Building Requirements-Flow Models using Bayesian Networks and Designed Simulation Experiments

Terril N.Hurst[1], Jarom J. Ballantyne[2], Allan T. Mense[3]

Raytheon Missile Systems, Tucson, AZ 85756

Abstract

This paper describes a method for constructing and evaluating Bayesian networks (BNs) as a logically consistent model of a set of derived system requirements. A BN consists of (1) a directed acyclic graph, (2) a set of fully defined states for each node in the graph, and (3) a conditional probability table (CPT) for each of the nodes. Designed simulation experiments are central to the construction and evaluation of BN models to predict system performance given a set of subsystem requirements. Probability estimates in each CPT are first made by mining data from simulation experiments on prior, similar systems, and then—using engineering judgment—the CPT entries are altered to explore options for satisfying top-level requirements of the system under current design. Sensitivity analysis is conducted to prioritize and assign values to each requirement.

A notional weapon kill-chain example illustrates the construction and evaluation of a BN requirements-flow model. Benefits of using Bayesian networks are reported, including the ability to (1) analyze design margin, (2) allocate subsystem tolerances, (3) estimate the achievable upper bound on system performance for a proposed design improvement, and (4) integrate results from designed simulation experiments involving multiple subsystems.

Key Words: Bayesian networks, probabilistic requirements management

1. Introduction

PROBABILITY and logic have a long history of application to rigorous scientific inquiry. The writings of E.T. Jaynes illustrate the benefits of constructing logically consistent probability models.[1] Jaynes advocated Bayesian statistical analysis as a way of combining current, often limited data with external ("prior") knowledge of a specific domain to make logical inferences.[2] This paper introduces an application of Bayesian networks (BNs) and designed simulation experiments (DASE) to systems engineering. Despite the name, either "frequentist" or Bayesian statistics can be employed to construct BNs. The example in Section IV employs the former. In either case, central to the BN paradigm is the definition of *conditional probability*, commonly attributed to Bayes and later popularized by Laplace:

$$\Pr\left(A \mid B\right) = \frac{\Pr(A,B)}{\Pr(B)} \tag{1}$$

where *A* is a simple or compound event of interest and *B* is a second, simple or compound event for which evidence is given. The conditional probability is computed by dividing

[1] Engineering Fellow, Modeling, Simulation & Analysis Center, Bldg 805 M/S C1.

[2] Sr. Systems Engineer II, Signal Processing Center, Bldg M01 M/S 1.

[3] Principal Engineering Fellow, Bldg M02 M/S 5T.

the joint probability of events *A* and *B* by the marginal probability of event *B*, which is found by summing across each of the "nuisance variables" that are not of interest in the conditional probability query. Although *A* and *B* can be represented by *continuous* random variables, in this paper, they are represented by *discrete*—binomial or multinomial—random variables.

System-level requirements are often stated as probabilities and must be decomposed into several lower-level, "derived" subsystem requirements before system- and subsystem-level design work can proceed. In the past, various ad-hoc quantitative and graphical methods have been used to articulate and visualize "flow-down" requirements. Ambiguous references are often made to "probabilities." These methods have proven useful for guiding and prioritizing design decisions. However, when the time arrives to verify compliance with the requirements or to troubleshoot a non-compliant case, issues often arise. These issues can stem from the ambiguous references to probabilities as well as from the difficulty in using the informal methods to satisfy quantitatively specific "what-if" queries.

The high dimensionality of the joint probability distribution of a complex, engineered system makes it difficult to use directly in analysis. However, the concepts of independence, conditional independence, and marginal probability enable straightforward computation of all probabilities of interest without direct reference to the joint probability distribution. Moreover, as illustrated in Sections II and IV, the concept of a causal flow of events enables simplified expression of the joint probability distribution and computation of the desired conditional probability. Once constructed, acausal, diagnostic questions can be asked of a probability-based requirements model. For this reason, the model is called a *requirements-flow model*, emphasizing the inherent inclusion of data-driven feedback for reasoning about system-level implications of specific, subsystem requirements that are under consideration.

Section II briefly reviews the basic elements of Bayesian networks (BNs) and includes several references for finding further details. Section III summarizes the synergy between BNs and the design and analysis of simulation experiments (DASE), which are used to collect data for estimating conditional probabilities within BNs. Section IV describes an example of constructing a BN model of a notional kill chain for an anti-aircraft missile. Section V summarizes the ideas and work accomplished thus far using BNs and DASE for developing requirements-flow models.

