Publication I

Jirka Poropudas and Kai Virtanen. 2011. Simulation metamodeling with dynamic Bayesian networks. European Journal of Operational Research, volume 214, number 3, pages 644-655.

European Journal of Operational Research 214 (2011) 644–655

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Stochastics and Statistics

Simulation metamodeling with dynamic Bayesian networks

Jirka Poropudas [*], Kai Virtanen

Systems Analysis Laboratory, Aalto University, School of Science, P.O. Box 11100, FI-00076 Aalto, Finland

ARTICLE INFO

Article history:
Received 5 May 2010
Accepted 3 May 2011
Available online 11 May 2011

Keywords:
Simulation
Dynamic Bayesian networks
Discrete event simulation
Simulation metamodeling

ABSTRACT

This paper presents a novel approach to simulation metamodeling using dynamic Bayesian networks (DBNs) in the context of discrete event simulation. A DBN is a probabilistic model that represents the joint distribution of a sequence of random variables and enables the efficient calculation of their marginal and conditional distributions. In this paper, the construction of a DBN based on simulation data and its utilization in simulation analyses are presented. The DBN metamodel allows the study of the time evolution of simulation by tracking the probability distribution of the simulation state over the duration of the simulation. This feature is unprecedented among existing simulation metamodels. The DBN metamodel also enables effective what-if analysis which reveals the conditional evolution of the simulation. In such an analysis, the simulation state at a given time is fixed and the probability distributions representing the state at other time instants are updated. Simulation parameters can be included in the DBN metamodel as external random variables. Then, the DBN offers a way to study the effects of parameter values and their uncertainty on the evolution of the simulation. The accuracy of the analyses allowed by DBNs is studied by constructing appropriate confidence intervals. These analyses could be conducted based on raw simulation data but the use of DBNs reduces the duration of repetitive analyses and is expedited by available Bayesian network software. The construction and analysis capabilities of DBN metamodels are illustrated with two example simulation studies.

1. Introduction

Many real world systems are complex, include convoluted stochastic phenomena, and have intricate internal dynamics evolving in time. The analysis of such systems is often conducted using discrete event simulation (DES, e.g. [1]). In a DES model, the state of simulation is defined using state variables whose values change at discrete time instants. The state changes, i.e., simulation events, imitate the behavior of the actual system. The changes are dictated by the simulation input including values of simulation parameters used to define alternative system configurations and operational environments as well as by the internal logic and random factors of the model. Simulation analysis is based on simulation data that consist of the values of the simulation input and output. The data are produced by replicating the simulation several times with different realizations of the random factors determined by non-overlapping pseudorandom number streams. The performance of the system with the given values of the simulation parameters is studied by calculating descriptive statistics or empirical distributions from the simulation data. For a more detailed presentation of simulation analysis, see, e.g. [1,2].

The repetition of simulations may be time consuming and the sheer size of simulation data sets can make them unwieldy. By using simulation metamodels [2–4] to represent the dependence between the simulation input and output, the simulation data can be concentrated and refined into a more manageable form while retaining its significant features and properties. Thus, simulation metamodels provide compact representations for the same system as the simulation model under consideration. They include, e.g. regression models [5,6], spline models [3], neural networks [7], radial basis functions [8], Kriging or spatial correlation models [2,9], frequency domain models [10], response surfaces [3,11], and game theoretic models [12]. All these models are input–output mappings that project the values of the simulation input to the values of the simulation output. They are used for simplifying and interpreting simulation models [4], conducting sensitivity and what-if analyses [4] as well as optimizing the simulation output [13].

The existing simulation metamodels measure system performance using the average value of the simulation state during the simulation or the final state of the simulation. For many purposes, this is sufficient but one should note that such models do not explicitly present the time evolution of the simulation. Thus, they give no information about the simulation state at specific time instants. Furthermore, these models do not depict the dependence between the simulation state variables at different time instants,

[*] Corresponding author. Tel.: +358 9 470 23064.
E-mail addresses: jirka.poropudas@tkk.fi (J. Poropudas), kai.virtanen@tkk.fi (K. Virtanen).

0377-2217/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2011.05.007

i.e., how the fixed simulation state affects the following simulation. Overall, the existing simulation metamodels do not capitalize all the available information produced by a DES model which can be of interest in the analysis of time dependent systems such as queues [14], maintenance of aircraft [15], and air combat [12].

In this paper, this deficiency is tackled by using dynamic Bayesian networks (DBNs) [16] as simulation metamodels. The DBNs have been previously used in the analysis of simulation data [17] but now they are presented as a new type of simulation metamodel. The DBNs are a particular kind of Bayesian networks (BNs) [18–20] that describe the joint probability distribution of a collection of random variables using a network whose nodes represent the random variables and their interconnecting arcs indicate dependencies between them. Each node is associated with a conditional probability table containing the probability distribution of the corresponding random variable. In DBNs, the set of variables is partitioned into time slices so that each slice consists of variables representing a single time instant.

In a DES model, the time dependent simulation state corresponds to a stochastic process, i.e., a sequence of random variables. Each simulation replication produces a realization of this process, i.e., a time series. The time series are used to construct a DBN metamodel for the joint probability distribution of the sequence of random variables describing the simulation state at discrete times. The resulting DBN captures also the dependencies between the simulation states at different time instants revealing the time evolution of the simulation.

This paper presents a construction process for optimally structured DBN metamodels. In general, the construction of a metamodel consists of several stages [4]: estimation of the model, analysis of modeling assumptions, and validation of the model. In the case of a DBN, the estimation includes both the definition of the structure of the network and the estimation of its parameters, i.e., the probability tables, see, e.g. [20]. The definition of the structure involves also the selection of time instants of the time slices, i.e., the time instants at which the DBN represents the simulation state [17]. This selection is now formulated as an optimization problem that is solved using a genetic algorithm [21]. When constructing DBNs, no assumptions about underlying probability distributions are made and therefore the analysis of modeling assumptions is omitted. In the validation stage, probability distributions given by the DBN are compared with corresponding distributions estimated from an independent validation data set using appropriate confidence intervals for the probabilities or χ^2-test for goodness of fit. In this way, it is confirmed that the DBN metamodel is an accurate representation of the simulation model.

Overall, the construction of a DBN metamodel is an iterative process where some of the stages may be repeated before a satisfying model is found. Nevertheless, the construction process is straightforward due to existing software tools for building and estimating BNs (e.g. [22]). The construction is furthermore aided by prior knowledge of the simulated system as DBNs seamlessly combine expert knowledge with data produced by simulation.

A DBN metamodel enables various novel simulation analyses that are easy to carry out using the available BN software. The DBN is used to calculate marginal and conditional probability distributions of the simulation state variables as a function of time which reveals the time evolution of the simulation. In what-if analysis, the value of the simulation state at a given time instant is fixed and the probability distributions of the state variables at other time instants are updated. The updating of conditional probability distributions with the DBN is faster than filtering of the original simulation data for cases that fulfill the conditions imposed on the simulation state. Therefore, the use of the DBN reduces the duration and computational requirements of repetitive analyses where many cases are to be considered. The accuracy of analyses is studied by calculating confidence intervals for the estimates of probabilities and expected values related to the simulation state.

Simulation parameters can also be included in a DBN metamodel as external random variables. Such a DBN describes the dependence between the values of the parameters and the evolution of the simulation. Furthermore, the parameter values can be treated as uncertain factors and the effect of this uncertainty on the simulation can be studied. The DBN containing the simulation parameters can also be applied for probabilistic reasoning where the simulation state is fixed at a given time and the DBN is used to update the conditional probability distributions of the external random variables representing the simulation parameters.

The paper is structured as follows. A short introduction to BNs and DBNs as required in simulation metamodeling is given in Section 2. The process for constructing a DBN metamodel based on simulation data is presented in Section 3. Section 4 introduces the utilization of such a metamodel in simulation analyses. The analysis capabilities are further explored and illustrated in Section 5 where two example simulation analyses are presented. Finally, the paper is summarized and conclusions are given in Section 6.

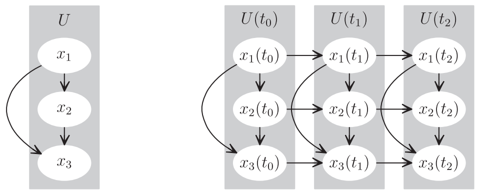
2. Dynamic Bayesian networks

A BN [18–20] is a probabilistic model that presents the joint distribution of a set of discrete random variables on three levels: relational, functional, and numerical [23]. On the relational level, the BN is a graphical representation of dependencies between the random variables using nodes and arcs. Conditional probability distributions and a chain rule [18] are used for the functional presentation of the joint distribution of the variables. Finally, efficient algorithms, e.g. [19,20], are used to calculate marginal and conditional distributions of individual variables on the numerical level.

In a BN representing the domain of discrete random variables $U = \{x_1, \ldots, x_n\}$, the nodes of the network correspond to the variables $x_1, \ldots, x_n$, see Fig. 1(a). Each variable x_i has a set of possible values denoted by $k \in K_i$. When the value of variable x_i is known, i.e., $x_i = k$ where $k \in K_i$, it is said to be instantiated. The instantiation of a random variable can be interpreted as observing its value. An instantiation k_X for a set of variables $X \subseteq U$ is defined as $X = k_X$ where each $x_i \in X$ is instantiated as $x_i = k \in K_i$, i.e., the values of variables $x_i \in X$ are fixed.

