Arithmetic Circuits of the Noisy-Or Models

Jiří Vomlel and Petr Savický

Academy of Sciences of the Czech Republic

Loučeň, December 2, 2008

BN2O model

- It is a Bayesian network (BN).

BN2O model

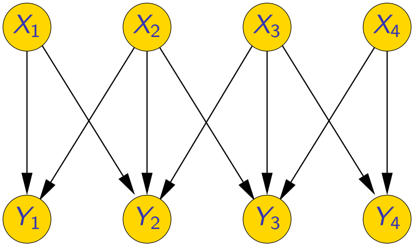
- It is a Bayesian network (BN).
- It has two sets of variables: $\{X_j, j = 1, \ldots, k\}$ and $\{Y_i, i = 1, \ldots, \ell\}$.

BN2O model

- It is a Bayesian network (BN).
- It has two sets of variables: $\{X_j, j = 1, \ldots, k\}$ and $\{Y_i, i = 1, \ldots, \ell\}$.
- Variables are Boolean:
  X_j takes values $x_j \in \{0, 1\}$ and Y_i takes values $y_i \in \{0, 1\}$.

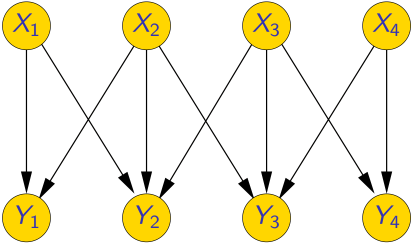
BN2O model

- It is a Bayesian network (BN).
- It has two sets of variables: $\{X_j, j = 1, \ldots, k\}$ and $\{Y_i, i = 1, \ldots, \ell\}$.
- Variables are Boolean:
  X_j takes values $x_j \in \{0, 1\}$ and Y_i takes values $y_i \in \{0, 1\}$.
- The depedendency structure is given by a bipartite graph:



BN2O model

- It is a Bayesian network (BN).
- It has two sets of variables: $\{X_j, j = 1, \ldots, k\}$ and $\{Y_i, i = 1, \ldots, \ell\}$.
- Variables are Boolean:
  X_j takes values $x_j \in \{0, 1\}$ and Y_i takes values $y_i \in \{0, 1\}$.
- The depedendency structure is given by a bipartite graph:

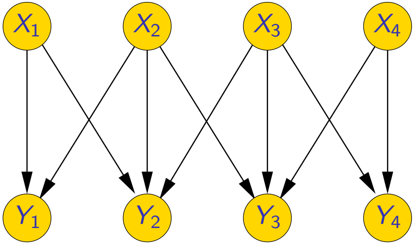


Example

The nodes from the first level represent diseases and the nodes from the second level their symptoms.

BN2O model

- It is a Bayesian network (BN).
- It has two sets of variables: $\{X_j, j = 1, \ldots, k\}$ and $\{Y_i, i = 1, \ldots, \ell\}$.
- Variables are Boolean:
  X_j takes values $x_j \in \{0, 1\}$ and Y_i takes values $y_i \in \{0, 1\}$.
- The depedendency structure is given by a bipartite graph:



Example

The nodes from the first level represent system faults and the nodes from the second level the observations.

BN2O model - conditional probability tables

- The conditional probability tables (CPT) $P(Y|X_1, \ldots, X_n)$ are noisy-or gates.

BN2O model - conditional probability tables

- The conditional probability tables (CPT) $P(Y|X_1, \ldots, X_n)$ are noisy-or gates.
- Each CPT is defined for $(x_1, \ldots, x_n) \in \{0, 1\}^n$ by

$$P(Y = 0|X_1 = x_1, \ldots, X_n = x_n) \ = \ \prod_{j=1}^{n} (p_j)^{x_j} \ ,$$

BN2O model - conditional probability tables

- The conditional probability tables (CPT) $P(Y|X_1, \ldots, X_n)$ are noisy-or gates.
- Each CPT is defined for $(x_1, \ldots, x_n) \in \{0, 1\}^n$ by

$$P(Y = 0 | X_1 = x_1, \ldots, X_n = x_n) \quad = \quad \prod_{j=1}^{n} (p_j)^{x_j} \quad ,$$

- where $0^0 = 1$ (and $0^1 = 0$).

