Lifted Junction Tree Algorithm
Counting and Conjunctive Queries

Tanya Braun, Ralf Möller

Institute of Information Systems
University of Lübeck

August 21, 2017
Problem: Propositional Models Explode
Problem: Propositional Models Explode
Approach: Parameterization and Lifted Inference for Query Answering (QA)

Propositional
→ 17 random variables
→ 13 factors

Lifted
→ 6 (parameterized) random variables
→ 4 domains $\mathcal{D}$ of logical variables
→ 3 parametric factors
Next Problem: Make Query Answering More Practical

Workshop Contribution

Extensions to
Lifted Junction Tree Algorithm (LJT)
[Braun & Möller, 2016]

- Fewer groundings by exploiting technique of counting
- From multiple atomic queries to multiple conjunctive queries
Lifted Variable Elimination (LVE) – Single Queries
Poole (2003), de Salvo Braz (2007), Milch et al. (2008), Taghipour (2013)
Motivation  Query Answering  Extended LJT  Analysis

Lifted Variable Elimination (LVE) – Single Queries
Poole (2003), de Salvo Braz (2007), Milch et al. (2008), Taghipour (2013)

\[ f_3^2 = \phi(Hot, AttCnf.eve, Pub.eve.p_1) \]

<table>
<thead>
<tr>
<th>Hot</th>
<th>AttCnf.eve</th>
<th>Pub.eve.p_1</th>
<th>( \phi )</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>0.4</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>0.2</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Overall, 8 cases

\[ \sum_{v \in \text{range}(Pub.eve.p_1)} \phi(Hot, AttCnf.eve, v) \]
LVE: Avoiding Grounding by Exponentiation

\[ g_3 = \phi(\text{Hot}, \text{AttCnf}(X), \text{Pub}(X, P)) \]

<table>
<thead>
<tr>
<th>Hot</th>
<th>AttCnf(X)</th>
<th>Pub(X, P)</th>
<th>( \phi )</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>0.4</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>0.2</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>

Overall, 8 entries

\[
\sum_{\text{Pub}(X, P)} g_3 := \left( \sum_{v \in \text{range}(\text{Pub}(X, P))} \phi(\text{Hot}, \text{AttCnf}(X), v) \right)^{|D(P)|}
\]

\( \text{Pub}(X, P) \) contains all logical variables in \( g_3 \).
LVE: Avoiding Grounding by Counting
LVE: Avoiding Grounding by Counting

\[
\sum_{v \in \text{range}(\text{App}.a_1)} \phi(\text{Hot}, v, \text{Biz}.m_1, \text{Biz}.m_2) \rightarrow 16 \text{ cases}
\]

\[
= \sum_{v \in \text{range}(\text{App}.a_1)} \phi(\text{Hot}, v, \#_M[\text{Biz}(M)])
\]
LVE: Count-conversion

\[ g_1 = \phi(Hot, App(A), Biz(M)) \]

<table>
<thead>
<tr>
<th>Hot</th>
<th>App(A)</th>
<th>Biz(M)</th>
<th>( \phi )</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>0.4</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Overall, 8 cases

\[ g'_1 = \phi(Hot, App(A), \#_M[Biz(M)]) \]

<table>
<thead>
<tr>
<th>Hot</th>
<th>App(A)</th>
<th>#</th>
<th>( \phi )</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>[2, 0]</td>
<td>0.4^2</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>[1, 1]</td>
<td>0.4 \cdot 0.2</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>[0, 2]</td>
<td>0.2^2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Overall, 12 cases (instead of 16)
LVE: Count-conversion

\[ g_1 = \phi(\text{Hot}, \text{App}(A), \text{Biz}(M)) \]

<table>
<thead>
<tr>
<th>Hot</th>
<th>App(A)</th>
<th>Biz(M)</th>
<th>(\phi)</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>0.4</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Overall, 8 cases

\[ g'_1 = \phi(\text{Hot}, \text{App}(A), \#_M[\text{Biz}(M)]) \]

