Scalable MAP inference in Bayesian networks based on a Map-Reduce approach

Darío Ramos-López¹, Antonio Salmerón¹, Rafael Rumí¹, Ana M. Martínez², Thomas D. Nielsen², Andrés R. Masegosa³, Helge Langseth³, Anders L. Madsen²,4

¹Department of Mathematics, University of Almería, Spain
²Department of Computer Science, Aalborg University, Denmark
³Department of Computer and Information Science, The Norwegian University of Science and Technology, Norway
⁴Hugin Expert A/S, Aalborg, Denmark
Outline

1 Motivation
2 MAP in CLG networks
3 Scalable MAP
4 Experimental results
5 Conclusions
Outline

1 Motivation

2 MAP in CLG networks

3 Scalable MAP

4 Experimental results

5 Conclusions
Motivation

▶ **Aim:** Provide scalable solutions to the MAP problem.

▶ **Challenges:**
  ▶ Data coming in streams at high speed, and a quick response is required.
  ▶ For each observation in the stream, the most likely configuration of a set of variables of interest is sought.
  ▶ MAP inference is highly complex.
  ▶ Hybrid models come along with specific difficulties.
The AMiDST project: Analysis of MassIve Data STreams
http://www.amidst.eu

- Large number of variables
- Queries to be answered in real time
- Hybrid Bayesian networks (involving discrete and continuous variables)
  - Conditional linear Gaussian networks
Outline

1 Motivation

2 MAP in CLG networks

3 Scalable MAP

4 Experimental results

5 Conclusions
Conditional Linear Gaussian networks

\[ P(Y) = (0.5, 0.5) \]
\[ P(S) = (0.1, 0.9) \]
\[ f(w | Y = 0) = \mathcal{N}(w; -1, 1) \]
\[ f(w | Y = 1) = \mathcal{N}(w; 2, 1) \]
\[ f(t | w, S = 0) = \mathcal{N}(t; -w, 1) \]
\[ f(t | w, S = 1) = \mathcal{N}(t; w, 1) \]
\[ f(u | w) = \mathcal{N}(u; w, 1) \]
Querying a Bayesian network

Belief update: Computing the posterior distribution of a variable:

\[
p(x_i|x_E) = \frac{\sum_{x_D} \int_{x_C} p(x, x_E) dx_C}{\sum_{x_{D_i}} \int_{x_{C_i}} p(x, x_E) dx_{C_i}}
\]

Maximum a posteriori (MAP): For a set of target variables \(X_I\), seek

\[
x_i^* = \arg \max_{x_i} p(x_I|X_E = x_E)
\]

where \(p(x_I|X_E = x_E)\) is obtained by first marginalizing out from \(p(x)\) the variables not in \(X_I\) and not in \(X_E\)

Most probable explanation (MPE): A particular case of MAP where \(X_I\) includes all the unobserved variables
Querying a Bayesian network

- **Belief update:** Computing the posterior distribution of a variable:

\[
p(x_i|x_E) = \frac{\sum \int_{x_C} p(x, x_E) dx_C}{\sum \int_{x_{C_i}} p(x, x_E) dx_{C_i}}
\]

- **Maximum a posteriori (MAP):** For a set of target variables \(X_I\), seek

\[
x_i^* = \arg\max_{x_i} p(x_I|X_E = x_E)
\]

where \(p(x_I|X_E = x_E)\) is obtained by first marginalizing out from \(p(x)\) the variables not in \(X_I\) and not in \(X_E\)

- **Most probable explanation (MPE):** A particular case of MAP where \(X_I\) includes all the unobserved variables
Outline

1 Motivation

2 MAP in CLG networks

3 Scalable MAP

4 Experimental results

5 Conclusions
MAP by Hill Climbing

\textsc{HC\_MAP}(B, x_I, x_E, r)

\textbf{while stopping criterion not satisfied} \textbf{do}

\hspace{1em} x_I^* \leftarrow \text{GenerateConfiguration}(B, x_I, x_E, r)

\hspace{1em} \textbf{if} \quad p(x_I^*, x_E) \geq p(x_I, x_E) \quad \textbf{then}

\hspace{2em} x_I \leftarrow x_I^*

\hspace{1em} \textbf{end}

\textbf{end}

\textbf{return} \quad x_I
MAP by Hill Climbing

\[ \text{HC\_MAP}(B, x_I, x_E, r) \]

while stopping criterion not satisfied do

\[ x_I^* \leftarrow \text{GenerateConfiguration}(B, x_I, x_E, r) \]

if \[ p(x_I^*, x_E) \geq p(x_I, x_E) \] then

\[ x_I \leftarrow x_I^* \]

end

end

return \[ x_I \]

