the increase of signal-to-noise ratio (γ) result in improved the performance for all methods. The proposed method yielded superior performance on both levels of γ.

TABLE VVI
MEAN AND STANDARD ERRORS (WITHIN PARENTHESIS) OF PREDICTION ERROR OVER 100 SIMULATION REPLICATIONS

n^k	$\tilde{n}^k$	τ	ρ	γ	Proposed	IBOSS	Random	Stratified
5000	86.32	5	0.3	3	**1.186 (0.040)**	1.243 (0.042)	1.281 (0.043)	1.281 (0.044)
5000	86.68	5	0.3	7	**0.797 (0.024)**	0.836 (0.026)	0.863 (0.027)	0.859 (0.027)
5000	84.8	5	0.7	3	**1.826 (0.044)**	1.960 (0.048)	2.030 (0.050)	2.031 (0.050)
5000	85.64	5	0.7	7	**1.302 (0.033)**	1.389 (0.036)	1.444 (0.038)	1.442 (0.038)
5000	82.12	10	0.3	3	**1.113 (0.035)**	1.181 (0.037)	1.212 (0.038)	1.214 (0.038)
5000	81.64	10	0.3	7	**0.790 (0.024)**	0.839 (0.025)	0.868 (0.026)	0.866 (0.026)
5000	80.92	10	0.7	3	**1.711 (0.043)**	1.846 (0.046)	1.917 (0.047)	1.921 (0.048)
5000	81.6	10	0.7	7	**1.242 (0.029)**	1.342 (0.032)	1.396 (0.033)	1.401 (0.033)
10000	85.68	5	0.3	3	**1.131 (0.034)**	1.187 (0.036)	1.222 (0.037)	1.220 (0.037)
10000	85.56	5	0.3	7	**0.853 (0.022)**	0.891 (0.023)	0.924 (0.024)	0.926 (0.025)
10000	83.76	5	0.7	3	**1.775 (0.045)**	1.896 (0.048)	1.984 (0.051)	1.979 (0.051)
10000	84.52	5	0.7	7	**1.284 (0.033)**	1.369 (0.036)	1.426 (0.037)	1.427 (0.038)
10000	81.08	10	0.3	3	**1.173 (0.037)**	1.242 (0.039)	1.275 (0.040)	1.281 (0.040)
10000	82	10	0.3	7	**0.758 (0.023)**	0.804 (0.024)	0.830 (0.025)	0.830 (0.025)
10000	81.36	10	0.7	3	**1.708 (0.042)**	1.839 (0.045)	1.918 (0.046)	1.917 (0.047)
10000	81.8	10	0.7	7	**1.203 (0.031)**	1.295 (0.033)	1.352 (0.034)	1.348 (0.034)

V. INGOT GROWTH MANUFACTURING CASE STUDY

In the case study, we proposed and implemented testbed for an ingot growth manufacturing with a Fog Computing system [3] using DDFM. Compared with Cloud Computing, Fog Computing allows various data analysis tasks to be distributed and executed at the local computation units (i.e., Fog nodes) of each manufacturing process. As a result, it can support the corresponding data analysis and decision-making with low computational time latency [37].

The architecture of the testbed is shown in Fig. 3. There are three hierarchical layers in the testbed. From top to bottom, the first layer is the Cloud-based orchestrator, which directly controls the communication flow (i.e., data flow and computation flow) [3]. Moreover, the corresponding computation and communication utilization information can be recorded to monitor the status of the process on the orchestrator [1]. In the proposed testbed, a high-performance workstation (CPU: Intel i7-6700K) is employed as the orchestrator.

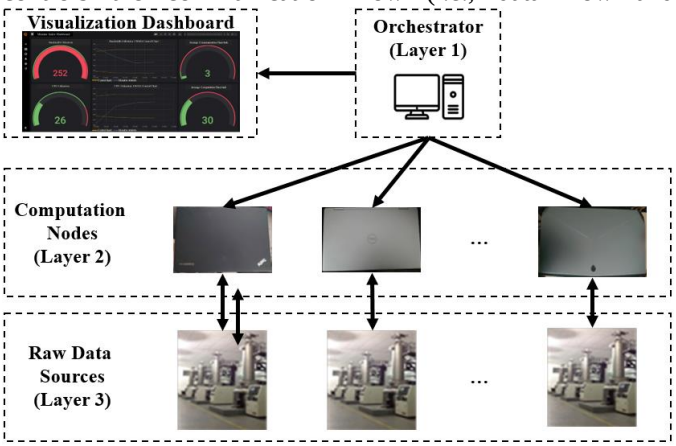


Fig. 3. The Architecture of the Fog Manufacturing Testbed [1]

