Lifted Junction Tree Algorithm

Preventing Groundings and Handling Evidence

Tanya Braun, Ralf Möller

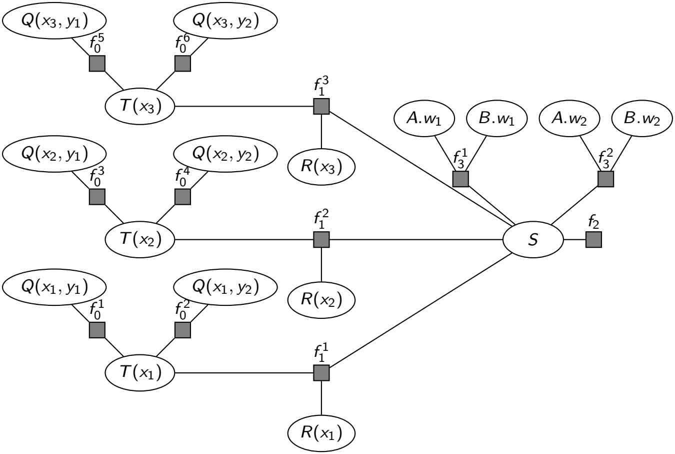
Institute of Information Systems
University of Lübeck

September 28, 2017

Problem: Practical Query Answering (QA)

Large models

Many queries



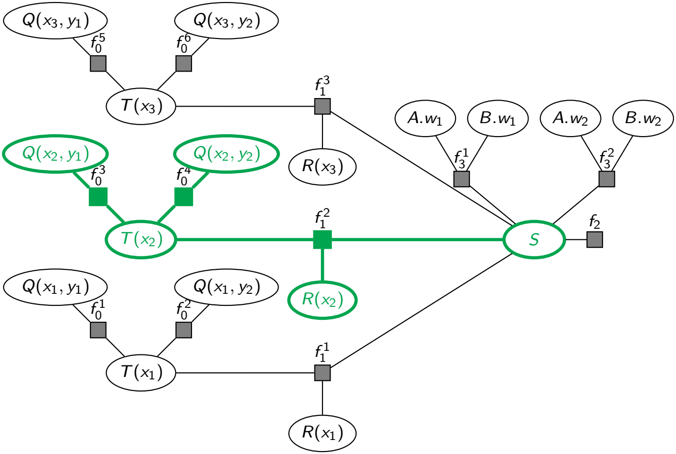
Symmetries

Clusters

Problem: Practical Query Answering (QA)

Large models

Many queries



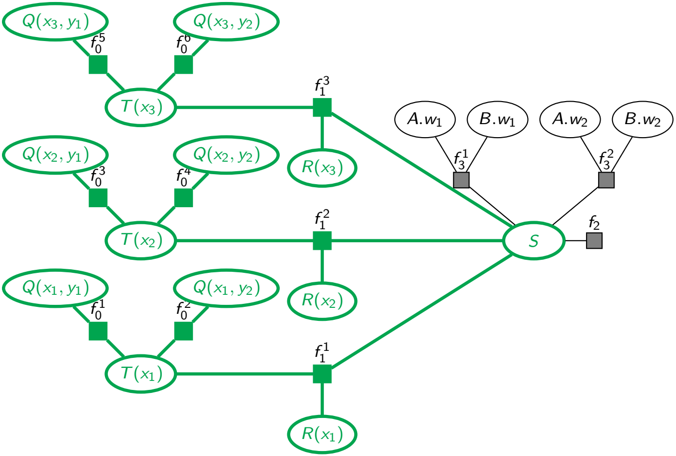
Symmetries

Clusters

Problem: Practical Query Answering (QA)

