Visualization of Analytical Processes

Ole J. Mengshoel
Carnegie Mellon University
Silicon Valley Campus

December 9, 2011
FODAVA Annual Review
Georgia Tech
Project Overview

Goal: Improve the visualization of analytical processes, in particular for probabilistic graphical (Bayesian network) models and other networks models.

Faculty: Mengshoel, Selker, and Ilic

Period of performance: 2009-2011 (currently no-cost extension)
Areas of Research

Probabilistic and statistical models, algorithms:

Probabilistic graphical models: Bayesian networks, …
Inference: Diagnosis, prognosis, …
Machine learning

Interactive visualization:

Network visualization
Multi-view, Multi-focus, …

Stochastic and randomized algorithms:

Stochastic local search
Evolutionary algorithms

Applications and demonstrations:

Challenging and large-scale applications, multi-media data sets, …
Scalability of algorithms, visualizations, user interactions, …
Research Directions

Electrical Power System Diagnosis using Probabilistic Computation

Fast Belief Propagation Using GPU Parallelization in Junction Trees

Understanding Scalability of Bayesian Network Computation using Junction Tree Growth Curves

Multi-Fisheye, Multi-View for Interactive Visualization of Large Networks

Stochastic Search for Computing Most Probable Explanations in Bayesian Networks
Bayesian Network Inference

- Bayesian network inference answers these queries:
  - *Marginal/MLV*: Given evidence at some nodes, infer posterior probability/most likely value (MLV) over one node
  - *Most probable explanation (MPE)*: Given evidence, find explanation with greatest probability over remaining nodes
  - *Maximum a posteriori probability (MAP)*: Given evidence, find explanation with greatest probability over some nodes

- Computational hardness [Cooper, 1990; Shimony, 1994; Roth 1996]:
  - Care is needed, in modeling, machine learning, and inference

- Inference algorithms:
  - *Approximate*: Stochastic local search [Kask & Dechter, 1999; Mengshoel, 1999; Mengshoel 2008]; Variational inference; …
On September 2, 1998, Swissair 111 crashed into the Atlantic Ocean, killing all 229 people onboard. Probably, wires short-circuited and led to a fire.

A battery failure occurred on the Mars Global Surveyor on November 2, 2006. A software error caused the battery to overheat due to over-exposure to sunlight.

In 1999, the Mars Polar Lander crashed into the surface of Mars, most likely due to a premature engine shutdown because of spurious lander leg signals.

For the Mars rover SPIRIT, a full on-board file system caused reboot-loop after landing.

On June 4, 1996, software on the Ariane V rocket, reused from Ariane IV, overflowed and lead to its destruction.
Multi-View Overview+Detail for Networks

**OBJECTIVE**
Making multiple and multi-step comparisons across different parts of a data corpus and across multiple representational levels in a complex data set.

**FEATURES**
- Visual encoding of data properties
- Overview + detail
- Multi-focus + context
- Bubbles anchoring node information to the network

**RESULTS**
In experiments with data from an electrical power network we demonstrated how NetEx makes fault diagnosis easier.
Results from a **user study** with 25 subjects suggests that NetEx enables more accurate isolation of faults in multi-fault situations.
Multi-View Focus+Context for Networks

OBJECTIVE
Improve the applicability of multi-fisheye to exploration of labeled networks, including Bayesian network (BN) problem instances. Focus on large-scale but in-memory networks.

DESCRIPTION
A focus+context visualization tool that supports visualization of multiple fisheye distortions in network (Bayesian networks, for example). Voronoi edges separate the fisheyes, and data boxes with details (such as Bayesian network conditional probability tables) are created for fisheyes and their neighboring nodes.

RESULTS
The tool supports interactive and simultaneous creation of up to 10-20 readable node labels by means of fisheye distortion in large-scale (Bayesian) networks. Node context, including network edge connection patterns and relative location, is preserved.
Belief Propagation by Fast GPU Message Passing in Junction Trees

**OBJECTIVE**
Speed up Bayesian network computation when junction trees are being used; use graphics processing units (GPUs).

