
# A generic hierarchical clustering approach for detecting bottlenecks in manufacturing

Journal of Manufacturing Systems 55 (2020) 143–158

Contents lists available at ScienceDirect

Journal of Manufacturing Systems

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

# A generic hierarchical clustering approach for detecting bottlenecks in manufacturing

Mukund Subramaniyan[a,*], Anders Skoogh[a], Azam Sheikh Muhammad[b], Jon Bokrantz[a], Björn Johansson[a], Christoph Roser[c]

[a] Department of Industrial and Materials Science, Chalmers University of Technology, Gothenburg 41296, Sweden
[b] Department of Computer Science and Engineering, Chalmers University of Technology, Gothenburg 41296, Sweden
[c] Department of Management Science and Engineering, Karlsruhe University of Applied Sciences, Karlsruhe 76133, Germany

ARTICLE INFO

*Keywords:*
Throughput bottlenecks
Maintenance
Manufacturing system
Unsupervised machine learning
Data-driven

ABSTRACT

The advancements in machine learning (ML) techniques open new opportunities for analysing production system dynamics and augmenting the domain expert's decision-making. A common problem for domain experts on the shop floor is detecting throughput bottlenecks, as they constrain the system throughput. Detecting throughput bottlenecks is necessary to prioritise maintenance and improvement actions and obtain greater system throughput. The existing literature provides many ways to detect bottlenecks from machine data, using statistical-based approaches. These statistical-based approaches can be best applied in environments where the statistical descriptors of machine data (such as distribution of machine data, correlations and stationarity) are known beforehand. Computing statistical descriptors involves statistical assumptions. When the machine data doesn't comply with these assumptions, there is a risk of the results being disconnected from actual production system dynamics. An alternative approach to detecting throughput bottlenecks is to use ML- based techniques. These techniques, particularly unsupervised ML techniques, require no prior statistical information on machine data. This paper proposes a generic, unsupervised ML-based hierarchical clustering approach to detect throughput bottlenecks. The proposed approach is the outcome of systematic and careful selection of ML techniques. It begins by generating a time series of the chosen bottleneck detection metric and then clustering the time series using a dynamic time-wrapping measure and a complete-linkage agglomerative hierarchical clustering technique. The results are clusters of machines with similar production dynamic profiles, revealed from the historical data and enabling the detection of bottlenecks. The proposed approach is demonstrated in two real-world production systems. The approach integrates the concept of humans in-loop by using the domain expert's knowledge.

## 1. Introduction

Over the last decade, machine learning (ML) has made unprecedented progress in areas as diverse as finance, energy, e-commerce, geology, space and biology. ML is widely used in applications ranging from automating mundane tasks to offering intelligent insights and supporting human decision-making [1,2]. In finance, for example, which has mature ML applications, unsupervised ML techniques are used to analyse the stock prices of companies and identify companies with unique stock price behaviour when compared to others [3–5]. The insights obtained help financial analysts gain an overview of similarly and dissimilarly behaving stocks and thus enable financial investment decisions. The success of ML is due to the large volumes of available data and computational resources allowing advanced ML techniques to be run. Manufacturing companies are also warming to the idea of using cutting-edge advances in the ML field to enhance their production system operations [6–8]. With production systems becoming increasingly complex, managing them has become less amenable to manual administration. Advancements in the ML field can thus be exploited to better support the decision-making process of domain experts [9–12].

On a real-world shop floor, domain experts must make important decisions that aim to increase the overall system throughput [13,14]. One way to achieve this is to focus on reducing throughput-related losses in the production system. Real-world case studies indicate that 20–30 % of throughput losses are common in production systems [15]. These losses are due to disruption events such as random machine

* Corresponding author.
*E-mail address:* mukunds@chalmers.se (M. Subramaniyan).

https://doi.org/10.1016/j.jmsy.2020.02.011
Received 15 January 2020; Received in revised form 20 February 2020; Accepted 28 February 2020

downtime and variations in machine cycle time. Depending on the nature of the production system, disruption events in some machines will affect overall system throughput more than others, and these machines are called "throughput bottlenecks" [16]. To achieve steady or increased throughput, the domain experts must devote their efforts to increasing throughput in these bottleneck machines [17,13]. Accomplishing this means that domain experts must identify the throughput bottlenecks in their production systems.

From an academic point of view, to support the identification of bottlenecks, researchers have proposed several methods involving data collected from the production system [16,18–25]. These methods provide important contributions towards throughput bottleneck detection, and usually, the key ideas are built on statistical approaches to detecting bottlenecks (such as using descriptive and inferential statistical techniques). Employing statistical approaches is normally justified, as it accounts for the variability of data arising from randomness in the system [14,16,25]. Usually, a statistical approach will rely on various underlying assumptions about the machines' data to compute statistical descriptors such as distribution, statistical independence and stationarity. Thus, a fundamental challenge when using existing scientifically rigorous statistical approaches in the real-world is to ensure that the central assumptions of those approaches are satisfied.

In using any statistical approach, a significant amount of time is required to ensure that the underlying real-world data suits the set of assumptions. This task in itself requires domain experts to be truly knowledgeable about various statistical concepts and methods if they are to validate the data against those concepts and draw meaningful conclusions. A further challenge is that, as the production system's dynamics change over time, time-varying machine data (which behaves according to a certain statistical descriptor) may also change its patterns. For example, a normal distribution assumption might not hold good for time-varying data which tends to change distribution in future scenarios. This would require suitable statistical processing of the data each time before the statistical approaches are used to detect bottlenecks [25]. In such scenarios, it therefore becomes challenging for statistical approaches to maintain a high degree of accuracy and reliability. This is where carefully designed ML-based approaches can: a) help detect bottlenecks without relying on strict statistical assumptions, b) scale well with time-varying data, and c) produce quick, reliable results whilst keeping the domain experts in the loop. The problem of throughput bottleneck detection in production is analogous to that of analysing the stock prices of finance companies, which was successfully solved by ML techniques. Instead of finding companies that show a unique stock price over time compared to other stocks, the target is to identify machines that exhibit unique behaviour as compared to the other machines. The unique behaviour of those machines is a key indicator that they are throughput bottlenecks. Inspired by this analogous common phenomenon in the finance and manufacturing domains, we explored how ML techniques (widely used in finance for stock price analysis) can be used to detect throughput bottlenecks.

Accordingly, the purpose of the study is to improve throughput by facilitating the detection of throughput bottlenecks. The aim of the paper is to propose a generic, unsupervised, ML-based, hierarchical clustering approach to identify throughput bottlenecks in the production system and test the proposed approach on real-world production systems. The approach should be flexible enough to couple with any bottleneck detection method. In sum, this paper provides an exciting opportunity to advance the knowledge of state-of-the-art throughput bottleneck detection using unsupervised ML techniques.

The paper is structured as follows. Section 2 presents the theoretical background of different bottleneck detection methods and the associated statistical assumptions and unsupervised ML techniques. Section 3 then starts by presenting a modular breakdown of the proposed approach, covering the high-level descriptions of all the steps in each module. Data from a real-world production system is used to explain the practical aspects of applying each step within every module of the proposed approach. This integrated description of the generic steps (and their application to data from a real-world production system) aims to demonstrate that independent decisions are made within each module in the hierarchy and that an outcome is generated. This in itself is valuable and amenable to analysis by domain experts. The proposed approach is further evaluated by being applied to an additional real-world production system, with the results shown in Section 4. Section 5 presents a detailed discussion of the proposed approach, in terms of its contributions to academic and industrial practice (and its limitations) plus an outlook on future work. The paper concludes with Section 6 summarising the results of the paper.

## 2. Theoretical background

This section presents a theoretical background to throughput bottleneck detection. It also examines the statistical assumptions made in the existing literature and explains its implications in practice. An outline of the theoretical explanation, of different unsupervised ML techniques used in the proposed approach is then given. Finally, the active period bottleneck detection method (which is used as an example in the real-world industrial test studies for the proposed approach) is explained.

### 2.1. Existing assumptions in throughput bottleneck detection

The problem of throughput bottleneck detection in production lines is covered extensively in the literature. Various methods of identifying historical throughput bottlenecks in the production system have been proposed (active period method by Roser et al. [16]; turning point method by Li et al. [19]; inter-departure time variance by Betterton and Silver [20]; and overall equipment efficiency method by Tang [21]). In all these different methods, the overall approach is to detect bottlenecks using traditional statistical approaches, including descriptive (such as average, standard deviation, coefficient of variation) [19,20] and [21] and inferential statistical techniques (such as hypothesis testing) [16,22]. Yu and Matta [25] proposed an improved statistical approach that can be coupled with any of the bottleneck detection methods.

Inferential statistical techniques are used to account for variability in machine data when detecting bottlenecks. However, this application is no trivial task [26]. Various manually supervised steps are involved, such as: a) identifying statistical descriptors to define the structure of the machine data collected over a specified period, b) decision to use relevant test statistics and c) selection of significance level. Of these different steps, the most important one is identifying statistical descriptors for the collected machine data structure. This is because the structure guides the choice of relevant test statistics when applying inferential statistics. An incorrect choice of test statistics can lead to inaccurate bottleneck detection results. Different assumptions are made when identifying different statistical descriptors of machine data. The following is a summary of the four common statistical descriptors, as inferred from the existing literature. These are: (1) statistical distribution, (2) autocorrelation, (3) cross-correlation and (4) stationarity.

(1) Statistical distribution: this provides a parameterised mathematical function, describing the relationship between the data points of a machine dataset in sample space. The distribution is usually found using various statistical techniques, or assumed for convenience. It will guide the choice of parametric and non-parametric statistical tests when using inferential statistical techniques. The most common assumption on statistical distribution is that of normality. For example, Subramaniyan et al. [22] (p. 233) assumes the data to be taken from a normally distributed population, an assumption that is the same across all the machines in a production system. This choice of assumption facilitates the use of a *t*-test as a statistical hypothesis test to detect bottlenecks. However Subramaniyan et al. [22], (p. 233) did not check the validity of the different

assumptions when detecting bottlenecks. Roser et al. [16] (p. 951) did not report the distribution of machine data used to detect bottlenecks in the test production system. On the other hand, Yu and Matta [25] (p. 6321) recognises that the normality assumption is difficult in production systems and therefore proposed a non-parametric test to detect bottlenecks (the Mann-Whitney U test).

(2) Autocorrelation: this is the degree of similarity between data points in a machine dataset (collected chronologically) and the lagged version of the same data points over successive time intervals. Roser et al. [16] (p. 951) assumes that the data points in a given machine dataset are independent of each other. This assumption is later proved in Roser and Nakano [27] (p. 71), with limited empirical testing based on a discrete-event simulation environment of a production system. Subramaniyan et al. [22] (p. 233) assumes that the data points of a given machine dataset are independent of each other and they are not tested for their validity when detecting the bottlenecks. On the other hand, Subramaniyan et al. [28] (p. 536), in another study, assumes that the data points of a machine dataset are autocorrelated; this assumption is made in order to facilitate the adaptation of time series forecasting techniques, such as Auto-regressive Moving Average (ARIMA), to predict future bottlenecks. Similarly Li et al. [29], (p. 3) makes the assumption that the data points in a machine's data are autocorrelated, in order to use time series techniques to predict future bottlenecks. However, the validity of this assumption is, again, not tested. Yu and Matta [25] (p. 6322) recognises that data points in a given machine's data are usually not independent and proposes a batching means technique to generate independent data points.