BN2O model - conditional probability tables

- The conditional probability tables (CPT) $P(Y|X_1, \ldots, X_n)$ are noisy-or gates.
- Each CPT is defined for $(x_1, \ldots, x_n) \in \{0,1\}^n$ by

$$P(Y = 0|X_1 = x_1, \ldots, X_n = x_n) \;=\; \prod_{j=1}^{n} (p_j)^{x_j} \;,$$

- where $0^0 = 1$ (and $0^1 = 0$).
- p_j is called the inhibition probability of $X_j = 1$ since $Y = 0$ only if all its parents with value 1 are inhibited.

Example - deterministic or

CPT $P(Y|X_1, \ldots, X_n)$ represents an or gate if $p_j = 0$ for all $j \in \{1, \ldots, n\}$.

Example - deterministic or

CPT $P(Y|X_1, \ldots, X_n)$ represents an or gate if $p_j = 0$ for all $j \in \{1, \ldots, n\}$.

$$P(Y = 0|X_1 = x_1, \ldots, X_n = x_n) \quad = \quad \prod_{j=1}^{n} 0^{x_j}$$

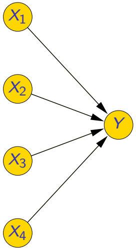
Example - deterministic or

CPT $P(Y|X_1, \ldots, X_n)$ represents an or gate if $p_j = 0$ for all $j \in \{1, \ldots, n\}$.

$$
\begin{aligned}
P(Y = 0 | X_1 = x_1, \ldots, X_n = x_n) &= \prod_{j=1}^{n} 0^{x_j} \\
&= \begin{cases} 1 & \text{if } x_j = 0 \text{ for } j \in \{1, \ldots, n\} \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}
$$

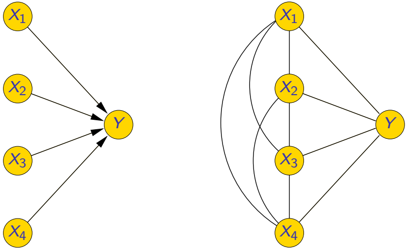
Compilation of a noisy-or gate - the standard BN approach

Lauritzen and Spiegelhalter (1988), Jensen et al. (1990), Shafer and Shenoy (1990)



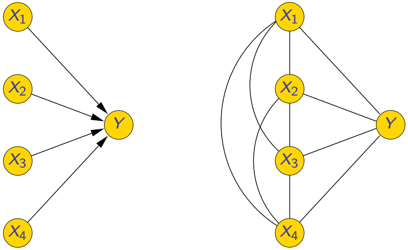
Compilation of a noisy-or gate - the standard BN approach

Lauritzen and Spiegelhalter (1988), Jensen et al. (1990), Shafer and Shenoy (1990)



Compilation of a noisy-or gate - the standard BN approach

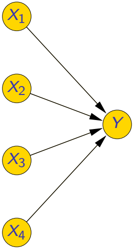
Lauritzen and Spiegelhalter (1988), Jensen et al. (1990), Shafer and Shenoy (1990)



The total table size is $2^5 = 32$.

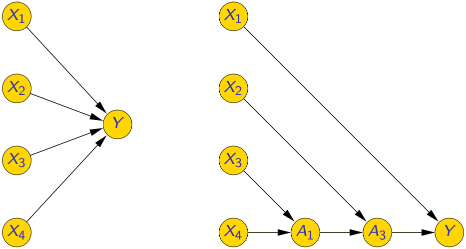
Compilation of a noisy-or gate - parent divorcing

Olesen et al. (1989)



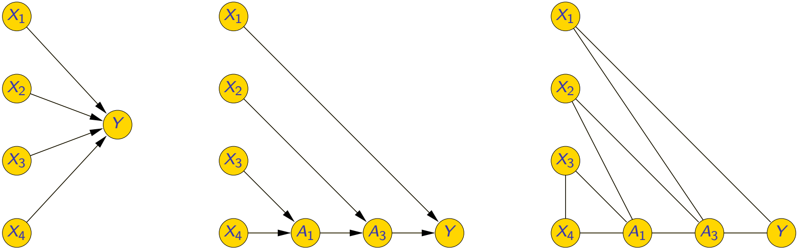
Compilation of a noisy-or gate - parent divorcing

Olesen et al. (1989)