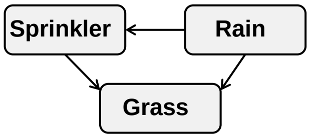
2. Basics of Bayesian Networks

Bayesian networks (BNs) were developed in the 1980s by Pearl and others in the field of artificial intelligence (AI).[3,4] Although AI researchers pursued several paths for automating reasoning, including formal logic, it became clear that another paradigm was needed to mimic the way that humans reason.[5,6] Central to the BN paradigm are the notions of *causality* and what Darwiche called *degree of belief*.[7,8]

Figure 1 illustrates a simplest-possible BN that is commonly used to introduce the elements of a BN and the mathematical concepts used in analyzing it.[9] Each BN has three distinct types of components: (1) a directed, acyclic graph (DAG), where the arcs indicate the cause-effect flow between nodes; (2) a set of precise definitions for the possible states of each node (e.g., what quantity and rate of water constitutes "wet grass" or "rain"); and (3) a set of conditional probability tables (CPTs), one per node in the DAG.

CPT_S (1 parent)

R	S	
	F	T
F	0.6	0.4
T	0.99	0.01



CPT_R (0 parents)

R	
F	T
0.8	0.2

Example query:

$$\Pr\left(R \mid G\right) = \frac{\Pr(G,R)}{\Pr(G)} = \frac{\sum\limits_{X \in \{S, \neg S\}} \Pr\left(G, X, R\right)}{\sum\limits_{\substack{X_1 \in \{S, \neg S\} \\ X_2 \in \{R, \neg R\}}} \Pr\left(G, X_1, X_2\right)}$$

CPT_G (2 parents)

R	S	G	
		F	T
F	F	1	0
T	F	0.2	0.8
F	T	0.1	0.9
T	T	0.01	0.99

Figure 1: DAG and CPTs constituting a simple Bayesian network (nodal state definitions not shown).[9]

For engineered systems, the state definitions are critical, both (a) for using designed simulation experiments to populate each CPT entry as an allocated requirement and (b) to evaluate compliance of each requirement, once the system has been implemented.[10]

After constructing the BN—including populating all CPT entries with values for fully defined states—the BN can be analyzed visually and mathematically. By inspection, any node with no incoming arcs from parent nodes (e.g., Rain in Figure 1) is understood to be *independent*, i.e. having no conditional cases within its CPT. A node with incoming arcs from one or more parent nodes is *conditionally independent* of its non-descendants, given evidence of its parents' states, listed (one column per parent) within the node's CPT. This *local Markov property* is useful in expressing the joint probability distribution of the nodes' states: instead of exhaustively writing down all conditional and marginal probabilities, the product chain rule can be applied, based upon the BN DAG, to express a simplified version of the joint probability distribution.

For a specific query, e.g., Pr(R | G) in Figure 1, there are *nuisance variables* in the joint and marginal probabilities—Pr(G,R) and Pr(G), respectively—across which probabilities must be summed. This query example involves nodes with only two complementary states, {X, ¬X}. Since there is only one nuisance variable {S} in the numerator and two nuisance variables {R, S} in the denominator, manually computing the sums and quotient is straightforward. For the k-node case, each having $m \geq 2$ possible states, some of which are known ("evidence"), mathematical operations quickly become cumbersome, of computational complexity O(m^k). This is where *inference engine* algorithms become useful, either to compute exact or approximate probabilities, depending on the computational complexity of the specific BN.

The work reported in this paper was accomplished using a MATLAB-based tool from Murphy called BNT (Bayesian Network Toolbox) and a JAVA-based tool called SamIam from UCLA's Automated Reasoning Group. Both tools include a suite of inference algorithms; SamIam also includes a graphical editor. Murphy compiled a list of several other BN tools.[11,12]

3. Synergy between BNs and Design & Analysis of Simulation Experiments

The principles and methods of experimental design pioneered by Fisher were later expanded by others for industrial applications, including simulation experiments to accomplish software-intensive systems engineering work.[13,14,15] Subsystem development has benefitted greatly from the design and analysis simulation experiments (DASE). DASE involves defining a *factor space* and one or more *responses* to be observed while sampling *treatments*—i.e. combination of factor *levels*—within the defined factor space. The most common experimental design in DASE involves collecting *space-filling* samples. For BNs, the useful output of DASE is a *summary statistic* estimate of each entry within the CPTs, representing the conditional probability of occurrence of response values for the factor space defined for each BN.

