Automated Uncertainty Quantification through Information Fusion in Manufacturing processes

Saideep Nannapaneni[1], Sankaran Mahadevan[11], Abhishek Dubey[2]
David Lechevalier[3], Anantha Narayanan[4], Sudarshan Rachuri[5]

[1]Department of Civil & Environmental Engineering, Vanderbilt University, Nashville, TN 37235, USA
[2]Department of Electrical Engineering & Computer Science, Vanderbilt University, Nashville, TN, 37235, USA
[3]Le2i, Université de Bourgogne, BP 47870, 21078 Dijon, France
[4]Department of Mechanical Engineering, University of Maryland, College Park, MD, 20742, USA
[5]Advanced Manufacturing Office, U.S Department of Energy, Washington D.C, 20585, USA

ABSTRACT

Evaluation of key performance indicators (KPIs) such as energy consumption is essential for decision-making during the design and operation of smart manufacturing systems. The measurements of KPIs are strongly affected by several uncertainty sources such as input material uncertainty, the inherent variability in the manufacturing process, model uncertainty and the uncertainty in the sensor measurements of operational data. A comprehensive understanding of the uncertainty sources and their effect on the KPIs is required to make the manufacturing processes more efficient.

Towards this objective, this paper proposes an automated methodology to generate a Hierarchical Bayesian network (HBN) for a manufacturing system from semantic system models, physics-based models and available data in an automated manner, which can be used to perform uncertainty quantification (UQ) analysis. The semantic system model, which is a high-level model describing the system along with its parameters is assumed to be available in the Generic Modeling Environment (GME) platform. Apart from semantic description, physics-based models, if available, are assumed to be available

[1] Corresponding author. Tel.: +1-615-322-3040
Email: sankaran.mahadevan@vanderbilt.edu

in model libraries. The proposed methodology is divided into two tasks – (1) Automated Hierarchical Bayesian network construction using semantic system model, available models and data, and (2) Automated uncertainty quantification (UQ) analysis. A metamodel of a HBN is developed using the GME, along with a syntax representation for the associated conditional probability tables/distributions. The constructed HBN corresponding to a system is represented as an instance model of the HBN metamodel. On the metamodel, a model interpreter is written to be able to carry out the UQ analysis in an automated manner for any HBN instance model conforming to the HBN metamodel. The proposed methodologies are demonstrated using an injection molding process.

Keywords: Bayesian network; meta-model; GME; uncertainty; Hierarchical; automation; semantic; injection molding;

Nomenclature

BN	Bayesian network
HBN	Hierarchical Bayesian network
UQ	Uncertainty quantification
GME	Generic Modeling Environment
BIC	Bayesian Information Criterion
UML	Unified Modeling Language
ABC	Approximate Bayesian Computation
MCMC	Markov Chain Monte Carlo
CPD	Conditional Probability Distribution

1. Introduction

Uncertainty quantification (UQ) involves the estimation of uncertainty in the output of a system or a model (such as KPIs), and it requires the aggregation of errors and uncertainty arising from several stages and processes of a manufacturing network. Bayesian networks (BNs) [1] have been used to perform UQ through aggregation of several uncertainty sources and information, and are being used in several domains such as information retrieval, data fusion and decision-making [2], safety assessment of software-based systems [3], computational biology and bioinformatics [4, 5], epidemiology [6], and civil infrastructure networks [7]. A BN allows the aggregation of multiple uncertainty sources that combine in different ways [8, 9] and offers a systematic approach for uncertainty integration and management by fusing heterogeneous information from multiple sources, and is therefore pursued in this paper, especially with semantic system models.

Uncertainty quantification in manufacturing process performance prediction has been previously attempted using fuzzy set theory [10-12], Monte Carlo simulations [13]. Our previous work [14] has developed a Bayesian network framework for uncertainty aggregation in manufacturing processes. The benefit of using a Bayesian network for uncertainty quantification is that it enables both forward uncertainty propagation and model calibration through Bayesian Inference. Bayesian networks are inherently capable of modeling dependence between several processes of a manufacturing network through conditional probability distributions and can handle both discrete as well as continuous variables. Fuzzy sets cannot handle both discrete and continuous variables together. In addition, inference in fuzzy sets is generally rule-based, where rules are set by the user whereas inference in Bayesian networks using Bayes theorem is mathematically well developed.

In this paper, we model the manufacturing systems as multi-level systems, where several processes occur at different levels of the hierarchy. A simple hierarchical classification of manufacturing systems

can be at unit process level, line level and factory level. A factory can be divided into several lines and each line can be further divided into several unit-level individual processes. For analysis of such multi-level system, we use the framework of Hierarchical Bayesian networks, which are extensions of Bayesian networks for multi-level systems. One advantage of using a BN is that it encodes the expert domain knowledge into the model. Experts, in general, may have a wealth of knowledge on a single unit process (such as die-casting, threading) that they have experience with and therefore can support in building a BN associated with that unit process. A HBN approach allows the integration of expert knowledge from multiple domains in modeling the manufacturing process. HBNs also enable scalability in modeling manufacturing processes. As the complexity of the manufacturing process increases, the number of lines and unit processes also increase. HBNs allow modeling up to different resolutions such as up to line level, or unit process level, or up to lower level unit processes depending upon the analysis requirements. As a result, HBNs provide better visualization of the models. More details regarding the benefits of using HBNs for modeling uncertainty in manufacturing systems are presented in our earlier work [15].

The focus of this paper is the development of an automated tool for performing UQ analysis of manufacturing processes, since such a tool will greatly benefit the industry practitioners to incorporate uncertainty in the evaluation of KPIs without having to rely on experts in data science and statistics. We have two types of models in manufacturing – (1) descriptive models, i.e. models that qualitatively describe the manufacturing processes such as semantic system models, and (2) analytics models, which are used to quantify the manufacturing process performance. Lechevalier et al., [16] propose the idea of using domain-specific modeling languages and tools to bridge the gap between the modeling and analytics procedures in the manufacturing domain. The key idea is to obtain analytical models from the manufacturing system models, which can be provided by the practitioners.

The Generic Modeling Environment (GME) [17] is a tool for creating high-level descriptive models of objects in various application domains. A domain is specified in GME by constructing a unique meta-model, which describes the various objects, properties, and relationships in that domain. The tool can then be used to build models of real-world objects in that domain.

Similar to the development of meta-models for describing manufacturing systems, meta-models can also be developed for representing analytical models. It should be noted that there are two types of meta-models, one to describe the manufacturing system and the other to represent an analytical model. The models that are constructed conforming to the meta-models are known as instance models. The industry practitioners can provide a descriptive model of the manufacturing system using the first meta-model and the analytical model corresponding to the manufacturing system under consideration is derived using its meta-model and the instance model of the manufacturing system.

Related work using domain-specific models for analytics is by Nannapaneni et al [18], in which domain-specific models are used to extract the reliability block diagram using the concepts of functional decomposition and function-component association to carry out reliability analysis. Lechevalier et al [19] extracted a neural network model for automated analysis in the manufacturing domain by developing a neural network meta-model. Aguila and Sagrado [20] developed a Bayesian network metamodel, which they referred to as BayNet, which has different modules for representation of the Bayesian network structure (BayNet structure) and inference (BayNet Reasoning). BayNet allows for modeling of discrete variables only and this paper seeks to develop a generalized methodology to handle discrete, continuous and functional nodes (i.e., nodes that are functions of other nodes, instead of being statistically correlated). This paper proposes a method to generate a BN from instance models, physics-based models and available data on the system automatically. In our earlier work [21], we developed an automated framework for

extracting a Bayesian network using a system model and data; in this work, we extend our previous work to Hierarchical Bayesian networks.

The overall objective of this paper is to assimilate semantic system models, physics-based models, and data (both historical and online) to automatically construct a BN for UQ analysis. While the idea of a BN for UQ analysis is not new, our first contribution is in developing a hybrid approach that fuses heterogeneous sources of information and combines both physics-based and data-driven approaches. Our second contribution is in automating the BN construction through GME. Our third contribution is building the BN in a hierarchical manner from component-level to system-level, this allowing analytics and decision-making at multiple levels of the system hierarchy.

The remainder of this paper is organized as follows. Section 2 introduces BNs, techniques for BN construction, BN learning algorithms, HBN and meta-modeling. Section 3 describes the proposed methodology and algorithms for HBN construction and UQ analysis. An injection molding example is used to demonstrate the proposed methodologies in Section 4 followed by conclusions in Section 5.

2. Background

In this section, we review Bayesian networks, Bayesian network construction techniques, hierarchical Bayesian networks, and modeling of complex systems.

