SYSM 6304: Risk and Decision Analysis
Lecture 5: Methods of Risk Analysis

M. Vidyasagar

Cecil & Ida Green Chair
The University of Texas at Dallas
Email: M.Vidyasagar@utdallas.edu

October 17, 2015

Outline

Outline

Some Popular Methods of Risk Analysis

Basic Premise: Bad outcomes arise due to *multiple failures,* not just one failure.

So to analyze the *cascading effects of multiple failures,* various approaches are used.

- Event trees
- Fault trees
- Decision trees
- Bayesian networks

All of them represent the problem as a *weighted directed graph,* where individual nodes represent outcomes of random variables and edge weights represent probabilities.

For the sake of completeness, we also review historical methods.

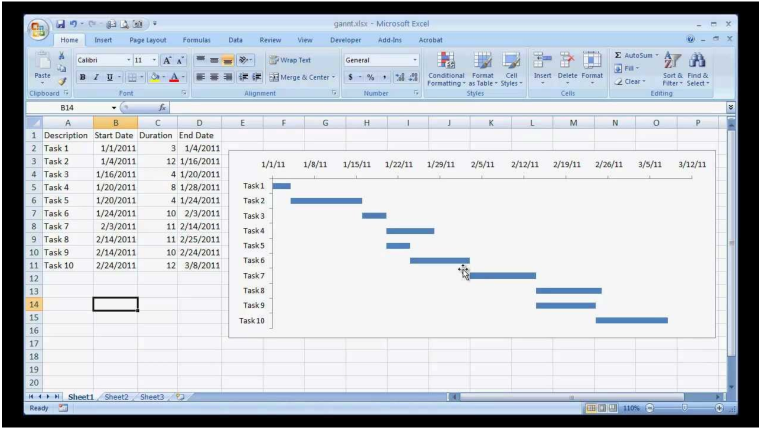
Outline

Outline

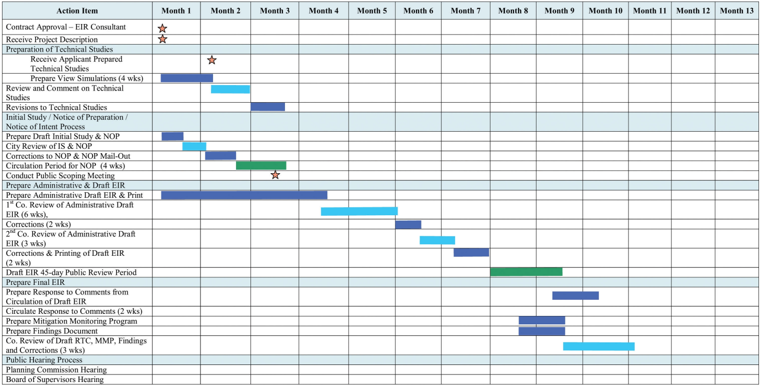
The Gantt Chart

The Gantt chart was invented by Henry Gantt in the 1910's. It represents each activity in a project as a horizontal bar, with start and end times indicated, as shown on the next slide.

Example of a Gantt Chart



Another Example of a Gantt Chart



Pros and Cons of Gantt Chart

Advantage: Preparing a Gantt chart forces you to think through all the work components and to make a proper schedule.

Disadvantages:

- It cannot capture *sequential dependence*, i.e. Step A must be completed before Step B, etc.
- No scope whatsoever for 'randomness' (or variations) in schedule.

Nevertheless, it was a very useful concept in its day!