Large models

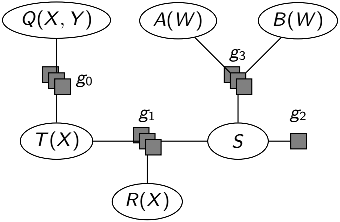
Many queries

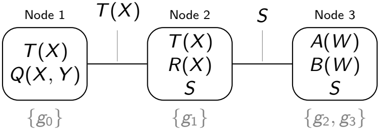


Symmetries

Clusters

Approach: Parameters and Clusters





- Parameterization
  - → Avoid explosion of nodes
- Lifted variable elimination (LVE)[1]
  - → Save computations
  - → Each query in isolation
- Multiple queries: Junction Tree[2]
  - → Cluster representation, messages
- Lifted Junction Tree Algorithm (LJT)[3]
  - → Save computations
  - → Ground marginal queries

[1] Poole (2003), de Salvo Braz (2007), Milch et al. (2008), Taghipour (2013),
[2] Lauritzen & Spiegelhalter (1988) [3] Braun & Möller (2016)

Conference Contribution

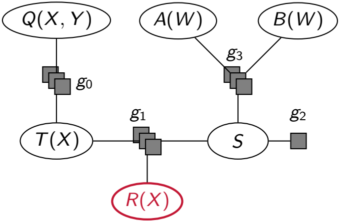
Extensions to LJT [Braun & Möller 2016]

- Expressivity extension for query language
    - Evidence-based queries
    - Same evidence for multiple queries
- Speed-up for specific inputs
    - Avoid unnecessary groundings
    - At least as good as LVE w.r.t. runtimes

→ Speed-up for all inputs with at least two clusters

Marginal Distribution Ground Queries

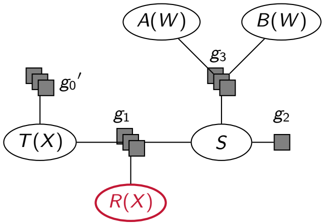
$P(R(x_1))$



QA: LVE eliminates

Marginal Distribution Ground Queries

$P(R(x_1))$

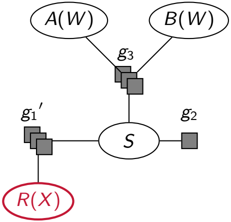


QA: LVE eliminates

- $Q(X, Y)$ from g_0

Marginal Distribution Ground Queries

$P(R(x_1))$

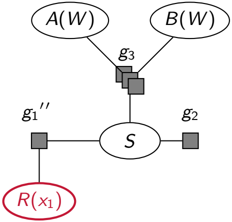


QA: LVE eliminates

- $Q(X, Y)$ from g_0
- $T(X)$ from $g_1 \cdot g_0'$

Marginal Distribution Ground Queries

$P(R(x_1))$

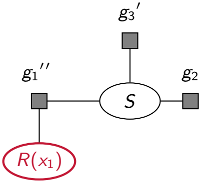


QA: LVE eliminates

- $Q(X, Y)$ from g_0
- $T(X)$ from $g_1 \cdot g_0'$
- $R(X), X \neq x_1$ from g_1'

Marginal Distribution Ground Queries

$P(R(x_1))$

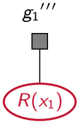


QA: LVE eliminates

- $Q(X, Y)$ from g_0
- $T(X)$ from $g_1 \cdot g'_0$
- $R(X), X \neq x_1$ from g'_1
- $A(W)$, $B(W)$

Marginal Distribution Ground Queries

$P(R(x_1))$

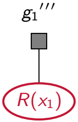


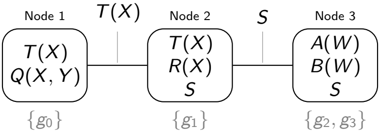
QA: LVE eliminates

- $Q(X, Y)$ from g_0
- $T(X)$ from $g_1 \cdot g_0'$
- $R(X), X \neq x_1$ from g_1'
- $A(W)$, $B(W)$, S

Marginal Distribution Ground Queries

$P(R(x_1))$





QA: LVE eliminates

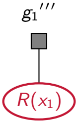
- $Q(X, Y)$ from g_0
- $T(X)$ from $g_1 \cdot g_0'$
- $R(X), X \neq x_1$ from g_1'
- $A(W)$, $B(W)$, S

