Journal of Intelligent Manufacturing
https://doi.org/10.1007/s10845-023-02188-z

A hierarchical ensemble causal structure learning approach for wafer manufacturing

Yu Yang[1] · **Sthitie Bom**[2] · **Xiaotong Shen**[1]

Received: 1 April 2023 / Accepted: 26 July 2023

Abstract
In manufacturing, causal relations between components have become crucial to automate assembly lines. Identifying these relations permits error tracing and correction in the absence of domain experts, in addition to advancing our knowledge about the operating characteristics of a complex system. This paper is motivated by a case study focusing on deciphering the causal structure of a wafer manufacturing system using data from sensors and abnormality monitors deployed within the assembly line. In response to the distinctive characteristics of the wafer manufacturing data, such as multimodality, high-dimensionality, imbalanced classes, and irregular missing patterns, we propose a hierarchical ensemble approach. This method leverages the temporal and domain constraints inherent in the assembly line and provides a measure of uncertainty in causal discovery. We extensively examine its operating characteristics via simulations and validate its effectiveness through simulation experiments and a practical application involving data obtained from Seagate Technology. Domain engineers have cross-validated the learned structures and corroborated the identified causal relationships.

Keywords Causal discovery · Data imbalance · Hierarchical ensemble · High-dimension · Wafer manufacturing

Introduction

In modern manufacturing, the production systems have become increasingly automated yet complex to achieve higher product quality and diversity (Liang et al., 2004). However, the increased complexity imposes challenges in understanding the underlying causal mechanisms of production lines and identifying the root causes of system failures and product defects. In such a situation, gaining knowledge of causal relations between components in the assembly line is crucial. It enhances engineers' understanding while permitting the root-cause tracing of a failure event to allow real-time error corrections in the absence of on-site engineers. Moreover, the causal relations provide precautionary warnings to potential future errors (Huegle et al., 2020), which reduces the chance of assembly line shutdown.

A standard practice in manufacturing is to learn causal relationships through the design of experiments, which can be both expensive and time-consuming given the thousands of factors examined in a production environment. Meanwhile, in high-volume manufacturing sectors like the semiconductor industry, precise control and surveillance of the production processes via sensors and metrology tools are vital for maintaining quality and efficiency (Jeong & Cho, 2006). With the advances in sensor technology, automatic metrology tools, and real-time data collection, we are better capable to acquire causal relationships from observational data by causal discovery (Spirtes et al., 2000; Pearl, 2009). Consequently, much progress has been made in applying causal discovery in the manufacturing domain, such as fault propagation analysis on industrial board machines (Landman & Jämsä-Jounela, 2016), failure precaution in automotive body production (Huegle et al., 2020), and root cause diagnosis in fluid catalytic cracking unit (Gharahbagheri et al., 2015) and semiconductor manufacturing (Shah et al., 2018).

✉ Yu Yang
yang6367@umn.edu

Sthitie Bom
sthitie.e.bom@seagate.com

Xiaotong Shen
xshen@umn.edu

[1] School of Statistics, University of Minnesota, Minneapolis, MN 55455, USA

[2] Seagate Technology, Bloomington, MN 55435, USA

Published online: 15 August 2023

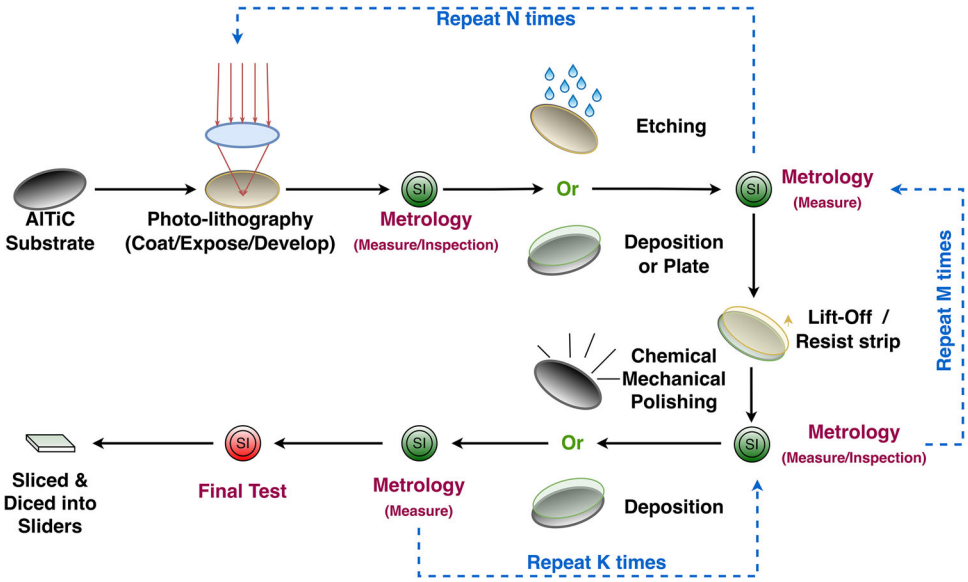

Fig. 1 Wafer Manufacturing Process

This paper focuses on learning causal relations in the wafer manufacturing domain, specifically examining the causal relations among the sensors and abnormal events in a wafer assembly line. The manufacturing of wafers involves a lengthy and complex process that follows a predetermined recipe for creating intricate structures within the wafer. As shown in Fig. 1, the recipe consists of a sequential series of nano-scale process steps, such as photo-lithography, etching, deposition, plating, lift-off, resist strip, and chemical mechanical polishing. Each process step is executed by specific machines[1] (not shown in the diagram), and the operations are monitored by sensors that track various control variables. As the demand for complex device designs with multi-layered and advanced functioning circuit structures increases, re-entrant multi-stage manufacturing lines have become necessary (Kumar, 1993). Consequently, the same set of processes, comprising a stage, are repeated for each circuit board layer. After the completion of each layer, a metrology tool is utilized to measure the product quality based on several criteria, called fault types, to ensure that the processing meets the required quality specifications. Every wafer undergoes all stages of the manufacturing process, and once a wafer product reaches the final stage, its routing history, along with all the sensor variables and the encountered abnormalities related to fault types, is recorded.

Through a case study on a wafer assembly line from Seagate Technology, we identify several notable challenges regarding causal discovery in wafer manufacturing. First, to help with error tracing, we need to learn the causal structure among the sensors and the abnormal events from all process steps along the assembly line. Given the considerable number of process steps, usually exceeding 500, and the multitude of sensors employed in each machine, the recorded data encompass tens of thousands of variables. This results in a high-dimensional scenario where the algorithms' scalability and theoretical consistency may not be guaranteed (Colombo and Maathuis, 2014; Nandy et al., 2018). Second, causal discovery methods tailored for specific data types may not be applicable when dealing with data from multiple sources (sensors and metrology tools), and methods designed for mixed data (Andrews et al., 2018; Cui et al., 2016) tend to be computationally inefficient in high-dimensional scenarios. Third, abnormal events are rare in practical production, resulting in imbalanced classes. Our simulations (as shown in Table 2) indicate that data imbalance has a substantial impact on the accuracy of causal discovery. Surprisingly, the research community has not given adequate attention to effectively addressing data imbalance in causal discovery. While Barnes et al. (2019) and Runge et al. (2019) acknowledge the challenge of class imbalance in causal discovery, they do not propose any effective method to address it. Fourth, in real-life production, the products often do not strictly follow the pre-specified manufacturing process flow. Such deviations lead to irregularly missing data and few samples in the merged data across all steps, upon which it is impossible to learn a causal structure. Existing approaches for missing data (Gao et al., 2022; Tu et al., 2019) are either computationally expensive or incapable of handling mixed data. Finally, the manufacturing process imposes temporal and domain constraints on the causal structure. Properly incorporating these constraints helps reduce the candidate causal graph complexity and helps avoid factual errors and hence requires deliberate attention.

To address these challenges, we propose a hierarchical ensemble approach that leverages temporal and domain constraints on the assembly line and quantifies the uncertainty

[1] *Machines* are also known as *tools* in the semiconductor industry.

of causal discovery. The approach consists of three phases: (1) block-level learning, in which we cluster all steps on the assembly line into blocks and learn a block-level structure with distilled constraints; (2) step-level learning, in which we learn constrained causal structures at a finer granularity based on the block-level structure; and (3) aggregation, where we aggregate the step-level structures and quantify the uncertainty of the learned relations. Our approach offers several advantages. First, it is scalable and can handle high-dimensional data with mixed types. Second, it addresses imbalanced data due to rare events and can handle irregular missing patterns. Finally, it incorporates domain and temporal knowledge to refine the candidate graphs, increasing the accuracy of causal discovery and avoiding factual errors. We demonstrate the effectiveness of our approach through simulations and an application to the wafer manufacturing data from Seagate Technology. The causal structure learned from the data is cross-validated by domain experts, and our proposed modeling pipeline and visualizing tool have received positive feedback from engineers and technicians.

The article consists of six sections. In 'Data characteristics and challenges' section, we delve into the data characteristics specific to the wafer manufacturing domain and outline the challenges associated with them. 'Preliminaries' section provides an introduction to the background of causal structure learning, while 'Methodology' section demonstrates the proposed methods. In 'Experiments' section, we present simulation experiments and an application that analyzes a real dataset from Seagate Technology to explore the operating characteristics of the proposed methods. Finally, we conclude the paper with a discussion in 'Conclusions' section.

Data characteristics and challenges

The data in our case study come from a wafer assembly line in Seagate Technology and was collected from January 1, 2019, to October 11, 2021. The data exhibit the following characteristics, posing several challenges in learning the causal structures.

1. *High dimensionality*: The assembly line consists of more than 500 steps, each involving processing or measuring. Each process step records hundreds of sensor values, whereas each metrology step takes ten types of abnormality measurements. To make the learned causal structure useful for error tracing, we need a fine granularity and establish causal relations that capture which abnormal event or sensor at a particular step is the cause of an abnormality at another step. However, such a fine granularity requires the dataset used in causal discovery algorithms to contain all the variables from all steps, resulting in tens of thousands of variables. Existing modifications targeting high-dimensional settings, such as Fast Greedy Equivalence Search-Markov Blanket (FGES-MB) (Ramsey et al., 2017) and adaptively restricted greedy equivalence search (ARGES) (Nandy et al., 2018), can handle thousands of variables or millions of Gaussian variables with sparse graph structures. However, they are unsuitable for our case where the data is mixed and the dimension is much greater than the thousands level.
2. *Multimodality*: The data are derived from two distinct sources: sensor data and metrology data. The sensor data consist of multiple sub-datasets, each associated with a specific machine and recording the sensor values as wafers pass through the machine. On the other hand, the metrology data capture ten different types of abnormalities observed at each metrology step, with each abnormality represented by a binary variable. The sensor data are continuous in nature, whereas the metrology data are binary.
   For such a mixed data type, methods designed for a single type, such as PC (Spirtes et al., 2000) and greedy equivalence search (GES) (Chickering, 2002), are not applicable. Existing constraint-based mixed causal discovery methods, such as Copula-PC (Cui et al., 2016), symmetric conditional independence tests (Tsagris et al., 2018), causal mixed graphical models (CausalMGM) (Sedgewick et al., 2019), and Kernel Alignment PC algorithm (KAPC) (Handhayani & Cussens, 2020), do not apply to high-dimensional situations due to their high computational complexity. Score-based methods, such as the Conditional Gaussian (CG) score and the Mixed Variable Polynomial (MVP) score (Andrews et al., 2018), require a large sample size in high dimensions and lack computational efficiency. Andrews et al. (2019) later proposed the degenerate Gaussian (DG) score, which is consistent and efficient in high-dimensional settings. However, this approach must be more flexible in incorporating constraints and is hence unsuitable (see detailed discussion in Appendix C).
3. *Imbalance*: In a well-functioning assembly line, abnormalities rarely occur, which means that the binary variables in the metrology data are imbalanced. When applied to imbalanced data, many learning algorithms are biased towards the majority group (Krawczyk, 2016), which makes the estimated effects of binary variables too weak to be significant. In causal structure learning, this implies that the learned structure will mainly capture the relations among the continuous variables and miss a significant proportion of those regarding binary variables, as shown in Table 2. Moreover, since multiple imbalanced binary variables exist, common downsampling or oversampling techniques for classification (Chawla et al., 2002) are not applicable. Downsampling or oversampling one variable might lead to no information on the other variables. This

problem has been pointed out in fields like atmospheric science (Barnes et al., 2019) and Earth systems (Runge et al., 2019), but effective methods have yet to be proposed in the literature as far as we know.