- Max. number of non-improving iterations
- Target prob. threshold
- Max. number of iterations
MAP by Hill Climbing

\[
\text{HC\_MAP}(B, x_I, x_E, r)
\]

\[
\text{while stopping criterion not satisfied do}
\]

\[
x_I^* \leftarrow \text{GenerateConfiguration}(B, x_I, x_E, r)
\]

\[
\text{if } p(x_I^*, x_E) \geq p(x_I, x_E) \text{ then}
\]

\[
x_I \leftarrow x_I^*
\]

\[
\text{end}
\]

\[
\text{end}
\]

return \( x_I \)

\[\text{\textbf{Never move to a worse configuration}}\]
MAP by Hill Climbing

\[ \text{HC}_\text{MAP}(B, x_I, x_E, r) \]

while stopping criterion not satisfied do
\[ x_I^* \leftarrow \text{GenerateConfiguration}(B, x_I, x_E, r) \]
\[ \text{if } p(x_I^*, x_E) \geq p(x_I, x_E) \text{ then} \]
\[ x_I \leftarrow x_I^* \]
end

end

return

\[ p(x_I, x_E) = \sum_{x^* \in \Omega_{x^*}} p(x_I, x_E, x^*) = \sum_{x^* \in \Omega_{x^*}} \frac{p(x_I, x_E, x^*)}{f^*(x^*)} f^*(x^*) \]
\[ = \mathbb{E}_{f^*} \left[ \frac{p(x_I, x_E, x^*)}{f^*(x^*)} \right] \approx \frac{1}{n} \sum_{i=1}^{n} \frac{p(x_I, x_E, x^{*(i)})}{f^*(x^{*(i)})}, \]
MAP by Hill Climbing

\[
\text{HC\_MAP}(B, x_I, x_E, r)
\]

\[
\text{while stopping criterion not satisfied do}
\]

\[
x^*_I \leftarrow \text{GenerateConfiguration}(B, x_I, x_E, r)
\]

\[
\text{if } p(x^*_I, x_E) \geq p(x_I, x_E) \text{ then}
\]

\[
x_I \leftarrow x^*_I
\]

\[
\text{end}
\]

\[
\text{end}
\]

\[
\text{return } x_I
\]

▶ We’ll see later
MAP by Simulated Annealing

\[
\text{SA\_MAP}(B, x_I, x_E, r)
\]

\[
T \leftarrow 1000; \quad \alpha \leftarrow 0.90; \quad \varepsilon > 0
\]

while \( T \geq \varepsilon \) do
    \[
x_I^* \leftarrow \text{GenerateConfiguration}(B, x_I, x_E, r)
    \]
    Simulate a random number \( \tau \sim U(0, 1) \)
    if \( p(x_I^*, x_E) > p(x_I, x_E)/(T \cdot \ln(1/\tau)) \) then
        \[
x_I \leftarrow x_I^*
        \]
    end
    \[
T \leftarrow \alpha \cdot T
\]
end
return \( x_I \)
MAP by Simulated Annealing

\[
\text{SA}_\text{MAP}(B, \mathbf{x}_I, \mathbf{x}_E, r)
\]

\[
T \leftarrow 1000; \quad \alpha \leftarrow 0.90; \quad \varepsilon > 0
\]

\[\text{while } T \geq \varepsilon \text{ do}\]
\[\mathbf{x}_i^* \leftarrow \text{GenerateConfiguration}(B, \mathbf{x}_I, \mathbf{x}_E, r)\]
\[
\text{Simulate a random number } \tau \sim \mathcal{U}(0, 1)
\]
\[\text{if } p(\mathbf{x}_i^*, \mathbf{x}_E) > p(\mathbf{x}_I, \mathbf{x}_E)/(T \cdot \ln(1/\tau)) \text{ then}\]
\[\mathbf{x}_I \leftarrow \mathbf{x}_i^*\]
\[\text{end}\]
\[T \leftarrow \alpha \cdot T\]
\[\text{end}\]

return \(\mathbf{x}_I\)

- Default values of the temperature parameters
  - \(T \gg 1\): almost completely random
  - \(T \ll 1\): almost completely greedy
MAP by Simulated Annealing

\[ \text{SA\_MAP}(B, x_I, x_E, r) \]
\[ T \leftarrow 1000; \alpha \leftarrow 0.90; \varepsilon > 0 \]
while \( T \geq \varepsilon \) do
\[ x^*_I \leftarrow \text{GenerateConfiguration}(B, x_I, x_E, r) \]
Simulate a random number \( \tau \sim \mathcal{U}(0, 1) \)
if \( p(x^*_I, x_E) > p(x_I, x_E)/(T \cdot \ln(1/\tau)) \) then
\[ x_I \leftarrow x^*_I \]
end
\[ T \leftarrow \alpha \cdot T \]
end
return \( x_I \)