2.1. BAYESIAN NETWORKS

Definition: A Bayesian network is a probabilistic and acyclic graphical model consisting of nodes and directed arcs, where the nodes represent the variables in the system and arcs represent the dependence relationships between variables quantified through conditional probability distributions. Mathematically, a Bayesian network represents the joint probability distribution of $\boldsymbol{X} = (X_1, X_2 \dots X_n)$ as

$$Pr_B(\boldsymbol{X}) = \prod_{i=1}^{n} Pr_B(X_i|\Pi_{X_i}) \quad (1)$$

where Π_{X_i} represents the set of parent nodes of X_i and $Pr_B(X_i|\Pi_{X_i})$ represents the conditional probability distribution of X_i, given its parent nodes. If X_i has no parent nodes, then $Pr_B(X_i|\Pi_{X_i})$ represents the marginal probability distribution of X_i.

Techniques for construction: The techniques for constructing a BN can be broadly classified into three types: (1) physics-based, (2) data-driven, and (3) hybrid approaches. The physics-based approach relies on mathematical models that represent the relationships between the system variables. The data-driven approach uses the available data to learn the BN structure and conditional probabilities using learning algorithms discussed later in this section. In some cases, mathematical models might be available for some segments of the system and data is available for other segments. In such a scenario, a hybrid approach is taken, where physics-based equations are used to model some dependence relations whereas the remaining relationships are learned from the available data. The BN is constructed in two stages – (1) a partial BN is obtained using the available physics-based models, and (2) the BN constructed in step 1 is used as a prior for learning the remaining dependence relations using the available data.

Learning algorithms: The goal of the learning algorithms is to identify a BN that best describes the available data. The learning process involves two tasks: structure learning and parameter learning. Structure learning involves finding a graphical structure that best represents the dependence between nodes based on available data. Parameter learning involves quantification of dependence among the nodes by estimating the parameters of the conditional probability distributions/tables [22].

The structure learning algorithms can be broadly divided into three categories: (1) constraint-based, (2) score-based, and (3) hybrid. Constraint-based methods employ conditional independence tests to learn an optimal BN. A commonly used conditional independence test is the mutual information test. The expressions for mutual information ($I_{X,Y}$) in the case of discrete variables and continuous variables are given in Equations (2) and (3), respectively, as

$$I_{X,Y} = \sum_{Y}\sum_{X} p(x,y)\ log\left(\frac{p(x,y)}{p(x)p(y)}\right) \tag{2}$$

$$I_{X,Y} = \int_{Y}\int_{X} p(x,y)\ log\left(\frac{p(x,y)}{p(x)p(y)}\right) \tag{3}$$

where $p(x,y)$ represents the joint probability distribution of X and Y, and $p(x)$ and $p(y)$ represent the marginal distributions of X and Y, respectively. Some other conditional independence tests that are used include linear correlation and conditional correlation for continuous variables, and G-test and Chi-square test [23] for discrete variables.

In score-based learning, every BN structure is assigned a network score based on the goodness-of-fit for available data and a set of heuristic optimization techniques are used to obtain the BN that optimizes the defined score. Some commonly used metrics that are used for scoring Bayesian networks include Log-likelihood, Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), Minimum Description Length (MDL) and Bayesian Dirichlet Equivalence (BDe) [22]. In this paper, we use the BIC score, defined as

$$BIC = k \times \ln(n) - 2 \times \ln(L) \tag{4}$$

In Equation (4), L, k and n represent the likelihood the observing available data given a BN, number of free parameters that are estimated and the number of available data samples respectively.

Hybrid algorithms employ both conditional independence tests and network scores for learning the BN structure. The conditional independence tests are first used to reduce the space of possible BN

structures and score-based methods are then used to obtain the optimal BN structure among them. Parameter learning algorithms estimate the parameters of the conditional probability distributions from available data using the maximum-likelihood approach.

2.2. HIERARCHICAL BAYESIAN NETWORKS

As mentioned in Section 1, a HBN can be considered as an extension of Bayesian networks for modeling multi-level systems. A HBN is a BN where a node can represent another lower-level BN. Any number of levels are possible. A node in a lower-level BN can further represent a much lower-level BN.

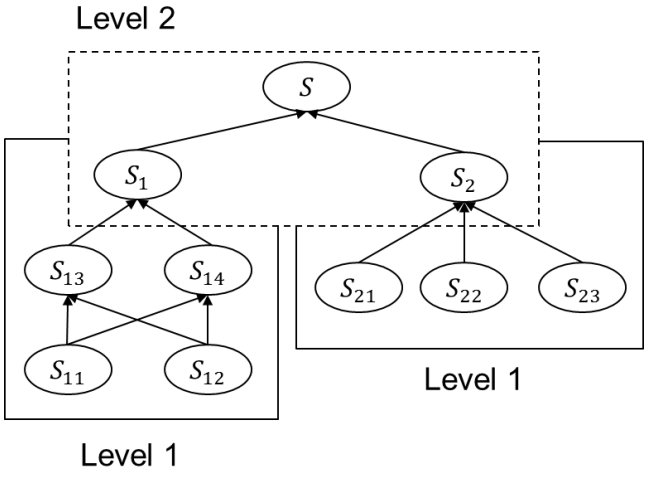


Fig. 1. A simple Hierarchical Bayesian Network

In Fig. 1, the HBN consists of two levels – Level 1 and Level 2. The root variables in the Level 2 BN (S_1,S_2) are associated with their lower-level BNs. The results from the analysis of lower-level BNs are then propagated to the next higher-level for further analysis.

2.3. MODELING COMPLEX SYSTEMS

Modeling tools have become increasingly essential and useful for the design and analysis of complex systems. Usage of such tools require the concepts of a process and a paradigm. The modeling process conforms to a set of rules that minimizes errors and facilitates the presentation and communication of

models. The modeling paradigm or the modeling language, such as GME, contains all syntactic, semantic and presentation information regarding a domain, and represents the rules that govern the construction of models. In recent years, the notion of meta-modeling has been added to process and paradigm. The outcome of the meta-modeling task is a model, called a meta-model that encodes all the concepts and rules of the modeling paradigm. GME offers a meta-modeling language called MetaGME, which is based on Unified Modeling Language (UML) class diagrams [24], to create domain-specific meta-models. The meta-models described in this paper were built using MetaGME. Note that the meta-modeling described here is different from surrogate modeling or reduced-order modeling (creation of inexpensive models to replace expensive physics models) which is also referred to as meta-modeling in the literature.

3. Methodology

The proposed methodology for UQ analysis of manufacturing processes can be divided into two steps 1) Automated HBN construction using available models and data, and 2) UQ analysis using the constructed HBN. We first describe the BN construction techniques using either models or data, and then extend these techniques for the construction of an HBN in the presence of a combination of models and data.

3.1. AUTOMATED BN CONSTRUCTION USING AVAILABLE MODELS AND DATA

As stated in Section 2, a BN can be constructed using mathematical models or data or a combination of both. Two cases are considered here: one using physics-based models and one using data. It is straightforward to construct a BN manually when models are available. This paper, however, focuses on the automated generation of a BN.

The variables required for construction of a BN can be obtained from the manufacturing system description. We incorporate this description into a domain-specific model in GME, which, as noted, is as an instance of the corresponding meta-model developed using MetaGME. The details of constructing a generic meta-model for manufacturing systems are not discussed here. This work considered only meta-modeling of BN and HBN, and not manufacturing systems. Refer to [25] for discussion regarding meta-modeling of manufacturing systems. The system variables in the descriptive system model are used as a basis for identifying the nodes and their preliminary ordering for the BN that will be generated. Data associated with the system variables is then used to obtain the BN representing the system.

3.1.1. Automated BN construction using physics-based models

Physics-based models are assumed available as equations in a text (.txt) file. The models can be present in any random order. The algorithm presented below will order the equations and build a BN from them.

1) Create two lists x_L and x_R to store the variables to the left and right of the equality sign.
2) Create a dictionary object D with the left hand side (LHS) variables of an equation as the key and the list containing all the right hand side (RHS) variables of that equation as the value.
3) Since a BN is a layered structure, the variables in the top layer, also called root variables, are given by $x_R - (x_R \cap x_L)$.
4) The second layer comprises all variables that can be defined by a subset of the top-level variables. This can be achieved by selecting the keys whose values are a subset of the first layer variables.
5) Similarly, every other layer consists of variables that can be defined by the variables in the above layers. The procedure specified in step 4, i.e., looking into the dictionary D, is used to select all the variables in the current layer.
6) Step 5 is repeated several times until all the variables in the system are defined.