4. *Missingness*. Ideally, wafers move along the assembly line following the prespecified recipe, but in practice, they often skip some steps based on their states. Wafers skip different steps from each other irregularly, which leads to an unstructured missing pattern.
   Following Little and Rubin (2019), there are three missing types: missing completely at random (MCAR), missing at random (MAR), and missing not at random (MNAR). In the MCAR case, two straightforward approaches apply to missing data: simply deleting examples with at least one missing value or performing imputation on the missing values. The former may lead to downgraded performance with a reduced sample size due to reduced statistical power (Städler & Bühlmann, 2012; Tu et al., 2019), whereas the latter may introduce biases in modeling the data distribution (Kyono et al., 2021).
   Researchers have made much progress in handling missing data in causal discovery in recent years. Sokolova et al. (2017) proposed a method to handle mixed and MAR data simultaneously. However, they assumed the continuous variables follow a non-paranormal distribution, and their estimation was based on the computationally expensive expectation–maximization (EM) algorithm (Dempster et al., 1977). Tu et al. (2019) proposed Missing Value PC (MVPC) that has been shown to be asymptotically correct even on data that are MNAR, but just like the PC algorithm, it cannot handle high-dimensional and mixed data. Regarding the MCAR case, Gao et al. (2022) proposed MissDAG upon additive noise models. However, it only applies to continuous data, and its estimation involves Monte Carlo EM, which can be time-consuming in high dimensions.
   The wafer data in our case study could contain variables of all three types, and due to the enormous problem scale, it is hard to identify the missing type for each variable. None of the existing methods suits our needs perfectly. If we make a compromise regarding the missing type and consider all the missing variables are MCAR and try the naive approach, which simply deletes the samples with at least one missing value, the challenge remains in that the merged data contains too many variables which might be missing, and after the deletion, no data are left.
5. *Temporal & Domain Constraints*. The production process imposes a natural temporal order onto measurements, suggesting the so-called causal order of occurrences of two events. Moreover, each step measures its data values simultaneously, making it sensible to assume the absence of contemporaneous effects. Namely, the variables at the same step are not causally related. In addition, certain causal relationships are not plausible based on domain knowledge. For example, two far-away nodes in the causal graph should have no linkage; see Appendix B for details. Properly leveraging these temporal and domain constraints can reduce the causal structure space complexity and improve the quality of structure learning. Failing to do so would lead to factual errors in the learned structure. However, incorporating constraints into some advanced algorithms (Andrews et al., 2019) is not straightforward.

Exploring causal relationships in wafer manufacturing is not a novel research area. However, most previous studies have approached the problem under simplified data settings, such as focusing on a small segment of the manufacturing process or assuming complete and continuous data. For instance, Abu-Samah et al. (2015) utilized GES combined with Minimum Description Length (MDL) score (Lam & Bacchus, 1994) to learn the structure, but their data consisted of only one module from the Thermal Treatment machine, which encompassed 10 process steps. Similarly, Sim et al. (2014) focused on a small segment of the manufacturing line, with 10 process steps only. Another study by Yang and Lee (2012) investigated an even smaller scale, examining just one process step involving 18 variables. Johnston et al. (2008) tackled a slightly larger-scale problem involving a total of 450 variables, using genetic algorithms (Sastry et al., 2005) combined with linear regression. However, the authors did not utilize the sensor data or address challenges related to missing data and constraints incorporation. Marazopoulou et al. (2016) also worked on a large-scale problem involving 69,000 variables, all continuous. They incorporated the temporal constraints by modifying the PC algorithm. Nonetheless, they did not address the challenges of mixed data and missing data. To the best of our knowledge, we are the first to acknowledge all of the aforementioned five challenges in the context of wafer manufacturing. In the following sections, we propose tailored methods to strike a balance in addressing these challenges simultaneously.

Preliminaries

Directed graphical causal models

Here, we use directed graphical causal models (DGCMs) to represent causal relations. A DGCM consists of three main components: (1) a set of variables $\mathbf{X}$, which corresponds to a set of nodes in a graph G; (2) a set of directed edges E in the graph G; (3) a joint probability distribution characterized by

$$P(\mathbf{X}) = \prod_{i=1}^{p} P\left(X_i \mid \mathrm{Pa}(G, X_i)\right), \tag{1}$$

where $\text{Pa}(G, X_i)$ is the set of all parents of X_i in the graph.

To interpret causal relations, we require that X_i is a direct cause of X_j if $X_i \rightarrow X_j \in E$, that is, an intervention on X_i changes the distribution of X_j when keeping the other variables fixed (Glymour et al., 2019). For a directed acyclic graph (DAG), learning causal relations from observational data requires three assumptions (Pearl, 2009; Scutari & Denis, 2021; Spirtes et al., 2000).

1. *Causal Markov.* Each variable $X_i \in \mathbf{X}$ is conditionally independent of its nondescendants given its parents.
2. *Causal Faithfulness.* There must exist a DAG faithful to the joint probability distribution P, meaning that only the dependencies arising from d-separation (Pearl, 2009) in the DAG can appear in P.
3. *Causal Sufficiency.* There cannot be any unmeasured variables acting as confounding factors.

Causal discovery

Causal discovery works on learning the causal structure out of observational data, and there are generally three types of approaches: constraint-based, score-based, and functional causal model (FCM) based. Constraint-based algorithms, such as PC (Spirtes et al., 2000) and Fast Causal Inference (FCI) (Spirtes et al., 2000), estimate the graph skeleton by conditional independence tests and orient the edges following some rules. Score-based algorithms use heuristics to learn the causal structure by maximizing a score function that quantifies the goodness of fit of the graphical network, for example, the Bayesian Information Criterion (BIC) score (Maxwell Chickering & Heckerman, 1997) and the Bayesian Dirichlet equivalent uniform (BDe) score (Heckerman et al., 1995). Finally, FCM-based algorithms, such as the Linear Non-Gaussian Acyclic Model (LiNGAM) (Shimizu et al., 2006), parameterize a functional causal model in the form of $X_i = f(\text{Pa}(G, X_i), e_i)$ and define a graph from the estimated model.

Methodology

As discussed in 'Data characteristics and challenges' section, no existing approaches can handle the five challenges at the same time. Therefore, we need to make some compromises and try to balance these aspects. In the design of our proposed method, scalability and the capability to handle missingness is of top priority, followed by constraint incorporation and mixed data modeling.

In the rest of the section, we first discuss data preprocessing and how to incorporate the constraints. Then, we introduce three constrained causal structure learning methods for mixed data and describe the hierarchical ensemble modeling strategy. Finally, we describe three model evaluation metrics and the overall pipeline in production.

Data preprocessing

Apart from some primary data preprocessing described in Appendix A, we apply principal component analysis (PCA) to alleviate high-dimensionality. At each process step, we perform PCA to generate m_{step} principal components to reduce the sensor data dimension. These principal components would then be the sensor-related nodes in the learned graph. To trace back to a specific sensor, we keep a record of the principal components' loading matrix to identify which sensors are the most relevant given a principal component associated with an abnormality's cause.[2]

Incorporating constraints

To incorporate the temporal and domain constraints, we transform them into implausible causal relations. For each step on the assembly line, set a unique time stamp[3] associated with the sensor and abnormality variables to represent the manufacturing order on the assembly line. Let $\mathbf{X}_t$ represent all the variables at time t. The temporal constraints then imply that the variables at time $t + j$ cannot be the causes of those at time t, $\forall j \geq 0$, namely, $\{X \rightarrow X' \mid \forall X \in \mathbf{X}_t, \forall X' \in \mathbf{X}_{t'}, t \geq t' \geq 1\}$ are implausible. Similarly, we derive the implausible links based on the domain knowledge in Appendix B. Different algorithms use these implausible relations differently, as explained subsequently.

Constrained causal structure learning on mixed data

To treat the issue of mixed data types, we propose three causal structure learning methods: Discretization-based learning, Regression-based learning, and Constrained Kernel Alignment PC (CKAPC). Specifically, the first method discretizes all continuous variables and uses existing methods to learn a discrete causal network. The second fits node-wise regularized generalized linear regressions and then combines the selected variables to form the global causal structure. Finally, the third applies the idea of kernel alignment to calculate a pseudo-correlation matrix on the mixed data and then feeds it into the PC algorithm (Spirtes et al., 2000) to learn the causal structure.

[2] Recall that the loading matrix of PCA contains the weights for each original variable when calculating the principal components, so we can check the absolute weight values in the loading matrix to identify which sensors are contributing the most to the principal component.

[3] Note that the time stamp here is different from that in time-series causal discovery in that each time stamp here represents the manufacturing order of different steps and includes different sets of sensors or abnormalities whereas time-series causal discovery focuses on the same set of variables that change over time.

Although more advanced mixed data causal discovery methods exist, they are either inflexible with constraints incorporation (Andrews et al., 2019) or computationally expensive in high-dimensions (Andrews et al., 2018; Cui et al., 2016). The three proposed methods are sub-optimal but allow for convenient constraint insertion and represent three schemes for distilling constraints. Examining all three would give us a more comprehensive idea of the suitable modeling choice.

Discretization-based learning

Given mixed data, one straightforward idea is to discretize all continuous variables into factors and then learn a discrete DAG. Discretization allows us to employ existing methods and improves computational speed, with the price of losing potentially important information. Commonly used discretization methods include quantile discretization, interval discretization, and information-preserving discretization (Scutari & Denis, 2021).

After discretizing the data, we apply the existing methods[4] to learn a discrete network. To ensure the constraints are satisfied, we input the list of implausible relations as the blacklist argument in the learning algorithms (Scutari & Denis, 2021).

