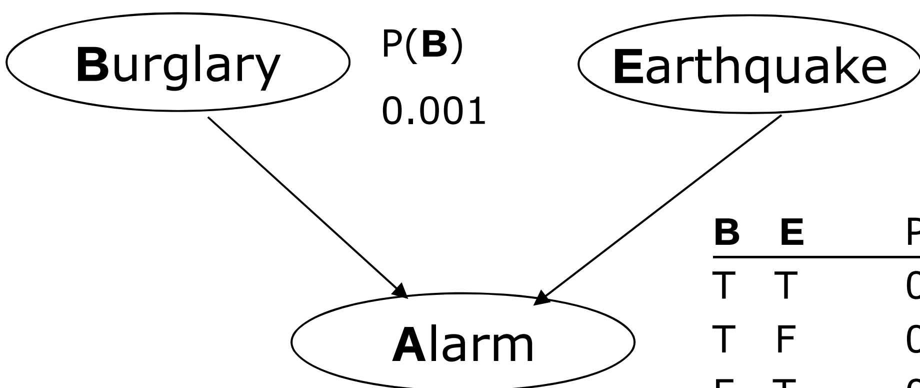
Bayesian Networks
Learning Bayesian Networks

Nysret Musliu

Bayesian Networks

- Set of variables
- Set of directed arcs between variables
- Acyclic graph
- Probabilities for every variable (node) given its parents

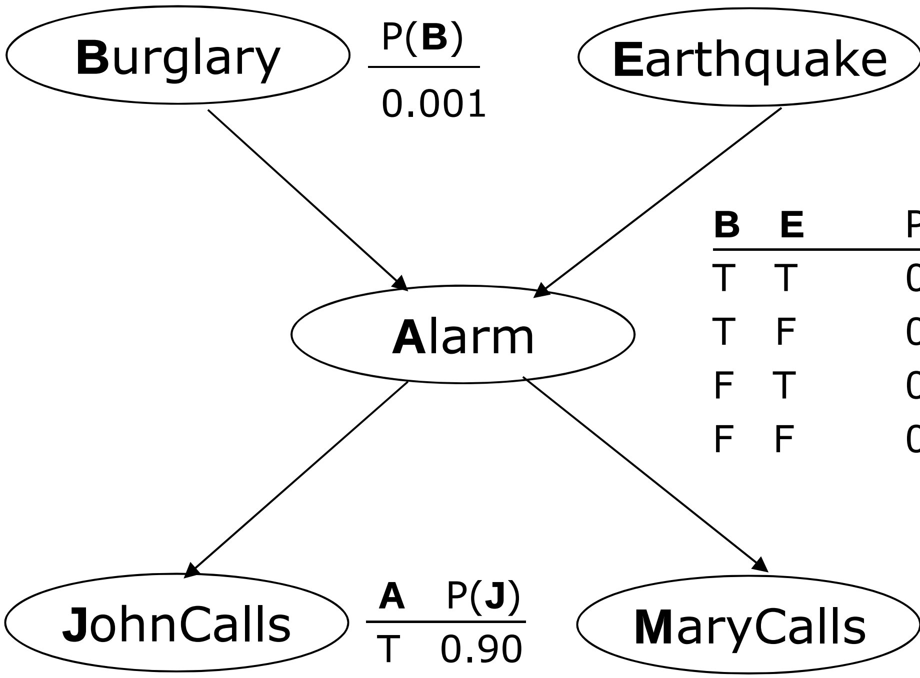


P(**E**)

0.002

B	**E**	P(**A**)
T	T	0.95
T	F	0.94
F	T	0.29
F	F	0.001

Bayesian Networks (Example)



P(**B**)
0.001

P(**E**)
0.002

B	**E**	P(**A**)
T	T	0.95
T	F	0.94
F	T	0.29
F	F	0.001

A	P(**J**)
T	0.90
F	0.05

A	P(**M**)
T	0.70
F	0.01

Bayesian Networks

- Components
  - Structure (how to determine it?)
  - Numerical parameters (probabilities)
- To answer questions we need to know joint probability distribution
- Example:

  n variables (values 0 or 1):

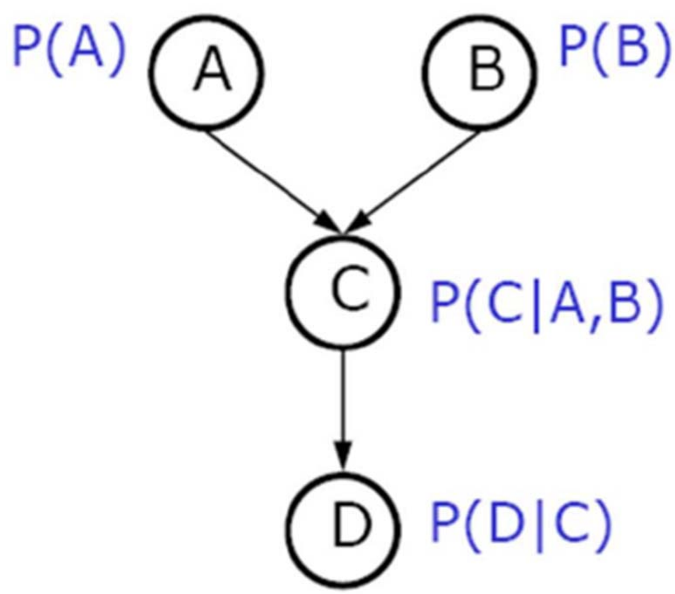
  2^n number are needed to specify joint probability distribution
- Exploit independence between variables
  - D-seperation