It should be noted that these physics-based models could be either deterministic models or stochastic models. In the numerical example, stochastic models with response following a Gaussian distribution with a known constant variance are implemented. For illustration, consider a variable y dependent on independent variables $x1$ and $x2$. If the dependence is stochastic with a mean equal to $2 \times x1 + x2$ and standard deviation of 2, then the dependence in a .txt file is represented as $y = N(2 \times x1 + x2, 2)$. The variables associated with deterministic relationships can be defined as functional nodes whereas the variables with stochastic relationships can be defined as stochastic nodes.

3.1.2. Automated HBN construction using physics-based models

For the construction of an HBN, we assume the availability of multiple .txt files corresponding to multiple BNs at several levels of the manufacturing network hierarchy. In addition, we consider a possibility that a same physical parameter (such as density) may be denoted using different names (such as d, rho) in different .txt files. This might be possible when the models are obtained from several different libraries or experts. Hence, we also have another .txt file that clarifies the nomenclature for variables across multiple .txt files. A sample line from the nomenclature .txt file is $\rho = d, rho, D$. With the available information, we implement the following procedure for building an HBN.

(1) Create another dictionary $D1$ with a key to identify a given .txt file and a value, which is a list of three lists. These three lists contain the root variables (nodes with no parent nodes), intermediate-level variables (nodes with both parent and child nodes) and end variables (nodes with no child nodes) in a given .txt file.

(2) Make sure all .txt files with mathematical models follow the same notation for variables as in the nomenclature .txt file.

(3) Identify the key k of the .txt file for which the end variable(s) are not a root variable(s) in other .txt files. The BN associated with the .txt file with key k represents the top-level BN

(4) Construct the BN associated with .txt file of key k using the algorithm presented in Section 3.1.1. If k is not unique, then all the BNs associated with .txt files with keys in k are constructed.

(5) We then build the BNs from the .txt files whose end variables are the root variables of the last constructed BN. This step is repeated until all the BNs are constructed from their .txt files.

(6) If there exists a duplication of variables, such that a parameter exists in BNs from multiple levels or several BNs at the same level, the duplication is removed by retaining the variable in the lower-level BN and removing the variable from the higher-level BN. All dependences associated with the removed variable are added to the same variable in the lower-level BN. Thus, an HBN can be constructed from physics-based models.

3.1.3. Automated BN construction using available data

When physics-based models are not available, we propose to use the system model and the data associated with the process variables in the system model to construct the BN. The benefit of using a system model is that it provides qualitative information regarding the ordering of variables (i.e., BN topology). Incorporation of such prior system model information makes the BN learning accurate and efficient. We use the scored-based learning method called Hill-Climbing [22] and the BIC criterion as a scoring metric for illustration. A Gaussian conditional probability distribution (CPD) with a constant but unknown variance is fit for each continuous node, and its mean is a linear combination of the values taken by the parent nodes. (This approximation is referred to as a linear Gaussian CPD in the literature [22]).

The constructed BN can later be validated using model validation techniques [26] such as model-reliability metric, area-metric etc. The available dataset can be divided into training and test data; the BN can be constructed with the training set and validated with the test set.

3.1.4. Automated HBN construction using available data

The HBN for a manufacturing network can be constructed in three steps [15]: (1) BNs for individual processes at multiple levels using available physics models and/or data as described in Section 3.1.1 and Section 3.1.2, (2) a preliminary HBN following the manufacturing network topology, and (3) Learning the dependences across multiple levels or across multiple BNs at the same level by using the preliminary HBN constructed in the second step as a prior. The knowledge regarding the system model is used in obtaining a preliminary HBN. In the third step, we do not allow any new dependence between variables in a BN within a particular level as we assume any such dependence should have been captured in the first step.

3.2. UNCERTAINTY QUANTIFICATION USING THE BAYESIAN NETWORK

Uncertainty quantification analysis using a BN (or HBN) can be divided into two tasks: 1) Model calibration using Bayesian Inference, where the unknown model parameters are estimated using any observation data, and 2) Forward uncertainty propagation, where the posterior distribution of the output quantity of interest is constructed using posterior distributions of estimated model parameters. The procedure for performing automated UQ analysis using a BN can be divided into three steps. The first step is to transform the BN (constructed using Section 3.1) into an instance model in GME. The second step is to implement the above UQ methodology for the BN meta-model on the GME platform. The algorithms written for a BN meta-model can be used for all instance models associated with the BN meta-

model. The third step is to run the UQ methodology on a BN instance model. Using the instance model of the BN, the UQ analysis methodology, and any new data, uncertainty quantification can be carried out for any BN (or HBN) constructed in Section 3.1, in an automated manner. In our earlier work [21], we have created a BN meta-model and have written algorithms for UQ on the BN meta-model. In this paper, we extend the previous work to create a meta-model for an HBN and write UQ algorithms for it. We first describe the BN meta-model and the syntax representation of the associated conditional probability distributions and then move onto the HBN.

3.2.1 Bayesian network meta-model

Fig. 2 shows the BN meta-model created using the MetaGME within the GME modeling paradigm. In the meta-model, 'BayesianNetwork' represents the root component, and 'Node' represents any node in the Bayesian network. There can be three types of nodes in the Bayesian network: 'DiscreteNode', 'ContinuousNode', and 'FunctionalNode'. A discrete node represents a node with a finite possible number of states. Similarly, a continuous node represents a variable that is continuous in nature over a defined range. A functional node represents a variable which can be known deterministically when the values of its parent nodes are known.

The three possible types of nodes (Discrete, Continuous and Functional) are represented by the 'Node' component through an inheritance entity (denoted by the triangle icon). The 'Node' component is modeled as an abstract component since any node in a BN is either continuous or discrete or functional. In a BN, a node is connected to another node forming an edge; this is represented in the meta-model by the 'Edge' component using the 'src' (source) and 'dst' (destination) tags at the 'Node' component.

The next step after creating the components is to define the attributes to be associated with each of them. The 'BayesianNetwork' component is associated with two attributes – 'Filelocation' and

'Information'. The 'Information' attribute is of 'enumeration' type and can take one of the values – 'Models' and 'Data'. The location of the file that contains either the models or data is provided in the 'Filelocation' attribute. The attributes that are common to all three types of nodes such as 'Name', 'Parents', and 'Postprocessing' are associated with the 'Node' component. Since the three types of nodes are connected to the 'Node' component via the inheritance property, the attributes associated with 'Node' also apply to the three types of nodes. The parent nodes associated with a node are provided in 'Parents' attribute. 'Postprocessing' is a Boolean variable, that specifies whether the variable requires post processing analysis. Apart from the common attributes, each type of node has a specified set of attributes.

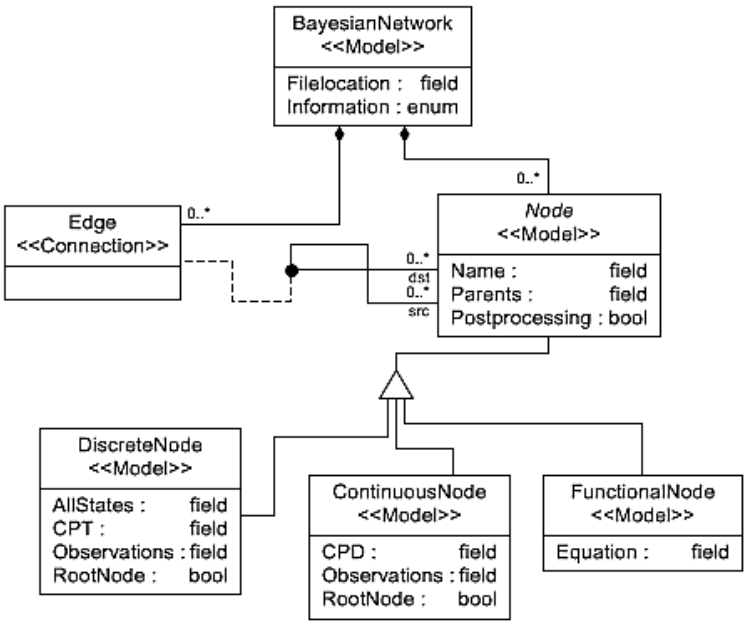


Fig. 2. Meta-model for the Bayesian network

Additional attributes for a discrete node include 'RootNode', 'CPT', 'AllStates' and 'Observations'. A node with no incoming edges (i.e., with no parent nodes) is called a 'root node'. The Boolean attribute 'RootNode' is provided to identify root nodes. Only continuous and discrete nodes in a BN can be root nodes. All the possible finite states of the discrete variables are provided in the 'AllStates' attribute. Any new observational data is provided with the 'Observations' attribute. The conditional probability table for

the discrete variable or marginal probability table (for root nodes) is defined in the 'CPT' attribute. For illustration, Table I defines a discrete parent node 'A' with three possible states 'A_1','A_2','A_3' and marginal probabilities of 0.1,0.6,0.3 respectively. Other attributes such as 'Observations' and 'Postprocessing' are not mentioned below because the goal here is to demonstrate the definition of a CPT.