Regression-based learning

Recall that regression usually cannot determine the causes and effects due to the bi-directionality of correlation. For example, if a regression model $Y \sim X$ is significant on X, then the causal relation can be either $X \to Y$ or $Y \to X$ given that there are no confounders. Traditional FCM-based methods, such as LiNGAM (Shimizu et al., 2006), need to assume the non-Gaussian noise and apply independent component analysis (ICA) (Hyvarinen, 1999) to find the causal order. However, with prior knowledge of $t_X < t_Y$, we can infer that only $X \to Y$ is plausible. Therefore, temporal constraints open the door to learning causal relations via regressions.

As in Sedgewick et al. (2019), we assume the data follow a Mixed Graphical Model (MGM) (Lee & Hastie, 2015), where Gaussian linear and logistic regression specifies the conditional distributions. Given the conditional distributions, we identify the conditional independence directly from regression coefficients, with nonzero indicating conditional dependence. For each variable X_i, which corresponds to a node in the causal graph, we solve a regularized conditional log-likelihood problem subjective to constraints:

$$\min_{\theta} \ \ell\left(\theta; X_i \mid \mathbf{X}_{\setminus i}\right) + \lambda \cdot Penalty(\theta),$$
$$\text{s.t. } \theta_j = 0 \text{ if } X_j \to X_i \text{ is implausible by constraints,} \quad (2)$$

where $\ell(\cdot)$ is the log-likelihood function, θ_j is the coefficient for variable $X_j \in \mathbf{X}_{\setminus i}$, $Penalty(\cdot)$ is the penalty function that induces sparsity,[5] and λ is the tuning parameter for the penalty.[6] Then, we estimate the parent set as $\widehat{Pa}(X_i) = \left\{X_j : \hat{\theta}_j \neq 0\right\}$. After fitting the regression models for all variables, we obtain the overall structure as $\widehat{E} = \cup_{X_i \in \mathbf{X}} \cup_{X_j \in \widehat{Pa}(X_i)} \{(X_j \to X_i)\}$.

Constrained kernel alignment PC (CKAPC)

We adopt the KAPC method of Handhayani and Cussens (2020). First, we apply the RBF kernel for continuous variables and the Categorical Kernel (Belanche et al., 2013) for binary variables while transforming each variable into a single Gaussian variable in the feature space. Then we apply the idea of kernel alignment to build a pseudo-correlation matrix. The original KAPC method considers no constraints, whereas CKAPC reinforces the constraints by modifying the PC algorithm.

Starting with a complete graph, we remove the contemporaneous edges and then orient the edges by temporal constraints, after which we remove implausible ones from domain constraints. On top of that, we then perform conditional independence tests using the pseudo-correlation matrix to remove more edges and get the ultimate structure. Since all edges are oriented temporally, we do not have to use the orientation rules as in the traditional PC algorithm.

Hierarchical ensemble modeling

As mentioned in 'Data characteristics and challenges' section, although existing causal discovery methods for missing data can handle the MCAR (Sokolova et al., 2017; Gao et al., 2022) and even the MNAR case (Tu et al., 2019), they are usually computationally expensive and do not fit high-dimensional and mixed data. As a compromise, we assume all missing variables are MCAR and employ a straightforward approach that deletes examples with at least one missing value. However, suppose we try to learn the causal structure in one run and use the overall merged data, which contains all the variables from all steps. Then, after applying the missing deletion strategy, we are likely to end up with few examples.[7] In response, we propose a hierarchical modeling strategy, which learns the causal structure at two granularity levels. In addition, we fuse in the ensemble idea similar to that in

[4] See Chapter 6 of Scutari and Denis (2021) for a comprehensive review.

[5] Common choices include LASSO (Tibshirani, 1996), the smoothly clipped absolute deviation (SCAD) (Fan & Li, 2001) penalty, the minimax concave penalty (MCP) (Zhang, 2010), and the truncated L_1 penalty (TLP) (Shen et al., 2013).

[6] We tune λ using cross-validation in each node-wise regression model.

[7] In our case study, we get no examples left at all, making it impossible to learn the structure.

random forest (Breiman, 2001) to alleviate the imbalance problem and to provide uncertainty measures for the learned relations. The modeling framework is shown in Fig. 2.

Hierarchical Modeling. There are three main steps in hierarchical modeling: block-level learning, step-level learning, and aggregation. To perform this modeling strategy, domain knowledge is required in terms of which steps along the assembly line can be clustered into a block.[8] The block-level learning will capture which two blocks are causally related, whereas step-level learning is finer-grained and captures which two sensor/abnormality variables are causally related. Moreover, once we learn the block-level structure, we can perform the step-level learning in parallel and then union all step-level results to get the final structure.

1. *Block-level Data Processing and Structure Learning.* After discussing with expert engineers and establishing n_{block} blocks, for each block, we aggregate the metrology data such that as long as a certain type of abnormality occurs inside this block, no matter from which step, we label the corresponding abnormality type indicator to be 1. And the sensor data are aggregated by PCA with m_{block} principal components. The processed data for each block contains ten abnormality variables and m_{block} sensor variables. Next, we merge the n_{block} block-level data sets into one big data set, with rows representing wafers and columns representing abnormality or sensor variables from blocks. This data set has enough records for learning in that although few wafers go through all steps, many wafers pass through all blocks. Based on this data, we apply the proposed constrained mixed-data causal structure learning methods to learn a causal graph where each node is an abnormal event or a sensor-related variable in a certain block. Finally, we map this graph to a block-level graph G_{block} where each node represents a block.
2. *Step-level Data Processing and Structure Learning.* For the jth block b_j, $1 \leq j \leq n_{\text{block}}$, we pick out its parent set in G_{block} and denote it as $Pa(G_{\text{block}}, b_j)$.
   We merge the data from all steps inside blocks $\{b_j\} \cup Pa(G_{\text{block}}, b_j)$. Since we are merging a subset of steps, we get much more observation records than merging across all steps. Based on the step-level merged data,
   we then learn a step-level causal structure $G_{\text{step}}^{(b_j)}$, where each node represents an abnormality variable or a sensor-related variable from some step.
3. *Combination of Step-level Causal Structures.* In the end, we combine all the learned step-level causal structures and get the ultimate causal structure $\bigcup_{1 \leq j \leq n_{\text{block}}} G_{\text{step}}^{(b_j)}$.

This scheme alleviates both the missingness and the high-dimensional problem. Instead of merging all steps and obtaining a small sample with high dimensions, we only merge the steps inside a subset of blocks. This subset consists of significantly fewer steps, resulting in a substantial number of remaining records after removing missing data, and the merged data has a reduced dimensionality. Moreover, the hierarchical scheme grants us two levels of causal structures, providing the engineers with more insights. Meanwhile, it is worth noting that the quality of this scheme highly depends on the block division given by the engineers, so a thorough conversation with the engineers is crucial.

Ensemble Modeling. In manufacturing, including extra relations costs additional engineers' validation time, whereas missing true relations leads to false root cause diagnosis, which is much more costly. Also, missing block-level relations in hierarchical modeling leads to accumulative errors in subsequent step-level modeling. Therefore, we prefer conservative estimates and want to control the false negative rate. We employ stability selection (Meinshausen & Bühlmann, 2006) and bagging (Breiman, 1996) to construct an ensemble model for both the block-level and the step-level structure learning. First, we resample the original data with replacement to generate K new data sets, each of the original data sample size. Then, we learn a causal structure G_k on the kth resampled data for $\forall k \in \{1, 2, \cdots, K\}$. Finally, different from the threshold aggregation in Meinshausen and Bühlmann (2006) and mean aggregation in Breiman (1996), we union these K learned structures and obtain $\cup_{k=1}^{K} G_k$ as the ultimate causal structure. The temporal constraint enforced on the structures guarantees the directions of causal relations.

There are three benefits of ensemble modeling. First, it helps alleviate the bias towards the majority group and the weak signal problem due to imbalanced data. With ensemble modeling, we have many resampled data, some of which might magnify the weak signals of certain imbalanced variables. Every single model acts as a weak learner to capture the partial information, granting the ensemble model more capability to detect signals. Second, the estimate is stabler with the union aggregation, which is crucial for engineers' validation and technicians' usage. Third, it allows us to quantify the strength of the learned relations. As in (3), we define the strength of a causal relation e in $\cup_{k=1}^{K} G_k$ as the proportion it appears in the K learned structures. Such a measure not only quantifies uncertainty[9] but also helps with valida-

[8] The process steps within a specific stage should be consolidated into the same block. During the creation of these blocks, engineers usually conduct clustering at the stage level and subsequently determine the assignment of individual steps to their respective blocks.

[9] The greater the strength, the higher our confidence and the less uncertainty in the learned relation.

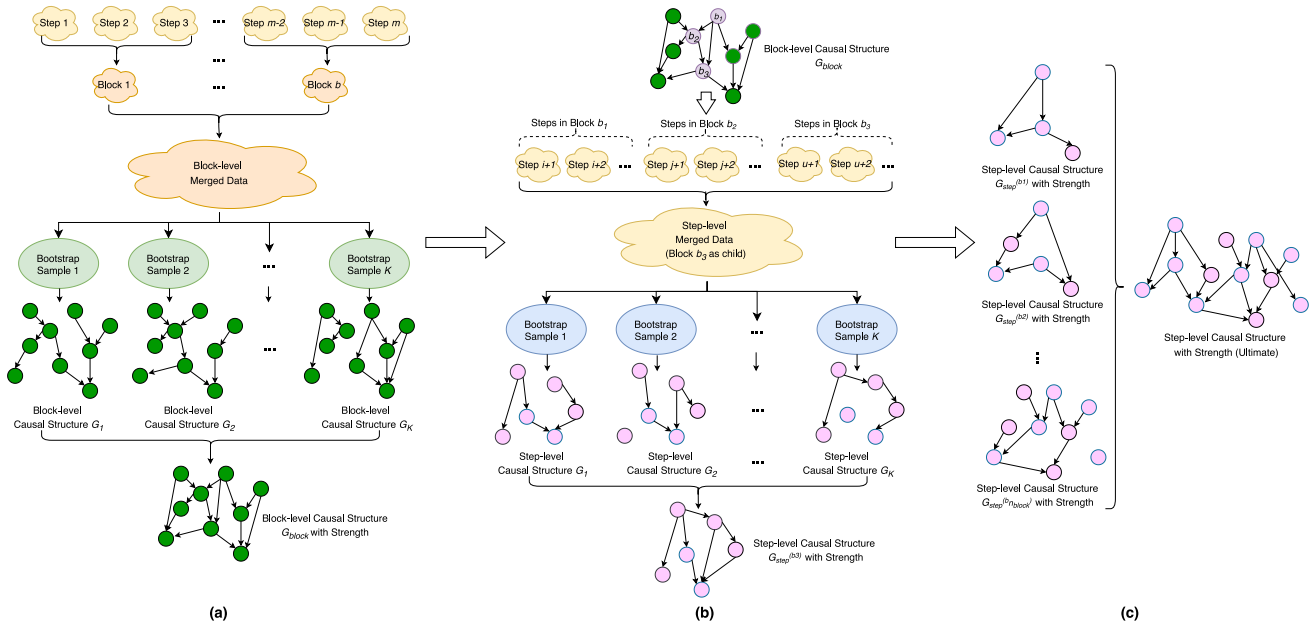

Fig. 2 Hierarchical ensemble modeling. (**a**) for block-level data processing and structure learning, (**b**) for step-level data processing and structure learning, and (**c**) for union aggregation of step-level structures

tion. Given a large number of learned causal relations to be fully validated, engineers can start by checking the ones with strengths above a threshold and provide feedback to the modelers timely. In this way, the pipeline loop runs at a faster pace. Also, with the reduced workload, the engineers would be more willing to get involved, a factor we must consider in a real-life production environment.

$$s_e = \frac{\sum_{k=1}^{K} \mathbf{1}\,(e \in G_k)}{K}, \quad \forall e \in \cup_{k=1}^{K} G_k. \tag{3}$$

Evaluation

We propose three metrics to evaluate the results in real production, collectively assessing a method's performance.