(3) Cross-correlation: for every pair of machines in the production system, cross-correlation is a measure of similarity, with data from one machine compared to that from another. Usually, in a production system, disturbances in one machine will affect the downstream and upstream machines. In other words, the datasets of two different machines need to be tested for correlation; this guides the selection of appropriate hypothesis tests. There was no mention of correlating different machine data in the research reported by Roser et al. [16]. Subramaniyan et al. [22] (p. 234) assumed the machines to be independent and used an independent *t*-test to detect bottlenecks. Yu and Matta [25] (p. 6321) used a Mann-Whitney U test for the hypothesis tests, which indicates that machines are assumed to be independent.

(4) Stationarity: this means that the statistical properties (average, variance and so on) of the machine data are constant over time. The existing studies assume that machine data collected chronologically is stationary. This means that the mean and variance do not change over time. Subramaniyan et al. [28] (p. 541) show, by means of a real-world example, that machine data is non-stationary. In other words, the mean and variance change over time. There was no mention of stationarity in Roser et al. [16], Li et al. [29], and Yu and Matta [25]. Thus, using descriptive statistics to identify the bottlenecks for non-stationary data risks inducing errors when detecting throughput bottlenecks.

To summarise, from the existing studies it is unclear whether the researchers have properly considered the reasons for not validating the different assumptions, or tested whether the different statistical approaches are robust against violations of the assumptions. In real-world production systems, the machine data is often too noisy. Each machine's behaviour can differ, and it is difficult to generalise assumptions for all machines. Moreover, Li and Ni [13] indicates, through real-world studies, that the assumptions (specifically those on statistical distribution) may not reliably capture the machine data's dynamics. If different assumptions are used for convenience, there is a risk that the results will be disconnected from the real-world process [26].

### 2.2. Unsupervised ML techniques

ML-based approaches are a natural alternative when the data does not conform to the assumptions of some existing statistical approaches, or when the data comes from a time-varying dynamic system, possibly entailing situations which make the data deviate from its normal behaviour. This section outlines the unsupervised ML techniques, especially hierarchical clustering for time series from the academic ML literature. The different tools and techniques for cluster generation and analysis from ML literature are then presented. The aim of this section is to provide readers with the necessary information on unsupervised ML techniques to allow interpretation and replication of the approach proposed in this paper.

#### 2.2.1. Hierarchical clustering of time series

The goal of hierarchical clustering is to capture the underlying structures of the time series data, and it produces a set of nested clusters, organised as a hierarchical tree [30,31]. One main advantage of hierarchical clustering is that it can capture more complex and intricate cluster structures. There are different types of hierarchical clustering, such as agglomerative (also called "bottom-up") and divisive (also called "top-down"). A detailed explanation of these types can be found in Hastie et al. [30]. Of all the different types, agglomerative hierarchical clustering is the most commonly used in the data science domain [32]. The advantages of agglomerative clustering are that it produces complete hierarchical structures inductively and is relatively easy to implement [33]. One way of presenting the results of hierarchical clustering neatly is to use a two-dimensional diagram, known as a dendrogram [34,30].

The main idea behind agglomerative is to treat each time series as an individual cluster and then at each step we recursively merge the closest pair of clusters until one cluster is left. Agglomerative clustering is conducted by defining the inputs regarding the distances between the time series and distances between the clusters. The algorithmic details of the agglomerative hierarchical clustering are found in Hastie et al. [30]. Before conducting different time series clustering operations, a distance measure needs to be defined between all pairs of time series. For this purpose, a number of measures are defined in the literature. For example, Euclidean distance, dynamic time warping (DTW), Mahalanobis distance, Fourier coefficients, auto-regressive models, edit distance, minimum jump models [35]. Out of several distance measures, it has been indicated in the literature that DTW outperforms for comparisons of time series because of its non-linear mapping capabilities [36]. After establishing the distance measure between the time series, there is a need to establish a measure of the distance between the clusters to facilitate the clustering process. Five common distance measures of the distance between clusters are defined in the literature. Those are single linkage, complete linkage, average linkage, average group linkage, and wards method [30]. Clustering results also depend on the type of linkage that is chosen, and different types will give different results. However, it is indicated by Hirano and Tsumoto [37] that complete linkage produces better results compared to other linkages for producing more compact and balanced clusters.

#### 2.2.2. Tools to facilitate generation and analysis of clusters

To facilitate cluster generation, the required number of clusters must first be determined. Various tools are defined in the literature, which can be used to help fix a number. These include the elbow method, gap statistic method, mode and maximum difference [38]. Determining the appropriate tool depends on the application domain. Of the different tools, the elbow method (also called a "scree plot") is a relatively straightforward method and examines the acceleration in the jumps of variance. The elbow method explains the percentage of variance for different numbers of clusters. If this value is plotted in a graph against the number of clusters, the number of clusters corresponding to the point at which the marginal gain in variance drops is the optimal

number [39]. Once the clusters are formed, they need to be analysed. To do this, a representative time series can be generated, representing each cluster formed. The main idea behind this is to facilitate deeper analysis of the clusters using regression, trend detection tests, and so on. A representative time series is constructed by computing the averages of each data point between two time series sequences [40]. The algorithmic details for generating a representative time series are explained in Baheti and Toshniwal [40], and an application example is shown in Baheti and Toshniwal [40,41]. A sample application for analysing the road accident time series is given in Kumar and Toshniwal [42]. This uses the averaging algorithm to generate a representative time series.

### 2.3. Active period bottleneck detection method

The active period method of bottleneck detection was first proposed by Roser et al. [16] and demonstrated using a discrete event-simulation-based environment. The basic idea behind the method is to classify different machine states (a state represents an activity that is carried out by the machine) during a production run into two binary states. The first is called the “active” state; the other, the “non-active” state. The active state is when an activity is conducted on or by the machine that is causing blockage or starvation in other machines. Examples include producing state (when the machine is producing a product), downstate (when the machine has broken down and is being repaired) and setup state (when setup activities are being conducted on the machine). When the active state durations are combined and compared with all other machines in the production system, the bottleneck machines can be identified, as they are the ones with the highest active duration. [22] proposed an event-log-based, data-driven statistical approach based on the active period method, to detect bottlenecks in the production system.

## 3. Proposed generic hierarchical clustering approach to throughput bottleneck detection

In this section, we propose an approach to detecting throughput bottlenecks, using machine event log data extracted from real-world production systems and unsupervised ML techniques. The structure of this approach takes its inspiration from the Cross Industry Process for Data Mining (CRISP-DM) [43,44] and consists of seven modules: (1) data collection, (2) selecting a suitable bottleneck detection method, (3) data pre-processing, (4) applying a hierarchical clustering technique, (5) cluster computation and generation, (6) representative time series generation, and (7) throughput bottleneck detection. These are shown in Fig. 1. In every module, we define the generic steps and the methods specific to that module. We conclude each module by demonstrating the proposed steps on the event log data extracted from the real-world production system. The main aims of discussing the generic steps and their demonstration on a real-world production system are: a) to explain how event log data from machines can be processed so that unsupervised ML techniques may be applied, and b) to show and explain relevant inputs from production system domain experts in every module.

Demonstration of the proposed approach in a real-world production system was carried out in R software (Version 3.4.3) by uploading the machine event log datasets. R is widely used software in data science and ML applications. Moreover, it has libraries relevant to the ML techniques used in the demonstration. The libraries used in this demonstration are dplyr, dtwclust, tseries, TSclust, ggplot2, zoo and cluster.

### 3.1. Module 1: data collection

This module covers the collection of event log data from a real-world production system. A real-world production system consists of many resources (such as machines and humans) working together to produce products. Every resource carries out actions or has actions conducted on them during a production run. These actions may be recorded digitally with corresponding timestamp information. The data records containing this information on actions and time stamps are called “event logs”. Once the event log data has been identified, the next step is to define the time interval of interest for throughput bottleneck detection. This is important because it sets the parameters for the required historical event log data from the production system being studied. The minimum data required for successful application of the proposed bottleneck detection approach should cover enough past production runs of the system so that all events relevant to the analysis are well-represented. The setting up of limits is best done by obtaining the input of domain experts in the production system. They can determine what duration of historical data would represent the production system dynamics well. The output of this module is the event log data in tabular format, containing the events described and their timestamps.

In the demonstration conducted in this study, a real-world automotive component manufacturing production line is used to demonstrate the unsupervised ML approach and detect throughput bottlenecks. The production line consists entirely of Computer Numerical Controlled (CNC) machines, which conduct various machining operations on the component. The layout of the production line is shown in Fig. 2. The component enters machine M1 in the production line and emerges from the line once operations at M13 are complete. Each CNC line machine is connected to a Manufacturing Execution System (MES) which records machine events and their corresponding timestamps.

For this production line, it was decided to take data from 30 historical production runs, to identify throughput bottlenecks (a single production run constitutes 17 h in a day, running from 06:00:00 to 23:00:00). Once the time interval has been defined, the raw event log data for all machines is extracted from the MES for the specified time interval. A sample of event log data from machine M1 is shown in Table 1. It shows the events represent unevenly spaced time series data, in which the spacing of events is not constant. These events, the machine ID, and the corresponding timestamp form the raw logs used as a starting point for the proposed approach.

### 3.2. Module 2: selection of suitable bottleneck detection method

Once the event log data is extracted, the next step is to select a suitable throughput bottleneck detection method for a given production system. This selection is important when executing the other modules, as their steps correlate to the selected method. As explained in Section 2.1, the literature covers various rich methods which may be selected to detect bottlenecks. However, two things are important in selecting a suitable method. Firstly, a decision needs to be made on the type of bottleneck detection method that suits the production system. This is where domain experts will play a key role, because of their deep understanding of the underlying system. Thus, a decision about the target bottleneck detection method is best made by them, based on their domain knowledge and on practical requirements. Secondly, exploring the event log data of the production system to make sure that the necessary information required to apply the chosen bottleneck detection method can be extracted. This requirement is handled by introducing a feedback loop between Modules 1 and 2, to verify the requirements of the selected bottleneck detection method from the collected data. This module’s output is selection of suitable bottleneck detection method, to facilitate the execution of other modules.

