Aprendizaje Estructural

Serafín Moral

Universidad de Granada

Febrero, 2022

Structure Learning

There are two basic approaches:

- Independence tests
- Score + Search procedures

Now, score methods are more used than independence tests, but there are pros and cons for both of them.

We are going to see the PC Algorithm, an algorithm based on independence tests.
Spirtes, Glymour, Scheines (1993) *Causation, Prediction, and Search*

Basic Hypothesis

- The independence relationships have a perfect representation by a DAG
- We have a very large database
- Statistical tests have no errors

Under these conditions, the algorithm will discover and equivalent Bayesian network.

Statistical Tests

The algorithm is based in asking for the true of independence relationships of the form:

$$I(X_i, X_j|A)$$

where A is a subset of variables.
It can work with any source providing this kind of information.
It we have a database, this is answered by means of statistical tests of independence.

Kulback-Leibler Distance

If X is a random variable and P and Q are two probability distributions, then the Kullback-Leibler distance of Q to P is defined as:

$$D(P||Q) = \sum_x P(x) \log \left(\frac{P(x)}{Q(x)} \right)$$

- It is not symmetrical.
- It is 0 if and only if $P \equiv Q$
- If for an x, we have $P(x) > 0$ and $Q(x) = 0$, then the distance is infinite.
- The entropy of P is $\log(r) - D(P||U)$ where r is the number of cases of the variable and U is the uniform distribution

Cross Entropy

If X, Y are random variables with joint probability distribution P the cross entropy between them is

$$CE(X,Y) = \sum_{x,y} P(x,y) \log \left(\frac{P(x,y)}{P(x).P(y)} \right)$$

It is a non-negative value, bounded by the entropy of X and Y. In fact, if $H(X)$ denotes the entropy of X, then

$$CE(X,Y) = H(X) - H(X|Y)$$

where $H(X|Y)$ is the entropy of X conditioned to Y:
$H(X|Y) = \sum_y H(H|y)P(y)$
It is 0.0 only for independent variables and measures the degree of dependence of two variables (any kind of relation).

Example

Compute the cross entropy between X and Y in the following three cases (joint probability)

	y_1	y_2
x_1	0.03	0.54
x_2	0.40	0.03

	y_1	y_2
x_1	0.54	0.03
x_2	0.03	0.40

	y_1	y_2
x_1	0.04	0.15
x_2	0.18	0.63

The first two are close to $\log 2$ and the last one close to 0.

The Conditional Cross Entropy

Given three variables X, Y, Z the cross entropy of X and Y given Z is defined as

$$CE(X,Y|Z) = \sum_z P(z) \sum_{x,y} P(x,y|z) \log \left(\frac{P(x,y|z)}{P(x|z).P(y|z)} \right)$$

It verifies the $CE(X,Y|Z) = H(X|Z) - H(X|Y,Z)$
This value is also called the Mutual Information.
I can be analogously defined when Z is a set of variables.
It verifies similar properties to unconditional entropy. It measures the degree of dependence of X and Y given Z. In particular, it is equal to 0.0 when this conditional independence is verified.

Independence Test

To test whether X and Y are conditional independent given A, we compute the cross entropy $CE(X, Y|A)$ where the probabilities are their maximum likelihood estimators from the database (relative frequencies).
The statistic used for the test is G^2 which is $2mCE(X, Y|A)$ where m is the sample size.
It is known that, under the independence assumption, G^2 follows a χ^2 distribution with degrees of freedom equal to:

$$(r_X - 1)(r_Y - 1) \prod_{Z \in A} r_Z$$

where r_W is the number of values of variable W.
It is possible to decrease in one the number of degrees of freedom for each configuration of values of the variables that does not appear in the database.

Examples

Determine whether there is independence for variables X_1 and X_2 (with values 1 and 2) according to the following databases,

Case	1	2	3	4	5	6	7	8
X_1	1	1	2	2	2	2	1	2
X_2	2	1	1	2	1	1	2	2

can not be rejected with p-value 0.47. Independence

Case	1	2	3	4	5	6	7	8
X_1	1	1	1	1	2	2	2	2
X_2	1	1	1	1	2	2	2	2

Independence can not be rejected with p-value 0.001. Dependence

Sample Size

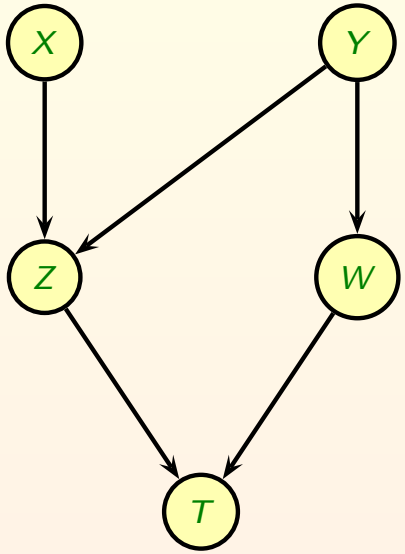
- The significance level is usually 0.01 or 0.05 (greater values are better).
- It assumes dependence when the probability of a χ^2 with the corresponding degrees of freedom is greater than G^2 is smaller than the confidence level.
- In case of accepting independence it does not mean that data support independence, but that there is no evidence in the data against it.
- When sample size or the conditional set is big, then the possibility of rejecting the null hypothesis is lower and independence will be assumed: Lack of support implies independence

The Algorithm Structure

1. Find a graph pattern (gp): an undirected graph
2. Find some head to head links by testing independences
3. Orient the rest of links without producing cycles

Remark: There is some degree of arbitrariness and sometimes, though independences can be represented by a DAG the direction of the arrows is counterintuitive with causality.

Graph Pattern: The Basic Condition



Two nodes, X and Y, are connected if and only if there is no subset S_{XY} of the set of vertices V such that $I(X, Y|S_{X,Y})$.

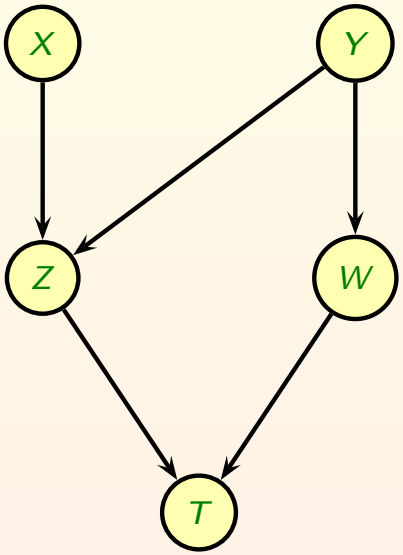
We could try to discover the graph pattern following this criterion, but it will be inefficient (too many tests) and inaccurate (conditioning to many variables).

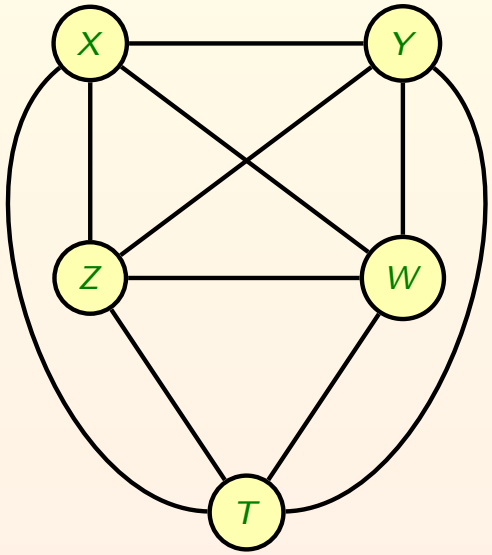
Finding the Graph Pattern

V is the set of nodes, and every independence relationships can be tested. Each node has a set of adjacent nodes ADJ_X.

1. Start with a complete undirected graph gp
2. $i = 0$
3. **Repeat**
    4. **For each** $X \in V$
        5. **For each** $Y \in ADJ_X$
            6. Determine if there is $S \subseteq ADJ_X \setminus \{Y\}$ with $|S| = i$ and $I(X, Y|S)$
            7. **If** this set exists
                8. Make $S_{XY} = S$
                9. Remove $X - Y$ link from gp
    10. $i = i + 1$
11. **Until** $|ADJ_X| \leq i, \quad \forall X$

