Treedy: A Heuristic for Counting and Sampling Subsets

Teppo Niinimäki, Mikko Koivisto
July 12, 2013

University of Helsinki
Department of Computer Science
Outline

Problem definition

Application: Bayesian network learning

Algorithms

Experiments
Outline

Problem definition

Application: Bayesian network learning

Algorithms

Experiments
Problem overview

Subset counting problem:
Compute

\[ W(Q) = \sum_{S \subseteq Q} w(S) \]
Problem setting

Given:

Ground set \( N \) of \( n \) elements

Example

\[ N = \{A, B, C, D\} \quad (n = 4) \]
Problem setting

Given:
Ground set $N$ of $n$ elements
Collection $C \in \mathcal{P}(N)$ (downward closed)

Example

$N = \{A, B, C, D\}$ (n = 4)

$C = \{\emptyset, A, B, C, D, AB, AC, AD, BC, BD, CD\}$
Problem setting

Given:

Ground set $N$ of $n$ elements

Collection $C \in \mathcal{P}(N)$ (downward closed)

Weights

\[
\begin{cases}
    w(S) \geq 0 & \text{for } S \in C \\
    w(S) = 0 & \text{otherwise}
\end{cases}
\]

Example (continued)

<table>
<thead>
<tr>
<th>$S \in C$</th>
<th>$w(S)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>80</td>
</tr>
<tr>
<td>A</td>
<td>85</td>
</tr>
<tr>
<td>B</td>
<td>70</td>
</tr>
<tr>
<td>C</td>
<td>13</td>
</tr>
<tr>
<td>D</td>
<td>50</td>
</tr>
<tr>
<td>AB</td>
<td>99</td>
</tr>
<tr>
<td>AC</td>
<td>60</td>
</tr>
<tr>
<td>AD</td>
<td>90</td>
</tr>
<tr>
<td>BC</td>
<td>11</td>
</tr>
<tr>
<td>BD</td>
<td>14</td>
</tr>
<tr>
<td>CD</td>
<td>12</td>
</tr>
</tbody>
</table>
**Problem definition**

Query set $Q \subseteq N$

---

**Example**

$$Q = \{B, C, D\}$$

<table>
<thead>
<tr>
<th>Subset</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>0.320</td>
</tr>
<tr>
<td>$B$</td>
<td>0.280</td>
</tr>
<tr>
<td>$C$</td>
<td>0.052</td>
</tr>
<tr>
<td>$D$</td>
<td>0.200</td>
</tr>
<tr>
<td>$BC$</td>
<td>0.044</td>
</tr>
<tr>
<td>$BD$</td>
<td>0.056</td>
</tr>
<tr>
<td>$CD$</td>
<td>0.048</td>
</tr>
<tr>
<td>$BCD$</td>
<td>0.000</td>
</tr>
</tbody>
</table>

$W(Q) \approx 250 \in [200, 300]$
Problem definition

Query set $Q \subseteq N$

Counting problem:
Compute

$$W(Q) = \sum_{S \subseteq Q} w(S)$$

Example

$Q = \{B, C, D\}$

<table>
<thead>
<tr>
<th>$S \subseteq Q$</th>
<th>$w(S)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>80</td>
</tr>
<tr>
<td>B</td>
<td>70</td>
</tr>
<tr>
<td>C</td>
<td>13</td>
</tr>
<tr>
<td>D</td>
<td>50</td>
</tr>
<tr>
<td>BC</td>
<td>11</td>
</tr>
<tr>
<td>BD</td>
<td>14</td>
</tr>
<tr>
<td>CD</td>
<td>12</td>
</tr>
<tr>
<td>BCD</td>
<td>0</td>
</tr>
</tbody>
</table>

$W(Q) = 250$
Problem definition

Query set $Q \subseteq N$

**Counting problem:**
Compute

$$W(Q) = \sum_{S \subseteq Q} w(S)$$

**Sampling problem:**
Draw $S \subseteq Q$ s.t.

$$\Pr(S) = \frac{w(S)}{W(Q)}$$

**Example**

$Q = \{B, C, D\}$

<table>
<thead>
<tr>
<th>$S \subseteq Q$</th>
<th>$w(S)$</th>
<th>$\Pr(S)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>80</td>
<td>0.320</td>
</tr>
<tr>
<td>B</td>
<td>70</td>
<td>0.280</td>
</tr>
<tr>
<td>C</td>
<td>13</td>
<td>0.052</td>
</tr>
<tr>
<td>D</td>
<td>50</td>
<td>0.200</td>
</tr>
<tr>
<td>BC</td>
<td>11</td>
<td>0.044</td>
</tr>
<tr>
<td>BD</td>
<td>14</td>
<td>0.056</td>
</tr>
<tr>
<td>CD</td>
<td>12</td>
<td>0.048</td>
</tr>
<tr>
<td>BCD</td>
<td>0</td>
<td>0.000</td>
</tr>
</tbody>
</table>

