Integrating Materials and Manufacturing Innovation (2019) 8:95–106
https://doi.org/10.1007/s40192-019-00140-9

THEMATIC SECTION: 5TH WORLD CONGRESS ON INTEGRATED COMPUTATIONAL MATERIALS ENGINEERING

Bayesian Framework for Inverse Inference in Manufacturing Process Chains

Avadhut Sardeshmukh[1] · Sreedhar Reddy[1] · B. P. Gautham[1]

Received: 1 February 2019 / Accepted: 7 May 2019 / Published online: 28 May 2019

Abstract

Process-property relations are central to ICME. Engineers are often interested in using these relations to make decisions on process configurations to achieve desired properties. This is known as the inverse problem and is typically solved using forward models (physics-based or data-based) in an optimization loop, which can sometimes be expensive and error prone, especially when used on process chains with multiple unit steps. We propose a Bayesian networks-based approach for modeling process-property relations that can be used for inverse inference directly. The solutions thus found can serve as good starting points for a more detailed simulation-based search. We also discuss how unit process models can be composed to do inverse inference on the process chain as a whole. We demonstrate this in a wire-drawing process where a wire is drawn in multiple passes to achieve desired properties. We learn a Bayesian network for a unit pass and compose it multiple times to infer process parameters of all passes together.

Keywords Bayesian networks · Inverse inference · Parallel tempering · Manufacturing process chain · Bayesian network composition

Introduction

Modeling the process-structure-property correlations for a material system is perhaps one of the most important problems in ICME. Toward this end, physics-based simulation models of manufacturing processes are often used to perform "what-if" types of analyses. But, more often the engineers are interested in inverse analysis—that is, inferring process configurations required to achieve desired properties. The relationships between processing parameters and resulting properties are often highly nonlinear and complex. Also, multiple process configurations may lead to the same final properties. As a result, inverse inference is typically an ill-posed problem. Commonly known discriminative machine learning approaches (such as neural networks) do not work well. The problem is further complicated by the fact that a manufacturing process consists of multiple unit steps and so the design space is huge. The solution space needs to be searched in an integrated manner so that the decisions made in one unit step are systematically propagated to the previous unit step and through the entire process chain. Probabilistic models can handle ill-posed inference and systematically quantify uncertainty in predictions. By characterizing multi-modal posterior distributions, multiple solutions to the inverse problem can be extracted (see Section "Parallel Tempered MCMC"). We propose to model the problem using a variant of conditional linear Gaussian (CLG) Bayesian network, which can learn the abovementioned nonlinearity through a suitable piecewise linear approximation. A major challenge in leveraging probabilistic models is the computation of posterior distribution. Approximate methods such as Markov chain Monte Carlo (MCMC) sampling are generally used for this purpose. However, as the inverse problem could have multiple solutions, the posterior in our case is expected to be a multi-modal distribution. MCMC techniques typically find

This article is part of the Topical Collection on *5th World Congress on Integrated Computational Materials Engineering*

✉ Avadhut Sardeshmukh
avadhut.sardeshmukh@tcs.com

[1] TRDDC, TCS Research, Tata Consultancy Services, Pune, India

it difficult to explore different modes of a multi-modal distribution. We use a sampling method from statistical mechanics, called parallel tempered MCMC, which uses multiple samplers at different temperatures to make movement between modes easier. A further challenge is to build a complete manufacturing process chain model from the individual unit step models. We perform Bayesian network composition to achieve this. Inverse inference on the composite model can then be performed using our method based on parallel tempering. We demonstrate the approach on the problem of multi-pass wire drawing. In multi-pass wire drawing, a wire of smaller cross section is drawn from a core with larger cross section. This is typically done in multiple passes. Determining optimal number of passes and their configurations required to achieve the desired reduction while optimizing properties such as tensile strength, strain distribution, and energy consumption is the design problem. We learn a Bayesian network model of a unit pass and compose these unit pass models to get a multi-pass model. We discuss the results of inverse inference on the composite model using our method. The rest of the paper is organized as follows: Related work is discussed in Section "Related Work." The proposed methodology for inverse inference on a manufacturing process chain is described in Section "Methodology." Section "Evaluation" discusses the experiments and results obtained on the wire-drawing process model. Finally, Section "Conclusion" concludes with a summary of the approach and achieved results.

Related Work

With the recent advances in machine learning and artificial intelligence, there is a renewed interest among materials scientists to leverage these advances for building better models of process-structure-property correlations ([1, 2] and [3]). A recent review of the impact of machine learning in materials and process engineering appears in [4]. The authors point to the need for increasing the adoption of these techniques in materials engineering, similar to what has been done in biology and chemistry communities. Some of the recent work in this direction includes application of machine learning to characterize material microstructure and link it to properties using deep neural networks (e.g., [5–7]). However, the inverse problem has not received as much attention. Traditionally, the inverse problem is solved by optimization around approximate forward models. For example, genetic algorithms have been used in [8] for inversion of elastic properties in a class of materials. However, this method can be slow due to the large number of forward model evaluations required in optimization. Moreover, obtaining multiple solutions using optimization is very difficult. A large number of optimizations, starting at various points in the design space (sampled carefully such that the space is filled) are required to ensure convergence to different solutions (see Section "Comparison with Optimization Approach" for more discussion on this). A set of related works by Zabaras et al. [9, 10] focus on developing variational inference methods for inverse problem. Variational inference is being proposed as a faster alternative to sampling techniques for approximate Bayesian inference. However, this is still in the exploratory state. To the best of our knowledge, there is no reported work on a compositional approach for (forward and) inverse inference in manufacturing process chains.

The specific problem of inverse inference in wire drawing has been previously addressed in [11]. The authors present an optimization method based on their forward finite element analysis (FEA) simulation model of the wire-drawing process. Pandita et al. [12] use Bayesian global optimization to further accelerate the forward search by quantifying the merits of evaluating the forward FEA model at a given point in design space. In [13] and [14], the present authors have formulated a Bayesian approach for inverse inference in a unit manufacturing process step using MCMC sampling. However, sampling using just MCMC from a multi-modal posterior is difficult (see Section "Parallel Tempered MCMC"). The method proposed in [14] relies upon an exhaustive mode-sweeping approach, which leads to multiple number of samplings and does not scale well to a composite model.