The second layer consists of a group of Fog nodes. In this layer, we adopt five laptops (CPU1: Intel i7-4720HQ; CPU2: i7-5600U; CPU3: i7-8705G; CPU4: i7-6600u; CPU5: i7-6820HQ) as five Fog nodes. Each Fog node can collect the data from the manufacturing process and store the raw data in a local database for further data analysis. The orchestrator can coordinate the Fog nodes to execute computation tasks with a specific dataset simultaneously [38].

The third layer is the raw data source. In this case study, we

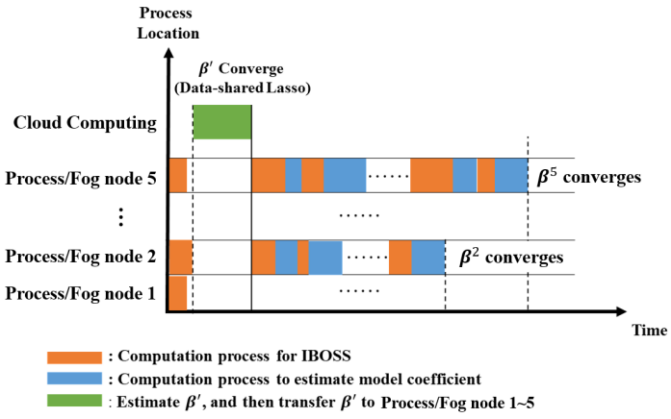


Fig. 2. The Schematic of Computation Flow

adopt the ingot growth manufacturing network [2], containing the production data from five similar-but-non-identical manufacturing furnaces. To prevent the preliminary study from interrupting physical manufacturing processes, we collected and stored data in advance at each perspective Fog node [2].

Fig. 3 shows the schematic of the computation flow for the proposed DDFM method on the testbed. Step 1 of Algorithm 1 is implemented on the Cloud (the orchestrator) and Step 2 of Algorithm 1 is implemented at each individual Fog node for each process in parallel.

To evaluate the performance of the proposed method, we adopt the same benchmark methods from the simulation study, including IBOSS, random sampling, and stratified sampling. We evaluated the performance of the proposed filtering method with data randomly partitioned into training and test sets in 90%-10% split for each process over 100 replications. The prediction performance of the model is evaluated with root-mean-squared prediction error (RMSPE) and its standard error (SE) over 100 replications.

The prediction RMSPEs and corresponding SE from both Cloud and Fog scenarios are summarized in TABLE VII

The table shows that the proposed method yields the best performance compared with other benchmarks. Furthermore, the Cloud and the Fog computing yield comparable performance as the Fog Computing executes the same computational tasks as the Cloud version does.

To demonstrate the advantages of Fog Computing compared with using the Cloud Computing only, the proposed method is also executed in the Cloud (i.e., the orchestrator of our testbed, which has better computing power than each Fog node does). In this case, all computation tasks for different processes will be executed sequentially. To compare the performance of these two approaches, six performance metrics are calculated following Zhang, et al. [3] in Fig. 4: 1) Scaled RMSPEs; 2) Scaled standard error (SE) of RMSPEs among replications; 3) time latency; 4) redundancy; 5) computation utilization; 6) bandwidth utilization. The RMSPE and SE of the proposed methods are rescaled from TABLE VII to a range between 0 and 1 for better illustration. Furthermore, Time latency represents the total time consumption to finish the corresponding computation task. Redundancy represents how fast a single process can perform its assigned computing task twice so that it will complete the computing tasks during the downtime of other processes [3]. Computing utilization represents the average CPU utilization while performing the computing task. Bandwidth utilization represents how much communication bandwidth is required among Fog nodes and the orchestrator. Similar to the metric 1) and 2), metric 3) to metric 6) were also scaled to the same range from 0 to 1 representing the worst (the center) to the best values (the edge) in Fig. 4 following the literature [3].