The active period bottleneck detection method [16] has been selected for the demonstration. The metric used by this method to detect bottlenecks is the active duration. The method was chosen over the others because it detects bottlenecks causing blockages and starvation in other machines by using all the action events (conducted either *by* the machine, or *on it* by humans). Thus, combining different events to

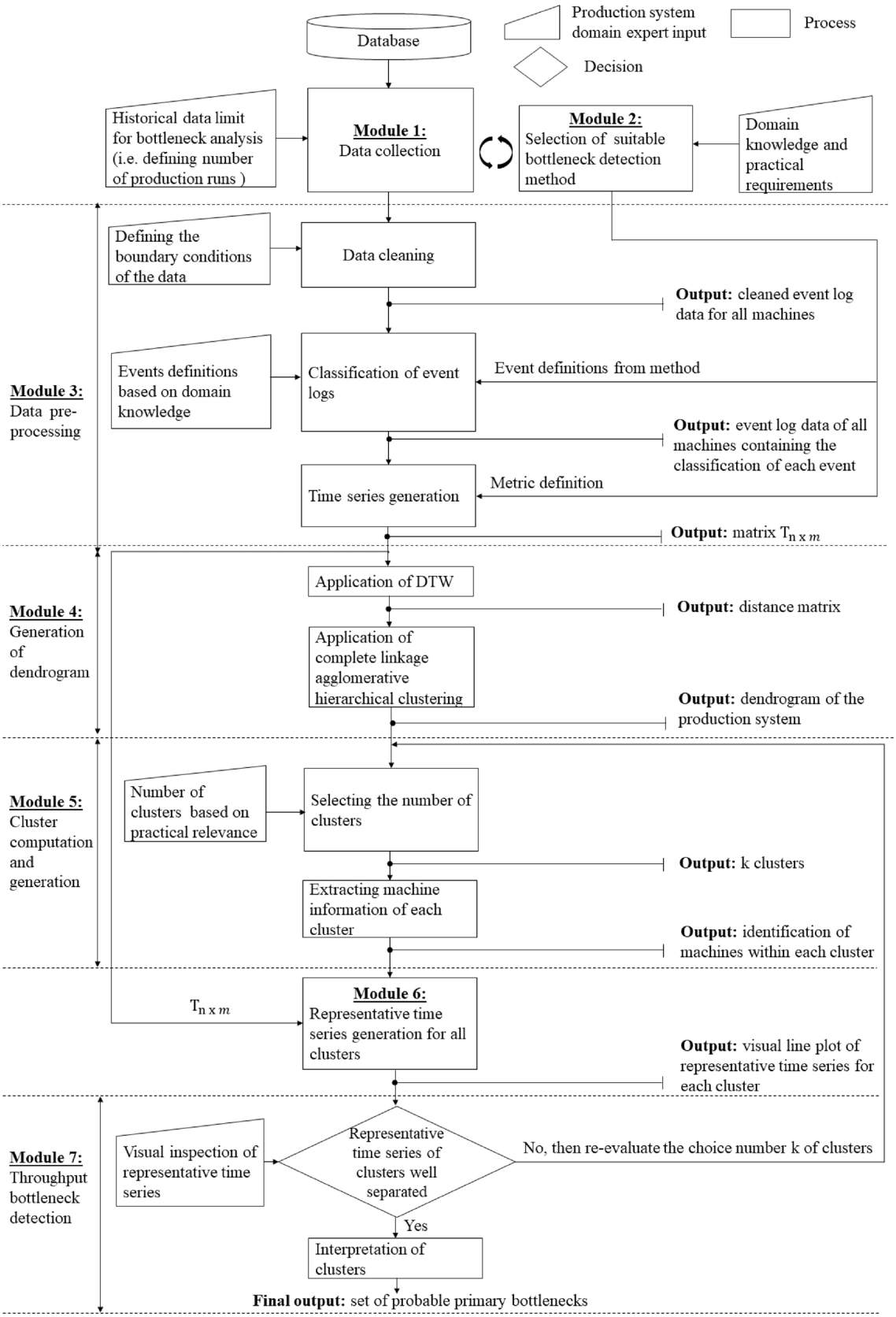


**Fig. 1.** Modular breakup of the proposed approach for throughput bottleneck detection.

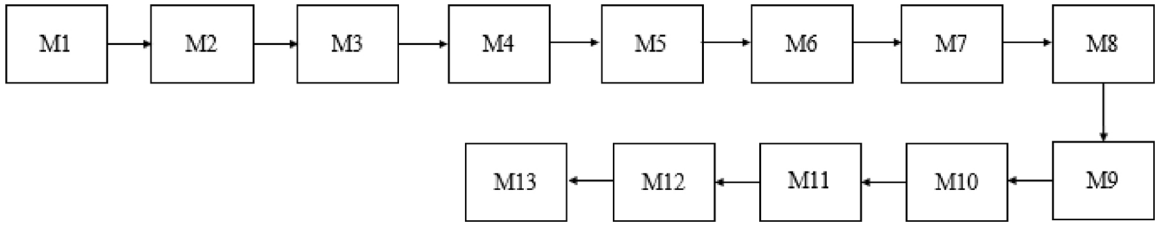


**Fig. 2.** Production line layout.

**Table 1**
Sample MES event log data from Machine 1.

| Machine ID | Event | Timestamp |
|---|---|---|
| M1 | Producing | 16-09-2017 06:00:01 |
| M1 | Stop tool failure | 16-09-2017 06:08:18 |
| M1 | Tool change | 16-09-2017 06:10:37 |
| M1 | Producing | 16-09-2017 06:20:54 |
| M1 | Waiting to unload | 16-09-2017 06:21:31 |

detect bottlenecks aids better understanding of their nature. Verification was also carried out if the event log data was deemed suitable for this method of bottleneck detection. As the event log data (Table 1) contains all the "active" events, the information on active durations can be obtained from it.

### 3.3. Module 3: data pre-processing

After selection of the bottleneck detection method, the next step is to pre-process the extracted event log data in accordance with the selected method. Each bottleneck detection method has a defined metric used to detect bottlenecks. The overall aim of the pre-processing module is to compute this metric based on the event log data for each machine in each production run within the defined time interval and then generate a time series for each machine. This time series is then used in the other modules to detect bottlenecks. The generic steps in pre-processing the event log data are: (1) data cleaning, (2) classification of event logs and (3) computation of metric from the required set of classes. The output of this module is a time series of the bottleneck detection metric for each machine. Each step is described below.

#### 3.3.1. Data cleaning

After extracting the event log data for each machine in the production systems, the event log data needs to be cleaned before further processing. The input of production system domain experts is needed in this step, as they can define time intervals across different production runs. Data cleaning can then be conducted. Common cleaning steps include: removing events that were recorded outside the defined time interval across different production runs; checking the event logs and removing duplications, as they might introduce bias in the analysis; removing events not of interest in bottleneck detection analysis (such as events recorded when running trail products, and so on). The output from this step is cleaned event log data for each machine.

In the demonstration production system, the cleaning was undertaken for each machine by checking and removing redundant events and those that were not of interest to the domain experts.

#### 3.3.2. Classification of event logs

The cleaned event log data contains entries for all events. However, some events might be redundant if they are not used in computing the required metric for the selected bottleneck detection method. In this step, the various events need to be selected and filtered for every machine based on the selected bottleneck detection method. This selection and filtering needs to be done based on event definitions. Event definitions are best provided by production system domain experts. This is important, as relevant events, corresponding to the selected method, will be used for further processing and to detect bottlenecks. The domain experts' involvement in providing these definitions for the set of unique events represented in the cleaned data is important, as no available ML technique can automatically define the events on their own. These event definitions are used to partition the data into a certain number of classes based on the selected bottleneck method. The output from this step is event log data from each machine, containing a classification for each event.

The demonstration production system follows the active period method of bottleneck detection. The different events in the event log data were classified as active or inactive based on the definitions of active and inactive states proposed in the active period method of bottleneck detection by Roser et al. [16]. To facilitate this process, the production system domain experts need to provide definitions for each event. There was a total of seven distinct types of pre-defined event, recorded across all machines. These events are classified as active or inactive, based on the definitions provided by the domain experts. The seven events and their classifications are shown in Table 2. Once the distinct events are classified, the classification is updated in the cleaned event logs for all machines.

#### 3.3.3. Time series generation

By this step, we have completed all the necessary pre-processing of event log data and are now ready to compute the metric corresponding to the chosen bottleneck detection method from the processed event log data. In this step, the metric values for each machine are computed across each individual production run for each machine. If required, these values are transformed into a uniform scale across all the machines in the production system, to ensure that scale differences are normalised before these values are used to generate time series. Let N

**Table 2**
Events definitions from domain experts and classification.

| Event | Description as given by domain experts | Classification made |
|---|---|---|
| Producing | Machine engaged in producing a product | Active |
| Stop tool failure | Machine stopped due to tool failure | Active |
| Producing with warning | Machine engaged in producing a product with a warning | Active |
| Breakdown | Machine is down or ongoing repair work | Active |
| Tool change | Machine under tool change activity | Active |
| Waiting to unload | Machine waiting for the unloading of the product after operations | Inactive |
| Waiting for incoming parts | Machine waiting for incoming parts | Inactive |

**Table 3**
An example calculation of active duration metric from event log data: from an active state event starting at 06:00:01 till an inactive state event occurring at 06:21:31.

| Machine ID | Event | Timestamp | Active duration metric (in seconds) |
|---|---|---|---|
| M1 | Producing | 16-09-2017 06:00:01 | 1290 |
| M1 | Stop tool failure | 16-09-2017 06:08:18 | |
| M1 | Tool change | 16-09-2017 06:10:37 | |
| M1 | Producing | 16-09-2017 06:20:54 | |
| M1 | Waiting to unload | 16-09-2017 06:21:31 | |

= {1,2,3,..., n} be the set of n production runs and M representing the set of m machines, such that M = {1,2,3,...,m}. The time series can be generated from this step and represented in the form of the matrix $T_{n\ x\ m,}$ with each row representing one production run and each column corresponding to a machine in the system. A cell $t_{ij} \in T$ records the scaled metric value of production run i for machine j. Thus, all values along column j represent the full time series corresponding to machine j.

In the demonstration production system, the active durations for all machines are computed during a single production run and across all production runs. This is calculated as the time elapsed between the start-time instant of an active state and the beginning of the inactive state during a production run. The algorithm for computing active durations from event log data is explained in Subramaniyan et al. [22] (page 233, equation 1). The same algorithm is used to compute the active duration of each machine across different production runs. A sample output of this step is shown in Table 3, which shows that the active durations are not evenly spaced time series.

Next, the active duration for each production run needs to be calculated by aggregating the different active durations during a production run. Thereafter, to put the active durations of different machines on a uniform scale, the active duration of each production run is expressed as a percentage of scheduled hours of that production run. The results are shown in the matrix T 30 x 13, in which each row represents the production run and each column represents the machine. All the values in a column are the time series for that particular machine.

$$T_{30\,x\,13} = \begin{bmatrix} 59.66 & 56.50 & 59.72 & 52.61 & 45.58 & 64.14 & 57.05 & 51.72 & 59.30 & 52.07 & 48.09 & 38.75 & 49.84 \\ 46.04 & 60.23 & 61.37 & 61.59 & 48.61 & 68.21 & 48.57 & 43.48 & 58.33 & 53.03 & 48.69 & 42.96 & 41.47 \\ 53.33 & 74.08 & 80.40 & 74.77 & 59.60 & 77.47 & 70.31 & 67.39 & 72.77 & 65.91 & 63.15 & 47.41 & 52.76 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 59.43 & 74.46 & 77.00 & 74.77 & 67.91 & 87.46 & 81.87 & 78.26 & 81.50 & 83.48 & 79.60 & 74.49 & 70.17 \\ 68.03 & 70.84 & 62.39 & 51.05 & 46.69 & 68.61 & 64.21 & 59.35 & 76.89 & 72.52 & 63.02 & 57.81 & 66.74 \end{bmatrix} \quad (1)$$

A time series plot is also constructed for each machine, as shown in Fig. 3. It can be seen that the time series plot is cluttered, and different machines have different temporal patterns. Spotting patterns manually in the time series between different machines and detect bottlenecks and non-bottlenecks is challenging. Applying ML techniques is advantageous for studying the individual time series profiles of machines, comparing them for their temporal behavioural similarities and differences, and separating the group of potential bottlenecks from the rest.