**DESCRIPTION**
An algorithm in which message passing is performed in parallel, benefiting situations with large cliques and large separators.

**RESULTS**
Analytical and experimental speed up – best experimental speed up result to date is 918%.

\[
\text{Speedup} = \frac{\sum_i \sum_{k \in 
\text{Ne}(c_i)} (|\phi_{x_i}^k| + |\phi_{x_k}^i|)}{2(n - 1)\tau + \sum_i \sum_{k \in 
\text{Ne}(c_i)} \frac{|\phi_{x_i}^k| + |\phi_{x_k}^i|}{|\phi_{S_{ik}}|}}.
\]
Bayesian Methods for Diagnostics (1)

**Tasks**
- Develop probabilistic diagnosis approach, ProDiagnose: Auto-generation of Bayesian network; Compilation of Bayesian networks to real-time arithmetic circuits; Diagnose discrete and continuous faults on-line.

**Vision**
- Tackle system health management and diagnostic challenges:
  - Large & complex systems; Hybrid systems (discrete & continuous behavior); Hard diagnostic problems; Real time requirements.

**Electrical Power System**
- Voltage
- Battery
- Breaker
- Relay
- Load
- Current
- Status
- Feedback
- Temp
- Wire

**Arithmetic Circuit (AC)**
- Offline Compilation
- Off-Line Phase
- Offline Generation
- Bayesian Network (BN)
- System Specification
- Our ProDiagnose algorithm & software
- Online Inference
- Online Phase
- Sensor, Commands
- Diagnosis: MLV, MPE, or MAP

Carnegie Mellon University Silicon Valley
Bayesian Methods for Diagnostics (2)

1. **ADAPT – Electrical power system testbed at NASA ARC.**

2. **Two conditions:**
   
   Our novel cumulative sum (CUSUM) technique (i) enabled or (ii) disabled.

<table>
<thead>
<tr>
<th>Metric</th>
<th>CUSUM Enabled</th>
<th>CUSUM Disabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detection Accuracy</td>
<td>92.31%</td>
<td>46.15%</td>
</tr>
<tr>
<td>False Positives Rate</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>False Negatives Rate</td>
<td>8.82%</td>
<td>61.76%</td>
</tr>
<tr>
<td>Mean Time To Detect</td>
<td>17.97 s</td>
<td>28.36 s</td>
</tr>
<tr>
<td>Mean Time To Isolate</td>
<td>72.27 s</td>
<td>51.14 s</td>
</tr>
</tbody>
</table>

Bayesian Reasoning for Diagnostics: *Operates in a state space of size > $2^{500}$ in time < 1 ms.*
GPUs for Speeding up Bayesian Network Computation
Parallel and Distributed Computing

Graphics processing units (GPUs): Promise to dramatically up the performance of processing in the cloud and on the mobile device.

Speed up performance of processing in the cloud – integration with analytics software.

GPUs are moving onto mobile devices, and within the next year or two we expect them to be programmable through CUDA or other programming languages.
Motivation and Approach

• Belief propagation in junction trees may be computationally intensive due to:
  o The topology and connectedness of Bayesian networks
  o High cardinality of one or more nodes in cliques with sufficiently high number of nodes

• Observations:
  o During message passing, computations associated with different separator tables are independent
  o Some junction trees contain large cliques and separators

• Our approach:
  o Compute each message in parallel
  o Substantial parallelism opportunity when neighboring cliques and separators are large
  o Non-invasive embedding in original junction tree message passing algorithms
Fast Message Passing

Step 1: Marginalization
\[ \phi_{S_{ik}}^* = \sum \frac{\phi \chi_i}{x_i / S_{ik}} \]

Step 2: Scattering
\[ \phi_{S^*_{ik}} = \phi \chi_e \frac{\phi_{S_{ik}}^*}{\phi_{S_{ik}}} \]
Algorithm 1 \textit{Message-Passing}(\phi x_i, \phi x_k, \phi s_{ik})

\textbf{Input:} \phi x_i, \phi x_k, \phi s_{ik}.