Chain Rule (Example)

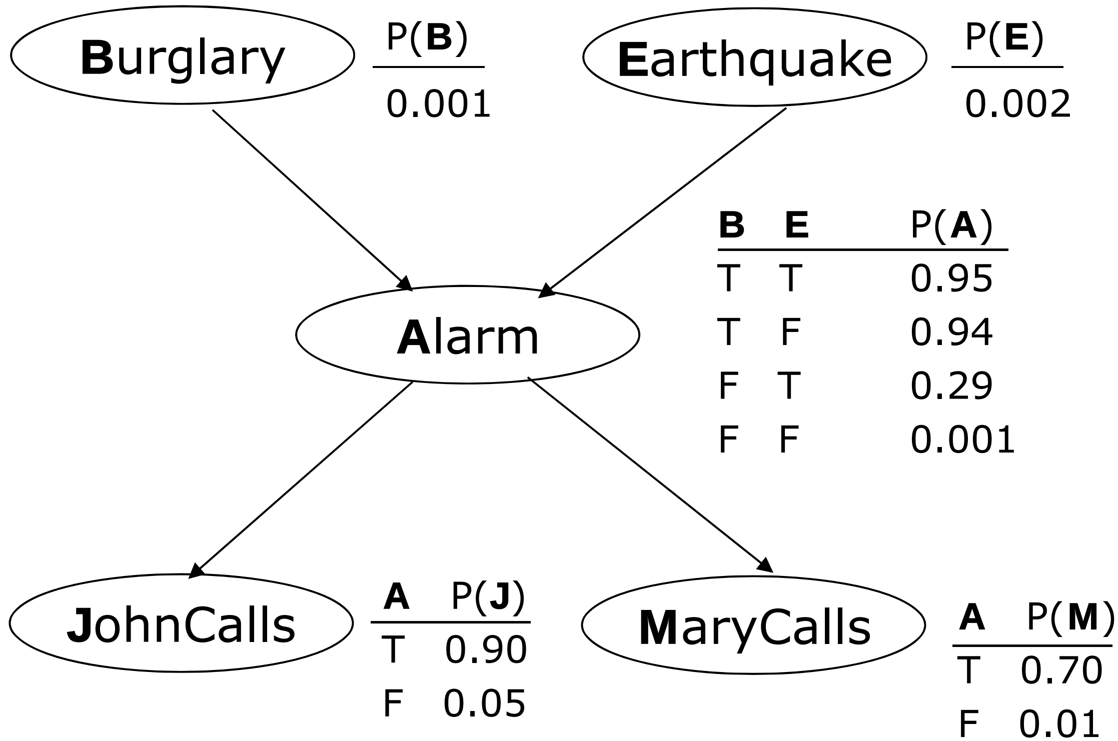
P(ABCD)=P(D|ABC) * P(ABC) =

P(D|C) * P(C|AB) * P (AB)=

P(D|C) * P(C|AB) * P (A) * P(B)



Inference in Bayesian Networks



Inference: answer questions based on some evidence

e.g.:

Evidence: JohnCalls

Query: probability of Burglary?

Types of Questions

Given some evidence, compute the probability distribution over query variables

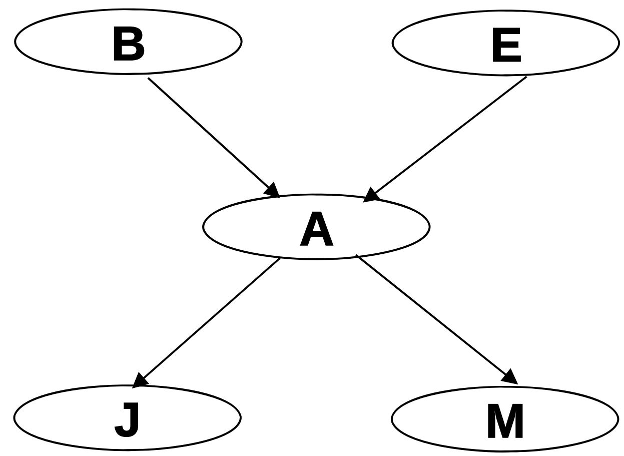
$P(Q_1, Q_2, \ldots | E_1=e_1, E_2=e_2, \ldots)$

Maximum a posteriori probability (most likely explanation)

$\mathrm{argmax}_q \; P(Q_1=q_1, Q_2=q_2, \ldots | E_1=e_1, E_2=e_2, \ldots)$

Answering Queries with Enumeration

Sum over variables not involved in the query

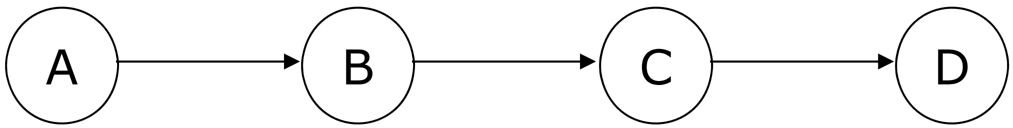


$P(B=t|J=t)=P(B=t, J=t)/P(J=t)$

$P(B=t, J=t)=\Sigma_{a \in dom(A)} \Sigma_e \Sigma_m P(J=t|A)*P(M|A)*P(A|B=t, E)*P(B=t)*P(E)$

