Bayesian Belief Networks in Reliability

Luigi Portinale

Luigi Portinale, Ph.D.
Department of Computer Science
Universita' del Piemonte Orientale "A. Avogadro"
Viale T. Michel 11
15121 Alessandria (Italy)
Internet (e-mail): luigi.portinale@unipmn.it

SUMMARY & PURPOSE

This tutorial will provide insights about the use of probabilistic graphical models based on the Bayesian belief network formalism for reliability and dependability analysis. Bayesian networks (BNs) have become a popular tool for modeling many kinds of statistical problems. Recently, we have also seen a growing interest for using BNs in the reliability analysis community. In this tutorial, we will discuss the properties of the modeling framework that make BNs particularly well suited for reliability applications. Modeling as well as analysis issues are discussed, together with a selection of case studies that will be used to explain the concepts that are introduced. We finally point to ongoing research that is relevant for practitioners in reliability.

Luigi Portinale, Ph.D.

Luigi Portinale is a Full Professor in Computer Science and Chairman of the Computer Science Department of the University of Piemonte Orientale “A. Avogadro” in Italy. He received his Ph.D. and Laurea degree all in Computer Science from the University of Torino (Italy). His main research interests are in the Artificial Intelligence (AI) area, with particular attention to intelligent decision support systems and probabilistic uncertain reasoning. Concerning the latter, he is currently involved in a research effort aimed at proposing and evaluating Probabilistic Graphical Models (like Bayesian Networks) as tools for reliability modeling and analysis. He is a Member of IEEE Computer Society, of the Association for the Advancement of Artificial Intelligence (AAAI) and of the Italian Association for Artificial Intelligence (AIIA). He has been program committe member, area chair and program chair of several computer science conferences.

Table of Contents

1. INTRODUCTION

Over the last decade, Bayesian networks (BNs) have become a popular tool for modeling many kinds of statistical problems. In particular, we assisted to a growing interest for using BNs in the reliability analysis community [14]. As the quantities in reliability studies are uncertain, the end result should be a mathematically sound statistical model describing a set of random variables. Furthermore, such models require a set of parameters to be fully specified, and either statistical data or expert judgment must be used to estimate them. One would like the formalism to minimize the number of parameters required by the model. Finally, the model must be represented such that the quantities we are interested in can be calculated efficiently. All of these requirements have led to reduced focus on traditional frameworks like fault trees (FTs), and more flexible modeling frameworks have received increased attention. One such framework which has gained popularity is the set of Bayesian Network (BN) models, originated in the field of artificial intelligence. Features regarding both modeling and analysis of reliability block diagrams and fault-trees have been compared to BNs, as well as state-space models; results have shown that BNs have significant advantages over the traditional frameworks. We will discuss some of these advantages in detail by way of specific case studies.

The tutorial is organized as follows: we will start by recalling most relevant notions about dependability and reliability that will be used throughout the tutorial, we then move to presenting the basics of the BN framework, by considering the BN modeling issues ad how they can be addressed by exploiting classical reliability modeling methods (for instance by showing how BN model can be automatically obtained from FT-based models). We then switch to the BN framework calculation algorithms, by showing how they can be exploited for reliability analysis. We will outline how dynamic system evolutions can be taken into account by considering dynamic generalizations of the basic BN formalism (Dynamic Bayesian Networks) and how they relate to dynamic generalization of FT like Dynamic Fault Trees (DFT). We will finally present a set of case studies ranging from simple reliability applications to more sophisticated approaches in the field of autonomous spacecraft FDIR.

1.1 Notation and Acronyms

FT	fault tree
FTA	fault tree analysis
TE	top event of a FT
DFT	dynamic fault tree
RBD	reliability block diagram
CTMC	continuous time Markov chain
DTMC	discrete time Markov chain
PN	Petri net
GSPN	generalized stochastic Petri net
AI	artificial intelligence
PGM	probabilistic graphical model
BN	Bayesian network
DBN	dynamic Bayesian network
GCTBN	generalized continuous time Bayesian network
JT	junction tree
CPT	conditional probability table
DAG	directed acyclic graph
FDIR	fault detection, identification and recovery
EU	expected utility
t, t_j	time
$X(t)$	system status function
$Pr[\bullet]$	probability function
$Pr_t[\bullet]$	probability function at time t
$E[\bullet]$	expected value
$A(t)$	availability
A	limiting availability
$MTTF$	mean time to failure
$MTTR$	mean time to repair
λ	failure rate
μ	repair rate
$R(t)$	reliability function
$U(t)$	unreliability function
X, X_i	discrete random variable
$pa(X)$	parent set of X
$nd(X)$	non descendant of X
$\perp\!\!\!\perp$	independence relation

2. DEPENDABILITY ISSUES

The term dependability is usually adopted to identify the ability of a system to deliver a service that can justifiably be trusted. It is currently considered as a composite concept with different features (see [15]). In particular, the following attributes are defined for dependability and security:

- *Reliability*: the continuity of correct service
- *Availability*: the readiness of correct service
- *Maintainability*: the ability to undergo modifications and repairs
- *Safety*: the absence of catastrophic consequences

Reliability and availability are closely related attributes that can be more formally defined as follows [25]; let the system status function be

$$X(t) = \begin{cases} 1 \text{ if system up at } t \\ 0 \text{ if system down at } t \end{cases}$$

then the *availability function* is defined as

$$A(t) = \Pr[X(t) = 1] = E[X(t)]$$

and the *limiting availability* is

$$A = \lim_{t \to \infty} A(t) = \frac{MTTF}{MTTF + MTTR}$$

While availability refers to a system potentially undergoing maintanance and repair, the reliability *R(t)* is intended to measure the probability that the system is able to perform the required function, in the interval *(0,t)*, given the stress and

environmental conditions. An example of a widely used *reliability function* is the negative exponential distribution

$$R(t) = e^{-\lambda t}$$

where a constant failure rate λ is assumed (so that $MTTF=1/\lambda$). One can also talk about the *unreliability function* defined as $U(t)=1-R(t)$, providing the probability that the system is not performing the required function at time t (i.e. that the system is failed at t).

