Designing Resource-Bounded Reasoners using Bayesian Networks: System Health Monitoring and Diagnosis

Ole J. Mengshoel
RIACS
NASA Ames Research Center
Mail Stop 269-3
Moffett Field, CA 94035
omengshoel@riacs.edu

Abstract

In this work we are concerned with the conceptual design of large-scale diagnostic and health management systems that use Bayesian networks. While they are potentially powerful, improperly designed Bayesian networks can result in too high memory requirements or too long inference times, to they point where they may not be acceptable for real-time diagnosis and health management in resource-bounded systems such as NASA's aerospace vehicles. We investigate the clique tree clustering approach to Bayesian network inference, where increasing the size and connectivity of a Bayesian network typically also increases clique tree size. This paper combines techniques for analytically characterizing clique tree growth with bounds on clique tree size imposed by resource constraints, thereby aiding the design and optimization of large-scale Bayesian networks in resource-bounded systems. We provide both theoretical and experimental results, and illustrate our approach using a NASA case study.

Introduction

In this work we take a systems point of view, and consider situations where a diagnostic process needs to be carefully designed within an overall computer system architecture. Such computational considerations come to the forefront in diagnosis applications where resources are severely limited. For example, diagnosis plays or could play a key role in many real-time systems with limited memory resources.

While we assume a Bayesian network approach to diagnosis, our work carries over to other model-based approaches. Bayesian networks provide a well-established approach to model-based diagnosis and monitoring (Andreassen *et al.* 1987; Shwe *et al.* 1991; Lerner *et al.* 2000; Roychoudhury, Biswas, & Koutsoukos 2006), and clique tree clustering is an important Bayesian network inference algorithm (Lauritzen & Spiegelhalter 1988; Andersen *et al.* 1989). Using the tree clustering approach, a BN's directed graph is compiled to an undirected graph, a clique tree (Lauritzen & Spiegelhalter 1988). In order to compute marginals or most probable explanations, evidence is propagated in the clique tree. The performance of clique tree clustering algorithms depends on the treewidth or the optimal maximal clique size of a BN's induced clique tree (Dechter & Fattah 2001). Unfortunately, it turns out that maximal clique size and total clique tree size can be prohibitively large in application BNs (Shwe *et al.* 1991) as well as in relatively small synthetic BNs (Mengshoel, Wilkins, & Roth 2006).

Early in system design, during conceptual design, there is a lack of information regarding likely BN topologies and consequently much is not known with respect to BN inference times as well as accuracy of diagnostic inference. Ideally, a BN designer would like to carefully investigate important but difficult questions such as the following:

- What is the *resource consumption*, with respect to time and space, of different BN models for diagnosis or system health management?
- What are the *resource bounds* imposed by different hardware and software platform alternatives?

We address these questions by using and extending high-level, macroscopic models of clique tree and inference time growth (Mengshoel, Wilkins, & Roth 2006; Mengshoel 2007). Our growth curves are based on theoretical considerations and on empirical data created through the use of problem instance generators (Mitchell, Selman, & Levesque 1992; Mengshoel, Wilkins, & Roth 2006; Mengshoel 2007). In this paper we show how clique tree growth curves, along with resource bounds, can be used for trade studies during conceptual design. NASA's next-generation crew launch vehicle is used as a case study (NASA 2005).

We believe this research is significant because resource constraints are fundamental in computer science and engineering, including in diagnostic and health management algorithms and software. This work enables improvements in how resource-bounded systems, in particular real-time systems with embedded health monitoring components, are designed and analyzed.

The remainder of this paper is organized as follows. First, we introduce a few fundamental concepts and results related to Bayesian networks. We then briefly discuss the resource-bounded nature of system health management applications of interest to NASA. Next, we present a framework for analyzing resource bounds and resource consumption, the latter as a function of parameters that characterize Bayesian networks. Finally, we apply our analytical framework in computational experiments.

Preliminaries

A Bayesian network (BN) is a tuple $\beta = (\boldsymbol{X}, \boldsymbol{E}, \boldsymbol{P})$, where $(\boldsymbol{X}, \boldsymbol{E})$ is a DAG with associated conditional probability distributions $\boldsymbol{P} = \{\Pr(X_1 \mid \Pi_{X_1}), \ldots, \Pr(X_n \mid \Pi_{X_n})\}$. Here, $\Pr(X \mid \Pi_X)$ is the conditional probability distribution (CPT) for $X \in \boldsymbol{X}$ given its parents Π_X. Let π_X represent an instantiation of Π_X. The independence assumptions encoded in $(\boldsymbol{X}, \boldsymbol{E})$ imply the joint probability distribution

$$\Pr(\boldsymbol{x}) = \prod_{i=1}^{n} \Pr(x_i \mid \pi_{X_i}), \tag{1}$$

where $\Pr(\boldsymbol{x}) = \Pr(X_1 = x_1, \ldots, X_n = x_n)$.

There are two broad classes of approaches to Bayesian network inference: Interpretation and compilation. In interpretation approaches, a Bayesian network is directly used for inference. In compilation approaches, a Bayesian network is off-line compiled into a secondary data structure, where the details depend on the approach being used, and this secondary data structure is then used for on-line inference. Due to their high level of predictability and fast execution times, compilation approaches are especially suitable for resource-bounded reasoning and real-time systems. Our focus here is therefore on compilation approaches, and in particular the arithmetic circuit approach and the clique tree clustering (or join tree) approach.

