Structure From Failure

Ralf Herbrich RHERB@MICROSOFT.COM
Microsoft Research Ltd., Cambridge, UK

Thore Graepel THOREG@MICROSOFT.COM
Microsoft Research Ltd., Cambridge, UK

Brendan Murphy BMURPHY@MICROSOFT.COM
Microsoft Research Ltd., Cambridge, UK

Abstract

We investigate the problem of learning the dependencies among servers in large networks based on failure patterns in their up-time behaviour. We model up-times in terms of exponential distributions whose inverse lifetime parameters lmay vary with the state of other servers. Based on a conjugate Gamma prior over inverse lifetimes we identify the most likely network configuration given that any node has at most one parent. The method can be viewed as a special case of learning a continuous time Bayesian network. Our approach enables us to easily incorporate existing expert prior knowledge. Furthermore our method enjoys advantages over a state-of-the-art rule-based approach. We validate the approach on synthetic data and apply it to five year data for a set of over 500 servers at a server farm of a major Microsoft web site.

1. Introduction

Murphy's Law[1] states that *"Anything that can go wrong will go wrong!"* and thus emphasises the unavoidability and pervasiveness of failure in the world. Building on Murphy's insight, this article proposes a Bayesian methodology for making use of failure data from server event logs for inferring the (failure) structure of networks.

Modern server farms exhibit highly complex, often incrementally grown network architectures with up to several thousand nodes which can be functionally and physically related. In such an environment the causes of system failures (crashes) cannot solely be attributed to isolated hardware or software defects, rather system management and environmental activities can be the dominant factors. The complex configuration of these sites increases the risk that a failure on one system can induce failures on other related systems (i.e. cascading failures) [4] (see also [2] for failure statistics on Windows systems). This complexity also increases the difficulty of diagnosing these cascading failures. A crucial first step in improving this situation was to develop an automated data collection process continuously capturing behavioural information from production systems on customer sites, such as the Digital Product Performance Programme at Digital, and later the Microsoft Reliability Analysis Service at Microsoft[2]. These systems require an on-system event logging mechanism which creates a continuous record of events occurring on the system, with event codes indicating the kind of event together with a time-stamp. In order to analyse the dependencies between nodes in a network, the server event logs of all the systems involved are collected, cleaned and stored in a SQL database. Following earlier work [4] we focus on the three events `clean shutdown`, `dirty shutdown` *and* `start eventlog`. *In contrast to Murphy's DOWS process [3], which identifies temporally related clusters of events from the time series, we learn a continuous time Bayesian network (CTBN)* [6, 7]. In the CTBN framework the up-times of any given node are modelled based on an exponential distribution with piece-wise constant hazard rates that may depend on the states of other, related nodes. CTBNs are models similar to dynamic Bayesian networks (DBNs) ([1],

[1]Named after Capt. Edward A. Murphy, who introduced the law in response to an inept technician at Edwards Airforce Base in 1949.

[2]See *http://www.microsoft.com/windowsserver2003/mras/default.mspx.*

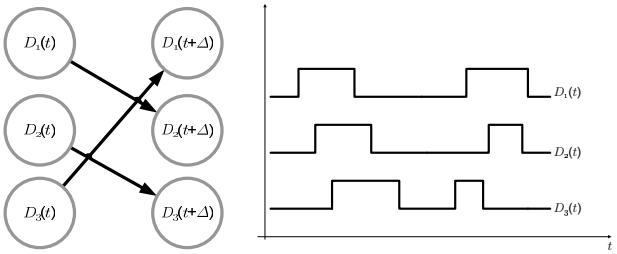

Figure 2.1. **(Left)** A segment of a (discrete time) dynamic Bayesian network with three servers. Note that this model does not contain any loops even if we consider the limit of $\Delta \to 0$. **(Right)** A sample drawn from (a) for servers D_1, D_2 and D_3.

and see [5] for a more general overview) but avoid the problems associated with the discretisation of time required for the application of conventional DBNs.