for $j = 1$ to $|\phi s_{ik}|$ in parallel do
    sep\_star=0;
    for $n = 1$ to $|\mu \phi x_i, s_j|$ do
        sep\_star[j] = sep\_star[j] + $\phi x_i(\mu x_i, s_j[n])$
    end for
    for $n = 1$ to $|\mu \phi x_k, s_j|$ do
        $\phi x_k(\mu x_k, s_j[n]) = \frac{\text{sep\_star}[j]}{\phi s_{ik}[j]} \phi x_k(\mu x_k, s_j[n])$
    end for
end for

\[\text{Speedup} = \frac{\sum_i \sum_{k \in Ne(C_i)} (|\phi x_i| + |\phi x_k|)}{2(n - 1)\tau + \sum_i \sum_{k \in Ne(C_i)} \frac{(|\phi x_i| + |\phi x_k|)}{|\phi s_{ik}|}}.\]
GPU Parallelization: Experiments

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Mildew</th>
<th>Diabetes</th>
<th>Barley</th>
<th>Pigs</th>
<th>Mumin2</th>
<th>Mumin3</th>
<th>Mumin4</th>
<th>Water</th>
</tr>
</thead>
<tbody>
<tr>
<td># of JT nodes</td>
<td>28</td>
<td>337</td>
<td>36</td>
<td>368</td>
<td>860</td>
<td>904</td>
<td>872</td>
<td>20</td>
</tr>
<tr>
<td>Max. CPT size</td>
<td>4,372,480</td>
<td>190,080</td>
<td>7,257,600</td>
<td>177,147</td>
<td>504,000</td>
<td>156,800</td>
<td>784,000</td>
<td>995,328</td>
</tr>
<tr>
<td>Min. CPT size</td>
<td>336</td>
<td>495</td>
<td>216</td>
<td>27</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>Ave. CPT size</td>
<td>341,651</td>
<td>32,443</td>
<td>512,044</td>
<td>1,927</td>
<td>5,653</td>
<td>3,443</td>
<td>16,444</td>
<td>173,297</td>
</tr>
<tr>
<td>Max. SPT size</td>
<td>71,680</td>
<td>11,880</td>
<td>907,200</td>
<td>59,049</td>
<td>72,000</td>
<td>22,400</td>
<td>112,000</td>
<td>147,456</td>
</tr>
<tr>
<td>Min. SPT size</td>
<td>72</td>
<td>16</td>
<td>7</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Ave. SPT size</td>
<td>9,273</td>
<td>1,845</td>
<td>39,318</td>
<td>339</td>
<td>713</td>
<td>553</td>
<td>2,099</td>
<td>26,065</td>
</tr>
<tr>
<td>BP on GPU [ms]</td>
<td>53</td>
<td>94</td>
<td>106</td>
<td>75</td>
<td>125</td>
<td>104</td>
<td>342</td>
<td>52</td>
</tr>
<tr>
<td>BP on CPU [ms]</td>
<td>355</td>
<td>397</td>
<td>974</td>
<td>51</td>
<td>210</td>
<td>137</td>
<td>473</td>
<td>120</td>
</tr>
<tr>
<td>Speedup</td>
<td>6.70</td>
<td>4.22</td>
<td>9.19</td>
<td>0.68</td>
<td>1.68</td>
<td>1.32</td>
<td>1.38</td>
<td>2.31</td>
</tr>
</tbody>
</table>
System Health Management using Bayesian Networks
Architecture using Bayesian Networks

Each health variable has at least two states (healthy and faulty), thus enabling the diagnoses of zero, one, two, or more faults.

