Proceedings of the 2023 Winter Simulation Conference
C. G. Corlu, S. R. Hunter, H. Lam, B. S. Onggo, J. Shortle, and B. Biller, eds.

CAUSAL DYNAMIC BAYESIAN NETWORKS FOR SIMULATION METAMODELING

Pracheta Amaranath
Peter J. Haas
David Jensen

Manning College of Information and Computer Sciences
University of Massachusetts Amherst
140 Governors Drive
Amherst, MA 01003, USA

Sam Witty

Basis Research Institute
315 W. 115th Street
Apt. 24
New York, NY 10026, USA

ABSTRACT

A traditional metamodel for a discrete-event simulation approximates a real-valued performance measure as a function of the input-parameter values. We introduce a novel class of metamodels based on *modular dynamic Bayesian networks* (MDBNs), a subclass of probabilistic graphical models which can be used to efficiently answer a rich class of *probabilistic and causal queries* (PCQs). Such queries represent the joint probability distribution of the system state at multiple time points, given observations of, and interventions on, other state variables and input parameters. This paper is a first demonstration of how the extensive theory and technology of causal graphical models can be used to enhance simulation metamodeling. We demonstrate this potential by showing how a single MDBN for an M/M/1 queue can be learned from simulation data and then be used to quickly and accurately answer a variety of PCQs, most of which are out-of-scope for existing metamodels.

1 INTRODUCTION

A simulation metamodel is broadly defined as an approximation of a simulation model intended to emulate the latter model's behavior (Kleijnen 1987). Metamodels avoid the need to run computationally expensive simulation experiments, facilitating rapid prediction, what-if experimentation, and optimization. Traditional metamodels for stochastic discrete-event simulations approximate a simulation model's behavior in a rather narrow sense. They approximate the statistical relationships between a set of input parameters and a single real-valued performance measure of interest (Barton 2020). For example, a metamodel of an M/M/1 queue might approximate quantities such as $E[\bar{L}_T]$, the expected time-average queue length over an interval $[0,T]$, or $E[L_t]$, the expected queue length at time t, or queue-length probability $P(L_t = 5)$, each as a function of the arrival rate λ and the service rate μ. This approach requires a separate metamodel for each performance measure under study. For example, estimating $P(L_t = 5)$ and $P(L_t = 6)$, or $P(L_t = 5)$ and $P(L_{t+h} = 6)$ for some $h > 0$, would require two different metamodels. This is true of many common types of metamodels (Chen et al. 2006), including polynomial regression, Gaussian processes (Salemi et al. 2019), regression trees, and methods based on artificial neural networks (Dunke and Nickel 2020). Separate metamodels for similar queries on the same system may be inconsistent with each other and require additional computational effort.

This paper presents a powerful, complementary approach to constructing simulation metamodels by applying tools and techniques developed by researchers in graphical models and causal inference. Specifically, we show how *modular dynamic Bayesian networks* (MDBNs), a subclass of probabilistic graphical models (Koller and Friedman 2009), can be used to efficiently estimate answers to a rich class of *probabilistic and causal queries* (PCQs) representing the joint probability distribution of the system state at multiple time points, given observations of, and interventions on, system states and simulation

979-8-3503-6966-3/23/$31.00 ©2023 IEEE

parameters at other time points. PCQs can also include "reverse inference" queries, which treat simulation parameters as Bayesian random variables and represent the probability distribution of the parameters given observations of system states. The results from such queries can be used, for example, to select the input parameters that would maximize the probability of achieving target system-state values at specific times.

For example, consider a simple M/M/1 queuing model that represents the arrival and service of customers at a store on a typical workday. Examples of PCQs that can be answered using a *single*, trained MDBN include:

- **PCQ1**: What is the distribution of the queue length at 3pm?
- **PCQ2**: What is the distribution of the queue length at 3pm given that we observe that 5 customers were in the queue at noon?
- **PCQ3**: What would be the joint distribution of the queue length at 3pm and at 4pm if we were to inject an additional 5 customers into the queue at noon?
- **PCQ4**: What would be the distribution of the queue length at 3pm if we were to double the service rate starting at noon?
- **PCQ5**: What value of arrival rate maximizes the probability that the queue length at 3pm is between 5 and 10 customers?

Accurate answers to these kinds of queries enable useful, efficient, fine-grained interrogation of the dynamic behavior of a simulation model that cannot be achieved via traditional simulation metamodels alone. Importantly, the described interventions need not be explicitly simulated while creating the training data for the MDBN. Note also that using the original simulation model to answer these queries would be time consuming, requiring multiple simulation runs. In contrast, answers to these queries can be quickly obtained from the MDBN via exact or approximate inference methods; there is a vast literature on inference techniques in graphical models that can be applied (Murphy et al. 1999; Jordan et al. 1999; Murphy 2002; Koller and Friedman 2009).

In this paper, we demonstrate the potential of MDBNs for metamodeling of discrete-event simulations, using a simple M/M/1 queue. We show how PCQ results from an MDBN metamodel can accurately approximate the ground truth results (as calculated analytically or via extensive simulation).

Related Work We are not the first to employ DBNs for metamodeling but, as far as we are aware, we are the first to use them to reason about the effects of interventions in the simulation model and for "inverse queries". The use of DBNs for simulation metamodeling was introduced in by Poropudas and Virtanen (2010) and followed up in subsequent papers; see Pousi et al. (2013) and references therein. The "probabilistic" DBN metamodels developed in the above literature are not modular; they cannot reason effectively about interventions on the simulation's states and instead only capture statistical dependencies. In addition, when modeling an M/M/1 queue, they fix the arrival and service rates to be constant throughout the time horizon, ruling out interventions and inverse queries.

Ouyang and Nelson (2017) use a combination of logistic regression and weighted least-squares regression over time to predict the time-varying probability of a single event of interest (e.g. "Is the queue network going to be blocked at some times $\{t_j\}$ having observed the state of the system at a previous time t_i?"). This approach is limited by the need to construct new metamodels for each such query as well as the need to specify basis functions can accurately represent the evolution of the simulation state over time.

Recurrent neural networks commonly used in time-series prediction problems can also be employed for simulation metamodeling, but existing approaches are limited by the restriction to non-time-varying input parameters or convergence issues when the network reaches a local minimum (Dimopoulos et al. 2000; Connor et al. 1994). Recent work by Cen and Haas (2022) uses graph neural network models with generative components to approximate distributions of a summary measure as well as time series of certain random variables associated with the simulation. These metamodels can specify conditions on the input distributions to generate output time series, which correspond to interventions on simulation parameters and

is closest to the methodology that we have explored in this paper. However, this metamodel cannot output the sample-path information required to answer PCQs and cannot handle interventions on the system state.

2 PROBABILISTIC AND CAUSAL QUERIES FOR DISCRETE-EVENT SIMULATIONS

We aim to create a simulation metamodel that can answer a broad set of probabilistic and causal queries (PCQs). We first describe PCQs for an M/M/1 queue, then define them for general stochastic simulations.