The paper is structured as follows: In Section 2 we introduce a Bayesian up-time model which can be viewed as a special case of the CTBN framework. In Section 3 we describe experiments on synthetic data to validate our model, compare the results to a state-of-the-art rule-based approach, and present our results for a server farm of a major Microsoft web site.

2. A Bayesian Up-time Model

2.1. Dynamic Bayesian Networks

In this section we describe our basic modelling approach which corresponds to a CTBN [6, 7] with at most one parent per node.

Consider N servers and denote the data of the ith server by D_i. In its purest form the data D_i is a list of times $t_{i,k} \in \mathbb{R}$ and event codes $e_{i,k}$ taking effectively[3] two different values `shutdown` and `start eventlog`. There exist a variety of reasons why a server might be stopped at any moment in time including power failure, server maintenance, software updates, an erroneous process, etc.

In a first attempt, we discretise the time axis into intervals of length Δ and measure the state $D_i(t) \in \{\uparrow, \downarrow, ?\} =: \mathcal{S}$ of the ith server at time $t = k\Delta$. In our particular case, we have three states: $\uparrow \equiv$ `server is running`, $\downarrow \equiv$ `server is down` and $? \equiv$ `server is in an unknown state`. The `unknown` state is necessary to take account of inconsistencies in the server event logs: A server can never be started/stopped twice in a row without having been stopped/started in between. Also, servers are sometimes taken off the network. We can specify a dependency structure between servers by modelling the transition probabilities between states $D_i(t)$ and $D_i(t+\Delta)$ given the states $\{D_j(t) \,|j \in \mathrm{pa}(i)\}$ of all the parents $\mathrm{pa}(i)$ of the ith server at time t. Note that such a network never exhibits loops because we condition every server's state at time $t+\Delta$ only on the states of servers at time t (see Figure 2.1). As a consequence, the entire network structure S can be fully represented by all parent relationships, $S := \{\mathrm{pa}(i) \,|i = 1, \ldots, N\}$.

In a Bayesian setting we are interested in computing the distribution over all possible network structures given the data $\{D_i\}$ of all servers, that is,

$$P(S\,|\{D_i\}) \;\propto\; P(\{D_i\}\,|S) \cdot P(S) = \left[\prod_{i=1}^{N} P(D_i|\,\{D_j\,|j \in \mathrm{pa}(i)\})\right] P(S)\;. \quad (2.1)$$

In this paper we shall only consider the case of at most one parent for every server, which already amounts to considering N^N possible network structures. We aim at finding the most likely network structure $S^*(\{D_i\}) := \mathrm{argmax}_S\, P(S\,|\{D_i\})$. Furthermore, our structural prior $P(S) = \prod_i P(\mathrm{pa}(i))$ factors over the N servers and each term $P(\mathrm{pa}(i)) \propto g(|\mathrm{pa}(i)|)$ is a (normalised) function of the number of parents only. Let us now focus on the data dependent terms $P(D_i)$ and $P(D_i\,|D_j)$.

2.2. Continuous Time Bayesian Networks

In the above argument, our resulting network structure S^* may be strongly dependent on the discretisation Δ. In order to overcome this artifact we consider the limit of $\Delta \to 0$ by introducing the notion of *hazard rates*. A hazard rate $\lambda(t)$ specifies the inverse average waiting time for the occurrence of an event, that is, a large hazard rate $\lambda(t)$ corresponds to high probability of the event in an infinitely small time interval around t (and vice versa). This is made more formal in the following proposition.

Proposition 2.1 (Generalised Exponential Distribution)**.** *Let $\lambda : \mathbb{R}^+ \to \mathbb{R}^+$ be any integrable function. The* generalised exponential distribution *is the limit distribution of the waiting time for an event where the average inverse waiting time at any moment $t \in \mathbb{R}^+$ is specified by $\lambda(t)$. It has the following density*

$$\forall d \in \mathbb{R}^+ : \quad \mathrm{GEx}(d;\lambda) := \frac{\lambda(d)\exp(-\Lambda(d))}{Z[\lambda]}\,,$$

where $\Lambda(d) := \int_0^d \lambda(x)\,dx$ and $Z[\lambda] := 1 - \lim_{d\to+\infty}\exp(-\Lambda(d))$.