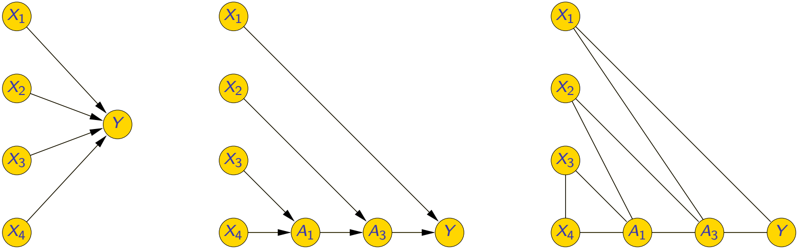
Compilation of a noisy-or gate - parent divorcing

Olesen et al. (1989)



Compilation of a noisy-or gate - parent divorcing

Olesen et al. (1989)



The total table size is $3 \cdot 2^3 = 24$.

Rank-one decomposition

Díez and Galán (2003), Vomlel (2002), Savický and Vomlel (2007)

Recall, the noisy-or definition.
For $(x_1, \ldots, x_n) \in \{0,1\}^n$:

$$P(Y = 0 | X_1 = x_1, \ldots, X_n = x_n) \;=\; \prod_{j=1}^{n} (p_j)^{x_j} \;,$$

Rank-one decomposition

Díez and Galán (2003), Vomlel (2002), Savický and Vomlel (2007)

Recall, the noisy-or definition.
For $(x_1, \ldots, x_n) \in \{0,1\}^n$:

$$P(Y = 0 | X_1 = x_1, \ldots, X_n = x_n) \;=\; \prod_{j=1}^{n} (p_j)^{x_j} \;,$$

For $y \in \{0,1\}$ and $(x_1, \ldots, x_n) \in \{0,1\}^n$ we can write:

$$P(Y = y | X_1 = x_1, \ldots, X_n = x_n) \;=\; (1-2y) \prod_{j=1}^{n} (p_j)^{x_j} + y \prod_{i=1}^{n} 1$$

Rank-one decomposition

Díez and Galán (2003), Vomlel (2002), Savický and Vomlel (2007)

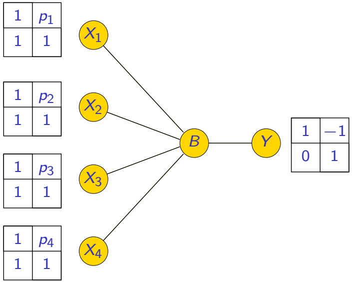
Recall, the noisy-or definition.
For $(x_1, \ldots, x_n) \in \{0,1\}^n$:

$$P(Y = 0 | X_1 = x_1, \ldots, X_n = x_n) \;=\; \prod_{j=1}^{n} (p_j)^{x_j} \;,$$

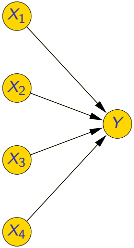
For $y \in \{0,1\}$ and $(x_1, \ldots, x_n) \in \{0,1\}^n$ we can write:

$$\begin{aligned} P(Y = y | X_1 = x_1, \ldots, X_n = x_n) &= (1-2y)\prod_{j=1}^{n} (p_j)^{x_j} + y \prod_{i=1}^{n} 1 \\ &= \sum_{b=0}^{1} \xi(b,y) \cdot \prod_{j=1}^{n} \varphi_j(b, x_j) \end{aligned}$$

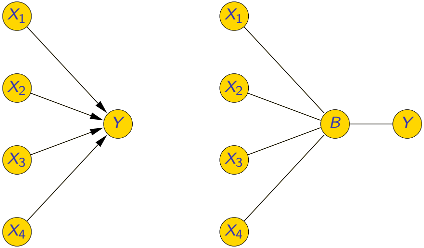
Rank-one decomposition (example)



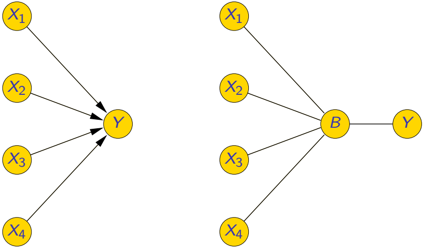
Compilation of a noisy-or gate - rank-one decomposition