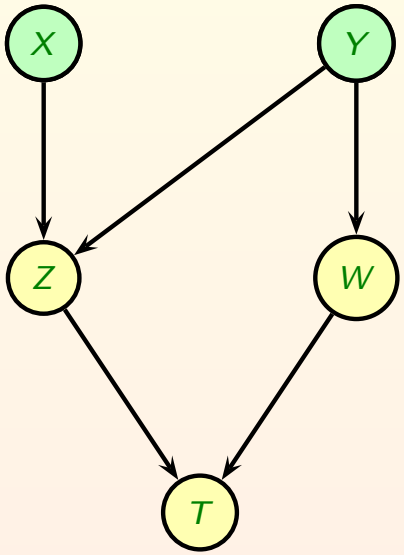
Example



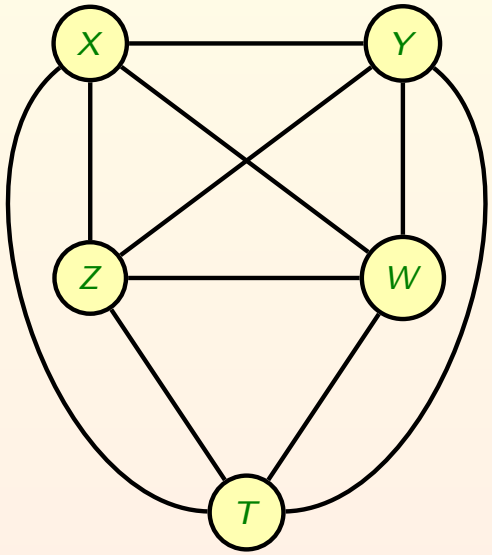


Example

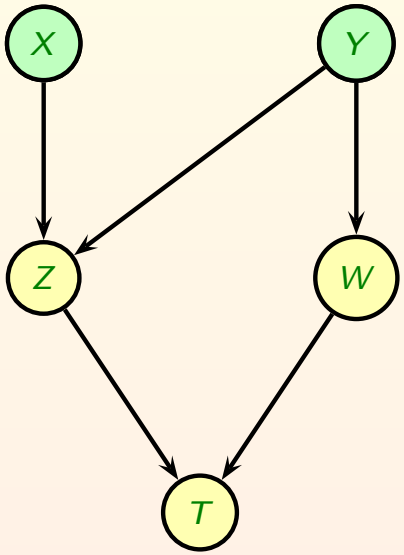
YES

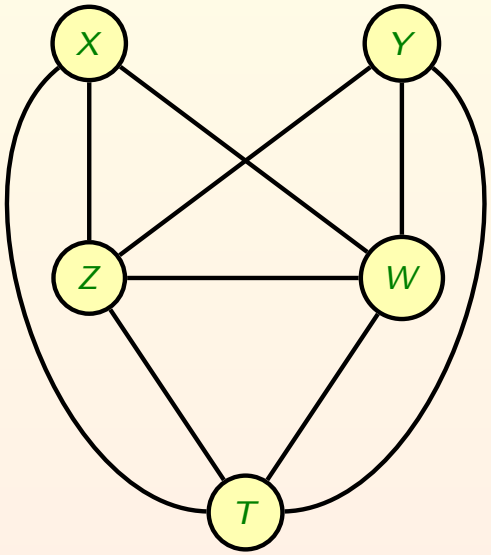


$i = 0$



Example



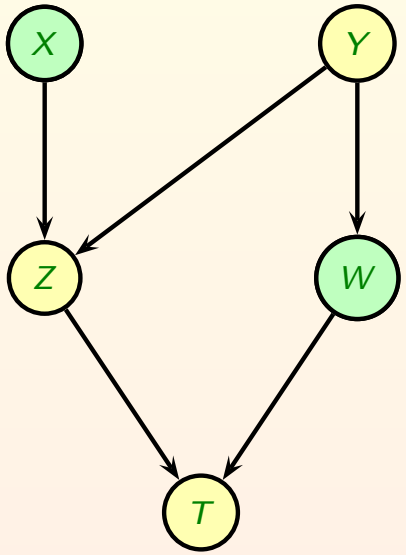


$i = 0$

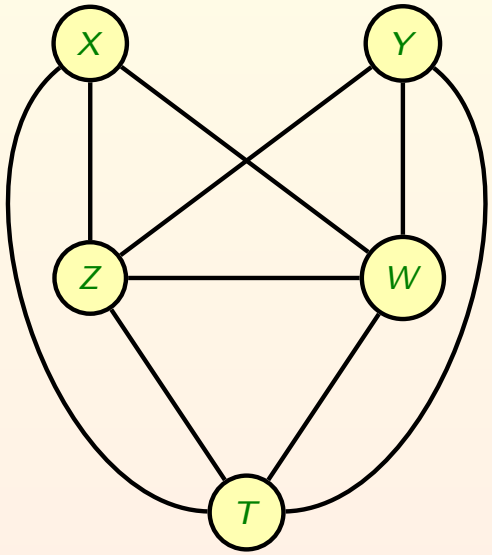
$S_{X,Y} = \emptyset$

Example

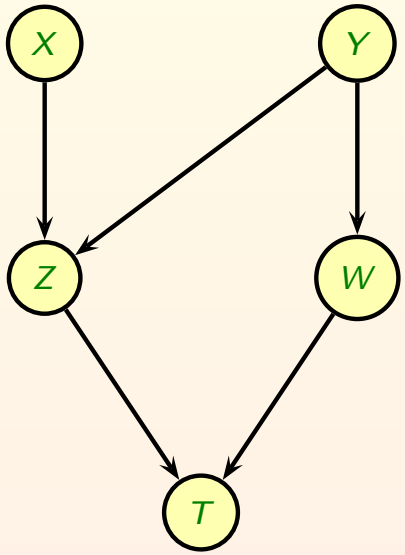
YES

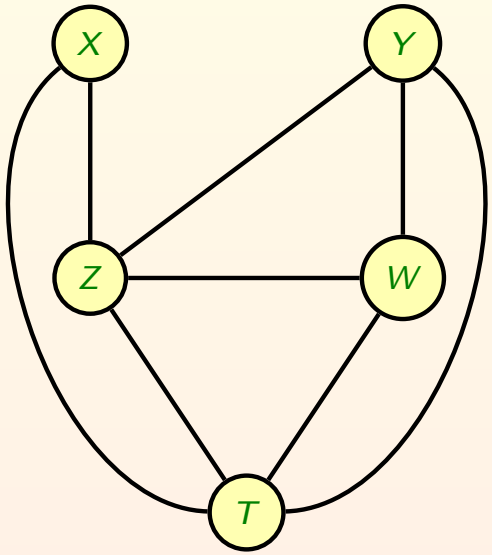


$i = 0$



Example



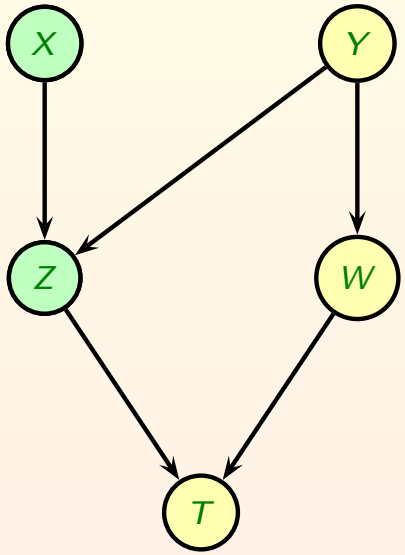


$i = 0$

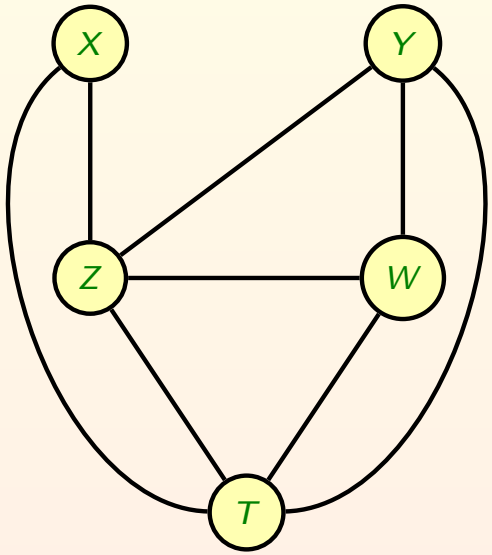
$S_{X,W} = \emptyset$

Example

NO

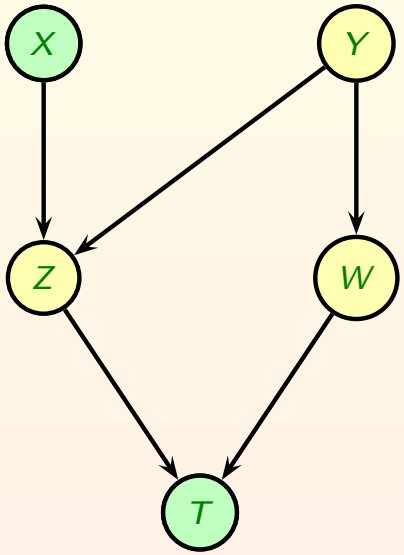


$i = 0$

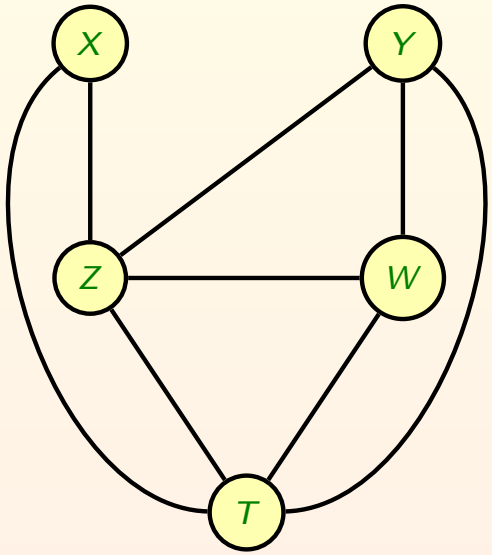


Example

NO

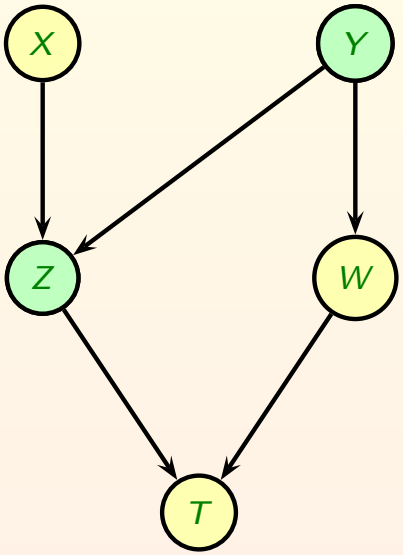


$i = 0$

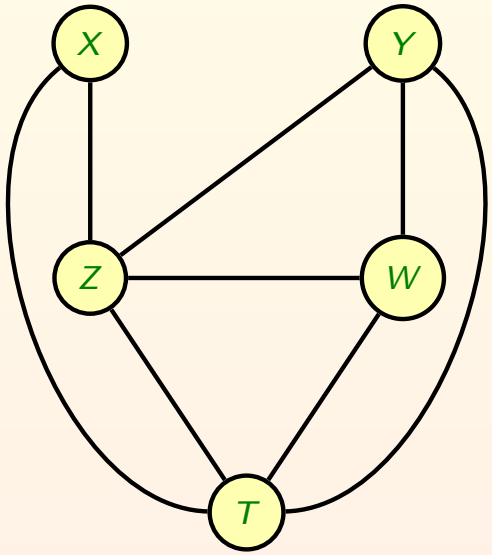


Example

NO

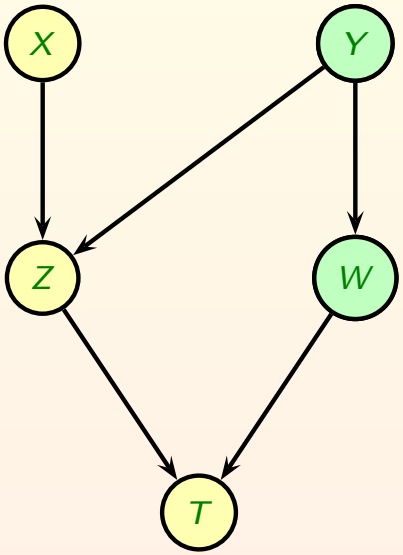


$i = 0$

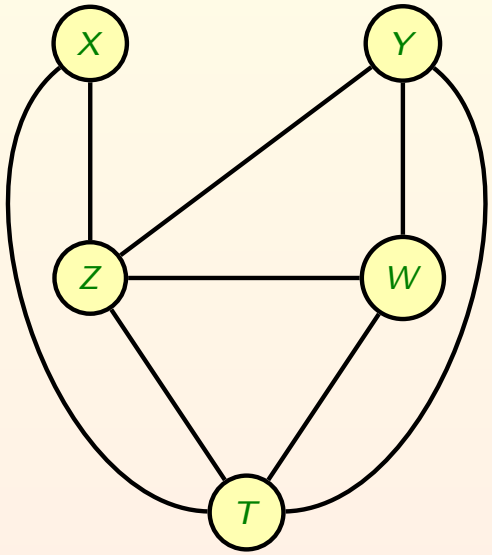


Example

NO

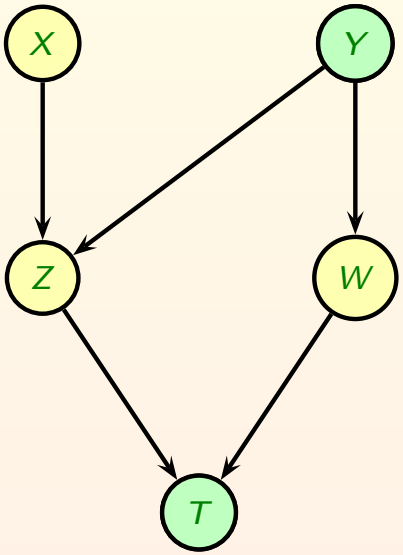


$i = 0$

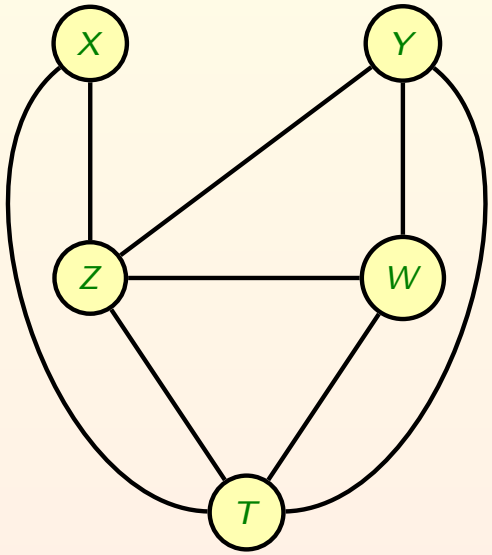


Example

NO

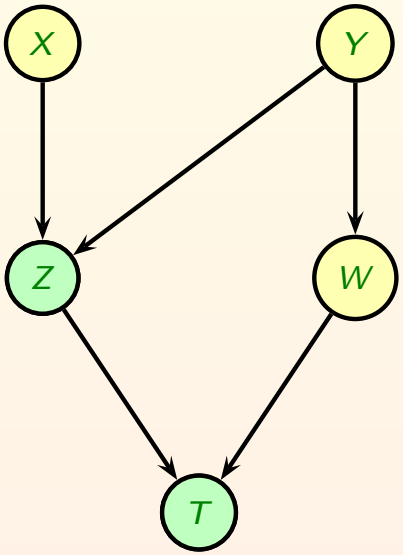


$i = 0$

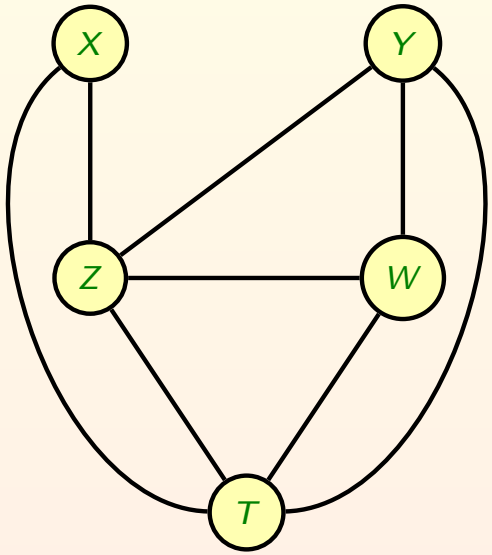


Example

NO

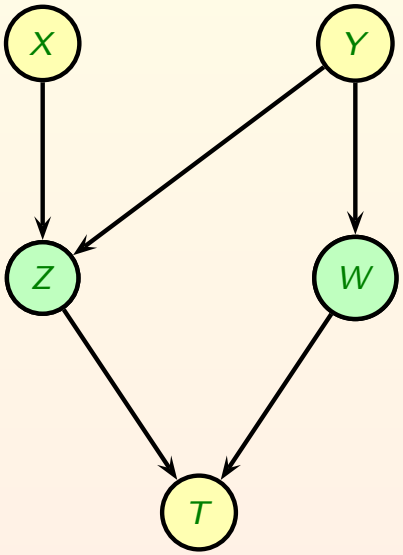


$i = 0$

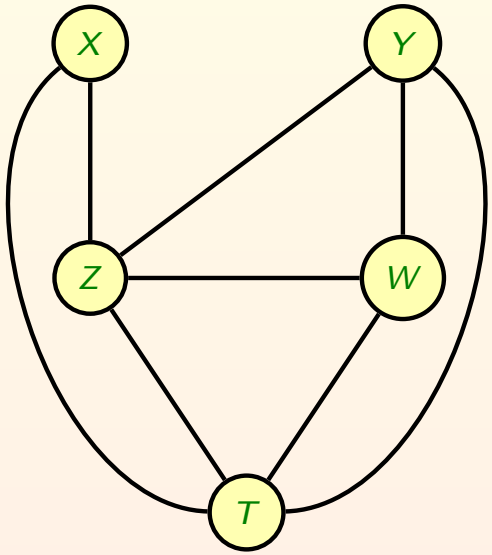


Example

NO

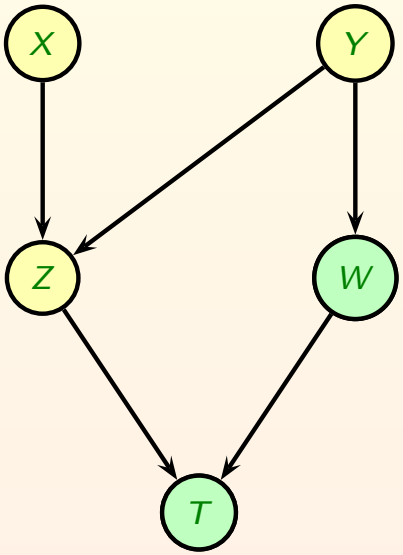