Arcs of the network denote dependencies between variables so that connected variables depend on each other, see Fig. 1(a). Together, the nodes and the arcs constitute a directed acyclical graph that presents the dependence structure between the variables U. The set of parents of x_i, i.e., the nodes with an arc pointing to node x_i, is denoted by Π_i. The root nodes of the network have no parents and then $\Pi_i = \emptyset$. The variables associated with the root nodes do not depend directly on the other variables.

Variable x_i follows a discrete probability distribution $P(x_i)$ that consists of probabilities $P(x_i = k)$ for all $k \in K_i$. The probability of



(a) BN consisting of three random variables $U = \{x_1, x_2, x_3\}$.

(b) DBN representing a system at time instants $t \in T = \{t_0, t_1, t_2\}$. A single time slice consists of random variables $U(t) = \{x_1(t), x_2(t), x_3(t)\}$.

Fig. 1. Example BN and DBN.

instantiation k_X for a set of variables X, i.e., the probability of variables $x_i \in X$ having values k_X, is denoted by $P(X = k_X)$. The probability distribution $P(X)$ for these variables includes the probabilities of all possible values k_X. The conditional probability of instantiation $X = k_X$ when $Y = k_Y$ is denoted by $P(X = k_X | Y = k_Y)$. The conditional probability distribution $P(X|Y)$ consists of the probabilities of all possible values k_X for all possible values k_Y.

In a BN, for each variable $x_i \in U$, the conditional probability distribution $P(x_i|\Pi_i)$ is presented using a probability table associated with the respective node. For root nodes, the probability table contains the marginal distribution of the variable, i.e., $P(x_i|\emptyset) = P(x_i)$. The joint probability distribution of U is calculated by using the chain rule: $P(U) = P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i|\Pi_i)$ [18]. The chain rule is also applied for calculating the marginal distributions of individual variables as well as for inference where the values of some variables are instantiated and then the conditional probability distributions of other variables are updated.

DBNs [16] are BNs with a special structure that describes a sequence of random variables. Let $P(U(t))$ be the probability distribution for a set of random variables $U(t) = \{x_1(t), \ldots, x_n(t)\}$ at time instant t. The variables associated with the same time instant are said to form a single time slice. A DBN with time slices at time instants $t \in T = \{t_0, t_1, \ldots, t_f\}$ is used to present a sequence of random variables at discrete times T. The structure within a time slice represents dependence between the simultaneous variables while the arcs between the time slices represent the dependence in time. In Fig. 1(b), the BN presented in Fig. 1(a) is extended to a DBN consisting of three time slices. In practice, DBNs are constructed in a similar manner to BNs. For a more detailed presentation of the construction of BNs, see, e.g. [20,24].

3. Construction of DBN metamodels

In the following, the construction of a DBN metamodel based on data produced by a DES model is introduced. The construction process begins with the selection of variables, the acquisition of simulation data, and the optimal selection of time instants. The data are then used in the determination of the network structure as well as in the estimation of probability tables. Then, the construction is finalized by possible inclusion of simulation parameters as well as the validation of a DBN metamodel. The construction is an iterative process and some of the abovementioned steps may have to be performed several times before an acceptable model is found.

3.1. Selection of variables

To construct a DBN metamodel, a set of simulation state variables is selected to represent the simulated system. Suppose that there are n state variables, denoted by $x_1(t), \ldots, x_n(t)$, needed to define the simulation state at a given time $t \in [t_0, t_f]$ where t_0 and t_f are the beginning and end of the simulation. For terminating simulations, the end time t_f is determined when a terminating condition is fulfilled [25]. In the case of a non-terminating simulation, a suitable value is selected for t_f. Then, the state of the simulation model at time t, denoted by $S(t) = \{x_1(t), \ldots, x_n(t)\}$, is given by the values of the state variables, i.e., $x_1(t) = k_1, \ldots, x_n(t) = k_n$ where $k_i \in K_i$. In a DES model, the simulation state $S(t)$ is a random variable whose probability distribution $P(S(t))$ changes continuously in time according to the internal logic and random factors of the model. In general, this probability distribution is unknown.

In the DBN metamodel, the simulation state variables are considered at discrete time instants $t \in T = \{t_0, t_1, \ldots, t_f\}$, i.e., the simulation state $S(t)$ at time instant $t \in T$ is described by the time slice $U(t) = \{x_1(t), \ldots, x_n(t)\}$, $t \in T$. Note that the time index in $S(t)$ is continuous, $t \in [t_0, t_f]$, while the time slices $U(t)$ included in the DBN metamodel are associated with discrete time instants, $t \in T$. Then, a DBN is constructed to represent the joint probability distribution of random variables $U = \bigcup_{t \in T} U(t)$. The joint probability distribution $P(U)$ is used to determine the time evolution of the simulation state, i.e., to estimate the probability distribution of the simulation state $P(S(t))$ as a function of discrete time. For the intermittent time instants, the probability distribution $P(S(t))$ can be approximated by using the probability estimates given by the DBN and, e.g. a piecewise linear interpolation.

3.2. Simulation data

In a DES model, time progresses continuously as simulation events can take place at any time instant $t \in [t_0, t_f]$. When simulations are performed, each replication yields a realization of the evolution of the simulation state, i.e., a time series representing the simulation state is observed for time interval $t \in [t_0, t_f]$. Several simulation replications are run and the simulation events as well as their times are recorded into a simulation data set. The data set is used in the construction of a DBN in two ways. First, it can be used to estimate $P(S(t)) = P(x_1(t), \ldots, x_n(t))$ at any given time instant t by screening the data for the state of the simulation at that time. To approximate the continuous time evolution of the simulation, the probabilities of the values of the simulation state variables as a function of time, i.e., the probability curves, are calculated from the data for a suitably fine grid of time instants. The probability curves are used in selecting the number of time slices and their optimal positioning as discussed in Section 3.3. Second, after the structure of the DBN is defined, the data set is used to estimate the probability tables of the DBN as discussed in Section 3.5.

The proper selection of the number of simulation replications, denoted by N, is essential in order to achieve a sufficient data set while avoiding unnecessary simulation. For a DBN metamodel, as is the case with most simulation metamodels, the appropriate value of N is unknown before any simulations have been performed. However, when constructing a DBN, the sufficient value of N can be approximated by considering the asymptotically normal distribution of the maximum likelihood estimator used in estimating the probability tables $P(x_i(t)|\Pi_i)$. This estimation is discussed in more detail in Section 3.5. First, the desired accuracy of the probability estimates provided by the DBN is determined. The accuracy is measured as the half-length of the $(1-\alpha)\%$ confidence interval, denoted by d. Then, a lower limit is imposed to the probability of the least likely combination of the values of the parent nodes that is to be studied in what-if analysis, denoted by $p := P(\Pi_i = k_\Pi)$. The number of observations needed to achieve the accuracy d is $N_{nec} \approx z_{1-\alpha/2}^2/(4d^2)$ where $z_{1-\alpha/2}$ is the $100(1-\alpha/2)$th percentile of the standard normal distribution [26]. When the conditional probability $P(x_i(t) = k | \Pi_i = k_\Pi)$ is estimated from N simulation replications, approximately $N \cdot p$ replications are actually of use. Thus, the lower limit for the number of replications is

$$N \geqslant N_{nec}/p \approx z_{1-\alpha/2}^2/(4d^2 p). \tag{1}$$

In practice, the value of N can be rounded up. Additionally, more simulations can be performed if this is found necessary in the validation of the DBN metamodel.

3.3. Optimal selection of time instants

The time slices of a DBN metamodel represent the probability distribution of the simulation state $P(S(t))$ at discrete time instants $t \in T$ although this distribution changes in continuous time. Because of this discrepancy, the representation for the time evolution of $P(S(t))$ given by a DBN is approximative. The number of time slices, denoted by $|T|$, and their positioning T are determined based on the probability curves discussed in Section 3.2 to find the best approximation for the evolution of the simulation.

The number of time slices $|T|$ is a compromise between the size of the model and the quality of the approximation. A greater number of time slices decreases the approximation error but, on the other hand, excess time slices unnecessarily increase the size of the model. Therefore, the value of $|T|$ is determined as the smallest number of time slices needed to produce an approximation where the probability curves appear almost linear between time slices. For example, Fig. 2(a) and (b) illustrate a case in which the probability curves can be approximated with six time slices provided that they are positioned optimally. The required number of time slices depends also on the intended utilization of the DBN metamodel. If the DBN is to be analyzed, e.g. within the time scale of seconds, the separation between successive time instants should be in a similar scale which can be used to assess the number of time slices.

When the value of $|T|$ is determined, the time instants of the time slices must be selected deliberately. Now, the time instants are selected such that they minimize an approximation error which results in an optimization problem. The objective function of the optimization problem, i.e., the approximation error, may be formulated in many ways. The objective function used in this paper is based on the probabilities $P(x_i(t) = k)$ that are estimated from the simulation data for all values $k \in K_i$ of all simulation state variables $x_i(t) \in S(t)$ at times $t \in T = \{t_0, t_1, \ldots, t_f\}$. The probabilities for other time instants $t \in [t_0, t_f]$ are approximated using a piecewise linear interpolation, denoted by $\widehat{P}(x_i(t) = k)$, that is

$$\widehat{P}(x_i(t) = k) := P(x_i(t_-) = k) + \frac{t - t_-}{t_+ - t_-}[P(x_i(t_+) = k) - P(x_i(t_-) = k)], \tag{2}$$

where $t_- := \max\{u \leqslant t | u \in T\}$ and $t_+ := \min\{u > t | u \in T\}$.