It is worth noting that, in case the system is not repairable (i.e. when there is no possibility of return to the correct behavior when failed), then $A(t)=R(t)$. On the contrary, the availability will depend also on the repair capabilities; for example if an exponential distribution with constant repair rate μ is assumed, then we can define $\Pr[X(t)=0 \mid X(0)=0] = e^{-\mu t}$ (corresponding to $MTTR=1/\mu$). It should be clear that, if no repair is available (e.g. $\mu=\infty$), then the limiting availability will be 0 (which is exactly what happens to the limiting reliability).

Concerning the so-called *threats* of the dependability taxonomy, a distinction is often performed among faults, errors and failures. A *failure* is defined to be a system deviation from the correct/expected service; different failure modes can be in principle be identified, corresponding to different ways of deviating from correct behavior. An *error* is a discrepancy between the intended behavior of a system component and its actual behavior; so the difference between error and failure stands in the boundaries on which they are defined (whole system for failures, single components for errors). Finally, a *fault* is a defect in the system, leading to a failure (a cause of a failure). These concepts are often assembled in the so called *fault-error-failure chain*: a fault, when activated, can lead to an error (which is an invalid state) and the invalid state generated by an error may lead to another error or a failure (which is an observable deviation from the specified behavior at the system boundary).

Reliability (or availability) techniques dealing with the above notions are usually model-based techniques. Model types can be roughly categorized as follows:

- Combinatorial models
- State-space models
- Local dependency models

Combinatorial models have poor modeling power matched with high analytical tractability and assume that components are statistically independent; example of such models are *fault tree* (FT)[24] and reliability block diagrams (RBD)[25].

State-space models rely on the specification of the whole set of the possible system states and of the possible transitions among them, for these reasons they suffer of the state space explosion problem; examples are Markov chains (either DTMC or CTMC)[25] and Petri nets (PN)[16].

More interesting for the aims of this tutorial, are the local dependency models, where component dependencies are specified only at specific places (i.e. locally). One of the first attempt to exploit local dependencies was the definition of an extended version of the FT language, namely Dynamic Fault Tree (DFT)[9]. In a DFT, in addition to standard FT gates, special dynamic gates are introduced (see Fig. 1):

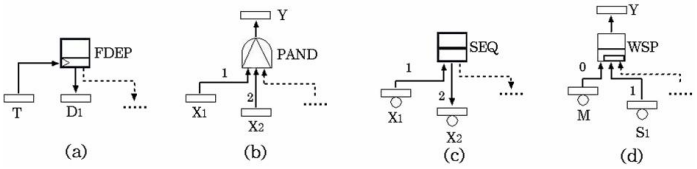


Fig. 1

- *Functional Dependency gate* (FDEP) – given as input events a trigger event T and a set of dependent events D_1, ..., D_m, the dependent events are forced to occur when the trigger event occurs;
- *Priority And gate* (PAND) - given X_1, ..., X_n as input events and Y as output event, Y fails if all X_1, ..., X_n have occurred and only in a specified order.
- *Sequence Enforcing gate* (SEQ) - given X_1, ..., X_n as input events and Y as output event, X_1, ..., X_n are forced to occur in a specified order; Y corresponds to the state of X_n (actually, the gate can be modeled as a special case of the next WSP gate).
- *Warm Spare gate* (WSP) - this gate models the presence of a set of warm spare components able to replace a main component when it fails; warm spares can be in three states: dormant (or stand-by), working, failed; the spare failure rate changes depending on its current state: if the failure rate of the spare is λ in the working state, $\alpha\lambda$ is its failure rate in the dormant state, with $0<\alpha<1$; α is called *dormancy factor*. The input events of this gate are the events corresponding to the failure of the main component and the events corresponding to the failure of the spares; the output event occurs if the main component fails and there are no available spares to replace it.

Due to dependencies, DFTs need state space analysis; however, it can be limited to dynamic modules that can be identified as sub-trees and separately solved through CTMC [11] or GSPN [4]. The results of module analysis can then be used as inputs for a standard FTA[10].

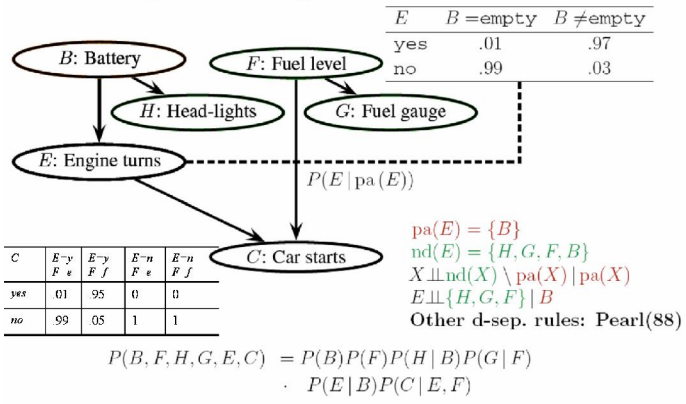
More recently, another set of formalisms based on local dependencies that have received a lot of attention in the reliability community is that of *Probabilistic Graphical Models* (PGM), whose main representatives are *Bayesian Networks* (BN) and *Dynamic Bayesian Networks* (DBN). Next section introduces the basics of such formalisms.

3. PROBABILISTIC GRAPHICAL MODELS: BAYESIAN NETWORKS AND DYNAMIC BAYESIAN NETWORKS

Bayesian (or Belief) Networks (BN) are a widely used formalism from AI (Artificial Intelligence) for representing uncertain knowledge in probabilistic systems, applied to a variety of real-world problems [20]. BN are defined by a directed acyclic graph in which (discrete) random variables are assigned to each node, together with the quantitative conditional dependence on the parent nodes (Conditional Probability Table or CPT). More formally, a BN is a pair N<G,P> where:

- G is a DAG whose nodes $X_1,...X_n$ are a set of discrete random variables and where an edge from X to Y means that Y depends on X (e.g. X causes Y);
- P is a probability distribution over $X_1,...X_n$ such that $\Pr[X_1,...X_n] = \prod_{i=1}^{n} \Pr[X_i \mid pa(X_i)]$ where *pa(X)* is the set of parent nodes of X in the DAG G.

The advantage of the BN representation is that one can get the joint probability of the variables of the model, by specifying a set of conditional probabilities local to X and to its parents (the CPT). Fig. 2 shows an example of a BN.