$i = 0$

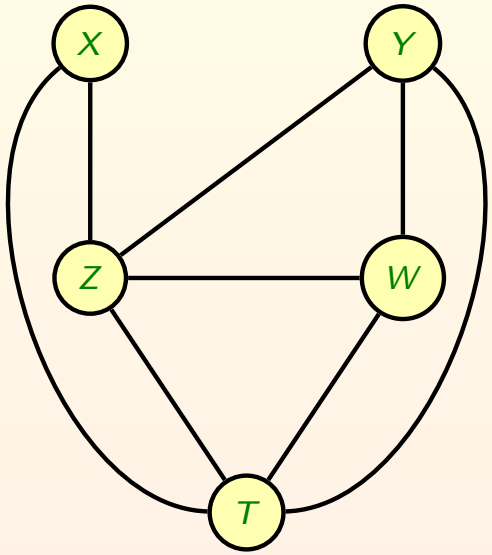


Example

NO

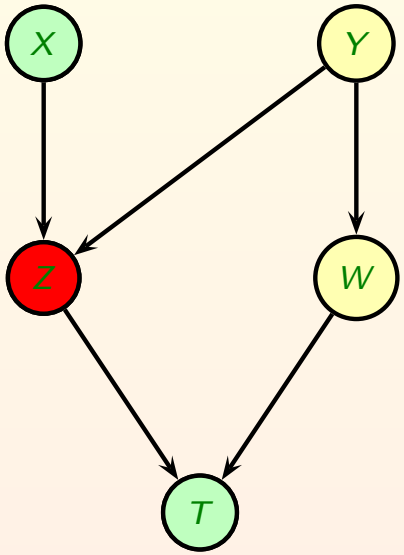


$i = 0$

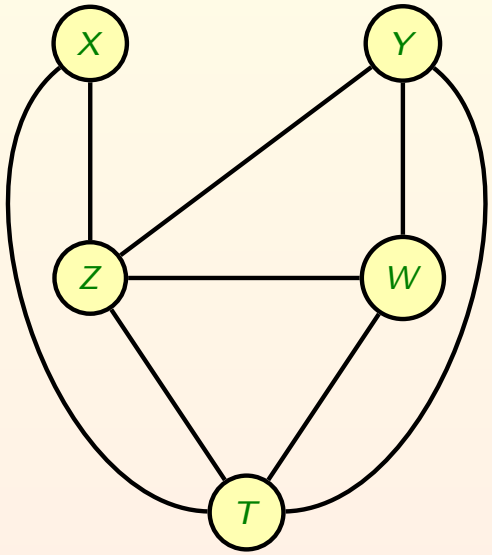


Example

NO

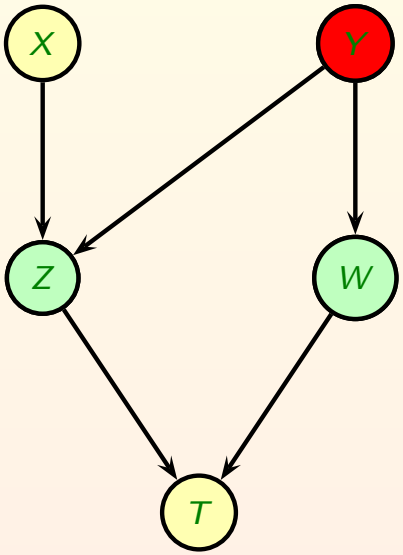


$i = 1$

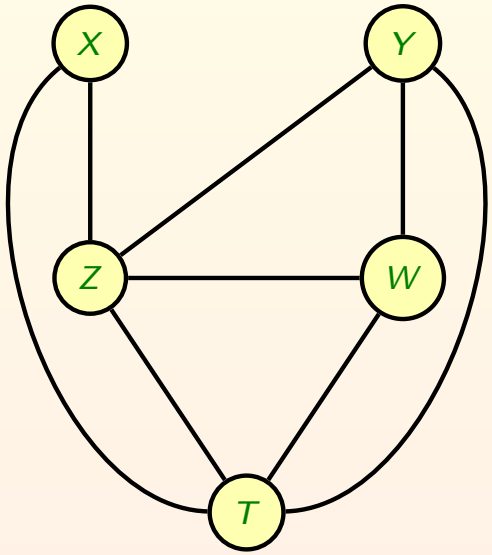


Example

YES

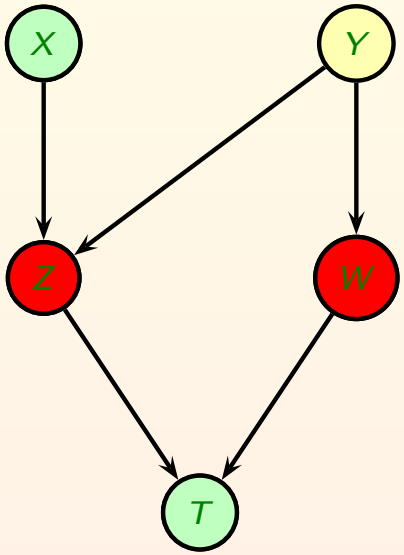


$i = 1$

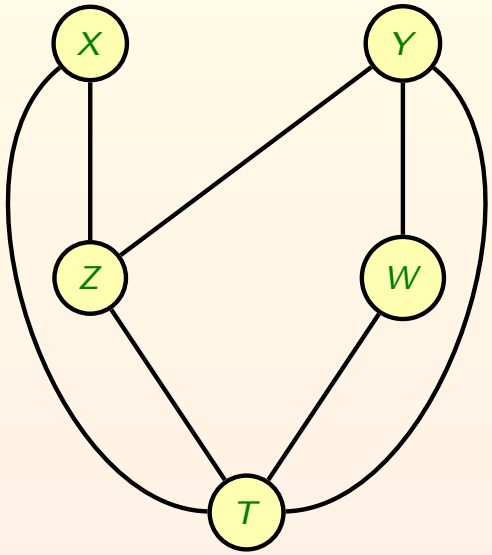


Example

YES



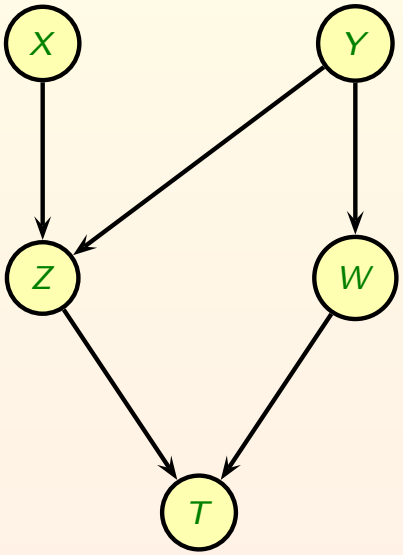
$i = 2$



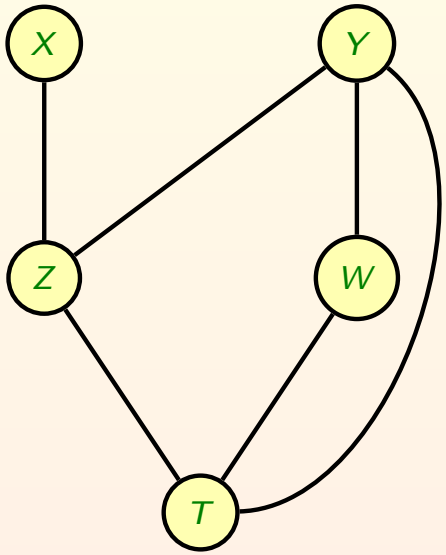
$$S_{Z,W} = \{Y\}$$

Example

YES



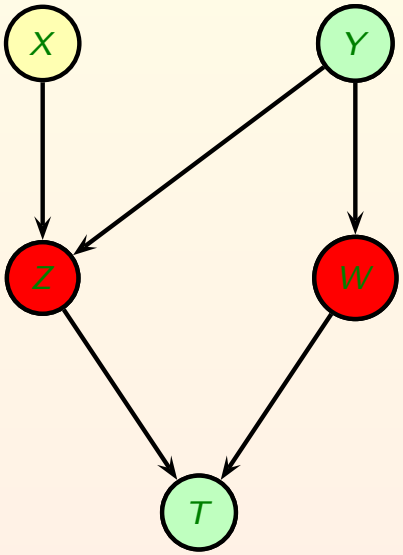
$i = 2$

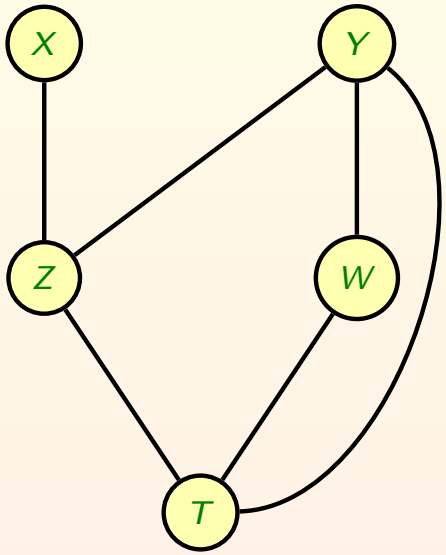


$$S_{X,T} = \{Z, W\}$$

Example

YES

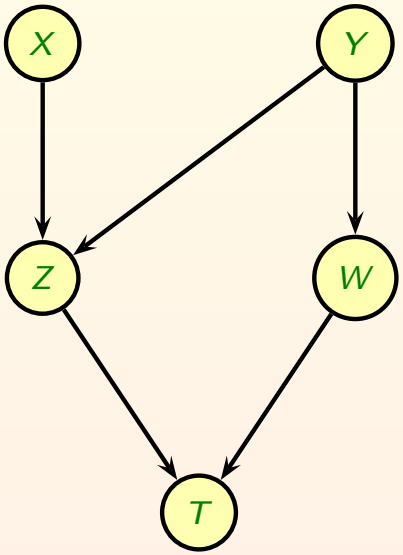


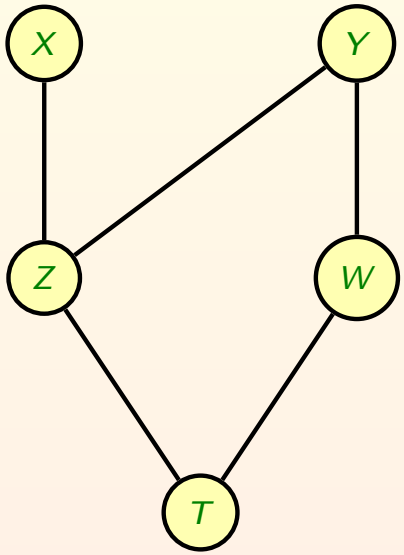


$$S_{Y,T} = \{Z, W\}$$

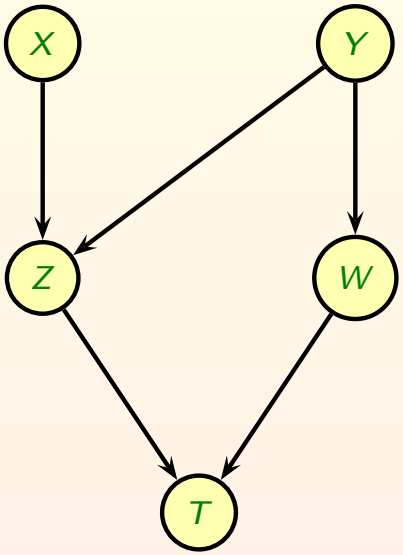
Example

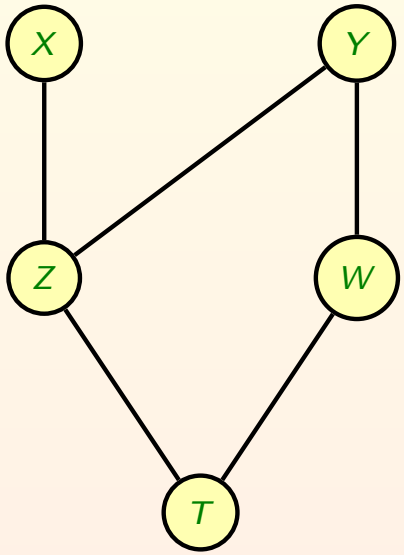
YES





Example





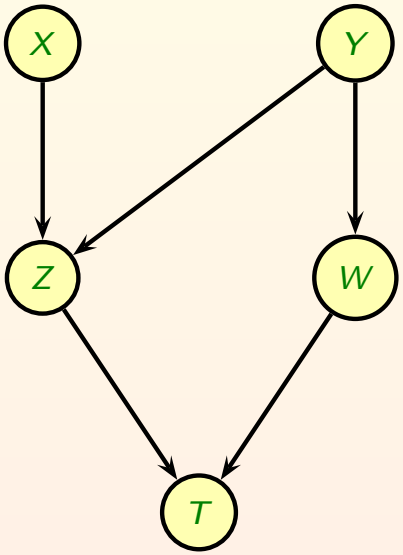
We stop because there is no node with more than 3 adjacent nodes

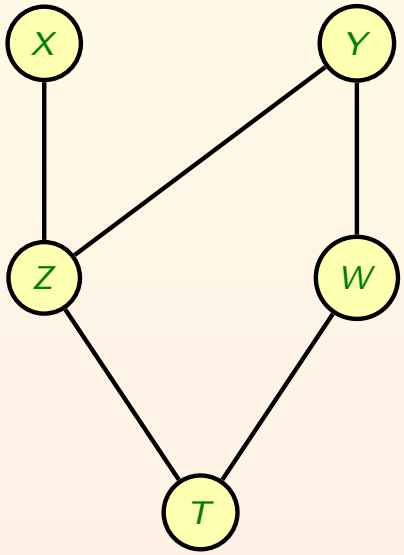
Finding Head-to-Head Links

1. **For each** uncoupled meeting $X - Z - Y$
    2. **If** $Z \notin S_{XY}$
        3. Orient $X - Z - Y$ as $X \rightarrow Z \leftarrow Y$

If a variable is connected with other two variables and it is not in the separator of them, then the arrows have to be head-to-head.

