Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Long-term forecasting of multivariate time series in industrial furnaces with dynamic Gaussian Bayesian networks

David Quesada [*], Gabriel Valverde, Pedro Larrañaga, Concha Bielza

Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid, Madrid, Spain

ARTICLE INFO

MSC:
00-01
99-00

Keywords:
Long-term forecast
Furnace optimization
Dynamic Gaussian Bayesian networks
Multivariate time series
Fouling

ABSTRACT

Many of the data sets extracted from real-world industrial environments are time series that describe dynamic processes with characteristics that change over time. In this paper, we focus on the fouling process in an industrial furnace, which corresponds to a non-stationary multivariate time series with a seasonal component, non-homogeneous cycles and sporadic human interventions. We aim to forecast the evolution of the temperature inside the furnace over a long span of time of two and a half months. To accomplish this, we model the time series with dynamic Gaussian Bayesian networks (DGBNs) and compare their performance with convolutional recurrent neural networks. Our results show that DGBNs are capable of properly treating seasonal data and can capture the tendency of a time series without being distorted by the effect of interventions or by the varying length of the cycles.

1. Introduction

One of the most common types of data found in real-world problems in different areas is time series (TS) data (Aminikhanghahi and Cook, 2017). This is due to the proliferation of sensors that describe complex dynamic industrial processes over time. In addition, when there are several sensors generating data that describe the evolution of different parts of the same system, multivariate TS data sets are the result.

A common objective in industrial environments of this kind is to be able to forecast one or more of the variables that compose the system in order to optimize some of its aspects. The difference from the univariate case is that we must take into account not only the autoregressive component but also the influence of the TS that conform the multivariate case between each other.

An additional problem that TS can have is the presence of seasonality and non-stationarity (Cheng et al., 2015). Seasonality arises when a TS exhibits some kind of cyclic behaviour during its lifespan. A TS is stationary when its statistical properties, such as the mean and variance, do not depend on time t (Brockwell et al., 2002). In real-world situations, TS tend to be non-stationary because they have different trends or scaling variances as time t increases. This non-stationary component typically has to be identified and addressed in order to apply models such as the autoregressive integrated moving average (ARIMA) (Peña et al., 2011) to the data. The seasonal component, on the other hand, requires specific treatment depending on the kind of cycles present and their characteristics.

Inside an industrial furnace, one of the processes that costs a great deal of money in terms of efficiency loss is the deposition of solidified impurities of the fluid being processed in the tubes where it is preheated before entering the furnace (Pogiatzis et al., 2012). This process is called *fouling*, and it forces the furnace tubes to be cleaned periodically. Fouling has been treated extensively in the literature with different techniques, ranging from physical models defining the process to more data-driven approaches. Some of the more classical methods focus on developing physical equation models that simulate the fouling effect inside the tubes. In Diaz-Bejarano et al. (2019), the authors simulate the growth of the fouling layer and its thermal resistance with a physical model over long spans of time. A similar approach is used in both Pogiatzis et al. (2012) and Santamaria and Macchietto (2019), where the authors simulate the fouling effect with a physical model, but they reformulate it afterwards into a non-linear programming problem which they use to optimize the cleaning schedule that must be performed to remove the fouling layer over time. An alternative approach to this optimization is proposed by Diaby et al. (2016), where instead of non-linear programming they opt to use a genetic algorithm to find the optimal cleaning schedule while simulating the fouling effect. A very popular alternative to these kind of models is a data-driven approach, where data from simulations or from operating furnaces is used to fit machine learning models to capture the fouling effect. In particular, artificial neural networks (ANN) have found a lot of use in this area in the past. A feed-forward network with three hidden layers is used in Radhakrishnan et al. (2007), where they forecast the temperature of the fluid inside the tubes over a span of two weeks. This temperature is also an indicative of the fouling effect over time, given that it will harder to increase it as the fouling layer grows if no countermeasure

* Corresponding author.
E-mail address: dquesada@fi.upm.es (D. Quesada).

https://doi.org/10.1016/j.engappai.2021.104301
Received 2 June 2020; Received in revised form 5 April 2021; Accepted 11 May 2021
Available online 18 May 2021

is taken. Lalot et al. (2007) identify drifts in the temperature TS that are a result of the fouling effect and use a mixed approach where they either recommend forecasting with a recurrent two hidden layer ANN when this drift is sudden and with a Kalman filter if the drift appears slowly over time. A simpler approach is proposed by Davoudi and Vaferi (2018) with a feed-forward dense network with 10 hidden neurons distributed in two hidden layers. Instead of forecasting the temperatures over time, they use the network to obtain an index that approximates the thermal resistance of the current fouling layer in a single instant. Sundar et al. (2020) propose a similar approach, but they apply an ensemble of ANNs with two hidden layers to predict the fouling resistance based on the state of the system. A review of similar methods of fouling prediction in industrial furnaces is discussed in Wang et al. (2015). In our case, this problem can be seen as one of multivariate TS that have a non-stationary component as they follow an evolving trend over time and a seasonal component of non-homogeneous cycles recovered from the furnace. We will focus on the approach of forecasting the temperature inside the tubes to represent the fouling effect over time.

Fouling causes a decrease in the thermal conductivity of the tube walls in the furnace. As a result, as the fouling layer grows, we have to increase the heat provided to maintain the temperature of the fluid constant inside the tubes. When the temperature of these tube walls rises above a certain threshold, the efficiency loses become too severe and a cleaning must be performed to remove the fouling layer. If we are able to predict the temperature that we need to have the tube walls at in the long term, we can estimate the number of hours left until the next cleaning must be performed. In addition, given that we want to gain some insight on the fouling phenomenon and how the variables in our system affect each other, we propose the use of dynamic Gaussian Bayesian networks (DGBNs) (Murphy, 2002) for long term forecasting of the system evolution. We will treat the TS recovered from the sensors of the furnace as different nodes in our network and aim to model the conditional probabilistic independence relationships among them. Once we learn the structure of the network and its parameters from the data, the dynamic part of the DGBN will model the temporal component of the system. We can then forecast the state of the system and the temperature of the tube walls up to a certain horizon. With the resulting structure of the DGBN, we are able to understand which sensors in the system are more relevant and how the variables interact with each other. As opposed to a black-box model, we have a clear picture of which variables influence future predictions. In addition, DGBNs are a good tool for making long-term forecasts that take into account the evolution of the tendencies in TS, which is key in the problem at hand. To provide a comparison with another state-of-the-art model, we will also train a convolutional recurrent neural network (CRNN). Neural networks are very popular in the literature, and a recurrent network will allow us to perform forecasting over different spans of time.

The rest of the paper is organized as follows. Section 2 introduces the fouling problem. Section 3 describes the characteristics of a DGBN model. Section 4 explains the preprocessing performed on the data recovered from an industrial furnace. Section 5 is devoted to the learning and inference methods used for the DGBN, the results obtained and the comparison with the CRNN. Section 6 concludes the paper and describes future work.

2. Forecasting the temperature of an industrial furnace

Inside an industrial furnace, fluids are circulated through tubes and heated to some desired temperature to make them chemically reactive. In our case, the fouling effect degrades the heat capacity of the tubes over time by creating an insulating layer on their walls (Fig. 1). The fouling degradation forces the preheat train to use more energy to attain the same temperature inside the tube, up to a limit point where the melting temperature of the tube walls is nearly reached. At this stage, the heating costs are very high as a result of the degenerated heat transfer coefficient, and the desired temperature of the fluid cannot be obtained because of this physical limit. To solve this problem, some cleaning of the insulating deposits has to be performed, either by physical or chemical means, after which the tube walls revert to their initial state and the fouling process begins again. This is what gives rise to the seasonality in the corresponding TS. In addition, because the fouling process depends on many different factors, the limit point is not reached after the same number of hours every time, which creates non-homogeneous cycles in the data.

Inside the furnace, there are four different sections of tubes being heated. These sections are grouped in pairs according to proximity, which means that their conditions are similar and they have some influence in their paired section (Fig. 2). To estimate the state of the system, sensors inside the furnace record hourly operational data of the temperature at different points, the pressure inside the tubes, the feed rate and the state of the heaters. Although fouling also depends on the composition of the fluid being processed, we do not have information about it. In this scenario, we take a data-driven approach by approximating the state of the fouling effect inside the furnace over time with only data related to the physical properties and then modelling the gradual degradation the furnace undergoes.