$P(J=t)=\Sigma_a \Sigma_e \Sigma_m \Sigma_b P(J=t|A)*P(M|A)*P(A|B, E)*P(B)*P(E)$

Example 2



$P(D=t)=\Sigma_a \Sigma_b \Sigma_c P(D=t|C)*P(C|B)*P(B|A)*P(A)$

Too many computations: For all combinations of possible values of A, B, and C the factors should be multiplied

Practical domains include large number of variables

Improving Efficiency

$$A \rightarrow B \rightarrow C \rightarrow D$$

$P(D=t) = \Sigma_a \Sigma_b \Sigma_c P(D=t|C)*P(C|B)*P(B|A)*P(A)$

$P(D=t) = \Sigma_c P(D=t|C) \Sigma_b P(C|B) \Sigma_a P(B|A)*P(A)$

Store intermediate results for $\Sigma_a P(B|A)*P(A)$ for each value of B in

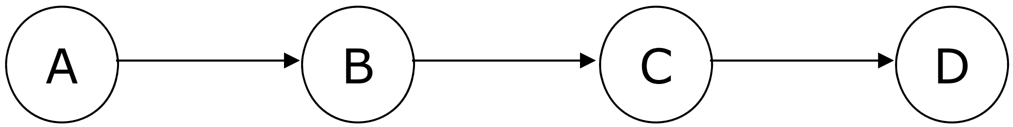
$f_1(b)$:

$f_1(b_1)= \Sigma_a P(B=b_1|A)*P(A)$

$f_1(b_2)= \Sigma_a P(B=b_2|A)*P(A)$

…

Improving Efficiency



$P(D=t)=\sum_c P(D=t|C) \sum_b P(C|B)\ f_1(b)= \sum_c P(D=t|C)\ f_2(c)$

$f_2(c)$:

$f_2(c_1)= \sum_b P(C=c_1|B)\ \ f_1(b)$

$f_2(c_2)= \sum_b P(C=c_1|B)\ \ f_1(b)$

…

Variable Elimination Algorithm

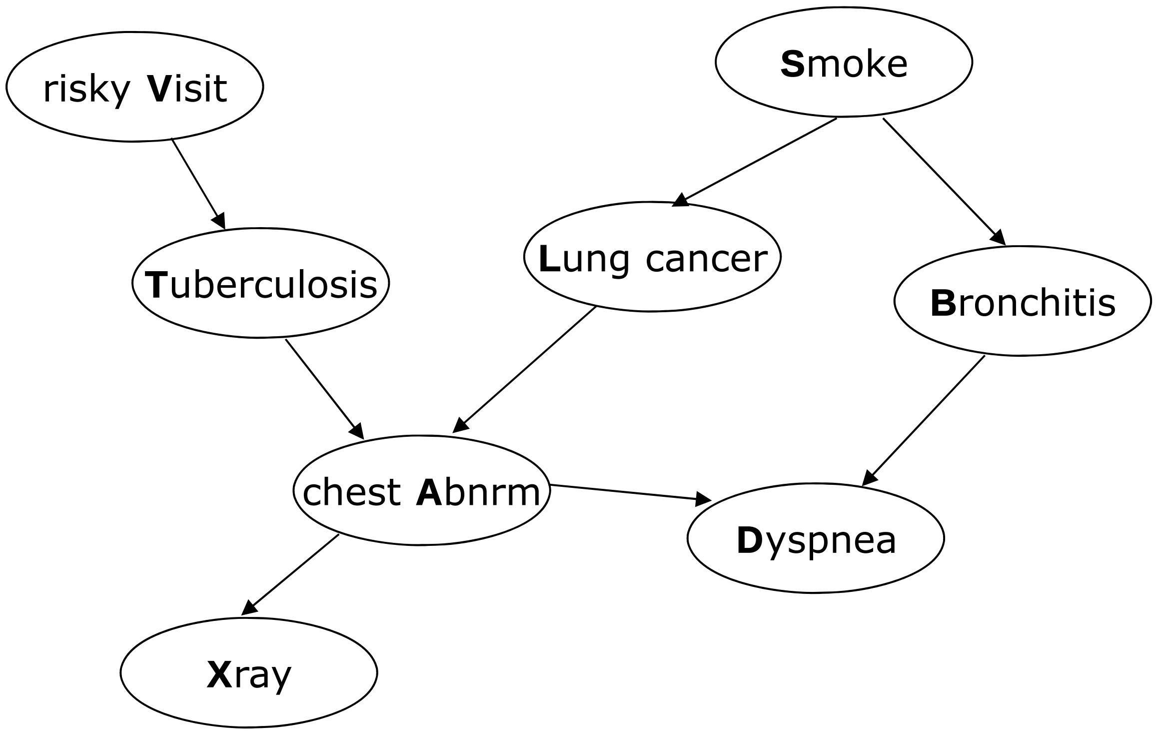
The previous algorithm is called Variable Elimination Algorithm

The computation can be performed more efficiently, if the order of variables is chosen carefully