E	B =empty	B ≠empty
yes	.01	.97
no	.99	.03

C	E=y F=e	E=y F=f	E=n F=e	E=n F=f
yes	01	95	0	0
no	99	05	1	1

Fig. 2 (by H. Langseth)

If we indicate as *nd(X)* the set of non descendant nodes of X, one basic assumption underlying the structure of a BN is that a variable X is independent from other non descendant variables, given its parents (written $X \perp\!\!\!\perp nd(X) \backslash pa(X) | pa(X)$). Several other independence rules can be extracted by simply looking at the structure of a BN, called d-separation rules [20].

Concerning the quantitative aspects of a BN, several probabilistic computations (called *inferences*) are possible, given that a BN can be used to answer any probabilistic query of the type *Pr[Q|e]*, where *e* is an instantiation of any set of variables in the net, called the *evidence*, and *Q* is a set of queried variables of interest. In particular we can perform:

- Predictive inference (*Pr[effect|cause]*)
- Diagnostic inference (*Pr[cause|effect]*)
- Combined inference (*Pr[intermed|cause, effect]*)

An example of predictive inference on the model of Fig. 2 is the computation of *Pr[C|B=empty]*; a diagnostic inference is the computation of *Pr[B|C=yes]*; a combined inference is the computation of *Pr[E|B=empty,C=yes]*.

Different classes of algorithms have been developed for performing inference on a BN, ranging from exact computation schemes to approximated ones. In the first category, the most important approach is based on the compilation of the network structure into a secondary structure called *Junction* (or *Join*) *Tree* (JT), derived from structural manipulation of the original DAG. The advantage is that inference can be performed by considering the joint probabilities (called *potentials*) of specific subsets of the BN variables, which are identified with the nodes of the JT (called *clusters*)[13]. Approximated inference can be performed by resorting to *stochastic simulation* techniques, where net's variables are sampled from the net's distribution according to specific orders, and posterior probabilities are estimated on the basis of the performed simulation runs; the most interesting techniques is perhaps the *Gibbs sampling* applied to BN, providing fast convergence in many cases [20].

However, BN are a static model, where time is not explicitly taken into account. To address this point, an extension to BN, called *Dynamic Bayesian Network* (DBN) has been proposed as a different type of PGM.

A DBN [7,19] is a discrete time model which is essentially a factorization, over a set of variables, of the states of a DTMC. Fig. 3 reports a simple example of a DBN and related notions.

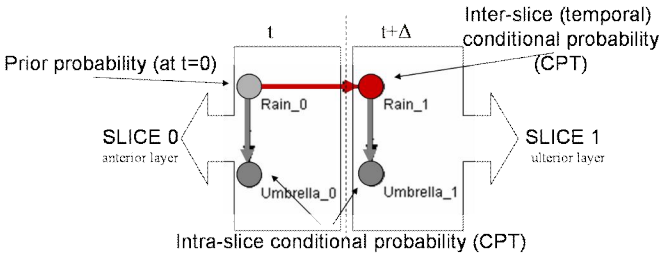


Fig. 3

A time discretization step Δ is defined, and a Markovian assumption allows to consider only 2 time slices (the so called *anterior* and *ulterior* layers) for the model (which is then also called 2TBN – 2-slices Temporal BN). In a canonical representation, intra-slice dependencies of the anterior layer can be omitted [2].

In a DBN several inference schemes can be performed as reported in Fig. 4;

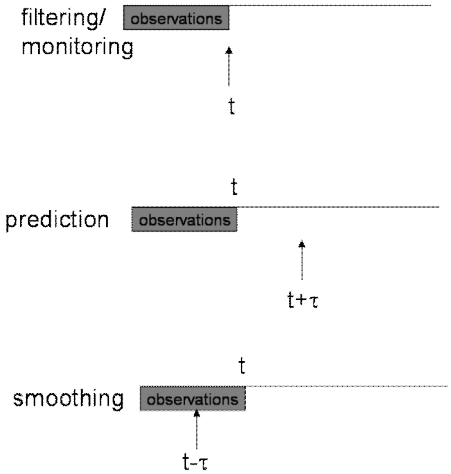


Fig. 4

filtering or *monitoring* corresponds to computing the posterior probability of any network variable at time *t*, given evidence up to time *t*; *prediction* corresponds to compute the probability of any network variable at time $t+\tau$ ($\tau>0$), given evidence up to time *t*; smoothing corresponds to compute the probability of any network variable τ instants in the past, given evidence up to time *t*. While monitoring and prediction

are related to on-line analysis of the system behavior, smoothing is more related to ex-post diagnostic analysis, reconstructing what may have occurred in the past, given all the information available at the current analysis time.

Also in the case of DBN, both exact as well as approximated inference algorithms are available to perform the above tasks; an exact JT based algorithm called *1.5JT* has been proposed by Murphy [19], exploiting large part of the machinery used by static JT algorithms. From the approximation side, a parametric JT based algorithm called *BK* is due to Boyen and Koller [Boi98]: the idea is to have a spectrum of approximations depending on the input parameters, that are set of nodes defining a partition of some net variables. Murphy's algorithm can be seem as a special case of BK, when the input is the whole set to be partitioned (the so called *interface nodes*). Again, stochastic simulation can also be employed on DBN for inference: particularly relevant are the algorithms called *particle filtering*, which avoid to consider samples with very low probability to be propagated, by resampling the population of samples according to their probability at each time step [8].

4. FROM FAULT TREES TO BAYESIAN NETWORKS

A natural way of evaluating the potential impact of BN in reliability analysis, is to relate BN modeling and analysis to FTA. BN may improve both the modeling and the analysis power wrt FT; concerning modeling, local conditional dependencies, probabilistic gates, multi-state variables, dependent failures and uncertainty in model parameters can be naturally addressed in the BN framework; from the analysis point of view, any probabilistic computation involving the network variables can be addressed, both predictive or diagnostic. Any FT can be transformed into a corresponding BN, by creating a binary BN node for each event in the FT, by setting the probability of BN root nodes (corresponding to basic events in the FT) to the probability of failure at the current analysis time *t*, and by building a CPT (with 0-1 probability entries) for any BN node corresponding to events which are output of a FT gate [1]. From the analysis point of view, we can compute the system unreliability (or unavailability if repair of components is allowed) by simply computing the unconditioned probability of the FT's Top Event at the analysis time t ($Pr_t[TE]$); the criticality of each component if a failure is observed can be obtained by considering the posterior probability of each component *C* given the system failure (Fussell-Vesely importance) at time *t*, in other words $Pr_t[C|TE]$); finally, we can also compute the criticality of sets of components at time *t* with queries like $Pr_t[C_1, ...C_k|TE]$ and looking for the most probable configurations of the components, leading to a failure (with a somewhat generalization of the concept of cut-sets).