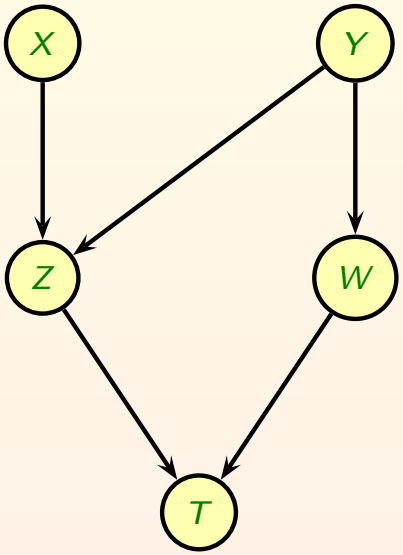
Example

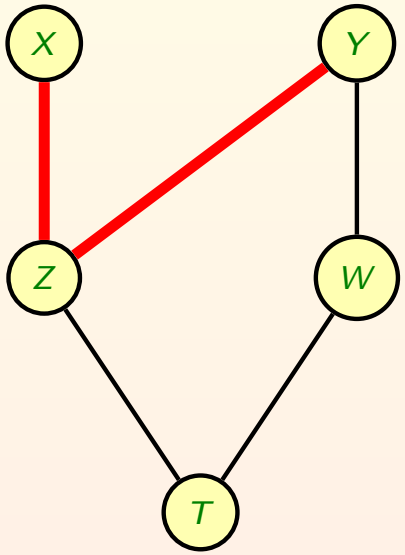




$$S_{X,Y} = \emptyset$$
$$S_{X,W} = \emptyset$$
$$S_{Z,W} = \{Y\}$$
$$S_{X,T} = \{Z, W\}$$
$$S_{Y,T} = \{Z, W\}$$

Example

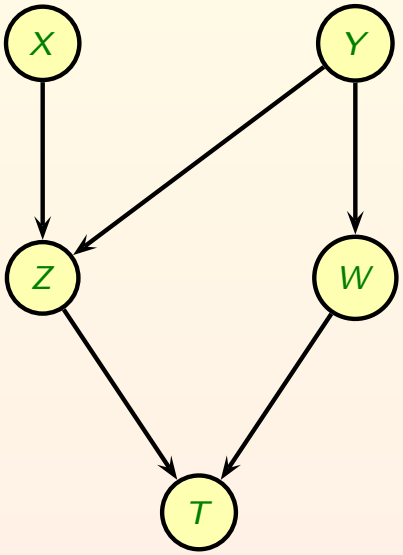


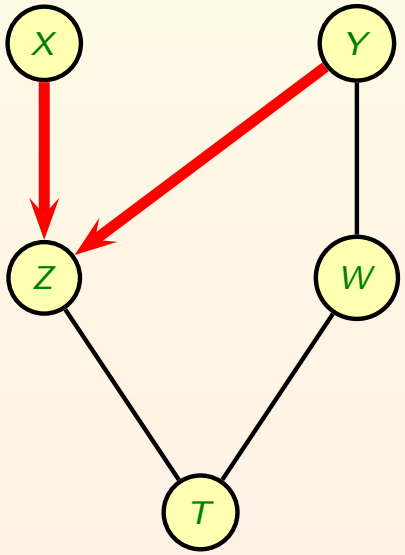


$$S_{X,Y} = \emptyset$$
$$S_{X,W} = \emptyset$$
$$S_{Z,W} = \{Y\}$$
$$S_{X,T} = \{Z, W\}$$
$$S_{Y,T} = \{Z, W\}$$

$$Z \notin S_{X,Y}$$

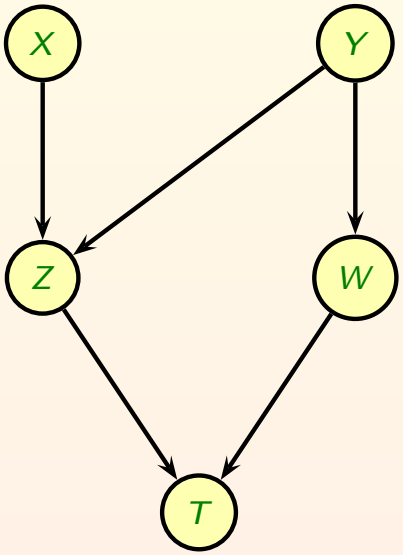
Example

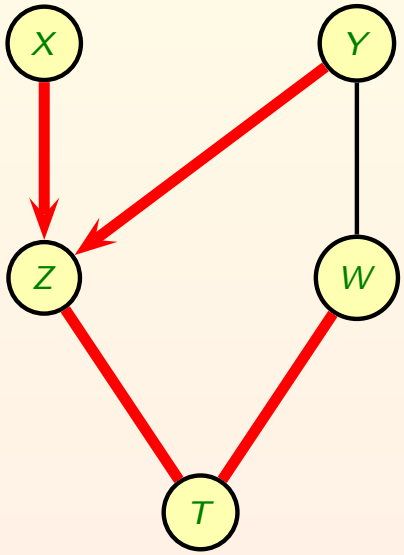




$$S_{X,Y} = \emptyset$$
$$S_{X,W} = \emptyset$$
$$S_{Z,W} = \{Y\}$$
$$S_{X,T} = \{Z, W\}$$
$$S_{Y,T} = \{Z, W\}$$

Example

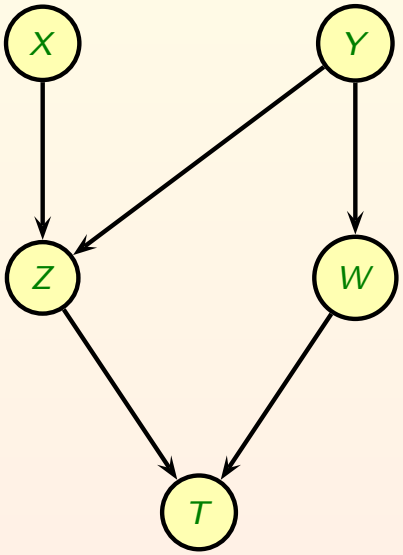


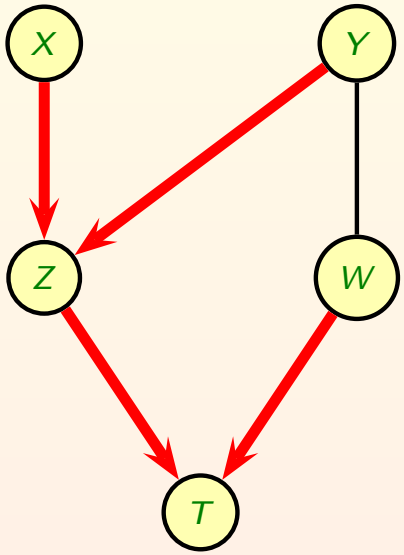


$$S_{X,Y} = \emptyset$$
$$S_{X,W} = \emptyset$$
$$S_{Z,W} = \{Y\}$$
$$S_{X,T} = \{Z, W\}$$
$$S_{Y,T} = \{Z, W\}$$

$$T \notin S_{Z,W}$$

Example





$$S_{X,Y} = \emptyset$$
$$S_{X,W} = \emptyset$$
$$S_{Z,W} = \{Y\}$$
$$S_{X,T} = \{Z, W\}$$
$$S_{Y,T} = \{Z, W\}$$

This finishes the initial partial orientation

More Orientations

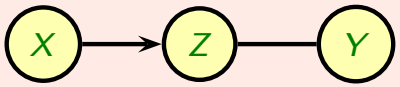
The basic idea is that no new head-to-head links are created and that the DAG condition is preserved.