TABLE I**.** Representation of a root discrete node

Attribute	**Value**
Name	A
RootNode	True
Parents	
AllStates	A_1, A_2, A_3
CPT	0.1,0.6,0.3

When defining the marginal probabilities, the order of probabilities should be the same as the order of states defined in 'AllStates' attribute. Since 'A' is a root node, it has no associated parent nodes; therefore, the value corresponding to Parents in Table I is empty. Next, consider a discrete node with discrete parents. Let A and B be the two parent nodes each with two states, A = {A_1, A_2} and B = {B_1, B_2}. Let C represent the child node with two states, C = {C_1, C_2}. The conditional probability table is given in Table II.

TABLE II. CPT of a discrete node with discrete parents

C | A, B	**A = A_1, B = B_1**	**A = A_1, B = B_2**	**A = A_2, B = B_1**	**A = A_2, B = B_2**
C = C_1	0.6	0.7	0.2	0.4
C = C_2	0.4	0.3	0.8	0.6

The case when the discrete child node has continuous parent nodes or a combination of continuous and discrete parent nodes is discussed below. The key ideas in dealing with continuous parent nodes involve discretizing their ranges and defining a conditional probability for the child node in each of the ranges. Let A, B represent a discrete and a continuous parent node of a discrete child node C. Assume A has two states, A = {A_1, A_2} and B follows a uniform distribution between 10 and 20. Let the range of B be divided

into two uniform intervals; therefore, B can be considered as a discrete variable. The corresponding conditional probability table is given as shown in Table III. In Table III, the squared brackets also include the equality whereas the parentheses do not. If B = [10, 15], then 10 <= B <= 15 whereas B = (15, 20] represents 15 < B <= 20. The representation of C is shown in Table IV.

TABLE III. CPT of a discrete node with discrete and continuous parents

C | A, B	**A = A_1, B = [10,15]**	**A = A_1, B = (15,20]**	**A = A_2, B = [10,15]**	**A = A_2, B = (15,20]**
C = C_1	0.6	0.7	0.2	0.4
C = C_2	0.4	0.3	0.8	0.6

TABLE IV. Representation of a child discrete node

Attribute	**Both parents are discrete**	**One discrete and one continuous**
Name	C	C
Root Node	False	False
Parents	A,B	A,B
AllStates	**C_1,C_2**	**C_1, C_2**
CPT	A_1,B_1 : 0.6,0.4; A_1,B_2 : 0.7,0.3;A_2,B_1 : 0.2,0.8; A_2,B_2 : 0.4,0.6	A_1, [10,15] : 0.6,0.4; A_1, (15,20] : 0.7,0.3; A_2, [10,15] : 0.2,0.8; A_2, (15,20] : 0.4,0.6

Consider the case when B is represented using a Normal distribution and divided into two disjoint intervals B <= 15 and B > 15, represented as (..15] and (15..) respectively. The same representation can be extended to the case when all the parent nodes are continuous. Each continuous node is discretized and treated as a discrete variable; a similar procedure can be followed for the case of a discrete node with all continuous parents.

The attributes for the continuous node include ‘RootNode’, ‘CPD’ and ‘Observations’. The definitions for ‘RootNode’ and ‘Observations’ are identical to their discrete counterparts. ’CPD’ represents the conditional probability distribution or marginal probability distribution (for root nodes). For illustration,

consider a normally distributed variable ‘A’ with parameters (mean, standard deviation) 10 and 1. Attributes such as ‘Postprocessing’ and ‘Observations’ are not mentioned below. Representation of ‘A’ is given in Table V.

TABLE V. Representation of a root continuous node

Attribute	Value
Name	A
RootNode	True
Parents	
CPD	Normal(10,1)

Next, different combinations of parent nodes, A and B, for a continuous child node C are considered and the corresponding representations are given in Table VI.

TABLE VI. Representation of a child continuous node

Attribute	Both parents are discrete	One discrete and one continuous	Both parents are continuous
Name	C	C	C
RootNode	False	False	False
Parents	A(A_1,A_2), B(B_1,B_2)	A (A_1,A_2), B(continuous)	A,B (Both continuous)
CPD	A_1,B_1 : Normal(5,1); A_1,B_2 : Uniform(10,14); A_2,B_1 : Normal(10,2); A_2,B_2 : Uniform(12,17)	A_1:Normal(2*B,1);A_2 : Uniform(B-2, B+2)	Normal(A+2*B, 1)

After discrete and continuous nodes, functional nodes are considered. As stated earlier, functional nodes are deterministically known when conditioned on all the parent nodes, either discrete or continuous. Functional nodes have only one additional attribute called ‘Equation’. The expression connecting the parent nodes to the child node is given here. If A and B represent the continuous parent nodes for a functional node C, and if C = A + 2*B, then the variable is represented as shown in Table VII.

TABLE VII. Representation of a functional node with continuous parents

Attribute	Value
Name	C
Parents	A,B
Equation	A + 2*B

When a child node has a functional node as a parent node, the functional node can be treated in the same manner as a continuous node for CPD/CPT representation. If a functional node has finite states, it can be modeled as a discrete node where one particular state has a conditional probability of 1 and all the other states have zero as their conditional probabilities. A functional node may also be modeled as a continuous node with zero variance. However, some BN packages do not allow modeling a variable with zero variance. In such cases, a functional node can be defined as continuous node with a very small variance. To avoid confusion, we have chosen to model functional nodes explicitly.

3.2.2 Hierarchical Bayesian Network meta-model

We now extend the meta-model of a Bayesian network described in Section 3.2.1 to describe a Hierarchical Bayesian network. Fig. 3 describes the meta-model of a Hierarchical Bayesian network. Similar to a BN metamodel, 'HBN' defines the root component with attributes of 'Filelocation' and 'Information'. 'Node', which could be a 'DiscreteNode', 'ContinuousNode', or a 'FunctionalNode' can represent a node at any level in the HBN. The dependence between nodes is represented using 'DependenceEdge' as opposed to just 'Edge' in the BN metamodel. A major difference in a HBN compared to a BN is that a node can contain other lower-level BNs. This is represented using the 'containment' connection, which is a solid line with a diamond shape end, originating from 'Node' and ending in itself ('Node'). To ensure dependence connections within a lower-level BN, 'DependenceEdge' is also contained in 'Node'. In some cases of a HBN, it might be possible that a variable can influence a

lower-level BN and also a higher-level BN. An example of such a case is the shrinkage variable affecting the lower-level BN of a melting process and the higher-level BN of total energy consumption as shown in Fig. 5. To denote such dependence across multiple levels, we will have two different nodes representing the same variable and connect both of them using an 'EqualityEdge'. A 'NodeRef' entity is of a 'Reference' type, which is used as a stand-in for a variable in a lower-level BN. For example, consider the shrinkage variable. 'Shrinkage_total' is a node representing shrinkage in the higher-level BN, which computes total energy. 'Shrinkage_melt' is also a node representing shrinkage in the lower-level BN, which corresponds to energy of melting process. To make sure they represent the same quantity, we create a reference object by the name 'Shrinkage_melt' and connect the reference object with 'Shrinkage_total' through an 'EqualityEdge'. To be able to create reference objects and equality edges in the lower-level BNs, we contain 'EqualityEdge' and 'NodeRef' into 'Node'.

Regarding the attributes, the 'Filelocation' attribute in the physics-based HBN construction takes a set of .txt files and a nomenclature .txt file. These files are input to the model separated by ','. It is also assumed that the last .txt file represents the nomenclature file.

3.2.3. Algorithms for automated UQ analysis

After creating the meta-model for an HBN, two algorithms are implemented to create instance models from available data or mathematical models, and to perform model calibration and uncertainty propagation analyses.

The first algorithm constructs an HBN using available physics-based models or data. If physics-based models are available, the procedure in Section 3.1.3 is used to construct the HBN. If data are available, the HBN is learnt using the score-based hill-climbing algorithm available as part of the *bnlearn* package in R; the procedure was laid down in Section 3.1.4. The constructed BN is stored as a JSON representation. The JSON file is then automatically converted into a GME instance model. The user can make any changes

to the HBN instance model (such as adding observations or selecting variables for post processing, deleting some nodes), directly using the GME interface. When all the required changes are made to the HBN instance model, the second algorithm is executed for UQ analysis. All the algorithms on the HBN meta-model are implemented in C#.