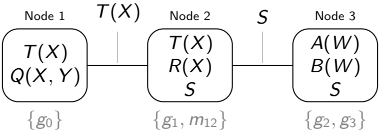
LJT passes messages (indep. of query)

- Message = query on edge variables
- $1 \longrightarrow 2$: $P(T(X))$

Marginal Distribution Ground Queries

$P(R(x_1))$





QA: LVE eliminates

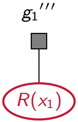
- $Q(X, Y)$ from g_0
- $T(X)$ from $g_1 \cdot g_0'$
- $R(X), X \neq x_1$ from g_1'
- $A(W)$, $B(W)$, S

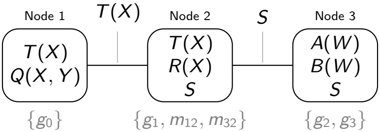
LJT passes messages (indep. of query)

- Message = query on edge variables
- $1 \longrightarrow 2$: $P(T(X))$

Marginal Distribution Ground Queries

$P(R(x_1))$





QA: LVE eliminates

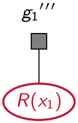
- $Q(X, Y)$ from g_0
- $T(X)$ from $g_1 \cdot g_0'$
- $R(X), X \neq x_1$ from g_1'
- $A(W)$, $B(W)$, S

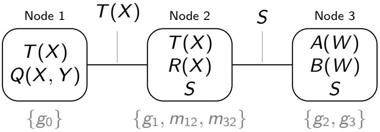
LJT passes messages (indep. of query)

- Message = query on edge variables
- $3 \longrightarrow 2$: $P(S)$

Marginal Distribution Ground Queries

$P(R(x_1))$





QA: LVE eliminates

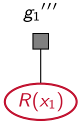
- $Q(X,Y)$ from g_0
- $T(X)$ from $g_1 \cdot g_0'$
- $R(X), X \neq x_1$ from g_1'
- $A(W)$, $B(W)$, S

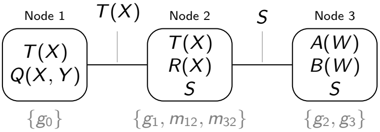
LJT passes messages (indep. of query)

- Message = query on edge variables
- $2 \longrightarrow 1$: $P(T(X))$
  - Eliminate $R(X)$
  - Ground $T(X)$
  - To eliminate S

Marginal Distribution Ground Queries

$P(R(x_1))$





QA: LVE eliminates

- $Q(X, Y)$ from g_0
- $T(X)$ from $g_1 \cdot g_0'$
- $R(X), X \neq x_1$ from g_1'
- $A(W)$, $B(W)$, S

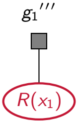
LJT passes messages (indep. of query)

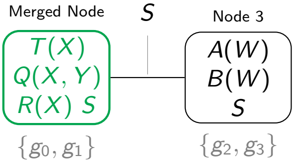
- Message = query on edge variables
- $2 \longrightarrow 1$: $P(T(X))$
    - Eliminate $R(X)$
    - Ground $T(X)$
    - To eliminate S

Merge nodes Compromise on cluster size and message complexity

Marginal Distribution Ground Queries

$P(R(x_1))$





QA: LVE eliminates

- $Q(X, Y)$ from g_0
- $T(X)$ from $g_1 \cdot g_0'$
- $R(X), X \neq x_1$ from g_1'
- $A(W)$, $B(W)$, S

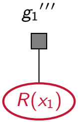
LJT passes messages (indep. of query)

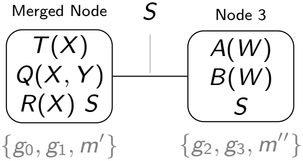
- Message = query on edge variables
- $2 \longrightarrow 1$: $P(T(X))$
  - Eliminate $R(X)$
  - Ground $T(X)$
  - To eliminate S

Merge nodes Compromise on cluster size and message complexity

Marginal Distribution Ground Queries

$P(R(x_1))$





QA: LVE eliminates

- $Q(X, Y)$ from g_0
- $T(X)$ from $g_1 \cdot g_0'$
- $R(X), X \neq x_1$ from g_1'
- $A(W)$, $B(W)$, S