To motivate the optimal selection of time instants, an example of probability curves and their piecewise linear interpolations is given in Fig. 2(a). The time instants are set equidistantly which results in a considerable approximation error. For example, between time instants t_0 and t_1, the interpolations do not expose the fact that the probability distribution of the simulation state remains unchanged at the beginning of the simulation. On the other hand, between times t_2 and t_4 there exists fluctuation in the distribution which is ignored by the interpolations. Overall, the piecewise linear interpolations presented in Fig. 2(a) do not give an acceptable representation for the probability curves.

Now, the objective for the optimal selection of time instants is to minimize the maximal absolute error of the piecewise linear interpolations which is denoted by $M(T)$. The maximal absolute error $M(T)$ is calculated with respect to the entire duration of the simulation $t \in [t_0, t_f]$, all state variables, and all their values. Thus, the optimization problem is of the form

$$\begin{aligned} &\min_T \quad M(T), \\ \text{where} \quad & M(T) := \max\nolimits_{t\in[t_0,t_f],x_i(t)\in S(t),k\in K_i} |e_{i,k}(t)|, \\ \text{and} \quad & e_{i,k}(t) := P(x_i(t) = k) - \widehat{P}(x_i(t) = k), \end{aligned} \tag{3}$$

where the time instants $T = \{t_0, t_1, \ldots, t_f\}$ are the decision variables. The solution of the problem (3) yields the optimal time instants providing the best approximation for the probability distribution of the simulation state when the number of time slices $|T|$ is fixed.

The optimization problem (3) is non-linear and non-convex. Therefore, it is impossible to solve with gradient-based optimization methods. Fortunately, the solution of the problem need not be the global optimum as the purpose of the optimization is to avoid completely inappropriate selection of the time instants. For example, situations where several time slices present the same unchanged distribution should be avoided. Similarly, the probability distribution may have sharp momentary peaks that have to be included in the DBN. For such purposes, a suboptimal solution for the problem (3) can be obtained with a genetic algorithm, e.g. [21].

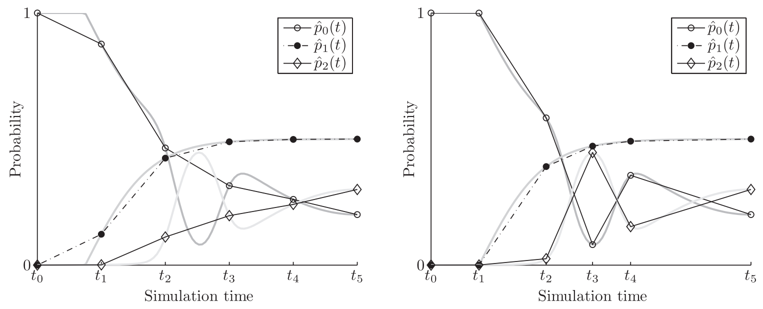
In this paper, the population of solution candidates used in the genetic algorithm consists of a number of real-valued vectors T. Whenever the algorithm requires the evaluation of the objective function (3), the maximal approximation error in interval $[t_0, t_f]$ is calculated based on the probability curves, i.e., the optimization is done based on the existing simulation data. The algorithm creates a new generation of potential candidates by selecting the best candidates as well as by crossing them over or mutating them.

The adequacy of the optimal solution is verified by comparing the probability curves and the piecewise linear interpolations. The interpolations should follow the probability curves in a satisfactory manner. For example, the comparison of Fig. 2(a) and (b) implies that the interpolations based on the optimized time instants give a remarkably better approximation.

If the optimization results in an inaccurate approximation, the genetic algorithm may have converged at an unacceptable local minimum and needs to be rerun. If satisfactory time instants are not found despite numerous trials, more time slices can be added, i.e., the value of $|T|$ is increased. On the other hand, the number of time instants can be decreased and the optimization problem resolved, if the optimal solution includes time instants that are close together and present nearly identical probability distributions.

3.4. Determination of structure

After the variables and the time instants have been selected, the structure of a DBN, i.e., dependencies between the variables, is



(a) Equidistant time instants. (b) Optimized time instants.

Fig. 2. Grey lines present the probability curves $P(x(t) = k)$ for the values $k \in \{0, 1, 2\}$ of the simulation state variable. The piecewise linear interpolations $\widehat{P}(x(t) = k)$ are calculated using the probabilities presented by the markers at time instants $t \in T = \{t_0, t_1, \ldots, t_f\}$.

determined using the same principles as for BNs [24,27–29]. The effective determination of the network structure combines expert knowledge and background information about the system at hand with the simulation data. In practice, the initial structure of the DBN is defined in two phases. First, the internal structure of individual time slices is determined and then dependencies between the time slices are assessed.

If expert knowledge is available, the initial structure of the DBN is obtained by first including the apparent dependencies into the network, i.e., by connecting the interdependent nodes with arcs. The initial structure can then be refined with additional arcs if the simulation data exhibit other dependencies. The structure of the DBN can also be estimated entirely from the simulation data without accessing any prior knowledge of the network structure. However, well-grounded estimation of the structure based only on the simulation data necessitates a large data set and this issue is exacerbated in the case of DBNs as the size of the network is multiplied by the number of time slices. The determination of the structure of DBNs is assisted by available software such as HUGIN [30] or GeNIe [31] that include graphical user interfaces for inclusion of expert knowledge as well as readily implemented algorithms for analysis of data.

3.5. Estimation of probability tables

In a DBN, each variable $x_i(t)$ is associated with a conditional probability table including conditional probability distribution $P(x_i(t)|\Pi_i = k_\Pi)$ for each combination of values k_Π of its parents Π_i. The conditional probabilities are estimated using maximum likelihood estimators (see, e.g. [32]) that maximize the likelihood of the simulation data for a given network structure [29]. For discrete random variables used in DBN metamodels, maximum likelihood estimators are given by the relative frequencies of the observed values of the simulation state variables [24].

Denote the true value of a conditional probability by $\theta_k := P(x_i(t) = k|\Pi_i = k_\Pi)$. The estimate $\hat{\theta}_k$ is based on a sample of observations $\mathbf{N} = (N_1, \ldots, N_{|K|})$ from multinomial distribution *Multinomial*$(N_\Pi, \boldsymbol{\theta})$ where N_Π is the number of simulation replications in which $\Pi_i = k_\Pi$, N_k is the number of simulation replications in which $x_i(t) = k$ and $\Pi_i = k_\Pi$, and $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_{|K|})$. Here, $|K|$ denotes the number of possible values for the random variable $x_i(t)$. Note that for the multinomial distribution, $Cov(N_k, N_k) = N_\Pi\theta_k(1-\theta_k)$ and $Cov(N_k, N_m) = -N_\Pi\theta_k\theta_m$, if $k \neq m$.

Using the above-mentioned sample, the maximum likelihood estimator for probability θ_k is given by $\hat{\theta}_k = N_k/N_\Pi$. The conditional probabilities of the root nodes are estimated based on all the simulation data and, thus, for them $N_\Pi = N$. In practice, the estimation of probability tables of the DBNs is carried out with the same software as the determination of the network structure. The asymptotic distribution of the maximum likelihood estimator [26] is

$$\hat{\theta}_k \sim N\left(\theta_k, \frac{\theta_k(1-\theta_k)}{N_\Pi}\right). \tag{4}$$

This distribution is used to calculate confidence intervals for probability estimates as discussed in Section 4.3. Recall also that these confidence intervals are used in Section 3.2 to determine the necessary number of simulation replications.

3.6. Inclusion of simulation parameters

In a DES model, alternative operating environments and system configurations are defined using parameters $z_1, \ldots, z_m$ each with a set of possible values $\ell \in Z_j$. The simulation parameters can be included in a DBN metamodel as external random variables. This approach enables also the treatment of deterministic parameters by setting the probability of a given set of parameter values equal to one. The dependence between the simulation parameters and the time evolution of the simulation is studied by running simulations with given values of $z_1, \ldots, z_m$ and constructing the DBN metamodel for the distribution $P(U(t_0), \ldots, U(t_f)|z_1, \ldots, z_m)$ based on the simulation data.

The simulation parameters can also represent uncertain factors that affect the simulated system. The effect of this uncertainty can be studied using a DBN metamodel containing variables corresponding to the parameters. The probabilities of the parameter values cannot be estimated from the simulation data because they are fixed for each simulation replication. Nevertheless, the probability of a given combination of simulation parameter values can be evaluated, e.g. using expert knowledge or historical data. This prior probability distribution is then included in the DBN representing the joint probability distribution $P(U) = P(U(t_0), \ldots, U(t_f), z_1, \ldots, z_m)$.

3.7. Validation

Once constructed, a simulation metamodel is validated by comparing it to simulation data [4,11,33,34]. Whenever possible, the validation is performed using a second independent data set to ascertain that the metamodel produces results that are generalizable [33]. In the case of DBNs, marginal and conditional probability distributions as well as probabilities given by the DBN are compared to distributions and probabilities estimated directly from the simulation data. The analysis of joint probability distribution $P(U)$ is inconvenient due to its high dimensionality. Therefore, the joint probability distribution is decomposed using the chain rule into conditional probability distributions of lower dimension, e.g. $P(U) = P(x_i(t)|U_{-i})P(U_{-i})$ where U_{-i} denotes the variables other than $x_i(t)$ that are included in the DBN. Alas, the exhaustive analysis is in general unattainable due to the computational complexity of estimating all possible conditional distributions from the simulation data and their possibly high number. Thus, the validation concentrates on the most relevant probabilities and probability distributions.