More importantly, the use of BN can greatly improve the modeling power in at least the following directions: the use of probabilistic gates, the use of multi-state variables, the modeling of sequentially dependent faults and the introduction of uncertainty on the model parameters. Fig. 5 shows how a probabilistic gate modeling imperfect coverage can be modeled in a BN; value *F* stands for *failed*, *W* stands for *working* and *c* is a coverage factor $0 \le c \le 1$; when $c=1$ (perfect coverage) we get an AND gate, while if $c=0$ we get an OR gate.

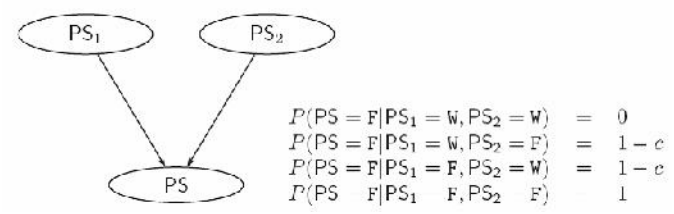


Fig. 5

Multi-state variables cannot be modeled in an ordinary FT. Fig. 6 shows how they can be usefully exploited for modeling sequentially dependent faults.

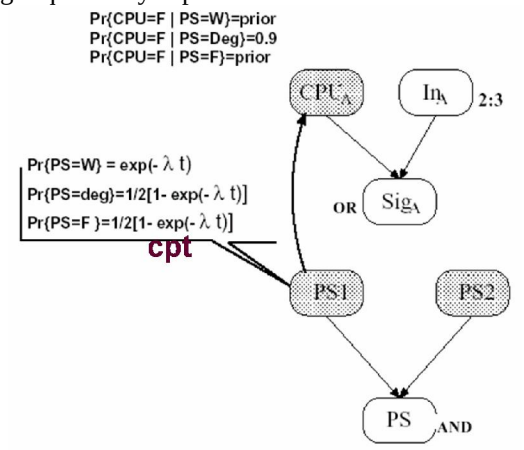


Fig. 6

In this example, a power supply device PS_1, when behaving in degraded mode `Deg`, may cause (with *.9* probability) a fault on component CPU_A.

Finally, uncertainty about some model parameters can be modeled by introducing new parameter nodes ϕ, in order to quantify such uncertainty. In Fig. 7 it is shown how to model the fact that we know that the actual failure rate of a component, usually considered to be equal to λ, may also have a decrease factor of *.9* or an increase factor of *1.1*.

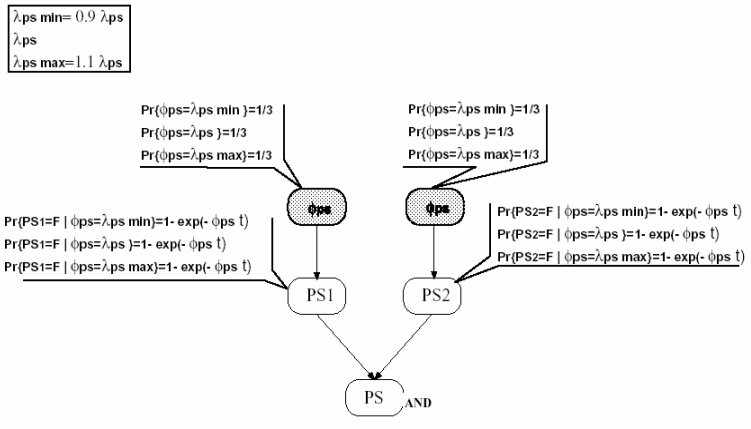


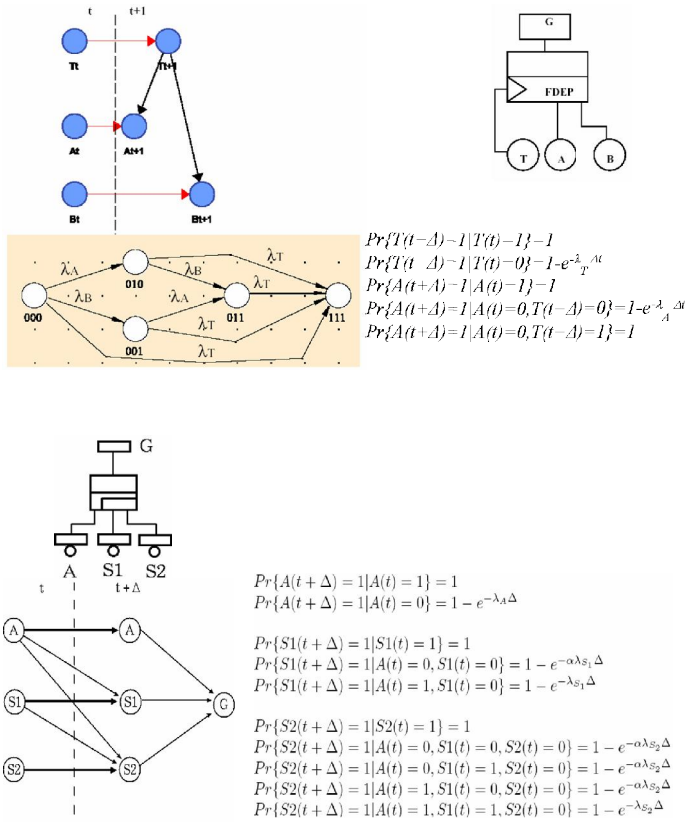
Fig. 7

A more sophisticated solution can also be devised with this approach, for example by considering a *Beta* or *Gamma*

distribution for the values of the failure rates, with expected value equal λ (of course, the adopted *Beta* or *Gamma* distribution should be adequately discretized on the parameter nodes ϕ).

5. FROM DYNAMIC FAULT TREES TO DYNAMIC BAYESIAN NETWORKS

As we mentioned in section 2, DFT generalize FT by allowing special dynamic dependencies, while still retaining the binary nature of FT. As BN have been shown to improve both the modeling and the analysis power of FT, so DBN can be successfully employed to show the same with respect to DFT. In particular, by following a modular approach, we can convert each gate of a DFT into a DBN fragment, and then combine such fragments into the final DBN model[17]. Fig. 8 shows how FDEP, WSP and PAND gates can be translated into DBN fragments.