$W(Q) = 250$
Problem definition

Too slow ⇒ Approximate with tolerance $d \in [0, 1]$

Query set $Q \subseteq N$

Counting problem:
Compute

$$W(Q) \approx \sum_{S \subseteq Q} w(S)$$

within relative error $d$.

Sampling problem:
Draw $S \subseteq Q$ s.t.

$$\Pr(S) \approx \frac{w(S)}{W(Q)}$$

within total variation distance $d$.

Example

$Q = \{B, C, D\}$  \hspace{1cm}  $d = 0.2$

<table>
<thead>
<tr>
<th>$S \subseteq Q$</th>
<th>$w(S)$</th>
<th>$\Pr(S)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>80</td>
<td>0.320</td>
</tr>
<tr>
<td>$B$</td>
<td>70</td>
<td>0.280</td>
</tr>
<tr>
<td>$C$</td>
<td>13</td>
<td>0.052</td>
</tr>
<tr>
<td>$D$</td>
<td>50</td>
<td>0.200</td>
</tr>
<tr>
<td>$BC$</td>
<td>11</td>
<td>0.044</td>
</tr>
<tr>
<td>$BD$</td>
<td>14</td>
<td>0.056</td>
</tr>
<tr>
<td>$CD$</td>
<td>12</td>
<td>0.048</td>
</tr>
<tr>
<td>$BCD$</td>
<td>0</td>
<td>0.000</td>
</tr>
</tbody>
</table>

$W(Q) \approx 250 \in [200, 300]$
Outline

Problem definition

Application: Bayesian network learning

Algorithms

Experiments
Bayesian network

- structure: directed acyclic graph

- conditional probabilities

\[ p(x) = \prod_{v \in N} p(x_v | x_{S_v}) \]
Structure learning

**Task:** Given data, find the probability of arc $a \rightarrow b$

<table>
<thead>
<tr>
<th>sample</th>
<th>variables</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 2 3</td>
</tr>
<tr>
<td>1</td>
<td>2 1 0</td>
</tr>
<tr>
<td>2</td>
<td>2 0 2</td>
</tr>
<tr>
<td>3</td>
<td>2 0 1</td>
</tr>
<tr>
<td>4</td>
<td>1 0 2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>5000</td>
<td>2 2 1</td>
</tr>
</tbody>
</table>

Exact computation infeasible for large instances

$\Rightarrow$ approximate with MCMC sampler
Order-MCMC (Friedman & Koller, 2003)

Sample node orderings

Draws $v_1 v_2 \cdots v_n$ from the (posterior) distribution

$$\Pr(v_1 v_2 \cdots v_n) = \prod_{i=1}^{n} W_i(\{v_1, \ldots, v_{i-1}\})$$

where

- $W_i(Q) = \sum_{S_i \subseteq Q} w_i(S_i)$
- $C_i = \{ \text{potential parent sets for } v_i \}$
- $w_i$ depends on the data

$\Rightarrow n$ subset counting queries
Order-MCMC (Friedman & Koller, 2003)

Sample structures

For every node $v_i$, samples a parent set $S_i \subseteq \{v_1, \ldots, v_{i-1}\}$ such that

$$\Pr(S_i) = \frac{w_i(S_i)}{W_i(\{v_1, \ldots, v_{i-1}\})}$$

$\Rightarrow$ $n$ subset sampling queries
Outline

Problem definition

Application: Bayesian network learning

Algorithms

Experiments
Algorithm outline

Set $S$ is *relevant* if both $S \in \mathcal{C}$ and $S \subseteq Q$.

$$\Rightarrow W(Q) = \text{"sum over relevant sets"}$$

"Collector algorithm" for counting

- Visit some sets $S \subseteq N$ in some order.
- Add up weights of relevant sets.
- Stop once approximation tolerance $d$ met.
Algorithm outline

Set \( S \) is \textit{relevant} if both \( S \in \mathcal{C} \) and \( S \subseteq Q \).
\( \Rightarrow W(Q) = \text{"sum over relevant sets"} \)

"Collector algorithm” for counting

- Visit some sets \( S \subseteq N \) in some order.
- Add up weights of relevant sets.
- Stop once approximation tolerance \( d \) met.