1. **While** no more edges can be oriented
    2. **For** each uncoupled meeting $X \rightarrow Z - Y$
        3. Orient $Z - Y$ as $Z \rightarrow Y$
    4. **For** each $X - Y$ such that there is a directed path from X to Y
        5. Orient $X - Y$ as $X \rightarrow Y$
    6. **For** each uncoupled meeting $X - Z - Y$ such that $X \rightarrow W, Y \rightarrow W, Z - W$
        7. Orient $Z - W$ as $Z \rightarrow W$

More Orientations

The basic idea is that no new head-to-head links are created and that the DAG condition is preserved.

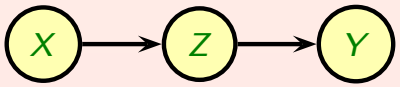
1. **While** no more edges can be oriented
    2. **For** each uncoupled meeting $X \rightarrow Z - Y$
        3. Orient $Z - Y$ as $Z \rightarrow Y$
    4. **For** each $X - Y$ such that there is a directed path from X to Y
        5. Orient $X - Y$ as $X \rightarrow Y$
    6. **For** each uncoupled meeting $X - Z - Y$ such that $X \rightarrow W, Y \rightarrow W, Z - W$
        7. Orient $Z - W$ as $Z \rightarrow W$



More Orientations

The basic idea is that no new head-to-head links are created and that the DAG condition is preserved.

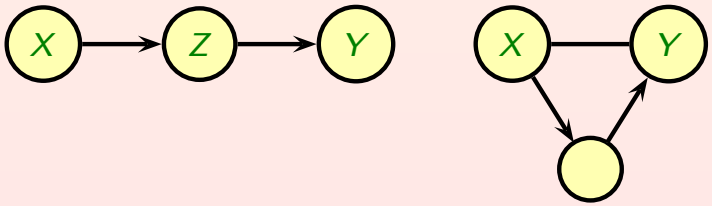
1. **While** no more edges can be oriented
    2. **For** each uncoupled meeting $X \rightarrow Z - Y$
        3. Orient $Z - Y$ as $Z \rightarrow Y$
    4. **For** each $X - Y$ such that there is a directed path from X to Y
        5. Orient $X - Y$ as $X \rightarrow Y$
    6. **For** each uncoupled meeting $X - Z - Y$ such that $X \rightarrow W, Y \rightarrow W, Z - W$
        7. Orient $Z - W$ as $Z \rightarrow W$



More Orientations

The basic idea is that no new head-to-head links are created and that the DAG condition is preserved.

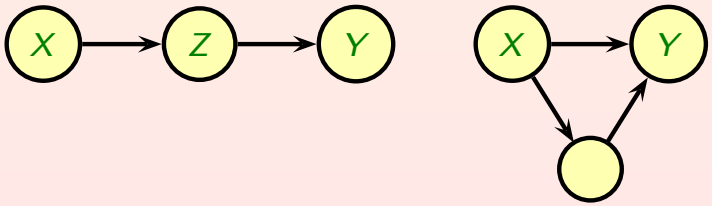
1. **While** no more edges can be oriented
    2. **For** each uncoupled meeting $X \rightarrow Z - Y$
        3. Orient $Z - Y$ as $Z \rightarrow Y$
    4. **For** each $X - Y$ such that there is a directed path from X to Y
        5. Orient $X - Y$ as $X \rightarrow Y$
    6. **For** each uncoupled meeting $X - Z - Y$ such that $X \rightarrow W, Y \rightarrow W, Z - W$
        7. Orient $Z - W$ as $Z \rightarrow W$



More Orientations

The basic idea is that no new head-to-head links are created and that the DAG condition is preserved.

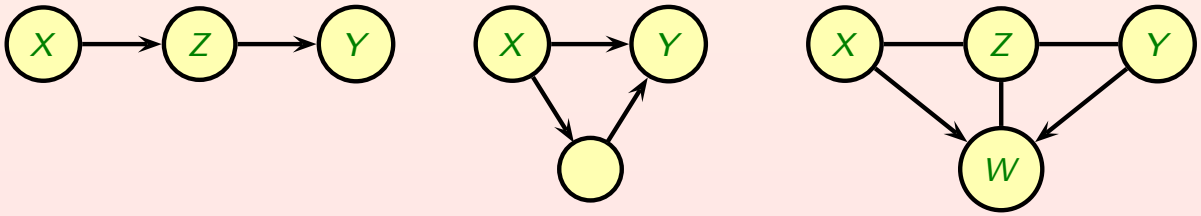
1. **While** no more edges can be oriented
    2. **For** each uncoupled meeting $X \rightarrow Z - Y$
        3. Orient $Z - Y$ as $Z \rightarrow Y$
    4. **For** each $X - Y$ such that there is a directed path from X to Y
        5. Orient $X - Y$ as $X \rightarrow Y$
    6. **For** each uncoupled meeting $X - Z - Y$ such that $X \rightarrow W, Y \rightarrow W, Z - W$
        7. Orient $Z - W$ as $Z \rightarrow W$



More Orientations

The basic idea is that no new head-to-head links are created and that the DAG condition is preserved.

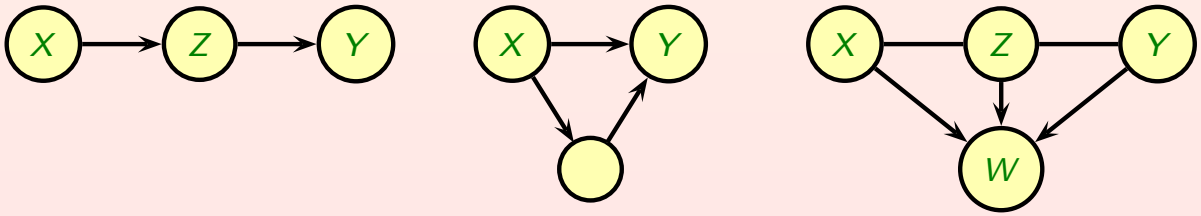
1. **While** no more edges can be oriented
    2. **For** each uncoupled meeting $X \rightarrow Z - Y$
        3. Orient $Z - Y$ as $Z \rightarrow Y$
    4. **For** each $X - Y$ such that there is a directed path from X to Y
        5. Orient $X - Y$ as $X \rightarrow Y$
    6. **For** each uncoupled meeting $X - Z - Y$ such that $X \rightarrow W, Y \rightarrow W, Z - W$
        7. Orient $Z - W$ as $Z \rightarrow W$



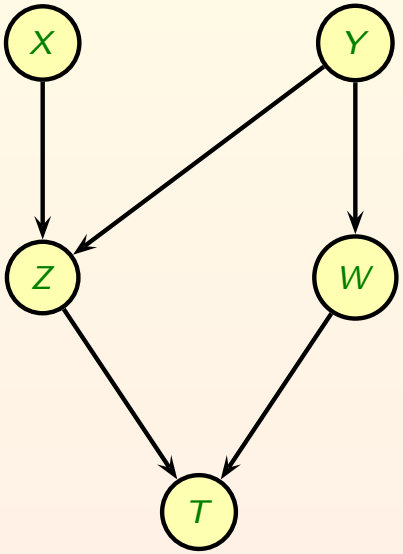
More Orientations

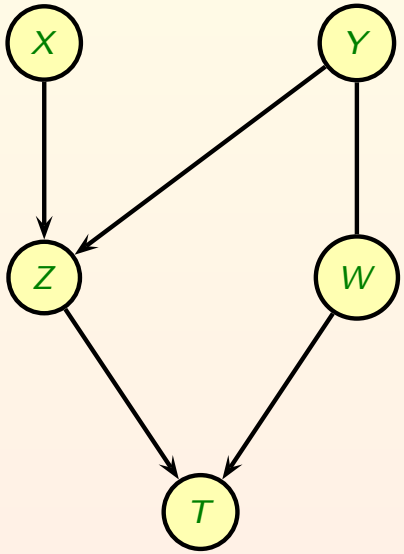
The basic idea is that no new head-to-head links are created and that the DAG condition is preserved.

1. **While** no more edges can be oriented
    2. **For** each uncoupled meeting $X \rightarrow Z - Y$
        3. Orient $Z - Y$ as $Z \rightarrow Y$
    4. **For** each $X - Y$ such that there is a directed path from X to Y
        5. Orient $X - Y$ as $X \rightarrow Y$
    6. **For** each uncoupled meeting $X - Z - Y$ such that $X \rightarrow W, Y \rightarrow W, Z - W$
        7. Orient $Z - W$ as $Z \rightarrow W$



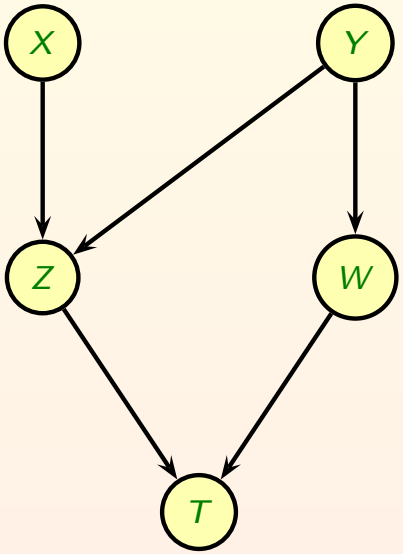
Example

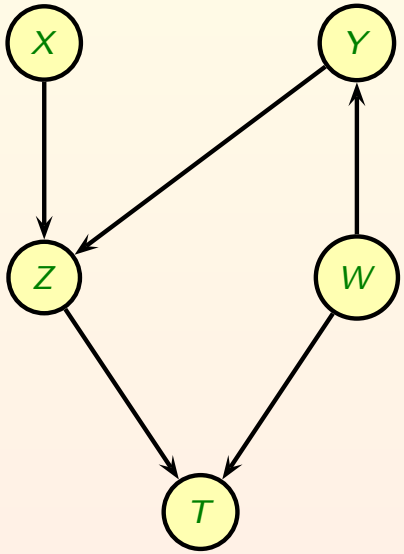




The rest of arcs are oriented on an arbitrary way, keeping the DAG conditioning and not creating head-to-head links.
This may be a cause of counterintuitive head to head links.