At section level, the cleaning of the tubes greatly affects the measurements of their sensors. It is important to note that while one section is in a cleaning period, the others are working normally. The main problem with this strategy is that cleaning a section may affect the state of its paired section, which is reflected as severe interventions on the TS (Fig. 3). Given that the cleanings are planned in such a way that at most one section is not operational at any one time, the effects of this phenomenon are visible in nearly all cycles. These spontaneous interventions, and the usual noise and outliers typical in industrial data, have to be taken into account in the preprocessing of the data.

In this scenario, our objective is to be able to forecast the evolution of the temperature we have to provide to the tube walls during a cycle in a time window of 2000 h. The end of a cycle is not only based on a threshold for the temperature of the tube walls but can also be cut short by the operators to avoid two or more sections undergoing cleaning operations at the same time.

3. Dynamic Gaussian Bayesian networks

The fouling process is defined by complex physical relationships among the variables that represent the state of the furnace and the chemical properties of the fluid being processed. In our case, the variables are defined as univariate TS that affect one another via unknown relationships. We wish to model this system with a DGBN to approximate the interactions among the variables and take into account the temporal component.

3.1. Gaussian Bayesian networks

Bayesian networks (BNs) (Koller and Friedman, 2009) are a type of probabilistic graphical model that represents the conditional independences among triplets of random variables with directed acyclic graphs (DAGs). They can deal with discrete variables, where each node of the net is modelled as a categorical distribution, and with continuous variables, where the nodes are usually assumed to be Gaussian and the joint probability distribution is a multivariate Gaussian. Each node in the DAG has an associated conditional probability distribution (CPD) that defines the probability distribution of the node given its parents in the DAG. Specifically, in a Gaussian Bayesian network (GBN) each node's CPD is represented as a linear Gaussian model:

$$p(x_j|\mathbf{Pa}_j) = \mathcal{N}(\beta_{0j} + \beta_{1j}x_{1(j)} + \cdots + \beta_{rj}x_{r(j)}; \sigma_j^2), \quad (1)$$

where $\mathbf{Pa}_j = \{X_{1(j)}, \ldots, X_{r(j)}\}$ is the set of parents of node X_j, β_{0j} is the independent coefficient, $\beta_{1j}, \ldots, \beta_{rj}$ are the coefficients assigned to each parent and σ_j^2 is the variance of X_j. If all the nodes in the net

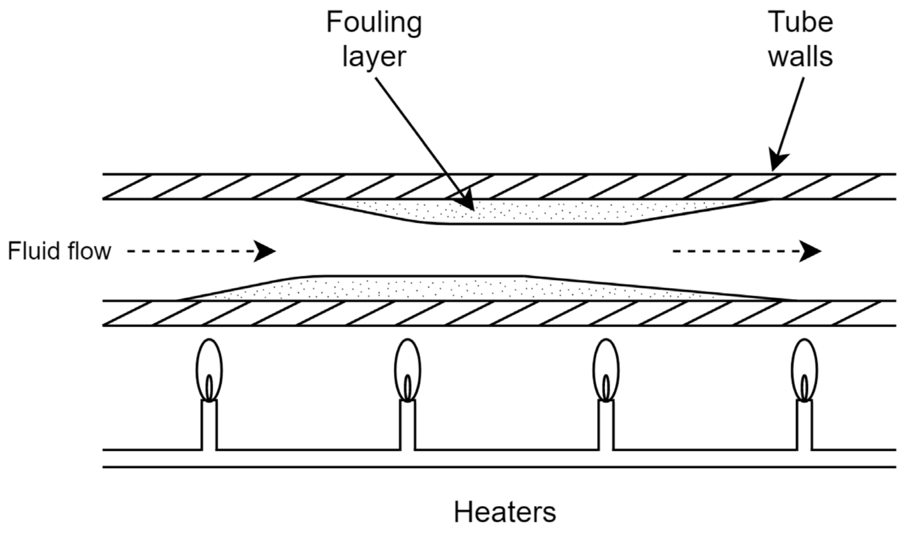


Fig. 1. Schematic representation of the fouling effect inside a tube in the furnace. The fouling layer appears over time and has to be removed when the working conditions degrade past a temperature that would melt the tube walls.

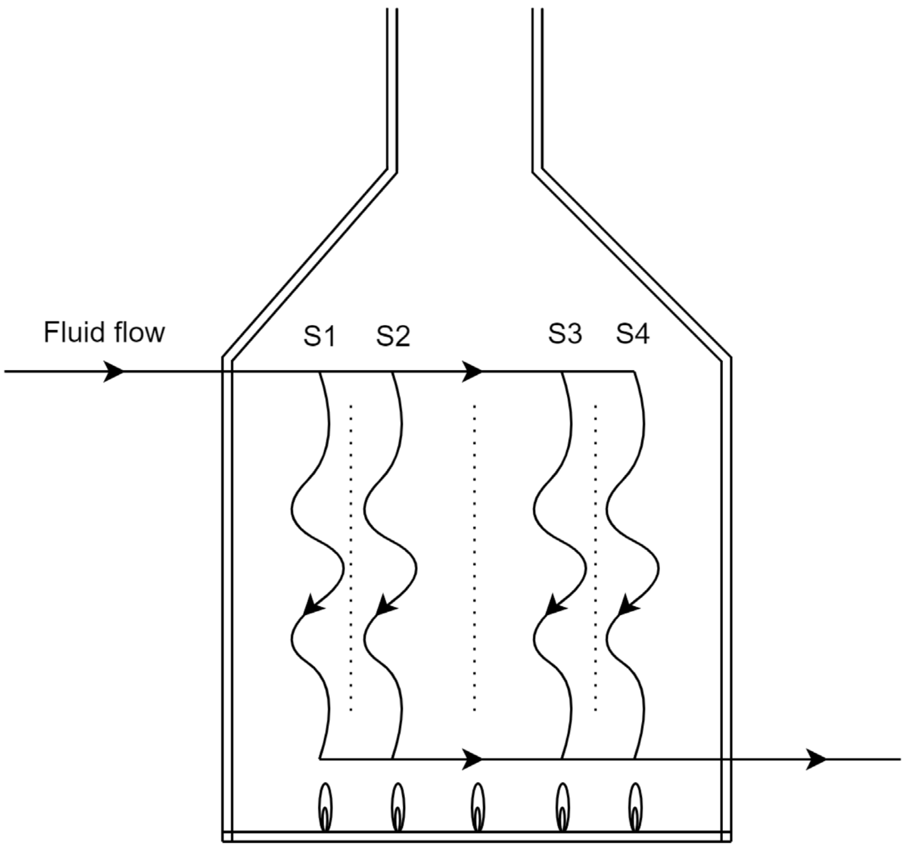


Fig. 2. Illustration of the layout of the four different sections (S1, S2, S3 and S4) inside the furnace. When one needs to be shut down to perform cleaning operations, the others remain operational.

have linear Gaussian CPDs, then their joint probability distribution is defined as

$$p(\mathbf{x}) = \prod_{i=1}^{n} p\left(x_i|\mathbf{Pa}_i\right) = \prod_{i=1}^{n} \mathcal{N}\left(\beta_{0i} + \sum_{j=1}^{r(i)} \beta_{ji} x_{j(i)}; \sigma_i^2\right) \tag{2}$$

where $\mathbf{x} = (x_1, \dots, x_n)$ is the number of nodes in the network and $r(i)$ is the number of parents of node X_i. The coefficients of each local probability distribution are estimated via linear regression models.

To apply the GBN model, we assume the variables in our problem are Gaussian. This means that the joint probability distribution of the variables in the linear Gaussian model is a multivariate Gaussian, which is not usually the case in real-world problems. However, we usually do not know the real distribution of the variables in this kind of problem, and the Gaussian assumption is a fair approximation in many cases. Moreover, the GBN model offers the advantage of simplicity in terms of more easily making inferences and learning parameters.