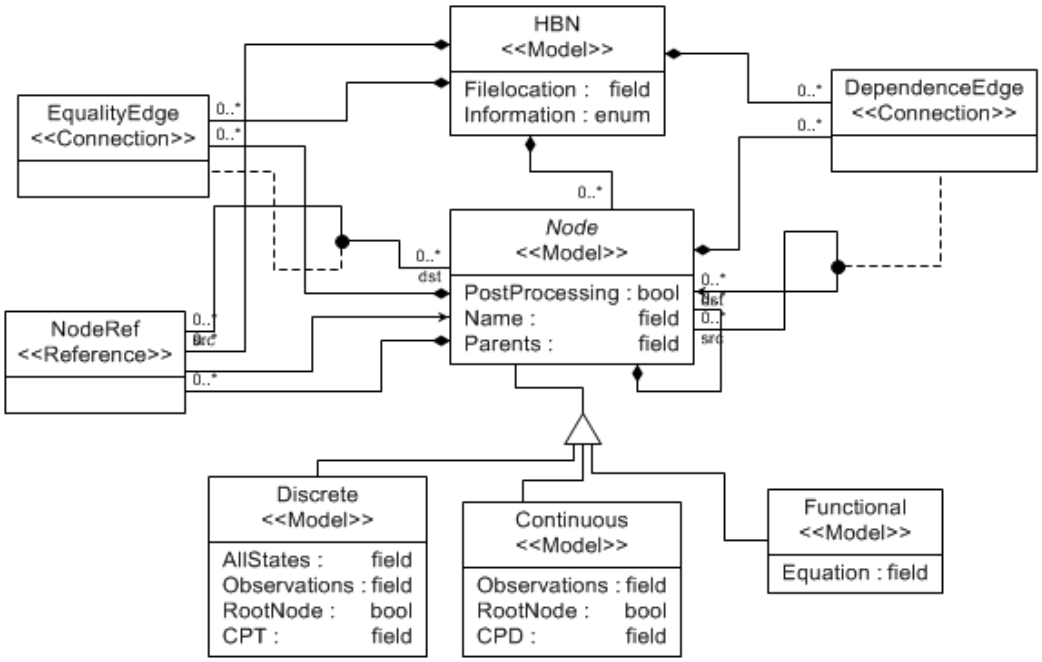


Fig. 3 Meta-model of a Hierarchical Bayesian network

Traditional methods for model calibration and UQ analysis such as Markov Chain Monte Carlo (MCMC) for large BNs or HBNs can become computationally expensive. Several approximate methods are available to replace the MCMC methods such as Approximate Bayesian Computation (ABC) [27], Bootstrap Filter [28] and Variational Bayes [29]. In this paper, we use the Approximate Bayesian Computation (ABC) technique for calibration and UQ analysis.

The ABC algorithm generates several samples of the calibration parameters from their prior distributions, and for each sample of the calibration parameters, several samples of the observation variable (i.e., the variable regarding which data is available) are generated. The generated data for a given

calibration parameter sample is then compared against the observation data; this comparison is typically done through a pre-defined distance measure. If the distance between the generated and observation data is low, then the probability of that sample to be in the posterior distribution is high; this probability, in this paper, is calculated using the likelihood function as described below. This process is repeated for several samples of calibration parameters and likelihood values are calculated for each of those samples. The posterior distributions are then obtained by sampling the prior samples according to their likelihood values.

Let $\boldsymbol{\theta}$ represent a vector of calibration parameters and $\boldsymbol{\theta}_i, i = 1,2,3 \dots k$ represent k samples of $\boldsymbol{\theta}$ obtained from their prior distributions. Let S represent the variable on which observations are available given by $S_j^{obs}, j = 1,2,3 \dots m$. As linear Gaussian CPDs are fit as part of the HBN learning process (Section 3.1.3), let σ_S represent the standard deviation associated with the linear Gaussian CPD of S. Since a HBN is a probabilistic model, the prediction of S_i for a given realization of $\boldsymbol{\theta}_i$ is not a single value but a PDF. Let S_i^m represent the expected value of S_i. Given this information, the likelihood of observing the data ($S_j^{obs}, j = 1,2,3 \dots m$) for a sample $\boldsymbol{\theta}_i$ can be computed as $L_i = \prod_{j=1}^{m} \Pr(S_j^{obs} | S_i^m, \sigma_S)$. This likelihood value is computed for all values of $\boldsymbol{\theta}_i, i = 1,2,3 \dots k$. Given all the likelihood values, the weights for each $\boldsymbol{\theta}_i, i = 1,2,3 \dots k$ can be computed by normalizing the likelihood values, given as $w_i = \frac{L_i}{\sum_{i=1}^{k} L_i}$. The prior samples ($\boldsymbol{\theta}_i, i = 1,2,3 \dots k$) and their updated weights ($w_i, i = 1,2,3 \dots k$ are used to construct their posterior distributions.

The HBN instance model is read and a JSON file is created, which represents the data corresponding to all nodes in a BN. The JSON file is read into R and used for calibration analysis. Since data might be available at multiple levels in an HBN, a multi-level calibration approach, as detailed in [15], is used for calibration analysis. For illustration, assume data is available at two levels of an HBN. The lower-level data is first used to obtain a posterior distribution and this posterior is used as a prior for calibration with

the higher-level data. In the current implementation, we have considered only continuous and functional nodes; discrete variables will be considered in future work.

Summary of the proposed methodology: In this work, we created a meta-model for the representation of a hierarchical Bayesian network (HBN) using the Generic Modeling Environment (GME). In addition, algorithms for learning and UQ analysis of an HBN were developed. The HBN for a manufacturing network is constructed through a combination of system model (semantic, conceptual), physics models (mathematical, numerical), and observation data (numerical), and represented as an instance model of the HBN meta-model. The user can then make any changes to the instance model (such as adding observation data); this modified instance model is used to perform model calibration and UQ analysis using the Approximate Bayesian Computation (ABC) algorithm.

4. Illustrative Example

INJECTION MOLDING PROCESS: UQ ANALYSIS USING COMBINED PHYSICS AND DATA

An injection molding process is used to demonstrate automated HBN construction and UQ analysis. The injection molding process can be considered as a combination of three sub-processes: melting of the polymer, injection into the mold, and cooling to form the part. Each sub-process is associated with a set of parameters. The goal is to estimate the uncertainty in energy consumption per part in the overall injection molding process, which is the sum of the energy consumed in the three individual sub-processes. Using either physics-based models or available data, one could construct a lower-level BN for the computation of energy consumption in each of the three individual sub-processes, and computation of the overall energy consumption forms the higher-level BN, thus forming a two-level HBN. To demonstrate BN learning, the BN for injection process is directly constructed from a physics-based model whereas the

BNs for the melting and cooling processes are learnt from synthetic data generated using physics models. Physics models associated with various stages of injection molding process are given below.

Melting process: In this stage, the polymer dye, which initially is in the solid state, is converted into the liquid state. The power consumption equation in melting the polymer is given as

$$P_{melt} = 0.5 \times \rho \times Q \times C_P \times \left(T_{inj} - T_{pol}\right) + 0.5 \times \rho \times Q \times H_f \quad (5)$$

where P_{melt} refers to power consumption in melting process, ρ, Q, C_p, H_f refer to the density, flow rate, heat capacity and heat of fusion of the polymer respectively. T_{inj} and T_{pol} represent the injection temperature and temperature of polymer respectively. If V_{part} represents the volume of a part, then the volume of a shot (V_{shot}) considering the shrinkage (ϵ), buffer (Δ), number of cavities (n) is given as

$$V_{shot} = V_{part} \times (1 + \frac{\epsilon}{100} + \frac{\Delta}{100}) \times n \quad (6)$$

Using the power consumption for melting and volume of a shot, the energy consumption for melting process (E_{melt}) is given as

$$E_{melt} = \frac{P_{melt} \times V_{shot}}{Q} \quad (7)$$

Injection process: In this stage, the molten polymer is injected into the mold. The energy consumed in the injection process (E_{inj}) is given as

$$E_{inj} = p_{inj} \times V_{part} \quad (8)$$

where p_{inj} refers to the injection pressure.

Cooling process: The molten polymer is cooled to form the final product. The energy consumption in cooling process (E_{cool}) is given as

$$E_{cool} = \frac{\rho \times V_{part} \times \left[C_p \times \left(T_{inj} - T_{ej}\right)\right]}{COP} \quad (9)$$

where T_{ej}, COP represent the ejection temperature and coefficient of performance of the cooling equipment respectively.