- Accept \( x^*_I \) if its prob. increases or decreases \( < T \cdot \ln(1/\tau) \)
MAP by Simulated Annealing

\[ \text{SA}_\text{MAP}(B, x_I, x_E, r) \]

\[ T \leftarrow 1000; \ \alpha \leftarrow 0.90; \ \varepsilon > 0 \]

\[ \text{while } T \geq \varepsilon \text{ do} \]
\[ \quad x_I^* \leftarrow \text{GenerateConfiguration}(B, x_I, x_E, r) \]
\[ \quad \text{Simulate a random number } \tau \sim \mathcal{U}(0, 1) \]
\[ \quad \text{if } p(x_I^*, x_E) > p(x_I, x_E)/(T \cdot \ln(1/\tau)) \text{ then} \]
\[ \quad \quad x_I \leftarrow x_I^* \]
\[ \quad \text{end} \]
\[ \quad T \leftarrow \alpha \cdot T \]
\[ \text{end} \]

\[ \text{return } x_I \]

\[ \text{Cool down the temperature} \]
Generating a new configuration of variables

Only discrete variables

- The new values are chosen at random
Generating a new configuration of variables

Hybrid models

- We take advantage of the properties of the CLG distribution

\[ P(Y) = (0.5, 0.5) \]
\[ f(w|Y=0) = N(w; -1, 1) \]
\[ f(w|Y=1) = N(w; 2, 1) \]
\[ f(t|w, S=0) = N(t; -w, 1) \]
\[ f(t|w, S=1) = N(t; w, 1) \]
\[ f(u|w) = N(u; w, 1) \]
Generating a new configuration of variables

Hybrid models
- We take advantage of the properties of the CLG distribution

A variable whose parents are discrete or observed

\[ P(Y) = (0.5, 0.5) \]
\[ P(S) = (0.1, 0.9) \]
\[ f(w|Y = 0) = \mathcal{N}(w; -1, 1) \]
\[ f(w|Y = 1) = \mathcal{N}(w; 2, 1) \]
\[ f(t|w, S = 0) = \mathcal{N}(t; -w, 1) \]
\[ f(t|w, S = 1) = \mathcal{N}(t; w, 1) \]
\[ f(u|w) = \mathcal{N}(u; w, 1) \]
Generating a new configuration of variables

Hybrid models

- We take advantage of the properties of the CLG distribution

A variable whose parents are discrete or observed

Return its conditional mean:

\[
f(w|Y = 0) = \mathcal{N}(w; -1, 1) \]
\[
f(w|Y = 1) = \mathcal{N}(w; 2, 1) \]
Generating a new configuration of variables

Hybrid models

- We take advantage of the properties of the CLG distribution

A variable with unobserved continuous parents

Simulate a value using its conditional distribution:

\[ f(t|w, S = 0) = \mathcal{N}(t; -w, 1) \]

\[ f(t|w, S = 1) = \mathcal{N}(t; w, 1) \]
Scalable implementation

Evidence $x_E$ → Map → Reduce → MAP estimate $x^*_I$

Multiple starting points → Local solutions
Outline

1 Motivation

2 MAP in CLG networks

3 Scalable MAP

4 Experimental results

5 Conclusions
Experimental analysis

Purpose

Analyze the scalability in terms of

- Speed
- Accuracy
Experimental analysis

Purpose

Analyze the scalability in terms of
- Speed
- Accuracy

Experimental setup

- Synthetic networks with 200 variables (50% discrete)
- 70% of the variables observed at random
- 10% of the variables selected as target \( \Rightarrow \) 20% to be marginalized out
Computing environment

- **AMIDST Toolbox** with Apache Flink
- **Multi-core** environment based on a dual-processor AMD Opteron 2.8 GHz server with 32 cores and 64 GB of RAM, running Ubuntu Linux 14.04.1 LTS
- **Multi-node** environment based on Amazon Web Services (AWS)
Scalability: run times

Scalability of MAP in a multi-core node

Scalability of MAP in a multi-node cluster

Execution time (s)

Number of cores

Execution time (s)

Number of nodes (4 vCPUs per node)

Speed-up factor

Method: HC Global, HC Local, SA Global, SA Local
Estimated log-probabilities of the MAP configurations found by each algorithm
Scalability: accuracy (Hill Climbing)

Estimated log-probabilities of the MAP configurations found by each algorithm
Conclusions

- Scalable MAP for CLG models in terms of accuracy and run time
- Available in the AMIDST Toolbox
- Valid for multi-cores and cluster systems
- MapReduce-based design on top of Apache Flink
Thank you for your attention

You can download our open source Java toolbox:

http://www.amidsttoolbox.com

Acknowledgments: This project has received funding from the European Union’s Seventh Framework Programme for research, technological development and demonstration under grant agreement no 619209