Finding best elimination order: NP-hard

Usually heuristics are used for finding good elimination order

Example 2

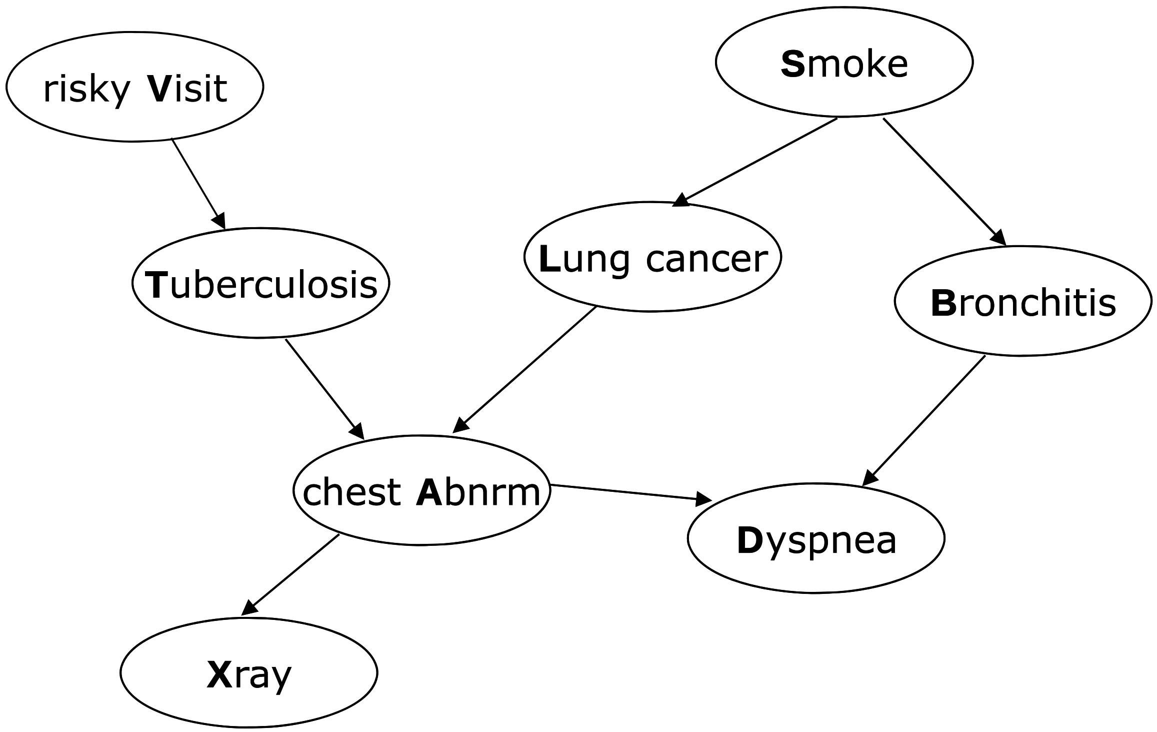


$P(D=t) = \Sigma_{abltsxv}\ P(D=t|A,B)\ P(B|S)\ P(L|S)\ P(S)\ P(V)\ P(T|V)\ P(X|A)\ P(A|T,L)$

Example from Kaelbling, MIT (Lectures 15, 16)

http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-825-techniques-in-artificial-intelligence-sma-5504-fall-2002/lecture-notes/