Overall energy consumption: Given the energy consumption for each of the three stages, the overall energy consumption of a part (E_{part}) is given as

$$E_{part} = \frac{1}{n} \times \left[\left(\frac{0.75 \times E_{melt} + E_{inj}}{\eta_{inj}} + \frac{E_{reset}}{\eta_{reset}} + \frac{E_{cool}}{\eta_{cool}} + \frac{0.25 \times E_{melt}}{\eta_{heater}} \right) \times \frac{n \times (1 + \epsilon + \Delta)}{\eta_{machine}} + P_b \times t_{cycle} \right] \quad (10)$$

where η_{inj}, η_{reset}, η_{cool}, η_{heater}, $\eta_{machine}$ refer to the efficiencies of injection, reset, cooling, heating and machine power respectively, t_{cycle} refers to the total cycle time, P_b refers to the power required for basic energy consumption when the machine is in stand-by mode, E_{reset} refers to the energy required for resetting the process and is given as

$$E_{reset} = 0.25(E_{inj} + E_{cool} + E_{melt}) \quad (11)$$

The power consumption when machine is in stand-by model (P_b) is not considered here because it depends on the type of machines used in the process. Refer [30] for more details. The Bayesian networks corresponding to the energy consumption in three stages and for the overall process, built using the physics-based models are provided in Fig. 3.

Note that the models for melting, cooling and overall energy consumption are used to generate synthetic dataset to demonstrate BN learning. They are not directly used for UQ analysis. In reality, such data comes from observations of the actual process.

Learning the HBN for the injection molding process

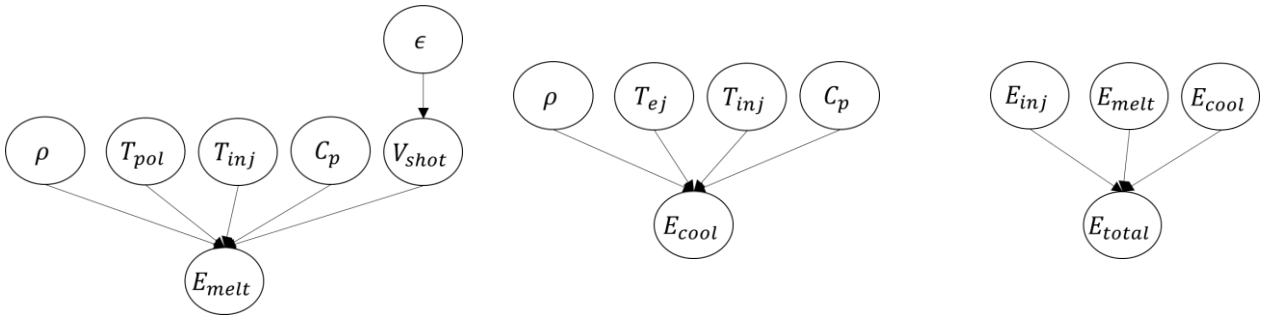
The HBN for the injection molding process is learnt using a synthetic dataset generated using physics-based models augmented with measurement uncertainty. The parameters for the synthetic dataset used for

the learning process are given in Table VIII. The sensor measurement errors of energy, volume and temperature are assumed Gaussian with zero mean and a standard deviation of 5e4, 1e-6 and 1 respectively. A dataset with 5000 samples is used for the learning process. Learning is carried out using the 'bnlearn' package in R and with 'hill-climbing' score-based algorithm.

TABLE VIII. Parameters for generating synthetic dataset used in HBN learning

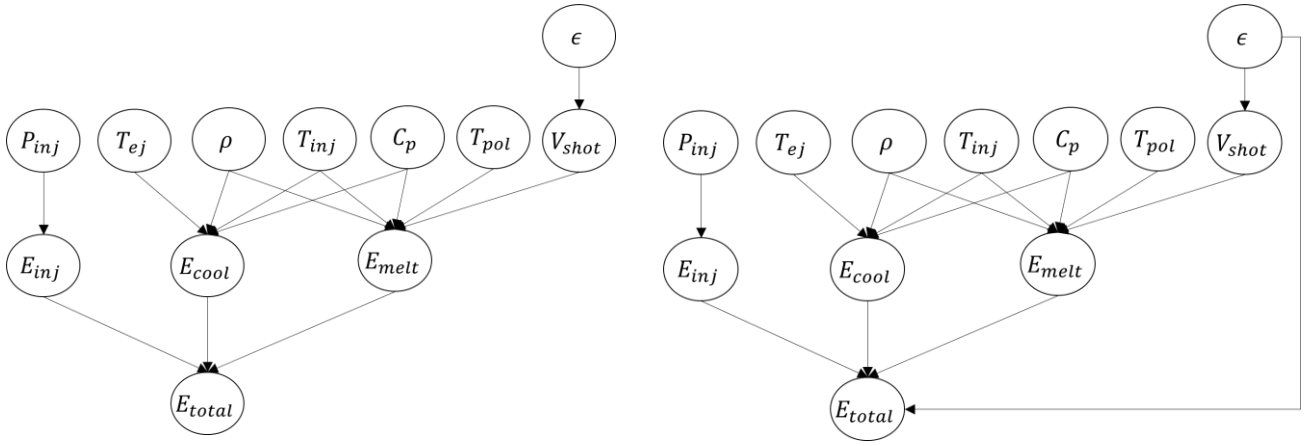
Parameter	**Value**
Shrinkage (ϵ)	Uniform(0.018,0.021)
Volume of a part (m^3)	0.002048
Buffer (Δ)	0.01
Polymer temperature (T_{pol})	Normal(50,2)
Injection temperature (T_{inj})	Uniform(205,220)
Density (ρ) (kg/m^3)	Uniform(960,990)
Heat of fusion (H_f) (kJ/kg)	240
Heat capacity (C_p) ($J/(kg\ K)$)	Uniform(2250,2290)
Pressure (Pa)	Uniform(90e6, 95e6)
Flow rate (m^3/s)	1.6e-5
Ejection temperature ((^oC)	Uniform(54,60)
All Efficiency coefficients (η, COP)	0.7

The individual BNs for the melting, cooling and overall energy consumption are learnt and the results are shown in Fig. 4. The variables for learning each of the individual BNs depend on the variables that arise in the actual physical process or on the expert opinion. The individual BNs are used to construct a preliminary HBN following the hierarchy of the Injection molding process. This preliminary HBN is then used as a prior to learn any additional dependences across multiple levels. The prior and posterior HBNs for the injection molding process are given in Fig. 5.



(a) (b) (c)

Fig. 4. Learnt individual BNs of (a) Melting, (b) Cooling and (c) Overall energy consumption



(a) (b)

Fig. 5. HBN for the Injection Molding process, (a) Prior, and (b) Posterior

From Fig. 5, an additional dependence between shrinkage (ϵ) and overall energy consumption (E_{total}) can be observed in the posterior HBN, which is not present in the prior HBN. This learnt dependence is in fact consistent with the physics equation for overall energy in Equation (12). The learnt HBN is then represented as an instance model of the HBN meta-model. The instance model is shown in Fig. 6. When the GME instance model for the above HBN is opened, the higher-level BN is seen. To see the lower-level BNs, we go inside the corresponding components by 'double-clicking' it. Thus, Fig. 6(b) can be seen by double-clicking on 'Einj' component in Fig. 6(a). Similarly, Fig. 6(c) and Fig. 6(d) can be obtained by double-clicking on 'Emelt' and 'Ecool' respectively. In the BN learning process using hill-climbing algorithm, the conditional probability distributions are assumed to be linear Gaussian distributions with

the mean defined as a linear combination of parent nodes and with a constant standard deviation. The learnt CPDs are provided in Table IX.

TABLE IX. Learnt conditional probability distributions for the Injection Molding process

CPD	Value
$E_{inj}\|P_{inj}$	$N(80541.8 + 1.175 \times 10^{-3} \times P_{inj}, 49648.7)$
$V_{shot}\|\epsilon$	$N(0.002069 + 0.0020166 \times \epsilon, 9.91 \times 10^{-7})$
$E_{melt}\|V_{shot}, \rho, T_{pol}, T_{inj}, C_p$	$N(1.6933 \times 10^5 - 5.277 \times 10^8 \times V_{shot} + 268.6 \times \rho - 2462.2 \times T_{pol} + 2447.6 \times T_{inj} + 292.23 \times C_p, 49742.85)$
$E_{cool}\|\rho, T_{inj}, C_p, T_{ej}$	$N(-2104628.2 + 1088.49 \times \rho + 6458.11 \times T_{inj} + 441.91 \times C_p - 5711.55 \times T_{ej}, 49375.03)$
$E_{part}\|E_{inj}, E_{cool}, E_{melt}, \epsilon$	$N(-71911.4 + 2.6369 \times E_{inj} + 3.34 \times 10^6 \times \epsilon + 2.617 \times E_{melt} + 2.6346 \times E_{cool}, 50456.21)$

Model Calibration

The constructed HBN can now be used for model calibration. Assume the parameters to be calibrated include injection temperature(T_{inj}), polymer temperature(T_{pol}) and polymer density(ρ). The data available for calibration include the sensor temperature measurements of T_{inj}, T_{pol} and E_{total}. Thus, data are assumed available at two levels of the HBN. The observation data is generated using the true values of the calibration parameters added with Gaussian measurement errors. For this calibration process, the shrinkage of the material, heat capacity, injection pressure and ejection temperature are assumed known at 0.0185, 2260 $J/(kgK)$, 93e6 Pa and 58 ^oC respectively. The true and assumed unknown values of the calibration parameters are 215 ^oC, 49 ^oC and 985 kg/m^3 respectively. Since data is available at multiple levels, the multi-level calibration process as described in Section 3.2 is carried out. The sensor measurement data regarding T_{inj} and T_{pol} are used to calibrate the true but unknown values of T_{inj} and T_{pol}. The posterior distributions are used as prior for calibration using higher-level data (E_{total}).