From counting to sampling
Sample sets visited by the collector algorithm.
\( \Rightarrow \) distribution within total variation distance \( d \)
Four counting algorithms

*Exact*
An exact baseline algorithm. Visits all relevant sets.
Four counting algorithms

**Exact**
An exact baseline algorithm. Visits all relevant sets.

**Ideal**
Idealized approximation. Visits the minimum number of heaviest relevant sets.
Four counting algorithms

*Exact*
An exact baseline algorithm. Visits all relevant sets.

*Ideal*
Idealized approximation. Visits the minimum number of heaviest relevant sets.

*Sorted*
Improved version of the heuristic used by Friedman and Koller in the order-MCMC.
Four counting algorithms

**Exact**
An exact baseline algorithm. Visits all relevant sets.

**Ideal**
Idealized approximation. Visits the minimum number of heaviest relevant sets.

**Sorted**
Improved version of the heuristic used by Friedman and Koller in the order-MCMC.

**Treedy**
A novel heuristic based on tree traversal.
Preprocessing:
- Sort $\mathcal{C}$ by weight.
- Compute partial sums $W_1, W_2, \ldots, W_{|\mathcal{C}|}$.

**Example ($Q=BCD, d=0.2$)**

<table>
<thead>
<tr>
<th>$S \in \mathcal{C}$</th>
<th>$w(S)$</th>
<th>$W_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AB</td>
<td>99</td>
<td>584</td>
</tr>
<tr>
<td>AD</td>
<td>90</td>
<td>485</td>
</tr>
<tr>
<td>A</td>
<td>85</td>
<td>395</td>
</tr>
<tr>
<td>$\emptyset$</td>
<td>80</td>
<td>310</td>
</tr>
<tr>
<td>B</td>
<td>70</td>
<td>230</td>
</tr>
<tr>
<td>AC</td>
<td>60</td>
<td>160</td>
</tr>
<tr>
<td>D</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>BD</td>
<td>14</td>
<td>50</td>
</tr>
<tr>
<td>C</td>
<td>13</td>
<td>36</td>
</tr>
<tr>
<td>CD</td>
<td>12</td>
<td>23</td>
</tr>
<tr>
<td>BC</td>
<td>11</td>
<td>11</td>
</tr>
</tbody>
</table>
Preprocessing:
- Sort $\mathcal{C}$ by weight.
- Compute partial sums $W_1, W_2, \ldots, W_{|\mathcal{C}|}$.

Query:
- Traverse $\mathcal{C}$ starting from heaviest.
- Add up relevant weights to $W'$.
- Stop once the weight of the remaining sets small enough.

Example ($Q = BCD$, $d = 0.2$)

<table>
<thead>
<tr>
<th>$S \in \mathcal{C}$</th>
<th>$w(S)$</th>
<th>$W_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AB</td>
<td>99</td>
<td>584</td>
</tr>
<tr>
<td>AD</td>
<td>90</td>
<td>485</td>
</tr>
<tr>
<td>A</td>
<td>85</td>
<td>395</td>
</tr>
<tr>
<td>$\emptyset$</td>
<td>80</td>
<td>310</td>
</tr>
<tr>
<td>B</td>
<td>70</td>
<td>230</td>
</tr>
<tr>
<td>AC</td>
<td>60</td>
<td>160</td>
</tr>
<tr>
<td>D</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>BD</td>
<td>14</td>
<td>50</td>
</tr>
<tr>
<td>C</td>
<td>13</td>
<td>36</td>
</tr>
<tr>
<td>CD</td>
<td>12</td>
<td>23</td>
</tr>
<tr>
<td>BC</td>
<td>11</td>
<td>11</td>
</tr>
</tbody>
</table>

$W' = 200$
Preprocessing:

- Start with a lexicographical tree for $C$. 