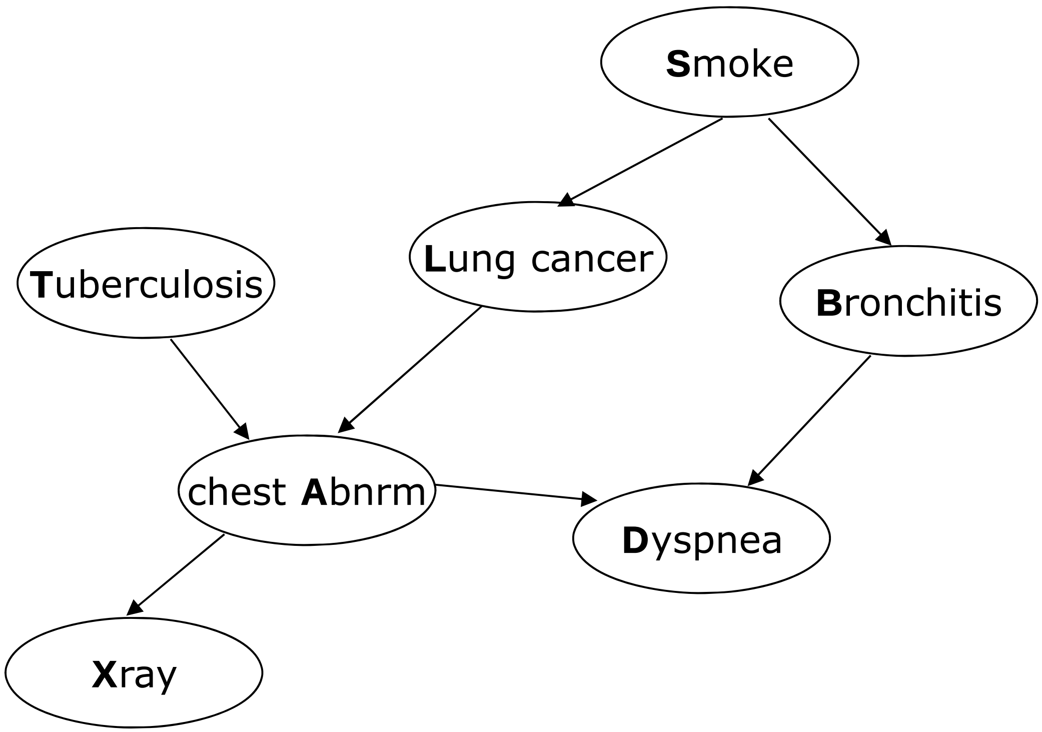
Example 2



Eliminate V : $f_1(t) = \sum_v P(T|V)\, P(V)$

$P(D=t) = \sum_{abltsx} P(D=t|A,B)\, P(B|S)\, P(L|S)\, P(S)\, P(X|A)\, P(A|T,L)\, f_1(t)$