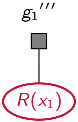
LJT passes messages (indep. of query)

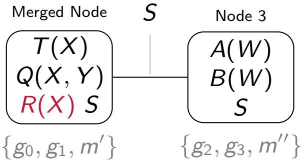
- Message = query on edge variables
- $2 \longrightarrow 1$: $P(T(X))$
  - Eliminate $R(X)$
  - Ground $T(X)$
  - To eliminate S

Merge nodes Compromise on cluster size and message complexity

Marginal Distribution Ground Queries

$P(R(x_1))$





QA: LVE eliminates

- $Q(X,Y)$ from g_0
- $T(X)$ from $g_1 \cdot g_0'$
- $R(X), X \neq x_1$ from g_1'
- $A(W)$, $B(W)$, S

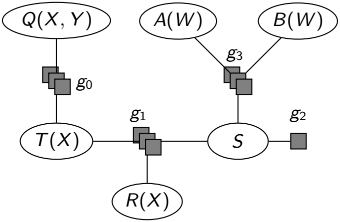
LJT passes messages (indep. of query)

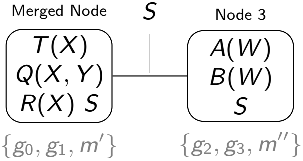
- Message $=$ query on edge variables
- $2 \longrightarrow 1$: $P(T(X))$
  - Eliminate $R(X)$
  - Ground $T(X)$
  - To eliminate S

Merge nodes Compromise on cluster size and message complexity

Marginal Distribution Ground Queries

$P(R(x_1))$, $P(T(x_2))$, $P(A(w_1))$, ...





QA: LVE eliminates

- $Q(X, Y)$ from g_0
- $T(X)$ from $g_1 \cdot g_0'$
- $R(X), X \neq x_1$ from g_1'
- $A(W)$, $B(W)$, S

LJT passes messages (indep. of query)

- Message = query on edge variables
- $2 \longrightarrow 1$: $P(T(X))$
  - Eliminate $R(X)$
  - Ground $T(X)$
  - To eliminate S

Merge nodes Compromise on cluster size and message complexity

Test Run: LVE vs. LJT

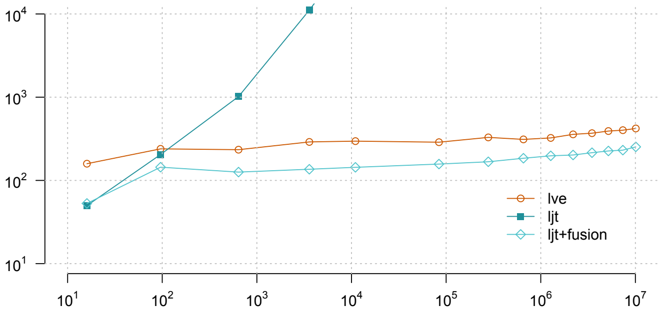


Figure: Runtimes [ms] over 6 queries with grounded model sizes from 16 to 10,100,000 (points connected for readability)

`lve`: Implementation by Taghipour (2013)
`ljt`: Our implementation of LJT
`ljt+fusion`: Our implementation of LJT with fusion

Analysis: LJT with Fusion

Algorithm steps

1. FO jtree construction
2. Fusion
3. Evidence entering
4. Message passing
5. Query answering

Effect of fusion on an FO jtree

Worst Case: Collapse into one node

$\rightarrow$ Direct LVE with LJT overhead

Best Case: No modification

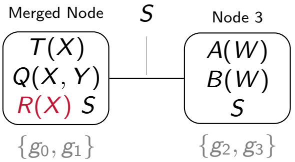
$\rightarrow$ Full potential of clusters