Methodology

As discussed in Section "Introduction," multiple process parameter configurations can lead to the same properties, that is, their relationship is typically M to 1 in nature. Due to this, discriminative modeling approaches such as artificial neural networks and support vector regression end up learning averages when they are used directly for inverse inference. A probabilistic generative model, on the other hand, learns the joint probability distribution of all variables, by utilizing independence assumptions from the domain. The joint distribution can then be used to perform either forward or inverse inference by obtaining conditional distributions.

Conditional Linear Gaussian Bayesian Network

Bayesian networks provide a compact representation of the joint distribution of variables in the problem by utilizing conditional independence assumptions from the domain [15]. A Bayesian network is a directed acyclic graph whose nodes represent variables in the problem and edges represent direct influence relations. A conditional probability

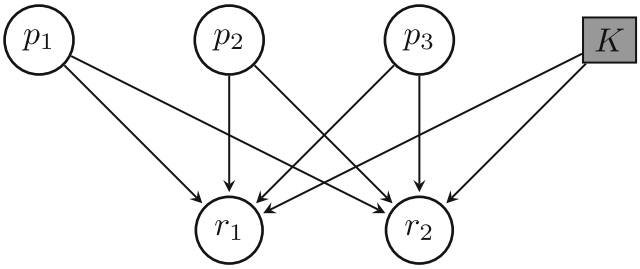

Fig. 1 Example CLG Bayesian network model of a manufacturing process

distribution (CPD) given the parents, is associated with every node of the graph. For continuous variables, a common choice of distributions is the Gaussian. With this assumption, the conditional probability distribution at each node is a Gaussian whose mean is a linear function of its parents. This formulation can be thought of as equivalent to linear regression with L_2 regularization [16]. Consequently, it is difficult to model complex nonlinear relationships accurately using this model. Instead, we use a variant known as conditional linear Gaussian (CLG) [17].

A conditional linear Gaussian (CLG) Bayesian network can have both discrete and continuous nodes. The CPD of a node having both continuous and discrete parents is a mixture of linear Gaussians—one component for each combination of discrete parents' values weighted by the probability of that combination. An example CLG Bayesian network is shown in Fig. 1, where a circle represents a continuous variable, and a rectangle, a discrete variable. It can be thought of as modeling a unit manufacturing process with process parameters $p_{1...3}$ and properties $r_{1...2}$. Consider the property r_1. It has continuous parents p_1, p_2, and p_3, and discrete parent K. So, for each possible value of K, the distribution of r_1 is a linear Gaussian in p_1, p_2, and p_3. In other words,

$$r_1|p_1, p_2, p_3 \sim \sum_{k=1}^{K} P(K = k).N(r_1; w_{k,0} + w_{k,1}p_1 + w_{k,2}p_2 + w_{k,3}p_3, \sigma_k^2) \quad (1)$$

Note that K here, represented by a shaded rectangle, is a latent node. The value of this node identifies the correct linear piece applicable for the current configuration of process parameters.[1] This formulation can be thought of as equivalent to piece-wise linear regression. Hence, we expect it to approximate the target nonlinear relationships better.

[1]This suggests there should be an edge from p_i's to K, which gives the exact model we have used. This model is a variant on CLG because in standard CLG, a discrete node can't have continuous parents. This variant is called mixture of experts [17].

Parameter Estimation and Structure Learning

The structure of the network usually comes from domain knowledge of which process parameters influence which properties. Structure-learning algorithms such as hill-climbing [18] (based on optimizing information-theoretic criteria such as Bayesian information criterion or Akaike information criterion) can also be used to learn the edges. We take a hybrid approach. We first use hill climbing to learn a structure from data. The learned structure is then validated with domain experts.

We use a single latent discrete node to model the mixture-of-Gaussian representation for all properties. Parameters of the network can be estimated by maximizing the likelihood of observed (either experimental or simulation) data. Maximum likelihood estimation in presence of latent variables can be done using expectation maximization. More details about training the network appear in Section "Experimental Setup," where we discuss the experimental setup in the context of the wire-drawing use case.

Inference

Once the structure and parameters of the network are learned, we have a compact representation of the joint distribution of all variables. To perform forward or inverse inference, we need to compute conditionals using this joint distribution. The conditional distribution required for forward inference (conditional distribution of properties given process parameters) is tractable, as can be seen from Eq. 1 given in Section "Conditional Linear Gaussian Bayesian Network." There exist algorithms such as junction tree for efficiently and exactly computing these distributions (see [15] for details of various inference problems, their tractability and algorithms). However, inverse inference is a much harder problem because of some intractable quantities involved. For example, consider the model shown in Fig. 1. To predict p_1, p_2, and p_3 for a given r_1, we need to compute $P(p_1, p_2, p_3|r_1)$. Using Bayes' theorem, this can be written as:

$$P(p_1, p_2, p_3|r_1) = \frac{P(p_1, p_2, p_3, r_1)}{P(r_1)} = \frac{P(p_1, p_2, p_3, r_1)}{\int_{p_1}\int_{p_2}\int_{p_3} P(p_1, p_2, p_3, r_1)dp_1dp_2dp_3} \quad (2)$$

The denominator in this equation is particularly difficult to compute because it involves multiple integration.

To cope with this difficulty, generally, approximate methods of inference are used. MCMC sampling is perhaps one of the most commonly used techniques for approximate

inference. Please see (especially the first chapter of) [19] for details of MCMC. However, in our case, we expect the posterior to be multi-modal because of the M to 1 nature of the relationship between parameters and properties. It is well-known that MCMC algorithms (especially the Metropolis-Hastings algorithm) find it very difficult to sample from a multi-modal distribution exploring all its modes, because it cannot easily cross the low-probability regions between two modes [20]. Hence, we make use of a method called parallel tempered MCMC [21] from statistical mechanics, designed to handle multi-modal distributions. This is described in the next section.