Creation of an arithmetic circuit from a Bayesian network is a relatively recent compilation approach (Park & Darwiche 2004; Chavira & Darwiche 2007). Here, Bayesian networks are represented as multivariate polynomials, translated into arithmetic circuits, and thus probabilistic inference translates into a process of operating on such circuits. These circuits have a relatively simple structure, but can be used to answer a wide range of probabilistic queries.

In this paper, though, our main focus is on the clique tree clustering approach (Lauritzen & Spiegelhalter 1988; Andersen *et al.* 1989). A clique tree β''' is constructed from a BN in the following way by the Hugin tree clustering algorithm (Andersen *et al.* 1989). First, an initial moral graph β' is constructed by making an undirected copy of β and then augmenting it as follows. For each node X in β, Hugin adds to β' an edge between each pair of nodes in Π_X if no such edge already exists in β'. Second, Hugin creates a triangulated graph β'' by adding fill-in edges to β' such that no chordless cycle of length greater than three exists. Third, a clique tree β''' is created from β''. Hugin uses essentially the same clique tree β''' for both belief updating (marginals) and belief revision (most probable explanations).

There are several reasons why we investigate tree clustering here. First, tree clustering is a theoretically well-founded approach to exact inference in BNs. The complexity of other exact BN inference algorithms — including conditioning and elimination algorithms — depends on treewidth τ^*, which is closely related to tree clustering's optimal maximal clique size h^* since $\tau^* = h^* - 1$ (Lauritzen & Spiegelhalter 1988; Dechter & Fattah 2001). Second, due to the use of clique trees for inference, tree clustering is quite predictable, which is a major advantage in resource-bounded environments. Third, tree clustering algorithms have been implemented in several high-quality software systems.

We investigate bipartite BNs, BNs in which the nodes $\boldsymbol{X}$ are partitioned into root nodes $\boldsymbol{V}$ and leaf nodes $\boldsymbol{C}$. Bipartite BNs are sampled using the BPART algorithm (Mengshoel, Wilkins, & Roth 2006), which is a generalization of an algorithm for randomly generating instances of the satisfiability problem (Mitchell, Selman, & Levesque 1992). The BPART algorithm operates as follows. First, $V = |\boldsymbol{V}|$ root nodes, each with S states, are created. Second, $C = |\boldsymbol{C}|$ leaf nodes, also each with S states, are created. For each leaf node, P parent nodes $\{X_1, \ldots, X_P\}$ are picked uniformly at random without replacement among the V root nodes. In this paper our main interest is in the structural parameters V, C, P, and S, and we do not consider the CPTs constructed by BPART. Consequently, the signature BPART(V, C, P, S) is used. The number of BN edges E created is given by $E = C \times P$ and the total number of nodes is $N = C + V$.

Our key idea is that growth curves provide estimates of resource consumption in terms of clique tree size. Such clique trees do not need to be generated from bipartite BNs, even though we emphasize them here. Bipartite BNs are important in applications. Naive Bayes models, successful in spam filtering, are a special case of bipartite BNs with only one root node. The QMR-DT BN is a bipartite medical BN — diseases are root nodes and symptoms are leaf nodes (Shwe *et al.* 1991). Special inference algorithms have also been designed for bipartite BNs (Ng & Jordan 2000).

For all BNs, including for bipartite BNs, it is important but also very difficult to understand and predict clique tree clustering's cycle-generation and fill-in processes. Further, these strongly non-linear processes need to be traded off against resource bounds. We will return to these topics after first introducing our perspective on resource-bounded diagnostic reasoning.

Designing Resource-Bounded Reasoners

Diagnosis and system health management in resource-bounded systems is of great interest to NASA and the aerospace community. Relevant examples of Bayesian networks exist, for example, in process monitoring and control (Lerner *et al.* 2000; Roychoudhury, Biswas, & Koutsoukos 2006) and medical diagnosis and monitoring (Andreassen *et al.* 1987; Shwe *et al.* 1991).

As a concrete case study, we will discuss NASA's next-generation crew launch vehicle (CLV). This vehicle, which will replace the space shuttle, is currently being developed. The CLV is likely to include a solid rocket engine; see Figure 1. Historically, a 91.4% success rate was observed for world-wide space launches in the time period 1957-2004, with the propulsion subsystem being the Achilles' heel of launch vehicles (Chang & Tomei 2005). To ensure the safety of the crew, it is important to estimate and monitor the health state of the solid rocket during launch, such that the crew can escape in a timely fashion if needed.

In a study of space exploration system architectures, NASA identified (i) key solid rocket failure modes, (ii) their potential detection methods, and (iii) approximate reaction times for crew abort (NASA 2005). Failure modes include

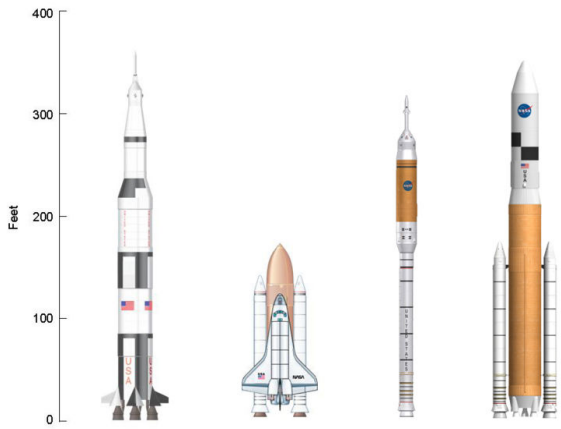

Figure 1: From left to right: The Saturn V, the Space Shuttle and two candidate designs for the next-generation Crew Launch Vehicle (CLV).

joint failures, structural failures, thermal failures, and performance failures. A total of 22 key failure modes have been identified (NASA 2005), however as the system is being developed this number might change. An interesting question, which we will return to in the experimental section, is what the impact would be if the number of failure modes or sensors were increased or decreased. Possible sensors include traditional cameras, infrared cameras, thermocouples, pressure transducers, burn-through wires, and strain gauges. Reaction times, which are hard real-time, range from close to 0 seconds for catastrophic failures such as case failure to 130 seconds for less threatening failures such as case insulator failures (NASA 2005).