TABLE VIIVIII
AVERAGE RMSPE IN CASE STUDY

	Proposed	IBOSS	Random	Stratified
Fog	**0.493 (0.001)**	0.575 (0.001)	0.527 (0.003)	0.516 (0.003)
Cloud	**0.494 (0.001)**	0.572 (0.001)	0.516 (0.003)	0.520 (0.003)

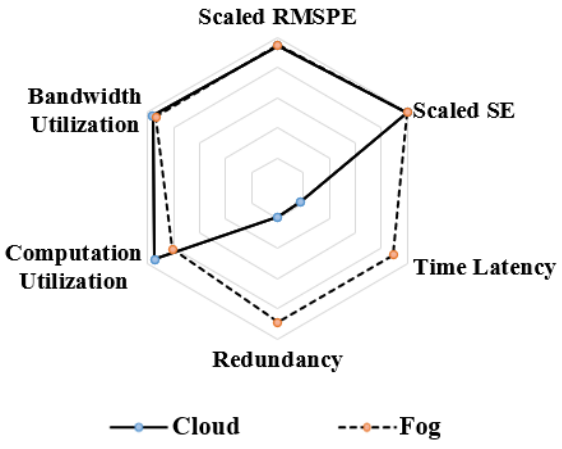


Fig. 4. A Radar Chart for Six Runtime Metrics

From the results shown in Fig. 4, it can be observed that the scaled RMSPE and the scaled SE are comparable in two scenarios. Fog Computing has much better performance than Cloud Computing for both time latency and redundancy. This is because the computation task is assigned to the five Fog nodes and executed in parallel in the Fog Computing, which can provide a responsive computation service. On the other hand, Cloud Computing has better CPU utilization performance than the Fog computing, since the CPU on the orchestrator is much better than the CPU on the Fog nodes, so that more resources can be preserved for other tasks. Moreover, Fog and Cloud computing have a comparable bandwidth utilization performance, it is because the total amount of data transmitted between Fog nodes and the orchestrator is negligible due to the application of the DDFS methods to reduce the sample size.

The real data generated from the case study also be utilized to validate the model assumptions. From Figure A-1, we can observe that there is little variation pattern in the residuals to be explained and no presence of non-normality. We conclude that the selection of the linear model in the proposed method is appropriate and there is no violation of the normality assumption of the residuals. Moreover, Figure A-2 shows the number of times that each variable was selected over 100 Replications, which validates that the processes modeled for ingot growth manufacturing are similar-but-non-identical.

Although the proposed DDFS method on Fog computing testbed outperforms in several performance metrics, one limitation is the proposed DDFS method requires significantly more iterations to convergence. As Fog nodes have inferior computational power, the time latency for the whole process can increase significantly due to the bottlenecks in some of the Fog nodes. A potential solution can be incorporating the predictive off-loading to reasonably assign the computation task to individual Fog node by considering their computation capability and balancing tasks among nodes [39].

VI. SUMMARY

By employing the Fog and Cloud computing, a Fog Manufacturing can provide a responsive and reliable computation service closed to the equipment and data source

with better data privacy. However, limited by the capability of Fog nodes, using all data for communication and computation tasks in the system can lead to an extremely high workload and a large time latency to restrict the real-time data-driven decision-making [ref]. Moreover, the data in the system also has information redundancy that makes the analysis on the full dataset unnecessary. Therefore, it is necessary to reduce the number of observations and keep the informative data at the same time.

In this paper, we propose a DDFM method to extract small but informative data subsets from the raw data by considering the similarity of the interconnected manufacturing processes. DDFM penalizes the sample size used for model estimation to reduce the computational cost. Additionally, DDFM method penalizes heterogeneities between the individual model of each process and the network baseline model to incorporate the process similarity assumption for model estimation. To accelerate the computation of the proposed method, we propose a heuristic algorithm that distributes the filtering and modeling of each process in each Fog Node. The proposed method yields better modeling performance compared with other benchmarks with the same data reduction ratio in the simulation study. Moreover, the proposed method not only has better performance metrics in Fog computing than Cloud computing, but also a better modeling performance as shown in the real case study with ingot growth processes.