The strengths of probability and logic come together with the strength of statistics and simulation by evaluating a BN requirements-flow model in the following 3-phase (usually iterative) sequence.

(1) Using the BN nodal state definitions, sample proportion estimates are computed for each CPT entry by mining data from space-filling simulation experiments on a system that is similar to the one under current design.

(2) Modifications to these "baseline" CPT entries are considered by subject matter experts, who must weigh the cost and feasibility of subsystem improvements against the potential system performance increase that proposed subsystem improvements/replacements offer.

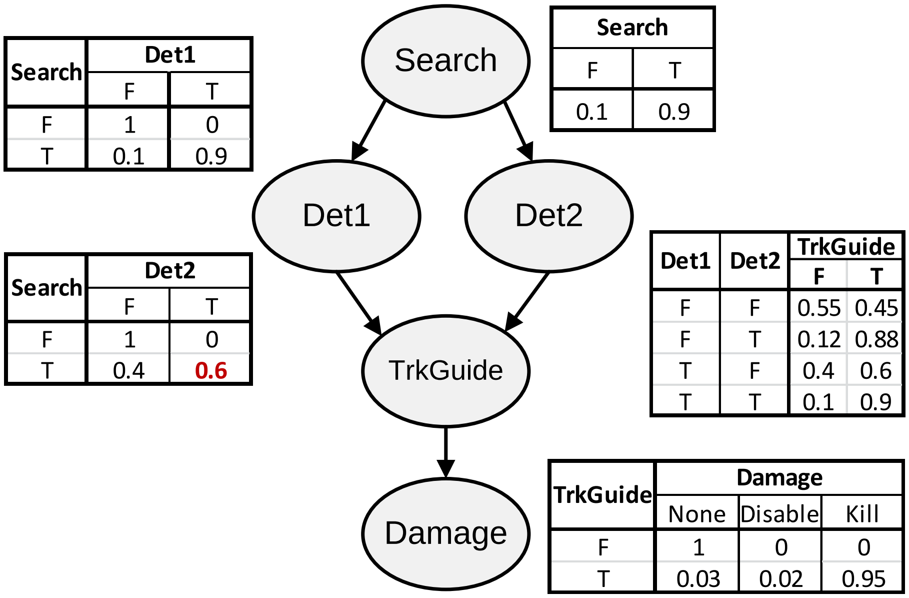
(3) Once designs are completed (i.e. subsystem replacement or improvement decisions are made), more simulation experiments are conducted to verify that the expected system performance improvement has been realized by incorporating the new designs.

In the past, although DASE has been instrumental in improving subsystems, when multiple subsystems have been integrated, it has often become clear that DASE-driven design decisions for each subsystem have resulted in suboptimal system performance and/or cost. Thus, *the synergy between BNs and DASE provides a natural framework for fine-grained, precise requirements definition and performance verification* that is needed for defining and integrating results from multiple simulation experiments.

4. Example Application: A Bayesian Model of a Weapon Kill Chain

Figure 2 illustrates a requirements-flow BN model for a notional kill-chain for an anti-aircraft weapon. The example is a simplified version of an unclassified, open-source example from Ball's textbook, which has been used previously in other published studies.[16,17] Although this section describes only a single, illustrative requirement modification, the BN model could be used to explore many more options.

The BN begins with the conditionally independent Search node, which represents the probability that the in-flight weapon is searching for the aircraft threat. Given a successful Search, the two parallel nodes Det1 and Det2 represent detection of the target using two independent sensors. These sensors could be multiple instances of the same on-board sensor, a single onboard bi-mode sensor, or one on-board and one remote sensor. Given target detection, the probability of target tracking, guiding the missile to the target, and then detonating the missile's warhead is represented by the TrkGuide node. Given successful TrkGuide, warhead detonation leads to one of three possible Damage outcomes: no damage, sufficient damage to disable the target from completing its mission, or killing the target. The Damage node is the only multinomial case; all other nodes are binomial (i.e. having *True/False* states).