Musliner and his coauthors identified three approaches to real-time AI (Musliner *et al.* 1995); we investigate what they call "embedding AI into a real-time system". In other words, we make system health management (SHM) a real-time task or a set of a real-time tasks. We call this embedded system health management (ESHM).

The ESHM approach is based on real-time tasks and may also utilize a hard real-time operating system (RTOS). A prototypical RTOS model has emerged over the last decade or so: An RTOS typically uses priority-based preemptive scheduling where each task has a fixed priority. The RTOS scheduler lets higher-priority tasks interrupt lower-priority tasks while tasks on the same priority level, if supported, are executed on a round-robin basis. A periodic RTOS task $\tau_i = (p_i, d_i, w_i)$ has a task period p_i, a deadline d_i, and a worst-case execution time (WCET) w_i. In other words, a task not only needs to compute correct outputs for all inputs, it also needs to do so within its deadline d_i. Formally, let T_i be the random execution time for τ_i; clearly $\Pr(T_i > w_i) = 0$ needs to hold. Here, T_i accounts for the different execution paths of τ_i due to input variations. Different RTOS scheduling algorithms exist. Two approaches that are amendable to mathematical analysis are known as earliest deadline first (EDF) scheduling and rate-monotonic (RM) scheduling. Through analysis, and given appropriate assumptions, one can determine whether all task deadlines can be met for a given set of tasks under RM or EDF.

Clearly, some AI approaches are suitable for this ESHM approach, while others are not (Musliner *et al.* 1995). Among the real-time task parameters discussed above, a predictable, bounded worst-case execution time w_i is essential. Clique tree size, along with the application-dependent hardware and software platforms, essentially determine w_i. Further, once a clique tree has been generated by compilation, its memory requirement has been determined, which is also desirable in real-time and resource-bounded systems. All in all, BN inference using tree clustering provides a solid foundation for resource-bounded reasoning.

In the conceptual design phase, there is typically a lack of detailed knowledge about the overall system being designed, with the implication that a large number of candidate ESHM BNs exists. Generation of all these BNs and processing them using clique tree clustering is seldom feasible. At the same time, given the computational hardness of BN inference (Cooper 1990; Shimony 1994) as well as the observed difficulty of application BNs (Andreassen *et al.* 1987; Shwe *et al.* 1991; Lerner *et al.* 2000; Roychoudhury, Biswas, & Koutsoukos 2006) and synthetic BNs (Mengshoel, Wilkins, & Roth 2006) it is clear that a certain amount of care is well-advised, in particular when the size and connectivity of Bayesian networks increase. This motivates our discussion of resource bounds and resource consumption in the following.

Resource Bounds

Memory and time bounds are fundamental in computer science, including in SHM algorithms and software. In this section we develop novel bounds on clique tree size to express time- and memory-bounds.

Clique tree sizes can be translated into quantities directly indicating memory usage, once a memory model such as the following is introduced.

Definition 1 (Linear memory model) *Let $\phi = \beta'''$ be a clique tree with size $k(\phi)$. A linear memory resource model for ϕ is given by*

$$r_m(\phi) = a \times k(\phi) \text{ bytes} + b \text{ bytes},$$

where $a \in \mathbb{N}^+$ and $b \in \mathbb{N}$ are system parameters.

The parameters a and b that determine the amount of primary memory (RAM) required for processing are system-dependent. Here is an example.

Example 2 *Consider a clique tree ϕ, and suppose that each clique tree component uses a* `double` *data type requiring $a = 8$ bytes while the fixed overhead is $b = 0$ bytes. The amount of RAM r_m needed to accommodate a clique tree consisting of one clique with $h = 24$ binary ($S = 2$) nodes is then:*

$$r_m(\phi) = a \times k(\phi) + b \text{ bytes} = 8 \times S^h \text{ bytes} = 2^{24} \times 8 \text{ bytes} = 134,217,728 \text{ bytes} = 128MB.$$

We now introduce memory-induced clique tree bounds $\lambda_m(m_{\max})$ as follows.

Definition 3 (Memory-bounded maximal clique tree) *Let Φ be a set of clique trees, and put $\Phi' = \{\phi \mid \phi \in \Phi$ and $r_m(\phi) \leq m_{\max}\}$. The memory-bounded maximal clique tree size $\lambda_m(m_{\max})$, determined by Φ and the memory bound $m_{\max}$, is defined as*[1]

$$\lambda_m(m_{\max}) = k\left(\arg\max_{\phi\in\Phi'} r_m(\phi)\right). \tag{2}$$

In words, Definition 3 expresses the idea of picking as large a clique tree as possible without going beyond the memory bound $m_{\max}$. The sizes of the diagnostic BNs we can develop are constrained by the maximal memory $m_{\max}$ available to their clique trees.

To be useful for real-time diagnosis and health management, not only must an ESHM Bayesian network fit into memory, it must also support the timely computation of a diagnosis. To capture this constraint, we introduce the following linear time model.