<table>
<thead>
<tr>
<th>Hot</th>
<th>App(A)</th>
<th>#</th>
<th>(\phi)</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>[2,0]</td>
<td>0.4^2</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>[1,1]</td>
<td>0.4 \cdot 0.2</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>[0,2]</td>
<td>0.2^2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Overall, 12 cases (instead of 16)

\[
\sum_{\text{App}(A)} g_1 := \left( \sum_{\nu \in \text{range}(\text{App}(A))} \phi(\text{Hot}, \nu, \#_M[\text{Biz}(M)]) \right)^{|\mathcal{D}(A)|}
\]
First-order Decomposition Trees (FO Dtrees)
Darwiche (2001), Taghipour (2013)
First-order Junction Trees (FO Jtrees)

FO dtree

∀a

∀x

∀m

g″

∀p

Hot, Biz(M)

Hot, Biz(M), App(a)

Hot, Biz(m), App(a)

Hot, Biz(m), App(a)

Hot, Biz(m), App(a)

Hot, Biz(M)

Hot, Biz(M), App(a)

Hot, Biz(M), App(a)

Hot, Biz(M), App(a)

Hot, AttC(x)

Hot, AttC(x)

Hot, AttC(x)

Hot, AttC(x)

Hot, AttC(x)

Hot, AttC(x), Pub(x, p)

Hot, AttC(x), Pub(x, p)

{g″}

{g2″}

{g3″}
First-order Junction Trees (FO Jtrees)

FO dtree

∀a

∀x

∀m

∀p

g_1''

g_2'

g_3''

Clusters

Hot

Hot, Biz(M)

Hot, Biz(M), App(a)

Hot, Biz(M), App(a)

Hot, Biz(m), App(a)

Hot, Biz(m), App(a)

{g_1''}

{g_2'}

Hot, AttC(x)

Hot, AttC(x)

Hot, AttC(x)

Hot, AttC(x), Pub(x, p)

Hot, AttC(x), Pub(x, p)

Hot, AttC(x), Pub(x, p)

FO jtree

Hot, App(A)

Biz(M)

{g_1}

Hot

Hot, AttCnf(X)

Res(X)

{g_2}

Hot, AttCnf(X)

Pub(X, P)

{g_3}
LJT – Multiple Queries
Lauritzen & Spiegelhalter (1988), Koller & Friedman (2009), Braun & Möller (2016)

Atomic queries on ground random variables, e.g., Res.eve, Pub.eve.p1

Input
- Model G
- Queries Q

Algorithm
1. Build FO jtreet for G.
2. Pass messages
   - inbound
   - outbound
3. Answer queries Q.
   (without evidence)
Extended LJT: Example

Input + Construction

**Model**  \( G = \{g_1, g_2, g_3\} \),
- \( g_1 = \phi_1(Hot, App(A), Biz(M)) \)
- \( g_2 = \phi_2(Hot, AttCnf(X), Res(X)) \)
- \( g_3 = \phi_3(Hot, AttCnf(X), Pub(X, P)) \)

**Queries**  \( Q = \{Q_1, Q_2\} \)
- \( Q_1 = P(Pub(eve, p_1)) \) (atomic query)
- \( Q_2 = P(Pub(eve, p_1), Res(eve)) \) (conjunctive query)
Extended LJT: Example

Input + Construction

Model $G = \{ g_1, g_2, g_3 \},$

- $g_1 = \phi_1(Hot, App(A), Biz(M))$
- $g_2 = \phi_2(Hot, AttCnf(X), Res(X))$
- $g_3 = \phi_3(Hot, AttCnf(X), Pub(X, P))$

Queries $Q = \{ Q_1, Q_2 \}$

- $Q_1 = P(Pub(eve, p_1))$ (atomic query)
- $Q_2 = P(Pub(eve, p_1), Res(eve))$ (conjunctive query)