Search	Det1	
	F	T
F	1	0
T	0.1	0.9

Search	
F	T
0.1	0.9

Search	Det2	
	F	T
F	1	0
T	0.4	**0.6**

Det1	Det2	TrkGuide	
		F	T
F	F	0.55	0.45
F	T	0.12	0.88
T	F	0.4	0.6
T	T	0.1	0.9

TrkGuide	Damage		
	None	Disable	Kill
F	1	0	0
T	0.03	0.02	0.95

Figure 2: A Bayesian network model for a notional anti-aircraft missile kill chain.

In this simple example, a single CPT entry is modified to increase the weapon's probability of kill, Pr(damage = *Kill*), a marginal probability that hereafter is referred to simply as *Pk*. By observing the CPTs for each node in the graph, we can count the total number of nodal states involved, i.e. *4* x *2 + 3 = 11,* and compute the number of possible combinations of these states, 3^1 *x* 2^4 *= 48*, most of which become nuisance variables to be summed out for any specific query. The CPT entries within each row of a CPT must sum to one, but assignment of the child-node's states usually differ row-by-row. Although not shown in Figure 2, the third element of a BN is the set of precise definitions for each state of each node. Simulation results are converted into sample proportion estimates of each CPT entry via these definitions.

The CPT values in Figure 2 were chosen to illustrate a typical situation that might arise within a requirements-flow study. The situation involves changing subsystem requirements in order to increase a prior weapon's marginal *Pk* from 0.70 to 0.77. A common misunderstanding is to confuse such a *marginal* probability with a *conditional* probability—in this case, confusing *Pk*, which was computed using an inference engine, with Pr(*Damage* = *Kill* | *TrkGuide* = *True*), which was estimated from a simulation experiment and later modified ("allocated") to meet the new *Pk* system requirement.

An opportunity for improving *Pk* is highlighted in the Det2 CPT in Figure 2. If Pr(Det2 = *True* | Search = *True*) can be increased above 0.60, the increased *Pk* value might be met without altering other attributes of the weapon. Consider the values in the TrkGuide CPT. When Det2 = *True* and Det1 = *False* then the conditional probability of TrkGuide is 0.88. On the other hand, when Det2 = *False* and Det1 = *True*, TrkGuide is 0.60. In other words, Det2 success alone leads to higher probability of reaching the target than Det1 success alone, (also illustrated is the higher TrkGuide success that results from *both* detectors being successful, but simultaneously improving both detectors was not considered in this example).

To further explore the sensitivity of *Pk* to improving either Det1 or Det2, we consider what would happen, assuming different values of detection success. Figure 3 illustrates how *Pk* changes as values change (one sensor at a time, holding the other sensor constant) for Pr(Det1 = *True* | Search = *True*) and Pr(Det2 = *True* | Search = *True*). The sensitivity for Det1 is +0.062, compared to +0.268 for Det2. Thus, work to improve Pr(Det2 = *True* | Search = *True*) will provide a higher incremental return for improving *Pk*. Note that it will likely be easier to implement a change in the initially lower Det2 value than Det1, since the latter is already much closer to 1.0.

SamIam's inference engine was used to determine the value for Pr(Det2 = *True* | Search = *True*) that would meet the new *Pk* requirement of 0.77. The result was Pr(Det2 = *True* | Search = *True*) = 0.85. This prediction was then verified by running another simulation experiment, using the improved Det2 sensor having the increased CPT value.