Preprocessing:

Example

```
AB
w: 99
AC
w: 60
AD
w: 90
BC
w: 11
BD
w: 14
CD
w: 12

A
w: 85
B
w: 70
C
w: 13
D
w: 50
∅
w: 80
```
Preprocessing:
- Start with a lexicographical tree for $C$.
- Compute weight potentials $\phi(S)$.
Preprocessing:

Example

AB
\( w: 99 \)
\( \phi: 99 \)

AC
\( w: 60 \)
\( \phi: 60 \)

AD
\( w: 90 \)
\( \phi: 90 \)

BC
\( w: 11 \)
\( \phi: 11 \)

BD
\( w: 14 \)
\( \phi: 14 \)

CD
\( w: 12 \)
\( \phi: 12 \)

A
\( w: 85 \)
\( \phi: 334 \)

B
\( w: 70 \)
\( \phi: 95 \)

C
\( w: 13 \)
\( \phi: 25 \)

D
\( w: 50 \)
\( \phi: 50 \)

\( \emptyset \)
\( w: 80 \)
\( \phi: 584 \)
Preprocessing:

- Start with a lexicographical tree for $C$.
- Compute weight potentials $\phi(S)$.
- Sort siblings by $\phi(S)$ and reorganize links.
Preprocessing:

Example

- AB: \(w: 99\), \(\phi: 99\)
- AD: \(w: 90\), \(\phi: 90\)
- AC: \(w: 60\), \(\phi: 60\)
- BD: \(w: 14\), \(\phi: 14\)
- BC: \(w: 11\), \(\phi: 11\)
- CD: \(w: 12\), \(\phi: 12\)
- A: \(w: 85\), \(\phi: 334\)
- B: \(w: 70\), \(\phi: 95\)
- D: \(w: 50\), \(\phi: 50\)
- C: \(w: 13\), \(\phi: 25\)
- \(\emptyset\): \(w: 80\), \(\phi: 584\)
Preprocessing:

Example

AB
w: 99
φ: 99
AD
w: 90
φ: 90
AC
w: 60
φ: 60
BD
w: 14
φ: 14
BC
w: 11
φ: 11
CD
w: 12
φ: 12
A
w: 85
φ: 334
B
w: 70
φ: 95
D
w: 50
φ: 50
C
w: 13
φ: 25
∅
w: 80
φ: 584
Preprocessing:

- Start with a lexicographical tree for $C$.
- Compute weight potentials $\phi(S)$.
- Sort siblings by $\phi(S)$ and reorganize links.
- Compute aggregate potentials $\psi(S)$.
Preprocessing:

Example

AB: w: 99, φ: 99, ψ: 249
AD: w: 90, φ: 90, ψ: 150
AC: w: 60, φ: 60, ψ: 60
BD: w: 14, φ: 14, ψ: 25
BC: w: 11, φ: 11, ψ: 11
CD: w: 12, φ: 12, ψ: 12

A: w: 85, φ: 334, ψ: 504
B: w: 70, φ: 95, ψ: 170
C: w: 13, φ: 25, ψ: 25
∅: w: 80, φ: 584, ψ: 584
Preprocessing:
■ Start with a lexicographical tree for $\mathcal{C}$.
■ Compute weight potentials $\phi(S)$.
■ Sort siblings by $\phi(S)$ and reorganize links.
■ Compute aggregate potentials $\psi(S)$.

Query:
■ Traverse the tree greedily w.r.t. $\psi$. 
Preprocessing:

- Start with a lexicographical tree for $C$.
- Compute weight potentials $\phi(S)$.
- Sort siblings by $\phi(S)$ and reorganize links.
- Compute aggregate potentials $\psi(S)$.

Query:

- Traverse the tree greedily w.r.t. $\psi$.
- Add up relevant weights to $W'$. 
Preprocessing:
- Start with a lexicographical tree for $C$.
- Compute weight potentials $\phi(S)$.
- Sort siblings by $\phi(S)$ and reorganize links.
- Compute aggregate potentials $\psi(S)$.

Query:
- Traverse the tree greedily w.r.t. $\psi$.
- Add up relevant weights to $W'$.
- Ignore and skip irrelevant branches.
Preprocessing:
- Start with a lexicographical tree for $C$.
- Compute weight potentials $\phi(S)$.
- Sort siblings by $\phi(S)$ and reorganize links.
- Compute aggregate potentials $\psi(S)$.