[3] For the purpose of our analysis, we are grouping the events `clean shutdown` and `dirty shutdown` into `shutdown`.

Proof. For any $\Delta > 0$, let us divide the positive real line into consecutive intervals of length Δ and define $\pi(t,\Delta) := 1 - \exp(-\Delta\lambda(t))$ which specifies the probability that an event occurs in the time interval $[t, t+\Delta)$. Then, following the geometric distribution, the probability that an event occurs for the first time at $d = k\Delta$ is

$$P_\Delta(d) := \frac{\prod_{i=1}^{\frac{d}{\Delta}-1}(1-\pi(i\Delta,\Delta))\,\pi(d,\Delta)}{c(\Delta)},$$

where $c(\Delta)$ ensures normalisation. Taking limits and using l'Hospital's rule we find $\lim_{\Delta\to 0}(P_\Delta(d)/\Delta) = \mathrm{GEx}(d;\lambda)$. □

Going from event probabilities to hazard rates, the data is converted to a list of durations $d_{i,k} := t_{i,k+1} - t_{i,k}$. We shall assume that the hazard rate of the up-state $\uparrow$ is constant if the state of the parent does not change. Note that this assumption would be doubtful for the down-state $\downarrow$, which may depend on response time and availability of the system administrators. Let us characterise the piecewise exponential distribution which we will use to model up-times.

Proposition 2.2 (Piecewise Exponential Distribution)**.** *Suppose we are given l hazard rates $\boldsymbol{\lambda} \in (\mathbb{R}^+)^l$ and l sets $\mathcal{T}_k$ of non-overlapping intervals such that $\left(\bigcup_k \bigcup_{T\in\mathcal{T}_k} T\right) = \mathbb{R}^+$ and $\left(\bigcap_k \bigcap_{T\in\mathcal{T}_k} T\right) = \emptyset$ where the average waiting time in each of the intervals T in $\mathcal{T}_k$ is governed by the hazard rate λ_k. The waiting time distribution of the* piecewise exponential distribution *has the following density.*

$$\mathrm{PEx}(d;\boldsymbol{\lambda},\boldsymbol{\mathcal{T}}) := \prod_{k=1}^{l} \lambda_k^{a_k(d,\mathcal{T}_k)} e^{-\lambda_k b_k(d,\mathcal{T}_k)}, \quad (2.2)$$

$$a_k(d,\mathcal{T}_k) := \sum_{[t_0,t_1)\in\mathcal{T}_k} \mathbb{I}_{d\in[t_0,t_1)}, \quad (2.3)$$

$$b_k(d,\mathcal{T}_k) := \sum_{[t_0,t_1)\in\mathcal{T}_k} (t_1-t_0)\,\mathbb{I}_{d>t_1} + (d-t_0)\,\mathbb{I}_{d\in[t_0,t_1)}. \quad (2.4)$$

Proof. This follows from Proposition 2.1 using $\lambda(t) = \sum_{k=1}^{l} a_k(t,\mathcal{T}_k)\cdot\lambda_k$ which equals $\prod_{k=1}^{l}\lambda_k^{a_k(t,\mathcal{T}_k)}$ because for every t exactly one $a_k(t)$ is non-zero and equal to one. □

The appeal of this up-time model becomes apparent when considering that it still enjoys a conjugate prior w.r.t. $\boldsymbol{\lambda}$.