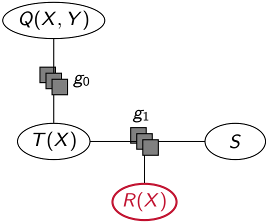
Average Case: Modification, no collapse

$\rightarrow$ Compromise on cluster size and message complexity

Conditional Distribution Ground Queries

$P(R(x_1)|R(x_2) = true, R(x_3) = true)$



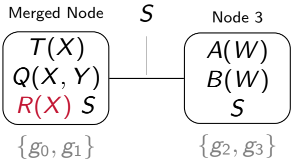


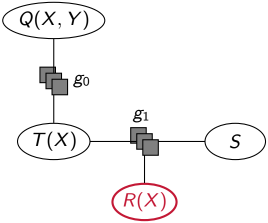
Evidence: LJT

- Evidence at node

Conditional Distribution Ground Queries

$P(R(x_1)|R(x_2) = true, R(x_3) = true)$





Evidence: LJT

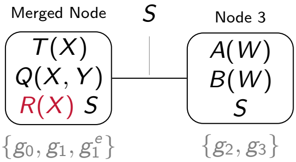
- Evidence at node

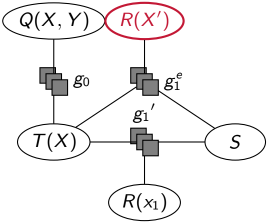
Evidence at node: use LVE

- Each factor containing $R(X)$

Conditional Distribution Ground Queries

$P(R(x_1)|R(x_2) = true, R(x_3) = true)$





Evidence: LJT

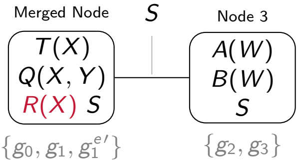
- Evidence at node

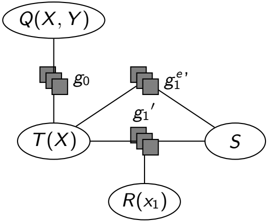
Evidence at node: use LVE

- Each factor containing $R(X)$
- Splits $R(X)$ w.r.t. $X \in \{x_2, x_3\}$

Conditional Distribution Ground Queries

$P(R(x_1)|R(x_2) = true, R(x_3) = true)$





Evidence: LJT

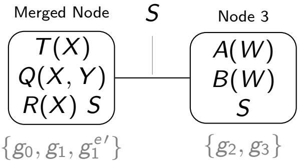
- Evidence at node

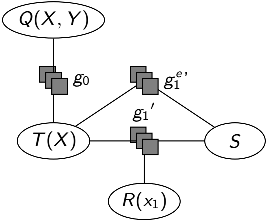
Evidence at node: use LVE

- Each factor containing $R(X)$
- Splits $R(X)$ w.r.t. $X \in \{x_2, x_3\}$
- Absorbs $R(X) = true, X \in \{x_2, x_3\}$

Conditional Distribution Ground Queries

$P(R(x_1)|R(x_2) = true, R(x_3) = true)$





Evidence: LJT

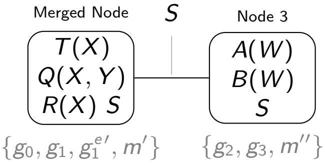
- Evidence at node
- Continue with LJT

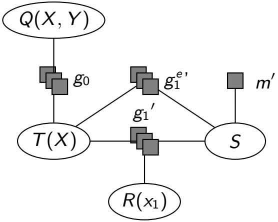
Evidence at node: use LVE

- Each factor containing $R(X)$
- Splits $R(X)$ w.r.t. $X \in \{x_2, x_3\}$
- Absorbs $R(X) = true, X \in \{x_2, x_3\}$

Conditional Distribution Ground Queries

$P(R(x_1)|R(x_2) = true, R(x_3) = true)$





Evidence: LJT

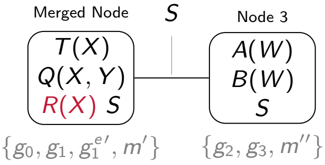
- Evidence at node
- Continue with LJT
  - $\rightarrow$ Pass messages