Example





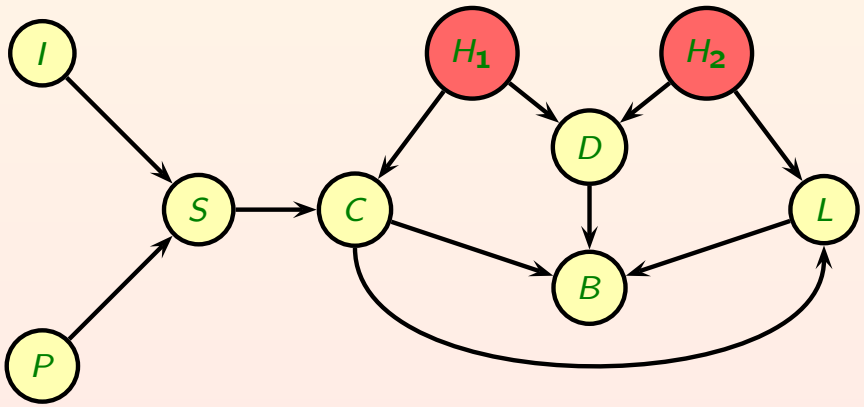
The rest of arcs are oriented on an arbitrary way, keeping the DAG conditioning and not creating head-to-head links.
This may be a cause of counterintuitive head to head links.

Exercice

Consider the following variables

- I Patient's income
- S Patient's smoking history
- P Patient's parents smoking history
- C Patient's level of cilia damage
- D Patient's level of heart disease
- L Patient's lung capacity
- B Patient's level of breathing disfuction

Consider also to hidden (non-observed variables): H_1 (pollution) and H_2 (Patient's genetic); in such a way that the relationships are:



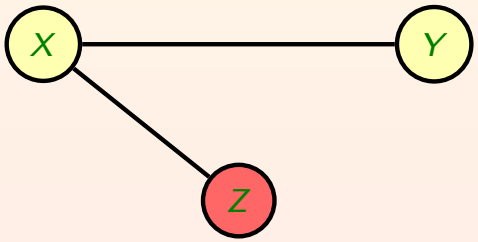
Build the network induced by PC algorithm for the observed variables considering the independence relationships of the network.

NPC Algorithm

Steck, H. (2001). Constrained-Based Structural Learning in Bayesian Networks Using Finite Data Sets, PhD Thesis.

It tries to solve some of the problems of the PC algorithm, mainly related with the test reliability (small data sets).

First, they introduce the concept of necessary path condition. Consider the following simple example:

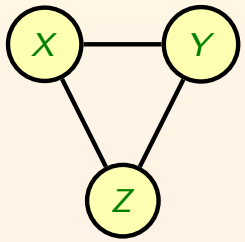


Imagine that we have finished with $i = 0$ independences and we are starting with $i = 1$. The PC algorithm is considering $I(X, Y|Z)$. This is completely unnecessary if tests were correct, but due to a limited set of data we can get *independence* even if this is not theoretically possible.

Ambiguous Situations

NPC algorithm only tests independences conditioned to variables in the path.

Another point is the case of ambiguous situations. It can be the case that for three variables:

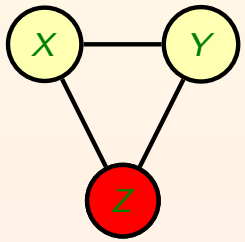


We have that considering unconditional independences none of them is verified, but all the conditional independences to one variable are verified.
So the result can depend on the order in which the tests are carried out.

Ambiguous Situations

NPC algorithm only tests independences conditioned to variables in the path.

Another point is the case of ambiguous situations. It can be the case that for three variables:



We have that considering unconditional independences none of them is verified, but all the conditional independences to one variable are verified.
So the result can depend on the order in which the tests are carried out. For example, if we test the independence of X and Y given Z, we remove the link between X and Y. No more tests are carried out.

Other Features

- In Hugin implementation of NPC, ambiguities are solved by interaction with the user. We worked in this problems publishing some papers in which ambiguities were solved asking to experts.
- Another possibility could be to measure the strength of independence relationships and keep the stronger links. User information can be better, but some times is not available.
- The direction of links is also solved with user interaction.
- In general, it would be good to take advantage of any 'a priori' information the user can give.
- In Hugin you can specify sure or prohibited links (directed or undirected).
- Other possibilities (order relations) could be useful.

Algoritmo Grow-Shrink (GS): Cálculo de la frontera de Markov

Frontera de Markov

Si tenemos un conjunto de variables X e $Y \in X$, la frontera de Markov de Y es el conjunto minimal $B(Y)$ de nodos Z tal que $I(Y, X \setminus (Z \cup \{Y\}) | Z)$

En una red bayesiana: Los padres, más los hijos, más los nodos que tienen un hijo en común con Y.

Algoritmo para calcular $B(Y)$:

- Comienza con $B(Y) = \emptyset$
- Fase ascendente: mientras haya Z tal que no se de la independencia $I(Y, Z | B(Y))$ entonces añadir Z a $B(Y)$.
- Fase descendente: mientras haya $Z \in B(Y)$ tal que se de la independencia $I(Y, Z | B(Y) \setminus \{Z\})$ entonces eliminar Z de $B(Y)$.

El algoritmo GS completo

1. **Calcular las frontereas de Markov** para todos los nodos Y, $B(Y)$.
2. **Calcular la estructura del grafo** con test de independencia, similar a PC, pero considerando solo los nodos de la frontera de Markov. $N(X)$ son los elementos de la frontera que quedan como vecinos directos (padres o hijos) de X.
3. **Orientar los enlaces** Si X es un nodo e Y uno de sus vecinos, si existe $Z \in N(X) \setminus (N(Y) \cup \{Y\})$ tal que Y y Z son dependientes dado $T \cup \{X\}$ para todo $T \subseteq H$, donde H es el más pequeños de los conjuntos $B(Y) \setminus \{X, Z\}$ y $B(Z) \setminus \{X, Y\}$, orientar $Y \rightarrow X$.
4. **Eliminar ciclos**
5. **Invertir enlaces**
6. **Propagar direcciones**

En definitiva: refinar la estrategia del PC calculando una frontera de Markov.

Otros algoritmos en **bnlearn**

- Incremental Association (IAMB) a (Tsamardinos et al., 2003) parecido a GS en el cálculo de la frontera de Markov, pero tratando de hacer más eficiente la fase de crecimiento.
- MMPC and HITON-PC: Los más escalables. Solo calculan el esqueleto.
- HPC (híbrido padres e hijos): Combinación de varios algoritmos que realizan distintas tareas

Todos suponen que las independencias se representan de forma exacta mediante una red bayesiana y son correctos (recuperan una red equivalente) si los tests no se equivocan.

Structural Learning: Score+Search Methods

Serafín Moral
Dpt. Computer Science and Artificial Intelligence
University of Granada, Spain

Structure Learning

There are two basic approaches:

- Independence tests
- Score + Search procedures

We are going to study the Score+Search Methods
Cooper, Herskovits (1992) A Bayesian approach for the induction of probabilistic models from data. *Machine Learning*

Determine a score that has to measure how well a Bayesian network describes a set of data and, at the same time, has to put some penalty in model complexity.

Pros and Cons

- Can make compromises (ambiguous regions)
- Well justified
- Can take into account simplifications of conditional probabilities.
- It is computationally difficult and then the solutions are almost always approximate, however in the last years there is a lot of progress with exact A^* algorithms for mediun size problems.

The Likelihood Score

If we have a data set D and a Graph G with parameters θ, then we can compute the likelihood of this graph and parameters given the data:

$$L(G,\theta:D)=\prod_{i=1}^{n}\prod_{j=1}^{q_i}\prod_{k=1}^{r_i}\theta_{ijk}{}^{N_{ijk}}$$

Usually, the log of this value is considered (the log-likelihood):

$$\log L(G,\theta:D)=\sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i}N_{ijk}\log\theta_{ijk}$$

If we consider that the sample is a probability distribution, this value is the entropy of the data (that is constant) minus the Kulback distance between the sample and the network.

Likelihood Score: Computation

We can maximize likelihood by first maximizing the parameters:

$$\hat{\theta}_{ijk} = \frac{N_{ijk}}{N_{ij}}$$

where $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$.
And then,

$$\log L(G, \hat{\theta} : D) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \log \frac{N_{ijk}}{N_{ij}}$$

As larger the likelihood the minimum the distance of the network to the sample. The problem is that the minimum distance is obtained for the original sample. So the Bayesian network will overfit the data (will reproduce exactly the sample).

Avoiding Overfitting

- Restricting the search space (example: learning trees)
- Adding a penalty for model complexity. For example the minimum length description principle. To minimize:

  Descr. of data given model + Descr. of the model
- The Bayesian methods that instead of computing the maximum likelihood parameters, average with respect to an 'a priori' distribution of the parameters.

Penalized Maximum Likelihood

The maximum likelihood criterion only consider the fitness to data, but does not add a penalty for the model complexity.
The penalized likelihood scores have the form:

$$\sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} N_{ijk}\log\frac{N_{ijk}}{N_{ij}} - Pen(N)Dim(G)$$

- $Dim(S)=\sum_{i=1}^{n} q_i(r_i-1)$ is the model dimension
- $Pen(N)$ is a non negative penalty (N sample size)
  - $Pen(N)=1$. Akaike's Information Criterion (AIC)
  - $Pen(N)=1/2\log N$ Bayesian Information Criterion (BIC). It is compatible with the minimum length description principle.

The Bayesian Score

Applying Bayes Rule:

$$P(G|D) = \frac{P(D|G).P(G)}{P(D)}$$

where

- $P(D)$ is the probability of data, which is constant.
- $P(G)$ is the 'a priori' probability of graph, which can be uniform or incorporate any other information that the experts can provide.
- $P(D|G)$ is called the marginal likelihood

If we assume $P(G)$ constant, then maximizing the 'a posteriori' information is equivalent to maximizing the marginal likelihood.

The Marginal Likelihood

We can express the marginal likelihood as:

$$P(D|G) = \int P(D|\theta, G).P(\theta|G)d\theta$$

A graph with more parameters implies a lower value: the probability of two identical values is greater than the product of the probability of the two values.

Marginal Likelihood Computation

- Under local and global parameter independence
- If 'a priori' distributions are Dirichlet
- If the data are all complete

Then, the marginal likelihood is

$$P(D|G) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(N_{ij} + \alpha_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + N_{ijk})}{\Gamma(\alpha_{ijk})}$$

Where each θ_{ij} is a Dirichlet $D(\alpha_{ij1}, \ldots, \alpha_{ijr_i})$ and $\alpha_{ij} = \sum_{k=1}^{r_i} \alpha_{ijk}$.

Normally, the logarithm of the marginal likelihood is considered.

Parameters

The K2 metric is obtained when all the Dirichlet parameters are set to one:

$$P(D|G) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} (N_{ijk})!$$

With this score, equivalent graphs (they represent the same independencies) do not have the same score.
An equivalent score can be obtained by considering a global sample size S and then selecting the parameters for all the conditional probabilities according to the expression:

$$\alpha_{ijk} = \frac{S}{r_i . q_i}$$

Example

Neapolitan (pages 438-40).
Study about college students married by age 30. We want to test whether there is correlation between college graduation and getting divorced.
Variables:

- X_1 (1: graduate, 2: do not graduate)
- X_2 (1: divorced by 50, 2: do not divorce by 50)

We have two possible network structures:

Assume that we compute the equivalent sample size with $S = 4$. 'a priori' probabilities are D(2,2) and conditional probabilities are D(1,1).