When probabilities are considered, the DBN is used to calculate marginal or conditional probability estimates and their confidence intervals for values of state variables. The resulting confidence intervals are compared with probabilities estimated from the independent simulation data. Now, the confidence intervals given by a proper DBN metamodel should cover the independent probability estimates.

Probability distributions produced by the DBN for given time instants are studied by estimating the same distributions from the independent simulation data. The goodness of fit between the two distributions is tested using χ^2-test (see, e.g. [26]). Now, the null hypothesis is that the DBN reproduces the distribution given by the simulation data and the alternative hypothesis is that the distributions are not the same. If significant differences are not found by testing the most relevant distributions, it is concluded that the DBN is a good representation for the joint probability distribution.

Overall, if notable discrepancy between the DBN metamodel and the independent simulation data is observed, it may be necessary to re-assess the structure of the network, include additional time slices, or increase the size of the simulation data set.

One should also note that if an independent simulation data set is unavailable, cross-validation techniques (see, e.g. [11]) can be used. That is, some of the observations are excluded from the data used in the construction of the DBN metamodel. Then, the validity of the DBN is measured by comparing it with these data.

4. Utilization of DBN metamodels

Next, the utilization of a constructed and validated DBN in simulation analysis is presented. The time slices of the DBN $U(t) = \{x_1(t),\ldots,x_n(t)\}$ correspond to the simulation state at time instants $t \in T = \{t_0,\ldots,t_f\}$ and the simulation parameters are included as external variables $z_j \in \{z_1,\ldots,z_m\}$. The constructed DBN presents the joint probability distribution $P(U(t_0),\ldots,U(t_f),z_1,\ldots,z_m)$ that allows the calculation of marginal and conditional distributions such as $P(U(t))$, $P(x_i(t))$, $P(U(t)|U(t'))$, and $P(x_i(t)|\, x_i(t'))$ where $t, t' \in T$. These calculations are easy to conduct by using available BN software. For intermittent time instants not included in T, the probability distributions are approximated by using the probabilities given by the DBN and a linear interpolation similar to Eq. (2).

4.1. Time evolution of simulation

In a DBN metamodel, the time evolution of the probability distribution $P(U(t))$, $t \in T$, describes the progress of the simulation, i.e., how the simulation state changes during the simulation. Thus, the DBN is used to assess the probability that the simulation is in state k_U at time $t \in T$, i.e., $P(U(t) = k_U)$. A single simulation state variable $x_i(t)$ is studied by tracking the evolution of the distribution $P(x_i(t))$. Furthermore, the probability of a particular value of the state variable $P(x_i(t) = k)$, where $k \in K_i$, may also be tracked over time instants $t \in T$. Additionally, in the case of terminating simulations, the DBN is used to calculate the distribution of the final simulation state represented by $P(U(t_f))$ or $P(x_i(t_f))$.

If a state variable has meaningful expected value, i.e., $k \in K_i$ is measured on interval or ratio scale, the average value of the state variable is studied using its expected value $E(x_i(t)) = \sum_{k\in K_i} k \cdot P(x_i(t) = k)$. The average evolution of the simulation state is then presented by the expected values $E(x_i(t))$ where $x_i(t) \in U(t)$ and $t \in T$.

4.2. What-if analysis

A DBN metamodel is applied for what-if analysis where the value of one or more simulation state variables $x_i(t') \in U(t')$ at time $t' \in T$ is fixed, i.e., the corresponding node in the DBN is instantiated. When conducting what-if analysis, the DBN can be used to calculate the probability of the presented conditions, e.g. $P(U(t') = k_U)$ or $P(x_i(t') = k)$, which yields information about the likelihood of such an occurrence and the accuracy of the following analysis. For example, if N simulation replications are used to construct a DBN and the probability of the fixed value is p, the what-if analysis is based on approximately $N \cdot p$ simulation replications. If the analysis is conducted based on highly unlikely observations, i.e., the value of $N \cdot p$ is small, the conditional probability estimates resulting from the what-if analysis may have wide confidence intervals.

The consequences of the fixed values of the simulation state variables are analyzed by calculating the conditional probability distributions of the state variables at other time instants. The conditional distributions are updated using the DBN resulting in, e.g. $P(U(t)|U(t') = k_U)$, $P(U(t)|x_i(t') = k)$, $P(x_i(t)|U(t') = k_U)$, or $P(x_i(t)|x_i(t') = k)$. By comparing alternative values of the state variables at time $t' \in T$, one can see how they affect the overall evolution of the simulation. The final simulation state can also be included in the analysis by calculating probability distributions such as $P(x_i(t_f)|x_i(t') = k)$ or $P(x_i(t)|x_i(t_f) = k)$.

The external random variables representing the simulation parameters are studied similarly by instantiating the value of one or more external variables $z_j = \ell$ and updating the distributions of the state variables accordingly which gives, e.g. $P(U(t)|z_j = \ell)$ or $P(x_i(t)|z_j = \ell)$. These conditional distributions are examined to see how the simulation is affected by the values of the simulation parameters. Such an analysis provides information about the evolution of the simulation state in different operating environments or with alternative system configurations.

The above what-if analyses can be combined by fixing both the values of the simulation state variables at a given time and the values of the external variables. The respective conditional probability distribution, e.g. $P(U(t)|x_i(t') = k, z_j = \ell)$ where $t \in T$, is then calculated using the DBN. This type of analysis reveals how the simulation is affected by the observed simulation state in the case of the given values of the simulation parameters.

It is also possible to conduct what-if analysis where some state variables are fixed and the prior distributions of the external variables are updated by calculating the respective conditional probabilities, e.g. $P(z_1,\ldots,z_m|x_i(t) = k)$ or $P(z_j|x_i(t) = k)$. Such an analysis enables the inference over the possible values of the simulation parameters when observations of the simulation state are made.

Conditional distributions can also be used to study how the final simulation state depends on the observed simulation state at some time instant or the values of the simulation parameters. For example, one can determine the conditional distributions $P(U(t_f)|U(t') = k_U)$, $P(U(t_f)|x_i(t') = k)$, or $P(U(t_f)|z_j = \ell)$.

When applicable, conditional probability distributions obtained in what-if analysis can be used to calculate the conditional expected value of the simulation state at time $t \in T$, e.g. $E(x_i(t)|x_i(t') = k) = \sum_{m\in K_i} m \cdot P(x_i(t) = m|x_i(t') = k)$ or $E(x_i(t)|z_j = \ell) = \sum_{m\in K_i} m \cdot P(x_i(t) = m|z_j = \ell)$. On the other hand, one can also study the conditional expected value of an external variable such as $E(z_j|x_i(t') = k) = \sum_{\ell\in Z_j} \ell \cdot P(z_j = \ell|x_i(t') = k)$.

4.3. Accuracy of estimates of probabilities and expected values

Let θ_k denote the true value of any conditional or marginal probability discussed in Sections 4.1 and 4.2. The accuracy of the probability estimate $\hat{\theta}_k$ is studied by calculating its confidence interval based on Eq. (4). The confidence interval is $\hat{\theta}_k \pm z_{\alpha/2}\sqrt{Var(\hat{\theta}_k)}$ where $z_{\alpha/2}$ is the $(1-\alpha/2)$th percentile of the standard normal distribution and $Var(\hat{\theta}_k) = \hat{\theta}_k(1-\hat{\theta}_k)/N_\Pi$. It should be noted that confidence intervals constructed in this way may not be entirely accurate (see, e.g. [35]).

Next, let μ denote an expected value or a conditional expected value discussed in Sections 4.1 and 4.2. The probability estimates $\hat{\theta}_k$ given by a DBN are used for calculating an estimate for the expected value as $\hat{\mu} = \sum_{k\in K} k \cdot \hat{\theta}_k$ where K is the set of values of a simulation state variable under consideration. The confidence interval for the expected value is $\hat{\mu} \pm z_{\alpha/2}\sqrt{Var(\hat{\mu})}$. Here, the variance is

$$Var(\hat{\mu}) = Var\left(\sum_{k\in K} k \cdot \hat{\theta}_k\right) = \sum_{k\in K}\sum_{m\in K} Cov\left(k \cdot \hat{\theta}_k, m \cdot \hat{\theta}_m\right)$$
$$= \sum_{k\in K}\sum_{m\in K} k \cdot m \cdot Cov\left(\hat{\theta}_k, \hat{\theta}_m\right),$$

where $Cov\left(\hat{\theta}_k, \hat{\theta}_k\right) = Cov(N_k, N_k)/N_\Pi^2 = \hat{\theta}_k\left(1-\hat{\theta}_k\right)/N_\Pi$ and $Cov\left(\hat{\theta}_k, \hat{\theta}_m\right) = Cov(N_k, N_m)/N_\Pi^2 = -\hat{\theta}_k\hat{\theta}_m/N_\Pi$, if $k \neq m$. Recall that $Cov(N_k, N_k)$ and $Cov(N_k, N_m)$ were discussed in Section 3.5.