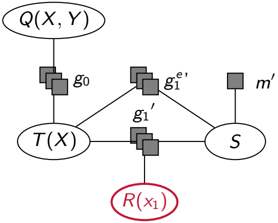
Evidence at node: use LVE

- Each factor containing $R(X)$
- Splits $R(X)$ w.r.t. $X \in \{x_2, x_3\}$
- Absorbs $R(X) = true, X \in \{x_2, x_3\}$

Conditional Distribution Ground Queries

$P(R(x_1)|R(x_2) = true, R(x_3) = true)$





Evidence: LJT

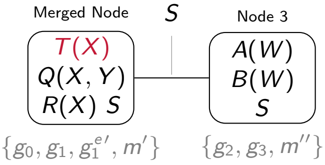
- Evidence at node
- Continue with LJT
  - → Pass messages
  - → Answer queries $P(R(x_1))$

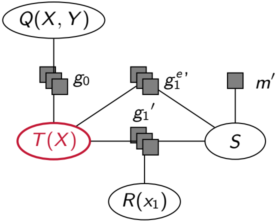
Evidence at node: use LVE

- Each factor containing $R(X)$
- Splits $R(X)$ w.r.t. $X \in \{x_2, x_3\}$
- Absorbs $R(X) = true, X \in \{x_2, x_3\}$

Conditional Distribution Ground Queries

$P(R(x_1)|R(x_2) = true, R(x_3) = true)$, $P(T(x_2)|R(x_2) = true, R(x_3) = true), \ldots$





Evidence: LJT

- Evidence at node
- Continue with LJT
  - → Pass messages
  - → Answer queries
    $P(R(x_1))$
    $P(T(x_2))$
    ...

Evidence at node: use LVE

- Each factor containing $R(X)$
- Splits $R(X)$ w.r.t. $X \in \{x_2, x_3\}$
- Absorbs $R(X) = true, X \in \{x_2, x_3\}$

Test Run: LVE vs. LJT – Evidence

Curve Shapes and Difference between Curves Independent of Grounded Model Size

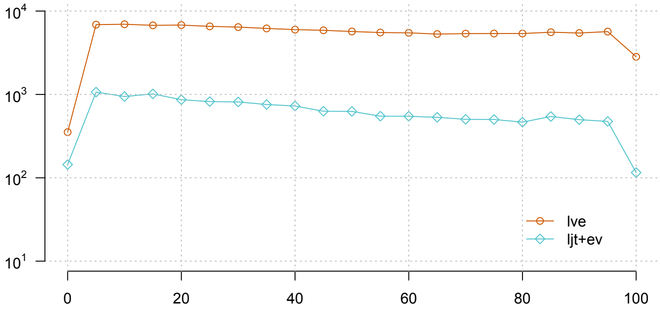


Figure: Runtimes [ms] over 6 queries with evidence ranging from 0% to 100% (points connected for readability)

`lve`: Implementation by Taghipour (2013)
`ljt+ev`: Our implementation of LJT with evidence handling

Analysis: LJT with Evidence

Algorithm steps

1. FO jtree construction
2. Fusion
3. Evidence entering
4. Message passing
5. Query answering

Effect of evidence entering

One pass over FO jtree

$\rightarrow$ Continue with LJT as before

Same evidence for all queries

$\rightarrow$ QA as before

New evidence

$\rightarrow$ Restart at step 3

Conference Contribution

Extensions to LJT [Braun & Möller 2016]

- Expressivity extension for query language
  - Evidence-based queries
  - Same evidence for multiple queries
- Speed-up for specific inputs
  - Avoid unnecessary groundings
  - At least as good as LVE w.r.t. runtimes

$\rightarrow$ Speed-up for all inputs with at least two clusters

Conference Contribution

Extensions to LJT [Braun & Möller 2016]

- Expressivity extension for query language
  - Evidence-based queries
  - Same evidence for multiple queries
- Speed-up for specific inputs
  - Avoid unnecessary groundings
  - At least as good as LVE w.r.t. runtimes

→ Speed-up for all inputs with at least two clusters

Future Directions

- Incrementally changing models
- Dynamic variant