Parallel Tempered MCMC

The main idea behind parallel tempering [21] is to modify the target probability distribution using a "temperature" parameter such that it is easier to sample from. As the temperature is raised, the distribution becomes flatter. For example, consider the probability distribution defined by density:

$$p(x) \propto e^{-\frac{1}{T}(x^2-1)^2}$$

Figure 2 shows the effect of raising the temperature T of this distribution. In general, a tempered form of a given distribution $p(x)$ can be written as follows:

$$p(x, t_k) = e^{-\frac{1}{t_k}\log p(x)}$$

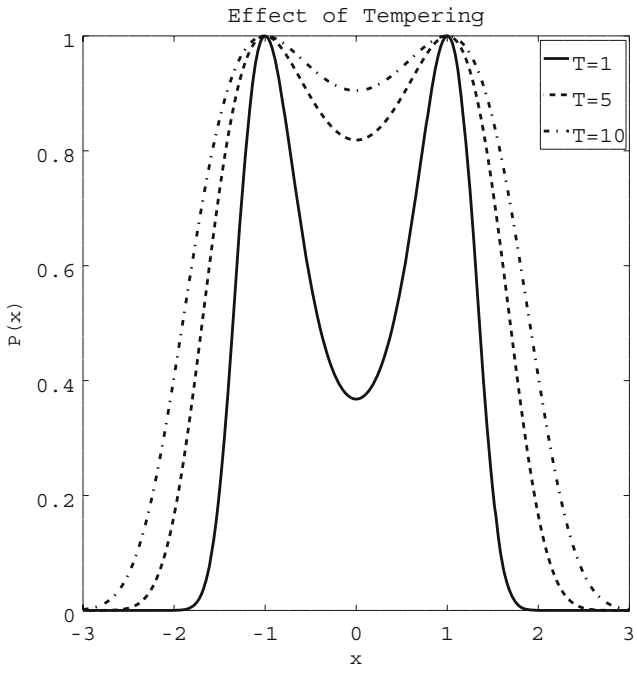

Fig. 2 Effect of raising temperature T in $p(x) \propto e^{-\frac{1}{T}(x^2-1)^2}$

Thus, using a set of increasing temperatures t_k for $k = 1 \ldots n$ (also known as the temperature ladder), a set of tempered replicas of the original target distribution are obtained. Due to the effect of temperature, these distributions become increasingly flat, as we move up the temperature ladder. A basic MCMC sampler is then run for each of the targets. As observed already, it is easier to sample from the high-temperature targets due to their flatter surfaces. But the original target distribution is the one at temperature 1 (i.e., the non-tempered one). The purpose of tempering is to enable movement between multiple modes of the original target distribution by leveraging the free movement in high-temperature targets. For this, adjacent chains periodically propose a swap of their states. This swap is proposed and accepted (or rejected) just like a normal MCMC move. Due to the swap move, all the samplers (at all temperatures) get coupled together to form a joint chain. As the swap is a symmetric move, the detailed balance condition in MCMC is satisfied.

The success of parallel tempering critically depends on how well the information (and movement) from the hottest chain propagates to the coolest chain. The temperature ladder (number of temperatures and spacing between them) has to be carefully chosen to ensure that the swaps between any two chains occur with almost uniform probability. It was shown in [22] that a uniform swap acceptance ratio (i.e., ratio of number of swaps accepted to total number of swaps proposed) of 0.24 in general leads to good mixing. We make use of this while tuning the temperature ladder. More on this is discussed in Section "Single-Pass Inverse Inference."

Bayesian Network Composition

A manufacturing process chain typically consists of multiple unit steps. Many of these unit steps are common among multiple manufacturing process chains, for example, heating. A relevant question in this context is, can we reuse models of unit steps to compose a model of the process chain? If so, we can build a library of unit step models and reuse them as required. It has been shown by Koller et al. [23] that Bayesian networks are amenable to such composition. A Bayesian network has the property that a node is independent of all other nodes, given its parents. Therefore, two unit step models can be composed if outputs of one model match with the inputs of the other. We say that an output out_i^p of process p "matches" with the input in_j^q of process q if (i) they refer to the same physical quantity (e.g., a stress value) and (ii) the possible value range of out_i^p is contained in the possible value range of in_j^q. For example, a gear design process might consist of the unit processing steps such as carburization, diffusion, quenching, and tempering. Figure 3 shows the composition of the first two of these unit steps, namely carburization

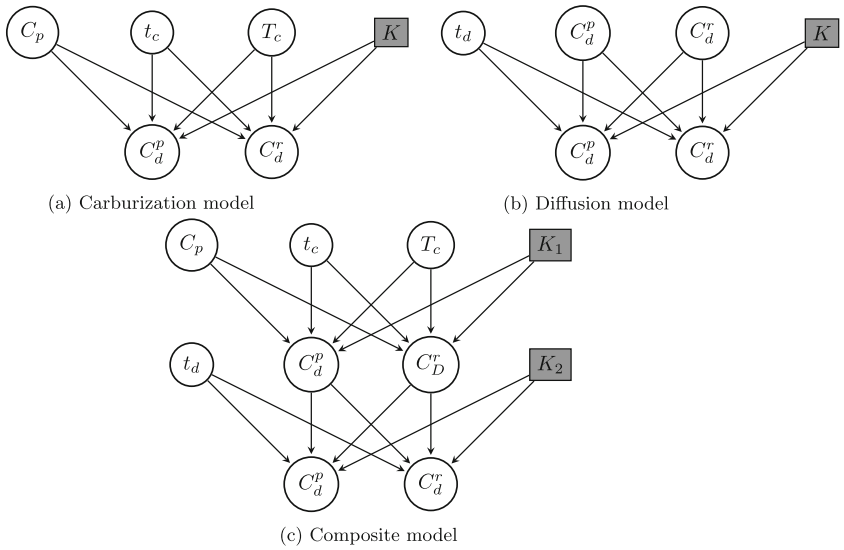

Fig. 3 Composition of Bayesian network models of carburization and diffusion processes

(Fig. 3a) and diffusion (Fig. 3b). Parameters of carburization are carburization time (t_c), temperature (T_c), and carbon potential (C_p). Similarly, parameters of diffusion are, time t_d, and case depth. Figure 3c shows the composition of these two, where outputs of carburization flow as inputs to diffusion. For the purpose of illustration, the figure shows only case depth at pitch C_d^p and case depth at root C_d^r, though there are more outputs from carburization feeding into diffusion. In the composite model, the conditional distributions of case depth at pitch C_d^p and case depth at root C_d^r are taken from the carburization process (note that they do not have conditional distributions in diffusion because they are at the root level).

This process can be repeated to obtain a composite Bayesian network model for the whole manufacturing process chain. Inverse inference on the chain can then be performed using parallel tempered MCMC as explained in Section "Parallel Tempered MCMC."