M/M/1 Queue Example. The input variables for an M/M/1 queue simulation are denoted by the tuple $\theta_t = (\lambda_t, \mu_t)$, which denote the arrival and service rate parameters at time t respectively. The output variable is L_t, which describes the queue length at time t. A *probabilistic and causal query* (PCQ) is a desired conditional probability distribution of the variables in the simulation model. The conditioned quantities are *point actions* or *interval actions* that apply to variables at specified time points or over specified time intervals. In addition, actions can be 'observational' (denoted by $=$) or 'interventional' (denoted by $\leftarrow$). For example, "$L_t = \ell$" means that the queue length is observed to be equal to ℓ at time τ and "$\lambda_{0:\tau} \leftarrow c$" means that the arrival rate is externally fixed at the value c over the time interval $[0,\tau]$. The PCQs given in Section 1 are abstractly represented as follows:

- **Q1:** $P(L_\tau \mid \lambda_{0:\tau} \leftarrow c_1, \mu_{0:\tau} \leftarrow c_2)$.
- **Q2:** $P(L_\tau \mid L_{\tau-h} = \ell, \lambda_{0:\tau} \leftarrow c_1, \mu_{0:\tau} \leftarrow c_2)$, where $0 < h < \tau$.
- **Q3:** $P(L_{\tau_1}, L_{\tau_2} \mid L_{\tau_1-h} \leftarrow L_{\tau_1-h} + 5, \lambda_{0:\tau_2} \leftarrow c_1, \mu_{0:\tau_2} \leftarrow c_2)$, where $0 < h < \tau_1 < \tau_2$.
- **Q4:** $P(L_\tau \mid \lambda_{0:\tau} \leftarrow c_1, \mu_{0:t} \leftarrow c_2, \mu_{t:\tau} \leftarrow 2c_2)$, where $0 < t < \tau$.
- **Q5:** $\text{argmax}_\lambda\, P(L_\tau \in [5..10] \mid \lambda)$.

Note that even the simple query **Q1** cannot be handled easily by traditional metamodels, because **Q1** is simultaneously estimating the probability that $L_\tau = 0$, $L_\tau = 1$, and so on. The query **Q5** is of interest if, for example, we can control the arrival rate λ by controlling admission to the queue. To simplify the notation, we treat λ as constant over time, but uncertain, with a prior distribution, and treat μ as known and constant over time. We denote by P_λ the distribution of $\{L_t\}_{t\geq 0}$ when λ is selected according to the prior distribution. To compute the answer to **Q5**, first compute the PCQ **Q5'** $= P(\lambda \mid L_\tau \in [5..10])$. By Bayes' Theorem,

$$P(L_\tau \in [5..10] \mid \lambda) = \alpha P(\lambda \mid L_\tau \in [5..10]), \tag{1}$$

where $\alpha = P_\lambda(L_\tau \in [5..10])/P_\lambda(\lambda)$. The value of λ that maximizes the right side of (1)—which we can compute from **Q5'**—will maximize the left side of (1), and hence solves **Q5**. A more elaborate version of the problem might consider a policy that uses one value of λ over $[0,\tau-h]$ and another over $[\tau-h,\tau]$.

As with **Q5** above, we can potentially use the results of PCQs to compute other quantities of interest. For example, we can easily compute moments, quantiles, and other summary statistics of the system state, e.g. $\mathrm{E}[L_\tau \mid \lambda_{0:\tau} = c_1, \mu_{0:\tau} = c_2]$ and $\mathrm{Corr}[L_{\tau_1}, L_{\tau_2}, \mid \lambda_{0:\tau} = c_1, \mu_{0:\tau} = c_2]$. As another example, we can approximate $\beta = \mathrm{E}[\bar{L}_\tau]$, where $\bar{L}_\tau = (1/\tau)\int_0^\tau L_t\, dt$, by setting $v_0 = 0$ and $v_n = \tau$, computing $P(L_{v_0}, L_{v_1}, \ldots, L_{v_n} \mid \lambda_{0:\tau} = c_1, \mu_{0:\tau} = c_2)$, and then computing $\hat{\beta} = \sum_{i=0}^{n-1} \mathrm{E}[L_{v_i} \mid \lambda_{0:\tau} = c_1, \mu_{0:\tau} = c_2](v_{i+1} - v_i)$; here $(v_0, v_1, \ldots, v_n)$ is a sufficiently fine sequence of time points over $[0,\tau]$.

In general, we define PCQs as follows: Let $\{X_t\}_{t\geq 0}$ be a continuous-time stochastic process having discrete state space $\mathscr{X} \subseteq \mathbb{Z}^l$ for some $l \geq 1$ and piecewise-constant sample paths, representing the output process of a discrete-event simulation model $\mathscr{M}$ with l state variables. We assume that the simulation model takes as input a vector $\theta = (\theta_1, \ldots, \theta_d)$ of d (≥ 1) *simulation model parameters*; we denote by $\Theta \subseteq \mathbb{R}^d$ the set of possible values of θ. We are often interested in queries involving intervention on the parameters at various time points and/or optimization over the parameters. In the former case, we want to represent parameters as time-varying functions. In the latter case, we want to represent the parameters as uncertain, as is usually done in Bayesian analysis (see above). We therefore consider the augmented process $\{Y_t\}_{t\geq 0}$ with probability distribution P and state space $\Upsilon = \mathscr{X} \times \Theta$, where $Y_t = (X_t, \theta_t)$. A PCQ is

a desired probability distribution of the form

$$\mathbf{Q} = P\big(\Pi_1(Y_{\tau_1}), \Pi_2(Y_{\tau_2}), \ldots, \Pi_n(Y_{\tau_n}) \mid \mathscr{A}_{t_1}, \mathscr{A}_{t_2}, \ldots, \mathscr{A}_{t_k}, \mathscr{A}_{u_1:u_2}, \mathscr{A}_{u_3:u_4}, \ldots, \mathscr{A}_{u_{m-1}:u_m}\big),$$