### 3.4. Module 4: generating a dendrogram

The first three modules jointly focused on preparing a time series format of the event log data that adheres to the chosen bottleneck detection method. Preparing the data in a suitable format is vital to revealing underlying patterns in the data and achieving the desired goal. The last step of the previous module also ensured that the data was ready for the application of unsupervised machine learning techniques; specifically, agglomerative hierarchical clustering to generate a dendrogram. Therefore, agglomerative hierarchical clustering is applied in this module. The reason for choosing hierarchical clustering and, in particular, the suitability of agglomerative hierarchical clustering, is summarised below.

Hierarchical clustering is advantageous for production system analysis because it is able to give a complete hierarchy of the machines in a production system. When used prior to running, this type of hierarchical clustering in production systems has been shown to find scheduling bottleneck clusters in job-shop type production systems [45]. Moreover, the hierarchical clustering results can be visualised through a dendrogram that explains how machines are grouped into a hierarchy. Various decisions need to be made before conducting hierarchical clustering, so as to select the right type. The whole strategy we followed in selecting the different variants of hierarchical clustering for bottleneck detection is illustrated in Fig. 4. Firstly, a choice needs to be made between agglomerative and divisive hierarchical clustering. In the proposed approach, agglomerative hierarchical clustering is suitable for bottleneck detection. It generates a complete tree, starting with individual machines, and orders the machines, which is useful for interpretation [45]. This will help domain experts better understand how different machines are grouped into a cluster. Agglomerative clustering works by iteratively combining the two closest machines into a new cluster and repeating this until one large cluster remains, encompassing all the machines in the production system. However, this method requires two input parameters: a measure for comparing individual time series, and choice of an appropriate linkage for clustering.

**Measures to compare individual time series**: There are various distance measures for comparing time series; these are explained in Section 2.2.1. The choice of a particular measure is crucial because different distance measures can give different results. Making an informed decision requires domain knowledge insights, plus arguments deduced from a literature study of real applications of ML. Various measures have been considered and it has been found that, as a distance measure, DTW is highly suitable for bottleneck detection. We summarise the reasons for this as follows. In real-world production systems, throughput bottlenecks tend to shift between machines during different production runs. This is also indicated in the literature by Li et al. [29] and Subramaniyan et al. [28]. Accordingly, this phenomenon indicates that the time series of different bottleneck machines may be time-shifted but the computations made using a common metric, such as active durations, can still show the same behaviour in limiting system throughput. On the other hand, using DTW can remove the time-shifts by wrapping the time axis of the machine such that maximal coincidence is attained which enables to achieve high similarity score with other machines exhibiting similar behavioural patterns. For instance, as seen in Fig. 3, there are non-linear fluctuations occurring in active durations (y-axis) of different machines versus time (x-axis). To address this, DTW uses a time normalisation effect where the fluctuations in the

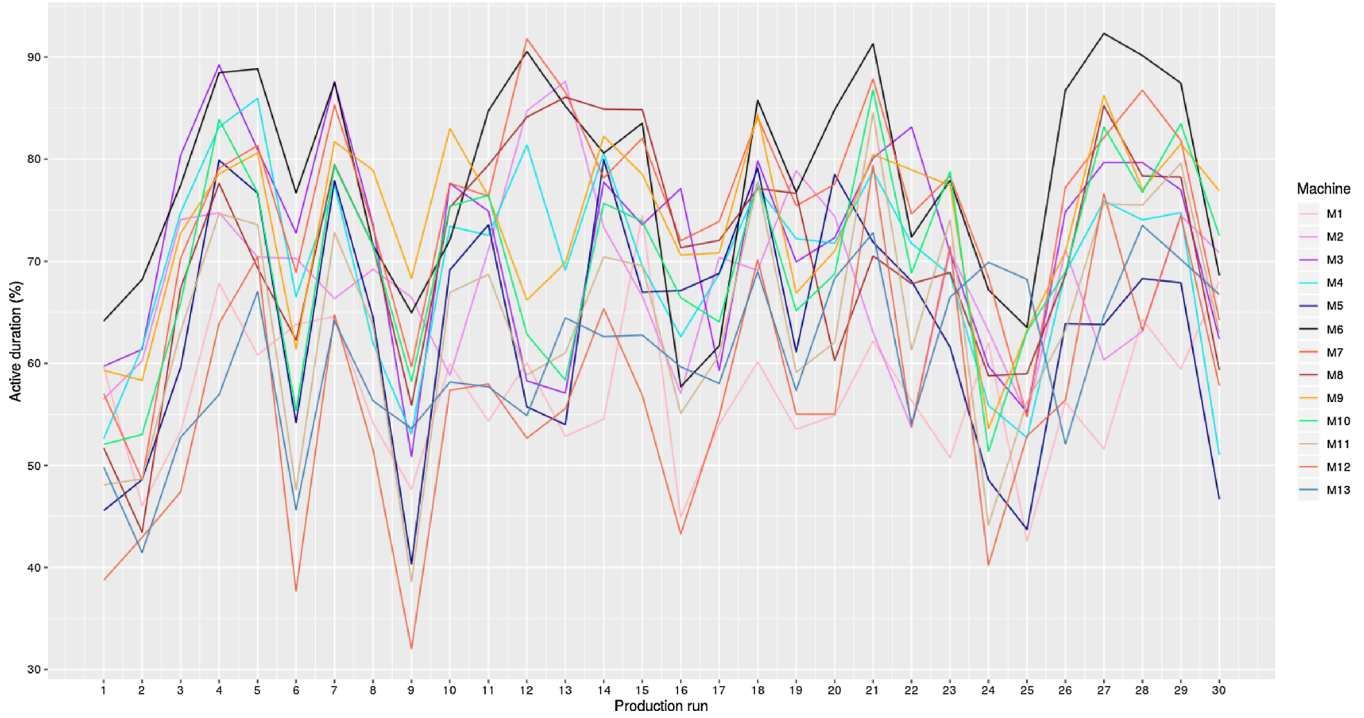


**Fig. 3.** Time series of individual machines.

time (x-axis) is modelled using a non-linear time-wrapping function [46]. Hence, using DTW as similarity measure can point to the group of throughput bottleneck machines that show similar behaviour in terms of limiting system throughput. An efficient dynamic programming based implementation of DTW is used to obtain a distance matrix which represents the distance measures for all pairs of machines in the production system.

**Choosing of an appropriate linkage method for agglomerative hierarchical clustering**: Once the distance matrix has been generated from the individual time series, a linkage method is required. This is used in agglomerative hierarchical clustering, to generate a dendrogram. A linkage method is simply a way of joining individual times series based on their behavioural characteristics in comparison to other time series. The behaviour is already captured and recorded in the form of a distance matrix. Like a distance measure, the linkage method also effects the final outcome. Selection of an appropriate linkage method should, therefore, be likewise motivated by a combination of domain knowledge and state-of-the-art practices in ML for similar problems. As mentioned in Section 2.2.1, among the various available choices, the complete linkage method was found suitable for bottleneck detection. This is because, in real-world production systems, there is a higher probability of bottlenecks shifting from one machine to the other during different production runs due to various reasons such as random maintenance stops in the machine, random processing times, and so on Li et al. [29] and Subramaniyan et al. [28]. This can also be inferred from Fig. 3, that the time series have large fluctuations in active durations (y-axis) versus time (x-axis). This means that there is not always one single machine which stays as throughput bottleneck across the entire set of productions runs, however, at times, it can be due to noise. The dynamism of changing bottlenecks is best captured in the chosen DTW distance measure. Now, a method which operates on the distance measures (DTW distance matrix) should be able to carry over the preserved behavioural patterns to produce a balanced dendrogram while as the same time be less susceptible to noise. The complete linkage method tends to fulfil these demands as indicated in the ML literature [37].

The two specific steps in this module are (1) application of DTW and (2) application of complete linkage agglomerative hierarchical clustering. These steps are defined below.

#### 3.4.1. Application of DTW

In this step, DTW is applied to compute the distance matrix between all pairs of machines, based on its time series represented in the matrix $T_{n\ x\ m}$. The output from this step is the distance matrix.

In the demonstration, DTW is applied to the time series data as shown in $T_{30 x 13}$. This results in a distance matrix, as shown in Table 4. The smaller the values for a pair of machines, the more similar their behaviour. For example, machine M1 achieves smallest distance with M13, which means M1's behaviour is most similar to M13.

#### 3.4.2. Application of complete linkage hierarchical clustering

In this step, the generated distance matrix is used to conduct hierarchical clustering with complete linkage criteria by applying DTW recursively. The result of the clustering procedure is a dendrogram representing the hierarchical relationship between the machines.

In the demonstration, agglomerative hierarchical clustering with complete linkage is applied based on distance matrix shown in Table 4. The resultant dendrogram is shown in Fig. 5. Along the horizontal axis in this figure are the machines in the production system. The vertical axis is the distance, represented as the height between different pairs of machines. A quick visual inspection of the dendrogram indicates which machines are close to which others, with respect to their behavioural characteristics captured from the respective time series. For example, M6 shows completely different behaviour from M5 and M11. It is easy to see such obvious differences from the dendrogram. However, at this point, it cannot be confirmed which machine is potentially a throughput bottleneck, so further analysis is required. Overall, the dendrogram assists in immediately identifying and highlighting the machine that shows behaviour distinct from the rest of the machines in the system. We then need a way to recommend how many clusters can be generated through the dendrogram. This is explained in Module 5.