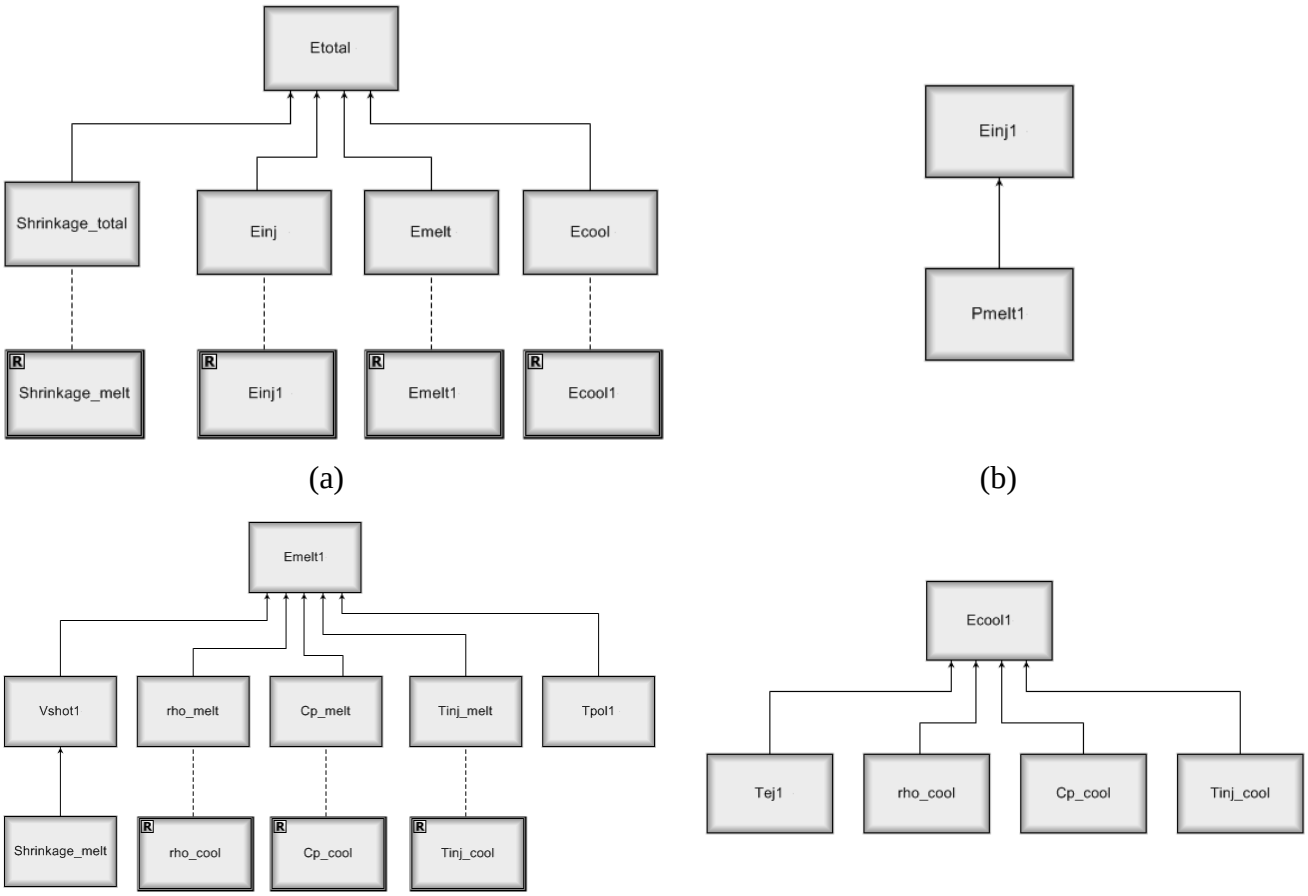


Fig. 6. Hierarchical Bayesian network of injection molding process using the HBN meta-model

Table X. Calibration parameters and their prior distributions

Calibration parameter	**Distribution**	**Range**
Injection temperature (T_{inj})	Beta(5,3)	[212,220]
Polymer temperature (T_{pol})	Beta(4,3)	[47,53]
Polymer density (ρ)	Beta(3,4)	[970.990]

As mentioned in Section 3.2.3, the Approximate Bayesian Computation (ABC) technique [26] is used for calibration. The prior distributions for the calibration parameters are given in Table X. 10,000 samples of the calibration parameters are obtained from their prior distributions for model calibration using ABC

and 100 observation data were assumed to be available; the data was generated using the true values of calibration parameters given above. The prior and posterior distributions of the calibration parameters are provided in Fig. 7.

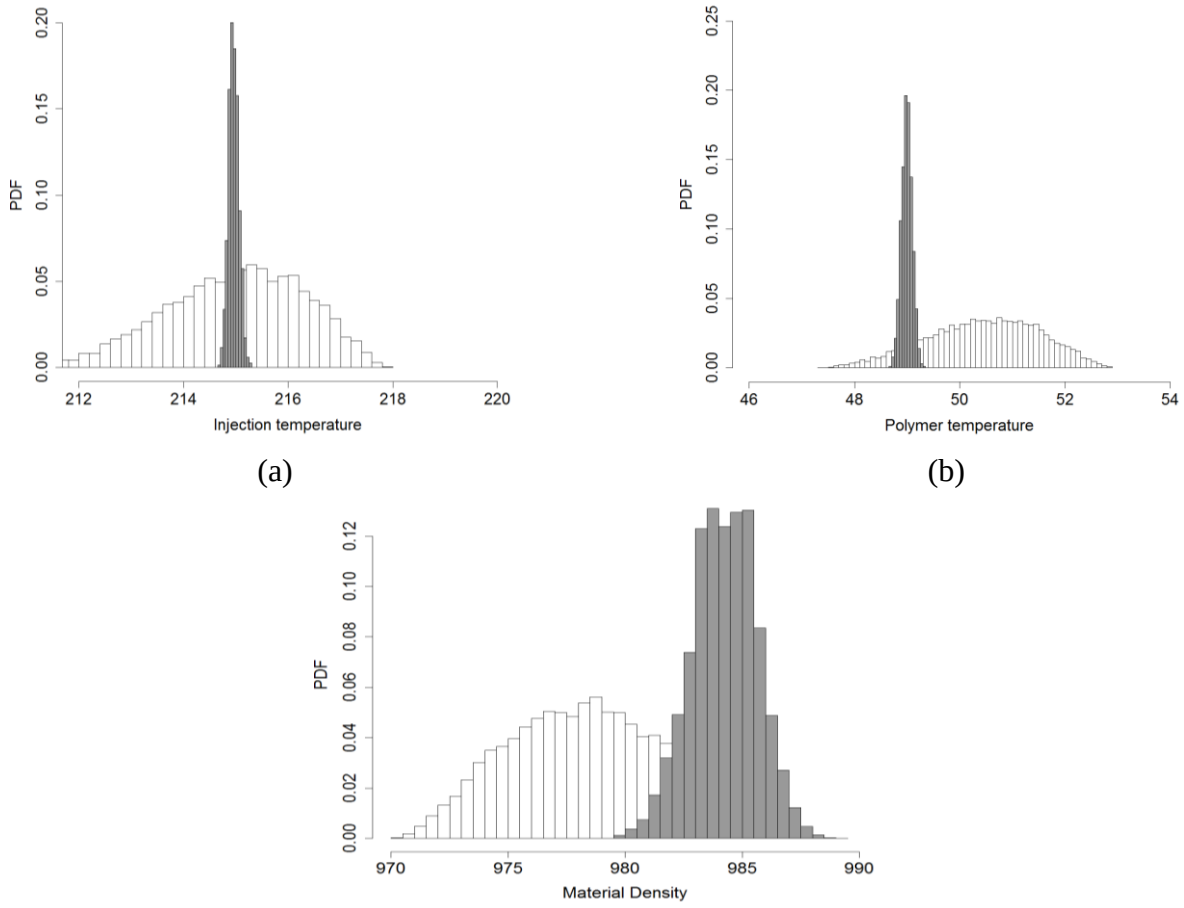


Fig. 7 Prior and Posterior distributions of calibration parameters (a) Injection temperature, (b) Polymer temperature, and (c) Material density

5. Conclusion

This paper proposed an automated uncertainty quantification analysis methodology using a hierarchical Bayesian network. The HBN is constructed by fusing the information from available system models, physics-based models, and data. The system model is assumed available in a domain-specific modeling

framework such as the GME platform. Physics-based mathematical models regarding multiple individual processes in a manufacturing network are assumed to be available in multiple .txt files.