Compilation of a noisy-or gate - rank-one decomposition

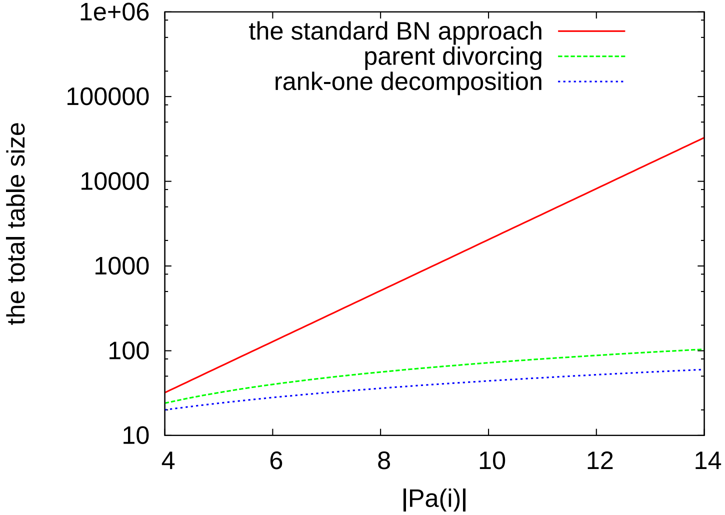


Compilation of a noisy-or gate - rank-one decomposition



The total table size is $5 \cdot 2^2 = 20$.

Comparisons for the noisy-or gate



A computational structure - arithmetic circuits

A computational task is: given evidence on some variables in BN compute the probabilities for remaining variables.

A computational structure - arithmetic circuits

A computational task is: given evidence on some variables in BN compute the probabilities for remaining variables.

Definition (Arithmetic circuit (AC))

An AC is a rooted acyclic directed graph whose leaf nodes correspond to its inputs and whose other nodes are labeled with multiplication and addition operations. The root node corresponds to the output of the AC.

A computational structure - arithmetic circuits

A computational task is: given evidence on some variables in BN compute the probabilities for remaining variables.

Definition (Arithmetic circuit (AC))

An AC is a rooted acyclic directed graph whose leaf nodes correspond to its inputs and whose other nodes are labeled with multiplication and addition operations. The root node corresponds to the output of the AC.

Circuit inputs:

A computational structure - arithmetic circuits

A computational task is: given evidence on some variables in BN compute the probabilities for remaining variables.

Definition (Arithmetic circuit (AC))

An AC is a rooted acyclic directed graph whose leaf nodes correspond to its inputs and whose other nodes are labeled with multiplication and addition operations. The root node corresponds to the output of the AC.

Circuit inputs:

- *model parameters* (i.e, values in probability tables)

A computational structure - arithmetic circuits

A computational task is: given evidence on some variables in BN compute the probabilities for remaining variables.

Definition (Arithmetic circuit (AC))

An AC is a rooted acyclic directed graph whose leaf nodes correspond to its inputs and whose other nodes are labeled with multiplication and addition operations. The root node corresponds to the output of the AC.

Circuit inputs:

- *model parameters* (i.e, values in probability tables)
- *evidence indicators*

$$\lambda_x = \begin{cases} 1 & \text{if state } x \text{ of } X \text{ is consistent with evidence } \mathbf{e} \\ 0 & \text{otherwise.} \end{cases}$$

If there is no evidence for X, then $\lambda_x = 1$ for all states x of X.

A computational structure - arithmetic circuits

A computational task is: given evidence on some variables in BN compute the probabilities for remaining variables.

Definition (Arithmetic circuit (AC))

An AC is a rooted acyclic directed graph whose leaf nodes correspond to its inputs and whose other nodes are labeled with multiplication and addition operations. The root node corresponds to the output of the AC.