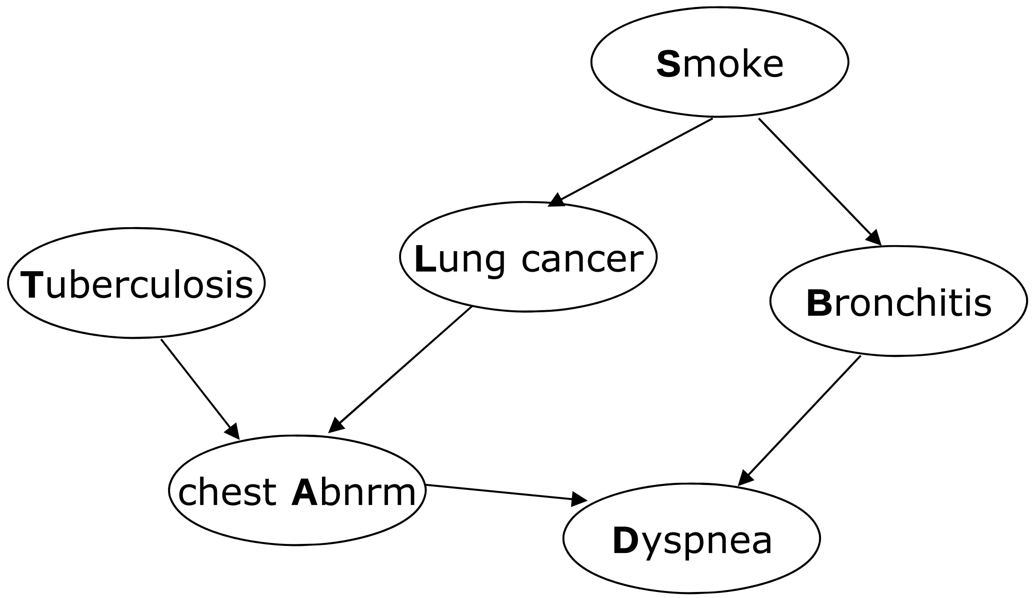
Example 2



Eliminate X : $\sum_x P(X|A) = 1$

$P(D=t) = \sum_{ablts} P(D=t|A,B)\, P(B|S)\, P(L|S)\, P(S)\, P(A|T,L)\, f_1(t)$

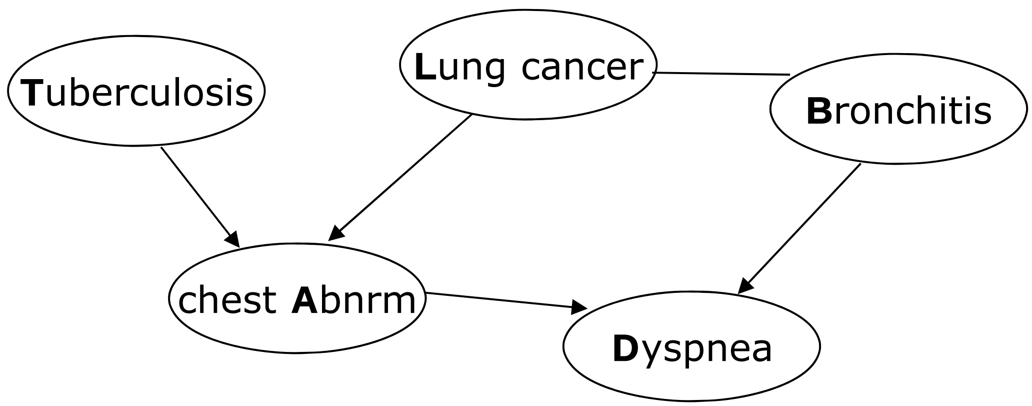
Example 2



Eliminate S : $f_2(b,l) = \sum_s P(B|S)\ P(L|S)\ P(S)$

$P(D=t) = \sum_{ablt} P(D=t|A,B)\ P(A|T,L)\ f_2(b,l)\ f_1(t)$

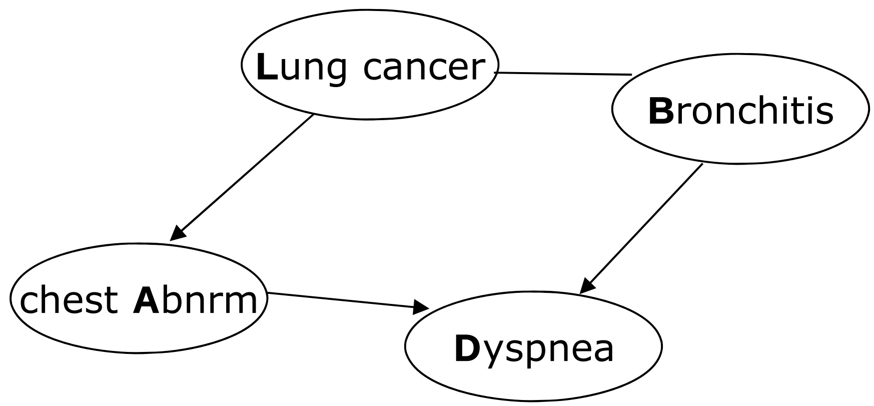
Example 2



Eliminate T : $f_3(a,l) = \Sigma_t \, P(A|T,L) \, f_1(t)$

$P(D=t) = \Sigma_{abl} \, P(D=t|A,B) \, f_2(b,l) \, f_3(a,l)$

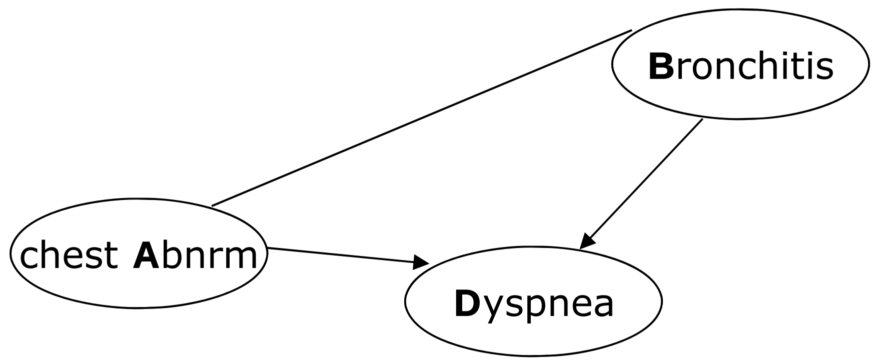
Example 2



Eliminate L: $f_4(a,b) = \sum_l f_2(b,l)\, f_3(a,l)$

$P(D=t) = \sum_{ab} P(D=t|A,B)\, f_4(a,b)$

Example 2



Eliminate B: $f_5(a) = \sum_b P(D{=}t|A,B)\ f_4(a,b)$

$P(D{=}t) = \sum_a f_5(a)$

Good Variable Elimination can improve the efficiency, but:

Exponential in size of largest factor

In large networks other methods which do approximation are used (e.g. Sampling)

Learning Bayesian Networks

Possibilities for building networks:

- Human experts
- Learning from data
- Combination of both approaches

Human experts are good on finding a structure

Computers better on calculating probabilities

Learning from data

Data are given:
   e.g.
   Data of patients and their and information if they had a particular disease

Structure known (for example from human expert):
   Learn probabilities

Structure not known:
   Learn the structure of the Bayesian network and the probabilities

Another case:
   Not all variables are observable
   Not subject of this course

Learning BN with known structure

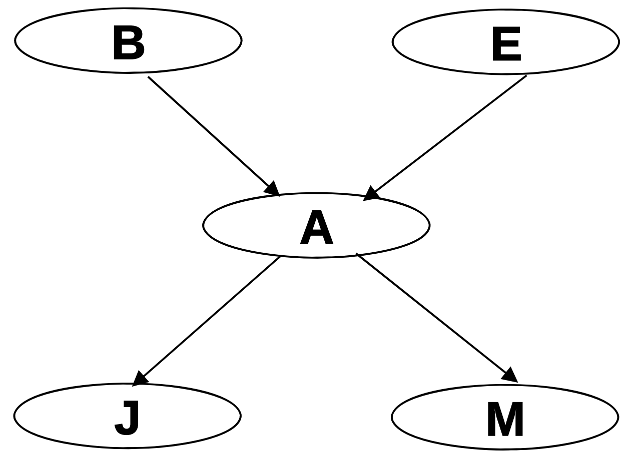
Bayesian Network is given

Data set (D) is given

B	E	A	J	M
t	f	t	t	f
f	f	f	f	t
f	t	f	t	f
...				

Find model M (conditional probability tables - CPTs)

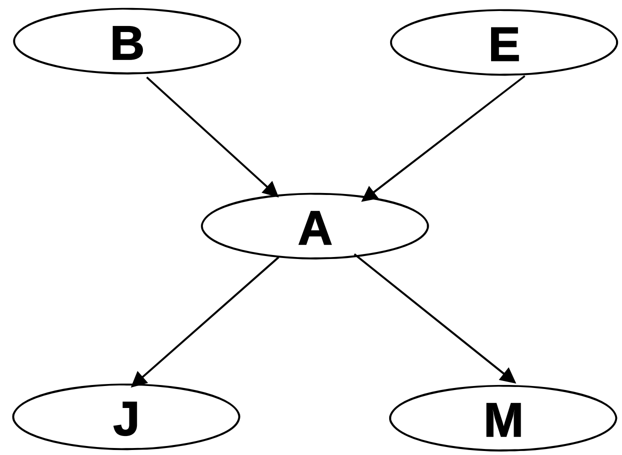
Maximum likelihood model maximize P(D|M)



A	P(**J**)
T	?
F	?