1. *Engineers' Validation*. Given the learned causal relations and associated strengths, domain engineers validate the causal relations with strengths higher than a prespecified threshold.
2. *Comparison against Known Knowledge*. Given some known relations on which two steps are causally related, we validate the results by transforming the learned relations to step-wise relations (if $\exists X \in \text{step}_i, Y \in \text{step}_j$ s.t. $X \to Y$, then the corresponding step-wise relation is $\text{step}_i \to \text{step}_j$).
3. *Conditional Independence Tests*. Given a learned network, we select a few nodes of interest and test if the conditional independence relationship given by d-separation (Pearl, 2009) is the same as that by conditional independence tests on the data. Specifically, if S d-separates X from Y according to the learned graph, implying $X \perp\!\!\!\perp Y|S$, but the conditional independence tests tell us that $X \not\perp\!\!\!\perp Y|S$, then this relationship is not trustworthy. We consider three conditional independence tests for the mixed data, including Pearson X^2 test, the symmetric conditional independence test (Tsagris et al., 2018), and the Conditional Dependence Coefficient (CODEC) measure (Azadkia & Chatterjee, 2021).

Modeling pipeline in production

The overall modeling pipeline is a positive feedback loop, as shown in Fig. 3. First, we query data from the databases and consult expert engineers to establish the domain constraints. Second, we apply hierarchical ensemble modeling to learn the causal relations among the variables along the assembly line. Third, we validate the learned results through comparisons with the known causal relations and via engineers' validation. When the engineers determine an identified relationship to be definite, we add it to the inclusion (if true) or the exclusion (if false) databases. Otherwise, we take no action. Afterward, the engineers update the constraints based on these included and excluded relations.

This approach applies to online and offline data. For a streaming pipeline, we dynamically update the database and causal structures. Otherwise, we circle through the pipeline for several rounds by updating the constraints and rerunning the structure learning model. As a result, we keep improving the accuracy of the learned structure. The inclusion and exclusion databases also serve as a knowledge base for technicians to trace the root causes of failure events. In some

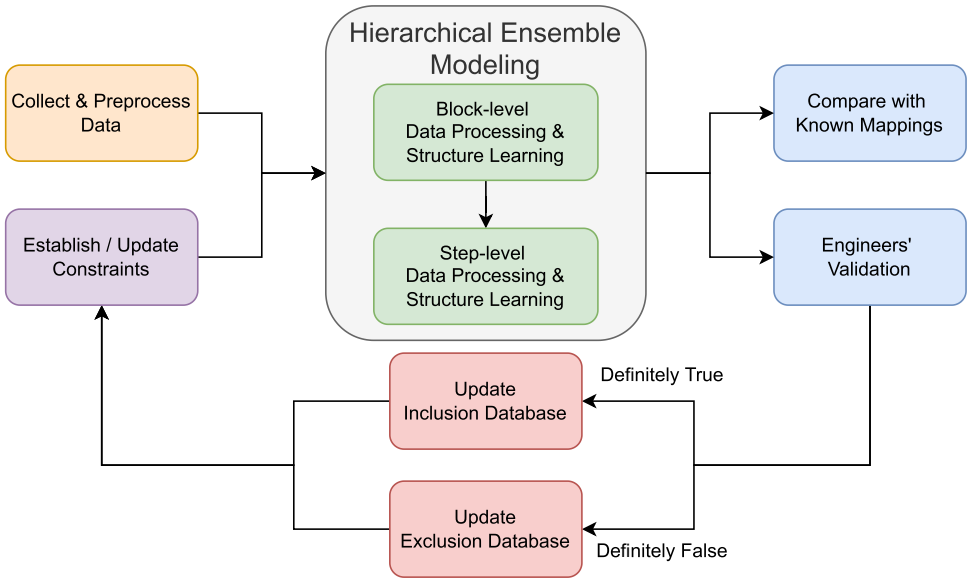

Fig. 3 Modeling Pipeline in Production

situations, the technicians can check the databases to resolve an issue without consulting an expert engineer.

Experiments

To validate the efficacy of the proposed method, we conduct simulation studies and apply it to real-world data.

Simulation studies

We conduct three simulation experiments to examine the performance of structure learning methods on mixed data, the influence of ensembles on data imbalance, and the effect of hierarchical modeling. As discussed in 'Data characteristics and challenges' section and Appendix C, existing methods for mixed data are either not scalable (Cui et al., 2016; Handhayani & Cussens, 2020) or inflexible in incorporating constraints (Andrews et al., 2019). Furthermore, to the best of our knowledge, no approaches have been proposed to simultaneously address the challenges of imbalanced, high-dimensional, and missing data in causal discovery. Hence, we do not compare our methods with other approaches. Instead, we focus on investigating the operational characteristics of the proposed methods under different scenarios to enhance understanding and provide guidance to practitioners.

Effects of mixed data structure learning methods

We first examine the proposed constrained mixed-data structure learning methods. The simulation mimics the assembly line subject to temporal constraints. We vary the sample size $n \in \{100, 300, 600, 6000\}$, the number of variables $p \in \{20, 50, 100, 600\}$, and the number of steps $k \in \{2, 5, 10, 60\}$, while fixing the number of variables in each step to be 10, the vertex degree to be 4, and the proportion of continuous variables to be 0.2.[10] We first generate graphs with the constraints enforced and then generate the data following the MGM framework (Lee & Hastie, 2015). All simulations run for 30 repetitions with different seeds and are implemented on a Linux server with an Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz and 124GB RAM.

We compare six methods, including three discretization-based methods: *tabu-bic* (Tabu search (Glover, 1989, 1990) with BIC score), *tabu-bde* (Tabu search with BDe score), and *pc.stable* (PC-Stable algorithm (Colombo & Maathuis, 2014)); two regression-based methods: *lasso-min* (Lasso-penalized regressions (Tibshirani, 1996) with $\lambda_{\min}$ in cross-validation) and *lasso-1se* (Lasso-penalized regressions with λ_{1se} in cross-validation); and one CKAPA method ($\sigma = 0.01$ for the RBF kernel; $\theta = 1$ for the Categorical kernel).

Four metrics are employed to assess the model performance: adjacency precision (AP), adjacency recall (AR) (Jia et al., 2022), Structural Hamming Distance (SHD) (Tsamardinos et al., 2006), and elapsed time. Given the true graph $\mathcal{G}_{\text{true}}$ and the learned graph $\mathcal{G}_{\text{learn}}$, denote their respective binary lower-triangular adjacency matrices as $\mathcal{A}_{\text{true}}$ and $\mathcal{A}_{\text{learn}}$. AP and AR are defined as the precision and recall of these two adjacency matrices, as shown in (4) and (5), respectively. On the other hand, SHD quantifies the number of edge insertions, deletions, or flips required to transform

[10] We pick these numbers to mimic the real data in our case study.

$\mathcal{G}_{\text{true}}$ into $\mathcal{G}_{\text{learn}}$.[11]

$$\begin{aligned} \text{AP} &= \frac{\#\text{ edges correctly identified in } \mathcal{G}_{\text{learn}}}{\#\text{ edges in } \mathcal{G}_{\text{learn}}} \\ &= \frac{\sum_{i,j} (\mathcal{A}_{\text{true}} \odot \mathcal{A}_{\text{learn}})_{ij}}{\sum_{i,j} (\mathcal{A}_{\text{learn}})_{ij}} \end{aligned} \tag{4}$$

$$\begin{aligned} \text{AR} &= \frac{\#\text{ edges correctly identified in } \mathcal{G}_{\text{learn}}}{\#\text{ edges in } \mathcal{G}_{\text{true}}} \\ &= \frac{\sum_{i,j} (\mathcal{A}_{\text{true}} \odot \mathcal{A}_{\text{learn}})_{ij}}{\sum_{i,j} (\mathcal{A}_{\text{true}})_{ij}} \end{aligned} \tag{5}$$

We report the mean performance and their standard deviations (in parentheses) in Table 1. When $n \geq p$, discretization-based learning generally achieves the highest performance in terms of AP. On the other hand, regression-based learning tends to outperform others in terms of AR, which aligns with our expectations considering the tendency of Lasso to over-select variables. Across all settings, *tabu-bic* and *pc.stable* consistently rank among the top two algorithms in terms of SHD. In terms of runtime, *tabu-bic* is typically the fastest, especially in high-dimensional scenarios, while CKAPC proves to be too slow for all tasks except Setting 5. When $n < p$, as demonstrated in Settings 7 and 8, *pc.stable* significantly outperforms the other algorithms in terms of AP, although it comes with a considerable increase in time cost.

Effects of ensemble on data imbalance

We also investigate how the ensemble strategy alleviates the weak signal problem due to data imbalance. We simulate the data as before, with $n = 3000$, $p = 200$, $k = 20$, and an additional hyper-parameter controlling the imbalance ratio. All simulations run ten repetitions with different seeds and are implemented in parallel on 20 CPU cores. We use *tabu-bic* as the base model and set the number of weak learners $K = 1000$ in ensemble models. We compare two model types: single models and ensemble models with a strength threshold $\alpha \in \{0, 0.05, 0.1, \ldots, 0.95, 1\}$. The focus is to reduce the false negative rate, so we consider different types of AR while controlling the overall AP.

Table 2 shows the mean performance and their standard deviations with $\alpha \in \{0, 0.05, 0.1, 0.15, 0.2\}$[12] under six imbalanced scenarios. We can see that data imbalance impacts the AR, particularly those relating to discrete nodes. If we can tolerate a low AP, then *ensem_0* (ensemble model with no threshold) greatly helps identify the edges the base model easily misses. If, however, we want to control the AP at an acceptable level, then $\alpha \in [0.05, 0.2]$ would be a good choice, and a smaller α gives higher ARs but a lower AP.

Effects of hierarchical modeling

We now investigate the effects of hierarchical modeling. The data are simulated similarly to before, except that we now impose a hierarchical structure by adding a few hyper-parameters: the number of blocks n_{block}, *missing_rate* which specifies the probability that a step is missing,[13] *block_tightness*, which controls the relative closeness of the within-block versus between-block relations, and *max_step_distance*, which specifies the maximum steps apart for two connected nodes. We set $n = 2000$, $p = 600$, $k = 60$, $n_{\text{block}} = 10$, *max_step_distance* $= 30$, the imbalance ratio to be 0.1, *missing_rate* $\in \{0, 0.02, 0.05, 0.1\}$, and *block_tightness* $\in \{0.3, 0.5, 0.7, 0.9, 1.0\}$. We use *tabu-bic* as the base model, set the number of weak learners K = 1000 in ensemble models, and perform hierarchical learning with strength threshold $\alpha = 0.2$ at both the block level and the step level. We examine both levels' precision, recall, SHD, and elapsed time. All simulations run ten repetitions with different seeds and are implemented in parallel on 20 CPU cores.