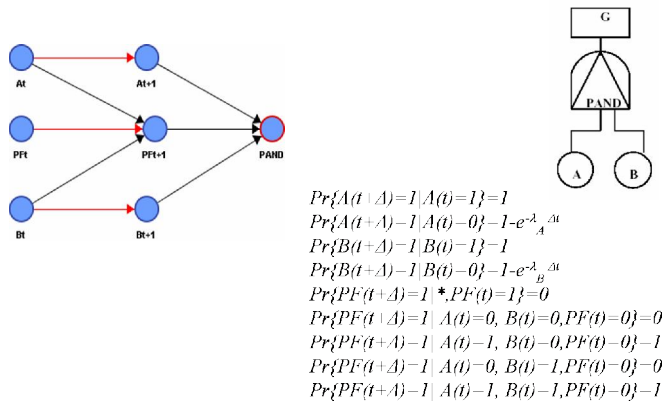


$Pr\{T(t+\Delta)=1|T(t)=1\}=1$
$Pr\{T(t+\Delta)=1|T(t)=0\}=1-e^{-\lambda_T \Delta t}$
$Pr\{A(t+\Delta)=1|A(t)=1\}=1$
$Pr\{A(t+\Delta)=1|A(t)=0,T(t+\Delta)=0\}=1-e^{-\lambda_A \Delta t}$
$Pr\{A(t+\Delta)=1|A(t)=0,T(t+\Delta)=1\}=1$

$Pr\{A(t+\Delta)=1|A(t)=1\}=1$
$Pr\{A(t+\Delta)=1|A(t)=0\}=1-e^{-\lambda_A \Delta}$

$Pr\{S1(t+\Delta)=1|S1(t)=1\}=1$
$Pr\{S1(t+\Delta)=1|A(t)=0,S1(t)=0\}=1-e^{-\alpha\lambda_{S_1}\Delta}$
$Pr\{S1(t+\Delta)=1|A(t)=1,S1(t)=0\}=1-e^{-\lambda_{S_1}\Delta}$

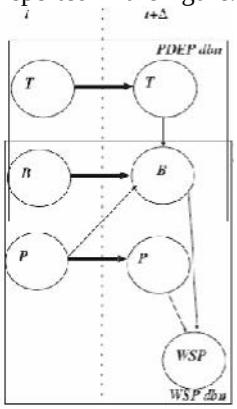
$Pr\{S2(t+\Delta)=1|S2(t)=1\}=1$
$Pr\{S2(t+\Delta)=1|A(t)=0,S1(t)=0,S2(t)=0\}=1-e^{-\alpha\lambda_{S_2}\Delta}$
$Pr\{S2(t+\Delta)=1|A(t)=0,S1(t)=1,S2(t)=0\}=1-e^{-\alpha\lambda_{S_2}\Delta}$
$Pr\{S2(t+\Delta)=1|A(t)=1,S1(t)=0,S2(t)=0\}=1-e^{-\alpha\lambda_{S_2}\Delta}$
$Pr\{S2(t+\Delta)=1|A(t)=1,S1(t)=1,S2(t)=0\}=1-e^{-\lambda_{S_2}\Delta}$



$Pr\{A(t+\Delta)=1|A(t)=1\}=1$
$Pr\{A(t+\Delta)=1|A(t)=0\}=1-e^{-\lambda_A \Delta t}$
$Pr\{B(t+\Delta)=1|B(t)=1\}=1$
$Pr\{B(t+\Delta)=1|B(t)=0\}=1-e^{-\lambda_B \Delta t}$
$Pr\{PF(t+\Delta)=1|*,PF(t)=1\}=0$
$Pr\{PF(t+\Delta)=1|A(t)=0,B(t)=0,PF(t)=0\}=0$
$Pr\{PF(t+\Delta)=1|A(t)=1,B(t)=0,PF(t)=0\}=1$
$Pr\{PF(t+\Delta)=1|A(t)=0,B(t)=1,PF(t)=0\}=0$
$Pr\{PF(t+\Delta)=1|A(t)=1,B(t)=1,PF(t)=0\}=1$

Fig. 8

The combination of each fragment is then performed, at the structural level, by merging common nodes from the different parts. More attention has to be paid to the merging of CPTs from different fragments on the same node; the well-known notion of causal independence [12] can then be exploited in this case, by considering the contribution of each fragment as conditionally independent from the others and by exploiting combination rules like the Noisy-Or[20,21] or the MSP[18]. For example, in the MSP (Most Severe Prevailing) interaction, the maximum of the entries to be combined is assumed as a results. An example is provided in Fig. 9; here it is modeled the integration of a fragment from an FDEP gate with a fragment from a WSP gate, sharing a common node B at the ulterior layer. The original CPTs as well as the resulting one are also reported in the figure.



Probability of failure of $B(t+\Delta)$ in the PDEP DBN

$B(t)$	$T(t+\Delta)$	Failure of $B(t+\Delta)$
0	0	0.05
0	1	0.8
1	0	1
1	1	1

Probability of failure of $B(t+\Delta)$ in the WSP DBN

$B(t)$	$P(t)$	Failure of $B(t+\Delta)$
0	0	0.05
0	1	0.1
1	0	1
1	1	1

Probability of failure of $B(t+\Delta)$ in the combined network

$B(t)$	$T(t+\Delta)$	$P(t)$	Failure of $B(t+\Delta)$
0	0	0	$0.05 = \max(0.05, 0.05)$
0	0	1	$0.1 = \max(0.05, 0.1)$
0	1	0	$0.8 = \max(0.8, 0.05)$
0	1	1	$0.8 = \max(0.8, 0.1)$
1	0	0	$\max(1, 1)$
1	0	1	$\max(1, 1)$
1	1	0	$\max(1, 1)$
1	1	1	$\max(1, 1)$

Fig. 9

6. CASE STUDIES

In this tutorial, several case studies involving BN or DBN inference will be presented. In the present notes, a couple of them will be shortly described.