Definition 4 (Linear time model) *Let ϕ be a clique tree. A linear WCET time model for ϕ is given by*

$$r_t(\phi) = c \times r_m(\phi)\ ms + d\ ms,$$

where $c \in \mathbb{N}^+$ and $d \in \mathbb{N}$ are system parameters.

Definition 4 expresses the fact that execution time $r_t(\phi)$ is partly determined by clique tree memory requirements $r_m(\phi)$, but $r_t(\phi)$ also depends on several hardware- and software-specific factors as reflected in the parameters c and d. Hardware-specific factors include the processor clock speed, amount of primary memory, speed of primary memory, and amount of cache. Software-specific factors include the type of data structures, programming language, compiler, and operating system used to implement and deploy the clique tree clustering system. We leave detailed investigations of the impact of different hardware and systems software for future research.

We now introduce the time-induced clique tree bound $\lambda_t(t_{\max})$ as follows. For simplicity, we consider exactly one ESHM RTOS task $\tau_1 = (p_1, d_1, w_1)$, and put $t_{\max} = w_1$.

Definition 5 (Time-bounded maximal clique tree) *Let Φ be a set of clique trees, and put $\Phi' = \{\phi \mid \phi \in \Phi$ and $r_t(\phi) \leq t_{\max}\}$. The time-bounded maximal clique tree size $\lambda_t(t_{\max})$, determined by Φ and the time bound $t_{\max}$, is defined as*

$$\lambda_t(t_{\max}) = k\left(\arg\max_{\phi\in\Phi'} r_t(\phi)\right). \tag{3}$$

In words, Definition 5 defines the size of the largest clique tree whose execution time does not exceed $t_{\max}$.

During conceptual design, $t_{\max}$ and $m_{\max}$ may in fact not be fixed. The reason for this is that different system and ESHM design alternatives are in general considered early in development, which translates into a range of possible values for $t_{\max}$ and $m_{\max}$. Consequently, a diagnostic system designer may consider subsets $\{m_1, \ldots, m_k\}$ and $\{t_1, \ldots, t_n\}$ rather than $t_{\max}$ and $m_{\max}$ respectively. Let $m \in \{m_1, \ldots, m_k\}$ and $t \in \{t_1, \ldots, t_n\}$. Then we have from (2) and (3) the following joint resource-induced clique tree bound $\lambda(m, t)$:

$$\lambda(m, t) = \min(\lambda_m(m), \lambda_t(t)). \tag{4}$$

Inputting $(m, t) \in \{m_1, \ldots, m_k\} \times \{t_1, \ldots, t_n\}$ into (4) results in a set of resource-induced clique tree bounds $\Lambda := \{\lambda_1, \lambda_2, \ldots\}$. The advantage of using resource-induced bounds Λ on total clique tree size (or, along similar lines, on arithmetic circuit size) is that they uniformly represent both memory bounds and time bounds, and can easily be compared to clique tree growth curves expressing resource consumption. In the experimental section, we will use a set $\{\lambda_1, \lambda_2, \lambda_3\}$ of resource-induced bounds.

Resource Consumption

As the size and connectivity of a BN increases, so does the consumption of computational resources required for its processing. Therefore, when embarking on a BN development effort, one would like to avoid developing a BN model that cannot be processed within the resource bounds of a particular computer system. To avoid this problem, we estimate resource consumption ahead of time by analytical means. Here, we discuss an analytical framework for characterizing clique tree growth (Mengshoel 2007).

In a diagnostic bipartite BNs, the number of root nodes V represents the number of failure modes, components, or diseases, and the number of leaf nodes C represents the number of sensors, tests, or symptoms. From a bipartite BN, compilation produces a clique tree consisting of root cliques and mixed cliques (Mengshoel, Wilkins, & Roth 2006). *Root cliques* are cliques with root nodes only, while *mixed cliques* are cliques with both root nodes and leaf nodes.

Based on previous research (Mengshoel, Wilkins, & Roth 2006; Mengshoel 2007), we now provide a qualitative discussion of the growth of clique trees generated from BPART BNs in terms of the C/V-ratio. We identify three broad stages of clique tree growth: The initial growth stage, the rapid growth stage, and the saturated growth stage. The *initial growth stage*, where the C/V-ratio is "low" (for $P = 2$, up to approximately $C/V \approx 1$), is characterized by "few" leaf nodes relative to the number of root nodes. There is consequently a relatively low contribution by root cliques to the clique tree. This stage is generally dominated by mixed cliques — indeed as $C/V \to 0$ there are no root cliques with more than one root node. During the *rapid growth stage*, where the C/V-ratio is "medium" (for $P = 2$, from approximately $C/V \approx 1$), non-trivial root cliques start appearing, causing dramatic growth in the total size of root cliques on average. This growth is due to formation of cycles where fill-in edges are required in order to triangulate the moral graph. Because of fill-in edges, the root cliques gradually dominate the mixed cliques in terms of their contribution to total clique tree size. The *saturated growth stage*, where the C/V-ratio is "high", is characterized by a "large" number of leaf nodes relative to the number of root nodes. As $C/V \to \infty$, one root clique with V BN root nodes and size

[1] If there are multiple candidates for $\arg\max_{\phi\in\Phi} r_m(\phi)$, we arbitrarily pick one of them since their size is the same.