Table 3 shows the mean performance and their standard deviations. As the missing rate increases, the block-level recovery does not change much, whereas the step-level learning accuracy decreases significantly. When the missing rate is 0.02, close to the real data scenario, the step-level AP and AR are acceptable, implying the applicability to the real data. In addition, we note that high block tightness usually associates with low block-level recovery accuracy and high step-level AP.[14] In contrast, the step-level AR is pretty robust against the change in block tightness. As discussed in 'Hierarchical ensemble modeling' section, we usually prefer to control the false negative rate in manufacturing. Hence, step-level AR is the most important out of all the metrics. Its robustness against block tightness is desirable because even if the steps are not well clustered, the final fine-grained AR will not be affected much.

[11] For the algorithmic implementation of SHD, interested readers can refer to the code available at https://github.com/cran/pcalg/blob/afe34d671144292350173b6f534c0eeb7fd0bb90/R/pcalg.R#L1086.

[12] For clarity, we do not show all the results here. Refer to Yang (2023) for full results with other thresholds and other types of AP.

[13] To mimic the case study data, if a step is missing, then all variables in this step are missing. Moreover, we set the probability to be the same for all wafers and all steps in the simulation.

[14] This is as expected. For example, when block tightness is 1, only the initial node of each block is connected with other blocks, and hence the between-block relations are too weak to be well detected. Furthermore, once we get into the step level, the high value of block tightness implies smaller candidate parent node sets, and hence a sparser learned graph and a higher step-level AP.

Table 1 Simulation results on constrained mixed data causal structure learning methods

Setting	Method	AP ↑[a]	AR ↑	SHD ↓	Elapsed time[b] ↓
Setting 1 (n=600 p=100 k=10)	tabu-bic	0.863 (0.037)	0.497 (0.036)	**88.6 (10.988)**	**0.693 (0.063)**
	tabu-bde	0.848 (0.049)	0.478 (0.032)	92.567 (12.016)	0.709 (0.072)
	pc.stable	**0.882 (0.041)**	0.475 (0.038)	89.867 (10.67)	5.087 (0.523)
	lasso-min	0.26 (0.02)	**0.735 (0.039)**	360.467 (47.963)	15.064 (2.428)
	lasso-1se	0.664 (0.056)	0.601 (0.042)	107.733 (14.249)	15.985 (4.25)
Setting 2 (n=6000 p=100 k=10)	tabu-bic	**0.967 (0.018)**	0.742 (0.038)	**43.3 (7.452)**	**2.843 (0.291)**
	tabu-bde	0.963 (0.021)	0.73 (0.04)	45.333 (7.941)	2.901 (0.267)
	pc.stable	0.929 (0.025)	0.759 (0.038)	45.833 (8.91)	7.207 (0.947)
	lasso-min	0.259 (0.021)	**0.911 (0.022)**	412.167 (44.772)	65.127 (5.172)
	lasso-1se	0.878 (0.056)	0.814 (0.035)	45.767 (10.251)	65.065 (5.106)
Setting 3 (n=600 p=600 k=60)	tabu-bic	0.702 (0.016)	0.497 (0.012)	799.133 (25.704)	**40.766 (4.482)**
	tabu-bde	0.686 (0.022)	0.478 (0.011)	828.8 (28.602)	41.905 (3.814)
	pc.stable	**0.796 (0.019)**	0.435 (0.017)	**756.767 (30.514)**	5449.623 (142.001)
	lasso-min	0.171 (0.006)	**0.695 (0.014)**	4110.5 (178.198)	1888.124 (187.423)
	lasso-1se	0.48 (0.027)	0.599 (0.015)	1177.133 (79.856)	1904.291 (170.195)
Setting 4 (n=6000 p=600 k=60)	tabu-bic	**0.908 (0.011)**	0.746 (0.011)	**368.567 (15.174)**	**117.23 (13.325)**
	tabu-bde	0.9 (0.014)	0.73 (0.01)	392.267 (17.066)	119.978 (13.454)
	pc.stable	0.901 (0.011)	0.716 (0.014)	405.667 (22.202)	5792.206 (203.995)
	lasso-min	0.184 (0.008)	**0.898 (0.011)**	4580.933 (247.047)	4804.765 (587.206)
	lasso-1se	0.782 (0.033)	0.826 (0.013)	453.433 (54.219)	4772.73 (565.064)
Setting 5 (n=6000 p=20 k=2)	tabu-bic	**0.997 (0.014)**	0.748 (0.135)	2.9 (1.826)	0.363 (0.041)
	tabu-bde	0.99 (0.032)	0.748 (0.144)	2.967 (2.025)	0.35 (0.041)
	pc.stable	0.972 (0.047)	**0.797 (0.098)**	**2.633 (1.671)**	**0.329 (0.029)**
	lasso-min	0.408 (0.224)	0.762 (0.355)	13.2 (9.264)	2.916 (1.354)
	lasso-1se	0.799 (0.37)	0.649 (0.315)	3.533 (1.795)	2.898 (1.373)
	ckapc[c]	0.228 (0.099)	0.769 (0.19)	36.833 (21.001)	277.488 (345.531)
Setting 6 (n=6000 p=50 k=5)	tabu-bic	**0.984 (0.023)**	0.745 (0.068)	**15.533 (3.875)**	**0.896 (0.104)**
	tabu-bde	0.979 (0.028)	0.732 (0.067)	16.5 (4.125)	0.915 (0.116)
	pc.stable	0.936 (0.044)	0.782 (0.055)	15.833 (4.684)	1.083 (0.159)
	lasso-min	0.316 (0.04)	**0.916 (0.049)**	122.3 (22.426)	15.409 (1.862)
	lasso-1se	0.91 (0.056)	0.807 (0.06)	15.8 (3.8)	15.423 (1.636)
Setting 7 (n=300 p=600 k=60)	tabu-bic	0.705 (0.021)	0.442 (0.013)	**846.333 (34.262)**	**23.337 (0.557)**
	tabu-bde	0.683 (0.034)	0.428 (0.012)	879.1 (44.167)	23.818 (0.86)
	pc.stable	**0.8 (0.026)**	0.307 (0.014)	876.9 (28.672)	3360.496 (44.588)
	lasso-min	0.159 (0.007)	**0.734 (0.02)**	4737.233 (289.529)	224.692 (31.804)
	lasso-1se	0.49 (0.035)	0.632 (0.022)	1177.367 (117.802)	250.967 (34.115)
Setting 8 (n=100 p=600 k=60)	tabu-bic	0.434 (0.016)	0.288 (0.008)	1240.167 (28.314)	**22.949 (1.188)**
	tabu-bde	0.334 (0.022)	0.286 (0.007)	1466.133 (56.033)	44.685 (13.178)
	pc.stable	**0.651 (0.029)**	0.159 (0.007)	**1055.433 (19.199)**	3220.144 (40.112)
	lasso-min	0.136 (0.006)	**0.566 (0.019)**	4613.5 (236.002)	53.01 (1.916)
	lasso-1se	0.382 (0.02)	0.452 (0.022)	1460 (74.525)	55.262 (1.816)

The entries highlighted in bold indicate the top performers within the group for their respective metrics

[a] ↑ means the higher the better, ↓ means the lower the better.

[b] In seconds.

[c] Settings 1–4 and Setting 6 have no *ckapc* results as CKAPC is too slow to be finished

Table 2 Simulation results on the effect of ensemble on data imbalance

Imbalance ratio[a]	Model	Overall AP	Overall AR	To discrete AR[b]	From discrete AR[c]	From discrete to discrete AR[d]	To continuous AR[e]	From continuous AR[e]	From continuous to continuous AR[e]
0.5051 (0.0246)	ensemble_0[f]	0.0594 (0.0044)	0.8805 (0.008)	0.9048 (0.0164)	0.8614 (0.0207)	0.8961 (0.0261)	0.8583 (0.0131)	0.9017 (0.0169)	0.8941 (0.0314)
	ensemble_0.05	0.5178 (0.0242)	0.7909 (0.0147)	0.7903 (0.0217)	0.7364 (0.0344)	0.7308 (0.0516)	0.792 (0.0185)	0.8493 (0.028)	0.8492 (0.0354)
	ensemble_0.1	0.7183 (0.0199)	0.7741 (0.0182)	0.7689 (0.0261)	0.7131 (0.0363)	0.7017 (0.0533)	0.7795 (0.02)	0.8397 (0.0283)	0.843 (0.035)
	ensemble_0.15	0.8214 (0.0193)	0.7646 (0.0187)	0.757 (0.0242)	0.6985 (0.0364)	0.685 (0.052)	0.7718 (0.0225)	0.8355 (0.029)	0.8414 (0.033)
	ensemble_0.2	0.8729 (0.018)	0.7584 (0.0168)	0.7531 (0.0219)	0.6883 (0.0364)	0.6797 (0.0485)	0.764 (0.02)	0.8337 (0.0293)	0.8404 (0.0328)
	single[g]	0.9379 (0.0124)	0.7056 (0.0206)	0.6965 (0.0276)	0.6229 (0.0399)	0.6127 (0.0488)	0.715 (0.0178)	0.7943 (0.024)	0.8078 (0.0273)
0.2524 (0.0099)	ensemble_0	0.0514 (0.0039)	0.8673 (0.013)	0.8866 (0.0199)	0.8502 (0.0236)	0.8839 (0.0337)	0.8494 (0.0139)	0.8856 (0.0265)	0.8871 (0.0277)
	ensemble_0.05	0.4794 (0.0267)	0.7432 (0.0272)	0.7084 (0.037)	0.6721 (0.0455)	0.6267 (0.0626)	0.7778 (0.0221)	0.8192 (0.0309)	0.8492 (0.0299)
	ensemble_0.1	0.696 (0.0209)	0.7148 (0.0322)	0.6648 (0.039)	0.6328 (0.0467)	0.5645 (0.0577)	0.764 (0.0261)	0.8034 (0.0352)	0.8397 (0.0252)
	ensemble_0.15	0.816 (0.0173)	0.7 (0.0299)	0.6427 (0.0367)	0.6152 (0.0454)	0.5388 (0.0566)	0.7561 (0.0227)	0.7915 (0.0312)	0.8344 (0.0243)
	ensemble_0.2	0.8789 (0.0168)	0.6881 (0.0316)	0.6201 (0.0368)	0.5985 (0.0441)	0.507 (0.0509)	0.7543 (0.0238)	0.7849 (0.0332)	0.8344 (0.0243)
	single	0.9359 (0.0125)	0.6347 (0.0273)	0.5503 (0.0274)	0.5356 (0.0426)	0.4294 (0.0546)	0.7164 (0.0244)	0.7418 (0.0257)	0.8111 (0.0325)
0.1005 (0.0034)	ensemble_0	0.0423 (0.0025)	0.8526 (0.0135)	0.8793 (0.0223)	0.81 (0.028)	0.8531 (0.0394)	0.8278 (0.0255)	0.8994 (0.0209)	0.8966 (0.0285)
	ensemble_0.05	0.3965 (0.0203)	0.6408 (0.0286)	0.5396 (0.0235)	0.5148 (0.0352)	0.3815 (0.0431)	0.7376 (0.0287)	0.7767 (0.0306)	0.8515 (0.0239)
	ensemble_0.1	0.611 (0.024)	0.6003 (0.038)	0.4741 (0.0362)	0.4626 (0.0499)	0.2984 (0.0566)	0.7218 (0.0358)	0.7498 (0.031)	0.8425 (0.0203)
	ensemble_0.15	0.7421 (0.0217)	0.5792 (0.0417)	0.4391 (0.0359)	0.4356 (0.0548)	0.2558 (0.0511)	0.7136 (0.038)	0.7353 (0.0318)	0.8383 (0.0209)
	ensemble_0.2	0.8124 (0.0222)	0.5614 (0.0413)	0.414 (0.0374)	0.4133 (0.0544)	0.2244 (0.0545)	0.7029 (0.0373)	0.7226 (0.0292)	0.8319 (0.0209)
	single	0.9002 (0.0215)	0.5136 (0.0262)	0.3482 (0.0204)	0.3599 (0.0371)	0.1641 (0.0334)	0.6725 (0.0333)	0.6808 (0.0224)	0.8187 (0.0264)
0.0503 (0.0018)	ensemble_0	0.0387 (0.0025)	0.8159 (0.0262)	0.8395 (0.0305)	0.7478 (0.0304)	0.7954 (0.0387)	0.7941 (0.0356)	0.89 (0.0311)	0.9006 (0.0259)
	ensemble_0.05	0.3438 (0.0192)	0.5534 (0.0265)	0.3922 (0.0249)	0.3859 (0.033)	0.1824 (0.0419)	0.7075 (0.0324)	0.7334 (0.0342)	0.8559 (0.0293)
	ensemble_0.1	0.5584 (0.0198)	0.5089 (0.0291)	0.324 (0.0251)	0.3371 (0.0319)	0.1136 (0.0337)	0.6849 (0.0352)	0.6943 (0.0237)	0.8428 (0.0221)
	ensemble_0.15	0.7016 (0.0193)	0.4872 (0.0316)	0.2911 (0.0263)	0.3135 (0.0346)	0.0857 (0.0231)	0.6742 (0.0334)	0.675 (0.0216)	0.8406 (0.0216)
	ensemble_0.2	0.7851 (0.0178)	0.4683 (0.0316)	0.2654 (0.022)	0.2932 (0.0322)	0.0646 (0.0173)	0.6622 (0.033)	0.6577 (0.0277)	0.8371 (0.0215)
	single	0.8665 (0.0293)	0.4231 (0.0225)	0.2188 (0.0273)	0.2476 (0.022)	0.0395 (0.0191)	0.6186 (0.0265)	0.613 (0.028)	0.8148 (0.0237)