Proposition 2.3 (Conjugate Prior)**.** *Let* $\mathrm{PGa}(\boldsymbol{\lambda};\boldsymbol{\alpha},\boldsymbol{\beta})$ *denote the density of the* product of Gamma distribution,

$$\mathrm{PGa}(\boldsymbol{\lambda};\boldsymbol{\alpha},\boldsymbol{\beta}) := \prod_{k=1}^{l} \frac{\lambda_k^{\alpha_k-1}\exp(-\lambda_k\beta_k)}{Z(\alpha_k,\beta_k)}, \quad (2.5)$$

$$Z(\alpha,\beta) := \frac{\Gamma(\alpha)}{\beta^\alpha}. \quad (2.6)$$

Then, using (2.3) and (2.4), we have the following relationships

$$\begin{aligned} p(d\,|\boldsymbol{\lambda}) &= \mathrm{PEx}(d;\boldsymbol{\lambda},\boldsymbol{\mathcal{T}}), \\ p(\boldsymbol{\lambda}) &= \mathrm{PGa}(\boldsymbol{\lambda};\boldsymbol{\alpha},\boldsymbol{\beta}), \\ p(\boldsymbol{\lambda}\,|d) &= \mathrm{PGa}(\boldsymbol{\lambda};\boldsymbol{\alpha}+\mathbf{a}(d,\boldsymbol{\mathcal{T}}),\boldsymbol{\beta}+\mathbf{b}(d,\boldsymbol{\mathcal{T}})), \\ p(d) &= \prod_{k=1}^{l} Z(\alpha_k+a_k(d,\mathcal{T}_k),\beta_k+b_k(d,\mathcal{T}_k)). \end{aligned}$$

Proof. Since $p(\boldsymbol{\lambda}\,|d) \propto p(d\,|\boldsymbol{\lambda})\,p(\boldsymbol{\lambda})$ is a function of $\boldsymbol{\lambda}$, the update equations for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ follow directly from (2.2) and (2.5). Note that the normalisation constant is simply $p(d)$ whose form follows from (2.5). □

2.3. Structure from Failure

Returning to the original problem of learning a network structure S^*, we are now in a position to formulate the data-dependent terms $P(D_i\,|D_j)$ and $P(D_i)$. Note that we only model the up-time of the ith server and will subdivide each up-time interval $T_{i,k} = [0, d_{i,k})$ of length $d_{i,k}$ of the ith server into three non-overlapping sets of intervals $\boldsymbol{\mathcal{T}}_{i,k} = \{\mathcal{T}_{i,k}^{\uparrow}, \mathcal{T}_{i,k}^{\downarrow}\mathcal{T}_{i,k}^{?}\}$ depending on the state of the jth server in the interval $T_{i,k}$. Hence, using Proposition 2.3 we have

$$\begin{aligned} P(D_i\,|D_j) &= \prod_{s\in\mathcal{S}} Z\left(\alpha_s + \sum_{k=1}^{n_i} a_s\left(d_{i,k},\mathcal{T}_{i,k}^s\right), \beta_s + \sum_{k=1}^{n_i} b_s\left(d_{i,k},\mathcal{T}_{i,k}^s\right)\right), \\ P(D_i) &= Z\left(\alpha+n_i, \beta+\sum_{k=1}^{n_i} d_{i,k}\right). \end{aligned}$$

Note that these equations implicitly incorporate Occams' razor because of the constraints $\sum_{s\in\mathcal{S}} a_s(d_{i,k},\mathcal{T}_{i,k}^s) = 1$ and $\sum_{s\in\mathcal{S}} b_s(d_{i,k},\mathcal{T}_{i,k}^s) = d_{i,k}$. The more complex parent model $P(D_i\,|D_j)$ must provide a much better fit to the data than the independent model $P(D_i)$ in order to compensate for the additional parameters.