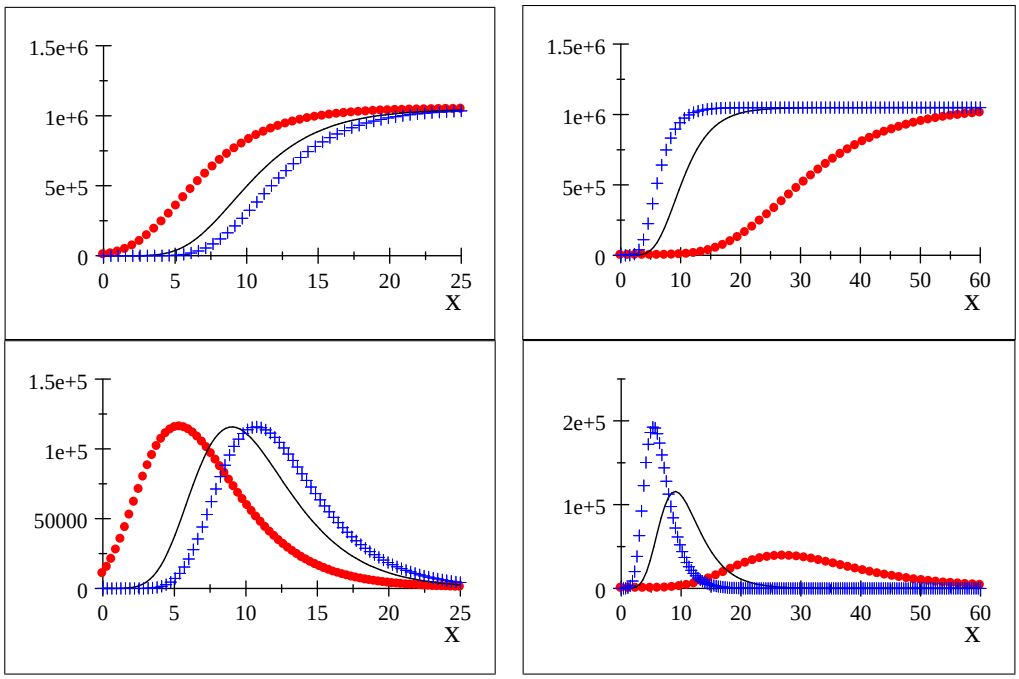

Figure 2: Gompertz curves of the form $g(x) = g(\infty)e^{-\zeta e^{-\gamma x}} = 2^{20}e^{-\zeta e^{-\gamma x}}$, where the parameters ζ and γ are varied. *Top left:* The curves plotted have the form $g(x) = 2^{20}e^{-\zeta e^{-0.3x}}$, with $\zeta = 5$ (red dotted curve), $\zeta = 15$ (black solid curve), and $\zeta = 25$ (blue crossed curve). *Top right:* The curves plotted have the form $g(x) = 2^{20}e^{-15e^{-\gamma x}}$, with $\gamma = 0.1$ (red dotted curve), $\gamma = 0.3$ (black solid curve), and $\gamma = 0.5$ (blue crossed curve). *Bottom left:* Growth rates $g'(x)$ for different $g(x)$ directly above. *Bottom right:* Growth rates $g'(x)$ for different $g(x)$ directly above.

S^V emerges. The phase of greatest interest to developers of large-scale BNs is the rapid growth stage, where the size of induced clique trees quickly transition from feasible to infeasible for resource-bounded reasoners.

Clique trees are discrete structures, however we here use continuous growth models in order to simplify mathematical analysis.

Definition 6 (Clique tree growth curve) *Let $g_R(x)$ be the growth curve for all root cliques and $g_M(x)$ the growth curve for all mixed cliques. The (total) clique tree growth curve is defined as*

$$g_T(x) = g_R(x) + g_M(x).$$

Restricted growth has been modeled using different sigmoidal growth curves ("S-curves"), including the logistic, Gompertz, Complementary Gompertz, and Richards growth curves (Banks 1994; Lindsey 2004). We emphasize Gompertz growth curves, since they are theoretically plausible and turn out to give best fit empirically (Mengshoel 2007).

Definition 7 (Gompertz growth curve) *The Gompertz growth curve is*

$$g(x) = g(\infty)e^{-\zeta e^{-\gamma x}}, \tag{5}$$

where $g(\infty)$ is the asymptote as $x \to \infty$.

The independent variable x in (5) is $x = C$ or $x = C/V$ in this paper. The derivative

$$g'(x) = \frac{d}{dx}\left(g(\infty)e^{-\zeta e^{-\gamma x}}\right) = g(\infty)\zeta\gamma e^{-\gamma x}e^{-\zeta e^{-\gamma x}},$$

which we call the growth rate, shows how Gompertz growth changes as a function of x.

We now introduce, from related work (Mengshoel 2007), a Gompertz growth curve for BPART.

Theorem 8 (Total Gompertz growth curve) *The total Gompertz growth curve for clique trees generated from BPART(V, C, P, S) BNs, where $x = C$ is the independent parameter, is*

$$g_T(x) = S^V e^{-\zeta e^{-\gamma x}} + xS^{P+1}. \tag{6}$$

Growth curves are frequently used in medicine, biology, and sociology (Banks 1994; Lindsey 2004), however our use of them in designing resource-bounded reasoning systems is, to our knowledge, novel.

In Figure 2 we investigate how varying the parameters ζ and γ impacts the shape of Gompertz curves. The factor $g_R(\infty) = S^V = 2^{20}$ is obtained by considering bipartite Bayesian networks with $V = 20$ binary (so $S = 2$) root nodes. Figure 2 also shows how the growth rate $g'(x)$ changes when ζ and γ are varied.