6.1 Cascading Failures

This case study is related to the analysis of a power grid where failure may propagates over the lines[5]. The scenario refers to the IEEE118 Bus Test Case [3] whose scheme is reported in Fig. 10. The main idea is to model the power grid scheme into a series-parallel diagram (as shown in the lower part of Fig. 10) and then to convert it into a DBN. In the model 3-state variables are needed, since each line can be in *working*, *overloaded* or *outaged* (failed) status. This means that the series-parallel diagrams are a generalization of standard boolean RBD with an ordering of the elements and some specific propagation rules explained in the following.

The normal state is the initial one; an element *e* (that may be a single line or a module composed by more lines) becomes overloaded as a consequence of the outage of another element

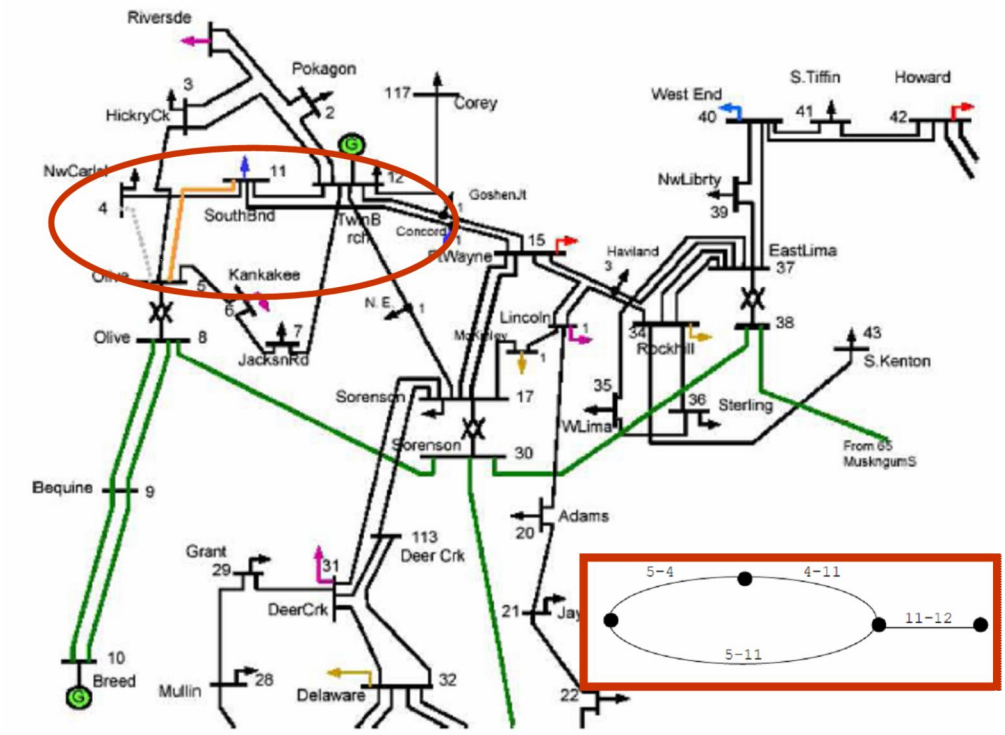


Fig. 10

e' having some influence on *e* according to the structure of the power grid; then, the overload may be propagated from *e* to other elements. The outage state can be reached from both the normal and the overload state: in the case of lines, such a state transition is a stochastic event ruled by some probability distribution (we assume an exponential distribution); the state of a module instead, depends on the state of its internal elements. Moreover, we assume that an outaged line can be repaired; such an event is still stochastic and determines the normal state of the line and possibly of other elements influenced by the repaired line.

If we consider a module M composed by the elements (lines or modules) $e_1,...,e_n$, we suppose that different kinds of dependency hold among $e_1,...,e_n$:

(1) if any element (line or module) e_i of M is overloaded for any reason, then the overload has to be propagated to the following element e_{i+1}. If on the contrary, e_i is in the normal state, such state has to be propagated to e_{i+1};

(2) if M is a series module, the direct consequence of the outage of an element e_i of M, is the propagation of the normal state to e_{i+1} (e_{i+1} becomes isolated: it cannot transport any power). The repair of an element e_i propagates the normal state to $e_{i+1},...,e_n$ because of (1). Finally, the series module M is in the normal state if all the elements $e_1,...,e_n$ are normal; M is in the outage state if at least one element among $e_1,...,e_n$ is outaged; M is in the overload state in any other case;

(3) if M is a parallel module, if at least one element among $e_1,...,e_n$ is outaged, the overload state is propagated to the other elements of M. If instead,

no elements among $e_1,...,e_n$ are outaged as a consequence of a repair, and M is not the following element of an overloaded one, then the normal state is propagated to e_1 (and consequently to $e_2,...,e_n$ as well). The parallel module M is in the outage state if all the elements e_1,y,e_n are outaged; M is in the overload state if at least one element among e_1,y,e_n is overloaded; M is in the normal state in any other case.

Fig. 11 shows the possible state transitions by a line.

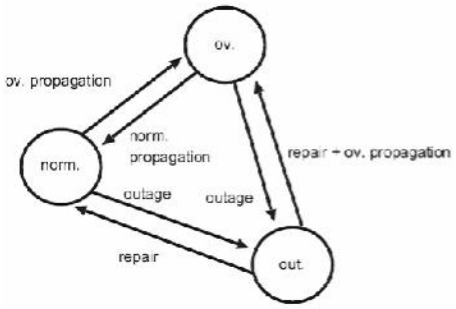


Fig. 11

Fig. 12 shows the DBN obtained by converting the series-parallel diagram in the lower-right rectangle of Fig.12.

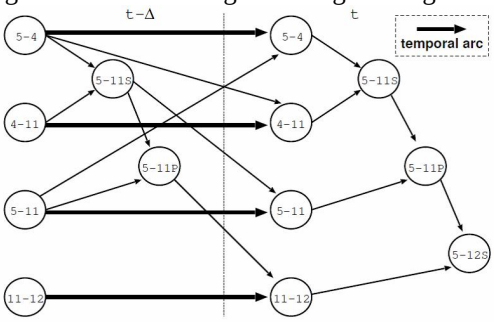


Fig. 12

In the example we consider exponentially distributed failure rates: if the line is not outaged, the outage rate can be λ, $\alpha\lambda$ or $\beta\lambda$; if the line is outaged, the repair rate is μ. The choice of the outage rate depends on the state of the line in t: if normal then the rate is λ, if overloaded directly caused (in t-Δ) by the overload of another element, then is $\alpha\lambda$, if overloaded directly caused by the outage of another element, then is $\beta\lambda$ (with $\beta>\alpha>1$). As an example the following Table 1 reports the CPT for the states outaged (failed) of line 5-4 of Fig. 12.