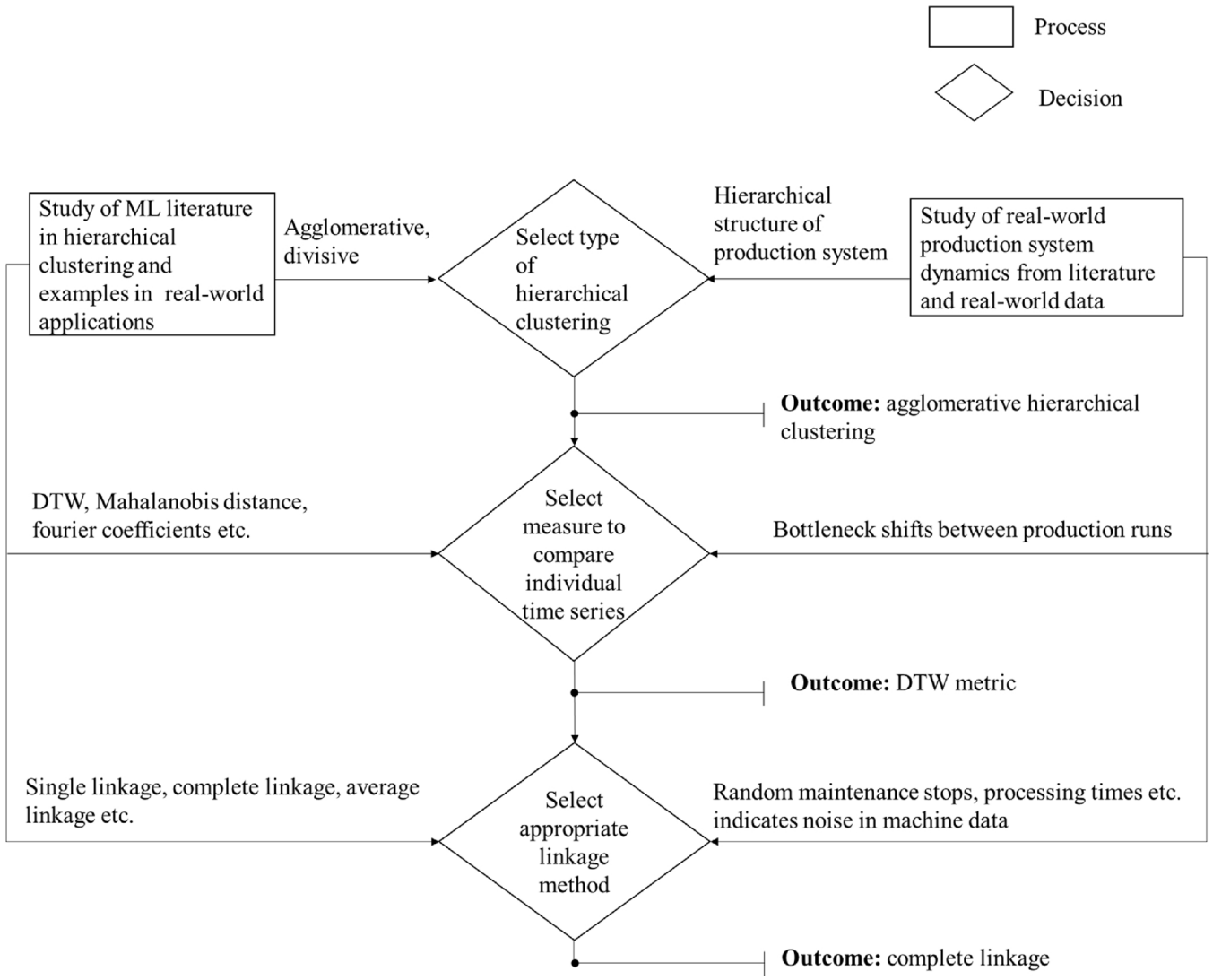


**Fig. 4.** Strategy followed to develop Module 4.

### 3.5. Module 5: cluster computation and generation

Once a dendrogram has been generated, clusters must be generated from it. This module explains how. Two steps are involved: (1) selecting the number of clusters and (2) extracting the machine information for each cluster. The output of this module is the number of clusters generated and identification of machines within each one.

#### 3.5.1. Selecting the number of clusters

In this step, the number of clusters needs to be selected. This can be done by placing a horizontal partition line across the dendrogram. However, partition line should typically not be placed at the level of granularity where every machine is its own cluster and not at this coarse granularity, where all machines are in one cluster. The challenge is finding the best position in the dendrogram to place the line. This can be solved by using a combined approach of domain experts' input and

**Table 4**
Distance matrix, applying DTW.

| | M1 | M2 | M3 | M4 | M5 | M6 | M7 | M8 | M9 | M10 | M11 | M12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| M2 | 283.94 | | | | | | | | | | | |
| M3 | 380.27 | 263.51 | | | | | | | | | | |
| M4 | 297.12 | 210.54 | 216.90 | | | | | | | | | |
| M5 | 335.25 | 296.86 | 265.50 | 256.68 | | | | | | | | |
| M6 | 496.22 | 317.97 | 305.41 | 332.52 | 415.95 | | | | | | | |
| M7 | 348.95 | 264.13 | 220.46 | 265.89 | 313.92 | 236.51 | | | | | | |
| M8 | 311.59 | 232.48 | 237.32 | 216.65 | 292.14 | 273.49 | 202.71 | | | | | |
| M9 | 320.23 | 224.71 | 261.60 | 228.43 | 341.62 | 265.21 | 224.02 | 231.86 | | | | |
| M10 | 323.51 | 267.06 | 239.82 | 250.96 | 292.54 | 298.14 | 282.32 | 224.72 | 204.38 | | | |
| M11 | 295.17 | 281.61 | 278.17 | 300.53 | 250.32 | 392.62 | 323.94 | 300.44 | 322.45 | 284.79 | | |
| M12 | 283.74 | 340.26 | 383.78 | 345.59 | 350.91 | 526.07 | 362.51 | 334.22 | 407.40 | 378.86 | 267.20 | |
| M13 | 214.45 | 231.00 | 351.95 | 277.56 | 278.43 | 462.75 | 347.01 | 309.18 | 318.06 | 280.62 | 264.23 | 257.55 |

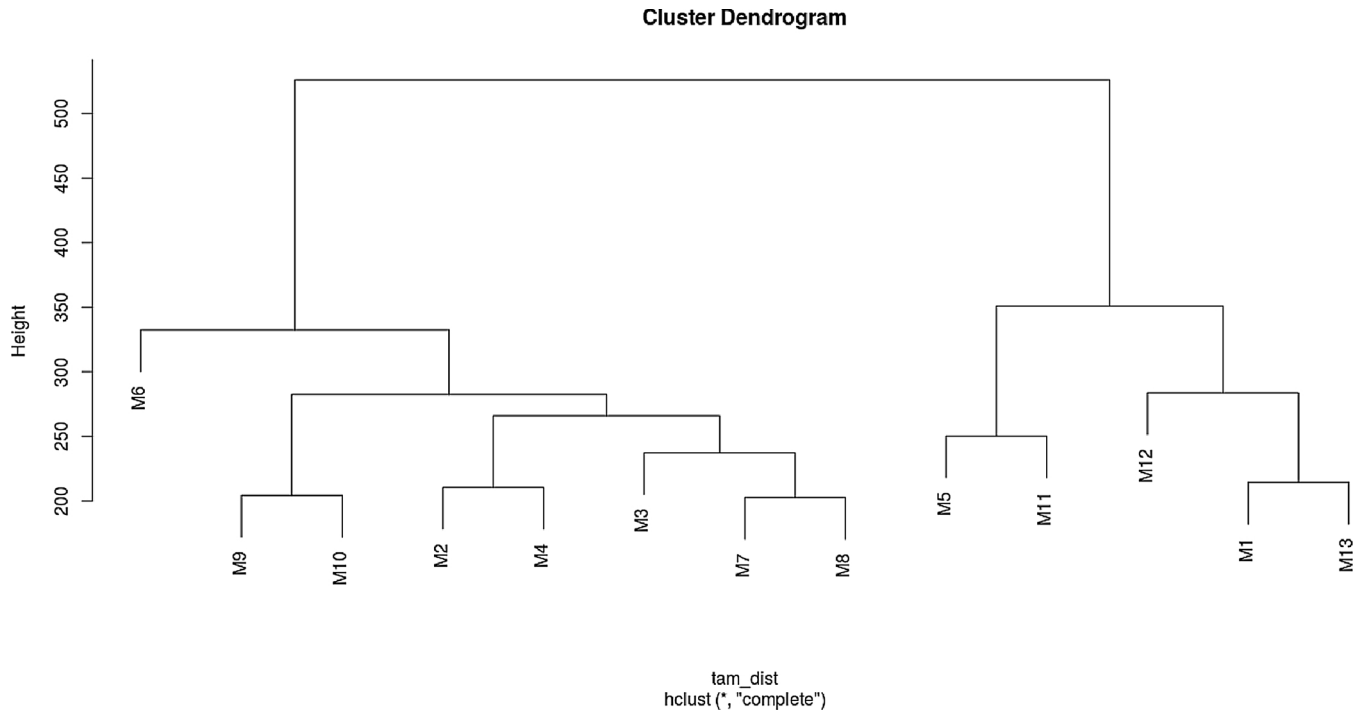


**Fig. 5.** Hierarchical clustering results represented in a dendrogram.

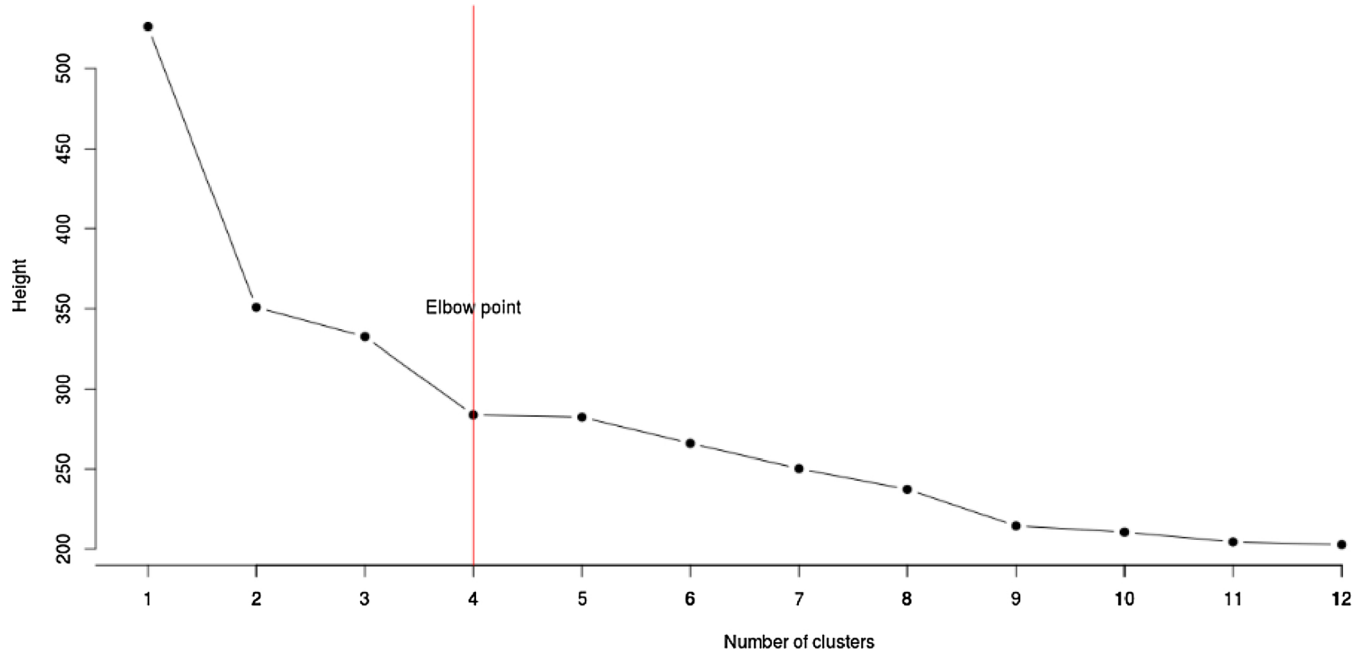


**Fig. 6.** Elbow plot as a support tool to determine the number of clusters.

visual aids provided by the algorithms, as explained in Section 2.2.2. This is because, when placing a partition line, a lot of domain-specific intuition and contextual information on the given production system comes into play. Moreover, this combined approach is important in quantifying the trade-off between the complexity of having a large number of clusters (not useful in practice) and having too few clusters (which reduces the resolution when trying to find bottlenecks). Once the partition line is placed, the clusters can be generated.