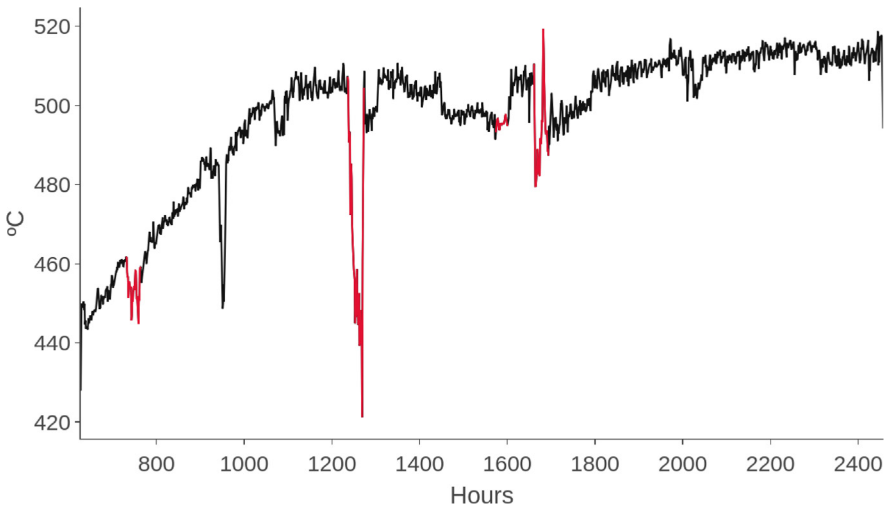


Fig. 3. The heat supplied to the tubes during a cycle in one section. The periods in which cleaning is being performed in another section are shown in red. One of these cleanings severely affects the temperature of the tube. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

3.2. Dynamic Gaussian Bayesian networks

The GBN model can be further extended to a dynamic scenario, where we consider the temporal component of a dynamic Bayesian network (DBN). In such a scenario, we usually discretize time into time slices in a given period. This period is usually determined by the frequency with which the sensors in our system collect data. In each time slice, we have a *static* GBN that, in addition to the usual intraslice parents, has parents in the previous slices and is connected to the next slice, as shown in Fig. 4.

The joint probability distribution now accounts for all the time slices until a certain horizon T:

$$p\left(\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_T\right) = p\left(\mathbf{X}_{0:T}\right) = p\left(\mathbf{X}_0\right) \prod_{t=0}^{T-1} p\left(\mathbf{X}_{t+1} | \mathbf{X}_{0:t}\right) \tag{3}$$

where $\mathbf{X}_t = \left(X_{1t}, X_{2t}, \ldots, X_{nt}\right)$ is the vector of all the nodes in a time slice t and $t = 0, 1, \ldots, T$. Eq. (3) requires all the previous time slices to be taken into account to calculate the product. To simplify this situation, we can use the Markov assumption (Koller and Friedman, 2009):

Definition 3.1. A DBN satisfies the Markov assumption if for all $t \geq 0$, the future time $t+1$ is independent of the past $0 : t-m$ given the last m time slices. This m is known as the Markovian order of the network.

The Markovian order defines the number of time slices required to confidently assume that the present is independent of the past. By applying a Markovian order one to the network, the resulting joint probability distribution will be:

$$p\left(\mathbf{X}_{0:T}\right) = p\left(\mathbf{X}_0\right) \prod_{t=0}^{T-1} p\left(\mathbf{X}_{t+1} | \mathbf{X}_t\right) \tag{4}$$

Fig. 4 shows the structure of a Markovian order one DBN with $n = 3$ and $T = 3$. When we combine this temporal component with the GBN structure, we obtain DGBNs characterized by linear Gaussian CPDs in each time slice. To compute the joint density over time we have to take into account the nodes in the current time slice and the parent nodes in the previous time slice only.

In DBN models, it is common to assume that the structure of the network is homogeneous over time, i.e., that the structure remains constant independently of t. This means that the inter- and intraslice arcs and the parameters are replicated each time the network is unrolled. If we create a Markovian order one DGBN, the future is independent from the past given the present time slice, and the whole DGBN can be represented by the first two time slices. This also means that only inter-slice arcs are permitted from $t-1$ to t. As the Markovian order increases, more arcs can appear from earlier lags to the present. If we find that the state of the system depends on more than one temporal lag, we can increase the Markovian order of the network at the expense of greater complexity in learning its structure, its parameters and for performing inference.

4. Preprocessing

The multivariate TS input data were composed of the hourly recordings of the sensors that describe the furnace state over a time span of five years. These data came from several years of sensor readings inside an industrial furnace heating a fluid prone to fouling. This TS exhibited the typical characteristics of industrial generated data, such as noise, outliers and missing values. Prior to the modelling phase, the data set had to be preprocessed to reduce its dimensionality and to address some of its irregularities, such as the effect of cleanings in other sections and noise in the signals.

4.1. Characteristics of the data

The input data set consisted of 226 variables recovered from the same number of sensors inside the furnace. These sensors record data that describe the heating process, such as the flow pressure inside the tubes, temperatures at specific points and the state of the heaters. Some of these variables show seasonality caused by the cleanings performed on the sections, and others are unaffected by them. In addition, there are many variables with redundant information that come from sensors that record the same characteristic and are placed next to each other. This results in very similar TS or even copies of the same TS shifted in time.

Regarding the rows in the data set, there are a total of 43,415 with a time difference of one hour taken from two consecutive instances. The sensors had different periods, and the shortest period was hourly. Thus, we reduced all of them to the hourly dimension in the recording process to avoid the problem of having mixed sampling frequencies in the TS (Andreou et al., 2010).

4.2. Cleaning and imputation

Many of the TS in the data set had missing values due to sensor malfunctions or system shutdowns. In the span of five years, this is a common scenario that produces information loss. Before treating

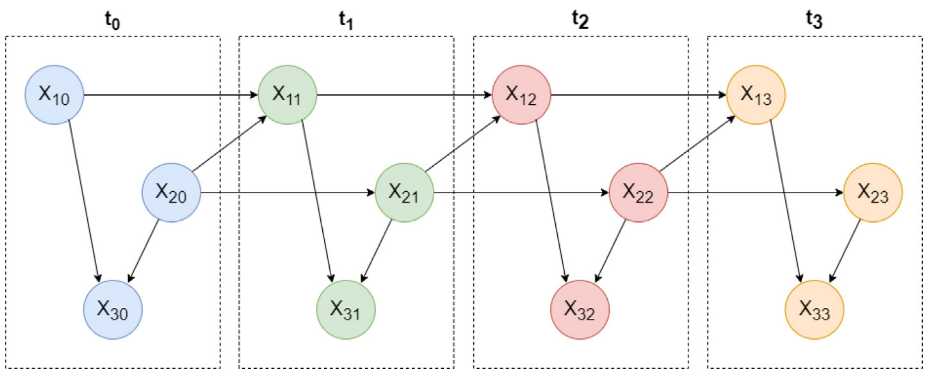


Fig. 4. Example of the structure of a Markovian order one DBN with four time slices $t_0, \ldots, t_3$ and $n = 3$. For simplicity, t_0 has the same structure as the rest of the time slices, but it need not.

the TS, these missing values had to be processed either to accept the information loss or to try to impute the missing values. In our data set, we encountered three different situations:

1. Sporadic missing values of less than ten hours. These cases were imputed using linear interpolation, because such short periods in a time span of 2000 h should not create severe fluctuations, and they should have a mild impact on the subsequent training of the model.
2. Prolonged periods of approximately 100 h of missing data. These are more severe cases with a greater impact on the information that a cycle can provide, given that cycles last 2000 h on average. In this case, linear or polynomial interpolation could introduce much undesired noise into the model, so we decided to use ARIMA interpolation (Moritz et al., 2015). This procedure fits an ARIMA model to the data immediately before the missing block and then tries to forecast the missing values.
3. Extreme cases where thousands of hours are missing. This happened in two columns of the data set. In these cases, we opted to drop these variables entirely, as we could not afford to impute such long periods and then use these synthetic data to fit a model and forecast from it.

Interpolations were performed in R software with the package *ImputeTS* (Moritz and Bartz-Beielstein, 2017).