Constructed FO jtree
Extended LJT: Example Continued

Message Passing

Node 1

Hot App(A)
Biz(M)

{g_1}

Node 2

Hot AttCnf(X)
Res(X)

{g_2}

Node 3

Hot AttCnf(X)
Pub(X, P)

{g_3}

m_{12} = \sum_{M[Biz(M)]} \sum_{App(A)} g_1

m_{32} = \sum_{Pub(X, P)} g_3
Extended LJT: Example Continued

Message Passing

Node 1
\(\text{Hot App}(A)\,\text{Biz}(M)\)
\(\{g_1\}\)

Node 2
\(\text{Hot AttCnf}(X)\,\text{Res}(X)\)
\(\{g_2\} \cup \{m_{12}, m_{32}\}\)

Node 3
\(\text{Hot AttCnf}(X)\,\text{Pub}(X, P)\)
\(\{g_3\}\)

\[m_{21} = \sum_{\text{AttCnf}(X)} m_{32} \sum_{\text{Res}(X)} g_2\]

\[m_{23} = m_{12} \sum_{\text{Res}(X)} g_2\]
Extended LJT: Example Continued

Query Answering

Node 1

Hot App(A)
Biz(M)

\{g_1\} \cup \{m_{21}\}

Node 2

Hot AttCnf(X)
Res(X)

\{g_2\} \cup \{m_{12}, m_{32}\}

Node 3

Hot AttCnf(X)
Pub(X, P)

\{g_3\} \cup \{m_{23}\}

Query

\[ Q = P(Pub(eve, p_1)) \]

Submodel

\[ G^Q = \{g_3\} \cup \{m_{23}\} \]
Extended LJT: Example Continued

Query Answering

Node 1

\[
\text{Hot App(A) Biz(M)}
\]

\[
\{g_1\} \cup \{m_{21}\}
\]

Node 2

\[
\text{Hot AttCnf}(X) \text{ Res}(X)
\]

\[
\{g_2\} \cup \{m_{12}, m_{32}\}
\]

Node 3

\[
\text{Hot AttCnf}(X) \text{ Pub}(X, P)
\]

\[
\{g_3\} \cup \{m_{23}\}
\]

Query

\[
Q = P(\text{Pub(eve, } p_1))
\]

Submodel

\[
G^Q = \{g_3\} \cup \{m_{23}\}
\]

\[
Q \propto \sum_{\text{Hot AttCnf}(X)} \sum m_{23} \sum_{\text{Pub}(X, P), \ X \neq \text{eve, } \ P \neq p_1} g_3
\]
Extended LJT: Example Continued

Query Answering

**Node 1**
- **Hot App(A)**
- **Biz(M)**

\[ \{g_1\} \cup \{m_{21}\} \]

**Node 2**
- **Hot AttCnf(X)**
- **Res(X)**

\[ \{g_2\} \cup \{m_{12}, m_{32}\} \]

**Node 3**
- **Hot AttCnf(X)**
- **Pub(X, P)**

\[ \{g_3\} \cup \{m_{23}\} \]

**Query**

\[ Q = P(Pub(\text{eve}, p_1), \text{Res(eve)}) \]

**Submodel**

\[ G^Q = \{g_2\} \cup \{g_3\} \cup \{m_{12}\} \]
Extended LJT: Example Continued

Query Answering

Node 1
\( \text{Hot App}(A) \)  
\( \text{Biz}(M) \)
\( \{g_1\} \cup \{m_{21}\} \)

Node 2
\( \text{Hot AttCnf}(X) \)
\( \text{Res}(X) \)
\( \{g_2\} \cup \{m_{12}, m_{32}\} \)

Node 3
\( \text{Hot AttCnf}(X) \)
\( \text{Pub}(X, P) \)
\( \{g_3\} \cup \{m_{23}\} \)

Query
\[ Q = P(\text{Pub}(\text{eve}, p_1), \text{Res}(\text{eve})) \]