Evaluation

We apply the methodology described in Section "Methodology" to the problem of multi-pass wire drawing. In a multi-pass wire drawing process, a wire of larger cross section is reduced to a smaller cross section by pulling it through a series of dies. This process is used to produce high-strength wires of desired diameter by imparting large cold deformation. Figure 4 shows the schematic of a pass. The die angle α and reduction ratio $\frac{D_{\text{out}}}{D_{\text{in}}}$ are the key parameters which impact the properties of the wire. A pass schedule mainly consists of these two parameters for each pass. Designing a pass schedule required to achieve the desired final reduction, while optimizing properties of wire is of significant importance. This can also be seen as a special case of inverse inference on a manufacturing process chain. The unit steps in this chain are the passes and they are all identical processes. We first learn a conditional linear Gaussian Bayesian network model of a single pass (unit step model) and show the results of inverse inference on it using parallel tempered MCMC. We then create a composite 2-pass model of wire drawing by composing two single-pass models (two copies of the same model) and show results of both forward and inverse inference on the composite model.

Experimental Setup

Dataset

To generate data, we used an FEA simulation model of the wire drawing process [24]. The data is generated for an 8-pass setting which is the most commonly used configuration in industry. Reduction ratio and die angle (and draw speed) for each of the 8 passes, and the initial diameter

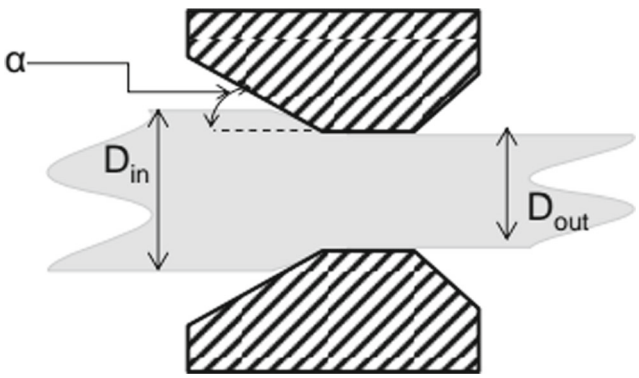

Fig. 4 Schematic of a wire drawing pass

Table 1 Input variable ranges for data generation

Input (Units)	Minimum	Maximum
Initial diameter(mm)	5	14
Reduction ratio (%)	5	20
Die angle (Degree)	3	18
Draw speed (m/s)	2	7

(input to the first pass) were generated randomly from pre-specified ranges as shown in Table 1. These were given as inputs to the simulator and resulting output properties of interest at each pass were recorded. These included output diameter, strain non-uniformity factor (SNUF), ultimate tensile strength (UTS), energy consumed, hydraulic failure factor (HFF), and various stresses (such as axial and radial). Each simulation resulted in an 8-pass data point. We got 3750 such data points in all. To learn a model of a single pass, we need input-output data of a single pass. In a multi-pass setting, some of the inputs to a pass come from the outputs of previous pass, while some are unique to a pass (die angle and reduction ratio). So each 8-pass data point can give us 7 single-pass data points—one for each pair of passes. We got total 26250 single-pass data points in this way. Out of these, 2000 data points were chosen randomly and used for training and cross-validation. Another set of 200 randomly chosen points was kept aside for testing the single-pass model. Yet another set of 200 2-pass data points (combining data of pass 5 and 6) was kept aside for testing the composite model.

Implementation Details

The structure of the Bayesian network was learned using hill-climbing algorithm with the BIC objective. We used the bnlearn [25] package from R for this purpose. Parameters of the network were learned using the BN Toolkit (BNT) by Kevin Murphy [26]. We implemented parallel tempering-based inverse inference using the packages PyMC [27] and Emcee [28]. While the actual sampling was done using emcee, the log-likelihood computation at each step during sampling was done using PyMC.

To demonstrate the composition method, we composed two copies of the single-pass model to get a 2-pass model as discussed in Section "Bayesian Network Composition." Inverse inference was then performed on the composite model as discussed above.

Experiments

The structure of the Bayesian network for a single pass learned using hill climbing (see Section "Parameter Estimation and Structure Learning") is shown in Fig. 5. The discrete node and its edges are not shown in this figure, because they are not learned by hill climbing. However, they are added to the network before parameter estimation.

The pass specific inputs are reduction ratio (RR) and die angle (DA). The other inputs that come from the previous pass are wire diameter (DIA), strain non-uniformity (PSN), axial stresses (center line-PCA and max-PMA), radial stresses (center line-PCR and max-PMR), and max hoop stress. The outputs of a pass include these variables (which become inputs for next pass) and hydraulic failure factor, energy consumed (per ton) and ultimate tensile strength (UTS).

Parameter estimation was done using expectation maximization. The number of values that the discrete node can take is treated as a hyper-parameter which we optimize using cross-validation of forward inference. We observed that with 30 values (i.e., a 30-component mixture model for each property), we get sufficiently good accuracy. We performed following three types of inference using the learned model.

1. Forward inference

   We compute the conditional distribution of properties, given process parameters. We predict the wire properties at the end of one (two in case of composite inference) pass using this distribution. The required conditional distribution is computable, as discussed in

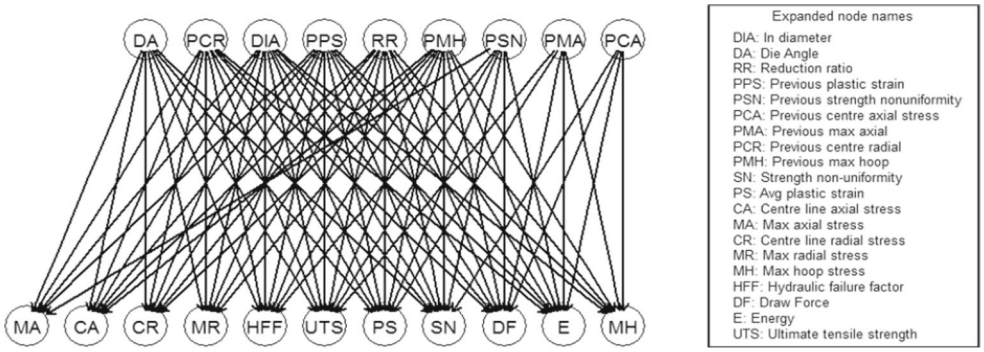

Fig. 5 Bayesian network structure for a pass of wire-drawing

Table 2 Accumulation of error from a single-pass to a two-pass forward inference

	Plastic strain	SNUF	Axial stress	HFF	Energy	UTS
Single pass	0.0015	0.1053	0.1122	0.1251	0.1728	0.0028
Two pass	0.0023	0.1407	0.1108	0.1226	0.1712	0.0031

Section "Inference." Hence, instead of sampling, we used the junction tree algorithm from BNT, which is more efficient.