In this case, seasonal-trend decomposition (Cleveland et al., 1990) could not be applied properly because the seasonal component was very irregular and did not show a clear period. This was aggravated by the fact that some cycles were stopped early by the operators of the furnace to avoid having more than one section on a cleaning period at the same time. When they expected that at the end of a cycle two sections would need cleaning, they stopped one of them early to keep at least three out of the four sections operational at all times.

Once the data were complete, some signal denoising and outlier smoothing were required, as some of the sensors corresponding to temperatures inside the furnace had clear trends but very noisy readings. In addition, we treated the effect of the cleanings in nearby sections as if they were outliers and smoothed them. For this purpose, we used Friedman's smoother implementation in the *forecast* R package (Hyndman et al., 2007). This method consists of locally fitting linear least-squares regressions to outlier points of the TS with a temporal window around them to obtain a smoothed outcome.

4.3. Cycle treatment

In this case, the seasonal component is not *innate* to the process, but rather forced via human intervention to revert the system to a prior state. The information to model is the behaviour of a cycle over time without being conditioned by its ending point. A cycle always ends with a cleaning, and with this approach we can forecast when this cleaning will be mandatory due to the limit temperature being reached. Instead of treating the TS as a whole, we use the points where the cleaning interventions occur and divide the series into independent and identically distributed (*iid*) cycles, as shown in Fig. 5. The objective is to predict the temperature to be provided to the tube walls as the fouling process worsens over time. This information is important to the operator of the furnace to estimate when a section will need to be cleaned.

To achieve this, the cycles have to be identified and the data set indexed by them. In total, there were 22 cycles.

For a DGBN, given that the network has some Markovian order, we can adapt the full training data set to learn the structure and parameters by dividing it into a set of *iid* cycles. Once it is divided into different cycles, we learn the parameters and the structure globally from all of them.

To learn the dynamic relationships among variables, we need to adapt our data set so that each row contains the values of all the variables in an instant and in all of the previous instants that are needed for the desired Markovian order. To perform this operation, we use the following definition:

Definition 4.1. Given a data set D with M rows, we define a *shift* function that takes as input a column of D and an order O and returns its last $M - O$ rows.

In practice, shifting a column in a data set D with $O = 1$ means that all but the first row will be returned. When we combine a column C with its shifted version for $O = 1$, we obtain the value in each row at $t = 0$ and $t = 1$, effectively preparing that column to learn the influence of the previous time slice on the present. The shift function can also be used with higher orders to see the influence of older time slices. To obtain a data set D with the columns shifted to the desired Markovian order K for a DGBN, we need to apply the *shift* function to all the columns in D for all orders O in $1, \ldots, K$. An example of adapting a data set to learn a Markovian order one DGBN is shown in Fig. 6.

4.4. Feature selection

The motivation for selecting relevant features is that it is a means to reduce the dimensionality, and it is a way to detect which sensors are not providing sufficiently relevant information and which are providing redundant information. A posterior evaluation of the model can then decide whether the irrelevant sensors are worth keeping.

With 226 variables for each section, the resulting Markovian order one DGBN would have twice the number of nodes. This situation is feasible for GBNs, but the amount of redundant information is expected to be high. Many of the sensors record data about the same specific characteristic but at relatively close points in the furnace.

In this situation, TS clustering (Montero et al., 2014) was performed to group variables based on their behaviour over time. We

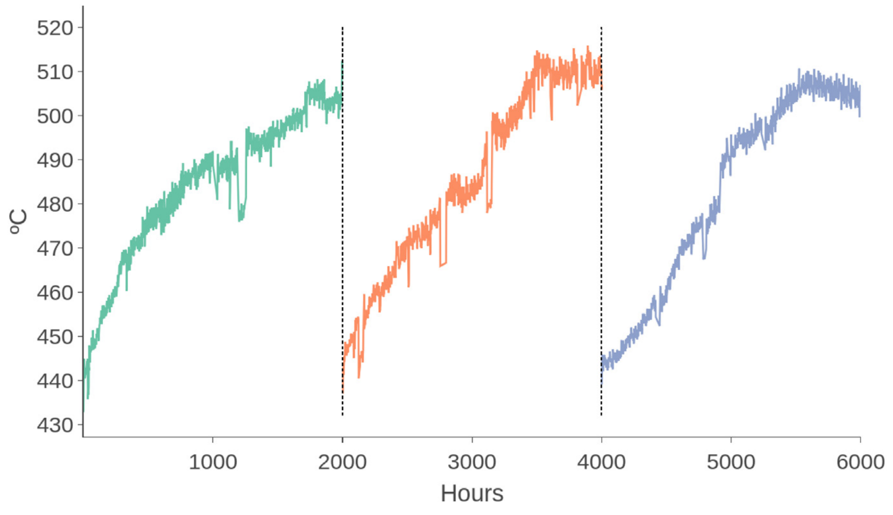
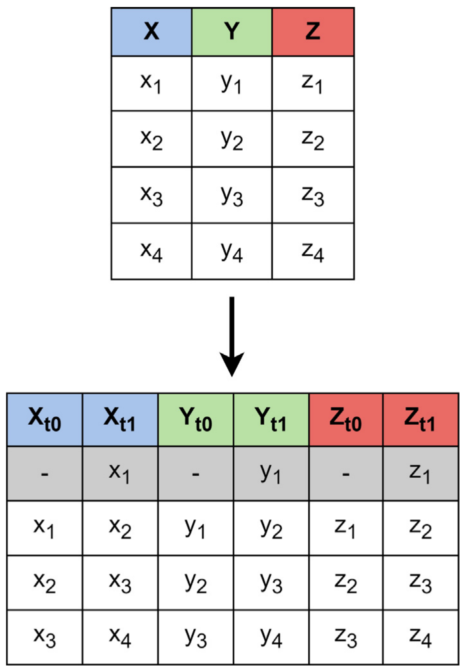


Fig. 5. A segment of the temperature TS on the tube walls. Three different cycles are shown in different colours. The black dotted lines show the limits of each cycle. If we split the multivariate TS into these independent cycles, we obtain several *iid* instances of the same underlying process.



X	Y	Z
x_1	y_1	z_1
x_2	y_2	z_2
x_3	y_3	z_3
x_4	y_4	z_4

X_{t0}	X_{t1}	Y_{t0}	Y_{t1}	Z_{t0}	Z_{t1}
-	x_1	-	y_1	-	z_1
x_1	x_2	y_1	y_2	z_1	z_2
x_2	x_3	y_2	y_3	z_2	z_3
x_3	x_4	y_3	y_4	z_3	z_4

Fig. 6. Adapting a data set D to learn a Markovian order one DGBN. All columns are shifted with $O = 1$ and added to the data set. The grey row contains missing values and should be deleted.

used hierarchical clustering with the adaptive dissimilarity index distance (Chouakria and Nagabhushan, 2007). This metric takes into account the proximity of the values in two TS and similar changes in the behaviour of both of them. To address the effect of having TS of different orders of magnitude, we performed max–min normalization. Given that our objective was to filter out redundant variables, we established a conservative cut-off point for the clusters, as shown in Fig. 7. In this way, we make sure that selecting one variable as the representative of the cluster will not lead us to discard useful information.

We obtained 35 variables for each section. As expected, the redundancy among the sensors was very high, and there were clusters of five or more sensors that yielded TS with minimal differences between them.

5. Methods and results

To apply a DGBN model, we must first define the structure learning algorithm and the inference method to follow. We will train multiple DGBNs to compare the behaviour and accuracy of their forecasts. Afterwards, we will also train a CRNN to compare its performance with the DGBN model.

5.1. Structure learning

The structure of the DGBN was learned with the adapted data set and a version of the dynamic max–min hill climbing (DMMHC) algorithm as explained in Trabelsi (2013). This is an extension of the max–min hill climbing (MMHC) algorithm (Tsamardinos et al., 2006), a hybrid method that searches the space of possible structures with a local search and then directs the arcs and scores the resulting networks with an evaluation criterion.