where $n \geq 1$, $k, m \geq 0$, $0 \leq t_1 < t_2 < \cdots < t_k$, $0 \leq \tau_1 < \tau_2 < \cdots < \tau_k$, and $u_i < u_{i+1}$ for $i \in [1..m-1]$. Here each $\Pi_i(y)$ is a projection function onto one or more of the $l+d$ coordinates of state y. Moreover, each $\mathscr{A}_{t_i}$ represents a point action of the form "$Y_{t_i} \in R_i$" for some $R_i \subset \Upsilon$, denoting an observation of the state at time t_i. Almost always, the definition of the set R_i imposes explicit constraints on only a small subset of components in the state vector, so that the "observation" is really only a partial observation of a small subset of the state variables. Alternatively, a point action can be of the form "$Y_{t_i} \leftarrow f_i(Y_{t_i})$", denoting an intervention that sets the state to $f_i(Y_{t_i})$ at time t_i, where $f_i : \Upsilon \mapsto \Upsilon$. In the causal graphical models literature, an intervention is also denoted by "$\mathrm{do}\big(Y_{t_i} = f_i(Y_{t_i})\big)$". In practice, the intervention function f_i usually updates only a small subset of the state components. Finally, each $\mathscr{A}_{u_i:u_j}$ represents an *interval action* that is defined as follows. Set $Y_{u_i:u_j} = \{Y_t\}_{u_i \leq t \leq u_j}$. An interval action can be of the form "$Y_{u_i:u_j} \in R_{i:j}$" for some $R_{i:j} \subset \Upsilon$, denoting a partial observation in which $Y_t \in R_{i:j}$ for every $t \in [u_i, u_j]$. Alternatively, an interval action can be of the form "$\Pi_{i:j}(Y_{u_i:u_j}) \leftarrow f_{i:j}(Y_{u_i})$", denoting an intervention where certain components of the state vector at time u_i are set to a constant value $f_{i:j}(Y_{u_i})$ over the interval $[u_i, u_j]$ where $f_{i:j} : \Upsilon \mapsto \Upsilon$. Set $T_{\mathbf{Q}} = \max(\tau_n, t_k, u_m)$, so that $T_{\mathbf{Q}}$ is the *time horizon* of the PCQ query, i.e., the largest time point referenced by the query. We also define the *time scale* $\delta_{\mathbf{Q}}$ of query $\mathbf{Q}$ as follows. Let $s_1, s_2, \ldots, s_r$ be the superposition of the τ_i's, t_i's, and u_i's with duplicate values removed, and set $s_0 = 0$. Then $\delta_{\mathrm{Q}} = \mathrm{median}(s_1 - s_0, s_2 - s_1, \ldots, s_r - s_{r-1})$.

We delineate several important subclasses of PCQs. If any of the actions are interventions, we call the PCQ an *interventional* query. Queries **Q1**–**Q4** are all interventional, and **Q4** is particularly noteworthy because it contains an intervention on the state variable L_τ. A query of type **Q5** is referred to as an *inverse query*. If the time horizon $T_{\mathbf{Q}}$ is greater than the maximum length T of the simulations used to train the MDBN metamodel, then we refer to $\mathbf{Q}$ as an *extrapolative query*.

We emphasize that what is desired is the *joint probability distribution* of $\Pi_1(Y_{\tau_1}), \ldots, \Pi_n(Y_{\tau_n})$. Our goal is to learn a metamodel $\tilde{\mathscr{M}}$ that can efficiently (approximately) answer PCQs of the above form.

3 MODULAR DYNAMIC BAYESIAN NETWORKS

In this section, we present a brief overview of modular dynamic Bayesian networks (MDBNs) and outline their key characteristics that render them appropriate for metamodelling.

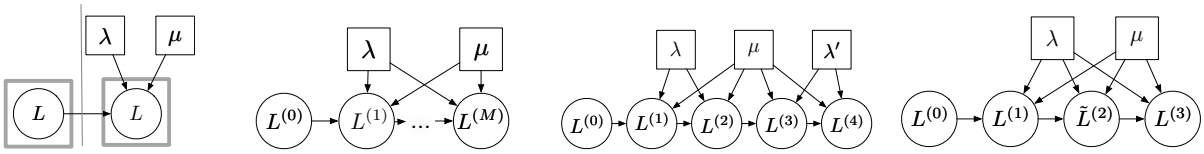
Bayesian Networks A Bayesian Network (BN) comprises a directed acyclic graph $G = (V, E)$ and a set $\mathscr{P}$ of associated conditional probability distributions (CPDs), where $|\mathscr{P}| = |V|$. Here V is a set of vertices (representing simulation state and input parameter variables in our setting), E is a set of edges encoding their statistical relationships, and $\mathscr{P}$ *compactly* encodes the joint probability distribution $P(V)$. The compactness is a result of the conditional independencies that arise from the structure of the graph G. Using these independencies, the full joint distribution can be written as a product of the conditional distributions: $P(V) = \prod_{V_i \in V} P(V_i \mid Pa(V_i))$ where $Pa(V_i)$ are the parents of V_i in the graph G. When the domain of all variables in V is discrete, the probability distribution $P(V_i \mid Pa(V_i))$ is a conditional probability table that stores the probability of each V_i value for each possible combination of parent values. BNs are used to efficiently compute posterior marginal and/or conditional probabilities of a set $V' \subseteq V$ of the variables given observations on the state of other variables $V \setminus V'$; this task is known as *inference*.

Often, the edges of a BN only encode *statistical associations* between its variables. However, if the edges signify direct *causation*, BNs can be used to reason about effects of interventions (provided a few other assumptions also hold). For example, a conditional probability $P(V_i \mid V_j = v_j)$ represents the probability of V_i having *observed* $V_j = v_j$. We may, however, be interested in computing *interventional distributions* of the form $P(V_i \mid V_j \leftarrow v_j)$ where V_j is *set* to a value v_j by external manipulation. The assumptions that need to hold are the *causal Markov condition (CMC)*, *minimality*, and *modularity*. In simple terms, the CMC states that every variable $V_i \in V$ is conditionally independent of its non-descendants given its parents. The minimality assumption ensures that no subgraph of G over V satisfies the CMC. Additionally, we make

the assumption that the BN is *modular*: interventions on any variable $V_i \in V$ do not affect the CPD of any $V \setminus \{V_i\}$. The modularity property ensures that the mechanism underlying the direct causation in the edges of the graph structure G remains invariant upon intervention. It allows us to intervene on a subset of variables in V and observe the effect of those interventions had everything else in the BN stayed the same.

To answer an interventional query, we apply well-known results to translate the query into an equivalent query in terms of only marginalization and conditioning, so that we can apply standard inference algorithms for probabilistic BNs. One set of such syntactical transformations is the *backdoor criterion* (Shpitser et al. 2010). The backdoor criterion identifies a set of *adjustment variables* Z for the estimation of an interventional distribution $P(Y|X \leftarrow expr)$ such that, conditioning upon all adjustment variables Z, an intervention $X \leftarrow expr$ is equivalent to the observation $X = expr$. If the set of all interventions in a conditional distribution can be replaced by observations, it is deemed *identifiable*. If a variable has no parents in the graph structure, then interventions are equivalent to observations because the set of adjustment variables Z is empty in this case. We use this rule extensively in simulation metamodeling when intervening on the input parameters (which generally have no parents).

Dynamic Bayesian Networks So far, we have described models that encode statistical and causal relationships between variables measured at a fixed time. In many cases, we are interested in reasoning about system states evolving over time. BNs can be extended to model dynamic systems using the formalism of *dynamic Bayesian networks* (DBNs) (Murphy 2002). Informally a DBN comprises "snapshots" of the system taken at $M+1$ equally-spaced time points $\delta_0, \delta_1, \ldots, \delta_M$, where we take $\delta_0 = 0$ without loss of generality. We refer to these snapshots as *time slices* and denote by δ the time between two consecutive snapshots, also referred to as the *sampling interval*. Thus $\delta_j = j \cdot \delta$ for $j \in [0..M]$. Denote by $V_i^{(j)}$ the value of variable V_i in the jth time slice (observed at time δ_j). Effectively, a DBN is a compact representation of the trajectory of its variables at these discrete time instances. It can contain "intra-slice" dependencies within a time slice as well as "inter-slice" dependencies between time slices.