Query:
- Traverse the tree greedily w.r.t. $\psi$.
- Add up relevant weights to $W'$. 
- Ignore and skip irrelevant branches.
- Stop once the weight of the remaining branches small enough.
Query:

Example ($Q=BCD$, $d=0.2$)

```
<table>
<thead>
<tr>
<th>Node</th>
<th>w</th>
<th>φ</th>
<th>ψ</th>
</tr>
</thead>
<tbody>
<tr>
<td>AB</td>
<td>99</td>
<td>99</td>
<td>249</td>
</tr>
<tr>
<td>AD</td>
<td>90</td>
<td>90</td>
<td>150</td>
</tr>
<tr>
<td>AC</td>
<td>60</td>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td>BD</td>
<td>14</td>
<td>14</td>
<td>25</td>
</tr>
<tr>
<td>BC</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>CD</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>A</td>
<td>85</td>
<td>334</td>
<td>504</td>
</tr>
<tr>
<td>B</td>
<td>70</td>
<td>95</td>
<td>170</td>
</tr>
<tr>
<td>D</td>
<td>50</td>
<td>50</td>
<td>75</td>
</tr>
<tr>
<td>C</td>
<td>13</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>∅</td>
<td>80</td>
<td>584</td>
<td>584</td>
</tr>
</tbody>
</table>
```
Query:

Example ($Q=BCD, d=0.2$)
Query:

Example \((Q=BCD, d=0.2)\)
Treedy

Query:

Example \((Q = BCD, d = 0.2)\)
Query:

Example ($Q = BCD, d = 0.2$)
### Example summary

Example \((Q = BCD, d = 0.2)\)

<table>
<thead>
<tr>
<th>(S \in C)</th>
<th>(w(S))</th>
<th>Exact</th>
<th>Ideal</th>
<th>Sorted</th>
<th>Treedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>AB</td>
<td>99</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AD</td>
<td>90</td>
<td>2</td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>85</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>(\emptyset)</td>
<td>80</td>
<td>1</td>
<td>(\checkmark)</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>70</td>
<td>3</td>
<td>(\checkmark)</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>AC</td>
<td>60</td>
<td></td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>50</td>
<td>8</td>
<td>(\checkmark)</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>BD</td>
<td>14</td>
<td>5</td>
<td></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>13</td>
<td>6</td>
<td></td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>12</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BC</td>
<td>11</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Outline

Problem definition

Application: Bayesian network learning

Algorithms

Experiments
Experiment setting

Data

Artificial instances

- Different types of random weights: "flat", "steep", "mixture", "shuffled"

Bayesian network learning

- Data sampled from Bayesian networks: ALARM, HAILFINDER
- Datasets from UCI repository: Votes2, Chess, 10xPromoters, Splice
Experiment setting
Counting queries

Artificial instances
- Random query sets of different sizes

Bayesian network learning
- Run order-MCMC using different counting algorithms

Limit the size of sets in $\mathcal{C}$ to be at most $k \in \{3, 4, 5\}$

$\Rightarrow$ measure runtime as a function of tolerance $d \in [10^{-8}, 0.5]$
Artificial instances

Runtimes for different types of artificial weight functions. ($n = 60, k = 5$)
## Bayesian network learning

<table>
<thead>
<tr>
<th>Name</th>
<th>$n$</th>
<th>#Samples</th>
<th>$k$</th>
<th>#Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alarm</td>
<td>37</td>
<td>50–5000</td>
<td>5</td>
<td>1000</td>
</tr>
<tr>
<td>Hailfinder</td>
<td>56</td>
<td>50–5000</td>
<td>4</td>
<td>1000</td>
</tr>
<tr>
<td>Votes2</td>
<td>34</td>
<td>435</td>
<td>5</td>
<td>10000</td>
</tr>
<tr>
<td>Chess</td>
<td>37</td>
<td>3196</td>
<td>5</td>
<td>5000</td>
</tr>
<tr>
<td>10xPromoters</td>
<td>58</td>
<td>1060</td>
<td>4</td>
<td>2000</td>
</tr>
<tr>
<td>Splice</td>
<td>61</td>
<td>3190</td>
<td>4</td>
<td>2000</td>
</tr>
</tbody>
</table>
Bayesian network learning

Runtime of order-MCMC for data generated from ALARM-network. \((n = 37, k = 5)\)
Bayesian network learning

Runtime of order-MCMC for data generated from HAILFINDER-network. ($n = 56$, $k = 4$)
Bayesian network learning

Runtime of order-MCMC for UCI datasets.
Conclusion

- Problem: approximate subset counting and sampling queries
- Application: Bayesian network structure learning
- Two heuristic methods: Sorted and Treedy
- Significant speed-ups over the Exact algorithm
- However, still room for improvements (compared to the Ideal algorithm)

Implementation (C++): cs.helsinki.fi/u/tzniinim/uai2013
Runtime as a function of query set size.