Circuit inputs:

- *model parameters* (i.e, values in probability tables)
- *evidence indicators*

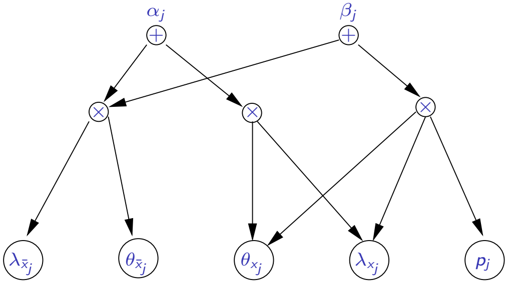
$$\lambda_x = \begin{cases} 1 & \text{if state } x \text{ of } X \text{ is consistent with evidence } \mathbf{e} \\ 0 & \text{otherwise.} \end{cases}$$

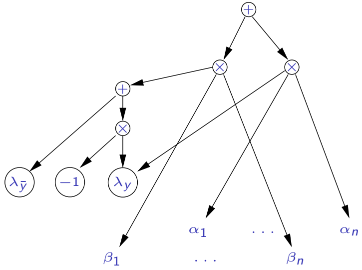
If there is no evidence for X, then $\lambda_x = 1$ for all states x of X.

Circuit output:

- probability of evidence $P(\mathbf{e})$.

AC of a noisy-or gate





Arithmetic circuits (ACs) - Part I

- After an upward pass through the AC we get $P(\mathbf{e})$ in the root node.

Arithmetic circuits (ACs) - Part I

- After an upward pass through the AC we get $P(\mathbf{e})$ in the root node.
- When it is followed by a downward pass through the AC we get $P(X, \mathbf{e})$ for all BN variables X.

Arithmetic circuits (ACs) - Part I

- After an upward pass through the AC we get $P(\mathbf{e})$ in the root node.
- When it is followed by a downward pass through the AC we get $P(X, \mathbf{e})$ for all BN variables X.
- An AC may be used to represent the computations in a junction tree.

Arithmetic circuits (ACs) - Part I

- After an upward pass through the AC we get $P(\mathbf{e})$ in the root node.
- When it is followed by a downward pass through the AC we get $P(X, \mathbf{e})$ for all BN variables X.
- An AC may be used to represent the computations in a junction tree.
- An AC may also represent more efficient computations due to specific properties of the initial BN (e.g., determinism, context specific independence).

Arithmetic circuits (ACs) - Part I

- After an upward pass through the AC we get $P(\mathbf{e})$ in the root node.
- When it is followed by a downward pass through the AC we get $P(X, \mathbf{e})$ for all BN variables X.
- An AC may be used to represent the computations in a junction tree.
- An AC may also represent more efficient computations due to specific properties of the initial BN (e.g., determinism, context specific independence).
- The size of an AC (i.e. number of its edges) can be used as a measure of inference complexity

Arithmetic circuits (ACs) - Part II

- Darwiche et al. proposed two different methods for constructing ACs of BNs - c2d and tabular - both are implemented in a BN compiler called Ace (by Chavira and Darwiche).

Arithmetic circuits (ACs) - Part II

- Darwiche et al. proposed two different methods for constructing ACs of BNs - c2d and tabular - both are implemented in a BN compiler called Ace (by Chavira and Darwiche).
- Ace uses the parent divorcing method for preprocessing noisy-or models.

Arithmetic circuits (ACs) - Part II

- Darwiche et al. proposed two different methods for constructing ACs of BNs - c2d and tabular - both are implemented in a BN compiler called Ace (by Chavira and Darwiche).
- Ace uses the parent divorcing method for preprocessing noisy-or models.
- We use the size of ACs to compare the effect of preprocessing Bayesian networks by Ace's parent divorcing giving (what we call) the original model and by rank-one decomposition giving the transformed model.

Experiments

- Experiments were performed using Ace running on `aligator.utia.cas.cz`: 8x AMD Opteron 8220, 64GB RAM but the maximum possible memory for 32 bit Ace is 3.6 GB RAM.

Experiments

- Experiments were performed using Ace running on `aligator.utia.cas.cz`: 8x AMD Opteron 8220, 64GB RAM but the maximum possible memory for 32 bit Ace is 3.6 GB RAM.
- We carried out experiments with BN2O models of various sizes:

`x`	is the number of nodes in the top level,
`y`	is the number of nodes in the bottom level,
`e`	is the total number of edges in the BN2O model, and
`e/y`	the number of parents for each node from the bottom level.

Experiments

- Experiments were performed using Ace running on `aligator.utia.cas.cz`: 8x AMD Opteron 8220, 64GB RAM but the maximum possible memory for 32 bit Ace is 3.6 GB RAM.
- We carried out experiments with BN2O models of various sizes:

x	is the number of nodes in the top level,
y	is the number of nodes in the bottom level,
e	is the total number of edges in the BN2O model, and
e/y	the number of parents for each node from the bottom level.