Data

Suppose that we have the data in the following table:

Case	1	2	3	4	5	6	7	8
X_1	1	1	1	2	1	2	1	2
X_2	1	2	1	2	1	1	1	2

$$P(D|G_1) = \left(\frac{\Gamma(4)}{\Gamma(4+8)}\frac{\Gamma(2+5)\Gamma(2+3)}{\Gamma(2)\Gamma(2)}\right)\left(\frac{\Gamma(2)}{\Gamma(2+5)}\frac{\Gamma(1+4)\Gamma(1+1)}{\Gamma(1)\Gamma(1)}\right)\left(\frac{\Gamma(2)}{\Gamma(2+3)}\frac{\Gamma(1+1)\Gamma(1+2)}{\Gamma(1)\Gamma(1)}\right) = 7.2150\times 10^{-6}$$

$$P(D|G_2) = \left(\frac{\Gamma(4)}{\Gamma(4+8)}\frac{\Gamma(2+5)\Gamma(2+3)}{\Gamma(2)\Gamma(2)}\right)\left(\frac{\Gamma(4)}{\Gamma(4+8)}\frac{\Gamma(2+5)\Gamma(2+3)}{\Gamma(2)\Gamma(2)}\right) = 6.7465\times 10^{-6}$$

The first model (dependence) has more score (but little difference).

Exercices

Compute the scores for the two models with the following data:

Case	1	2	3	4	5	6	7	8
X_1	1	1	1	1	2	2	2	2
X_2	1	1	1	1	2	2	2	2

And with the following data:

Case	1	2	3	4	5	6	7	8
X_1	1	1	1	1	2	2	2	2
X_2	1	1	2	2	1	1	2	2

Searching an Optimal Graph

The space of possible directed acyclic graphs for n variables is huge:

$$\begin{cases} f(0) & = 1; \\ f(1) & = 1; \\ f(n) & = \sum_{i=1}^{n} (-1)^{i+1} . \binom{n}{i} 2^{i(n-i)} d(n-i) \end{cases}$$

Heuristics methods are necessary to optimize the score in this space.

The K2 Algorithm

- It is assumed an ordering of the nodes
- It is also assumed a maximum number of parents for each node
- Initial node does not have parents
- Then for each node, it starts with the empty set of parents and then add as parent the node that preceding it in the order, produces a bigger increasing in the score.
- It continues adding parents while the score increases and the number of parents in less than the maximum.

Decomposability

The key property of the score that makes this algorithm feasible is decomposability.
This score,

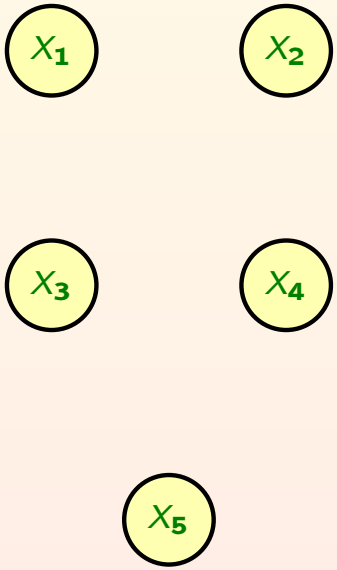
$$P(D|G) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(N_{ij} + \alpha_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + N_{ijk})}{\Gamma(\alpha_{ijk})}$$

or the logarithm of it:

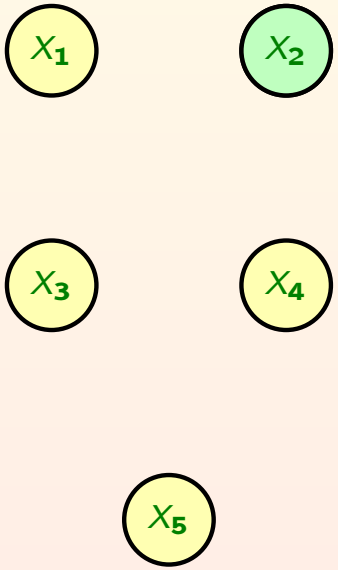
$$\log P(D|G) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \log \frac{\Gamma(\alpha_{ij})}{\Gamma(N_{ij} + \alpha_{ij})} \sum_{k=1}^{r_i} \log \frac{\Gamma(\alpha_{ijk} + N_{ijk})}{\Gamma(\alpha_{ijk})}$$

is that local changes imply only local computations.
The log marginal likelihood is a sum of functions depending of nodes and its parents. If only one node changes its parents, then only the part of the score corresponding to this node, has to be recomputed.

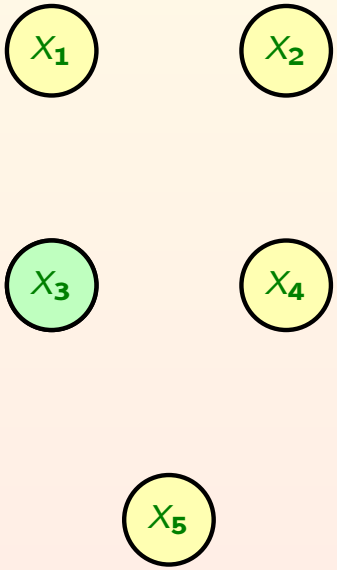
Example



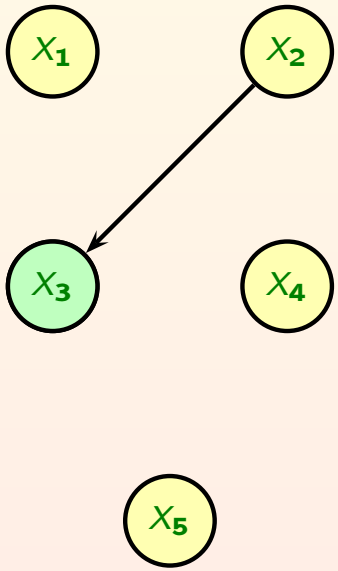
Example



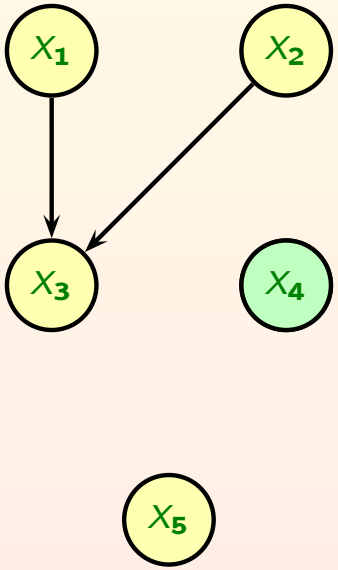
Example



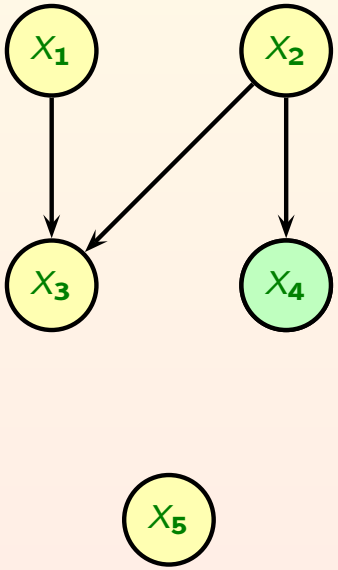
Example



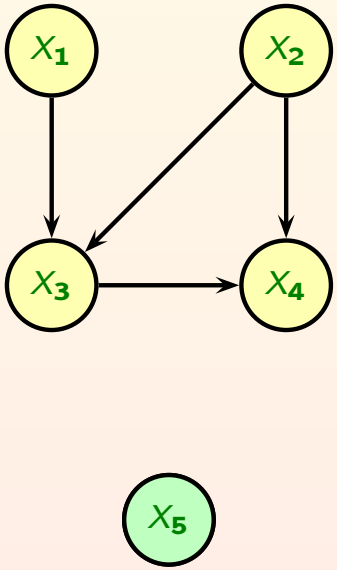
Example



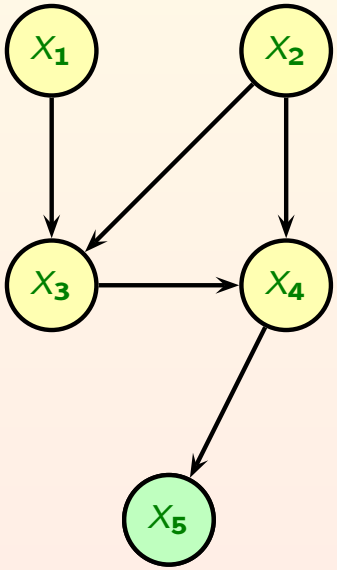
Example



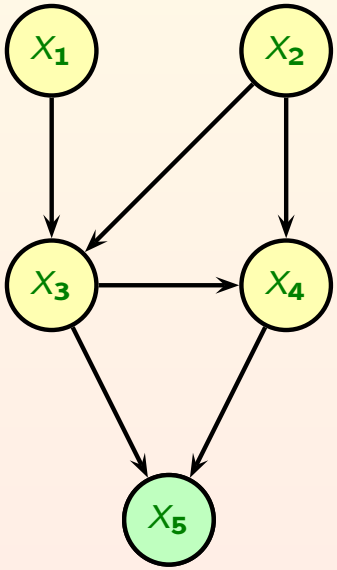
Example



Example

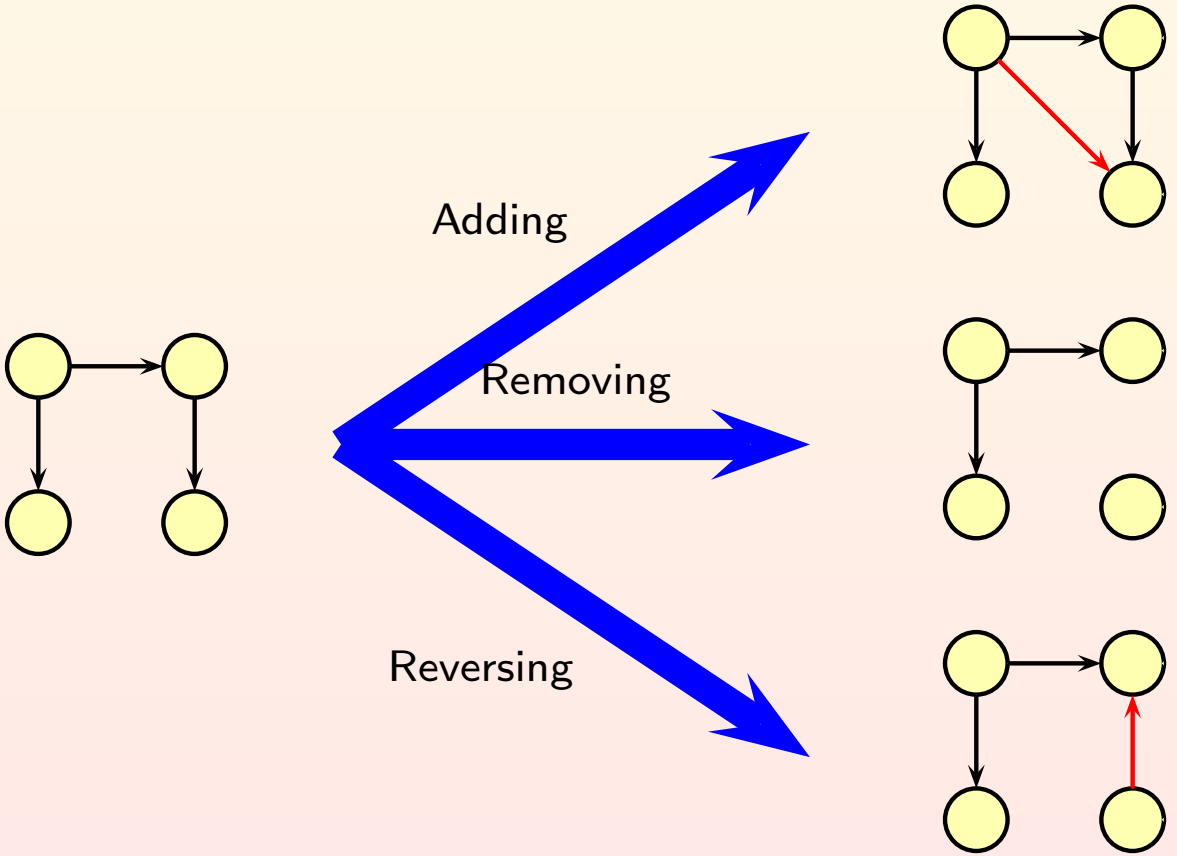


Example



Heuristic Search

More sophisticated search procedures have been considered. In general, they start with some Bayesian network (empty, K2, tree). The search space is explored with some basic movements. Typical ones are:



Exploring the Space

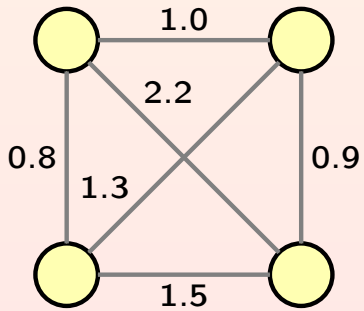
Different methods have been using to explore the space (NP-hard problem):

- Hill climbing
- Simulated annealing
- Genetics algorithms
- Variable neighborhood search
- Búsqueda tabú (en **bnlearn**).