Outline

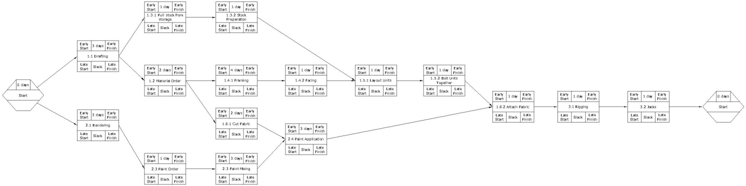
PERT Charts

PERT = Project Evaluation Review Technique

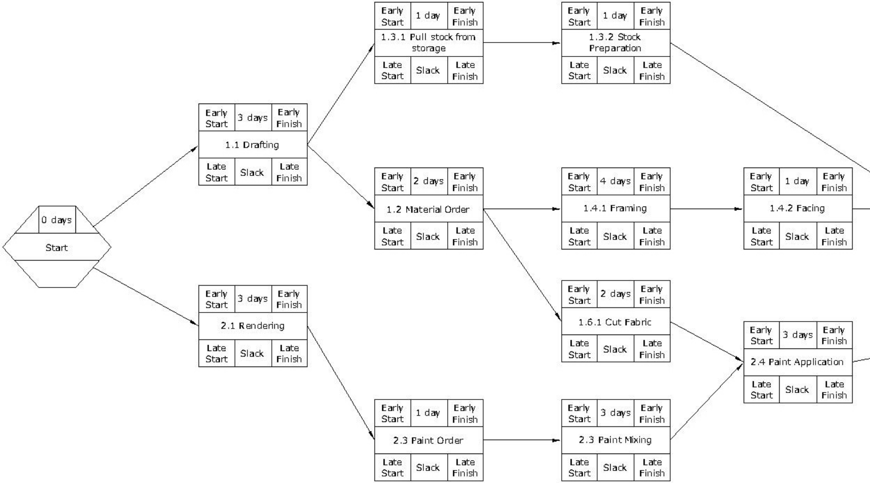
Starting point: Break down overall project into a sequence of sub-projects.

Graphical representation also captures *sequencing information* – a vast improvement over Gantt charts!

PERT Chart Example



PERT Chart Example (Zoom-In)



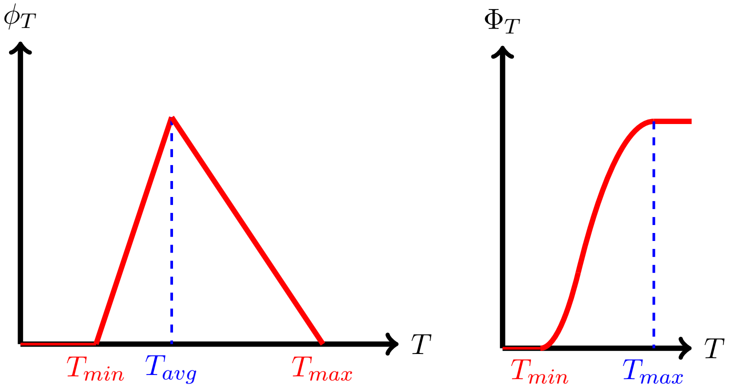
PERT Technique

PERT is not just about capturing sequential information. For each stage, 'experts' were also asked to estimate the maximum time (or cost), minimum time, and average time. Then a simple distribution (usually a 'triangular' distribution' shown on next slide) is fit to the time to complete that stage.

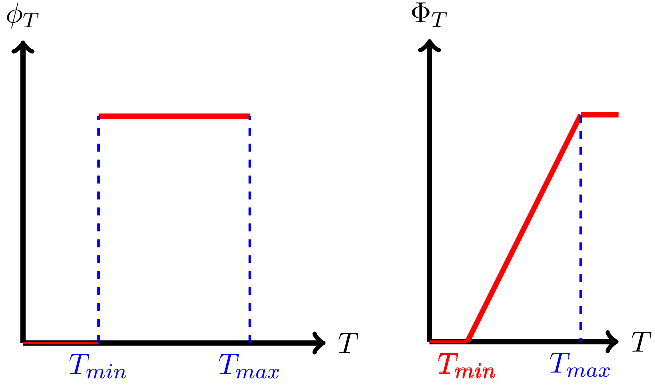
If only minimum and maximum times are available, one can also use a 'uniform' probability distribution (shown in next after next slide).

Graphical structure makes it easy to combine individual estimates into an overall estimate.