The MMHC algorithm first performs a local search of the structure around each node, that is, its potential parents and children. This is done by finding the Markov blanket of each node, which is the group of nodes that makes it conditionally independent from the other nodes. In the presence of continuous variables, there are several measures of this conditional independence given the data. In our case, we use the exact t test for Pearson's correlation coefficient (Edwards, 2012):

$$t(X, Y|\mathbf{Z}) = \rho_{X,Y|\mathbf{Z}} \sqrt{\frac{n - |\mathbf{Z}| - 2}{1 - \rho^2_{X,Y|\mathbf{Z}}}}, \quad (5)$$

where $\rho_{X,Y|\mathbf{Z}}$ is the partial correlation coefficient of X and Y given the set of variables $\mathbf{Z}$ and $|\mathbf{Z}|$ is the number of nodes in the set $\mathbf{Z}$. Let $\mathbf{G}$ be the set of all nodes in the graph, and let $\mathbf{Y} \equiv \mathbf{G} \setminus \{X, \mathbf{Z}\}$. Then the main idea is to find the set of nodes $\mathbf{Z}$ that makes a node X independent from all nodes $Y \in \mathbf{Y}$. An undirected arc between X and all the nodes in $\mathbf{Z}$ is then added whenever the test is not statistically significant. Once the local structure of all the nodes is found, we obtain the skeleton of the network by combining them, which constrains the step of scoring and finding the best structure. Thus, to direct the arcs in the network, only edges that are present in the skeleton of the net are allowed to be included, and the resulting structures are scored with the Bayesian information criterion (BIC) (Schwarz, 1978). The BIC score measures how well a model fits the data, and at the same time, it penalizes complexity as measured by the number of parameters in the model. Once the structure of the network is determined, the parameters are fitted via maximum likelihood estimation.

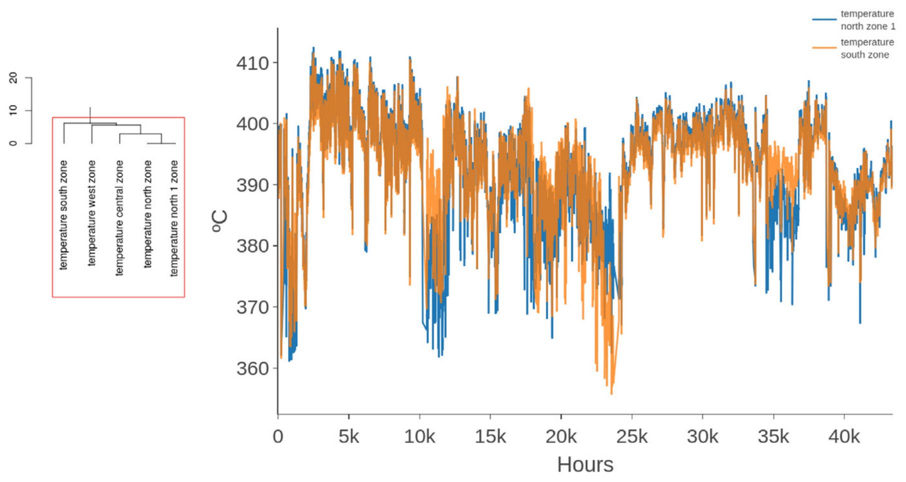


Fig. 7. One of the clusters obtained from the hierarchical clustering. A comparison between the two most distant series in the cluster is shown. Variables inside the same cluster are mostly redundant TS with some variations in scale.

To create DGBNs and perform forecasting for time series, we created an R package called *dbnR*,[1] freely available in the CRAN repository, with which we implement all the phases of structure learning and inference and the visualization methods required. This package began as an extension of the *bnlearn* package (Scutari, 2010) to dynamic scenarios. To learn the structure of a DGBN, we use bnlearn's MMHC implementation and partially adapt it by following the DMMHC algorithm. This is done in two steps: first, we learn the static structure of the network with the MMHC algorithm and then we learn the inter-time slice arcs of the net. Moreover, we added the option to modify the Markovian order of the network arbitrarily to adapt the autoregressive order empirically as needed. To our knowledge, there are no R packages that cover all these aspects of DGBN learning and inference.

5.2. DGBN models

The differences among the networks we trained were mostly structural, relative to the size of the temporal window considered and the amount of initial evidence provided. In particular, we focused on creating DGBNs with increasing Markovian orders, where we allowed arcs from a time slice to any more recent one. By this means, we can take into account longer temporal lags, where variables in earlier states of the system may be able to add relevant information to the present state.

Increasing the Markovian order results in more complex models in terms of the number of nodes and arcs, and we aim to trade off the efficiency costs with better forecasting accuracy. If we overestimate the Markovian order, the learning of the structure and parameters from the data will take too long and the predictions will degrade due to overfitting.

We also added a variable representing the time since the beginning of the cycle. This gave the model an idea of the state that the system should be in based on how long it has been running. As a result, the tendency was better captured.

It is important to note that we decided not to normalize the TS before using them to fit the DGBN model. The only visible difference when normalizing was in the values of the weights inside the linear Gaussian CPDs, given that those coefficients had to account for the differences in scales between the parent nodes and their children. On this note, we also did not standardize the TS to remove their tendencies. The reason is that we aim to model these tendencies with the Markovian order of the networks. As we provide the model with more initial evidence, we find that the future tendency of the TS is captured better in the forecasts.

The initial resulting Markovian order one network has 70 nodes and 466 arcs without limiting the number of parents of each node in the DMMHC algorithm. It is a dense network, but feasible in terms of exact inference in a DGBN. Each order increase adds 35 new nodes and all the new arcs that appear from the new time slices to more recent time slices.

5.3. Inference method

Once the structure of the DGBN and its parameters are learned from the training data, we need to make inference on it. In inferring, we need to provide the DGBN with some evidence to forecast the most likely state of the system over the coming hours. In the dynamic case, the evidence from past time slices should be used to predict the next time slices. Depending on the Markovian order of the DGBN, the amount of evidence needed for the past time slices will vary.

In the baseline case of Markovian order one, the procedure followed to forecast the multivariate TS is to provide it with some initial evidence of the nodes at time slice $t-1$ and predict the state of the nodes that conforms the system at t. In the next iteration, the forecasted values of the nodes at t will be used as evidence for $t+1$. In the general case of arbitrarily large temporal windows, evidence is provided for all time slices except the present one. Then, this evidence is used to predict the values of the remaining nodes. When we have values for all time slices, the evidence will be passed from one time slice to the previous one iteratively, and the oldest slice in the system will be forgotten. In this way, we are able to perform forecasting of the state of the furnace from an initial point to any horizon in the future. The drawback of this method is that the only real evidence from the system provided to the DGBN is that used for the initial state, and this initial point becomes more diluted the further we forecast into the future. Our idea is that by starting from a valid point with real data, we can forecast what the increase in the temperature is going to be and when it will begin to stabilize. Given that the horizon is 2000 h into the future from the initial 2 to 7 h of evidence, we do not expect to precisely forecast the behaviour of each series; rather, our goal is to predict the profile of the curves and the time at which they are going to reach their cleaning point due to the temperature required.

One advantage of this model is that it can perform inference at any point in time; it does not necessarily need to be executed at the beginning or the end of a cycle. Once trained, the system only needs an initial state to perform arbitrarily distant forecasts. Furthermore, forecasting can be performed with some kind of intervention in mind by

[1] https://CRAN.R-project.org/package=dbnR

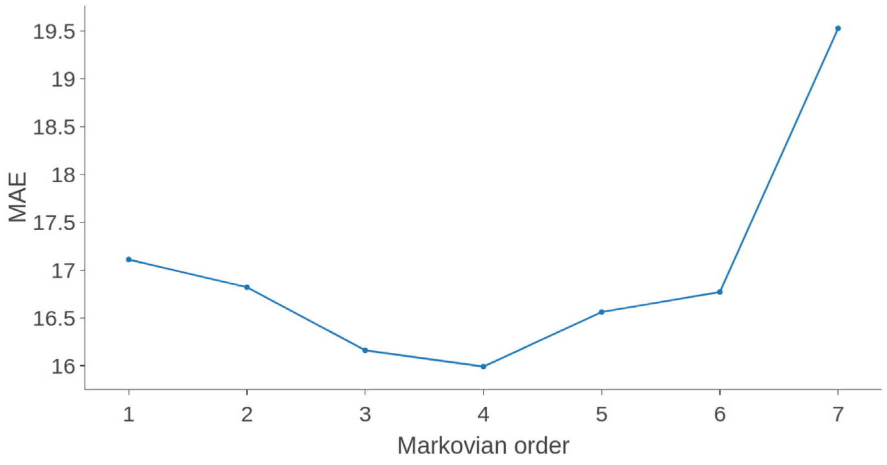


Fig. 8. As the Markovian order of the trained networks increases, the error in the forecasts decreases to a point where the networks start overfitting and accumulating error.