Computing CPTs

B	E	A	J	M
t	f	t	t	f
f	f	f	f	t
f	t	f	t	f
…				



P(**E**)

A	P(**J**)
T	?
F	?

Use counts

$P(J|A)=(\#A = true \wedge J = true) / \#(A = true)$

Avoid 0 probabilities (Laplace correction):

$P(J|A)=(\#(A = true \wedge J = true)+\mathbf{1})/ (\#(A = true) +\mathbf{2})$

$P(E) = (\#(E = true)+\mathbf{1}) / (k+\mathbf{2})$

k: number of samples

Laplace correction

Goodness of Fit

Data set D

Model M

Goodness of the fit:

$P(D|M)=\Pi_j\, P(s^j|M) = \Pi_j\, \Pi_i\, P(N_i=v_i^j|Parents(N_i), M)$

log likelihood

$\log P(D|M)= \Sigma_j\, \Sigma_i\, P(N_i=v_i^j|Parents(N_i), M)$

Learning the Structure

Objective

Find a network that

- is a good fit to data
- has low complexity (less parameters)

Maximize:

$$\log P(D|M) - \alpha \, \#M$$

α -> parameter which indicates how important is to reduce the complexity of the network

Search problem

Search for a Bayesian network that maximizes the objective

Exhaustive search not practical (to many networks)

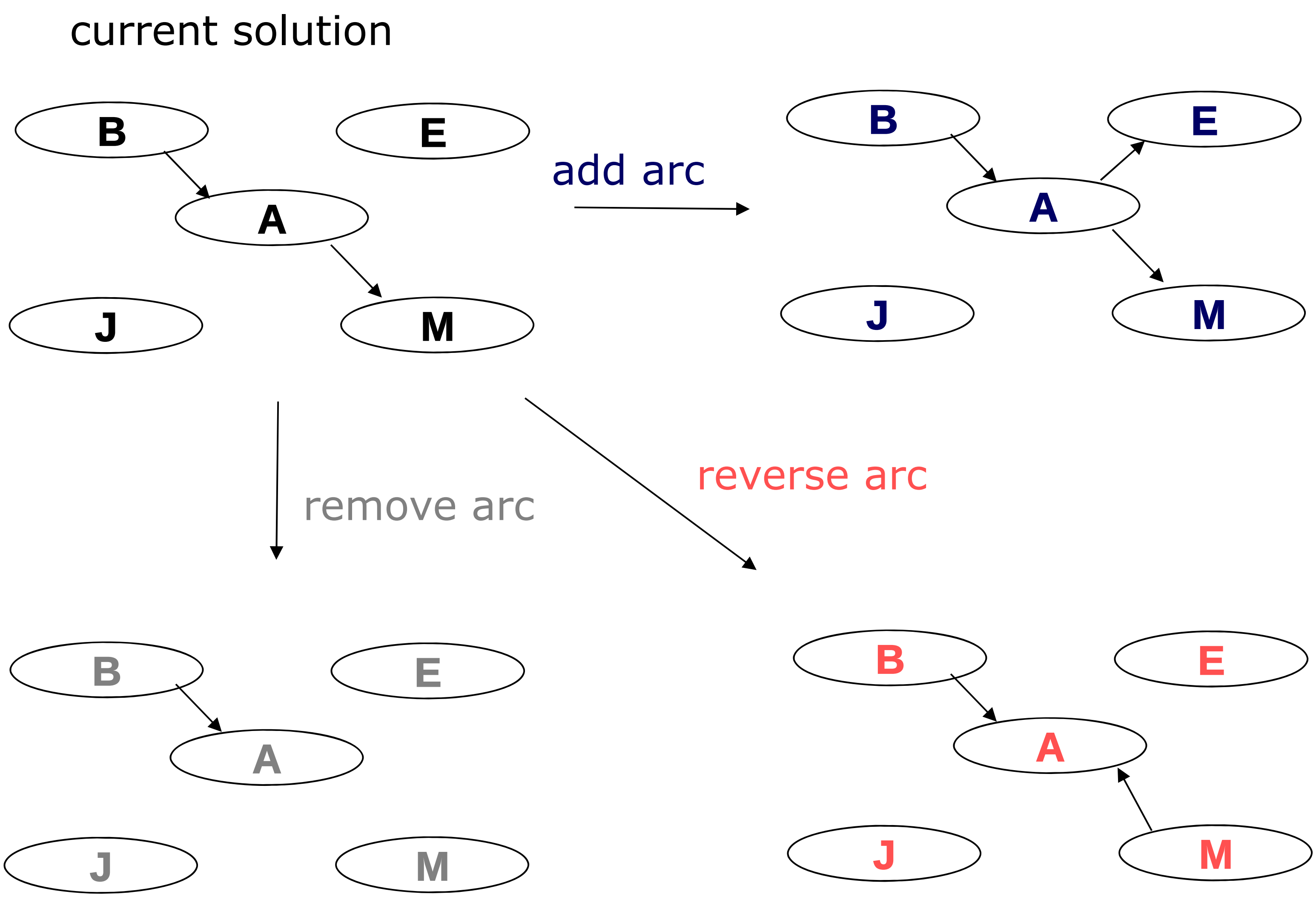
Use heuristic techniques

- local search
- population based techniques
- …

Local search for Bayesian Networks

1. Construct an initial network
2. Calculate the score of the current BN network (with learned probabilities)
3. Generate the neighborhood by modifying the current network
4. Select one of networks in the neighborhood as a new current network for the next iteration
5. Go to Step 3 if termination criteria is not fulfilled