(a) MDBN Plate model (b) Ground network $\mathbb{G}$ (c) $\lambda_{\delta_0:\delta_3} \leftarrow c_1, \lambda_{\delta_3:\delta_4} \leftarrow c_2$ (d) $\tilde{L}^{(2)} \leftarrow f(L^{(2)})$

Figure 1: MDBN for the M/M/1 queue, with examples of interventions on the simulation input parameters and the queue length.

A DBN is a pair of Bayesian networks $(B_0, B_\rightarrow)$ where B_0 specifies the distribution of variables at time 0 and $B_\rightarrow$ represents the distribution of the variables across any two adjacent time slices, i.e. transition probabilities. B_0 is also referred to as the "prior BN" and $B_\rightarrow$ is referred to as "two time-slice Bayesian Network" or 2TBN for short. Although DBNs model dynamic systems, the conditional distributions of the 2TBN remain constant throughout. We call such networks *time-homogeneous*.

Plate models provide a concise visual representation of DBNs (e.g., Figure 1a). The subgraph of variables and edges within a plate (thick gray square) represent the intra-time-step dependencies of the DBN. This subgraph can be instantiated multiple times within the model, and the conditional probability distributions within any two instantiations are identical. Any variable outside the plate is only instantiated once and is connected to all instantiations of its child plate variables. For any given time horizon $T_\mathbf{Q} \geq 0$, the joint distribution of all variables in the DBN over the interval $[0, T_\mathbf{Q}]$ is defined via an unrolled *ground network* where the structure and CPDs of $V^{(0)}$ are the same as those for $V \in B_0$ and the structure and CPDs of $V^{(j)}$ for $j > 0$ are the same as those for $V \in B_\rightarrow$; see Koller and Friedman (2009) for further details. Figure 1b shows the ground network corresponding to the plate model in Figure 1a. The ground network is a BN representing the evolution of the simulation states over all times of interest. We use the notation

$\mathbb{G} = (\mathbb{V}, \mathbb{E})$ for the graph structure, set of variables, and edges in the ground network. The corresponding conditional probability distributions are denoted by $\mathbb{P}$. When the previously stated assumptions for causal manipulation in BNs also hold for B_0 and $B_{\rightarrow}$, and the semantics of interventions for dynamic variables are clearly specified, the DBN can also be used to infer interventional distributions. We call such DBNs *modular DBNs* (MDBNs). Learning a complete MDBN from data involves two components: determining its structure and estimating the set of conditional probability distributions (CPDs) for B_0 and $B_{\rightarrow}$.

MDBNs for simulation metamodeling In this paper, we impose some simplifying assumptions on the general MDBN framework, consistent with our application to discrete-event simulation models. First, we assume that the DBN variables are discrete and their domain is limited to the values that appear in the simulation training data. (We therefore only consider a finite set Θ_λ and Θ_μ of possible values for λ and μ, respectively, in the M/M/1 model.) We also assume a known structure (set of edges $\mathbb{E}$ for the DBN), as it can be determined using expert knowledge from the simulationist. Next, we assume for simplicity that each time τ_i in a PCQ coincides with some sampling time δ_j, and similarly for each time t_i and u_i. This does not entail a great loss of generality, especially if δ is relatively small. For example, if a query $\mathbf{Q}$ is supposed to return the (possibly conditional) distribution of L_{τ_i}, where τ_i lies between δ_j and δ_{j+1}, then we would estimate the distributions of L_{δ_j} and $L_{\delta_{j+1}}$ and interpolate. As discussed below, the sampling interval δ is determined by the time scale of the query, or set of queries, of interest. Finally, we assume that all variables of interest are included in the MDBN and there are no unobserved variables that may impact the state of the system (as often happens with causal inference in purely observational settings).

Inference using MDBNs We now describe how to estimate an answer to a PCQ using an MDBN. Inferring marginal and/or conditional probability distributions using the ground network $\mathbb{G}$ and the associated set of CPDs $\mathbb{P}$ is a well-studied problem in the probabilistic graphical models community. When exact inference is tractable, algorithms such as lazy propagation, variable elimination, or message passing may be used. Otherwise, approximate inference algorithms—e.g., loopy belief propagation, MCMC, variational inference—are available. For this paper, we treat the process of answering a purely *probabilistic query* (i.e. a query with no intervention operators) as a black box. We represent this process by the function INFERENCE($\mathbf{Q}, \mathbb{G}, \mathbb{P}$), which takes in a query $\mathbf{Q}$, a ground network $\mathbb{G}$, and an associated set of CPDs $\mathbb{P}$ and returns the desired probability distribution using any of the algorithms mentioned above. In general, we need a set of transformations (such as the backdoor criterion) that can convert interventional queries to probabilistic queries and then utilize the INFERENCE($\mathbf{Q}, \mathbb{G}, \mathbb{P}$) function to answer all types of PCQs. In this paper, two distinct types of BN transformations are used to implement interventions: (1) Modifications of the graph structure by adding new variables to the ground network $\mathbb{G}$, and (2) Modifications to the CPDs $\mathbb{P}$.

In detail, the inference process is as follows. Consider a *base* ground network $\mathbb{G}$ that corresponds to the simulation model with input parameters unchanging over time but possibly uncertain. We then transform the graph and CPDs to implement any interventions on input parameters and system states using the TRANSFORM($\mathbf{Q}, \mathbb{G}, \mathbb{P}$) function described below. which applies rules to convert intervention actions $\leftarrow$ to observation actions $=$. We can then perform standard inference on the resulting probabilistic MDBN. The overall procedure is given as Algorithm 1.

Algorithm 1 Compute PCQs

Input: PCQ $\mathbf{Q} = P(\mathscr{E} \mid \mathscr{A}_1, \mathscr{A}_2 \cdots)$; Base MDBN ground network $\mathbb{G}$; Base CPDs $\mathbb{P}$

1: **Q'**, $\mathbb{G}', \mathbb{P}'$ = TRANSFORM($\mathbf{Q}, \mathbb{G}, \mathbb{P}$).
2: Return INFERENCE(**Q'**, $\mathbb{G}', \mathbb{P}'$)

The function TRANSFORM($\mathbf{Q}, \mathbb{G}, \mathbb{P}$) is given as Algorithm 2. For ease of presentation, we assume that the parameter vector θ and the state vectors $X^{(i)}$ are all one-dimensional, and restrict attention to interval interventions on θ and interventions on $X^{(i)}$ that either set the value of $X^{(i)}$ to a constant c or increment $X^{(i)}$ by c (where $c \in \Re$). If θ is assigned the value c' over the interval $[\delta_i, \delta_j]$ (line 4), then, if $\delta_i = 0$, the intervention operator is turned into an observation operator in $\mathbf{Q}$ (line 5) and nothing else is

modified. If $\delta_i > 0$, a new variable θ' is created, with edges going to appropriate time slices, replacing the corresponding edges from θ; Figure 1c is an example. Interventions on state variables modify the CPDs. If the intervention $X_{\delta_i} \leftarrow X_{\delta_i} + c$ can result in a value that lies outside the pre-specified state space $\mathscr{X}$, we modify the definition of the state space such that the first and last values subsume all values below or above them, respectively.