4.4. Required computational effort

The computational effort needed in the utilization of a DBN metamodel, i.e., in the calculation of conditional probability distributions of simulation state variables, is next discussed. These distributions could also be determined directly based on simulation data. Nevertheless, it takes less computational effort to update conditional probability distributions with the DBN than to search the original data set for appropriate cases, i.e., those simulation replications where the simulation is in a given state at a given time instant. The difference in computational cost is outlined by considering the sizes of the DBN and the simulation data set.

In a DBN containing n variables, the probability of a given combination of values of the variables is computed as a product of n conditional probabilities [36]. If the simulation data are used for this computation, a data set of size N is first screened for the appropriate cases whose number is N_k. Then, the probability of the given combination is estimated as N_k/N. For all reasonable situations $N \gg n$. Thus, the screening of the data set requires more computational effort which makes the utilization of the DBN preferable – especially if similar analyses are performed repeatedly.

5. Example DBN metamodels

Next, the construction and use of DBN metamodels are illustrated with two examples that consider a queueing model [1,14] and an air combat simulation model [37]. In both cases, simulation data are collected and a DBN metamodel is constructed and analyzed following the principles presented in Sections 3 and 4 using the GeNIe software [31].

5.1. Queueing model

The first example illustrates the construction and validation of a DBN metamodel and the analysis possibilities provided by the DBN. These include the study of the time evolution of simulation and the effects of simulation parameters as well as the uncertainty associated with them. The simulation model under consideration represents a single server queueing system with Poisson arrivals and exponential service times with service intensity $\mu = 2.50$ [14]. The arrival intensity of the system is a random variable whose value is determined at the beginning of day to be either $\lambda = 1.50$ or $\lambda = 5.00$. These cases are referred to as quiet and busy days. The arrival intensity is a parameter of the simulation model that is included in the DBN metamodel as an external random variable.

The simulation state $S(t)$ is determined by a single state variable $x(t)$ that represents the number of customers in the system at time t. At the beginning of the simulation, the system is empty, i.e., $x(0) = 0$. The maximum length of the queue is limited so that the set of possible values of the simulation state is $K = \{0, 1, \ldots, 10\}$. The number of simulation replications N is determined using Eq. (1) with values $\alpha = 0.05$, $d = 0.05$, and $p = 0.05$. This gives a lower bound $N \geqslant 7683$ and it is rounded up to $N = 10000$. The duration of the simulation replications is set as 10 time units, i.e., $t_f = 10$, and the system is simulated N times for both values of the arrival intensity.

The simulation data are used to estimate the probability curves, i.e., the probabilities for the values of the simulation state variable as a function of time. Based on these curves, the number of time slices $|T|$ is determined and the optimization problem (3) is solved providing the optimal time instants. The number of time slices is set as $|T| = 10$ and the optimal time instants are found to be $T = \{0, 0.10, 0.31, 0.75, 1.76, 3.50, 4.50, 7.36, 8.94, 10.00\}$ by running the genetic algorithm for 5000 generations.

When constructing the initial structure of the DBN, it is assumed that the simulation state at a given time instant depends on the previous state as well as on the value of the external variable representing the arrival intensity. Then, additional dependencies are identified based on the simulation data and it is found that no additional arcs are needed which is expected due to the Markovian nature of the queueing system [14]. The final structure of the DBN is presented in Fig. 3. The data are also used in the estimation of the conditional probability tables for all the variables of the DBN. Finally, the prior distribution of the external variable λ is determined using expert knowledge. Now, it is assumed that the opinion of the expert is that there is a 80% chance that a randomly chosen day is quiet. Thus, the prior probability distribution of the external variable is $P(\lambda = 1.50) = 0.80$ and $P(\lambda = 5.00) = 0.20$.

The DBN metamodel is validated by comparing the probabilities given by it with the probability curves estimated from an independent simulation data set. In the validation, several representative conditional probabilities and their confidence intervals are studied. Examples of such probabilities are presented in Figs. 4 and 6. In Figs. 5 and 7, similar comparison is conducted using the expected and conditional expected value of the queue length with their confidence intervals. For all the cases studied, the DBN provides results that concur with the independent data. Thus, it is concluded that the DBN offers a good representation of the simulated queueing system.

Once validated, the DBN metamodel is used to study the progress of the simulation in time by tracking the probability distribution of the state variable $x(t)$ during the simulation. For example, Fig. 4 presents the probability of the system being full at time t, i.e., $P(x(t) = 10)$ and its confidence interval as given by the DBN. At the beginning of the simulation, $P(x(0.00) = 10) = 0.00$ as the queue starts empty. If the type of the day is unknown, the probability of the queue being full increases steadily and reaches its highest value at the end of the simulation that is $P(x(10.00) = 10) = 0.10$. The conditional probabilities for quiet and busy days, i.e., $P(x(t) = 10|\lambda = 1.50)$ and $P(x(t) = 10|\lambda = 5.00)$, and their confidence intervals are also presented in Fig. 4. On a quiet day, it is unlikely that the length of the queue reaches its upper limit. On the other hand, on a busy day, the probability of the system being full increases during the simulation and reaches 0.49 at the end of the simulation. Thus, one can conclude that the type of the day determines whether the queue will reach its maximal length and on approximately 50% of busy days the simulation terminates in state $x(10) = 10$.

Similar what-if analysis can be repeated for all other values of the simulation state variable. Due to the number of these values, the overall evolution of the simulation is presented using the expected value of the queue length $E(x(t)) = \sum_{k \in K} k \cdot P(x(t) = k)$ that is calculated using the prior probability distribution for the external variable λ. To study the effect of the arrival intensity on the simulation, the value of λ can also be fixed in order to calculate conditional expected values $E(x(t)|\lambda) = \sum_{k \in K} k \cdot P(x(t) = k|\lambda)$ where $\lambda = 1.50$ or $\lambda = 5.00$. The expected values presented in Fig. 5 imply

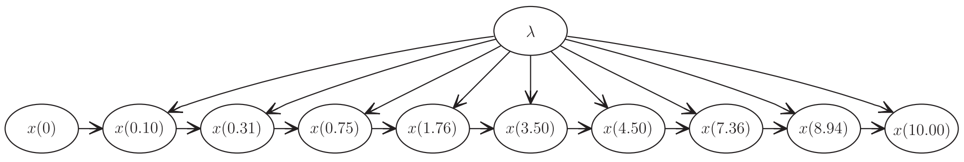


Fig. 3. DBN metamodel for the single server queueing system. The DBN has $|T| = 10$ time slices representing the simulation state $S(t) = x(t)$ at time instants $T = \{0, 0.10, 0.31, 0.75, 1.76, 3.50, 4.50, 7.36, 8.94, 10.00\}$. The external variable λ represents the arrival intensity of the customers.

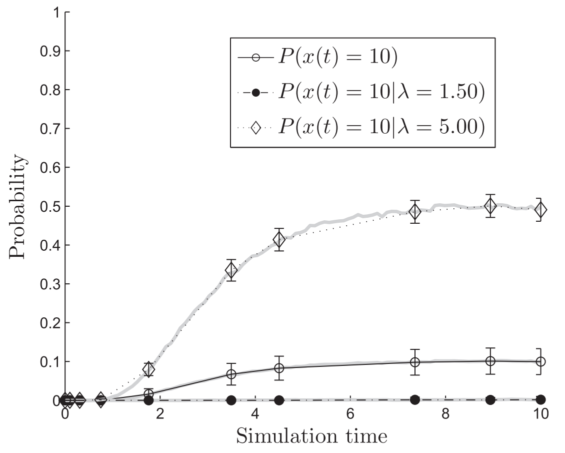


Fig. 4. Probability of the system being full, i.e., $P(x(t) = 10)$, as well as the conditional probabilities $P(x(t) = 10|\lambda = 1.50)$ and $P(x(t) = 10|\lambda = 5.00)$ for quiet and busy days as a function of time. The grey continuous lines present the probability curves estimated from the independent simulation data. The markers and the error bars denote the probabilities given by the DBN metamodel and their 95% confidence intervals.

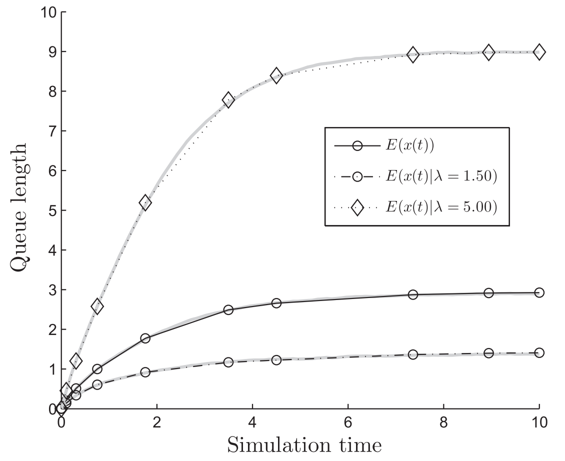


Fig. 5. Expected value of the queue length $E(x(t))$ as well as the conditional expected values $E(x(t)|\lambda = 1.50)$ and $E(x(t)|\lambda = 5.00)$ for quiet and busy days as a function of time. The grey continuous lines present the expected values estimated from the independent simulation data. The markers denote the expected values given by the DBN metamodel. Note that the maximum of the half-lengths of the confidence intervals is 0.034, i.e., they are so narrow that their illustration is omitted.

that the expected queue length starts at zero and increases during the simulation towards the value 2.92. In the case of a quiet day, the expected queue length at the end of the day is $E(x(10.00)|\lambda = 1.50) = 1.41$ and for a busy day $E(x(10.00)|\lambda = 5.00) = 8.98$. Thus, on a busy day, after ten time units the queue is almost full on average.