There are several directions worth of further investigation. First, the proposed DDFM method will be studied to extend to other data analytical methods, such as process monitoring, control, into the filtering. For example, we can study how to jointly filtering and monitoring for multiple manufacturing processes when they generate a large amount of data simultaneously [40]. Second, we will identify how to incorporate the proposed method in heterogeneous types of data in manufacturing, such as observation data and experimental data [41].

APPENDIX

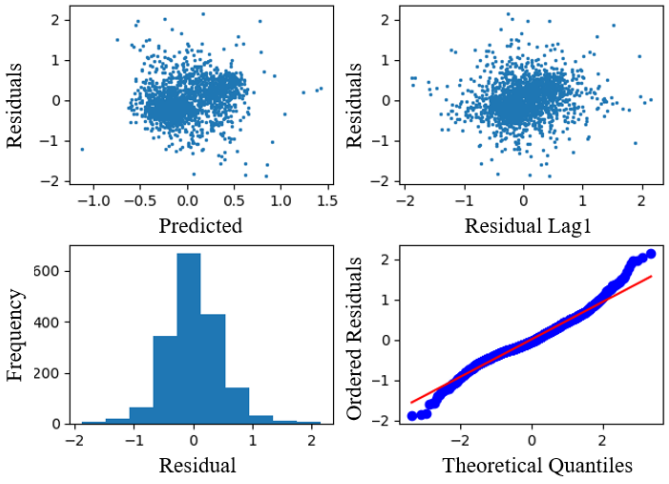


Figure A-1. The Assumption Check of the Modeling Residuals in the Ingot Growth Manufacturing Case Study

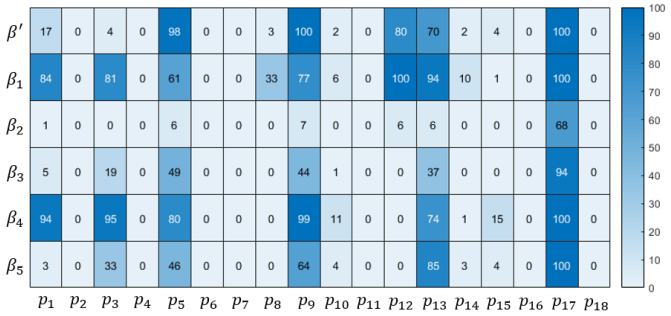


Figure A-2. The Times that Each Variable is Selected Over 100 replications for the Ingot Growth Manufacturing Case Study

REFERENCES

[1] L. Wang, Y. Zhang, and R. Jin, "A Monitoring System for Anomaly Detection in Fog Manufacturing," in *2020 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)*, 2020, pp. 88-93: IEEE.

[2] H. Sun, X. Deng, K. Wang, and R. Jin, "Logistic regression for crystal growth process modeling through hierarchical nonnegative garrote-based variable selection," *IIE Transactions*, vol. 48, no. 8, pp. 787-796, 2016.

[3] Y. Zhang, L. Wang, X. Chen, and R. Jin, "Fog Computing for Distributed Family Learning in Cyber-Manufacturing Modeling," in *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)*, 2019, pp. 88-93: IEEE.

[4] R. Jin, H. Yan, and D. Lee, "FMRG: Pipeline Recommendation, Adaptive Optimization, and Privacy-Preserving Computation for Future Fog Manufacturing (Under Review)," ed: NSF, 2020.

[5] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, 2012, pp. 13-16.

[6] G. Fisher, M. R. Seacrist, and R. W. Standley, "Silicon crystal growth and wafer technologies," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 1454-1474, 2012.

[7] J. Cao, P. Yu, M. Ma, and W. Gao, "Fast authentication and data transfer scheme for massive NB-IoT devices in 3GPP 5G network," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1561-1575, 2018.

[8] H. Wang, M. Yang, and J. Stufken, "Information-based optimal subdata selection for big data linear regression," *Journal of the American Statistical Association*, vol. 114, no. 525, pp. 393-405, 2019.

[9] I. Stoica *et al.*, "The berkeley data analysis system (BDAS): An open source platform for big data analytics," University of California, Berkeley Berkeley United States2017.

[10] T. Hegazy and M. Hefeeda, "Industrial automation as a cloud service," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 10, pp. 2750-2763, 2014.