Triangular Distributions Used in PERT



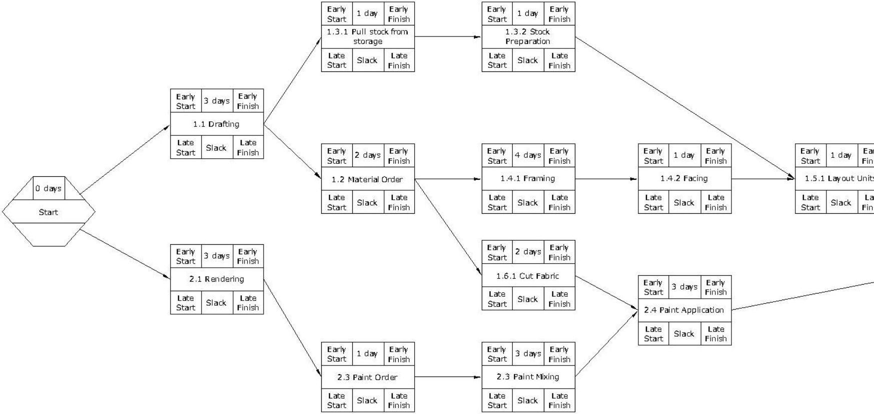
Uniform Distributions Used in PERT



Limitations of PERT

- PERT was evolved for an era when computation was difficult! Today we need not worry about doing complex computations!
- PERT cannot take into account *dependence among different steps* (dependence among different random variables). In other words, PERT assumes that all random variables are independent.

Possibility of Dependence Among Random Variables



Outline

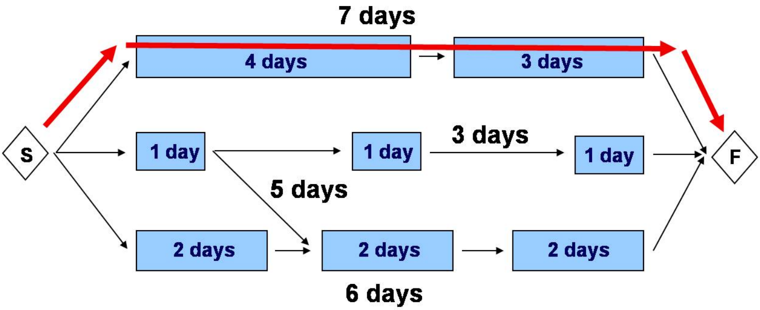
Critical Path Method (CPM)

In this method, one again lays out the components of the overall project, and works out the average (expected) time needed to complete each task.

The longest path through the graph is the **critical path**, because there is no slack in it at all.

Ergo: Pay attention to the critical path and don't worry about the rest (until they become critical!).

Critical Path Illustration



The top path is the critical path.

Outline

Basics of Graphical Representations

A **graph** consists of both "nodes" and "edges" between nodes. A graph is **directed** if the edges have directions, and **undirected** otherwise. A graph is **weighted** if the edges have weights (usually positive) associated with them, and **unweighted** otherwise.

Acyclic vs. Cyclic Graphs

A directed graph (weighted or unweighted) is said to be **connected** if there is a path between every pair of nodes, ignoring the direction of the edges.

A directed graph is said to be **acyclic** if there is no closed path from any node back to itself.

In any acyclic graph, there is at least one "source" node, without any incoming edges, an at least one "sink" node, without any outgoing edges.

In all of the methods studied, source nodes represent the initiating events while sink nodes represent outcomes.

Limitations of Graphical Methods

All of the methods discussed here work well in *relatively simple* situations, though some of them (fault-trees, Bayesian networks) can be extended to more complex situations.

All of these methods presuppose that

- The overall graph is "acyclic," that is, "no loops" in the graph.
- Only "one-step memory" is permitted, if at all. Often r.v.s are assumed to be independent.

For more complicated situations, *Monte Carlo simulation* would be best.

Outline

Outline

Event Trees: Definition

- Often an "outcome" is very simple: success vs. failure, safe operation vs. danger, etc. Often it is also binary.
- But the *path* to the simple (or binary) outcome is a cascade of multiple events.
- Event trees are based on modeling the final outcome as the cascade of multiple events, where
  - The events occur in a well-defined sequence of precedence.
  - Each event has two or more outcomes, with a known (or estimated) probability.
- **Objective:** Using the probabilities of individual events, estimate the probabilities of the various outcomes.

Constructing an Event Tree

- Identify an initiating event of interest.
- Identify
  - All events that are supposed to follow the initiating event
  - The possible outcomes of each event, and
  - The probabilities of the various possible outcomes.
- Repeat.