5-4 failed at t	*5-4* failed at $t-\Delta$	*5-4* over. at $t-\Delta$	*5-4* work. at $t-\Delta$
5-11 failed at $t-\Delta$	1	$1\text{-}e^{-\beta\lambda\Delta}$	0
5-11 over. at $t-\Delta$	1	$1\text{-}e^{-\alpha\lambda\Delta}$	0
5-11 work. at $t-\Delta$	1	*	$1\text{-}e^{-\lambda\Delta}$

Table 1

Notice that the entry indicated as "*" shows an impossible case (line 5-4 overloaded and line 5-11 working), so the actual CPT value in this entry is indifferent. CPT for other states of line 5-4 are set similarly.

Quantitative results can then be obtained through standard DBN inference as discussed in previous sections. Fig. 13 reports some filtering results concerning a different part of the power grid of Fig.10 (i.e. lines relative to nodes 11, 12, 15 and 17); they show the probability of overload and outaged states of some lines and modules by considering the following observations:

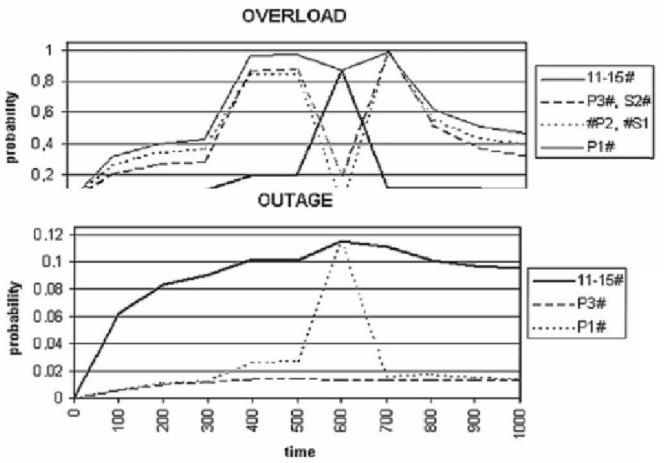


Fig. 13

6.2 FDIR for Autonomous Spacecrafts

The goal of this study (funded by European Space Agency ESA/ESTEC under grant *TEC-SWE/09259/YY*) is the development of a software architecture for FDIR analysis of autonomous spacecrafts (e.g. Mars rovers), based on DBN inference. Such architecture, called ARPHA (Anomaly Resolution and Prognostic Health management for Autonomy), has to distinguish between off-board and on-board software, with emphasis on on-board software capabilities. The choice has been to make an ex-novo implementation of standard DBN inference algorithms (in particular both 1.5JT and BK), by considering the JT as the operational model to be available on-board for inference and computation. On-board software ha to be able to....

However, the generation of the operational model has been thought as supported, as much as possible, by means of automatic translation from standard reliability modeling languages; in particular, we developed an extended version of DFT, able to take into account general stochastic dependencies between system components, as well as the possibility of modeling multi-state components, environmental conditions and control actions[22,23]. From this model a Dynamic Decision Network is derived and subsequently transformed in a DBN for inference purposes. The JT obtained by the DBN is then adopted as on-board model.

DBN inference is used to support the following tasks:

- *diagnosis*: identifying faults or anomalies in system components at the current time, given a stream of sensors observations
- *prognosis*: predicting the future state of system components (and so of the whole system), given the sensor information up to the current time

- *recovery*: providing the suitable control actions to an Autonomy Building Block in order to avoid undesirable consequences; it can be a *reactive recovery* (when the current state has been recognized as failed) or a *preventive recovery* (either if the current state is anomalous but not failed yet, or if the current state is normal, but prognosis identifies future problems)

As a benchmark, we identified the power management subsystem of a Mars rover. In particular we studied how to automatically switch the operating mode (normal, energy-saving or stop), in order to control the battery charge. Fig.14 reports a fragment of the DBN used for this task. Selection of the best recovery action (the switch to the most useful operating mode, represented by variable *Mode* in this example) is performed by exploiting DBN inference in order to compute the Expected Utility of each possible action, and then selecting that providing the maximum.

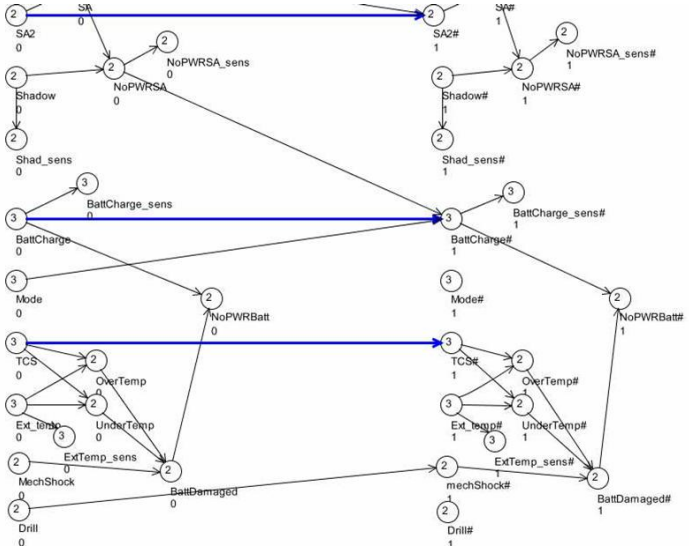
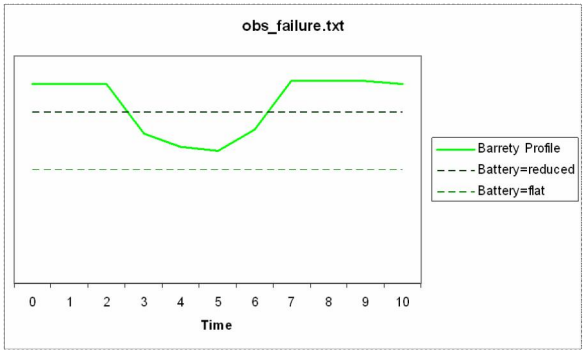


Fig. 14

This means that a utility table is associated with each actions and variables of interests (e.g. *BattCharge* in the example) and EU is computed by considering the posterior probability of such variables [20]. Fig. 15 shows an example of a battery charge profile and the corresponding recovery actions adopted by ARPHA in response to such a profile.