[11] P. W. Staar, M. Dolfi, C. Auer, and C. Bekas, "Corpus Conversion Service: A machine learning platform to ingest documents at scale," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 774-782.

[12] J. R. Sutton, R. Mahajan, O. Akbilgic, and R. Kamaleswaran, "PhysOnline: an open source machine learning pipeline for real-time analysis of streaming physiological waveform," *IEEE journal of biomedical and health informatics*, vol. 23, no. 1, pp. 59-65, 2018.

[13] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55-67, 1970.

[14] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267-288, 1996.

[15] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301-320, 2005.

[16] J. Fan and J. Lv, "Sure independence screening for ultrahigh dimensional feature space," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 70, no. 5, pp. 849-911, 2008.

[17] T. Evgeniou and M. Pontil, "Regularized multi--task learning," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004, pp. 109-117.
[18] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345-1359, 2009.
[19] J. Yu, Z. Shen, and X. Zheng, "Joint Feature and Label Adversarial Network for Wafer Map Defect Recognition," *IEEE Transactions on Automation Science and Engineering*, 2020.
[20] G. Wang, J. Qiao, J. Bi, W. Li, and M. Zhou, "TL-GDBN: Growing deep belief network with transfer learning," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 2, pp. 874-885, 2018.
[21] G. Obozinski, B. Taskar, and M. I. Jordan, "Joint covariate selection and joint subspace selection for multiple classification problems," *Statistics and Computing*, vol. 20, no. 2, pp. 231-252, 2010.
[22] P. Gong, J. Ye, and C. Zhang, "Multi-stage multi-task feature learning," *The Journal of Machine Learning Research*, vol. 14, no. 1, pp. 2979-3010, 2013.
[23] D. Wang, K. Liu, X. Zhang, and H. Wang, "Spatiotemporal Multitask Learning for 3-D Dynamic Field Modeling," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 2, pp. 708-721, 2019.
[24] K. L. Clarkson and P. W. Shor, "Applications of random sampling in computational geometry, II," *Discrete & Computational Geometry*, vol. 4, no. 5, pp. 387-421, 1989.
[25] H. Liu, R. G. Sadygov, and J. R. Yates, "A model for random sampling and estimation of relative protein abundance in shotgun proteomics," *Analytical chemistry*, vol. 76, no. 14, pp. 4193-4201, 2004.
[26] W. Chen, S. Gao, C.-H. Chen, and L. Shi, "An optimal sample allocation strategy for partition-based random search," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 1, pp. 177-186, 2013.
[27] J. E. Trost, "Statistically nonrepresentative stratified sampling: A sampling technique for qualitative studies," *Qualitative sociology*, vol. 9, no. 1, pp. 54-57, 1986.
[28] E. Liberty, K. Lang, and K. Shmakov, "Stratified sampling meets machine learning," in *Proceedings of 2016 International Conference on Machine Learning*, 2016, pp. 2320-2329.
[29] J. Wu, Y. Chen, S. Zhou, and X. Li, "Online steady-state detection for process control using multiple change-point models and particle filters," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 688-700, 2015.
[30] P. Drineas, M. W. Mahoney, S. Muthukrishnan, and T. Sarlós, "Faster least squares approximation," *Numerische mathematik*, vol. 117, no. 2, pp. 219-249, 2011.
[31] P. Ma and X. Sun, "Leveraging for big data regression," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 7, no. 1, pp. 70-76, 2015.
[32] P. Ma, M. W. Mahoney, and B. Yu, "A statistical perspective on algorithmic leveraging," *The Journal of Machine Learning Research*, vol. 16, no. 1, pp. 861-911, 2015.
[33] P. Drineas, M. Magdon-Ismail, M. W. Mahoney, and D. P. Woodruff, "Fast approximation of matrix coherence and statistical leverage," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 3475-3506, 2012.
[34] J. Kiefer, "Optimum experimental designs," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 21, no. 2, pp. 272-304, 1959.
[35] S. M. Gross and R. Tibshirani, "Data Shared Lasso: A novel tool to discover uplift," *Computational statistics & data analysis*, vol. 101, pp. 226-235, 2016.
[36] O. Burdakov, C. Kanzow, and A. Schwartz, "On a reformulation of mathematical programs with cardinality constraints," in *Advances in Global Optimization*: Springer, 2015, pp. 3-14.
[37] L. Wang, Y. Zhang, X. Chen, and R. Jin, "Online Computation Performance Analysis for Distributed Machine Learning Pipelines in Fog Manufacturing," presented at the IEEE International Conference on Automation Science and Engineering, 2020.
[38] G. Pemmasani, "dispy: Distributed and Parallel Computing with/for Python," 2018.
[39] X. Chen, L. Wang, C. Wang, and R. Jin, "Predictive offloading in mobile-fog-cloud enabled cyber-manufacturing systems," in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*, 2018, pp. 167-172: IEEE.
[40] R. Jin and K. Liu, "Multimode variation modeling and process monitoring for serial-parallel multistage manufacturing processes," *Iie Transactions*, vol. 45, no. 6, pp. 617-629, 2013.
[41] R. Jin and X. Deng, "Ensemble modeling for data fusion in manufacturing process scale-up," *IIE Transactions*, vol. 47, no. 3, pp. 203-214, 2015.