To conduct what-if analysis concerning the dependence between the simulation state at different time instants, the value of the simulation state variable is fixed at some time instant and the probability distributions of the variable at other time instants are updated. The conditional probability distributions reveal the effect of the observed simulation state on the evolution of the simulation. For example, in Fig. 6, the value of the state is set as

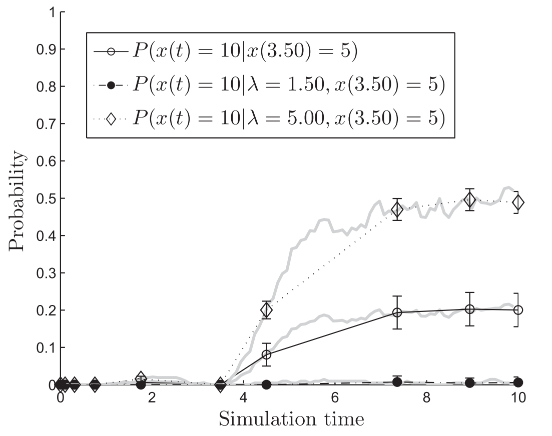


Fig. 6. Probability of the system being full when $x(3.50) = 5$, i.e., $P(x(t) = 10|x(3.50) = 5)$, as well as the conditional probabilities $P(x(t) = 10|\lambda = 1.50, x(3.50) = 5)$ and $P(x(t) = 10|\lambda = 5.00, x(3.50) = 5)$ for quiet and busy days as a function of time. The grey continuous lines present the probability curves estimated from the independent simulation data. The markers and the error bars denote the probabilities given by the DBN metamodel and their 95% confidence intervals.

$x(3.50) = 5$ and the probability of the system being full, $P(x(t) = 10|x(3.50) = 5)$, is updated accordingly. To compare a quiet and busy day, the respective conditional probability distributions, $P(x(t)|\lambda = 1.50, x(3.50) = 5)$ and $P(x(t)|\lambda = 5.00, x(3.50) = 5)$, are also presented.

Fig. 6 indicates that probability $P(x(t) = 10|x(3.50) = 5)$ remains at almost nil before time 3.50. Thus, regardless of the type of day, before $t = 3.50$ there is no time for the system to first fill up and then return to the state $x(3.50) = 5$. After $t = 3.50$, the conditional probabilities $P(x(t) = 5|\lambda = 1.50, x(3.50) = 5)$ and $P(x(t) = 5|\lambda = 5.00, x(3.50) = 5)$ approach values similar to the unconditioned values presented in Fig. 4.

To explore the dependence between the external variable representing the simulation parameter and the simulation state, what-if analysis may also be conducted in opposite direction. For example, it is assumed that the simulation state is observed to be $x(3.50) = 5$ and the distribution of the external variable is updated. The resulting conditional probabilities are $P(\lambda = 1.50|x(3.50) = 5) = 0.60$ and $P(\lambda = 5.00|x(3.50) = 5) = 0.40$. Recall that the prior probabilities are $P(\lambda = 1.50) = 0.80$ and $P(\lambda = 5.00) = 0.20$. Thus, the observation increases the likelihood of a busy day. This is also reflected in the conditional evolution of the simulation in Fig. 6 where the probability $P(x(t) = 10|x(3.50) = 5)$ approaches the value 0.20 which is twice the respective value 0.10 in Fig. 4.

The conditional overall evolution of the simulation is further studied by calculating the conditional expected values $E(x(t)|x(3.50) = 5)$, $E(x(t)|\lambda = 1.50, x(3.50) = 5)$, and $E(x(t)|\lambda = 5.00, x(3.50) = 5)$ which are presented in Fig. 7. After $t = 3.50$, the expected values corresponding to quiet and busy days diverge and approach similar values as in Fig. 5. Thus, on a quiet day, the system empties after the observation $x(3.50) = 5$ and on a busy day it fills up. Conditional on the observation, these courses are almost equally probable as $P(\lambda = 1.50|x(3.50) = 5) = 0.60$ and $P(\lambda = 5.00|x(3.50) = 5) = 0.40$. Therefore, the conditional expected queue length for unknown λ hovers just below 5 during the rest of the simulation which is seen in Fig. 7.

In the above what-if analysis, the probability distribution of the external variable is updated conditional on the observed simulation state $x(3.50) = 5$. Similar inference about the type of day can be performed for all possible values of the simulation state

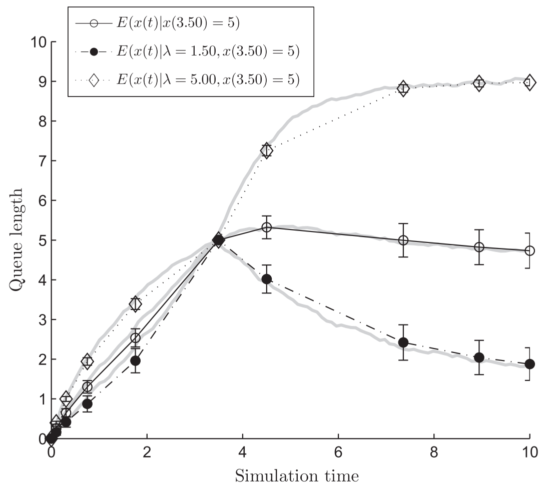


Fig. 7. Conditional expected value of the queue length $E(x(t)|x(3.50) = 5)$ as well as the conditional expected values $E(x(t)|\lambda = 1.50, x(3.50) = 5)$ and $E(x(t)|\lambda = 5.00, x(3.50) = 5)$ for quiet and busy days as a function of time. The grey continuous lines present the expected values estimated from the independent simulation data. The markers and the error bars denote the expected values given by the DBN metamodel and their 95% confidence intervals.

variable. For example, the probabilities $P(x(3.50) = k)$, the conditional probability distributions $P(\lambda|x(3.50) = k)$, $k \in K = \{0,1,\ldots,10\}$, and the corresponding 95% confidence intervals are presented in Table 1. The conditional probability distributions given by the DBN are also compared with the conditional probability distributions estimated from the independent validation data set using the χ^2-test for the goodness of fit. The P-values reported in Table 1 convey that no statistically significant differences between the distributions are found which supports the previously made conclusion about the validity of the model.

In Table 1, at time 3.50 the most likely state is 0 and the probabilities decrease with the queue length. The conditional probability distributions $P(\lambda|x(3.50) = k)$ indicate that longer queue lengths imply a higher likelihood of a busy day. If the queue length at time 3.50 is less than four, the day is almost certainly quiet. Similarly, the queue lengths of eight or more indicate a busy day. Thus, the observed simulation state at a given time can be used to update the prior probability distribution of the external variable which affects also the evolution of simulation.

Table 1
Probabilities $P(x(3.50) = k)$ and the conditional probability distributions $P(\lambda|x(3.50) = k)$ where $k \in K = \{0,1,\ldots,10\}$ as well as the corresponding 95% confidence intervals. The final row presents the unconditional probability distribution $P(\lambda)$ for comparison. The P-values refer to the χ^2-test in which the conditional probability distributions are compared with the independent data set.

$P(x(3.50) = k)$		$P(\lambda\|x(3.50) = k)$		χ^2-test
k	Prob.	1.50	5.00	P-value
0	0.353 (±0.009)	0.994 (±0.003)	0.006 (± 0.003)	0.975
1	0.209 (±0.008)	0.984 (±0.005)	0.016 (± 0.005)	0.999
2	0.121(±0.006)	0.958 (±0.011)	0.042 (± 0.011)	0.941
3	0.068 (±0.005)	0.913 (±0.021)	0.087 (± 0.021)	0.563
4	0.044 (±0.004)	0.797 (±0.038)	0.203 (± 0.038)	0.707
5	0.029 (±0.003)	0.598 (±0.056)	0.402 (± 0.056)	0.353
6	0.021 (±0.003)	0.348 (±0.064)	0.652 (± 0.064)	0.888
7	0.022 (±0.003)	0.155 (±0.048)	0.845 (± 0.048)	0.802
8	0.026 (±0.003)	0.036 (±0.023)	0.964 (± 0.023)	0.173
9	0.039 (±0.004)	0.019 (±0.014)	0.981(± 0.014)	0.520
10	0.067 (±0.005)	0.004 (±0.005)	0.996 (± 0.005)	0.263
–	–	0.800	0.200	–

5.2. Simulated air combat

The second example demonstrates the construction and validation of a DBN metamodel in the analysis of simulated air combat. The DBN is used to study the time evolution of the simulation and interdependencies between the simulation state at different time instants. As the air combat simulation is an example of terminating simulation, attention is paid to conducting what-if analyses where the effects of fixed values of the simulation state at different times on the final state of the simulation are studied. In the example, simulation data are produced with a discrete event air combat simulation model called X-Brawler [17,37,38] in order to construct and validate the DBN metamodel.

The air combat under consideration involves two aircraft, called blue and red. In the analysis, generic models for aircraft as well as for missiles, radars, and other hardware are used [37]. The initial geometry of the combat is advantageous for blue because blue is approaching red and preparing to launch a missile towards it. On the other hand, if the blue missile misses red, blue is at a considerable disadvantage as it has only one missile at its disposal while red carries two similar missiles.