Algorithm 2 TRANSFORM($\mathbf{Q}, \mathbb{G}, \mathbb{P}$)

Input: PCQ: $\mathbf{Q} = P(\mathscr{E} \mid \mathscr{A}_1, \mathscr{A}_2 \cdots)$; Base MDBN ground network: $\mathbb{G}$; Base CPDs: $\mathbb{P}$

Q' $= \mathbf{Q}$, $\mathbb{V}' = \mathbb{V}$, $\mathbb{E}' = \mathbb{E}$, $\mathbb{P}' = \mathbb{P}$
for all $\mathscr{A} \in \mathbf{Q}$ that have interventional operators **do**
  Q'$.pop(\mathscr{A})$ ▷ Remove the intervention action from **Q'**
  if $\mathscr{A} =$ '$\theta_{\delta_i:\delta_j} \leftarrow c$' **then** ▷ Interventions on simulation input parameters
    if $\delta_i = 0$ **then**
      Q' $= P(\mathscr{E} | \theta_{0,\delta_j} = c, \cdots)$
    else
      Let θ' be a random variable such that $\mathbb{P}(\theta' = c) = 1.0$.
      $\mathbb{V}' = \mathbb{V}' \cup \{\theta'\}$
      $\mathbb{E}' = \mathbb{E}' \cup \{(\theta', X^{(i)}), (\theta', X^{(i+1)}), \ldots, (\theta', X^{(j)})\} \setminus \{(\theta, X^{(i)}), (\theta, X^{(i+1)}), \ldots, (\theta, X^{(j)})\}$
      Q' $= P(\mathscr{E} | \theta'_{\delta_i:\delta_j} = c, \cdots)$ ▷ Convert interventional operator to observational operator
  if $\mathscr{A} =$'$X_{\delta_i} \leftarrow f_{ij}(X_{\delta_i})$' **then** ▷ Interventions on simulation state variables
    if $X_{\delta_i} \leftarrow c$ **then** ▷ If the intervention sets X_{δ_i} to a fixed value
      $\mathbb{P}'(X^{(i)} = c) = 1.0$
      $\mathbb{P}'(X^{(i)} \neq c) = 0.0$
    if $X_{\delta_i} \leftarrow X_{\delta_i} + c$ **then** ▷ If the intervention increases or decreases the current value of X_{δ_i}
      for all $x \in \mathscr{X}$ **do**
        $\mathbb{P}'(X^{(i)} = x) = \mathbb{P}(X^{(i)} = x - c)$ ▷ Shift $\mathbb{P}$; If $x - c \notin \mathscr{X}$, see caveat in text
return **Q'**$, \mathbb{G}', \mathbb{P}'$

4 CREATING AND USING AN MDBN METAMODEL OF AN M/M/1 QUEUE

In this section, we describe the process of constructing a modular DBN metamodel of an M/M/1 queue. We first specify a structure for the MDBN that includes all simulation states of interest. We then estimate the CPDs using data collected by running experiments on the simulation model. We can then query the MDBN (using Algorithm 1) to estimate PCQs of interest. We describe these steps in detail below.

Representation: Specifying the structure of the MDBN We model three system state variables: arrival rate (λ), service rate (μ), and queue length (L) for a desired number of time slices M. Simulation input parameters λ and μ together determine the queue length at each instant and are thus parents of L in the graphical model structure. These parameters are determined by the simulationist exogenously to the model, so that λ and μ have no causal parents in this graph. We show the plate model in Figure 1a. We use thin, black squares to denote simulation input parameters which have known, deterministic values for each simulation run, but can be treated as uncertain random variables for reverse inference queries. The queue length is a random variable, denoted by thin, black circles. Because of the continuous-time Markov property of the M/M/1 queue-length process, the distribution of each $L^{(j+1)}$ is completely determined by the values of $L^{(j)}$, λ, and μ. In the ground network, this dependency is denoted by the set of edges $(\lambda, L^{(j+1)}), (\mu, L^{(j+1)}), (L^{(j)}, L^{(j+1)})$ for $j \in [0..M]$. By unrolling the plate model and copying the CPDs of $B_{\rightarrow}$ for each time slice, we obtain the ground network $\mathbb{G}$ as shown in Figure 1b.

Learning the conditional probabilities of the MDBN For our MDBN metamodel of the M/M/1 queue, we need to specify the CPDs: $\mathbb{P}_1 = P(\lambda), \mathbb{P}_2 = P(\mu), \mathbb{P}_3 = P(L^{(0)})$ and $\mathbb{P}_4 = P(L^{(j+1)} | \lambda, \mu, L^{(j)})$. Note that $\mathbb{P}_1$–$\mathbb{P}_3$ are manually set by the analyst, but $\mathbb{P}_4$ needs to be estimated from data. (Of course, we can actually compute the conditional probabilities exactly for the M/M/1 queue, but we will pretend that they are not available, as would usually be the case for a more complex model.) We estimate $\mathbb{P}_4$ by first executing

multiple simulations over the time interval $[0,T]$ for some $T > 0$, using various values of λ, μ, and $L^{(0)}$. In this work, we use a full factorial design with an equal number of runs for all possible value combinations; we do this to enable accurate estimates for a large class of PCQs, which requires an adequate amount of data for each query that is posed. The simulation data for a given run then contains records for each simulated event: arrival rate, service rate, time of the event, type of event (arrival/departure), and queue length. For a specific sequence of MDBN sampling times $\delta_0, \ldots, \delta_M$, we then sample the training data as follows. For a given simulation run, denote by t_j^* the time of the jth simulated event (i.e., an arrival or departure) for $j \geq 1$. For a given time δ_i, we set the sampled value of $L^{(i)}$ equal to $L_{r(i)}$, where $r(i) = \max\{t_j^* : t_j^* \leq \delta_i\}$, and similarly for any time-varying simulation parameters. Finally, using the sampled data, we compute an estimate $\hat{\mathbb{P}}_4$ of $\mathbb{P}_4$. In this paper, we use maximum likelihood estimators for each conditional probability. Note that, because the queue-length process is time homogeneous, our MDBN is time homogeneous, and so we use $\hat{\mathbb{P}}_4$ in every time slice.

Computing PCQs using the MDBN We use Algorithms 1 and 2. The base ground network for the M/M/1 queue is shown in Figure 1b with corresponding CPD set $\mathbb{P}$. We assume that $\mathbb{P}_1(\lambda) = U(\Theta_\lambda)$, $\mathbb{P}_2(\mu) = U(\Theta_\mu)$, and $\mathbb{P}_3(L^{(0)}) = U([0..c])$ for some constant c, where $U(A)$ denotes a uniform distribution such that each element of the set A is equally likely. Figure 1c illustrates the network transformation when the arrival is changed from c_1 to c_2 at time δ_3. Figure 1d shows an intervention on the queue length at time δ_2: the ground network $\mathbb{G}$ does not change but the notation $\tilde{L}^{(2)}$ indicates that the CPD for $L^{(2)}$ has been modified.