Table 1
DGBN forecasting results.

Order	BIC	Training time	Exec. time	MAE
1	−3.56e6	4.72 min	4.6 s	17.11
2	−4.2e6	9.35 min	7.63 s	16.82
3	−4.83e6	20.31 min	9.84 s	16.16
4	−5.46e6	43.88 min	12.8 s	15.99
5	−6.09e6	1 h 32 min	16.54 s	16.56
6	−6.71e6	3 h 7 min	20.81 s	16.77
7	−7.33e6	6 h 39 min	25.62 s	19.53

fixing some of the initial evidence to some values that are to be tested. In this way, the operators of the furnace can have an idea beforehand of the effect that changing operational variables such as the temperature or the flow will have on the state of the system in the future. This kind of forecast is also known as system simulation.

We implemented an exact inference method by calculating conditional probabilities in the equivalent multivariate Gaussian. This method consists of converting the DGBN to its equivalent multivariate joint Gaussian $\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2)$, where $\mathbf{X}_1$ are the objective nodes and $\mathbf{X}_2$ are the instantiated nodes. Once we instantiate the evidence, we marginalize the objective nodes. Thus, once the network has been translated to a mean vector $\boldsymbol{\mu} = \binom{\mu_1}{\mu_2}$ and a covariance matrix $\boldsymbol{\Sigma} = \begin{pmatrix}\boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12}\\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22}\end{pmatrix}$, we can calculate the posterior mean and variance of the objective nodes as explained in Murphy (2012):

$$\begin{aligned} p\left(\mathbf{x}_1|\mathbf{x}_2\right) &= \mathcal{N}\left(\mathbf{x}_1|\boldsymbol{\mu}_{1|2}, \boldsymbol{\Sigma}_{1|2}\right)\\ \boldsymbol{\mu}_{1|2} &= \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\mathbf{x}_2 - \boldsymbol{\mu}_2)\\ \boldsymbol{\Sigma}_{1|2} &= \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21} \end{aligned} \tag{6}$$

5.4. Convolutional recurrent neural networks

In order to compare the results of the DGBN with another typical model in the literature, we will fit a convolutional recurrent neural network. Neural networks have been widely applied in similar problems, and the are shown to be a powerful tool for predicting the fouling phenomenon. In our case, we need our model to be recurrent in order to be able to forecast TS of variable length.

We will take a similar approach to the case of the DGBN model by providing the network with a temporal window of the 24 previous hours of operation and predicting the next 24 h. Afterwards, the predictions will become the inputs of the network to predict the next window. Similarly to the DGBN case, we will predict the whole state of the system rather than only the temperature to be able to use the predicted state as input. To train the model, we will use the same preprocessed data set, but in this case we will normalize the data. Normalization is a very common procedure with neural networks, and we saw a clear improvement of the training loss after we applied it. This model is coded in Python 3.8 using *TensorFlow* and *Keras*, and the code is available in a GitHub repository.[2]

5.5. Results

Among the 22 cycles available in the data set, two of them were degraded and had to be removed. The first degraded cycle lasted only 598 h, while the average cycle length was 2073.05 h. This is because the sensors only started collecting data at the end of this cycle. The other degraded cycle had a length of 787 h and was cut short by the operators of the furnace to avoid having two sections undergo a cleaning period at the same time. After removing those cycles the 20 remaining cycles ranged from 1046 to 3635 h.

With these 20 cycles, we prepared a 5-fold cross-validation experiment, leaving 16 cycles in each execution for training and 4 for testing. The model is trained and tested for all the folds and the MAE results are then averaged. Given that the cycles have different time spans, the forecast length is the same as that of the test cycle each time, up to a maximum of 2000 h. The resulting errors of the different models are shown in Table 1. For the error metric, we used the mean absolute error (MAE):

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |x_i - \hat{x}_i|}{s} \tag{7}$$

where x_i are the real values of the TS, $\hat{x}_i$ are the predictions and s is the total number of samples. The MAE measures how much the forecasted curve differs from the real profile of the cycle. A high MAE indicates that the forecasts differ greatly from the real behaviour of the furnace, either by over- or underestimating the temperature rise. If we miss the ending point temperature by too many degrees, this could mean that we estimate that a cleaning will occur several days or even weeks away from the real date on which it has to be performed. For this reason, the MAE of the model should be below 20, as a larger error could mean estimating the ending point of a cycle some days later than it should be.

The results show that the error decreases as the Markovian order increases up to order 4, as shown in Fig. 8, but the training time of the networks increases sharply and the BIC score of the networks decreases due to the increase in the number of nodes and arcs. Up to a Markovian order of 4, the accuracy of the forecasts continues to improve due to the better performance of the DGBN in learning the different tendencies that can appear in the data. By giving the network more past time

[2] https://github.com/dkesada/kTsnn

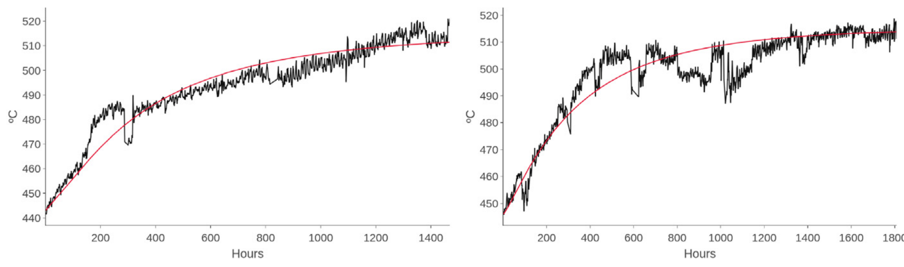


Fig. 9. Resulting predictions of a Markovian order four DGBN for two cycles with different characteristics. The black line represents the real temperature TS while the red curve is the estimation of the DGBN. Note how the predictions adapt to the observed tendency in the initial evidence provided. The predictions are extended to the full length of the cycles, with the first being an example of a cycle that ends early.

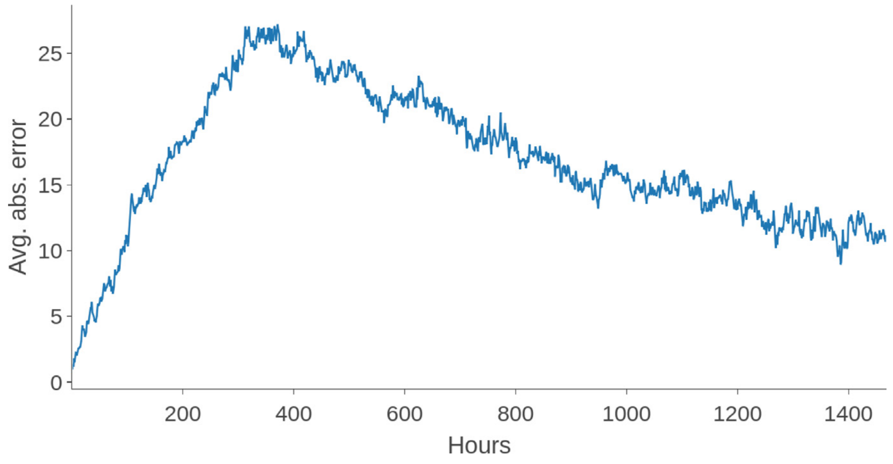


Fig. 10. Average absolute error over time of forecasting with the Markovian order four network. The absolute error is low in the first days and then increases due to the cycles individual behaviours. As the ending point of the cycles approaches, the error decreases.

slices and having the present time slice depend on them, the initial evidence given to the network helps it decide whether the tendency of the series will grow more sharply, as shown in Fig. 9. At Markovian order four and greater, increasing the Markovian order decreases the forecast accuracy due to the accumulated noise and overfitting, which increases because the network is extended backwards by many time slices.