**Specification language**

- Battery1 : battery : 0.0005;
- Wire1 : wire : 0.0000 : Battery1;
- Voltage1 : sensorVoltage : 0.0005 : Wire1;
- Current1 : sensorCurrent : 0.0005 : Wire1;
- Breaker1 : breaker : 0.0005 : Wire1;
- Status1 : sensorTouch : 0.0005 : Breaker1;
- Wire2 : wire : 0.0000 : Breaker1;
- Relay1 : relay : 0.0005 : Wire2;
- Feedback1 : sensorTouch : 0.0005 : Relay1;
- Load1 : load : 0.0005 : Relay1;
- Temp1 : sensorCurrent : 0.0005 : Load1;

Bayesian network

Arithmetic circuit

The ProDiagnose algorithm.
Fault Types

Independent faults
- Abrupt
  - Permanent
  - Discrete
  - Continuous (parametric)
- Intermittent
- Drift (incipient)

Dependent faults
- Common cause
- Cascading

Bayesian networks in general

Problem-1 (DP1) and Problem-2 (DP2) of diagnostics challenge

Continuous drift fault

Continuous abrupt (offset) fault
Cumulative Sum (CUSUM)

Mathematical definition of CUSUM:

\[ \delta_p(t) = [s_p(t) - s_p(t - 1)] + \delta_p(t - 1) \]

Graph illustrating CUSUM on current readings:

- The blue and orange plots represent the raw sensor readings (span of 4 minutes).
- The red and green plots represent the CUSUM values of these respective raw sensor readings.
- The vertical dotted line represents the time of fault injection.
- Benefit of CUSUM: It amplifies and normalizes signals of potentially faulty behavior.
CUSUM – Continuous Offset Faults

Nominal case

Fault case
Experimental Bayesian Network

Summary Statistics:
• DP1 Bayesian network:
  • Nodes: 148
  • Edges: 176
  • Cardinality: [2, 10]

Hypothesis: Similar networks can be constructed (by expert, machine learning, or combination) to detect, diagnose, predict, and mitigate in a broad range of systems.
Experiments, Simulated ADAPT Data

Comparison between Arithmetic Circuit Evaluation (ACE), Variable Elimination (VE) and Clique Tree Propagation (CTP)

Main conclusions:

– All three inference algorithms are quite efficient, thanks to auto-generation algorithm

– ACE outperforms VE (for MPE) and CTP (for marginals), both in Mean and St. Dev.

<table>
<thead>
<tr>
<th>Inference Time (ms)</th>
<th>MPE VE</th>
<th>ACE</th>
<th>Marginals JTP</th>
<th>ACE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum</td>
<td>19.30</td>
<td>0.2235</td>
<td>9.792</td>
<td>0.5721</td>
</tr>
<tr>
<td>Maximum</td>
<td>40.21</td>
<td>2.5411</td>
<td>65.34</td>
<td>5.9228</td>
</tr>
<tr>
<td>Median</td>
<td>19.81</td>
<td>0.2260</td>
<td>10.52</td>
<td>0.6006</td>
</tr>
<tr>
<td>Mean</td>
<td>20.13</td>
<td>0.2625</td>
<td>11.01</td>
<td>0.7854</td>
</tr>
<tr>
<td>St. Dev.</td>
<td>1.554</td>
<td>0.2028</td>
<td>4.101</td>
<td>0.6970</td>
</tr>
</tbody>
</table>

ACE is the approach used in ProDiagnose
Experiments, ADAPT Power System

Results summary (CUSUM enabled):

- DXC-10 training set
- Detection accuracy doubled
- False negative rate greatly improved
- Improvement in average detection time
- Average isolation time increased
  - The DX competition specifies that no isolation time be recorded for an incorrect mis-diagnosis.