Let us first consider varying ζ as illustrated to the left in Figure 2. By increasing ζ, the x-location of maximal growth

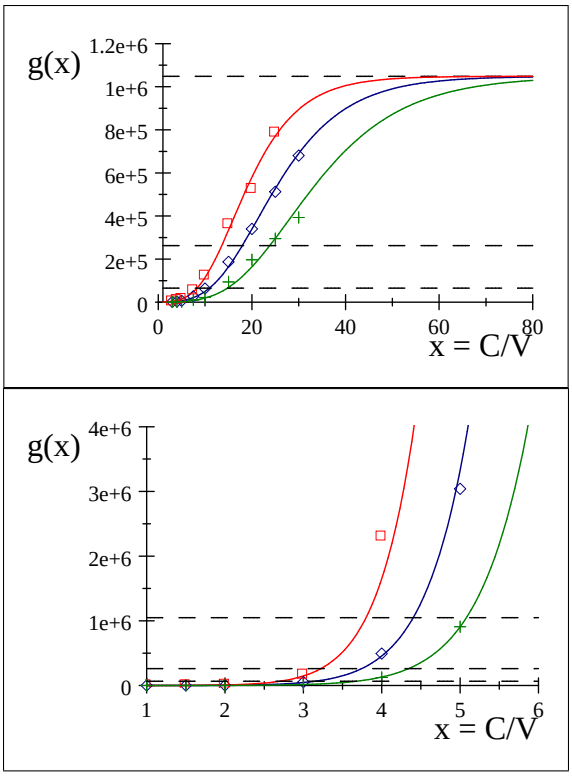

Figure 3: Gompertz growth curves g_{20} (top, for $V = 20$ root nodes) and g_{40} (bottom, for $V = 40$ root nodes) are shown here, along with sample means, using $x = C/V$ as the independent variable. Data from which the curves were created is shown in Table 1. Straight lines represent upper bounds on clique tree size and reflect different resource bounds.

rate $g'(x)$ is increased as well. In other words, the *initial growth stage* lasts longer as ζ increases, and the onset of the *rapid* and *saturated growth stage* are delayed. However, the maximal value of $g'(x)$ does not change with changing ζ.

Let us next consider varying γ as illustrated to the right in Figure 2. As γ increases, the x-location of maximal growth rate $g'(x)$ decreases. For example, $\gamma = 0.1$ has maximal $g'(x)$ at approximately 28, while $\gamma = 0.5$ has maximal $g'(x)$ at approximately 6. In addition, with increasing γ the maximal value of $g'(x)$ increases. In other words, the *initial growth stage* is shorter as γ increases, and the *rapid* and *saturated growth stage* start for smaller values of x.

It is clear from Figure 2 that Gompertz curves provide modelling flexibility and power that is potentially useful in modelling growth in resource consumption of model-based techniques including BNs. For instance, it can be very useful to know where a BN or a distribution of BNs is located on $g(x)$ and $g'(x)$.

Experimental Results

As a concrete case study, we investigate the design of an ESHM diagnostic system for a next-generation crew launch vehicle (CLV) (NASA 2005). The identified failure modes and detection methods can be represented as a bipartite Bayesian network where the failure modes are root nodes, and detectors or sensor are leaf nodes. Such a bipartite network could make up a time-slice in a dynamic, time-sliced Bayesian network, where each time slice contains both discrete and continuous random variables. Our goal is to obtain high diagnostic accuracy while also keeping clique tree size (and therefore memory requirement and inference time) below certain bounds. To achieve this, we consider upper and lower bounds on the number of root and leaf nodes. Formally, we introduce the parameters $V_{\min}$, $V_{\max}$, $C_{\min}$, and $C_{\max}$. $V_{\min}$ represents the minimal number of root nodes considered for the BN; $V_{\max}$ is the maximal number of root nodes. Similarly, $C_{\min}$ and $C_{\max}$ are bounds for the leaf nodes in a bipartite BN.

Empirical Gompertz Growth Curves
$g_{20}^{-}(x) = 2^{20}\exp(-\exp(2.158)\exp(-0.0769x))$
$g_{20}(x) = 2^{20}\exp(-\exp(2.108)\exp(-0.0993x))$
$g_{20}^{+}(x) = 2^{20}\exp(-\exp(2.113)\exp(-0.132x))$
$g_{40}^{-}(x) = 2^{40}\exp(-\exp(3.288)\exp(-0.130x))$
$g_{40}(x) = 2^{40}\exp(-\exp(3.269)\exp(-0.145x))$
$g_{40}^{+}(x) = 2^{40}\exp(-\exp(3.259)\exp(-0.166x))$

Table 2: Growth curves constructed from bipartite BNs with $V = 20$ and $V = 40$ root nodes. The underlying data is summarized in Table 1.

In experiments reflecting the current estimate $V = 22$ CLV failure modes (NASA 2005), we sampled bipartite BNs using BPART(V, C, 2, 2), using $V = V_{\min} = 20$ and $V = V_{\max} = 40$. The results reported here are based on experiments with a total of 2,700 BNs, with 1,800 BNs for $V_{\min} = 20$ and 900 BNs for $V_{\max} = 40$.

Results are reported in Table 1 and Figure 3. Along the figure's x-axis, we use $x = C/V$ as the independent parameter. Along the y-axis, we display the empirically determined Gompertz growth curves for the root cliques:

$$g(x) = g_R(x) = S^V e^{-\zeta e^{-\gamma x}}, \tag{7}$$

as well as results from the samples. We focus on the root clique growth curve $g_R(x)$ since it dwarfs the mixed clique growth curve $g_M(x)$ for the parameter values of interest here. In (7), we determined ζ and γ by experimentation. These two parameters therefore depend on C, V, P, and S as well as the BNs sampled. To determine a given pair of ζ and γ, an estimation approach based on Lindsey's (Lindsey 2004) was employed — see Figure 4 and elsewhere (Mengshoel 2007) for details.