Once we chose the Markovian order four DGBN, we assessed how the average error in the forecast varies as time increases. The majority of the cycles exhibited a low MAE in the first days of predictions and then an increase. Then, the error diminished progressively as the cycle reached its ending point. This behaviour can be seen in Fig. 10. This indicates that the model is appropriate for short-term predictions as well as for predicting the end of a cycle's life, which was our original objective. On the other hand, the predictions are less accurate around the 400-hour mark, where many of the predictions behave uniquely due to the interventions of the furnace operators. Our objective of predicting the evolution of the series in the long term is fulfilled by predicting what the initial growth of the series will be in the short term and what temperature the series will have by the end of the forecast.

The inference process maintains a good execution time overall, and the bottleneck is the learning of the structure. It is important to note that, rather than an exact prediction, we obtain a probability distribution that defines the estimate of the variable, and in our case we take the mean as the most probable value in calculating the metrics. In Fig. 9, we plot the mean of the distribution in each time slice.

If we examine the network, we can see which variables directly influence the value of the average oven temperature. In Fig. 11, we can see an example of our visualization tool with a Markovian order four DGBN. We can see that the time since the last cleaning and the pressure of the gas in the heaters, among other variables, appear as parents of the temperature, as well as the temperature in the last four temporal instants. With the time since the start of the current cycle we model the state of the fouling effect in the system, and the gas pressure directly influences the heat transferred to the tube walls. The appearance of the flow of fuel administered to the oven heaters four hours previously as a direct parent of the temperature indicates possible past interventions on a four hour-basis. With this structure, we can also simulate scenarios to see how the variation of certain temperature variables will affect the system in the future.

The results of the Markovian order four network prediction over the test cycles show it to be a powerful tool in forecasting the profile that the temperature curve will have over the coming months. By providing it with four hours of previous evidence of the behaviour of the furnace, the operators of the plant will be able to gain information on the life expectancy of the current cycle and plan the cleaning of the tubes accordingly.

In contrast, we can see the MAE results when forecasting in short, mid and long-term with the CRNN model in Table 2. The network averaged 44.7 min of training time on 300 epochs. It shows exceptionally good results on short and mid-term forecasting, but it degrades rapidly on the long-term. One interesting contrast that can be seen in Fig. 12 in comparison to forecasting with DGBNs is that the profile of the curve it predicts is less smooth, providing a prediction that looks closer to real data. This is due to the DGBN predicting the expected mean over time, not the exact value. The real value is expected to fall close to the mean in an interval defined by the variance. The results for short and mid term forecasting with the CRNN are better than the DGBN model, but the long-term forecasting degrades rapidly. In the scenario of finding the expected remaining useful life of each cycle, the DGBN model is able to give us a good estimate, while the CRNN is able to predict the near future more accurately.

The decrease in accuracy on the long-term is very likely due to lack of data to fit CRNNs that are able to predict a longer span of time.

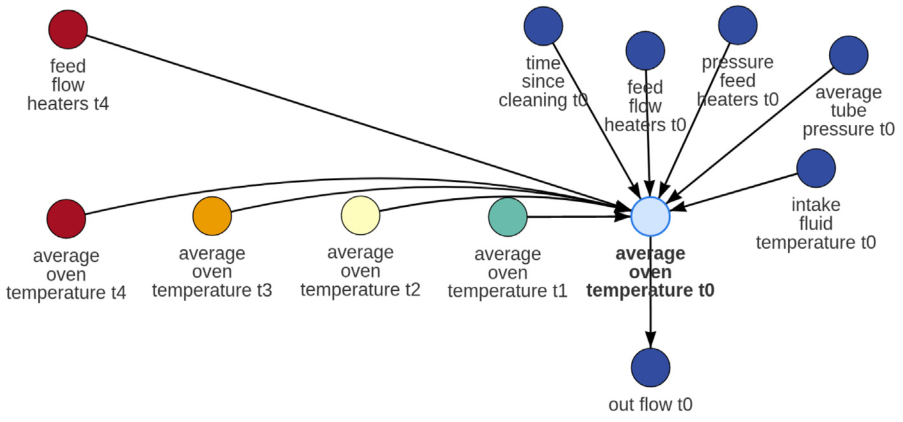

feed flow heaters t4
time since cleaning t0
feed flow heaters t0
pressure feed heaters t0
average tube pressure t0
intake fluid temperature t0
average oven temperature t4
average oven temperature t3
average oven temperature t2
average oven temperature t1
average oven temperature t0
out flow t0

Fig. 11. A screenshot of the visualization tool included in our package showing the parent and child nodes of the objective variable (cyan) in the Markovian order four DGBN. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

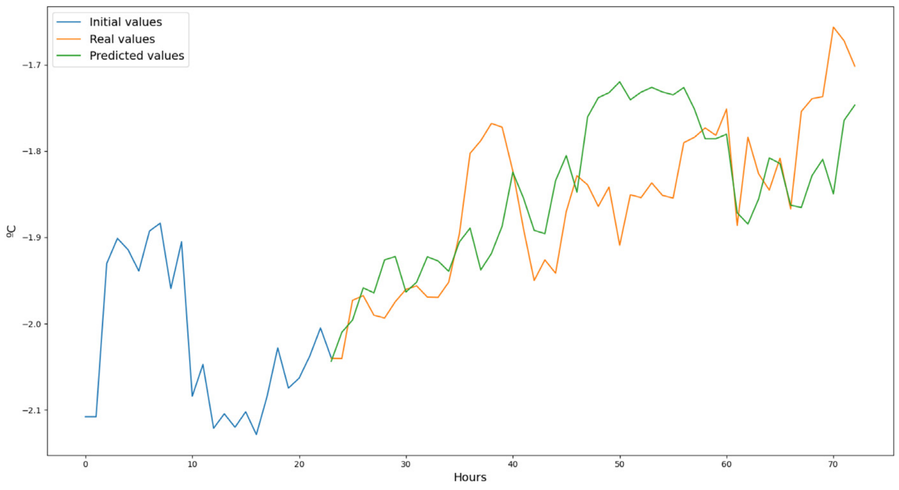


Fig. 12. Results of giving the CRNN the first 24 h of a cycle as inputs and forecasting the next 50 h. The forecasting returns an accurate profile in the short term. As opposed to the DGBN, this model is normalized with the mean and variance of the training data set.

Table 2
CRNN forecasting results.

Time span	Exec. time	MAE
50 h	0.98 s	4.63
100 h	1.6 s	7.05
250 h	3.4 s	14.01
500 h	6.57 s	47.43

This type of neural network models are very powerful for either static predictions or forecastings, but require enough previous samples to be able to converge to a stable state. In our case, generating new data takes months of continuous operation, and due to the typical noise or unforeseen problems that can arise in industrial applications, not all of this data recovered from sensors can be salvaged afterwards. From our point of view, a model for short or mid-term forecasting with CRNNs that can be fitted with reasonable sized data sets is a very useful and powerful tool for black-box forecasting of this kind of TS to make decisions that affect the near future. Moreover, instead of seeing DGBNs and CRNNs as exclusive choices of fouling models, it can be argued that they can complement each other in different scenarios. By fitting a DGBN model that is specifically tailored to perform long-term forecastings, we can combine it with a CRNN that excels at shorter forecastings and obtain the benefits of both. We can combine the interpretability of the DGBN to improve both models at the same time and the efficacy of the CRNN to perform accurate forecastings and decisions about the near future. Forecasting the temperature over a long span of time with the DGBN can give us an idea of the remaining useful life of our current cycle, and performing short-term forecasts with the CRNN as time passes can give us a clearer idea of how the profile of the curve is going to change in the near future and can help us adjust our decisions within the expected time frame that the long-term forecasting provided us.