In this demonstration, based on the nature of the production system, it was decided to partition the dendrogram at a height of 300, resulting in four clusters. This choice was verified by an elbow plot to assess the chosen number of clusters. The elbow plot for the hierarchical clustering is shown in Fig. 6. The height represented on the vertical axis represents the variance. As can be seen, there is a significant change in the slope before the fourth cluster. This indicates that four clusters could be a reasonable number to consider for further analysis. Fig. 7 shows the different clusters in the dendrogram.

### 3.5.2. *Extracting the machine information of each cluster*

Once the clusters are generated, the machines assigned to each one need to be extracted. This is done so as to understand which machines in the production system correspond to which cluster. This information

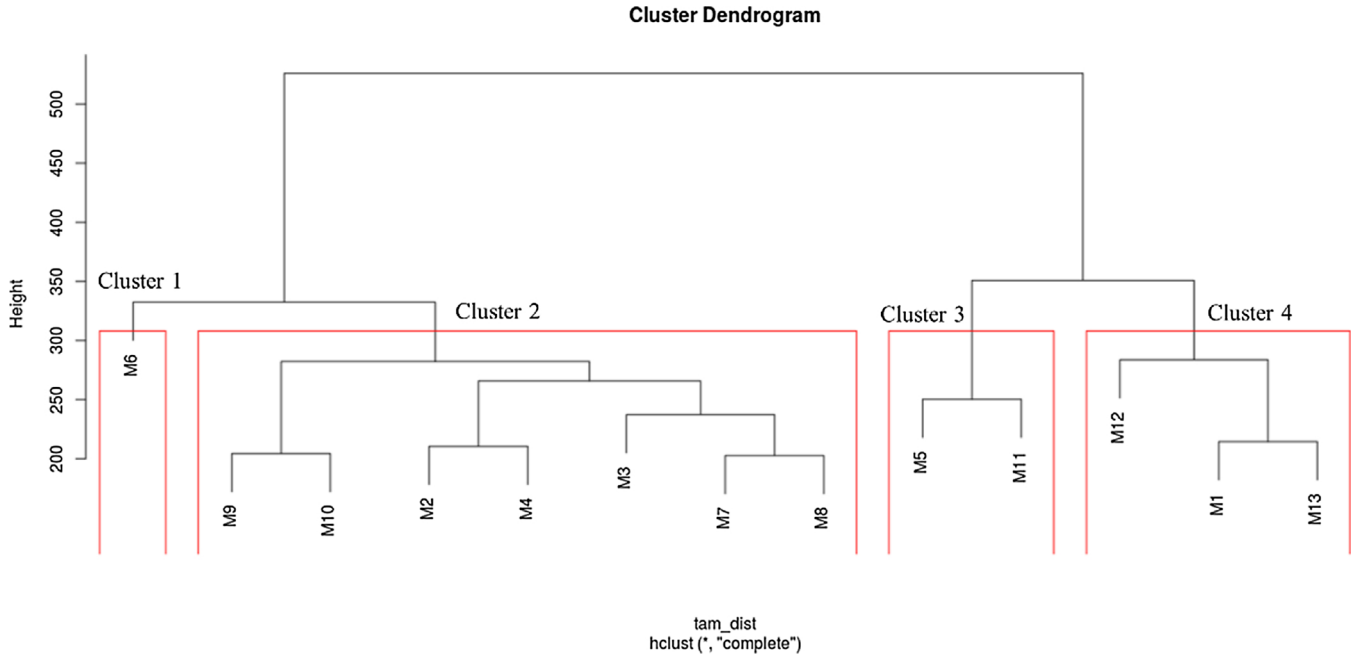


**Fig. 7.** Dendrogram representing the number of clusters.

**Table 5**
Machines in each cluster.

| Cluster number | Machines |
|---|---|
| 1 | M6 |
| 2 | M9, M10, M2, M4, M3, M7, M8 |
| 3 | M5, M11 |
| 4 | M12, M1, M13 |

is necessary to analyse the clusters and detect bottlenecks.

In the demonstration, Table 5 shows the assignment of machines to each cluster.

### 3.6. Module 6: representative time series generation

Information on the number of clusters and the assignment of each machine to those clusters was obtained from the previous module. In this module, the clusters need to be analysed in detail to facilitate throughput bottleneck detection. For this purpose, a representative time series can be generated for each cluster. These are constructed by computing the averages of each data point for the different individual time series in that cluster, as explained in Baheti and Toshniwal [40]. To generate the representative time series, we need the individual time series for each machine. These are extracted from the matrix $T_{n \times m}$. The output of this module is the generation of representative time series for each cluster and visual representation of them on a line graph.

In this demonstration, we generate a representative time series for each cluster shown in Table 5. The resultant representative time series are visualised in Fig. 8. By interpreting this figure, throughput bottlenecks can be detected visually, as explained in Module 7.

### 3.7. Module 7: throughput bottleneck detection

In this module, probable throughput bottlenecks are determined by visual analysis of the generated representative series of each cluster. As a first step in this module, the domain experts need to carry out a visual inspection of the representative time series for each cluster, according to the chosen bottleneck detection method. For this purpose, domain experts need to be aware of how to interpret basic line plots. If the representative time series for the clusters are well separated from each other overall, then the domain experts can proceed with interpreting the representative time series for each cluster and detect the cluster with bottleneck machines. If not, the domain experts can re-evaluate the number of clusters and repeat the computations of Modules 5 and 6. The process of repeating Modules 5 and 6 is depicted with a feedback loop from Module 7 to Module 5, as shown in Fig. 1.

In this demonstration, it can be seen from Fig. 8 that the representative time series for each of the four clusters is fairly well separated from one another. Also, it is clear that the active duration of Cluster 1 is higher than the other clusters across most of the production runs and that by using the active period method, the highest active duration is the bottleneck. Therefore, of the machines in Cluster 1, machine M6 is the probable primary bottleneck in the production system. Hence, it may be inferred that, compared to other machines in the production system, M6 shows unique throughput-limiting behaviour for most of the production runs. It can also be seen from Fig. 8, that for production runs 10, 17 and 18, Cluster 2 has the highest active duration. In other words, it may be said that for these production runs, the primary bottlenecks shift between the machines in Cluster 1 and those in Cluster 2. A deeper analysis, based on other contextual information for the production runs, is required to determine the reasons for shifting bottlenecks. Domain experts may examine other data sources to determine the reason for the shift during those production runs. Moreover, from Fig. 8, it can be seen that Clusters 3 and 4 have lower active period percentages than Clusters 1 and 2 for most of the production runs and are indicated as non-bottlenecks.

Overall, it may be seen that, from examining the time series for active duration profiles generated from individual machines' event data logs (as shown in Fig. 3), it was not immediately obvious which machine was the bottleneck. The proposed approach used hierarchical clustering (as shown in Fig. 8) to form groups of machines, based on their temporal behavioural patterns, and reveal the cluster in which the bottleneck(s) lay. Domain experts may augment the results obtained with the other contextual information about the machines (by, say,