2. Single-pass inverse inference

   As the goal is to predict process parameters that can lead to desired properties, and there can be multiple such configurations, it is not appropriate to compare the inferred parameters with actual values in test data. Instead, we put the inferred process parameters through forward simulation and compare the obtained outputs with the desired ones.

3. Composite inverse inference

   Given the properties at the end of two passes, we infer the parameters of pass 1 and 2 required to achieve those properties. For validation, we put the inferred process parameters through forward simulation and compare the obtained outputs of pass 2 with the desired ones.

Results and Discussion

We now present results of the experiments listed in Section "Experiments."

Forward Inference

For forward inference, we compare the predicted properties with the outputs of pass 2 from the test data. Table 2 shows the root mean-squared errors for single-pass and two-pass forward inference. As can be seen from the table, in most cases, the accuracy does not drop significantly from single pass to two pass. In one case (SNUF), the RMS error goes from 0.1 to 0.14. The reason behind this deviation is not clear. However, for this property, a root mean square (RMS) error of 0.14 is considered acceptable by domain experts.

Single-Pass Inverse Inference

Inverse inference consists of computing the conditional distribution of process parameters—that is, a joint distribution over reduction ratio and die angle—given the desired outputs. As explained already, this posterior distribution is expected to be multi-modal. Using the parallel tempering method described in Section "Parallel Tempered MCMC," we get samples from this multi-modal distribution. If the sampler has explored multiple peaks of the posterior (which we expect from a parallel tempering method), the set of samples should have more samples around those peaks. This depends upon how well the chain has mixed through the target distribution. A number of empirical tests for convergence and proper mixing of chains have been suggested in literature (for example, [29]). For a tempering-based method, the key factor for proper mixing is that the swaps of states between adjacent chains proposed periodically are accepted uniformly over the entire range of temperatures. This makes sure that the information from the hottest chain ripples down to the coolest chain. We manually tuned the number of temperatures to 20 so as to achieve a uniform probability of swap acceptance close to 0.24 (as suggested in [22]) using the default geometric spacing of temperatures suggested by Emcee. We ran an ensemble of 1000 walkers for 20 steps, after discarding first 5 (burn-in) to allow the sampler to mix well. Each sample is a three-dimensional vector of ⟨reduction_ratio, die_angle, discrete_node⟩ values. We cluster these samples by value of the discrete node. Centroids of the clusters represent modes of the multi-modal posterior. We use them as solutions to the inverse problem.[2] So for each possible value of the discrete node, we can potentially get one solution. These solutions are ranked by their log-likelihood which we compute using the learned model. The solution that has highest log-likelihood is given rank 1 (thus, the smaller the number, the higher the rank). We perform forward simulation starting with these solutions to get the outputs and compute root mean-squared error with desired values. The root mean squared error is computed in Z-score normalized space. Figure 6 shows for some of the important properties, how the root mean-squared error increases as we choose solutions ranked 1 to 30. This shows that the solutions with high ranks are indeed better, thereby experimentally validating the log-likelihood-based ranking. The next question is, how many of these top-ranked solutions should we pick as acceptable solutions for the inverse inference problem. To determine this, we plot the log-likelihood against solution rank. Figure 7 shows the average log-likelihood of solutions at each rank, over the test data set. It shows a clear trend. After the first few solutions (in this case 5), the log-likelihood drops sharply, suggesting a clear cut-off point.

The root mean squared errors using only the predictions from top-ranked solutions are shown in Table 3. We also show a scatter plot of desired vs. obtained property values

[2] The associated variance in the corresponding clusters can be used to quantify uncertainty. This is not reported in the present work, however.

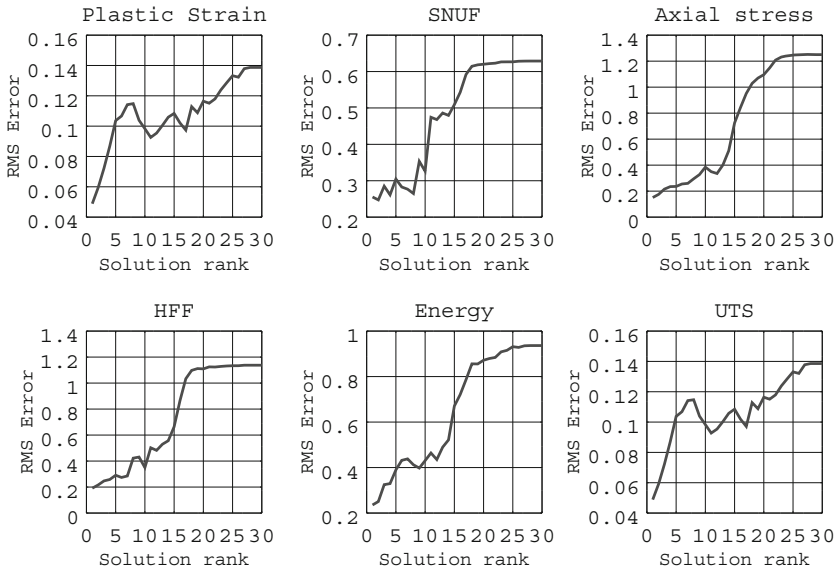

Fig. 6 RMS error for multiple solutions ranked by log-likelihood

using the top-ranked solution in Fig. 8. The predictions for UTS and average plastic strain are much more accurate as compared to others. This can be explained as follows. It is known that UTS and average plastic strain are strong functions of reduction ratio. Due to this, the training data itself has very low noise. This is not the case for other properties.

Composite Inverse Inference

The composite model obtained as explained in Section "Bayesian Network Composition" was used to perform inverse inference using parallel tempering method, similar to how it was done for the single pass model as discussed in Section "Single-Pass Inverse Inference". The difference here is that we now have four parameters to predict - the die angle and reduction ratio for two passes.