5 EXPERIMENTS: METAMODEL DESIGN PARAMETERS AND TRAINING DATA

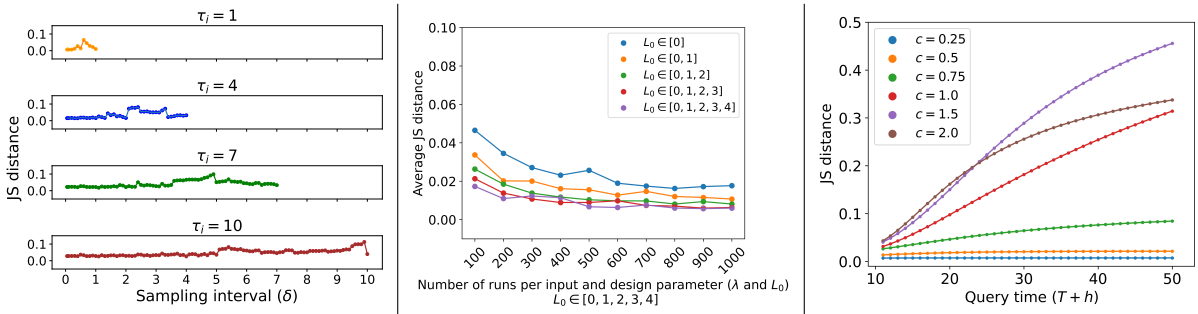
Learning an MDBN metamodel from data requires selection of the MDBN sampling interval δ and the number of simulation runs N that are used to generate the training data. Below we empirically investigate how varying these data collection parameters affects metamodel accuracy for a set of PCQs. We find that when details of the query workload W are known, such as the time horizon and time scale of the set of all PCQs, it is possible to choose values for δ and N to achieve high accuracy.

Sampling interval δ Recall that the variables at the jth time slice represent a snapshot of the system at time $j \cdot \delta$; e.g., Figure 1b depicts a DBN that models the M/M/1 queue for $M+1$ time slices. One approach chooses δ as the smallest time interval between any two events in the simulation training data. However, this may result in a huge number of time slices in the DBN, leading to intractable inference. An alternative approach sets $\delta = \rho \times \min_{\mathbf{Q} \in W} \delta_{\mathbf{Q}}$, where $\delta_{\mathbf{Q}}$ is the time scale for query $\mathbf{Q}$, as defined in Section 2, and $\rho < 1$ is a small fraction.

We evaluate the error for the latter approach on $\mathbf{Q}^1 : P(L_{\tau_i} \mid \lambda_{0:\tau_i} \leftarrow c, \mu_{0:\tau_i} \leftarrow 1.0, L_0 \leftarrow 0)$ empirically, for different choices of c and δ. We measure error as the Jensen-Shannon distance (JSD) between the inferred probability distribution and the ground-truth distribution (computed analytically). JSD lies between 0 and 1, with a small JSD value indicating that the two distributions are similar (low error). We run the simulation model for a total of $N = 3000$ iterations divided equally among arrival rates $c \in \{0.25, 0.5, 0.75, 1.0, 1.5, 2.0\}$ and a service rate of $\mu = 1.0$. We use the MDBN to estimate $\mathbf{Q}^1$ for 10 time instances $\tau_i \in [1.0, 2.0, \ldots, 10.0]$ and repeat the experiment for various sampling intervals $\delta \in [0.05, \tau_i]$. We show the results for $c = 0.5$, $\tau_i \in [1.0, 4.0, 7.0, 10.0]$ in Figure 2a, where each subplot explores the variation in JS distance for a combination of τ_i and δ.

The experiments show that when sampling interval δ is either equal to or a fraction of the time scale $\delta_{\mathbf{Q}}$ of the query, the error is low. While Figure 2a only shows the results of experiments for a single arrival rate $c = 0.5$ and a few values of τ_i, we found a similar trend for all values of c and τ_i. The plots suggest a tradeoff between computational efficiency and approximation error. As δ decreases, we sample the simulation data more frequently and the variation of queue length over time is captured with high accuracy, so that the JSD decreases. However, this is at the cost of decreasing computational efficiency.

We conclude that despite constructing a discrete-time model of a continuous-time process, when δ is a small fraction of the time scale of the query, the MDBN is able to emulate the relationship between the



(a) JSD of $\mathbf{Q}^1$ with $c = 0.5$ for $\delta \in [0.05, \tau_i]$ and various query times τ_i.

(b) JSD of $\mathbf{Q}^2$ for $N \in [100, 1000]$; Averaged over multiple (τ_i, c) values.

(c) JSD of $\mathbf{Q}^3$ for $T = 10.0, h \in [1..40]$ and various arrival rates.

Figure 2: Effect of δ and N on model accuracy, measured via Jensen-Shannon distance from ground truth, for probabilistic queries with time scale $\delta_{\mathbf{Q}} = 1.0$. Metamodel accuracy is high when $\delta = \rho\delta_{\mathbf{Q}}$, for $\rho < 1$. MDBNs can infer extrapolative queries with low error for stable queues and for unstable queues with extrapolated times close to the simulation time horizon T.

queue length and the simulator inputs with high accuracy. This conclusion is also supported by experiments on more complex PCQs involving multiple interventions as we will see in Section 6. For all further experiments, we use a time scale $\delta_{\mathbf{Q}} = 1.0$ for all PCQs and a sampling interval of $\delta = 0.5$.

Sample size N Increasing N leads to more accurate estimates of the CPDs. However, performing simulations can be costly, so we aim to determine the smallest N necessary for low error across a number of PCQs. Prior work on BN metamodels has provided analytical methods to obtain the minimum number of runs for a stipulated accuracy (Poropudas et al. 2011). However, these methods calculate the number of runs separately for each value of a single probability distribution—e.g. for $P(L_\tau = 1)$ and $P(L_\tau = 2)$—and require an estimate of the least likely combination of the input parameters for all workloads. Because our goal is a metamodel that accurately estimates the answers to multiple PCQs in a relatively large state space, we use simple experiments to determine an effective value of N for PCQs of interest. We defer the theoretical investigation of an optimal value to future research.

We evaluate the error for a query $\mathbf{Q}^2$: $P(L_{\tau_i} \mid \lambda_{0:\tau_i} \leftarrow c, \mu_{0:\tau_i} \leftarrow 1.0, L_0 \leftarrow 0)$ as the sample size N varies from $[100, 1000]$. As before, N represents the total number of simulation runs for all possible values of the arrival rate, with an equal number of runs for each $c \in \{0.25, 0.5, 0.75, 1.0, 1.5, 2.0\}$. For a given value of N, we compute the average JSD (error) over combinations of $\tau_i \in [1.0, \ldots, 10.0]$ and c. As expected, the error decreases as the sample size increases (denoted by the blue curve in Figure 2b).