6. Conclusions and future work

Although only the physical properties of the process of interest were available, our DGBNs were capable of making long-term predictions with an acceptable MAE while our CRNNs where able to make short and mid-term predictions with high accuracy. The strong outliers of the cleanings present in the data set did not have a severe impact on the forecasting, and the Markovian assumption for the network helped in modelling the tendency of the series. However, increasing the order of the DGBN drastically increases the learning time of the structure and decreases the BIC score, so a compromise must be found between the desired accuracy and the complexity of the model.

The reduced number of cycles resulted in the Markovian order one DGBN models learning the most common tendency. Given that the DGBN did not have enough evidence to discern between tendencies, all forecasts tended to the same curve profile, incurring in greater error. On the other hand, when we increased the Markovian order of the network, we allowed the model to identify the tendency in the previous time slices. This resulted in much better forecasts and greater robustness in cases with seasonality, where not all the cycles present the same characteristics. In comparison, the CRNN model used was able to provide very accurate predictions of the near future, but failed to predict spans of thousands of hours. We proposed a combination of both models, where we could take advantage of the strengths of both of them without having to resort to bigger datasets for fitting more complex models.

The structure of the network helps us understand the interactions among the variables inside the furnace. Inspecting the structure, we can see which variables make the objective temperature conditionally independent of the rest of the variables of the oven. This can help us decide which of the sensors are more relevant, and it allows the operator to see which variables affect the forecasting directly and the influence of each of them. In contrast to a black-box model, the structure of the network offers an explanation of how the model makes forecasts. The individual effect of a certain variable on the objective temperature can be checked in its corresponding node in the network, which helps in discovering new information on the relationships between the variables inside the furnace.

We have shown that DGBNs are useful models when treating TS in industrial environments, and we have provided the tools to apply them to specific scenarios and visualize the results. The comparison with the popular ANN models in the literature also showed some interesting interactions between both of them and the possibility of combining the two models. In the future, we would like to address the issue of automatically finding the optimal Markovian order of a DGBN to make the model easier to deploy. To accomplish this, we want to explore the relationship of the Markovian order to the autoregressive order in the ARIMA family of models and to the tendency of the TS. We would also like to apply DGBNs in cases where the chemical properties of the fluid are available in each cycle, as we would be able to extract more relevant conclusions related to the needed temperature given a specific composition. The last issue we would like to tackle is a more in-depth hybrid between DGBNs and CRNNs, where the initial forecasting is performed with the CRNN model and its outputs are fed to the DGBN model for long-term forecasting. This hybrid could potentially produce better results than their use as separate tools.

CRediT authorship contribution statement

David Quesada: Conceptualization, Investigation, Software, Writing - original draft. **Gabriel Valverde:** Conceptualization, Investigation. **Pedro Larrañaga:** Writing - review & editing. **Concha Bielza:** Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work was supported by the Spanish Ministry of Science, Innovation and Universities through the PID2019-109247GB-I00 project. This work has been partially supported by the BBVA Foundation (2019 Call) through the "Score-based nonstationary temporal Bayesian networks. Applications in climate and neuroscience" project.

References

Aminikhanghahi, S., Cook, D.J., 2017. A survey of methods for time series change point detection. Knowl. Inf. Syst. 51 (2), 339–367.
Andreou, E., Ghysels, E., Kourtellos, A., 2010. Regression models with mixed sampling frequencies. J. Econometrics 158 (2), 246–261.
Brockwell, P.J., Davis, R.A., Calder, M.V., 2002. Introduction to Time Series and Forecasting. Springer.
Cheng, C., Sa-Ngasoongsong, A., Beyca, O., Le, T., Yang, H., Kong, Z., Bukkapatnam, S.T., 2015. Time series forecasting for nonlinear and non-stationary processes: A review and comparative study. IIE Trans. 47 (10), 1053–1071.
Chouakria, A.D., Nagabhushan, P.N., 2007. Adaptive dissimilarity index for measuring time series proximity. Adv. Data Anal. Classif. 1 (1), 5–21.
Cleveland, R.B., Cleveland, W.S., McRae, J.E., Terpenning, I., 1990. STL: A seasonal-trend decomposition. J. Off. Stat. 6 (1), 3–73.
Davoudi, E., Vaferi, B., 2018. Applying artificial neural networks for systematic estimation of degree of fouling in heat exchangers. Chem. Eng. Res. Des. 130, 138–153.
Diaby, A.L., Miklavcic, S.J., Addai-Mensah, J., 2016. Optimization of scheduled cleaning of fouled heat exchanger network under ageing using genetic algorithm. Chem. Eng. Res. Des. 113, 223–240.
Diaz-Bejarano, E., Coletti, F., Macchietto, S., 2019. Modeling and prediction of shell-side fouling in shell-and-tube heat exchangers. Heat Transf. Eng. 40 (11), 845–861.
Edwards, D., 2012. Introduction to Graphical Modelling. Springer Science & Business Media.
Hyndman, R.J., Khandakar, Y., et al., 2007. Automatic Time Series for Forecasting: The Forecast Package for R, No. 6/07. Monash University, Department of Econometrics and Business Statistics.
Koller, D., Friedman, N., 2009. Probabilistic Graphical Models: Principles and Techniques. The MIT Press.
Lalot, S., Palsson, O., Jonsson, G., Desmet, B., 2007. Comparison of neural networks and Kalman filters performances for fouling detection in a heat exchanger. Int. J. Heat Exch. 8 (1), 151.
Montero, P., Vilar, J.A., et al., 2014. TSclust: An R package for time series clustering. J. Stat. Softw. 62 (1), 1–43.
Moritz, S., Bartz-Beielstein, T., 2017. imputeTS: Time series missing value imputation in R. R J. 9 (1), 207–218.
Moritz, S., Sarda, A., Bartz-Beielstein, T., Zaefferer, M., Stork, J., 2015. Comparison of different methods for univariate time series imputation in R. arXiv preprint arXiv:1510.03924.
Murphy, K., 2002. Dynamic Bayesian Networks: Representation, Inference and Learning (Ph.D. thesis). UC Berkeley, Computer Science Division.
Murphy, K.P., 2012. Machine Learning: A Probabilistic Perspective. The MIT Press.
Peña, D., Tiao, G.C., Tsay, R.S., 2011. A Course in Time Series Analysis. John Wiley & Sons.
Pogiatzis, T., Ishiyama, E.M., Paterson, W.R., Vassiliadis, V.S., Wilson, D.I., 2012. Identifying optimal cleaning cycles for heat exchangers subject to fouling and ageing. Appl. Energy 89 (1), 60–66.
Radhakrishnan, V., Ramasamy, M., Zabiri, H., Do Thanh, V., Tahir, N., Mukhtar, H., Hamdi, M., Ramli, N., 2007. Heat exchanger fouling model and preventive maintenance scheduling tool. Appl. Therm. Eng. 27 (17–18), 2791–2802.
Santamaria, F.L., Macchietto, S., 2019. Integration of optimal cleaning scheduling and control of heat exchanger networks under fouling: MPCC solution. Comput. Chem. Eng. 126, 128–146.
Schwarz, G., 1978. Estimating the dimension of a model. Ann. Statist. 6 (2), 461–464.
Scutari, M., 2010. Learning Bayesian networks with the bnlearn R Package. J. Stat. Softw. 35 (3), 1–22.
Sundar, S., Rajagopal, M.C., Zhao, H., Kuntumalla, G., Meng, Y., Chang, H.C., Shao, C., Ferreira, P., Miljkovic, N., Sinha, S., Salapaka, S., 2020. Fouling modeling and prediction approach for heat exchangers using deep learning. Int. J. Heat Mass Transfer 159, 120112.
Trabelsi, G., 2013. New Structure Learning Algorithms and Evaluation Methods for Large Dynamic Bayesian Networks (Ph.D. thesis). Université de Nantes.
Tsamardinos, I., Brown, L.E., Aliferis, C.F., 2006. The max-min hill-climbing Bayesian network structure learning algorithm. Mach. Learn. 65 (1), 31–78.
Wang, Y., Yuan, Z., Liang, Y., Xie, Y., Chen, X., Li, X., 2015. A review of experimental measurement and prediction models of crude oil fouling rate in crude refinery preheat trains. Asia-Pac. J. Chem. Eng. 10 (4), 607–625.