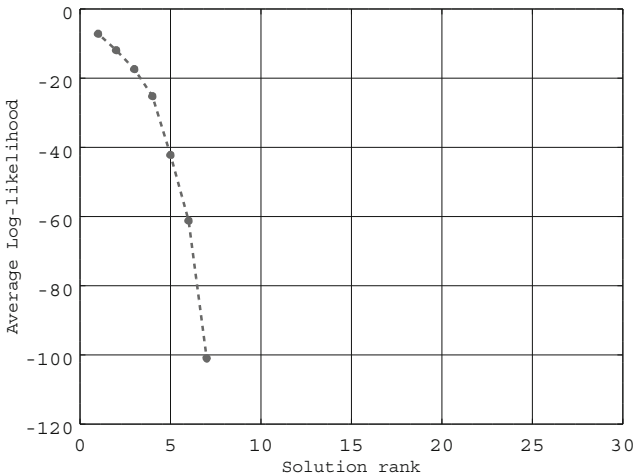

Fig. 7 Average log-likelihood of solutions at each rank

As the design space is larger, we need to tune the number of temperatures and the temperature spacing once again. We again manually tuned the number of temperatures to achieve a uniform swap acceptance of 0.24 and it turned out that we needed 30 temperatures. We ran an ensemble of 2000 walkers for 40 steps after discarding the first 10 (thus increasing the required number of samples 6 times as compared to single pass). We obtain four-dimensional solutions $\langle \text{die\_angle}_1, \text{reduction\_ratio}_1, \text{die\_angle}_2, \text{reduction\_ratio}_2 \rangle$, one for each combination of values of discrete nodes. As both of them can take 30 different values, we can potentially get 900 solutions for each test data point. We again rank them by their log-likelihood. We compute the final obtained properties by forward 2-pass simulation starting with these solutions. The obtained properties are then compared with the desired outputs of pass 2 in our test data. Figure 9 shows how the root mean squared error (computed in Z-score normalized space) grows as we choose lower ranked solutions. Following the discussion in Section "Single-Pass Inverse Inference," we only consider solutions corresponding to top five values of the mode variable $K^{(2)}$ (discrete node from the second-pass model). We have averaged the RMS error over blocks of 30 to smoothen the variations due to almost similar probabilities of the combinations $\langle K^{(2)}, K_i^{(1)} \rangle$ for $i = 1 \ldots 30$. So, we are considering the top five solutions coming from 150 combinations of the 900 possible. We also observed that we rarely get any samples corresponding to more than 200 combinations out of the possible 900.

Table 3 Single-pass inverse inference root mean-squared errors

Output	Plastic strain	SNUF	Axial stress	HFF	Energy	UTS
RMSE	0.0489	0.2554	0.1507	0.1912	0.2355	0.0488

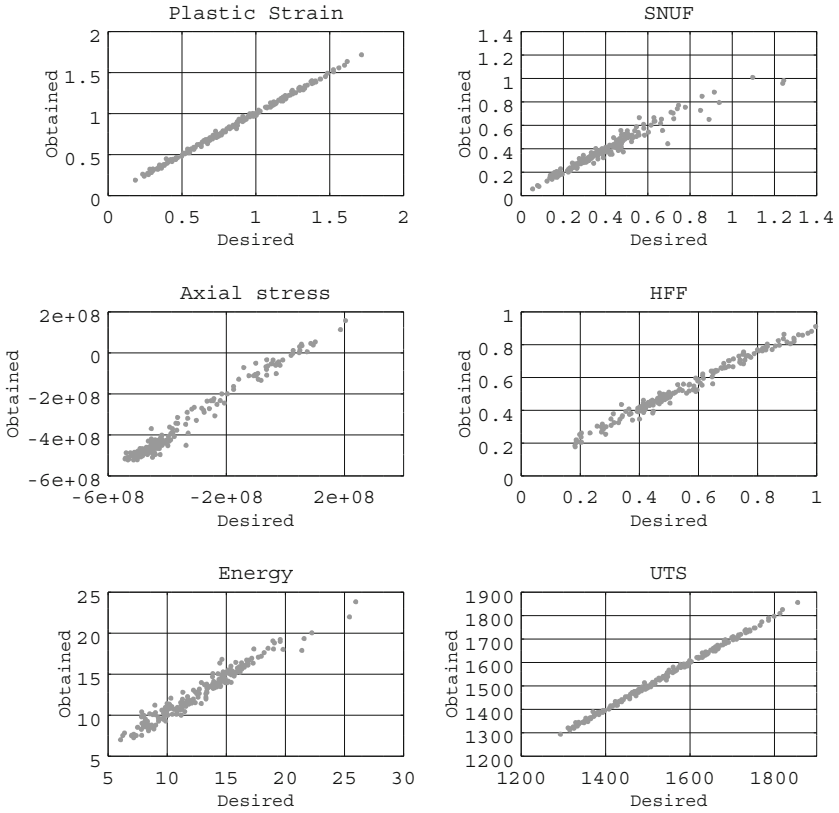

Fig. 8 Scatter plot of desired vs. obtained property values

Figure 10 shows the scatter plot of desired vs. obtained properties using top-ranked solutions and the corresponding rms errors are shown in Table 4. As can be seen from the figure, the scatter has increased compared to single-pass inverse inference shown in Fig. 8. This indicates uncertainty increasing as it gets propagated from one pass to the next. This is expected. To see why this is so, we can think of the probabilistic inference as a two step process. In the first step, given the outputs of pass 2, it infers its inputs. In the second step, it takes these inputs as outputs of pass 1 and infers its inputs. However, these intermediate inputs are not point estimates but distributions. These distributions have