A meta-model for the HBN is developed along with a syntactic representation of the conditional probability distribution/tables in GME. The meta-model is created such that the HBN can have discrete, continuous, and functional nodes. Two model interpreters (also called model translators) are written on the HBN meta-model to construct a HBN and to perform UQ analysis. An algorithm is presented to generate the HBN from a set of available physics-based models. If physics models are not available to construct the entire HBN, data is used to learn the HBN through learning algorithms.

The constructed HBN is then represented as an instance model of the HBN meta-model. The user has the capability to modify the instance model and include any further information such as observation data and post-processing information. The HBN instance model is then used to perform model calibration and UQ analysis. The Approximate Bayesian Computation (ABC) technique is used to perform model calibration as the traditional Markov Chain Monte Carlo (MCMC) methods can become expensive for large networks. The proposed method is illustrated using an injection molding process example.

The goal of this work is to assist manufacturers in performing uncertainty analysis using automated tools, without requiring expertise in UQ methods and UQ-specific tools. This will help manufacturers make better use of their data analytics capabilities by allowing them to consider uncertainty. Future work is needed to develop algorithms for the automation of analyses such as sensitivity analysis (for dimension reduction), model verification, and model validation. In addition, algorithms for the automated construction of dynamic Bayesian networks (for tracking manufacturing systems over time), and diagnostic and prognostic analysis need to be investigated.

Acknowledgment

The research reported in this paper was funded in part by the National Institute of Standards and Technology (NIST) under Cooperative Agreements No. 70NANB14H036 and No. 70NANB13H159, and NIST's Foreign Guest Researcher Program.

References

[1] Bae, H.R., Grandhi, R.V., and Canfield, R.A., "Epistemic uncertainty quantification techniques including evidence theory for large-scale structures," *Computers & Structures, 82*(13), pp.1101-1112, 2004.

[2] Jensen, F. V., *An Introduction to Bayesian Networks*, Springer-Verlag, 1996.

[3] Dahll, G., "Combining disparate sources of information in the safety assessment of software-based systems," *Nuclear Engineering and Design, 195*(3), pp.307-319, 2000.

[4] De Campos, L.M., Fernández-Luna, J.M., and Huete, J.F., "Bayesian networks and information retrieval: an introduction to the special issue," *Information processing & management, 40*(5), pp.727-733, 2004.

[5] Friedman, N., Linial, M., Nachman, I., and Pe'er, D., "Using Bayesian networks to analyze expression data," *Journal of computational biology, 7*(3-4), pp.601-620, 2000.

[6] Jiang, X., Neapolitan, R.E., Barmada, M.M., and Visweswaran, S., "Learning genetic epistasis using Bayesian network scoring criteria," *BMC Bioinformatics, 12*(1), p.1, 2011.

[7] Bensi, M., and Der Kiureghian, A., "Seismic hazard modeling by Bayesian network and application to a high-speed rail system," Proceedings of *International Symposium on Reliability Engineering and Risk Management*, (Ed: J. Li), Tongji University Press, Shanghai, China, September 2010.

[8] Sankararaman, S., Ling, Y., and Mahadevan, S., "Uncertainty quantification and model validation of fatigue crack growth prediction," *Engineering Fracture Mechanics, 78*(7), pp.1487-1504, 2011.

[9] Liang, B., and Mahadevan, S., "Error and uncertainty quantification and sensitivity analysis in mechanics computational models," *International Journal for Uncertainty Quantification, 1*(2), 2011.

[10] Reza, B., Sadiq, R., and Hewage, K., "A fuzzy-based approach for characterization of uncertainties in emergy synthesis: an example of paved road system," *Journal of Cleaner Production, 59*, pp.99-110, 2013.

[11] Hsu, B.M., and Shu, M.H., "Fuzzy inference to assess manufacturing process capability with imprecise data," *European Journal of Operational Research*, *186*(2), pp.652-670, 2008.

[12] Ullah, A.S., and Harib, K.H., "Manufacturing process performance prediction by integrating crisp and granular information," *Journal of Intelligent Manufacturing*, *16*(3), pp.317-330, 2005.

[13] Pehlken, A., Decker, A., Kottowski, C., Kirchner, A., and Thoben, K.D., "Energy efficiency in processing of natural raw materials under consideration of uncertainties," *Journal of Cleaner Production*, *106*, pp.351-363, 2015.

[14] Nannapaneni, S., Mahadevan, S., and Rachuri, S., "Performance evaluation of a manufacturing process under uncertainty using Bayesian networks," *Journal of Cleaner Production, 113*, pp.947-959, 2016.

[15] Nannapaneni, S. and Mahadevan, S., 2016. "Manufacturing Process Evaluation Under Uncertainty: A Hierarchical Bayesian Network Approach," *ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (pp. V01BT02A026-V01BT02A026). American Society of Mechanical Engineers*.

[16] Lechevalier, D., Narayanan, A., and Rachuri, S., "Towards a domain-specific framework for predictive analytics in manufacturing," Proceedings of the *2014 IEEE International Conference on Big Data (Big Data)*, (pp. 987-995). IEEE, 2014.

[17] Lédeczi, Á., Bakay, A., Maroti, M., Volgyesi, P., Nordstrom, G., Sprinkle, J., and Karsai, G., "Composing domain-specific design environments," *Computer*, *34*(11), pp.44-51, 2011.

[18] Nannapaneni, S., Dubey, A., Abdelwahed, S., Mahadevan, S., Neema, S., and Bapty, T., "Mission-based reliability prediction in component-based systems," *International Journal of Prognostics and Health Management*, *7*(001), 2016.

[19] Lechevalier, D., Hudak, S., Ak, R., Lee, Y. T., and Foufou, S., "A neural network meta-model and its application for manufacturing," Proceedings of the *2015 IEEE International Conference on Big Data (Big Data)*, (pp. 1428-1435). IEEE, 2015.

[20] Del Aguila, I. M., and Del Sagrado, J., "Metamodeling of Bayesian networks for decision-support systems development," *Knowledge Engineering and Software Engineering (KESE8)*: 8, August 2012.

[21] Nannapaneni, S., Mahadevan, S., Lechevalier, D., Narayanan, A. and Rachuri, S., 2015. "Automated uncertainty quantification analysis using a system model and data," *Proceedings of the IEEE International Conference on Big Data (Big Data), pp. 1408-1417, IEEE*.

[22] Scutari, M., 2009. "Learning Bayesian networks with the bnlearn R package," *arXiv preprint arXiv:0908.3817*..

[23] Neapolitan, R. E., *Learning Bayesian networks*, Pearson, 2004.

[24] The Unified Modeling Language. http://www.omg.org/spec/UML/2.5/ (Accessed 02/2017)

[25] Kulkarnia, A., Balasubramaniana, D., Karsaia, G., Narayananb, A. and Dennoc, P., 2016. "A domain-specific language for model composition and verification of multidisciplinary models," *Proceedings of the 2016 Annual Conference on Systems Engineering Researach, Huntsville, Alabama, USA*.

[26] Ling, Y., and Mahadevan, S., "Quantitative model validation techniques: New insights", *Reliability Engineering & Systems Safety, 111*, 217-231, 2013

[27] Sunnåker, M., Busetto, A.G., Numminen, E., Corander, J., Foll, M. and Dessimoz, C., 2013. "Approximate bayesian computation," *PLoS Comput Biol, 9(1), p.e1002803*.

[28] Smith, A.F. and Gelfand, A.E., 1992. Bayesian statistics without tears: a sampling–resampling perspective. *The American Statistician*, *46*(2), pp.84-88.

[29] Fox, C.W. and Roberts, S.J., 2012. A tutorial on variational Bayesian inference. *Artificial intelligence review*, *38*(2), pp.85-95.

[30] Madan, J., Mani, M., and Lyons, K. W., "Characterizing energy consumption of the injection molding process," Proceedings in *ASME 2013 International Manufacturing Science and Engineering Conference* collocated with the 41st North American Manufacturing Research Conference (pp. V002T04A015-V002T04A015), 2013.