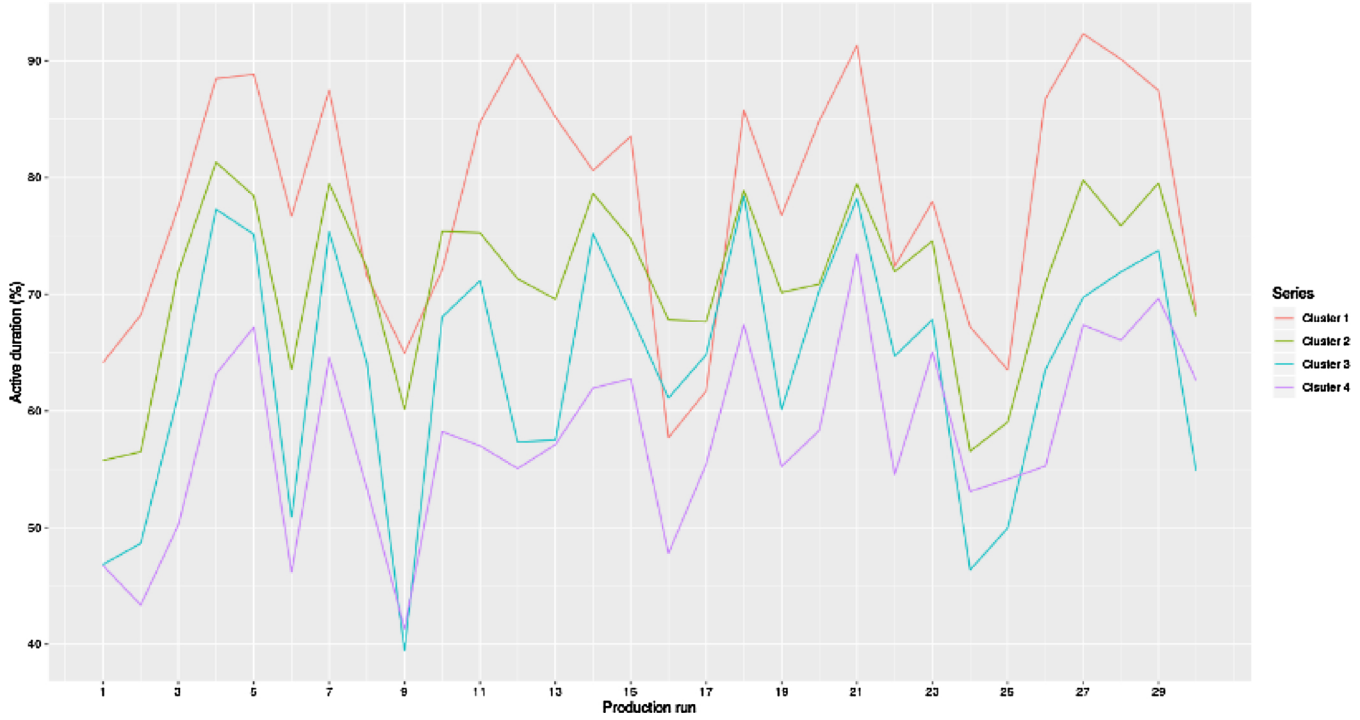


**Fig. 8.** Plot of the representative time series of each cluster.

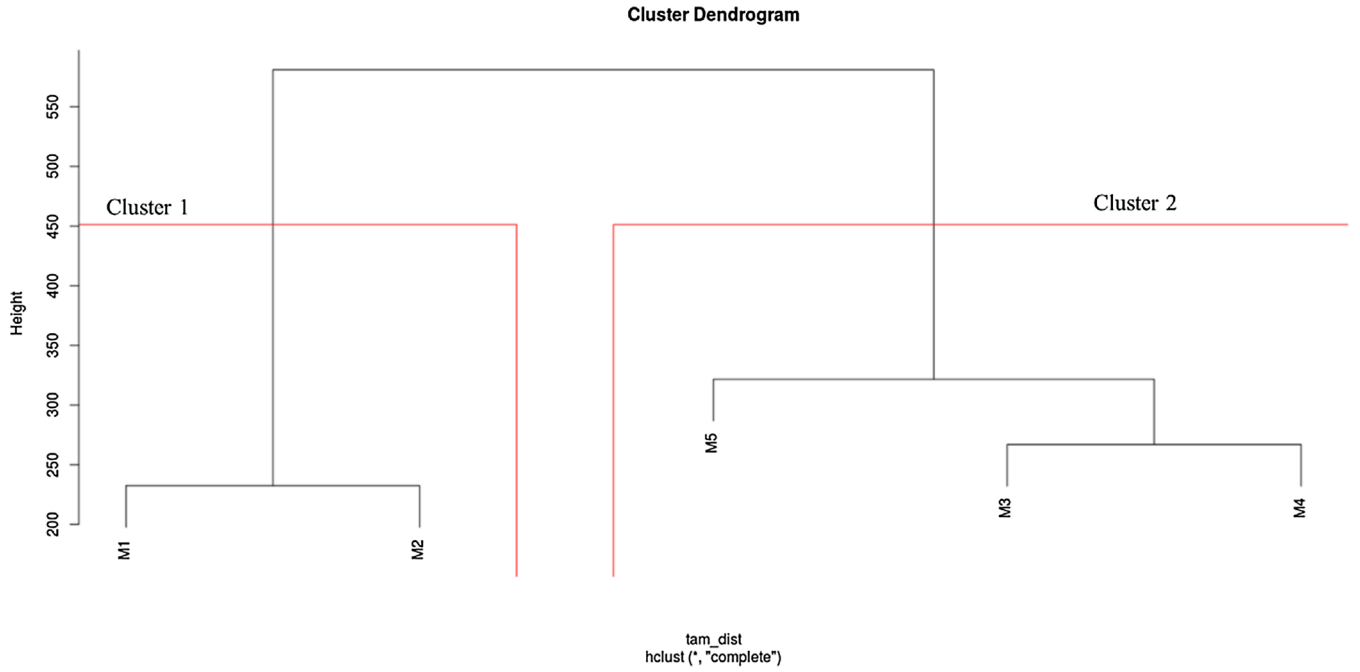


**Fig. 9.** Dendrogram representing possible machine clusters.

examining other information systems such as sensor data, logistical information, and so on) to determine which machines need to be prioritised for throughput improvement activities.

## 4. Additional real-world test study to evaluate the proposed approach

The proposed approach was applied across another real-world production system, to detect throughput bottlenecks using the active period method of bottleneck detection. As mentioned in Section 3, all the implementation of this test study was carried out using R software.

The automated serial production system has five crankshaft CNC machines (M1, M2, M3, M4 and M5) and the event log data from their 44 historical production runs was analysed to detect bottlenecks. The active period bottleneck detection method (with active duration metric) was selected to describe the machine behaviour and detect bottlenecks in this production system. The proposed approach was used to generate the dendrogram, as shown in Fig. 9. It was then decided to have two clusters, based on the nature of the production system. This was further verified using elbow plots. Fig. 9 shows that, after generating the clusters, M1 and M2 belonged to Cluster 1 and M3, M4 and M5 belonged to Cluster 2.


**Fig. 10.** Representative time series of Clusters 1 and 2.

Thereafter, representative time series were constructed for the two clusters, with the results shown in Fig. 10. This figure clearly shows that Cluster 2 has the highest active duration of all production runs (except production run 11), indicating that bottleneck machines are present within Cluster 2. From Fig. 9, it can also be seen that the height at which M3 and M4 merge with M5 happens quite quickly, indicating that the bottlenecks are very prone to shifting between M3, M4 and M5 across production runs. In other words, M3, M4 and M5 have similar active duration time series profiles. This can also be confirmed by visual inspection of the active duration time series for M3, M4 and M5 as shown in Fig. 11. This reveals a high level of shifting in the primary bottlenecks between M3, M4 and M5 in different production runs. Hence, it may be concluded that M3, M4 and M5 constitute a group of potential primary bottlenecks in the production system. These machines may be the focus of improvement actions aimed at increased throughput. These findings highlight the consistency of the proposed approach when applied to a different production system, yet are attuned to the particularities of the data.


**Fig. 11.** Individual time series of M3, M4 and M5.

## 5. Discussion

This section discusses the academic and practical contributions of the proposed throughput bottleneck detection method using unsupervised ML techniques. The limitations and some possible future directions based on the proposed research are then presented.

### 5.1. Academic contributions

The state of art in data-driven throughput bottleneck detection [16,18–25] has focused on detecting throughput bottlenecks via statistical approaches which use descriptive and inferential statistical techniques. When developing such statistical-based approaches, different assumptions were used which were not explicitly described in the literature. This study outlines the different assumptions (such as assumptions on statistical distribution, auto-correlation, cross-correlation and stationarity as explained in Section 2.1) made in different statistical-based approaches to detecting bottlenecks. Outlining these explicit assumptions can be useful to any researcher intending to use statistical-based approaches; they can check whether the assumptions hold good for the dataset being studied. On the other hand, ML-based approaches are becoming increasingly popular in developing better approaches, without relying on strict statistical assumptions. Taking advantage of such ML techniques, this study proposes a generic hierarchical clustering approach to detecting throughput bottlenecks, with its usefulness successfully demonstrated on two real-world production systems. The bottleneck machines in a production system are those which behave uniquely among the other machines in the system. A hierarchical clustering ML technique is used to study the behavioural patterns of machines. Machines which show similar behavioural characteristics are clustered, which leads to the detection of bottlenecks. The natural phenomenon of shop-floor operational dynamics often varies over time. Thus, dynamically changing patterns are better learned from the data itself, without imposing any strict statistical rules. Compared to the existing literature on statistical approaches, the proposed approach emerges as a thorough approach which achieves its novelty by using: (1) an unsupervised ML approach, (2) DTW as a generic distance measure, which best uncovers the dynamics of the time series representation of a pair of machines using the chosen bottleneck detection method and (3) recursive application of DTW in agglomerative hierarchical clustering, to facilitate the identification of clusters of machines forming different behavioural patterns based on historical data.

The proposed approach has three main advantages. Firstly, it uses the complete time series from the machine data and thus encompasses all temporal information to form clusters, rather than using multiple statistical descriptors. Secondly, we also note that the methods may be coupled with any bottleneck detection technique in Module 2, as explained in Section 4. In the demonstrations, the active period method is used to detect bottlenecks. However, other methods, such as turning point method [19] and inter-departure time variance method [20], can be coupled with the proposed approach by adjusting the computation of related metrics from the event log data. The underlying logic is that whichever detection method is applied (and when the corresponding metric is derived from MES data), the output is a number which can be turned into a time series. This allows the proposed approach to be applied. Lastly, the proposed approach can be scaled to detect bottlenecks in larger production systems and even up to factory level. Using the statistical-based approach requires statistical modelling for each machine and thus increases the computational complexity – a time-consuming task. Using our proposed approach, any number of machines can easily be grouped according to their behaviour. This enables rapid detection of bottlenecks.

### 5.2. Practical contributions

This section discusses the contributions of two types of practice. Firstly, there is a presentation of the contribution of bottleneck management to shop-floor practice. This discussion can help production system domain experts better understand the usefulness of the proposed approach, thereby enhancing practice. Secondly, there is a presentation of the contribution to data scientists developing ML-based techniques for production system analysis. This can help applied data scientists (who are developing ML-based approaches) to consider different perspectives when applying ML to production system analysis.

#### 5.2.1. Shop floor practice

On the shop floor, one of the main challenges faced by domain experts is identifying the set of throughput bottleneck machines in the system and initiating activities, such as prioritising reactive maintenance work orders in bottlenecks, buffering bottlenecks, and so on. Using our proposed approach, domain experts can use production system data to identify the probable set of bottlenecks constraining throughput. They can then augment their data-driven results with other contextual information from the production system and select the final set of throughput bottlenecks. These bottlenecks can then be prioritised for improvement activities to maximise overall system throughput. Moreover, the dendrogram generated, with hierarchical clusters and shown in Figs. 7 and 9, may correspond to meaningful taxonomies of the production system and can be used as a decision-support tool. The domain experts can then use this production system dendrogram, examining its different levels and why the relationships between machines seen in the data may, or may not, exist in practice. These experts can gain a systems perspective of the production system through hierarchical clustering; something not fully revealed by computing the descriptive and inferential statistics in machine data.

Also, over time, it is common practice for domain experts to develop a tacit understanding of the system as a whole, as well as the behaviours of individual machines relative to other machines. The proposed approach can be used to check whether machines believed to exhibit the same behaviour show up in the same cluster. At the same time, such beliefs could be proved incorrect by showing the range of production runs when those machines (or the system as a whole) behaved differently. For example, domain experts may believe that M6 (as demonstrated in Fig. 1) is always the bottleneck. Whilst the results obtained using the proposed approach (as shown in Fig. 8) also indicate that M6 is the bottleneck for most production runs, it also reveals those production runs where M6 was not. So, the proposed approach captures local variations whilst providing domain experts with a comprehensive outlook on the production system for the production runs being considered.

Overall, this paper has explored the use of unsupervised ML approaches to support the detection of bottlenecks on the shop floor using digital machine data. This is a potential contribution to the trend of using ML-based approaches for better decision-making in industrial practice, as highlighted by Wuest et al. [9] and Bokrantz et al. [11]. It may be anticipated that such a demonstration of a solution to a highly relevant practical problem will have the potential to accelerate practical implementation of Industry 4.0.

#### 5.2.2. ML applied in practice

Through the proposed approach, we provide data scientists with an easy-to-use yet powerful technique for the application of throughput bottleneck detection. Through it, we also provide guidelines on how data scientists may collaborate with production system domain experts to develop ML-based solutions to practical problems. Studying real-world data helps data scientists gain insights on real-world production system dynamics. Moreover, studying ML-based literature will give insights into the different ML techniques, which may be useful in designing the data-driven approach. Combining these two aspects, the

data scientists can develop rigorous ML-based, data-driven approaches to the problem. However, including the input of production system domain experts (with their significant, tacit and explicit knowledge of production systems) in designing an ML-based approach, will enhance the practical relevance of the developed approach to practical decision-making. Such approaches will also increase the accuracy and usability of such data-driven approaches [9]. Moreover, including domain experts' input in this way helps data scientists generate new hypotheses on real-world production systems; something not entirely evident just from studying real-world data. Incorporating the domain knowledge will generate plausible explanations and better augment the domain expert's decision-making on bottlenecks. Overall, production system domain experts can help data scientists to develop highly relevant ML-based solutions. They will have significant practical impact and their active involvement and input will contribute to the emerging concept of humans in-loop within modern ML practices as indicated by Cimini et al. [12].

### 5.3. Limitations and future research

Although there are certain limitations to the proposed study, the reported results reveal important directions which may, potentially, steer future work in using ML-based techniques to drive throughput improvement activities. Firstly, this study is limited to only detecting throughput bottlenecks from event log data. Although this is the first step in driving throughput improvement activities, additional data sources are required to diagnose throughput bottlenecks and then plan for specific improvement actions. Future work might involve coupling additional data sources in clustering machines, something which could provide broader support to the initial results reported here. Secondly, the proposed approach requires sufficient historical data. This information is necessary in order to detect throughput bottlenecks and is defined by the domain experts in this study. However, to complement this input, there is an opportunity to have an ML-based approach to automatically detect the length of historical data needed for throughput bottleneck analysis. This may be an interesting future direction for data scientists. Lastly, the proposed unsupervised clustering approach to bottleneck detection opens up an extremely promising direction, that of designing recommender systems to pinpoint specific and necessary action on bottlenecks. The recommender systems may be designed by blending the proposed approach with the action log data on bottlenecks.