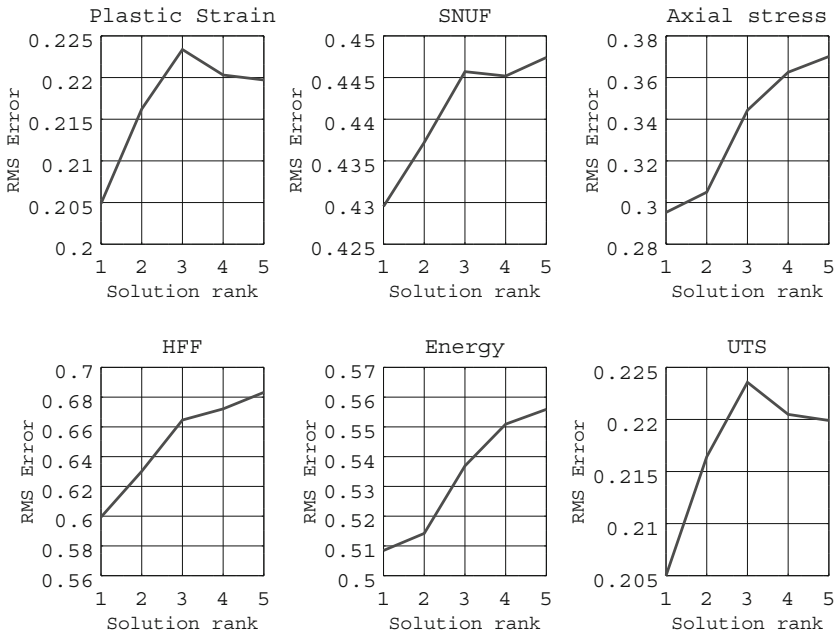

Fig. 9 RMS error for multiple solutions ranked by log-likelihood

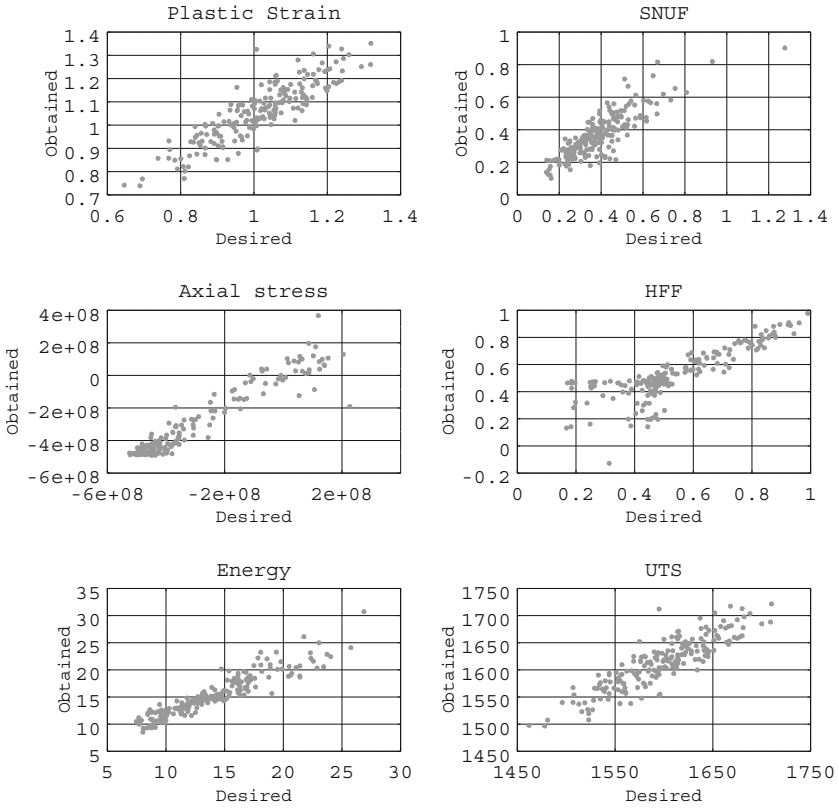

Fig. 10 Scatter plot of desired vs. obtained property values

multiple modes each with its own standard deviation. For example, for one of the test instances, we saw a standard deviation of 0.1 to 0.3 for top-ranked modes. Thus, in effect, the inference in the second step is over samples drawn from these intermediate distributions, rather than from point values, thus introducing more uncertainty. Even though uncertainty increases with the number of passes, the shape of the scatter plot shows that it still gives us a good starting point to conduct a more detailed, simulation-based search.

Comparison with Optimization Approach

It was shown in Section "Single-Pass Inverse Inference" that multiple solutions to the inverse problem can be extracted from the Bayesian network model. In particular, it was shown that up to 5 solutions could be extracted. Obtaining multiple solutions using optimization-based approach is very difficult and one needs to perform multiple optimizations starting at a number of initial points that cover the design space well. To illustrate this, we tried to get multiple solutions for a given set of desired wire properties, using optimization approach. We used the sequential quadratic programming implementation from Octave for this purpose. The same Bayesian network model is used as a forward model to compute the objective function of the optimizer. The search is guided by approximate numerical gradient computed using the forward model. We increased the number of starting points of the search until we get multiple solutions. The starting points were chosen using a Latin Hypercube sample of the design space. We observed that in a single-pass inverse inference, to get at least two different solutions, we needed 100 starting points. The time required (in minutes) on an Intel Xeon processor @ 2.4GHz with 2GB RAM for parallel tempering (first row) and optimization (second row) is shown in Table 5. It can be seen that the parallel tempering approach is faster than optimization for finding multiple solutions. Note that the Bayesian forward model used for computing objective function in the optimization approach is not the bottleneck because it uses junction tree algorithm and takes less than 3.5 s. Instead, most of the time is due to the iterations required and in each iteration the number of objective

Table 4 Accumulation of error from single pass to two-pass inverse inference

	Plastic strain	SNUF	Axial stress	HFF	Energy	UTS
Single pass	0.04	0.2554	0.1507	0.1912	0.2355	0.0488
Two pass	0.1790	0.4000	0.2524	0.5597	0.4522	0.1791

Table 5 Inverse inference time (in minutes) using parallel tempering(PT) and optimization (OPT) for finding one solution and multiple solutions

Method	Single pass		Two pass	
	One	Multiple	One	Multiple
PT	2.78	2.78 (5 solutions)	30.0	30.0 (5 solutions)
OPT	5.41	454 (2 solutions)	23.51	?

function evaluations required for calculating the gradient. For two pass, we were not able to get multiple solutions with up to 100 starting points. So, it was clear that it will take more than 2300 minutes (100 times the time for one optimization). We did not explore further as the computation time was getting unacceptably large. In general, it seems difficult to give a bound on the number of optimizations required to get multiple solutions.

We also performed an optimization search starting at the top five solutions given by parallel tempering. We were able to get 3 different solutions in about 50 min (as compared to 454 for 2 solutions in the case of pure optimization search). This indicates that Bayesian inverse inference can be used to provide good starting points around which to conduct a detailed simulations-based search. However, a point to note is that while it is significantly faster than the optimization-based approach, there is a large increase in running time of parallel tempering from single pass to two pass. This is largely due to the increase in number of temperatures and walkers (see Section "Composite Inverse Inference"). Also, as the model contains double the number of parameters, calculation of acceptance ratio in each sampling step requires double the time. There is a possibility of parallelizing the ensemble updates, which can substantially speed this up [28]. But we have not been able to try this out in our current experimentation as we could not get the required configuration of Emcee working.