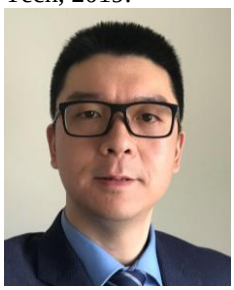
Yifu Li received a B.S. degree in Industrial and Systems Engineering from Virginia Tech in 2016. Currently, he is a Ph.D. candidate in the Grado Department of Industrial and Systems Engineering. He previously worked as Research Assistant at Grado Department of Industrial and Systems Engineering of Virginia Tech and Feinberg School of Medicine of Northwestern University. His research interest is mainly on studying the interface between manufacturing data-driven modeling and data quality.

Mr. Li is a member of INFORMS and IISE. Mr. Li's awards and honors include Best Poster Presentation at Center for Excellence in Logistics and Distribution (CELDi), Fayetteville AR, 2017, and Young Author Award at International Workshop on Intelligentized Welding Manufacturing (IWIWM), Lexington KY, 2019

Lening Wang received the B.S. degree in engineering mechanics from the Shandong University, Jinan, China in 2011, the M.S. degree in mechanical engineering from University of Massachusetts, Amherst, USA in 2014.
He is a research assistant with the Grado Department of Industrial and Systems Engineering at Virginia Tech, Blacksburg, VA, USA. His research mainly concentrates on statistical on-line quality control for advanced manufacturing, such as additive manufacturing, machining, etc., and smart manufacturing network integration.
He is a member of the Institute of Industrial and Systems Engineers, the Institute for Operations Research and the Management Sciences, and the institute of Electrical and Electronics Engineers.

Xiaoyu Chen is a Ph.D. candidate in the Grado Department of Industrial and Systems Engineering at Virginia Tech. He received a B.E. degree from School of Optoelectronics at Beijing Institute of Technology, China in 2015. He previously worked as Research assistant and Grado teaching assistant at Virginia Tech. His research focused on human-machine collaboration and Fog-Cloud computing in smart manufacturing systems.
Mr. Chen is a member of Institute for Operations Research and the Management Sciences (INFORMS), Institute of Industrial and Systems Engineering (IISE), and IEEE. His awards and honors include Doctoral Student of the Year in the Grado

Department of Industrial and Systems Engineering at Virginia Tech, 2019.



Ran Jin is an Associate Professor and the Director of Laboratory of Data Science and Visualization at the Grado Department of Industrial and Systems Engineering at Virginia Tech. He received his Ph.D. degree in Industrial Engineering from Georgia Tech, Atlanta, his Master's degrees in Industrial Engineering, and in Statistics, both from the University of Michigan, Ann Arbor, and his bachelor's degree in Electronic Engineering from Tsinghua University, Beijing.

He has been working with leading manufacturing companies in aerospace, semiconductor, personal care, optical fiber industries. His research focuses on machine learning in manufacturing, manufacturing computation services and cognitive-based interactive visualization.

He is a member of IEEE and currently serving as an Associate Editor for IISE Transactions and an Associate Editor for ASME Transactions, Journal of Manufacturing Science and Engineering. For more information about Dr. Jin, please visit his faculty website at Virginia Tech: https://ise.vt.edu/ran-jin.