Searching for Trees

Searching for trees structures (each node at most one parent) can be done in polynomial time by means of the Chow-Liu algorithm.

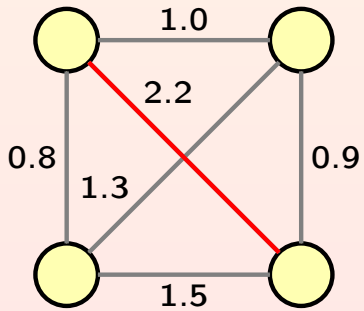
1. Construct a graph will all the nodes without arcs
2. For each pair X_i, X_j comp. weight $W(i,j) = Score(X_j|X_i) - Score(X_j)$
3. Find a tree with maximum weight
   (Kruskal algorithm)



Searching for Trees

Searching for trees structures (each node at most one parent) can be done in polynomial time by means of the Chow-Liu algorithm.

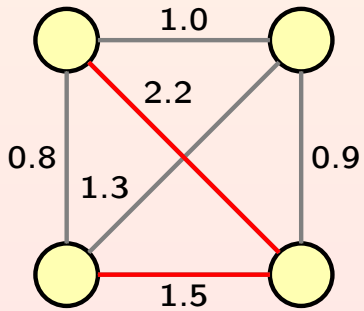
1. Construct a graph will all the nodes without arcs
2. For each pair X_i, X_j comp. weight $W(i,j) = Score(X_j|X_i) - Score(X_j)$
3. Find a tree with maximum weight
   (Kruskal algorithm)



Searching for Trees

Searching for trees structures (each node at most one parent) can be done in polynomial time by means of the Chow-Liu algorithm.

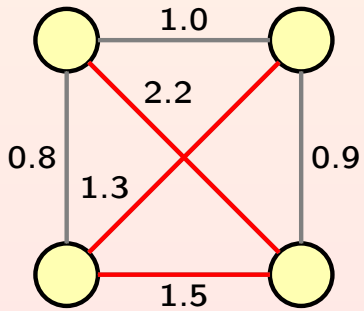
1. Construct a graph will all the nodes without arcs
2. For each pair X_i, X_j comp. weight $W(i,j) = Score(X_j|X_i) - Score(X_j)$
3. Find a tree with maximum weight
   (Kruskal algorithm)



Searching for Trees

Searching for trees structures (each node at most one parent) can be done in polynomial time by means of the Chow-Liu algorithm.

1. Construct a graph will all the nodes without arcs
2. For each pair X_i, X_j comp. weight $W(i,j) = Score(X_j|X_i) - Score(X_j)$
3. Find a tree with maximum weight
   (Kruskal algorithm)



Tree Construction

- Usually, it considers the mutual information as weight (likelihood score). The model complexity is controlled by the structure.
- It is efficient to build and sometimes provides good results
- Sometimes misses some important links, and it can add non-necessary links.

Model Selection-Model Averaging

In general, given some data we have selected an only model and all the predictions are done according to this model.
In a pure Bayesian approach to estimate the probability of an event E, we should average on all the possible models according to their 'a posteriori' probability given the data.

$$P(E|D) = \sum_{G} P(E|D,G)P(G|D)$$

In general, this is very difficult to compute and most of the models have very small probability. It is more common to select a set of more probable models $\mathcal{G}$ and then compute:

$$P(E|D) = \sum_{G \in \mathcal{G}} P(E|D,G)P(G|D)$$

Summary

- There are well founded scores balancing data fitting and model complexity.
- Searching for an optimal model is difficult (NP-hard) if we allow 2 parents per node.
- Arrows direction is not guaranteed (with more reasons when there are hidden variables).
- Expert knowledge could be integrated in the 'a priori' search function (presence of links, orders, etc.)

Métodos Híbridos: tests + scores

The Sparse Candidate Algorithm and MMHC:

1. Elegir G, usualment vacío.
2. Repetir los siguientes pasos hasta convergencia:
   1. restrict: Selecciona un conjunto C_i de candidatos a padre para X_i que debe de incluir los padres actuales en G.
   2. maximise: calcular el grafo G^* que maximiza un score eligiendo los padres de X_i entre C_i.
   3. $G \leftarrow G^*$.
3. Return G

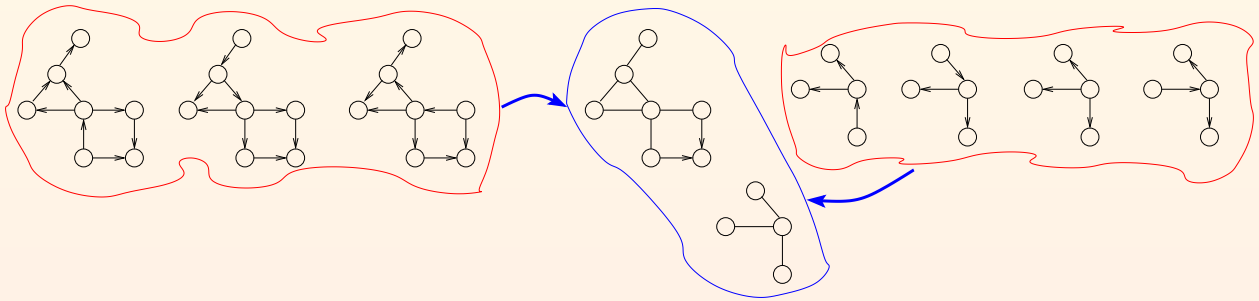
If we iterate only once, using MMPC for the restrict phase and hill-climbing for the maximise phase we obtain the Max-Min Hill-Climbing (MMHC) algorithm as a particular case.

Espacios de Búsqueda

- El espacio de búsqueda más habitual es el de los grafos dirigidos acíclicos (DAGs).

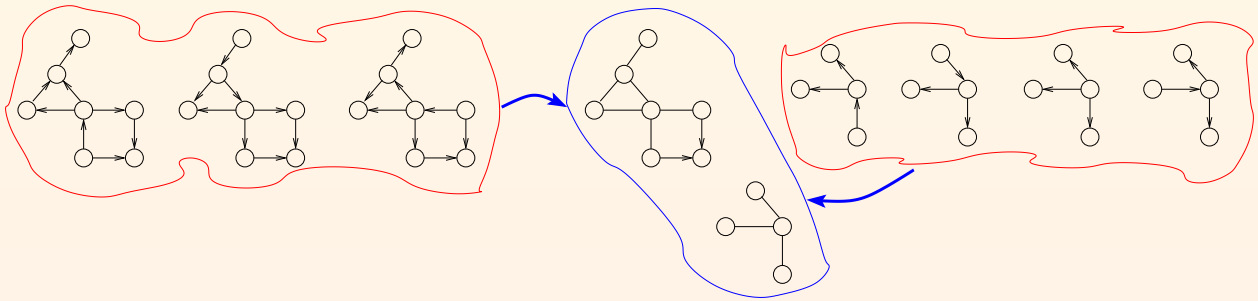
Espacios de Búsqueda

- El espacio de búsqueda más habitual es el de los grafos dirigidos acíclicos (DAGs).
- Espacios de clases de equivalencia de DAGs (grafos que representan las mismas relaciones de (in)dependencia, grafos esenciales, RPDAGs,...)



Espacios de Búsqueda

- El espacio de búsqueda más habitual es el de los grafos dirigidos acíclicos (DAGs).
- Espacios de clases de equivalencia de DAGs (grafos que representan las mismas relaciones de (in)dependencia, grafos esenciales, RPDAGs,...)

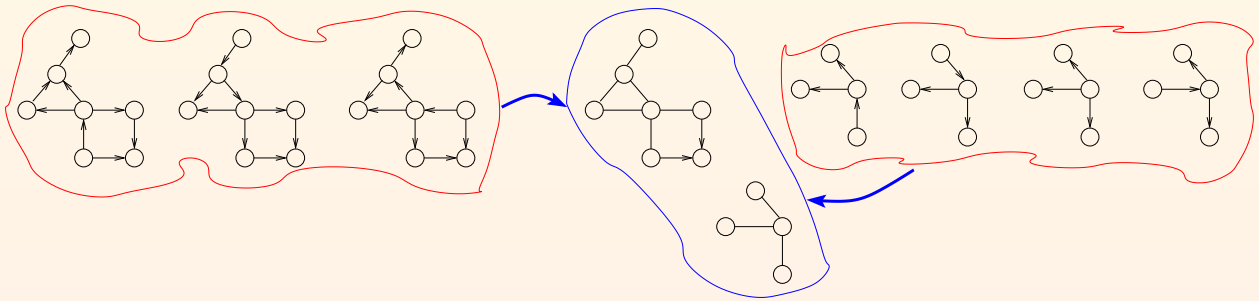


- El espacio de las ordenaciones entre variables (con una búsqueda secundaria en el espacio de los DAGs compatibles con el orden)

$R < S < C < H < L$, $S < C < R < H < L$, $L < C < H < R < S$, ... ($n!$)

Espacios de Búsqueda

- El espacio de búsqueda más habitual es el de los grafos dirigidos acíclicos (DAGs).
- Espacios de clases de equivalencia de DAGs (grafos que representan las mismas relaciones de (in)dependencia, grafos esenciales, RPDAGs,...)

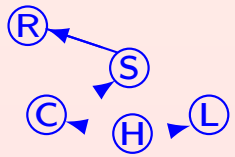


- El espacio de las ordenaciones entre variables (con una búsqueda secundaria en el espacio de los DAGs compatibles con el orden)

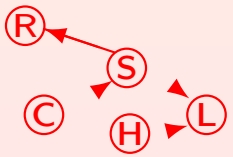
$R < S < C < H < L$, $S < C < R < H < L$, $L < C < H < R < S$, ... ($n!$)

Si $H < C < L < S < R$ entonces

válido



inválido



Structure Learning with Incomplete Data

- PC algorithm can be used, without making use of missing data.
- Friedman has developed and structural EM algorithm
- There is also some stochastic algorithms
- An usual approach is to use approximations. For example, the BIC score:

$$BIC = \log L(G|D, \hat{\theta}) - 1/2 \log(N) Dim(G)$$

Where $\hat{\theta}$ are the maximum likelihood parameters computed with the EM algorithm.

Structure Learning with Incomplete Data

- PC algorithm can be used, without making use of missing data.
- Friedman has developed and structural EM algorithm
- There is also some stochastic algorithms
- An usual approach is to use approximations. For example, the BIC score:

$$BIC = \log L(G|D, \hat{\theta}) - 1/2 \log(N) Dim(G)$$

Where $\hat{\theta}$ are the maximum likelihood parameters computed with the EM algorithm.

Existen también métodos basados en el score bayesiano y en métodos variacionales (usando el ELBO). Todo el problema es que la evaluación de los candidatos es lenta. por eso los métodos es basan en: seleccionar pocos candidatos y evaluarlos aprovechando las evaluaciones anteriores.