Training data Additionally, we examined the CPDs for the MDBN trained with simulation data where the initial queue length L_0 was always set to 0. We found that the CPDs were more accurate for smaller values of the queue length than for larger ones, i.e., probabilities for larger queue lengths are harder to estimate when $L_0 = 0$ because they are rarely seen in the training data. To improve these estimates, we trained the MDBN with additional data by varying L_0 between 0 and 4. In Figure 2b, we plot the error for $\mathbf{Q}^2$ as before, but with the training data having different value ranges for L_0. As the maximum initial queue length increases, the average error decreases, underlining the importance of ensuring that the training data supports accurate estimation of the CPDs. Henceforth, we use $N = 3000$ in our experiments, corresponding to 100 runs for each value of the 30 possible (λ, L_0) combinations.

6 EXPERIMENTS: M/M/1 METAMODEL ACCURACY FOR PCQS

Using the values for δ and N obtained in the previous section, we wish to evaluate the effectiveness of the MDBN metamodel over a broad range of PCQs. Specifically, we want to answer the following research

questions: (1) Does the MDBN consistently produce low error for all PCQs? (2) Can it be applied to answer PCQs for interventions on the simulation state that were not observed in the training data?

We consider three different types of queries. First, extrapolative queries where we infer the queue-length distribution for time instances beyond the time horizon T of the simulation data. Second, interventional queries where the arrival rate and/or the queue length is set to a different value partway through the simulation run. Lastly inverse queries, where we reason about possible causes of observed effects. For each type of PCQ, we compare the error of the inferred distribution to the ground truth distribution obtained using a closed-form expression wherever applicable or by modifying the simulation model where necessary and performing a large number of simulation replications. We have $L_0 \leftarrow 0$ in all queries.

For all experiments, we used an Intel Xeon Gold 6126 CPU with 4 GB of RAM to run the simulation model as well as the MDBN. The simulation model is written in Python 3.8 and the MDBN is constructed using the open source library pyAgrum (Ducamp et al. 2020). We use lazy propagation (Madsen and Jensen 1999)—an exact inference algorithm—for all the queries that follow.

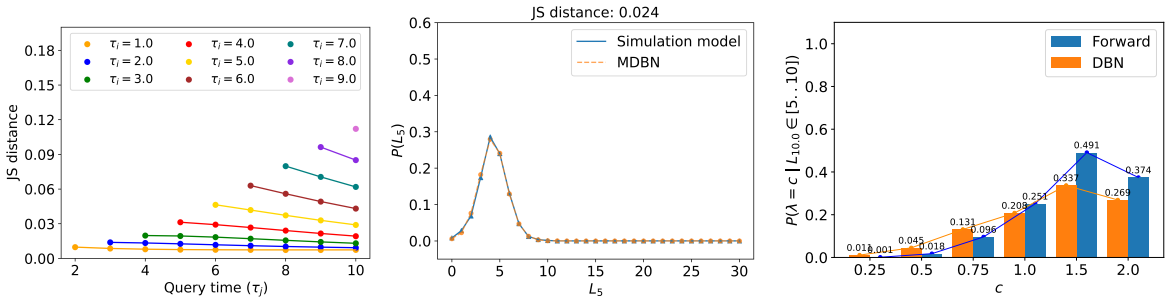
Extrapolative Queries Extrapolative queries infer the probability distribution of queue length for time $\tau_i > T$, where T is the time horizon of the simulations in the training data. We use the MDBN to answer the query $\mathbf{Q}^3$: $P(L_{T+h} \mid \lambda_{0:T+h} \leftarrow c, \mu_{0:T+h} \leftarrow 1.0)$ for $T = 10$ and $h \in [0..40]$ and compare the estimates to the closed-form expressions. We repeat this experiment for six different values of $c = \{0.25, 0.5, 0.75, 1.0, 1.5, 2.0\}$. We find (Figure 2c) that for stable queues ($c < 1.0$) the error is consistently low for all values of h. However, for unstable queues and larger values of h, the error increases because the unstable behavior of the queue results in queue lengths not observed in the training data.

Interventional Queries We illustrate the effectiveness of the MDBN in inferring the distribution of the queue length under interventions on the queue length and/or the arrival rate. Consider the query $\mathbf{Q}^4$: $P(L_{\tau_j} \mid \lambda_{0:\tau_i} \leftarrow c_1, \lambda_{\tau_i:\tau_j} \leftarrow c_2, \mu_{0:\tau_j} \leftarrow 1.0)$ for $\tau_i \leq \tau_j \leq T$. Thus there is an intervention at time τ_i and the query asks for the queue-length distribution at time τ_j. We use the values $c_1 = 1.0$, $c_2 = 0.25$, $\tau_i < \tau_j \in [1.0, \ldots, 10.0]$ and, for these experiments, $\delta = 0.5, N = 3000$, and $L_0 \in [0..4]$. Figure 3a compares the inferred and exact distributions. The overall JSD values are low despite the rather large drop in the arrival rate. Note that the error at query time τ_j gradually decreases as $\tau_j - \tau_i$ increases. Errors are more noticeable for larger values of τ_i, which correspond to larger queue sizes.

Next, we consider interventions on the queue length L_{τ_i}. Note that such interventions are typically not possible (or at best time-consuming) without modifying the internals of the simulation model. We modify the M/M/1 queuing model to allow changes to the queue length at arbitrary instances of time. Consider $\mathbf{Q}^5$: $P(L_5 | L_3 = 1, L_4 \leftarrow L_4 + 4, \lambda_{0:5} \leftarrow 0.5, \mu_{0:5} \leftarrow 1.0)$. This query denotes the distribution of the queue length at time 5 having observed the queue length at time 3 equal to 1 and intervened to add 4 customers to the queue at time 4. The arrival and service rates are constant throughout. In the simulation model, we increase the queue length by 4 customers at time 4 and compute the probability distribution of the queue length at time 5 while only considering the simulation runs where we observed $L_3 = 1$. For the MDBN, this can be inferred with a single query. The simulated and inferred distributions are shown in Figure 3b and are seen to be quite close.

Both experiments above demonstrate the utility of MDBNs to answer queries about interventions that have not been observed in the training data. This is a key benefit of the modularity of the DBN. We can think of the DBN as composed of homogeneous modules whose behavior can be learned by observing training data where the queue length for different arrival and/or service rates are recorded. The memoryless property of the exponential distributions ensure that future arrivals are independent of the number of arrivals in the past given the present. By exploiting both of the properties above, we can compute any interventional query without additional training data.

Inverse Queries We demonstrate how the MDBN metamodel can aid in simulation optimization. Specifically, we can find the value of λ that maximizes the probability of being in a target set of states at a specified time, without having to evaluate each possibility by running simulations. In detail, consider the query **Q5** in Section 2: $\max_c P(L_{10.0} \in [5..10] \mid \lambda = c)$, where $c \in \Theta_\lambda$. As shown in Equation (1), the



(a) Intervention on $\lambda : 1.0 \rightarrow 0.25$. (b) Intervention: $L_4 \leftarrow L_4 + 4$. (c) Optimizing λ to control $L_{10.0}$.

Figure 3: Demonstration of the MDBN for computing interventional and optimization queries.

value of c that maximizes $P(\lambda = c \mid L_{10.0} = [5..10])$ is the solution to **Q5**. We compute this distribution for $c \in \Theta_\lambda = \{0.25, 0.5, 0.75, 1.0, 1.5, 2.0\}$. The "forward" comparison computes the queue length at time 10.0 using roughly 5,000 simulation runs for each $c \in \Theta_\lambda$. The results are included in Figure 3c. Even though there are some disparities between the exact simulation and the MDBN metamodel, the metamodel selects the optimal value of $\lambda = 1.5$ without the need for expensive simulation runs.