For each value of V we present three growth curves, namely for minimums $g_V^{-}(x)$, for means $g_V(x)$, and for maximums $g_V^{+}(x)$. Maximums are conservative estimates and reflect, for example, situations in which minimal effort is available for optimization of the BN's structure in order to reduce clique tree size.

Figure 3 contains the following curves:

- The growth curves $g_{20}^{-}(x)$, $g_{20}(x)$, and $g_{20}^{+}(x)$ represent

C/V	1	1.5	2	3	4	5	7.5	10	15	20	25
Min, $V = 20$				584	1264	2312	10304	22016	94208	196608	294912
Mean, $V = 20$				1204	3221	6683	27292	63245	187580	339838	512953
Max, $V = 20$				2568	9712	13888	53504	122880	360448	524288	786432
Min, $V = 40$	54	392	996	15552	123696	907728					
Mean, $V = 40$	150	658	3010	52988	492283	3040530					
Max, $V = 40$	274	1232	9754	165000	2299072	9869824					

Table 1: Raw data and sample means for total size of root cliques in cliques trees, generated by varying the C/V-ratio for $V = 20$ and $V = 40$ root nodes. This data was used to create the Gompertz growth curves in Figure 3.

clique tree sizes for the minimal number of failure modes (or causes) $V_{\min} = 20$.

- The growth curves $g_{40}^{-}(x)$, $g_{40}(x)$, and $g_{40}^{+}(x)$ represent clique tree sizes for the maximal number of failure modes $V_{\max} = 40$.
- A straight line $\lambda_i \in \Lambda = \{\lambda_1, \lambda_2, \lambda_3\}$ represents a resource-induced clique tree bound, where λ_i is either a time-induced bound or a memory-induced bound according to (4). Here, $\lambda_1 = 2^{16}$, $\lambda_1 = 2^{18}$, and $\lambda_1 = 2^{20}$.

Table 2 contains empirically determined formulas for the growth curves. One can consult these curves to determine which point(s) or area(s) represents a reasonable trade-off between the various factors. We now give a few examples.

Suppose that a diagnostic system designer wants to investigate the impact of varying the size of the sensor suite for a CLV. If the number of sensors C is increased, then C/V increases as well. It is thus interesting to consider the different impact of large C/V values for $V = 20$ versus $V = 40$. For sake of argument, suppose that $x = C/V \geq 4$; i.e. there are four or more times as many sensors as state variables in a BN and we study the impact of varying $\lambda \in \Lambda = \{\lambda_1, \lambda_2, \lambda_3\}$. Consulting Figure 3, we easily see that for $V = 20$, $C/V \approx 4$ is not a problem, even for $\lambda_1 = 2^{16}$. For $V = 40$, on the other hand, λ_1 is clearly a challenge, even for the most optimistic curve $g_{40}^{-}(x)$. On the other hand, $g_{40}(4) < \lambda_3$ and thus λ_3 gives a little bit of a margin, while $g_{40}(4) > \lambda_2 > g_{40}^{-}(4)$. Using λ_2 thus means taking some risk, and one potentially needs to spend some time optimizing the BN or the inference algorithm.

We now discuss the growth curve formulas in Table 2. The biggest difference here, for both $V = 20$ and $V = 40$, is perhaps in the γ parameter. This shows not only an earlier maximal growth rate, as reflected in Figure 2, but also greater maximal growth rate for $g_V^{+}(x)$ compared to those for $g_V^{-}(x)$ and $g_V(x)$. In other words, if we are concerned about conservative estimates of clique tree growth as reflected in $g_V^{+}(x)$, and are in or approaching the rapid growth stage, then this results suggests that clique tree size can increase very dramatically as a result of rather minor increases in C. This effect is, we believe, relatively easy to see if one compares the growth curves in Table 2, but rather difficult to catch if one only looks at the data in Table 1.

Conclusion and Future Work

The goal of this paper has been to help bridge the gap between, on the one hand, diagnostic resource-bounded reasoning and, on the other hand, Bayesian network inference using clique tree clustering. Our conclusion is two-fold. First, diagnostic reasoning in resource-bounded and real-time systems is as much about predictability as speed. And clique tree clustering, being a dynamic programming algorithm, is in fact quite predictable once a particular BN has been developed and compiled into a static clique tree. A similar argument holds for the arithmetic circuit approach.

Second, we have discussed the use of growth curves in trade-off studies during conceptual design, where only information about distributions of BNs is available. Conceptual design presents a chicken-and-egg problem, since in order to compute clique tree size or inference time one needs to construct one or more Bayesian networks. However, in order to develop a Bayesian network by hand, one needs to perform extensive knowledge engineering, which is beyond the scope of conceptual design, especially for large-scale diagnostic BNs.

To tackle the challenge of conceptual design of Bayesian networks, we have developed a design paradigm utilizing computational resource bounds along with coarse-grained growth models based on processing sampled BNs. Our approach facilitates a better understanding of the trade-offs between the resource demands of clique trees (or arithmetic circuits) created from different BNs versus computational resource bounds. The approach has the potential to lead to fewer problems due to under-engineered systems during system implementation and integration, thus reducing cost and risk. Since more is known early in design, we potentially also reduce the chance of over-engineered systems, thereby potentially reducing their cost and weight, both major concerns in aerospace as well as in other disciplines.