Choosing prior parameters for (α,β) and $(\alpha_s,\beta_s)_{s\in\mathcal{S}}$ induces an odds ratio of $\prod_{s\in\mathcal{S}} Z(\alpha_s,\beta_s)/Z(\alpha,\beta)$ which should effectively be incorporated into the

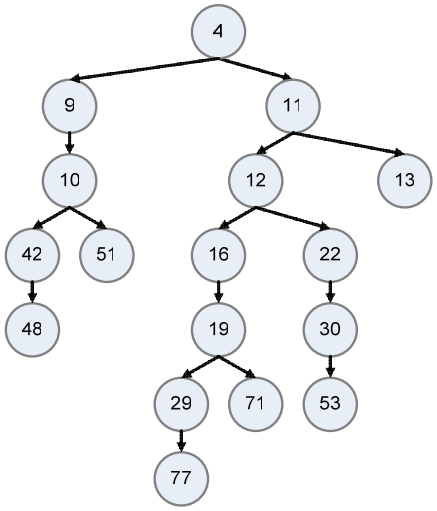

Figure 3.1. Learning a tree structure from two years of synthetic data. Results using CTBNs exactly recovering the ground truth.

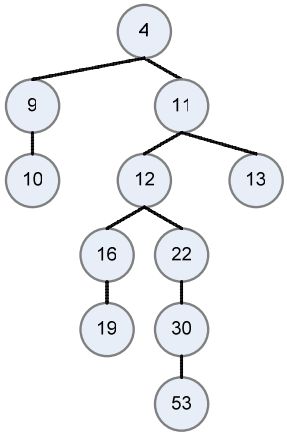

Figure 3.2. Learning a tree structure from two years of synthetic data. Results using the DOWS process (see also Figure 3.1 for comparison).

structural prior $P(S)$ from (2.1). We are now in a position to select a network structure $S^*(\{D_i\})$ by deciding with regard to the ith server whether there is sufficient evidence to justify a parent server $(P(D_i|D_j)P(\mathrm{pa}(i) = j) > P(D_i)P(\mathrm{pa}(i) = \emptyset)$ for at least one server j), and if so, which is the most likely parent server j. Repeating this procedure over all nodes leads to a (possibly cyclic) network structure S maximising the posterior over structures with at most one parent per node.

3. Experiments and Results

Since ground truth is not available for the task of determining the failure structure of large networks we pursue the following methodology. We study the properties of our up-time data model and the network model selection procedure on synthetic data generated from a distribution that plausibly represents data one may find in real world failure analysis. This experiment also serves to elucidate the relation between the CTBN model and Murphy's DOWS process [3]. After this validation step we apply both DOWS and our CTBN model to data from a server farm of a major Microsoft web site[4].

[4]Due to space restrictions, we only present a small subset of our results; the complete set of results can be found at *http://www.research.microsoft.com/~rherb/icml_failure.htm*. The resulting complex network structures can be browsed interactively including zoom/pan/search and node annotations using *Internet Explorer*.

3.1. The DOWS Process

DOWS refers to a state-of-the-art rule-based analysis process developed by B. Murphy in interaction with domain experts. After extracting and cleaning the event data, DOWS identifies related groups of events (bursts) based on temporal proximity governed by a duration parameter δt. The strength of a relationship between two nodes is calculated as a combination of different rule-based criteria including the number of co-occurring events in the same bursts and the string-distance between the server names. Note that a *direct* comparison with DOWS is difficult for two reasons. Firstly, DOWS uses extra information such as the server names, and secondly, DOWS aims at identifying short-term failure relationships between machines rather than general statistical dependencies.

3.2. Incorporation of Expert Knowledge

Incorporating prior knowledge of domain experts into data analysis can be difficult due to the typically intangible nature of human experience. In this particular case, it amounts to setting the parameters (α_s, β_s), $s \in \mathcal{S}$ for servers with a parent and (α, β) for servers without a parent. Clearly, setting the parameter α and β directly is difficult as they do not enjoy a meaningful interpretation individually. One idea that comes to mind is to let the expert decide on mean μ and variance σ^2 of the distribution of up-times and calculate the shape parameters as functions of μ and σ^2. Since this approach disregards the skew of the Gamma distribution we decided to specify the prior in terms of the symmetrical 10%- and 90%-quantiles $d_{0.1}, d_{0.9}$ of the up-time from which the shape paramet-