7 CONCLUSIONS AND FUTURE WORK

We have demonstrated how a particular type of probabilistic graphical model (an MDBN) can serve as a metamodel for a particular type of simulation model (an M/M/1 queue). However, our larger aim is to provide an entry point to the rich and highly expressive family of probabilistic graphical models (PGMs), along with its associated body of theory and technology. As we show above, this family of models has the potential to greatly expand the scope of simulation metamodels by allowing construction of a single model capable of efficiently estimating answers to a wide variety of useful queries, including some that are difficult or impossible to answer with the original simulation model. The metamodeling capabilities of PGMs that we demonstrate here can and should be extended in a variety of ways to exploit existing PGM capabilities. These include PGMs with more complex dependencies among non-adjacent time steps (Xuan Vinh et al. 2012), PGMs with temporal dependencies that vary with time (Husmeier et al. 2010), and PGMs that represent time-varying relationships among model components (Marazopoulou et al. 2015). We plan to extend our approach to handle complex interventions in complex Markovian and non-Markovian systems; strategies for the latter include approximation by Markovian systems (e.g., using phase-type distributions) and incorporating simulation clock readings for events into the state vector.

REFERENCES

Barton, R. R. 2020. "Tutorial: Metamodeling for simulation". In *Proceedings of the 2020 Winter Simulation Conference*, edited by K.-H. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesing, 1102–1116. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Cen, W., and P. J. Haas. 2022. "Enhanced Simulation Metamodeling via Graph and Generative Neural Networks". In *Proceedings of the 2022 Winter Simulation Conference*, edited by B. Feng, G. Pedrielli, Y. Peng, S. Shashaani, E. Song, C. Corlu, L. Lee, E. Chew, T. Roeder, and P. Lendermann, 2748–2759. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Chen, V. C., K.-L. Tsui, R. R. Barton, and M. Meckesheimer. 2006. "A Review on Design, Modeling and Applications of Computer Experiments". *IIE Transactions* 38(4):273–291.

Connor, J. T., R. D. Martin, and L. E. Atlas. 1994. "Recurrent Neural Networks and Robust Time Series Prediction". *IEEE Transactions on Neural Networks* 5(2):240–254.

Dimopoulos, K., C. Kambhampati, and R. Craddock. 2000. "Efficient Recurrent Neural Network Training Incorporating A Priori Knowledge". *Mathematics and Computers in Simulation* 52(2):137–162.

Ducamp, G., C. Gonzales, and P.-H. Wuillemin. 2020. "aGrUM/pyAgrum : A Toolbox to Build Models and Algorithms for Probabilistic Graphical Models in Python". In *10th International Conference on Probabilistic Graphical Models*. September 23th-25th, Skørping, Denmark, 609-612.

Dunke, F., and S. Nickel. 2020. "Neural Networks for the Metamodeling of Simulation Models with Online Decision Making". *Simulation Modelling Practice and Theory* 99:102016.

Husmeier, D., F. Dondelinger, and S. Lebre. 2010. "Inter-time Segment Information Sharing for Non-homogeneous Dynamic Bayesian Networks". *Advances in Neural Information Processing Systems* 23:901–909.

Jordan, M. I., Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. 1999. "An Introduction to Variational Methods for Graphical Models". *Machine Learning* 37:183–233.

Kleijnen, J. P. 1987. *Statistical Tools for Simulation Practitioners*. New York: Marcel Dekker.

Koller, D., and N. Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques*. Cambridge, Massachusetts: MIT Press.

Madsen, A. L., and F. V. Jensen. 1999. "Lazy Propagation: A Junction Tree Inference Algorithm based on Lazy Evaluation". *Artificial Intelligence* 113(1-2):203–245.

Marazopoulou, K., M. Maier, and D. Jensen. 2015. "Learning the Structure of Causal Models with Relational and Temporal Dependence". In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*. July 12th-16th, Amsterdam, The Netherlands, 572–581.

Murphy, K. P. 2002. *Dynamic Bayesian Networks: Representation, Inference and Learning*. Ph.D. thesis, University of California, Berkeley, California.

Murphy, K. P., Y. Weiss, and M. I. Jordan. 1999. "Loopy Belief Propagation for Approximate Inference: An Empirical Study". In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*. July 30th-August 1st, Stockholm, Sweden, 467–475.

Ouyang, H., and B. L. Nelson. 2017. "Simulation-based Predictive Analytics for Dynamic Queueing Systems". In *Proceedings of the 2017 Winter Simulation Conference*, edited by W. K. V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, 1716–1727. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Poropudas, J., J. Pousi, and K. Virtanen. 2011. "Multiple Input and Multiple Output Simulation Metamodeling using Bayesian Networks". In *Proceedings of the 2011 Winter Simulation Conference*, edited by S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu, 569–580. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Poropudas, J., and K. Virtanen. 2010. "Simulation Metamodeling in Continuous Time using Dynamic Bayesian Networks". In *Proceedings of the 2010 Winter Simulation Conference*, edited by B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, 935–946. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Pousi, J., J. Poropudas, and K. Virtanen. 2013. "Simulation Metamodelling with Bayesian Networks". *Journal of Simulation* 7:297–311.

Salemi, P., J. Staum, and B. L. Nelson. 2019. "Generalized Integrated Brownian Fields for Simulation Metamodeling". *Operations Research* 67(3):874–891.

Shpitser, I., T. VanderWeele, and J. M. Robins. 2010. "On the Validity of Covariate Adjustment for Estimating Causal Effects". In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*. July 8th-11th, Catalina Island, California, 527–536.

Xuan Vinh, N., M. Chetty, R. Coppel, and P. P. Wangikar. 2012. "Gene Regulatory Network Modeling via Global Optimization of High-order Dynamic Bayesian Network". *BMC Bioinformatics* 13:1–16.

AUTHOR BIOGRAPHIES

PRACHETA AMARANATH is a Ph.D. candidate at the University of Massachusetts Amherst, Manning College of Information and Computer Sciences. Her email address is pboddavarama@cs.umass.edu and her web page is https://prachetaba.github.io/. Her work was supported by the DARPA CAML and SAIL-ON programs.

SAM WITTY is a Research Scientist at Basis Research Institute. His email address is sam@basis.ai and his web page is https://samwitty.github.io. His work was supported by the DARPA ASKEM program.

PETER J. HAAS is a Professor at the University of Massachusetts Amherst, Manning College of Information and Computer Sciences. His email address is phaas@cs.umass.edu and his web page is https://www.cics.umass.edu/people/haas-peter.

DAVID JENSEN is a Professor at the University of Massachusetts Amherst, Manning College of Information and Computer Sciences. His email address is jensen@cs.umass.edu and his web page is https://groups.cs.umass.edu/jensen/. His work was supported by the DARPA CAML and SAIL-ON programs.