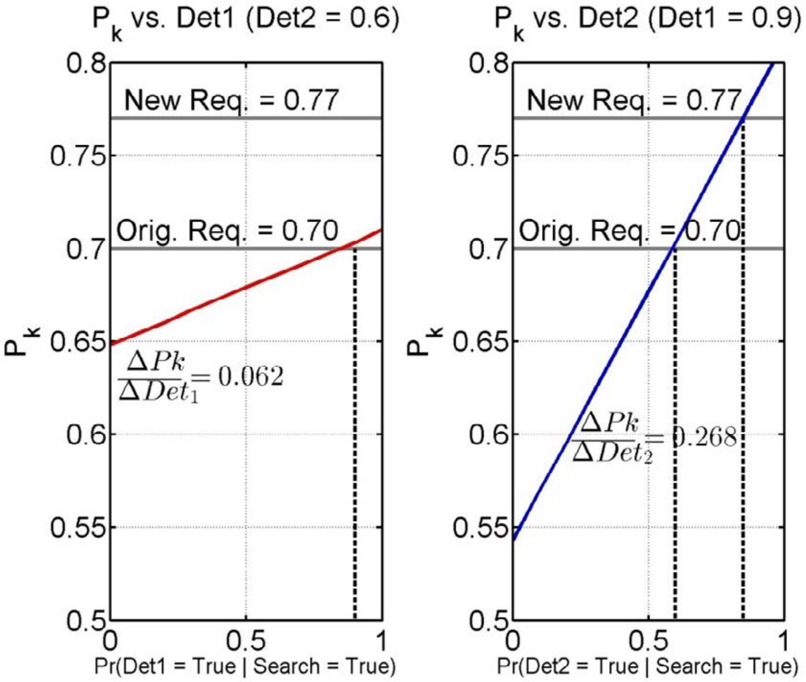


Figure 3: Sensitivity comparison between conditional sensor detection probabilities.

5. Summary and Conclusions

When combined with DASE, Bayesian networks (BNs) provide a way to evaluate subsystem requirements in a quantitative, comprehensive, manner. The power of logic and a rigorous probability model enable immediate evaluation of "what-if" queries when considering subsystem improvements. The requirements-flow model enables immediate sensitivity analysis and upper-bound estimates on the likely achievable gains of a proposed improvement, *prior to expending the development effort* to implement the proposed improvement. Moreover, BNs provide a natural framework for designing and evaluating results from multiple simulation experiments to mitigate the tendency for suboptimization. Tools exist to develop BNs, so the challenge is to educate engineers in the promise of BNs and the use of these tools.

References

[1]Jaynes, E.T., *Probability Theory: The Logic of Science*, Cambridge (2003).

[2]Jaynes, E.T., “Confidence Intervals vs. Bayesian Intervals,” *Foundations of Probability Theory, Statistical Inference, and Statistical Theories of Science*, Vol II, 175-257, Harper & Hooker (eds., 1976). Also available at http://bayes.wustl.edu/etj/articles/confidence.pdf

[3]Pearl, J., “Bayesian Networks: A Model of Self-Activated Memory for Evidential Reasoning,” *Proceedings*, Cognitive Science Society, Irvine, CA, pp., 329-334 (1986). Also available at http://ftp.cs.ucla.edu/tech-report/198_-reports/850017.pdf

[4]Ben-Gal, I., “Bayesian Networks,” in Ruggeri, F. et al, *Encyclopedia of Statistics in Quality & Reliability,* Wiley & Sons (2007). Also available at www.eng.tau.ac.il/~bengal/BN.pdf

[5]Genesereth, M.R. and N.J. Nilsson, *Logical Foundations of Artificial Intelligence*, Morgan Kaufmann (1987).

[6]Brooks, R.A., “Intelligence without Reason,” *Proceedings*, International Joint Conference on Artificial Intelligence (1991). Also available at http://people.csail.mit.edu/brooks/papers/AIM-1293.pdf

[7]Pearl, J., *Causality: Models, Reasoning, and Inference,* 2nd ed., Cambridge (2009).

[8]Darwiche, A., *Modeling and Reasoning with Bayesian Networks,* Cambridge (2009).

[9]Wikipedia, “Bayesian Networks,” http://en.wikipedia.org/wiki/Bayesian_network

[10]Hurst, T.N., C.F. Pouchet and A.T. Mense, “Verifying System Performance Using Designed Simulation Experiments,” *Proceedings*, 2012 U.S. Army Conference on Applied Statistics.

[11]Murphy, K., “Bayes Net Toolbox for Matlab,” http://code.google.com/p/bnt/

[12]Darwiche, A. *et al*, “SamIam: Sensitivity Analysis, Modeling, Inference, and More,” available at http://reasoning.cs.ucla.edu/samiam/

[13]Montgomery, D.C., *Design and Analysis of Experiments*, 8th ed., Wiley (2012).

[14]Sanchez, S., “Work Smarter Not Harder: Guidelines for Designing Simulation Experiments,” Proceedings, 2005 Winter Simulation Conference.

[15]Hurst, T.N. et al, “Designed Simulation Experiments, Part 2: DOE for the Digital Age,” *Proceedings*, 2008 AIAA Conference on Modeling and Simulation Technologies.

[16]Ball, R. E. (2003), *The Fundamentals of Aircraft Combat Survivability Analysis and Design*, 2nd ed., American Institute of Aeronautics (2003).

[17]Smith, R.M, *Using Kill-Chain Analysis to Develop CONOPS to Defend against Anti-Ship Cruise Missiles*, Dissertation, Naval Post Graduate School (2010).