Table 2 continued

Imbalance ratio[a]	Model	Overall AP	Overall AR	To discrete AR[b]	From discrete AR[c]	From discrete to discrete AR[d]	To continuous AR[e]	From continuous AR[e]	From continuous to continuous AR[e]
0.0252 (0.001)	ensemble_0	0.0372 (0.0024)	0.7431 (0.0245)	0.7148 (0.0311)	0.6057 (0.0351)	0.5517 (0.0469)	0.7719 (0.0283)	0.8919 (0.018)	0.9033 (0.0272)
	ensemble_0.05	0.2904 (0.0237)	0.472 (0.031)	0.2825 (0.0172)	0.2794 (0.031)	0.0814 (0.0288)	0.6536 (0.0355)	0.6804 (0.0316)	0.8638 (0.0246)
	ensemble_0.1	0.4736 (0.0306)	0.432 (0.03)	0.2295 (0.0197)	0.2413 (0.0283)	0.0451 (0.0212)	0.6267 (0.034)	0.6379 (0.0287)	0.8505 (0.0224)
	ensemble_0.15	0.6231 (0.0394)	0.4091 (0.0317)	0.1974 (0.0206)	0.2235 (0.0276)	0.0301 (0.0143)	0.6124 (0.0361)	0.6095 (0.0348)	0.8434 (0.0275)
	ensemble_0.2	0.7246 (0.0363)	0.3949 (0.0326)	0.1743 (0.0253)	0.2131 (0.0248)	0.0176 (0.0138)	0.6067 (0.0338)	0.5907 (0.0373)	0.8408 (0.0251)
	single	0.8595 (0.0318)	0.3454 (0.0295)	0.1298 (0.0231)	0.163 (0.0204)	0.0058 (0.0079)	0.5512 (0.0326)	0.5417 (0.0334)	0.8188 (0.0243)
0.017 (0.0006)	ensemble_0	0.0383 (0.0024)	0.7029 (0.0203)	0.6448 (0.0319)	0.5328 (0.0262)	0.4346 (0.0555)	0.7583 (0.0207)	0.886 (0.0219)	0.9103 (0.023)
	ensemble_0.05	0.2687 (0.0203)	0.4386 (0.0318)	0.2413 (0.0346)	0.2445 (0.0275)	0.0665 (0.0328)	0.6285 (0.0325)	0.6478 (0.0288)	0.8692 (0.0275)
	ensemble_0.1	0.4565 (0.0329)	0.396 (0.0311)	0.1833 (0.029)	0.2087 (0.0252)	0.0388 (0.0175)	0.6008 (0.0384)	0.5976 (0.0286)	0.8578 (0.0282)
	ensemble_0.15	0.5971 (0.0383)	0.3729 (0.0273)	0.153 (0.0302)	0.1862 (0.0205)	0.0232 (0.0101)	0.5838 (0.0348)	0.5736 (0.0287)	0.8543 (0.0282)
	ensemble_0.2	0.7031 (0.0424)	0.3548 (0.0266)	0.1333 (0.0302)	0.1705 (0.0194)	0.0146 (0.0103)	0.567 (0.0336)	0.5535 (0.0333)	0.8446 (0.0266)
	single	0.8553 (0.0337)	0.3147 (0.0328)	0.0958 (0.024)	0.1356 (0.0167)	0.0069 (0.0077)	0.5239 (0.0339)	0.5072 (0.0451)	0.8208 (0.0231)

[a]The proportion of the minority in binary variables.
[b]The AR for edges pointing to discrete nodes.
[c]The AR for edges starting from discrete nodes.
[d]The AR for edges with discrete nodes on two ends.
[e]The continuous counterparts have similar meanings as the discrete ones.
[f]*ensem_α* represents an ensemble model with threshold α.
[g]*single* represents a model without ensemble

Table 3 Simulation results on the effect of hierarchical modeling

Missing rate	Block tightness[a]	Block-level AP ↑[b]	Block-level AR ↑	Block-level SHD ↓	Step-level[c] AP ↑	Step-level AR ↑	Step-level SHD ↓	Elapsed time[d] ↓
0	0.3	**0.9358 (0.0476)**	**0.8109 (0.0556)**	**7.5 (2.6771)**	0.6248 (0.032)	0.5873 (0.0236)	660.8 (103.7827)	1.911 (0.4652)
	0.5	0.9124 (0.0499)	0.7472 (0.0462)	9.5 (2.0138)	0.65 (0.0278)	0.5898 (0.0256)	623.3 (79.0725)	2.229 (0.4902)
	0.7	0.8689 (0.0671)	0.7172 (0.1105)	10.8 (3.4897)	0.6854 (0.0228)	**0.5907 (0.0189)**	572.5 (40.4949)	0.8922 (0.3326)
	0.9	0.7585 (0.1001)	0.6938 (0.0912)	12.3 (3.3682)	0.7132 (0.0295)	0.582 (0.0223)	544.5 (43.2518)	**0.7701 (0.1766)**
	1	0.5047 (0.0645)	0.7098 (0.1787)	14.2 (3.1903)	**0.7256 (0.0245)**	0.5765 (0.0195)	**534.2 (38.8667)**	0.8945 (0.3986)
0.02	0.3	**0.9302 (0.052)**	**0.7755 (0.0535)**	**8.7 (2.2632)**	0.6377 (0.0297)	0.5225 (0.0275)	667.4 (95.6617)	1.5303 (0.3182)
	0.5	0.9062 (0.053)	0.7632 (0.0398)	9.3 (1.2517)	0.645 (0.0306)	0.5195 (0.0311)	654.5 (68.1929)	0.7342 (0.2213)
	0.7	0.8548 (0.0599)	0.7558 (0.0777)	10.4 (2.9515)	0.6814 (0.0213)	**0.5392 (0.022)**	599.4 (31.959)	1.1568 (0.307)
	0.9	0.7547 (0.0961)	0.691 (0.1114)	12.5 (2.9155)	0.7024 (0.0266)	0.5328 (0.0286)	**578.4 (43.5155)**	0.8314 (0.1795)
	1	0.536 (0.0871)	0.7274 (0.1206)	13.2 (2.8206)	**0.7074 (0.0332)**	0.516 (0.0329)	581.7 (44.7811)	**0.444 (0.13)**
0.05	0.3	**0.9373 (0.0446)**	0.7869 (0.0533)	8.1 (2.2336)	0.6118 (0.0284)	0.4254 (0.0353)	726.3 (91.6528)	0.55 (0.1023)
	0.5	0.9182 (0.0476)	**0.8045 (0.0582)**	**7.9 (2.0248)**	0.6245 (0.0196)	0.4227 (0.0371)	710.7 (66.7833)	1.2852 (0.2887)
	0.7	0.8969 (0.0498)	0.7472 (0.0864)	9.6 (2.4129)	0.6274 (0.0174)	0.4353 (0.0393)	691.8 (47.1659)	0.7723 (0.1202)
	0.9	0.7602 (0.0934)	0.6649 (0.1171)	12.6 (3.4705)	**0.6596 (0.0446)**	0.4432 (0.0331)	**655.9 (54.1489)**	0.3723 (0.0819)
	1	0.5313 (0.0765)	0.712 (0.1051)	13.2 (2.044)	0.6556 (0.0365)	**0.4434 (0.0369)**	657.3 (55.094)	**0.3517 (0.1105)**
0.1	0.3	**0.9327 (0.0509)**	0.7754 (0.0708)	8.6 (2.9515)	0.3785 (0.0645)	0.2455 (0.0312)	998.0 (98.1439)	0.3582 (0.0588)
	0.5	0.9243 (0.052)	**0.7904 (0.058)**	**8.1 (1.912)**	0.4198 (0.0317)	0.2502 (0.0168)	937.3 (80.0514)	0.4015 (0.0585)
	0.7	0.9064 (0.0787)	0.7566 (0.076)	9.0 (3.266)	0.4468 (0.0347)	0.2669 (0.034)	892.5 (68.6541)	0.461 (0.1066)
	0.9	0.7865 (0.0929)	0.6972 (0.0583)	11.5 (1.7795)	0.4739 (0.0451)	0.2967 (0.048)	860.2 (80.5161)	**0.2762 (0.0512)**
	1	0.5199 (0.0591)	0.7145 (0.1541)	13.5 (2.2236)	**0.4833 (0.0664)**	**0.3111 (0.0557)**	**850.2 (98.363)**	0.328 (0.1026)

The entries highlighted in bold indicate the top performers within the group for their respective metrics

[a] Probability that a non-initial node's parent comes from the same block. Higher values imply tighter within-block relations relative to between-block relations.