Conclusion

In this paper, we proposed a conditional linear Gaussian Bayesian network-based approach for inverse inference over process chains. We showed how unit process models can be learned and we also discussed how they can be composed to get a composite model for inverse inference over a process chain. We conducted experiments on a wire-drawing process chain to validate the approach. Results look encouraging. While uncertainty increases with number of unit processes in the process chain, we believe it still gives us a good starting point around which to conduct a detailed simulation-based search. The increase in running time with increasing number of unit processes is largely attributable to the increase in the size of the temperature ladder and the number of walkers required. Exploring alternative methods for sampling multi-modal distributions is an important direction of future work. For example, the Repelling-Attracting Metropolis (RAM) algorithm [30] is a modified version of Metropolis-Hastings that makes a composite move—downhill followed by uphill—to overcome the difficulty of crossing low-probability regions.

Compliance with Ethical Standards

Conflict of interest The authors declare that have they no conflict of interest.

References

1. LeSar R (2009) Materials informatics: an emerging technology for materials development. Statist Anal Data Mining 1(6):372–374
2. Rajan K (2012) Materials informatics. Mater Today 15(11):470–471
3. Ramprasad R, Batra R, Pilania G, Mannodi-Kanakkithodi A, Chiho PY (2017) Kim machine learning in materials informatics: recent applications and prospects npj. Comput Mater 3(1):2017
4. Dimiduk DM, Holm EA, Niezgoda S (2018) Perspectives on the impact of machine learning, deep learning, and artificial intelligence on materials, processes, and structures engineering. Integr Mater Manuf Innov 1–16:08
5. Kondo R, Yamakawa S, Masuoka Y, Tajima S, Asahi R (2017) Microstructure recognition using convolutional neural networks for prediction of ionic conductivity in ceramics. Acta Mater 141:29–38
6. Cecen A, Dai H, Yabansu YC, Kalidindi SR, Le S (2018) Material structure-property linkages using three-dimensional convolutional neural networks. Acta Mater 146:76–84
7. DeCost BL, Francis T, Holm EA (2017) Exploring the microstructure manifold: image texture representations applied to ultrahigh carbon steel microstructures. Acta Mater 133:30–40
8. Sun K, Hong K, Yuan L, Shen Z, Ni X (2014) Inversion of functional graded materials elastic properties from ultrasonic lamb wave phase velocity data using genetic algorithm. J Nondestruct Eval 33(1):34–42
9. Atkinson S, Zabaras N (2018) Structured Bayesian Gaussian process latent variable model arXiv e-prints, arXiv:1805.08665
10. Tsilifis P, Bilionis I, Katsounaros I, Zabaras N (2016) Computationally efficient variational approximations for Bayesian inverse problems. J Verif Valid Uncert Quantif 1(3):031004–031004–13
11. Singh SK, Gautham BP, Goyal S, Joshi A, Gudadhe D (2009) Optimization of multi-pass wire drawing operation. Wire J Int 42(9):82–88
12. Pandita P, Bilionis I, Panchal J, Bp G, Joshi A, Zagade P (2017) Stochastic multi-objective optimization on a budget: application to multi-pass wire drawing with quantified uncertainties. Int J Uncertain Quantif 8:06
13. Sardeshmukh A, Reddy S, Gautham BP, Joshi A, Panchal JH (2017) A data science approach for analysis of multi-pass wire drawing. In: ASME International design engineering technical conferences & computers and information in engineering conference (IDETC/CIE 2017). Cleveland. ASME
14. Sardeshmukh A, Reddy S, Gautham BP, Joshi A (2017) Bayesian networks for inverse inference in manufacturing. In: 2017

16th IEEE International conference on machine learning and applications (ICMLA), pp 626–631
15. Koller D, Friedman N (2009) Probabilistic graphical models - principles and techniques. MIT Press, Cambridge
16. Hastie T, Tibshirani R, Friedman J (2001) The elements of statistical learning. Springer series in statistics. Springer New York Inc., New York
17. Murphy KP (2012) Machine learning: a probabilistic perspective. The MIT Press, Cambridge
18. Scutari M (2017) Bayesian network constraint-based structure learning algorithms: parallel and optimized implementations in the bnlearn R package. J Stat Softw 77(2):1–20
19. Brooks S, Gelman A, Jones G, Meng X-L (2011) Handbook of Markov chain Monte Carlo. CRC Press, Boca Raton
20. Celeux G, Hurn M, Robert CP (2000) Computational and inferential difficulties with mixture posterior distributions. J Am Stat Assoc 95(451):957–970
21. Geyer CJ (1991) Markov chain Monte Carlo maximum likelihood. In: Proceedings of the 23rd symposium on the interface, pp 156–163. Retrieved from the University of Minnesota Digital Conservancy, http://hdl.handle.net/11299/58440
22. Earl DJ, Deem MW (2005) Parallel tempering: theory, applications, and new perspectives. Phys Chem Chem Phys 7(23):3910–3916
23. Koller D, Pfeffer A (1997) Object-oriented Bayesian networks. In: Proceedings of the thirteenth conference on uncertainty in artificial intelligence, UAI'97. Morgan Kaufmann Publishers Inc, San Francisco, pp 302–313
24. Singh SK, Gautham BP, Goyal S, Joshi A, Gudadhe D (2007) Development of a virtual wire drawing tool for process analysis and optimisation. Wire J Int 40(10):72–78
25. Scutari M (2010) Learning Bayesian networks with the bnlearn R package. J Stat Softw 35(3):1–22
26. Murphy K (2001) The Bayes net toolbox for matlab. Comput Sci Statist 33:11
27. Salvatier J, Wiecki TV, Fonnesbeck C (2016) Probabilistic programming in Python using PyMC3. PeerJ Comput Sci, 2:e55 apr
28. Foreman-Mackey D, Hogg D, Lang D, Goodman J (2012) emcee: the MCMC Hammer
29. Raftery AE, Lewis SM (1995) The number of iterations, convergence diagnostics and generic metropolis algorithms. In: Gilks WR, Spiegelhalter DJ (eds) Practical Markov chain Monte Carlo. Chapman and Hall, pp 115–130
30. Tak H, Meng X-L, van Dyk DA (2017) A repelling-attracting metropolis algorithm for multimodality. J Comput Graph Stat 27(3):479–490