## 6. Conclusion

Detecting throughput bottlenecks is necessary to improve production system throughput and increase productivity. As production systems become more and more complex, and as large-scale machine data become available, robust tools for detecting throughput bottlenecks are of critical importance. This paper proposes an unsupervised, ML-based, hierarchical clustering approach to throughput bottleneck detection. Our proposed approach studies the behaviour of machines within a defined timeframe and clusters them based on their temporal behavioural characteristics. Studying the characteristics of each cluster helps identify probable bottleneck machines in the production system. The proposed approach has been demonstrated on two real-world production systems and, when used in shop-floor practice, helps production system domain experts to quickly identify bottlenecks in the system from large sets of machine event log data. This study also emphasises the importance of including domain experts' input when developing ML-based solutions and shows how this can be realised for the problem of throughput bottleneck detection.

## 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.

## Acknowledgements

The authors would like to thank the FFI programme (funded by VINNOVA, the Swedish Energy Agency and the Swedish Transport Administration) for their funding of the Data Analytics in Maintenance Planning research project (DAiMP) [Grant number: 2015-06887], under which this research was conducted. The authors would also like to thank Anders Ramström, who furnished real-time data from a real-world production system. The authors would also like to thank the other industrial partners in the DAiMP research project, for sharing their views on the importance of bottleneck detection in manufacturing. This work has been conducted under the Sustainable Production Initiative and Production Area of Advance at Chalmers.

## References

[1] Miller S. AI: Augmentation, more so than automation. Asian Manag Insights 2018;5:1–20.
[2] Wilson HJ, Daugherty PR. Collaborative intelligence: humans and AI are joining forces humans. Harv Bus Rev 2018;96:115–23.
[3] Aghabozorgi S, Teh YW. Stock market co-movement assessment using a three-phase clustering method. Expert Syst Appl 2014;41:1301–14. https://doi.org/10.1016/j.eswa.2013.08.028.
[4] De Prado ML. Building diversified portfolios that outperform out-of-sample. J Portf Manag 2016.
[5] Emerson S, Kennedy R, Shea LO, Brien JO. Trends and applications of machine learning in quantitative finance. 8th Int. Conf. Econ. Financ. Res. (ICEFR 2019) 2019.
[6] Tao F, Qi Q, Liu A, Kusiak A. Data-driven smart manufactuirng. J Manuf Syst 2018;48:157–69. https://doi.org/10.1016/j.jmsy.2018.01.006.
[7] Xia T, Dong Y, Xiao L, Du S, Pan E, Xi L. Recent advances in prognostics and health management for advanced manufacturing paradigms. Reliab Eng Syst Saf 2018. https://doi.org/10.1016/j.ress.2018.06.021.
[8] Carvalho TP, Soares FAAMN, Vita R, Francisco R da P, Basto JP, Alcalá SGS. A systematic literature review of machine learning methods applied to predictive maintenance. Comput Ind Eng 2019. https://doi.org/10.1016/j.cie.2019.106024.
[9] Wuest T, Weimer D, Irgens C, Thoben K. Machine learning in manufacturing : advantages, challenges, and applications. Prod Manuf Res 2016;4:1–23. https://doi.org/10.1080/21693277.2016.1192517.
[10] Sharp M, Ak R, Jr TH. A survey of the advancing use and development of machine learning in smart manufacturing. J Manuf Syst 2018;48:170–9. https://doi.org/10.1016/j.jmsy.2018.02.004.
[11] Bokrantz J, Skoogh A, Berlin C, Wuest T, Stahre J. Smart maintenance: a research agenda for industrial maintenance management. Int J Prod Econ 2019. https://doi.org/10.1016/j.ijpe.2019.107547.
[12] Cimini C, Pirola F, Pinto R, Cavalieri S. A human-in-the-loop manufacturing control architecture for the next generation of production systems. J Manuf Syst 2020. https://doi.org/10.1016/j.jmsy.2020.01.002.
[13] Li L, Ni J. Short-term decision support system for maintenance task prioritization. Int J Prod Econ 2009;121:195–202. https://doi.org/10.1016/j.ijpe.2009.05.006.
[14] Wu K, Zhou Y, Zhao N. Variability and the fundamental properties of production lines. Comput Ind Eng 2016;99:364–71. https://doi.org/10.1016/j.cie.2016.04.014.
[15] Alavian P, Eun Y, Meerkov SM, Zhang L. Smart production systems : automating decision- making in manufacturing environment. Int J Prod Res 2019:1–18. https://doi.org/10.1080/00207543.2019.1600765.
[16] Roser C, Nakano M, Tanaka M. A practical bottleneck detection method. In: Peters B, Smith J, Medeiros D, Rohrer M, editors. Proc. 2001 Winter Simul. Conf., Arlington, VA 2001. p. 949–53. https://doi.org/10.1109/WSC.2001.977398.
[17] Goldrat E, Cox J. The goal: a process of ongoing improvement Third Revi. Great Barrington, MA: North River Press; 1990.
[18] Roser C, Nakano M, Tanaka M. Shifting bottleneck detection. Yucesan E, Chen C-H, Snowdon J, Charnes J, editors. Proc. 2002 Winter Simul. Conf. vol. 2. 2002. https://doi.org/10.1109/WSC.2002.1166360.
[19] Li L, Chang Q, Ni J. Data driven bottleneck detection of manufacturing systems. Int J Prod Res 2009;47:5019–36. https://doi.org/10.1080/00207540701881860.
[20] Betterton CE, Silver SJ. Detecting bottlenecks in serial production lines – a focus on interdeparture time variance. Int J Prod Res 2012;50:4158–74. https://doi.org/10.1080/00207543.2011.596847.
[21] Tang H. A new method of bottleneck analysis for manufacturing systems. Manuf Lett 2019. https://doi.org/10.1016/j.mfglet.2019.01.003.
[22] Subramaniyan M, Skoogh A, Salomonsson H, Bangalore P, Gopalakrishnan M, Sheikh Muhammad A. Data-driven algorithm for throughput bottleneck analysis of production systems. Prod Manuf Res 2018;6:225–46. https://doi.org/10.1080/21693277.2018.1496491.
[23] Li L. A systematic-theoretic analysis of data-driven throughput bottleneck detection of production systems. J Manuf Syst 2018;47:43–52. https://doi.org/10.1016/j.

jmsy.2018.03.001.
[24] Pehrsson L, Ng AHC, Bernedixen J. Automatic identification of constraints and improvement actions in production systems using multi-objective optimization and post-optimality analysis. J Manuf Syst 2016;39:24–37. https://doi.org/10.1016/j.jmsy.2016.02.001.
[25] Yu C, Matta A. Data-driven bottleneck detection in manufacturing systems: a statistical approach. Int J Prod Res 2016;54:6317–22. https://doi.org/10.1080/00207543.2015.1126681.
[26] Amrhein V, Trafimow D. Inferential statistics as descriptive statistics: there is No replication crisis if we don't expect replication. Am Stat 2019;73:262–70. https://doi.org/10.1080/00031305.2018.1543137.
[27] Roser C, Nakano M. Confidence interval from a single simulation using delta method. JSME Int J Ser C Mech Syst Mach Elem Manuf 2003;46:67–72. https://doi.org/10.1299/jsmec.46.67.
[28] Subramaniyan M, Skoogh A, Salomonsson H, Bangalore P, Bokrantz J. A data-driven algorithm to predict throughput bottlenecks in a production system based on active periods of the machines. Comput Ind Eng 2018;125:533–44. https://doi.org/10.1016/j.cie.2018.04.024.
[29] Li L, Qing C, Xiao G, Ambani S. Throughput bottleneck prediction of manufacturing systems using time series analysis. J Manuf Sci Eng 2011;133:1–8. https://doi.org/10.1115/1.4003786.
[30] Hastie T, Tibshirani R, Friedman J. The elements of statistical learning. Second edi New York: Springer Series in Statistics; 2001.
[31] Keogh E, Lin J. Clustering of time-series subsequences is meaningless : implications for previous and future research. Knowl Inf Syst 2005;8:154–77. https://doi.org/10.1007/s10115-004-0172-7.
[32] Balcan MF, Liang Y, Gupta P. Robust hierarchical clustering. J Mach Learn Res 2014;15:4011–51.
[33] Davidson I, Ravi SS. Agglomerative hierarchical clustering with constraints: theoretical and empirical results. Eur. Conf. Princ. Data Min. Knowl. Discov.. Berlin Heidelberg: Springer; 2005. p. 59–70. https://doi.org/10.1007/11564126_11.
[34] Jain A, Murty M, Flynn PJ. Data clustering: a review. ACM Comput Surv 1999;31:264–323.
[35] Serr J, Arcos JL. An empirical evaluation of similarity measures for time series classification. Knowledge-Based Syst 2014;67:305–14. https://doi.org/10.1016/j.knosys.2014.04.035.
[36] Mueen A, Keogh E. Extracting optimal performance from dynamic time warping. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min. 2016. p. 2129–30.
[37] Hirano S, Tsumoto S. Empirical comparison of clustering methods for Long time-series databases. In: Tsumoto S, Yamaguchi T, Numao M, Motoda H, editors. Act. Min. Berlin Heidelberg: Springer; 2005. p. 275–94.
[38] Zambelli AE. A data-driven approach to estimating the number of clusters in hierarchical clustering. F1000Research 2017;5:1–13. https://doi.org/10.12688/f1000research.10103.1.
[39] Thorndike RL. Who belongs in the family? Psychometrika 1953;18:267–76. https://doi.org/10.1007/BF02289263.
[40] Baheti A, Toshniwal D. Trend analysis of time series data using data mining techniques. IEEE Int. Congr. Big Data, IEEE 2014:430–7. https://doi.org/10.1109/BigData.Congress.2014.69.
[41] Baheti A, Toshniwal D. Finding representative time sequence for trend analysis. CSI Trans ICT 2014;2:181–90. https://doi.org/10.1007/s40012-014-0056-2.
[42] Kumar S, Toshniwal D. A novel framework to analyze road accident time series data. J Big Data 2016;3:1–11. https://doi.org/10.1186/s40537-016-0044-5.
[43] Wirth R, Hip J. CRISP-DM: towards a standard process model for data mining. Proc. Fourth Int. Conf. Pract. Appl. Knowl. Discov. Data Min. 2000:29–39. doi:10.1.1.198.5133.
[44] Huber S, Wiemer H, Schneider D, Ihlenfeldt S. DMME: data mining methodology for engineering applications - a holistic extension to the CRISP-DM model. Procedia CIRP 2019;79:403–8. https://doi.org/10.1016/j.procir.2019.02.106.
[45] Lei Q, Li T. Identification approach for bottleneck clusters in a job shop based on theory of constraints and sensitivity analysis. Proc Inst Mech Eng Part B J Eng Manuf 2017;231:1091–101. https://doi.org/10.1177/0954405415583884.
[46] Sakoe H, Chiba S. Dynamic programming algorithm optimization for spoken word recognition. IEEE Trans Acoust 1978;26:43–9.