<table>
<thead>
<tr>
<th>Metric</th>
<th>ADAPT DXC Tier 1</th>
<th>ADAPT DXC Tier 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ProADAPT</td>
<td>RODON</td>
</tr>
<tr>
<td>False positives (FP) rate</td>
<td>0.0333</td>
<td>0.0645</td>
</tr>
<tr>
<td>False negatives (FN) rate</td>
<td>0.0313</td>
<td>0.0968</td>
</tr>
<tr>
<td>Detection accuracy</td>
<td>0.9677</td>
<td>0.9194</td>
</tr>
<tr>
<td>Classification errors</td>
<td>2.0</td>
<td>10.0</td>
</tr>
<tr>
<td>Mean time to detect $T_d$ (ms)</td>
<td>1,392</td>
<td>218</td>
</tr>
<tr>
<td>Mean time to isolate $T_i$ (ms)</td>
<td>4,084</td>
<td>7,205</td>
</tr>
<tr>
<td>Mean CPU time $T_c$ (ms)</td>
<td>1,601</td>
<td>11,766</td>
</tr>
<tr>
<td>Mean peak memory usage (kb)</td>
<td>1,680</td>
<td>26,679</td>
</tr>
<tr>
<td>Score</td>
<td>72.80</td>
<td>59.85</td>
</tr>
<tr>
<td>Rank</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metric</th>
<th>CUSUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detection Accuracy</td>
<td>92.31%</td>
</tr>
<tr>
<td>False Positives Rate</td>
<td>0%</td>
</tr>
<tr>
<td>False Negatives Rate</td>
<td>8.82%</td>
</tr>
<tr>
<td>Mean Time To Detect</td>
<td>17.97 s</td>
</tr>
<tr>
<td>Mean Time To Isolate</td>
<td>72.27 s</td>
</tr>
</tbody>
</table>

9 competitors in Tier 1.  
6 competitors in Tier 2.
Scalability of Bayesian Network Computation
Bipartite Bayesian Networks

The number of sensors in mobile devices and infrastructure have increased dramatically. Are we taking full advantage of them, to understand the behavior of users as well as the communication and computation infrastructure?

Root nodes are estimated (output nodes): represent whether components or systems are working or failing; type of user behavior (fraudulent or legitimate); etc.

Leaf nodes are observed (input nodes): represent detectors / sensors / observables / tests
Clique Tree Clustering

Tree clustering: a major approach to BN inference

Tree clustering algorithms employ two phases:

Compilation: generate clique tree $\beta''$ from BN $\beta$

Propagation: do belief revision (MPEs) or belief updating (marginals) by propagation of evidence in $\beta''$

Details in [Lauritzen & Spiegelhalter, 88].
Gompertz Growth Curves

Gompertz growth curve:
\[ g(x) = g(\infty)e^{-\zeta e^{-\gamma x}} \]

- \( g_1(x) \) to \( g_3(x) \): Shift growth curve to right by increasing \( \zeta \) from \( \zeta = 5 \) to \( \zeta = 15 \).

- \( g_1(x) \) to \( g_2(x) \): Decrease maximal growth rate by decreasing \( \gamma \) from \( \gamma = 0.3 \) to \( \gamma = 0.2 \).
Growth of Bayesian Networks

Number of sensors - Bayesian network leaf nodes

Total Gompertz growth curve for $\text{BPART}(V, C, P, S)$:

$$g_T(x) = S^V e^{-\zeta e^{-\kappa x}} + xS^{P+1}$$
Comparing Growth Curves

Gompertz growth curve:

\[ g(x) = 2^{30} \times \exp(-19.14 \times \exp(-0.005874x)) \]

BNs of varying hardness generated with parameters \( V=30, S=2, P=2 \), and varying \( C \).
Current and Planned Work

Analytics:
- Improve Expectation Maximization (EM) algorithms for Bayesian network parameter estimation – exploit parallelism in modern hardware and software architectures
- Approach 1: Develop EM layer ”on top of” improved GPU-based approach to junction tree propagation
- Approach 2: Use MapReduce to explore data parallelism in Bayesian network parameter

Visualization:
- Improvements to current multi-focus, multi-view network visualizations
- Integration of novel and existing of analytics and visualization techniques

Experiments, demonstrations, and software:
- ADAPT datasets and Bayesian networks
- Synthetic Bayesian networks ("similar to ADAPT") and other Bayesian networks
- Other network data sets: VAST challenge; disaster and emergency management, social network data, …
- Hardening and distribution of Java software


O. J. Mengshoel, D. C. Wilkins, and D. Roth, “Initialization and Restart in Stochastic Local Search: Computing a Most Probable Explanation in Bayesian Networks”, IEEE Trans. on Knowledge and Data Engineering, 2011.