This research can be extended in several directions, of which we mention a few. First, it would be interesting to consider in more detail BNs beyond the bipartite model and more generally extend the class of BNs characterized using growth curves. Second, it could be useful to study other model-based approaches, algorithms, and applications where resource bounds are relevant.

Acknowledgments This material is based upon work supported by NASA under award NCC2-1426. The anonymous reviewers are acknowledged for their comments, which helped improve the article.

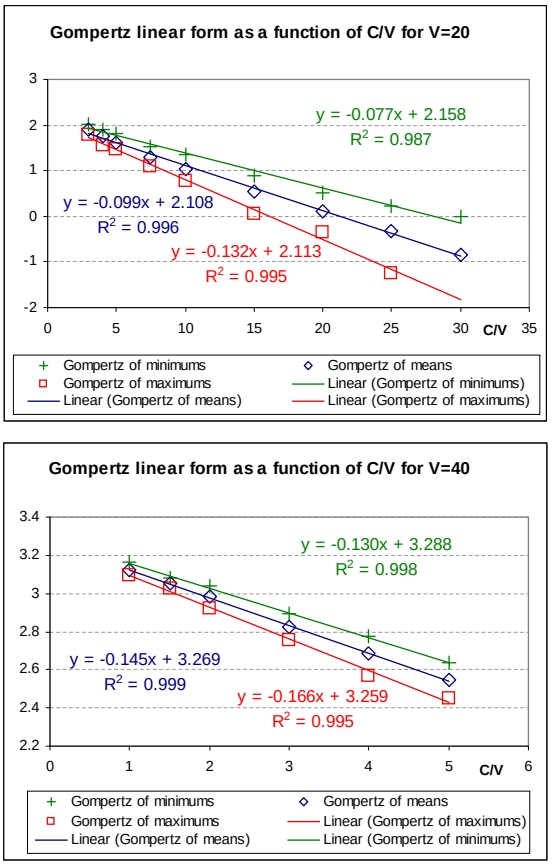

Figure 4: Linear forms summarizing the construction of the Gompertz growth curves are shown here. The regression lines provide the parameters ζ and γ used in the Gompertz growth curves.

References

Andersen, S. K.; Olesen, K. G.; Jensen, F. V.; and Jensen, F. 1989. HUGIN—a shell for building Bayesian belief universes for expert systems. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, volume 2, 1080–1085.

Andreassen, S.; Woldbye, M.; Falck, B.; and Andersen, S. 1987. MUNIN – A causal probabilistic network for interpretation of electromyographic findings. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, 366–372.

Banks, R. B. 1994. *Growth and Diffusion Phenomena*. New York: Springer.

Chang, I., and Tomei, E. J. 2005. Solid rocket failures in world space launches. In *41st AIAA Joint Propulsion Conference*.

Chavira, M., and Darwiche, A. 2007. Compiling Bayesian networks using variable elimination. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*, 2443–2449.

Cooper, F. G. 1990. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence* 42:393–405.

Dechter, R., and Fattah, Y. E. 2001. Topological parameters for time-space tradeoff. *Artificial Intelligence* 125(1-2):93–118.

Lauritzen, S., and Spiegelhalter, D. J. 1988. Local computations with probabilities on graphical structures and their application to expert systems (with discussion). *Journal of the Royal Statistical Society series B* 50(2):157–224.

Lerner, U.; Parr, R.; Koller, D.; and Biswas, G. 2000. Bayesian fault detection and diagnosis in dynamic systems. In *Proceedings of the Seventeenth national Conference on Artificial Intelligence (AAAI-00)*, 531–537.

Lindsey, J. K. 2004. *Statistical Analysis of Stochastic Processes in Time*. Cambridge: Cambridge.

Mengshoel, O. J.; Wilkins, D. C.; and Roth, D. 2006. Controlled generation of hard and easy Bayesian networks: Impact on maximal clique tree in tree clustering. *Artificial Intelligence* 170(16-17):1137–1174.

Mengshoel, O. J. 2007. Macroscopic models of clique tree growth for Bayesian networks. In *Proceedings of the Twenty-Second National Conference on Artificial Intelligence (AAAI-07)*.

Mitchell, D.; Selman, B.; and Levesque, H. J. 1992. Hard and easy distributions of SAT problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, 459–465.

Musliner, D.; Hendler, J.; Agrawala, A. K.; Durfee, E.; Strosnider, J. K.; and Paul, C. J. 1995. The challenges of real-time AI. *IEEE Computer* 28:58–66.

NASA. 2005. Exploration systems architecture study. Technical Report NASA-TM-2005-214062, National Aeronautics and Space Administration.

Ng, A. Y., and Jordan, M. I. 2000. Approximate inference algorithms for two-layer Bayesian networks. In *Advances in Neural Information Processing Systems 12 (NIPS-99)*. MIT Press.

Park, J. D., and Darwiche, A. 2004. A differential semantics for jointree algorithms. *Artificial Intelligence* 156(2):197–216.

Roychoudhury, I.; Biswas, G.; and Koutsoukos, X. 2006. A Bayesian approach to efficient diagnosis of incipient faults. In *Proceedings of the 17th International Workshop on Principles of Diagnosis (DX-06)*, 243 – 250.

Shimony, E. 1994. Finding MAPs for belief networks is NP-hard. *Artificial Intelligence* 68:399–410.

Shwe, M.; Middleton, B.; Heckerman, D.; Henrion, M.; Horvitz, E.; Lehmann, H.; and Cooper, G. 1991. Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base: I. The probabilistic model and inference algorithms. *Methods of Information in Medicine* 30(4):241–255.