The state of the simulated air combat at time t is described by two state variables $S(t) = \{x_1(t), x_2(t)\}$. The variable $x_1(t)$ is a indicator variable for "blue is alive at time t" and $x_2(t)$ for "red is alive at time t". As the state variables are binary, the sets of their values are $K_1 = K_2 = \{0,1\}$. Together, the variables define the state of the air combat at time t as

- *neutral*, i.e., both sides are alive, $x_1(t) = 1$ and $x_2(t) = 1$
- *blue advantage*, i.e., blue is alive and red has been killed, $x_1(t) = 1$ and $x_2(t) = 0$
- *red advantage*, i.e., blue has been killed and red is alive, $x_1(t) = 0$ and $x_2(t) = 1$
- *mutual disadvantage*, i.e., both sides have been killed, $x_1(t) = 0$ and $x_2(t) = 0$

To determine the necessary number of simulation replications, values $\alpha = 0.05$, $d = 0.05$, and $p = 0.05$ are used in Eq. (1) resulting in a lower limit $N \geqslant 7683$. Similarly to the first example, the number is rounded up to $N = 10000$. Each replication is terminated if both sides are killed. Otherwise, the simulation is terminated at $t_f = 500$ (seconds) which is also taken as the final time instant of the DBN metamodel.

In the construction of the DBN metamodel, the simulation data are first used to estimate the probability curves for the simulation state variables. Then, the optimization problem (3) is solved by running the genetic algorithm for 5000 generations in order to obtain the optimal time instants for the time slices of the DBN. The number of time instants is now set as $|T| = 8$ and the optimal time instants are $T = \{0,132,148,162,201,220,339,500\}$. The initial structure of the DBN is defined so that the state variables depend on both state variables at the previous time instant. The simulation data reveals also additional dependencies within the time slices as variables $x_1(t)$ and $x_2(t)$ are found to be dependent at $t = 148$, $t = 201$, and $t = 500$. There exists also dependencies across the time slices such that the state variables depend on the variables corresponding to time instants before the previous time slice. Thus, the simulated air combat appears to be non-Markovian and the simulation events may have long lasting and delayed effects on the evolution of the simulation. These dependencies are taken into account by adding arcs to the initial structure of the DBN. The final structure is presented in Fig. 8. The DBN metamodel is finalized by estimating conditional probability tables for all the nodes based on the simulation data.

The constructed DBN is validated by comparing the probabilities and their confidence intervals given by the DBN to the

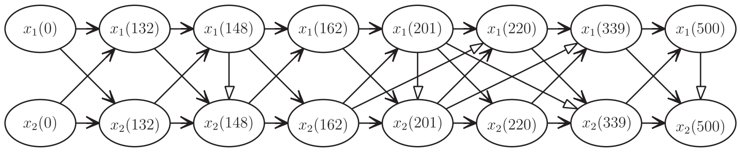


Fig. 8. DBN metamodel for the simulated air combat. The black arcs represent the initial structure of the DBN and the arcs with the white arrowheads are added for presenting further dependencies discovered in the simulation data.

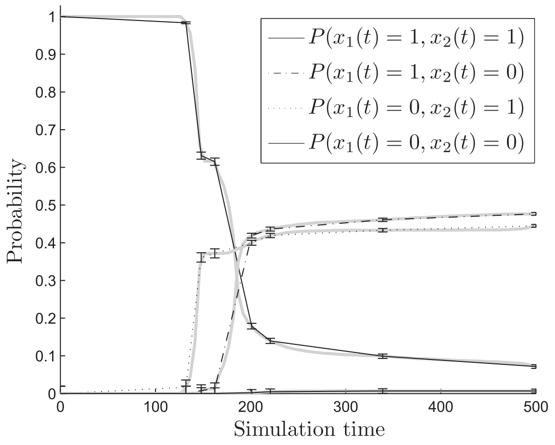


Fig. 9. Probability distribution of the state of the air combat simulation as a function of time. The grey continuous lines present the probability curves estimated from the independent simulation data. The error bars denote the 95% confidence intervals of the probabilities given by the DBN.

probability curves estimated from an independent data set. Overall, the DBN provides probabilities that reflect the independent data, see Fig. 9. As in the previous example, conditional probability distributions are also studied and it is concluded that the DBN is an appropriate representation of the simulation model.

The DBN is next applied for the study of the evolution of the simulation state. To understand the progress of the simulated air combat, the probabilities produced by the DBN metamodel are presented in Fig. 9. These probabilities imply that no prominent changes occur in the probability distribution of the simulation state before $t = 132$ or after $t = 340$. Thus, the most significant events take place in the time interval [132, 340]. The air combat starts in the *neutral* state where both sides are alive and remains in this state for the initial 132 seconds. At $t = 132$, the state of the simulation changes as the probability of blue shooting down red at this time is 0.36. The probability of *blue advantage* remains almost constant from this point on. If blue is to win the engagement, red is shot down at this juncture.

The probability of *red advantage* increases in the time interval [162, 201] by 0.41. After this time interval, the probability of *red advantage* increases only slightly. Red is unlikely to score a kill during the remainder of the simulation. Nevertheless, red is at a slight overall advantage in the combat as it is more likely to end in *red advantage* than in *blue advantage*. The probability distribution of the simulation state also reveals that the probability of *mutual disadvantage* is small, i.e., it is unlikely that the combat results in the destruction of both aircraft.

To demonstrate the analysis capabilities of the DBN metamodel related to the final simulation state, what-if analysis is conducted with respect to the final state of the air combat, i.e., $x_1(500)$ and $x_2(500)$. For example, the values of the simulation state variables $x_1(t')$ and $x_2(t')$ can be fixed at time instant $t' \in T$. Then, the probability distribution of the final state is updated using the DBN. Now, it is assumed that blue is still alive at time t' and the conditional probability distribution of the final simulation state is calculated. The probabilities $P(x_1(t') = 1)$ and the conditional probability distributions $P(x_1(500), x_2(500)|x_1(t') = 1)$ for all $t' \in T$ as well as the corresponding confidence intervals are presented in Table 2. The conditional probability distributions are also compared to the independent validation data set using the χ^2-test for the goodness of fit. The P-values related to this test are also reported in Table 2. The P-values show that no statistically significant differences between the distributions are observed at 0.05 significance level which supports the validity of the metamodel.

According to Table 2, blue is likely to survive until $t' = 162$ and observing $x_1(t') = 1$ before this time does not affect the outcome of the air combat. If blue survives the time interval [162, 201], the combat is likely to end in *blue advantage* whose probability increases during this time interval and continues to rise as long as blue stays alive. Similarly, the probability that the final simulation state is *neutral* increases during the simulation as long as blue is

Table 2
Probability distribution of the final simulation state $P(x_1(500), x_2(500)|x_1(t') = 1)$ conditional on blue being alive at time instants $t' \in T$ as well as the corresponding 95% confidence intervals. The values (i, j) of the simulation state variables at $t = 500$, where $x_1(500) = i$ and $x_2(500) = j$, correspond to the state of the air combat as follows: (1, 1) *neutral*, (1, 0) *blue advantage*, (0, 1) *red advantage*, and (0, 0) *mutual disadvantage*. The final row presents the unconditional probability distribution $P(x_1(500), x_2(500))$ for comparison. The P-values refer to the χ^2-test in which the conditional probability distributions are compared with the independent data set.

$P(x_1(t') = 1)$		$P(x_1(500), x_2(500)\|x_1(t') = 1)$				χ^2-test
t'	Prob.	(1,0)	(0,1)	(0,0)	(1,1)	P-value
0	1.000 (±0.000)	0.444 (±0.010)	0.475 (±0.010)	0.008 (±0.002)	0.073 (±0.005)	0.081
131	1.000 (±0.000)	0.444 (±0.010)	0.475 (± 0.010)	0.008 (±0.002)	0.073 (±0.005)	0.081
148	0.992 (±0.002)	0.447 (±0.010)	0.471 (± 0.010)	0.008 (±0.002)	0.074 (±0.005)	0.062
162	0.987 (±0.002)	0.449 (±0.010)	0.469 (± 0.010)	0.008 (±0.002)	0.074 (±0.005)	0.064
201	0.579 (±0.010)	0.766 (±0.011)	0.098 (± 0.008)	0.010 (±0.003)	0.126 (±0.009)	0.087
220	0.559 (±0.010)	0.794 (±0.011)	0.070 (± 0.007)	0.006 (±0.002)	0.130 (±0.009)	0.786
339	0.532 (±0.010)	0.833 (±0.010)	0.030 (± 0.005)	0.000 (±0.000)	0.137 (±0.009)	0.437
500	0.516 (±0.010)	0.859 (±0.010)	0.000 (± 0.000)	0.000 (±0.000)	0.141 (±0.010)	0.399
–	–	0.444 (±0.010)	0.475 (±0.010)	0.008 (± 0.002)	0.073 (±0.005)	–

not shot down. On the other hand, the longer blue is observed to have survived, the smaller is the conditional probability for the simulation ending in the *red advantage* state. Note that other analyses similar to ones carried out in the first example could also be conducted but their presentation is now omitted as it would provide no additional insight to the utilization of DBN metamodels.