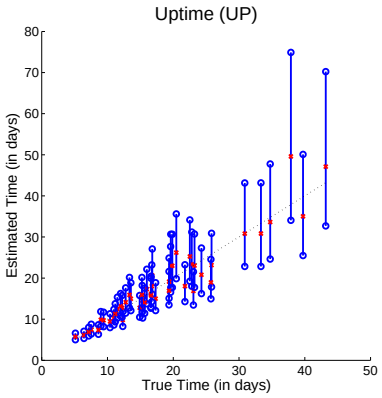

Figure 3.3. Learning parameters of the structure from two years of synthetic data. Each server with a parent has a true expected up-time (x-axis) and a posterior estimate (y-axis) shown as a 5%–95% posterior quantile interval with a mark indicating the median. Up-times for parent up.

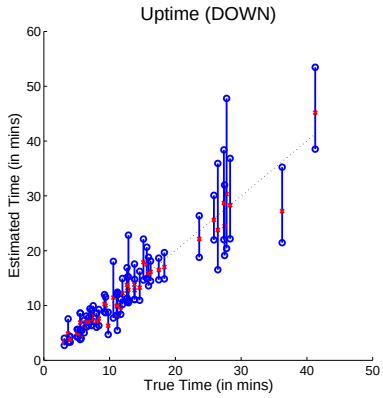

Figure 3.4. Learning parameters of the structure from two years of synthetic data. Each server with a parent has a true expected up-time (x-axis) and a posterior estimate (y-axis) shown as a 5%–95% posterior quantile interval with a mark indicating the median. Up-times for parent down.

ers $(\alpha, \beta)(d_{0.1}, d_{0.9})$ of the Gamma distribution can be obtained by Newton-Raphson.

3.3. Synthetic Data

We created a synthetic server event log by first sampling a random tree and generating random hazard rates for all servers in the random tree according to a Gamma prior. For each server, we then sampled random up-times according to its piecewise exponential distribution but fixed the down-time such that it covered 90% of the average up-time given a parent is in the down state $\downarrow$.

In one experiment we generated data for 100 servers over two years where the priors were set to $(\alpha, \beta)(7 \text{ days}, 25 \text{ days})$, $(\alpha_\uparrow, \beta_\uparrow)(10 \text{ days}, 30 \text{ days})$, $(\alpha_\downarrow, \beta_\downarrow)(5 \text{ mins}, 30 \text{ mins})$ and $(\alpha_?, \beta_?)(7 \text{ days}, 25 \text{ days})$. As a sanity check, we used the same prior for inferring the network structure and identified all parent relations correctly (see Figure 3.1 for the largest connected tree). Furthermore, for each server we compared the true hazard rates with the posterior estimates (see Figure 3.3 and 3.4) which consistently cover the true value within the 5%-95% quantile interval. In order to be able to compare these results to those of the DOWS process (see Figure 3.2) we took the DOWS matrix of pairwise relation strengths and used it to find the tree with the strongest relations. This tree was found to be a proper subtree of the ground truth indicating that the DOWS process implicitly uses similar distributional assumptions.

We decreased the length of the logging period down to 3 months (≈6 up-time samples per server) and chose a vague prior $(\alpha, \beta)(3 \text{ days}, 40 \text{ days})$, $(\alpha_\uparrow, \beta_\uparrow)(5 \text{ days}, 60 \text{ days})$, $(\alpha_\downarrow, \beta_\downarrow)(3 \text{ mins}, 120 \text{ mins})$ and $(\alpha_?, \beta_?)(3 \text{ days}, 40 \text{ days})$ for inference which resulted only in a 2% mismatch of the true parental relationship. In further experiments we found that in general the performance of the system degrades gracefully when altering the prior used for learning. In summary, our experiments validated that our method is capable of learning structure and parameters from very small data sets.