Neighborhood relations



Neighborhood exploration

- Generate neighborhood solutions by applying neighborhood relations

- Hill Climbing, Tabu Search
  - all solutions in the neighborhood are generated
- Simulated Annealing
  - only one random solution in the neighborhood is generated

Selection of the solution

- Hill Climbing
  - Best solution in the neighborhood is selected
- Simulated Annealing
  - The randomly generated solution is accepted based on some probability that depends from the quality of the generated solution and temperature T
    - Worse solution can be accepted for the next iteration
- Tabu Search
  - Best solution is selected if is not tabu (otherwise the solution that is not tabu is selected)
    - Worse solution can be accepted for the next iteration

Literature

MIT lectures:

http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-825-techniques-in-artificial-intelligence-sma-5504-fall-2002/lecture-notes/

- Artificial Intelligence: A Modern Approach (Third edition) by Stuart Russell and Peter Norvig (Chapter 14, 20)

- Bayesian Network Classifiers in Weka for Version 3-5-7. Remco R. Bouckaert

Appendix

A short introduction to local search techniques

Definition of search problem

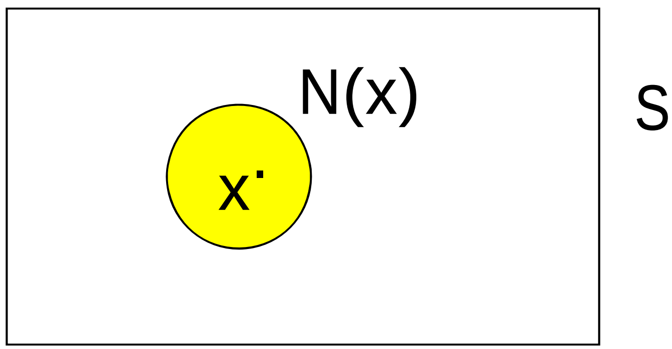
- Given e search space S together with its feasible part $F \subseteq S$, find $x \in F$ such that

  $eval(x) \leq eval(y)$ for all $y \in F$

- x that satisfies the above condition is called global optimum (for minimization problem)

Neighbourhood and local optima

- Region of the search space that is near particular point in the space



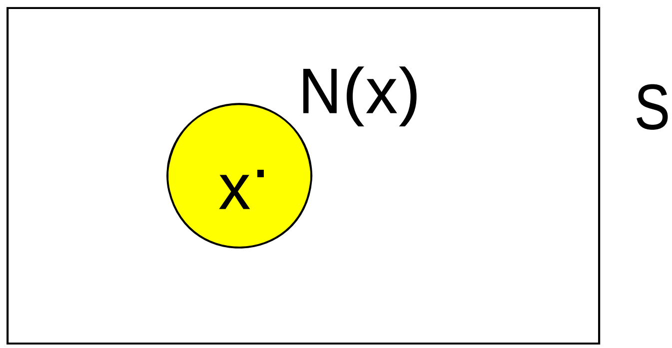
- A potential solution $x \in F$ ia a local optimum with respect to the neighborhood N, if and only if

  $eval(x) \leq eval(y)$,

  for all $y \in N(x)$

Local Search Techniques

- Are based on the neighbourhood of the current solution



- The solution is changed iteratively with so called neighbourhood relations (moves) until an acceptable or optimal solution is reached

Hill Climbing Algorithm

1. Pick a random point in the search space
2. Consider all the neighbours of the current state
3. Choose the neighbour with the best quality and move to that state
4. Repeat 2 through 4 until all the neighbouring states are of lower quality
5. Return the current state as the solution state

Simulated Annealing

```
Prozedure simulated annealing
 begin
    t=0
    Intialize T
    select a current solution v_c at random
    evaluate v_c
    repeat
     repeat
        select a new solution v_n in the neighborhood of v_c
         if eval(v_c) < eval(v_n) then  v_c =v_n
         else if random[0,1) < e^((eval(v_n)-eval(v_c))/T) then v_c=v_n
     until (termination-condition)
     T=g(T,t)
     t=t+1
    until (halting-criterion)
 end
```

Where the else-if condition is $random[0,1) < e^{\frac{eval(v_n)-eval(v_c)}{T}}$.

Basic Tabu Search

```
Procedure Tabu-Suche
begin
 Initialize tabu list
 Generate randomly Initial Solution s_c
 Evaluate s_c
 repeat
   Generate all neighborhood solutions of the solution s_c
   Find best solution s_x in the neighborhood
   if s_x is not tabu solution then s_c= s_x
   else if 'aspiration criteria' is fulfilled then
         s_c= s_x
   else
      find best not tabu solution in the neighborhood s_nt
      s_c = s_nt
   Update tabu list
 until (terminate-condition)
end
```

Simple Genetic Algorithm

```
{
    initialize population;
    evaluate population;
    while TerminationCriteriaNotSatisfied
    {
        select parents for reproduction;
        perform recombination and mutation;
        evaluate population;
    }
}
```

Genetic Algorithms: A Tutorial, Wendy Williams