- For each x-y-e type ($\mathrm{x}, \mathrm{y} = 10, 20, 30, 40, 50$ and $\mathrm{e}/\mathrm{y} = 2, 5, 10, 20$, excluding those with $\mathrm{e}/\mathrm{y} > \mathrm{x}$) we generated randomly ten models.

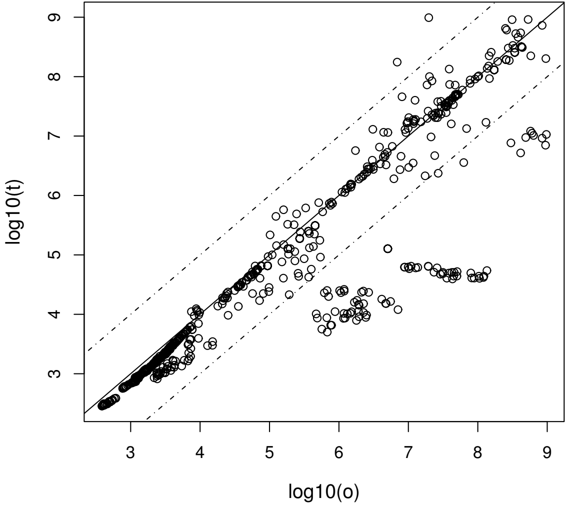
Experiments

- Experiments were performed using Ace running on `aligator.utia.cas.cz`: 8x AMD Opteron 8220, 64GB RAM but the maximum possible memory for 32 bit Ace is 3.6 GB RAM.
- We carried out experiments with BN2O models of various sizes:

`x`	is the number of nodes in the top level,
`y`	is the number of nodes in the bottom level,
`e`	is the total number of edges in the BN2O model, and
`e/y`	the number of parents for each node from the bottom level.

- For each `x-y-e` type ($\mathtt{x}, \mathtt{y} = 10, 20, 30, 40, 50$ and $\mathtt{e}/\mathtt{y} = 2, 5, 10, 20$, excluding those with $\mathtt{e}/\mathtt{y} > \mathtt{x}$) we generated randomly ten models.
- For every node from the bottom level we randomly selected `e/y` nodes from the top level as its parents.

Transformed vs. original model AC size



Summary of the experiments

- The AC of the transformed model was smaller in 88% of the BN2O models

Summary of the experiments

- The AC of the transformed model was smaller in 88% of the BN2O models
- In several cases we got significant reductions in the AC size - in a few cases multiple order of magnitude.

Summary of the experiments

- The AC of the transformed model was smaller in 88% of the BN2O models
- In several cases we got significant reductions in the AC size - in a few cases multiple order of magnitude.
- There are also eleven cases where the AC of the transformed model is at least three times larger - we will comment on these cases on the next slide.

Comments to the eleven cases with a significant loss

- The transformed model is a graph minor of the original model.

Comments to the eleven cases with a significant loss

- The transformed model is a graph minor of the original model.
- Hence, the treewidth of the transformed model can never be larger than the treewidth of the original model.

Comments to the eleven cases with a significant loss

- The transformed model is a graph minor of the original model.
- Hence, the treewidth of the transformed model can never be larger than the treewidth of the original model.
- However, if a heuristic triangulation method is used then it may happen that we get larger treewidth for the triangulated graph of the transformed model.

Comments to the eleven cases with a significant loss

- The transformed model is a graph minor of the original model.
- Hence, the treewidth of the transformed model can never be larger than the treewidth of the original model.
- However, if a heuristic triangulation method is used then it may happen that we get larger treewidth for the triangulated graph of the transformed model.
- We conducted additional experiments with all eleven models with significant loss in the AC size.

Comments to the eleven cases with a significant loss

- The transformed model is a graph minor of the original model.
- Hence, the treewidth of the transformed model can never be larger than the treewidth of the original model.
- However, if a heuristic triangulation method is used then it may happen that we get larger treewidth for the triangulated graph of the transformed model.
- We conducted additional experiments with all eleven models with significant loss in the AC size.
- In all of these eleven cases we were able to get the AC of the transformed model smaller than the AC of original model using the triangulation derived from the triangulation of the original model.