Time	*Selected Mode*	*EU(standard)*	*EU(en_sav)*	*EU(stop)*
1	**standard**	**.999997**	.800001	2.04e-06
2	**Standard**	**.999995**	.800001	3.52e-06
3	**standard**	**.998931**	.800354	.001405
4	**standard**	**.875614**	.841421	.165779
5	**en_sav**	.702058	**.899219**	.397151
6	**en_sav**	.703000	**.898890**	.395190
7	**standard**	**.907382**	.830836	.123234

Fig. 15

7. CONCLUSIONS AND OPEN ISSUES

Common aims and goals are currently being recognized by researchers in classical reliability theory and in the BN community; examples of fields of fruitful cooperation include probabilistic inference for fault detection and identification, monitoring, maintenance, and prediction. A lot of issues will still remain open and need more investigation; among them the use of continuous variables in BN models (hybrid BN) and the introduction of continuous time models in PGM (e.g. GCTBN [6]).

8. REFERENCES

[1] Bobbio A., and L. Portinale and M. Minichino and E. Ciancamerla, “Improving the Analysis of Dependable Systems by Mapping Fault Trees into Bayesian Networks”, Reliability Engineering and System Safety, vol 71, 2001, pp 249-260.

[2] Boyen, X., and D. Koller, “Tractable Inference for Complex Stochastic Processes”, in Proc. Intern. Conf on Uncertainty in Artificial Intelligence - UAI '98; 33-42, 1998.

[3] Chowdhury, B., and S. Baravc, “Creating cascading failure scenarios in interconnected power systems”, Proceedings of the IEEE Power Engineering Society General Meeting, 18–22, 2006.

[4] Codetta Raiteri, D., "The Conversion of Dynamic Fault Trees to Stochastic Petri Nets, as a case of Graph Transformation", In Electronic Notes on Theoretical Computer Science vol. 127(2), pages 45-60, Elsevier, March 2005.

[5] Codetta Raiteri, D., A. Bobbio, S. Montani, L. Portinale, “A dynamic Bayesian network based framework to evaluate cascading effects in a power grid”, Engineering Applications of Artificial Intelligence”, in press, online from July 2010.

[6] Codetta Raiteri, and L. Portinale, “Generalized Continuous Time Bayesian Networks and their GSPN Semantics” Proc. 5th European Workshop on Probabilistic Graphical Models , Helsinki, 2010.

[7] Dean, T., and K. Kanazawa, “A model for reasoning about persistence and causation”, Computational Intelligence 5(3): 142-150,1989.

[8] Doucet A., J.F.G. de Freitas and N.J. Gordon, “Sequential Monte Carlo Methods in Practice”, Springer Verlag, 2000.

[9] Dugan, J.B., S. J. Bavuso, M.A. Boyd, “Dynamic Fault-Tree Models for Fault-Tolerant Computer Systems”, IEEE Transactions on Reliability, volume 41, 1992, pages 363-377.

[10] Dutuit, Y., A. Rauzy, “A Linear-Time Algorithm to Find Modules of Fault Trees”, IEEE Transactions on Reliability, volume 45, 422-425, 1996.
[11] Gulati, R., and J. B. Dugan, “A Modular Approach for Analyzing Static and Dynamic Fault Trees”, Proc. of the Annual Reliability and Maintainability Symposium, pages 1-7, 1997.
[12] Heckermann, D., J.S. Breese, “Causal independence for probability assessment and inference using Bayesian networks”, IEEE Trans Syst Man Cybern, 26(6):826–31, 1996.
[13] Huang, C., and Adnan Darwiche, “Inference in Belief Networks: A Procedural Guide”, International Journal of Approximate Reasoning 15(3); 225-263, 1996.
[14] Langseth, H., and L. Portinale, “Bayesian networks in reliability”, Reliability Engineering and System safety, vol. 92, 92-108, 2007.
[15] Laprie, J.C. "Dependable Computing and Fault Tolerance: Concepts and terminology," Proc. 15th IEEE Symp. on Fault-Tolerant Computing, 1985.
[16] Malhorta, M., and K. Trivedi, Dependability modeling using Petri Nets, IEEE Tr. on Reliability, R-44:428-440,1995.
[17] Montani, S., L. Portinale, A. Bobbio, M. Varesio and D. Codetta Raiteri, “A Tool for Automatically Translating Dynamic Fault Trees Into Dynamic Bayesian Networks”, Proc. Annual Reliability and Maintainability Symposium (RAMS), Newport Beach (CA), 2006.
[18] Montani, S., L. Portinale, A. Bobbio, and D. Codetta Raiteri, “Radyban, a tool for reliability analysis of dynamic fault trees through conversion into dynamic Bayesian networks”, Reliability Engineering and System Safety, vol. 93, 922-932, 2008.
[19] Murphy, K., “Dynamic Bayesian Networks: Representation, Inference and Learning”, PhD Thesis, UC Berkley, 2002.
[20] Pearl, J., “Probabilistic Reasoning in Intelligent Systems”, Morgan Kaufmann, 1988.
[21] Poole D, and N.L. Zhang. “Exploiting causal independence in Bayesian network inference”, Journal of Artif Intell Research, vol. 5, 301–28, 1996.
[22] Portinale, L., and D. Codetta Raiteri, “ARPHA: an FDIR Architecture for Autonomous Spacecrafts based on Dynamic Probabilistic Graphical Models”, Technical Report TR-INF-2010-12-04-UNIPMN, http://www.di.unipmn.it...R-INF-2010-12-04-UNIPMN.pdf, 2010.
[23] Portinale, L., and D. Codetta Raiteri, “ARPHA: an FDIR Architecture for Autonomous Spacecrafts based on Dynamic Probabilistic Graphical Models”, Proc. IJCAI workshop on AI on Space, Barcelona, 2011.
[24] Schneeweiss, W.G. “The Fault Tree Method”, LiLoLe Verlag, 1999.
[25] Trivedi, K.S. “Probability and Statistics with Reliability, Queuing, and Computer Science Applications (2nd ed.)”, J. Wiley, 2001.