[b] ↑ means the higher the better, ↓ means the lower the better.

[c] Although the name is step-level, each node in the graph represents a sensor or an abnormality indicator instead of a step.

[d] In hours. Includes data simulation, block-level data processing and learning, step-level data processing and learning, aggregation, and evaluation

Real data analysis

We adhered to the pipeline illustrated in Fig. 3 and employed our proposed methods to analyze the wafer data obtained from Seagate Technology. The dataset encompasses information collected from metrology tools and 24 machines, which collectively form the complete assembly line and consist of 125 stages and 592 steps. The data incorporate a total of 2766 sensors,[15] ten fault types, and a total of 66996 wafers. All ensemble models, with $K = 1000$, were executed on a Linux server with an Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz and 124GB RAM, utilizing 30 CPU cores in parallel.

We performed data preprocessing as described in 'Data preprocessing' section, and set m_{step} to 4 for PCA on the sensor data. For block-level data processing, the engineers aggregate all the steps into $n_{\text{block}} = 68$ blocks. We set m_{block} to 4, creating merged block-level data with 8026 rows and 653 columns. We then applied the ensemble *tabu-bic* method, which has the shortest runtime and recovers the most known mappings from both simulations and analyses. The runtime for this phase was 2.25 hours. Finally, we discovered 1460 block-wise relations and identified 32 out of 46 known block-wise relationships.

After processing the data at the step level, we obtained 68 step-level datasets. We used the ensemble *tabu-bic* method to learn structures from these datasets and combined them to obtain the ultimate causal structure. The total time required for this process was approximately 6 hours. We discovered 9774 relationships among the abnormality and sensor variables, corresponding to 1439 step-wise relationships. Moreover, out of 56 known step-wise relationships, we recovered 29 of them.

The engineers examined the causal relations with a strength of 1 (totaling 122), of which approximately 50% were found to be true, 20% were likely true but required more context to verify, and nearly 30% were likely false. The 50% true relations were added to the inclusion database, while the 20% likely true and 30% likely false relations suggest that we have discovered unestablished relationships that could enhance the engineers' understanding of the manufacturing system with further investigation.

We also developed a zoomable network visualization tool that allows engineers and technicians to visualize and manipulate the causal structure. For example, Fig. 4 displays the causal structure filtered with a strength threshold of 0.8, while Fig. 5 shows a zoomed-in subgraph. Each node is labeled with a machine name (if it is a sensor node), stage, step, temporal order, and variable name, while each edge is labeled with a strength value indicating its credibility.

Fig. 4 Ultimate Learned Causal Structure (strength ≥ 0.8)

Conclusions

This article presents a causal structure learning approach for wafer manufacturing data that addresses several challenges, such as mixed data types, missing entries, high dimensionality, data imbalance, and temporal and domain constraints. We propose three constrained mixed-data models and a hierarchical ensemble strategy to handle these challenges simultaneously. Our simulation studies demonstrate that the discretization-based method performs the best in accuracy and time cost and that the ensemble models are more effective at capturing weak signals with imbalanced data. Moreover, the hierarchical modeling strategy performs well under reasonable missing rates and produces robust fine-grained recall rates against the hierarchy setup. Finally, our hierarchical ensemble approach helps alleviate problems with data missing, high dimensionality, and data imbalance, and we have validated its effectiveness with an application to a wafer manufacturing data set.

Although we focus on wafer manufacturing, the challenges and data characteristics we address are not unique to this field. Therefore, our proposed approach can be applied to other domains with similar challenges. However, it is essential to note that our approach aims to balance all five challenges and may not be optimal for addressing a single aspect. Moreover, our proposal profoundly depends on the existence of temporal constraints. For example, the regression-based method and the union aggregation in the ensemble modeling would be invalid in situations without global temporal order. Additionally, when a high AR rate is

[15] Since the sensors can be activated multiple times for different process steps within the same machine, the actual number of sensor variables is much larger.

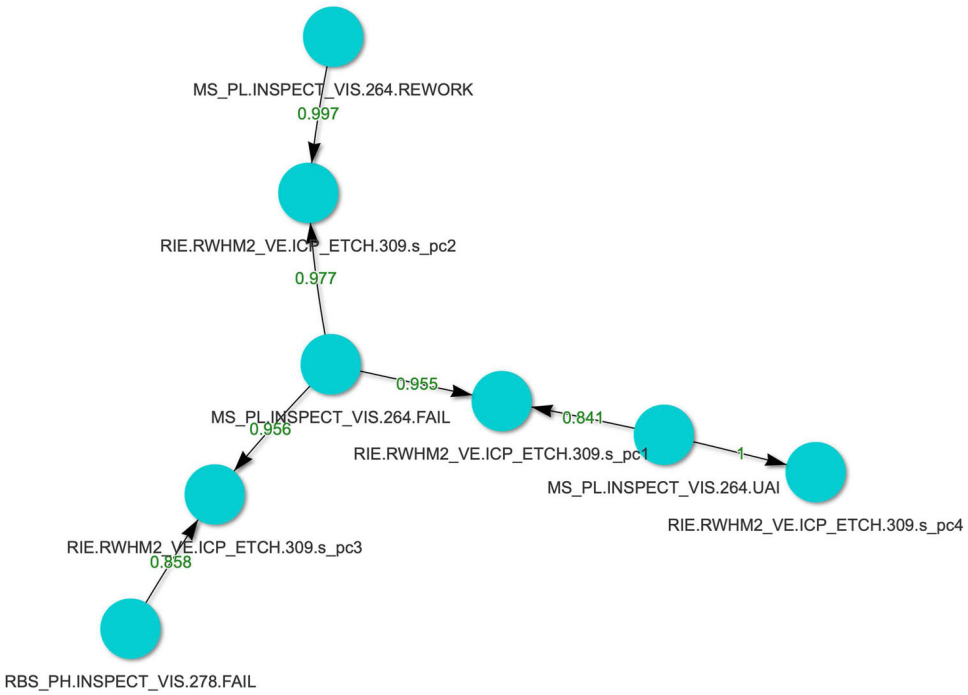

Fig. 5 A Zoomed-in Subgraph of Fig. 4. s_pc*k* represents the *k*th principal component of sensors and the green numbers represent the strengths

a top priority, and the dimension is not very high, regression-based methods may be the preferred option at the cost of increasing computing time. Therefore, we recommend that practitioners carefully examine the data characteristics before selecting a modeling method.

We have observed that most existing work focuses on solving a single problem, whereas industrial applications often require integrated approaches that handle multiple aspects simultaneously. Therefore, further research is necessary to design integrated methods for such applications.

Acknowledgements We thank Seagate Technology for providing the data and Govi Veeraraghavan and Karen Terry for their time and effort in domain support and validation. This work was supported in part by a grant from Seagate Technology, in part by the National Science Foundation (NSF) Division of Mathematical Sciences (DMS) Award 1952539, and in part by National Institutes of Health (NIH) Grants R01AG069895, R01AG065636, R01AG074858, U01AG07307.

Funding This work was supported in part by a grant from Seagate Technology, in part by the National Science Foundation (NSF) Division of Mathematical Sciences (DMS) Award 1952539, and in part by National Institutes of Health (NIH) grants R01AG069895, R01AG065636, R01AG074858, U01AG07307.

Declarations

Competing interest The authors have no competing interests to declare that are relevant to the content of this article.

Ethics approval Not applicable.

Consent to participate Yes.

Consent for publication Yes.

Availability of data and materials The real datasets analyzed in this study are not publicly available due to confidential company data by Seagate Technology. The full result file for simulations in 'Effects of ensemble on data imbalance' section is available in the Harvard Dataverse repository at https://doi.org/10.7910/DVN/ONTUOL.

Code availability Code for generating the simulated datasets is available from the corresponding author upon request.

Appendix A: Data preprocessing

The metrology and sensor data require basic preprocessing, such as removing missing, duplicate, and uninformative records. For the metrology data, the abnormal events at every step are a long list of strings, so we extract the abnormalities by textual processing and then transform them into binary variables, with one indicating abnormal. And for the sensor data, since they are collected every 3 seconds, and most values do not change much within a step, we aggregate them by averaging. We use the mean instead of the median because the latter is robust against outliers and insensitive to abnormalities.

Appendix B: Constraints by domain knowledge

1. For a node from 'MEAS(2)_MON', its parent must come from the same stage, and if its child node is also from 'MEAS(2)_MON', then the steps in between must be 'MEAS*' or 'INSPECT'.
2. Sensors from different tools cannot be linked.

3. If a child node does not come from 'MEAS*', 'INSPECT', or 'TEST*', then either its stage is '*_VE' or its step is one of ['DRYETCH', 'LAP_EBARA', 'ASH'].
4. Two nodes that are 60 steps apart cannot be linked.
5. The child node cannot be from 'INSPECT'.

Appendix C: Applicability analysis of the DG score-based approach

The degenerate Gaussian (DG) score-based approach (Andrews et al., 2019) lacks flexibility when it comes to incorporating constraints. Specifically, DG is implemented in Tetrad,[16] which has two versions: Tetrad-GUI and Tetrad-CLI. Tetrad-GUI relies on the manual input of nodes and edges for constraint incorporation, making it challenging to integrate with hierarchical modeling and pipeline modeling based on programming languages including R, Python, and SQL. The manual input requirement for domain knowledge makes it unsuitable for automatic causal learning. On the other hand, Tetrad-CLI allows for domain knowledge incorporation through a file, but it currently does not support the DG score. Therefore, despite DG demonstrating consistency and efficiency in high-dimensional scenarios, it does not apply to our specific case.

References

Abu-Samah, A., Shahzad, M., Zamai, E., et al. (2015). Failure prediction methodology for improved proactive maintenance using Bayesian approach. *IFAC-PapersOnLine, 48*(21), 844–851. https://doi.org/10.1016/j.ifacol.2015.09.632

Andrews, B., Ramsey, J., & Cooper, G. F. (2018). Scoring Bayesian networks of mixed variables. *International Journal of Data Science and Analytics, 6*(1), 3–18. https://doi.org/10.1007/s41060-017-0085-7

Andrews, B., Ramsey, J., & Cooper, G. F. (2019). Learning high-dimensional directed acyclic graphs with mixed data-types. In *The 2019 ACM SIGKDD workshop on causal discovery, PMLR* (pp. 4–21). http://proceedings.mlr.press/v104/andrews19a.html.

Azadkia, M., & Chatterjee, S. (2021). A simple measure of conditional dependence. *The Annals of Statistics, 49*(6), 3070–3102. https://doi.org/10.1214/21-AOS2073

Barnes, E. A., Samarasinghe, S. M., Ebert-Uphoff, I., et al. (2019). Tropospheric and stratospheric causal pathways between the mjo and nao. *Journal of Geophysical Research: Atmospheres, 124*(16), 9356–9371. https://doi.org/10.1029/2019JD031024

Belanche, M. L.A., & Villegas, M. (2013). Kernel functions for categorical variables with application to problems in the life sciences. In *Artificial intelligence research and development—Proceedings of the 16th international conference of the Catalan association for artificial intelligence, Vic, Catalonia, Spain, October 23–25, 2013, Frontiers in Artificial Intelligence and Applications* (Vol. 256, pp 171–180). IOS Press, https://doi.org/10.3233/978-1-61499-320-9-171.