Submodel
\[ G^Q = \{g_2\} \cup \{g_3\} \cup \{m_{12}\} \]

\[ Q \propto \sum_{\text{Hot}} m_{12} \sum_{\text{AttCnf}(X)} \sum_{\text{Res}(X), \ X \neq \text{eve}} g_2 \sum_{\text{Pub}(X, P), \ X \neq \text{eve}, \ P \neq p_1} g_3 \]
Analysis: LVE vs. LJT

**LVE**
- Eliminates complete model for each query
- Influence on QA: number of PRVs in model

**LJT**
- Finds subtree, eliminates smaller model in subtree
- Influence on QA: number of PRVs in submodel

PRV = parameterized random variable
## Analysis: LVE vs. LJT

<table>
<thead>
<tr>
<th>LVE</th>
<th>LJT</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Eliminates complete model for each query</td>
<td>• Finds subtree, eliminates smaller model in subtree</td>
</tr>
<tr>
<td>• Influence on QA: number of PRVs in model</td>
<td>• Influence on QA: number of PRVs in submodel</td>
</tr>
</tbody>
</table>

LJT preprocessing necessary → Static overhead

• New construction if structure changes
• New message passing if domain sizes or values change

PRV = parameterized random variable
Test Run: LVE vs. LJT

**Figure**: Accumulated Runtimes [ms] over 6 queries (points connected for readability)

- **gcfove**: Implementation of LVE by Taghipour (2013)
- **exfojt**: Own implementation of extended LJT
Problem: Make Query Answering More Practical

Workshop Contribution

Extensions to Lifted Junction Tree Algorithm (LJT)
[Braun & Möller, 2016]

• Fewer groundings by exploiting technique of counting
• From multiple atomic queries to multiple conjunctive queries
Problem: Make Query Answering More Practical

Workshop Contribution

Extensions to Lifted Junction Tree Algorithm (LJT)
[Braun & Möller, 2016]

• Fewer groundings by exploiting technique of counting
• From multiple atomic queries to multiple conjunctive queries

Future Directions
• Incrementally changing models
• Dynamic variant
Cutset, Context, Cluster
Darwiche (2001), Taghipour (2013)

\[
\text{cutset}(T) = \bigcup_{T_i, T_j \in \text{child}(T)} \text{RV}(T_i) \cap \text{RV}(T_j) \setminus \text{cutset}(T)
\]

\[
\text{acutset}(T) = \bigcup_{T' \in \text{ancestor}(T)} \text{cutset}(T')
\]

\[
\text{context}(T) = \text{RV}(T) \cap \text{acutset}(T)
\]

\[
\text{cluster}(T) = \text{cutset}(T) \cup \text{context}(T)
\]

\[
\text{cluster}(L) = \text{RV}(\phi_L), \text{L leaf}
\]

\[
\text{RV}(T) = \bigcup_{T' \in \text{child}(T)} \text{RV}(T')
\]

\[
\text{RV}(L) = \text{RV}(\phi_L), \text{L leaf}
\]
Cutset, Context, Cluster
Darwiche (2001), Taghipour (2013)

Labels: cutset(N) ∪ context(N)
Cutset, Context, Cluster

Darwiche (2001), Taghipour (2013), Labels: cutset($N$) $\cup$ context($N$)

\[
\text{cutset}(T) = \bigcup_{T_i, T_j \in \text{child}(T)} RV(T_i) \cap RV(T_j) \setminus \text{cutset}(T), \quad \text{acutset}(T) = \bigcup_{T' \in \text{ancestor}(T)} \text{cutset}(T')
\]

\[
\text{context}(T) = RV(T) \cap \text{acutset}(T), \quad RV(T) = \bigcup_{T' \in \text{child}(T)} RV(T'), \quad RV(L) = RV(\phi_L), \quad L \text{ leaf}
\]

\[
\text{cluster}(T) = \text{cutset}(T) \cup \text{context}(T), \quad \text{cluster}(L) = RV(\phi_L), \quad L \text{ leaf}
\]