3.4. Major Microsoft Website

In a second experiment, we applied our method to event logs taken over five years from a major Microsoft web site. The site is comprised of 564 servers with various servers added and removed from the site at different times. With this problem, we set our structural prior $P(S)$ to 1:1,000,000,000 against the parent model. Despite these rather unfavourable odds, we found 105 parent relationships. Figure 4.2 (a) displays an exemplary structure discovered by our method. The structure constitutes a replicated tree, a configuration not uncommon in web server farms. In the absence of ground truth, similarities of server names indicate plausible relationships. For example, the loop in Figure 4.2 (a) is between servers `SEKDEXIRH01` and `SEKDEX[A]01`, with children servers `SEKDEX[A]02` and `SEKDEX[A]03`. Figure 4.2 (b) shows relations between some of the same servers as found by the DOWS process. We are currently in the process of "closing the loop" with the network administrators in order to validate the structures found by our methods and refine

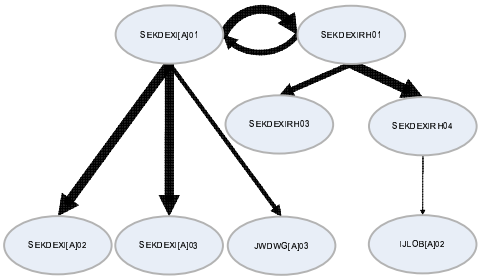

Figure 4.1. Learning a structure from five years of real world data of a major Microsoft web site. An exemplary structure learned using CTBNs.

them.

4. Discussion and Conclusions

We have brought the power of probabilistic modelling to the important problem of structural failure analysis in large scale computer networks. Continuous time Bayesian networks provide an appealing and efficient framework for analysing this data and can represent even loopy structures as found in tree-structured web farms with replication. Note, that our method finds *statistical* dependencies which can only provide as a basis for the discovery of causal failure relationships.

Ideally, one would like to extend the model to more than one parent per server. However, for a typical size of 1000 servers and at most k parents this requires the update, search and storage of $1000^{k+1} \times 3^k \times 2$ shape parameters corresponding to 72 Gigabyte of data for $k = 2$. Another interesting idea is to le[t nodes in the CTBN correspond to other, possibly external, influences such as time of day and day of week. Considering a small number of non-server influences in the structure learning *can* be done efficiently and is the focus of future work.

Acknowledgements Thanks to Patrick O'Broin for helping with data pre-processing.

References

[1] T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5:142–150, 1989.

[2] B. Levidow and B. Murphy. Windows 2000 dependability. Technical Report MSRC 2000-56, Microsoft Research, 1 Microsoft Way, Redmond, Washington, US, 2000.

[3] B. Murphy. Dependability of web server farms project, 2004. *http://www.research.microsoft.com/~bmurphy/DOWS.htm.*

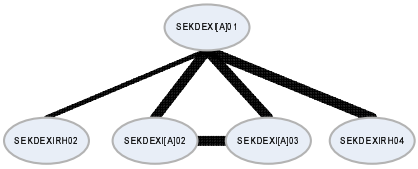

Figure 4.2. Learning a structure from five years of real world data of a major Microsoft web site. The corresponding structure to Figure 4.1 learned using DOWS.

[4] B. Murphy and T. Gent. Measuring system and software reliability using an automated data collection process. *Q & R Engineering International*, 11:341–353, 1995.

[5] K. P. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning.* PhD thesis, UC Berkeley, Computer Science Division, 2001.

[6] U. Nodelman, C. R. Shelton, and D. Koller. Continuous time Bayesian networks. In A. Darwiche and N. Friedman, editors, *Proc. of the 18th Intern. Conf. on Uncertainty in Artificial Intelligence*, pages 378–387. Morgan Kaufmann, 2002.

[7] U. Nodelman, C. R. Shelton, and D. Koller. Learning continuous time Bayesian networks. In C. Meek and U. Kjærulff, editors, *Proc. of the 19th Intern. Conf. on Uncertainty in Artificial Intelligence*, pages 451–458. Morgan Kaufmann, 2003.