Breiman, L. (1996). Bagging predictors. *Machine Learning, 24*(2), 123–140. https://doi.org/10.1007/BF00058655

Breiman, L. (2001). Random forests. *Machine Learning, 45*, 5–32. https://doi.org/10.1023/A:1010933404324

Chawla, N. V., Bowyer, K. W., Hall, L. O., et al. (2002). Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research, 16*, 321–357. https://doi.org/10.1613/jair.953

Chickering, D. M. (2002). Optimal structure identification with greedy search. *Journal of Machine Learning Research, 3*(Nov), 507–554. http://jmlr.org/papers/v3/chickering02b.html.

Colombo, D., Maathuis, M. H., et al. (2014). Order-independent constraint-based causal structure learning. *Journal of Machine Learning Research, 15*(1), 3741–3782. https://doi.org/10.5555/2627435.2750365

Cui, R., Groot, P., & Heskes, T. (2016). Copula pc algorithm for causal discovery from mixed data. In *Joint European conference on machine learning and knowledge discovery in databases* (pp. 377–392). Springer. https://doi.org/10.1007/978-3-319-46227-1_24.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological), 39*(1), 1–22. https://doi.org/10.1111/j.2517-6161.1977.tb01600.x

Fan, J., & Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association, 96*(456), 1348–1360. https://doi.org/10.1198/016214501753382273

Gao, E., Ng, I., Gong, M., et al. (2022). Missdag: Causal discovery in the presence of missing data with continuous additive noise models. https://doi.org/10.48550/arXiv.2205.13869. arXiv:2205.13869

Gharahbagheri, H., Imtiaz, S., Khan, F., et al. (2015). Causality analysis for root cause diagnosis in fluid catalytic cracking unit. *IFAC-PapersOnLine, 48*(21), 838–843. https://doi.org/10.1016/j.ifacol.2015.09.631

Glover, F. (1989). Tabu search—Part I. *ORSA Journal on Computing, 1*(3), 190–206. https://doi.org/10.1287/ijoc.1.3.190

Glover, F. (1990). Tabu search—Part II. *ORSA Journal on Computing, 2*(1), 4–32. https://doi.org/10.1287/ijoc.2.1.4

Glymour, C., Zhang, K., & Spirtes, P. (2019). Review of causal discovery methods based on graphical models. *Frontiers in Genetics, 10*, 524. https://doi.org/10.3389/fgene.2019.00524

Handhayani, T., & Cussens, J. (2020). Kernel-based approach for learning causal graphs from mixed data. In *International conference on probabilistic graphical models, PMLR* (pp. 221–232). http://proceedings.mlr.press/v138/handhayani20a.html.

Heckerman, D., Geiger, D., & Chickering, D. M. (1995). Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning, 20*(3), 197–243. https://doi.org/10.1007/BF00994016

Huegle, J., Hagedorn, C., & Uflacker, M. (2020). How causal structural knowledge adds decision-support in monitoring of automotive body shop assembly lines. In Bessiere, C. (Ed.), *Proceedings of the twenty-ninth international joint conference on artificial intelligence, IJCAI-20. International joint conferences on artificial intelligence organization*. https://doi.org/10.24963/ijcai.2020/758.

Hyvarinen, A. (1999). Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks, 10*(3), 626–634. https://doi.org/10.1109/72.761722

Jeong, B., & Cho, H. (2006). Feature selection techniques and comparative studies for large-scale manufacturing processes. *The International Journal of Advanced Manufacturing Technology, 28*, 1006–1011. https://doi.org/10.1007/s00170-004-2434-7

Jia, M., Yuan, D. Y., Lovelace, T. C., et al. (2022). Causal discovery in high-dimensional, multicollinear datasets. *Frontiers in Epidemiology, 2*(899), 655. https://doi.org/10.3389/fepid.2022.899655

Johnston, A. B., Maguire, L., & Mcginnity, T. (2008). Disentangling causal relationships of a manufacturing process using genetic algorithms and six-sigma techniques. *International Journal of Pro-*

[16] https://github.com/cmu-phil/tetrad

duction Research, 46(22), 6251–6268. https://doi.org/10.1080/00207540701427029

Krawczyk, B. (2016). Learning from imbalanced data: Open challenges and future directions. *Progress in Artificial Intelligence, 5*(4), 221–232. https://doi.org/10.1007/s13748-016-0094-0

Kumar, P. (1993). Re-entrant lines. *Queueing Systems, 13*(1–3), 87–110. https://doi.org/10.1007/BF01158930

Kyono, T., Zhang, Y., Bellot, A., et al. (2021). Miracle: Causally-aware imputation via learning missing data mechanisms. *Advances in Neural Information Processing Systems, 34*, 23806–23817. https://proceedings.neurips.cc/paper/2021/hash/c80bcf42c220b8f5c41f85344242f1b0-Abstract.html.

Lam, W., & Bacchus, F. (1994). Learning Bayesian belief networks: An approach based on the mdl principle. *Computational Intelligence, 10*(3), 269–293. https://doi.org/10.1111/j.1467-8640.1994.tb00166.x

Landman, R., & Jämsä-Jounela, S. L. (2016). Hybrid approach to casual analysis on a complex industrial system based on transfer entropy in conjunction with process connectivity information. *Control Engineering Practice, 53*, 14–23. https://doi.org/10.1016/j.conengprac.2016.04.010

Lee, J. D., & Hastie, T. J. (2015). Learning the structure of mixed graphical models. *Journal of Computational and Graphical Statistics, 24*(1), 230–253. https://doi.org/10.1080/10618600.2014.900500

Liang, S. Y., Hecker, R. L., & Landers, R. G. (2004). Machining process monitoring and control: The state-of-the-art. *The Journal of Manufacturing Science and Engineering, 126*(2), 297–310. https://doi.org/10.1115/1.1707035

Little, R. J., & Rubin, D. B. (2019). *Statistical analysis with missing data* (Vol. 793). Wiley. https://doi.org/10.1002/9781119013563

Marazopoulou, K., Ghosh, R., Lade, P., et al. (2016). Causal discovery for manufacturing domains. arXiv:1605.04056. https://doi.org/10.48550/arXiv.1605.04056.

Maxwell Chickering, D., & Heckerman, D. (1997). Efficient approximations for the marginal likelihood of Bayesian networks with hidden variables. *Machine Learning, 29*(2), 181–212. https://doi.org/10.1023/A:1007469629108

Meinshausen, N., & Bühlmann, P. (2006). High-dimensional graphs and variable selection with the lasso. *Annals of Statistics, 34*(3), 1436–1462. https://doi.org/10.1214/009053606000000281

Nandy, P., Hauser, A., & Maathuis, M. H. (2018). High-dimensional consistency in score-based and hybrid structure learning. *The Annals of Statistics, 46*(6A), 3151–3183. https://doi.org/10.1214/17-AOS1654

Pearl, J. (2009). *Causality* (2nd ed.). Cambridge University Press. https://doi.org/10.1017/CBO9780511803161

Ramsey, J., Glymour, M., Sanchez-Romero, R., et al. (2017). A million variables and more: The fast greedy equivalence search algorithm for learning high-dimensional graphical causal models, with an application to functional magnetic resonance images. *International Journal of Data Science and Analytics, 3*(2), 121–129. https://doi.org/10.1007/s41060-016-0032-z

Runge, J., Bathiany, S., Bollt, E., et al. (2019). Inferring causation from time series in earth system sciences. *Nature Communications, 10*(1), 2553. https://doi.org/10.1038/s41467-019-10105-3

Sastry, K., Goldberg, D., & Kendall, G. (2005). *Genetic algorithms* (pp. 97–125). Springer. https://doi.org/10.1007/0-387-28356-0_4

Scutari, M., & Denis, J. B. (2021). *Bayesian networks: With examples in R*. Chapman and Hall/CRC. https://doi.org/10.1201/9780429347436

Sedgewick, A. J., Buschur, K., Shi, I., et al. (2019). Mixed graphical models for integrative causal analysis with application to chronic lung disease diagnosis and prognosis. *Bioinformatics, 35*(7), 1204–1212. https://doi.org/10.1093/bioinformatics/bty769

Shah, S. Y., Dang, X. H., & Zerfos, P. (2018). Root cause detection using dynamic dependency graphs from time series data. In *2018 IEEE international conference on big data (big data), IEEE* (pp 1998–2003). https://doi.org/10.1109/BigData.2018.8622059.

Shen, X., Pan, W., Zhu, Y., et al. (2013). On constrained and regularized high-dimensional regression. *Annals of the Institute of Statistical Mathematics, 65*(5), 807–832. https://doi.org/10.1007/s10463-012-0396-3

Shimizu, S., Hoyer, P. O., Hyvärinen, A., et al. (2006). A linear non-gaussian acyclic model for causal discovery. *Journal of Machine Learning Research, 7*(10). http://jmlr.org/papers/v7/shimizu06a.html.

Sim, H., Choi, D., & Kim, C. O. (2014). A data mining approach to the causal analysis of product faults in multi-stage PCB manufacturing. *International Journal of Precision Engineering and Manufacturing, 15*, 1563–1573. https://doi.org/10.1007/s12541-014-0505-8

Sokolova, E., von Rhein, D., Naaijen, J., et al. (2017). Handling hybrid and missing data in constraint-based causal discovery to study the etiology of ADHD. *International Journal of Data Science and Analytics, 3*, 105–119. https://doi.org/10.1007/s41060-016-0034-x

Spirtes, P., Glymour, C. N., Scheines, R., et al. (2000). *Causation, prediction, and search*. MIT Press. https://doi.org/10.7551/mitpress/1754.001.0001

Städler, N., & Bühlmann, P. (2012). Missing values: Sparse inverse covariance estimation and an extension to sparse regression. *Statistics and Computing, 22*, 219–235. https://doi.org/10.1007/s11222-010-9219-7

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological), 58*(1), 267–288. https://doi.org/10.1111/j.2517-6161.1996.tb02080.x

Tsagris, M., Borboudakis, G., Lagani, V., et al. (2018). Constraint-based causal discovery with mixed data. *International Journal of Data Science and Analytics, 6*(1), 19–30. https://doi.org/10.1007/s41060-018-0097-y

Tsamardinos, I., Brown, L. E., & Aliferis, C. F. (2006). The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning, 65*(1), 31–78. https://doi.org/10.1007/s10994-006-6889-7

Tu, R., Zhang, C., Ackermann, P., et al. (2019). Causal discovery in the presence of missing data. In *The 22nd international conference on artificial intelligence and statistics, PMLR* (pp. 1762–1770). http://proceedings.mlr.press/v89/tu19a.html.

Yang, L., & Lee, J. (2012). Bayesian belief network-based approach for diagnostics and prognostics of semiconductor manufacturing systems. *Robotics and Computer-Integrated Manufacturing, 28*(1), 66–74. https://doi.org/10.1016/j.rcim.2011.06.007

Yang, Y. (2023). *Simulation results on the effect of ensemble on data imbalance*. Harvard Dataverse. https://doi.org/10.7910/DVN/ONTUOL

Zhang, C. H. (2010). Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics, 38*(2), 894–942. https://doi.org/10.1214/09-AOS729