6. Conclusions

This paper presented the construction of DBN metamodels based on simulation data and their utilization in simulation analyses. In this new metamodeling concept, the output of simulation is considered as a time series representing the evolution of the simulation instead of a single output variable. The DBN metamodels allow the study of the time evolution of the simulation. They can also be used for analyzing the effects of fixed values of the simulation state as well as the simulation parameters on the evolution of the simulation. Such studies cannot be performed with existing simulation metamodels but they could be carried out using the simulation data. Nevertheless, the use of the DBNs requires less computational effort and is thus less time consuming compared to the repeated re-screening of the data or performing additional simulations.

Two example simulation studies were presented to illustrate the application of DBN metamodels. The validation results imply that the probabilities and the expected values of the simulation state variables obtained with the DBNs are consistent with the values estimated from independent simulation data. In both examples, the time evolution of the simulation is studied and several what-if analyses are conducted. In the first example, a simulation parameter is included in the DBN metamodel. This enables statistical inference about the dependence between the evolution of the simulated system and its settings as well as about the value of the simulation parameter when the simulation state is fixed at a given time. The second example focuses on describing the dependence between the simulation state at a given time and the final state of the simulation. It should be noted that the presented examples are only illustrative demonstrations related to DBN metamodels and further what-if analyses could be performed with little computational effort.

The metamodeling concept presented in this paper can be extended into many directions. First, in this paper, the validation of DBN metamodels is carried out using independent data sets. If independent data is unavailable, bootstrapping might be employed in validation [39]. Second, the DBN metamodels can be used as a part of a BN that is constructed by using, e.g. expert knowledge, to represent a larger system or a system of systems. Third, in this paper, the values of simulation state variables are modeled at common time instants, i.e., all the time slices include all the state variables. One could also associate each of simulation state variables with their own time instants and time slices in the DBN. In this way, state changes taking place with different rates could be treated more effectively. Fourth, in this paper, the number of time slices is determined based on visual inspection. A more explicit formulation of this selection problem and the automatization of this step in the construction process would make the determination of the number of time slices more justifiable.

The depiction of external variables representing simulation parameters could also be further explored. In this paper, probability distributions of external variables are constant but these distributions could also depend on time to describe time dependent phenomena. Furthermore, external variables can also be taken as decision variables involved in a simulation based optimization or decision making problem. This leads to the use of influence diagrams [40] which are decision theoretical extensions of BNs. They can be constructed based on simulation data following the principles presented in this paper and used as simulation metamodels to solve the optimal values of the decision variables with respect to a given criterion [38] or even multiple criteria [23]. Simulation models representing settings with multiple decision makers could be analyzed by using influence diagram games [41] as game theoretic simulation metamodels [12].

Overall, DBN metamodels offer a compact and efficient representation for the evolution of simulation that expedites simulation studies as there is no need for repetitive re-screening of data. The computational advantages of the DBN metamodels proved to be instrumental in conducting several alternative what-if analyses that would be time consuming and arduous based only on simulation data. These metamodels provide significant additional insight to simulation analyses compared to the use of existing simulation metamodels and reveal dependencies within simulations that are not evident in the dissection of raw simulation data.

References

[1] A. Law, Simulation Modeling and Analysis, fourth ed., McGraw-Hill Science/Engineering/Math, New York, NY, 2006.
[2] J.P.C. Kleijnen, Design and Analysis of Simulation Experiments, first ed., Springer Science + Business Media, New York, NY, 2008.
[3] R.R. Barton, Simulation metamodels, in: Proceedings of the 1998 Winter Simulation Conference, Washington, DC, 1998, pp. 167–174.
[4] L.W. Friedman, The Simulation Metamodel, Kluwer Academic Publishers, Norwell, MA, 1996.
[5] R.W. Blanning, The sources and uses of sensitivity information, Interfaces 4 (4) (1974) 32–38.
[6] J.P.C. Kleijnen, Regression metamodels for generalizing simulation results, IEEE Transactions on Systems, Man, and Cybernetics 9 (2) (1979) 93–96.
[7] D.J. Fonseca, D.O. Navaresse, G.P. Moynihan, Simulation metamodeling through artificial neural networks, Engineering Applications of Artificial Intelligence 16 (3) (2003) 177–183.
[8] M. Hussain, R. Barton, S. Joshi, Metamodeling: Radial basis functions, versus polynomials, European Journal of Operational Research 138 (1) (2002) 142–154.
[9] B. Ankenman, B.L. Nelson, J. Staum, Stochastic kriging for simulation metamodeling, Operations Research 58 (2) (2010) 371–382.
[10] L.W. Schruben, V.J. Cogliano, An experimental procedure for simulation response surface model identification, Communication of the Association for Computing Machinery 30 (8) (1987) 716–730.
[11] J.P.C. Kleijnen, R.G. Sargent, A methodology for fitting and validating metamodels in simulation, European Journal of Operational Research 120 (1) (2000) 14–29.
[12] J. Poropudas, K. Virtanen, Game theoretic validation and analysis of air combat simulation models, IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans 40 (5) (2010) 1057–1070.
[13] R.C. Cheng, S.S. Currie, Optimization by simulation metamodelling methods, in: Proceedings of the 2004 Winter Simulation Conference, Washington, DC, 2004, pp. 473–478.
[14] H.M. Taylor, S. Karlin, An Introduction to Stochastic Modeling, third ed., Academic Press, San Diego, CA, 1998.
[15] V.A. Mattila, K. Virtanen, T. Raivio, Improving maintenance decision making in the Finnish air force through simulation, Interfaces 38 (3) (2008) 187–201.
[16] T. Dean, K. Kanazawa, A model for reasoning about persistence and causation, Computational Intelligence 5 (3) (1990) 142–150.
[17] J. Poropudas, K. Virtanen, Analyzing air combat simulation results with dynamic Bayesian networks, in: Proceedings of the 2007 Winter Simulation Conference, Washington, DC, 2007, pp. 1370–1377.
[18] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufman, San Mateo, CA, 1991.
[19] F.V. Jensen, Bayesian Networks and Decision Graphs (Information Science and Statistics), Springer-Verlag, New York, NY, 2001.
[20] R.E. Neapolitan, Learning Bayesian Networks, Prentice Hall, Upper Saddle River, NJ, 2004.
[21] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley Professional, Upper Saddle River, NJ, 1989.
[22] K.B. Korb, A.E. Nicholson, Bayesian Artificial Intelligence, Chapman & Hall, CRC, Boca Raton, FL, 2004.
[23] M. Diehl, Y.Y. Haimes, Influence diagrams with multiple objectives and tradeoff analysis, IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans 34 (3) (2004) 293–304.
[24] D. Heckerman, Learning in Graphical Models, MIT Press, Cambridge, MA, 1999. pp. 301–354 (Chapter A tutorial on learning with Bayesian networks).
[25] A.M. Law, W.D. Kelton, Simulation Modeling and Analysis, third ed., McGraw-Hill, New York, NY, 2000.

[26] J.S. Milton, J.C. Arnold, Probability and Statistics in the Engineering and Computing Sciences, McGraw-Hill, New York, NY, 1986.
[27] N. Friedman, I. Nachman, D. Peér, Learning Bayesian network structure from massive datasets: The sparse candidate algorithm, in: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden, 1999, pp. 206–215.
[28] D. Heckerman, Bayesian networks for data mining, Data Mining and Knowledge Discovery 1 (1) (1997) 79–119.
[29] W. Buntine, A guide to the literature on learning probabilistic networks from data, IEEE Transactions on Knowledge and Data Engineering 8 (2) (1996) 195–210.
[30] S.K. Andersen, K.G. Olesen, F.V. Jensen, HUGIN – A Shell for Building Bayesian Belief Universes for Expert Systems, Morgan Kaufman, San Francisco, CA, 1990.
[31] Decision Systems Laboratory, GeNIe (graphical network interface). Available from: <http://genie.sis.pitt.edu/>, 2010 (accessed 17.02.10).
[32] R. Davidson, J.G. MacKinnon, Estimation and Inference in Econometrics, Oxford University Press, New York, NY, 1993.
[33] M.I.R. dos Santos, A.M. Porta Nova, Statistical fitting and validation of non-linear simulation metamodels: A case study, European Journal of Operational Research 171 (1) (2006) 53–63.
[34] H. Hamad, S. Al-Hamdan, Discovering metamodels' quality-of-fit for simulation via graphical techniques, European Journal of Operational Research 178 (2) (2007) 543–559.
[35] L.D. Brown, T.T. Cai, A. DasGupta, Interval estimation for a binomial proportion, Statistical Science 16 (2) (2001) 101–133.
[36] G.F. Cooper, The computational complexity of probabilistic inference using Bayesian belief networks (research note), Artificial Intelligence 42 (2–3) (1990) 393–405.
[37] L-3 Communications Analytics Corporation, Vienna, VA, The X-Brawler Air Combat Simulator Management Summary, 2002.
[38] J. Poropudas, K. Virtanen, Influence diagrams in analysis of discrete event simulation data, in: Proceedings of the 2009 Winter Simulation Conference, Austin, TX, 2009, pp. 696–708.
[39] J.P. Kleijnen, D. Deflandre, Validation of regression metamodels in simulation: Bootstrap approach, European Journal of Operational Research 170 (1) (2006) 120–131.
[40] R.A. Howard, J.E. Matheson, Influence diagrams, Decision Analysis 2 (3) (2005) 127–143.
[41] D. Koller, B. Milch, Multi-agent influence diagrams for representing and solving games, Games and Economic Behavior 45 (1) (2003) 181–221.