A Simple Example: Fire Prevention System

A fire prevention system in a private home consists of three components operating sequentially:

- Smoke detector (SD)
- Sprinkler switch (SS)
- Water pump (WP)

If any of the components fails, the overall system fails.

A Simple Example: Fire Prevention System (Cont'd)

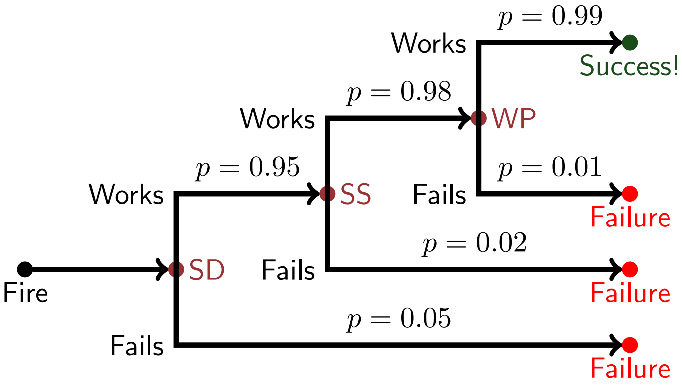
Let us assign failure probabilities to each of the three components, as follows:

$$p_{\mathrm{SD}} = 0.05, p_{\mathrm{SS}} = 0.02, p_{\mathrm{WP}} = 0.01.$$

Let us assume also that the failures of the three components are independent events.

This can be depicted by a very simple event tree (next slide).

Event Tree of Fire Prevention System



Analysis of Event Tree

There are four paths in all, out of which three lead to failure and only one leads to success.

The probability of each path is the product of the probabilities of the individual events (as events are assumed to be independent).

Therefore the probability of success is

$$p_S = 0.95 \times 0.98 \times 0.99 = 0.92169,$$

while the probability of failure is the *sum of the probabilities* of the three paths:

$$\begin{aligned} p_F &= (0.95 \times 0.98 \times 0.01) + (0.95 \times 0.02) + 0.05 \\ &= 0.07831. \end{aligned}$$

Some Common Erroneous Statements

The literature contains some erroneous statements about event trees.

In an event tree, every node can have only two outgoing edges (i.e., every event can have only two possible outcomes).

This is incorrect. It is permissible for a node to have more than two outgoing edges. But the weights (probabilities) of all outgoing edges must, of course, add up to one.

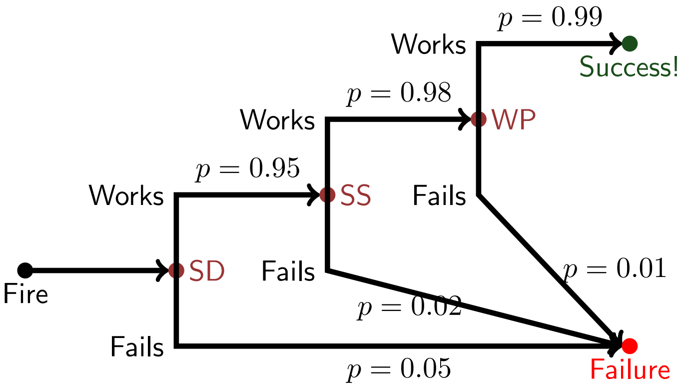
Some Common Erroneous Statements (Cont'd)

In an event tree, every node can have only one incoming edge (i.e., every event can have only one possible way to happen).

This too is incorrect. It is permissible to have multiple edges pointing into one node.

In fact the previous example can be redrawn as shown in the next slide.

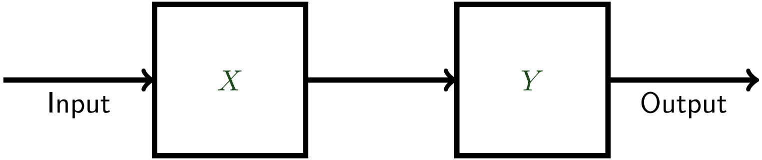
Event Tree of Fire Prevention System (Redrawn)



Dependent Events Example

The events at each node need not also be independent.

Consider the following two-stage processing example:



The total processing is of course $Z = X + Y$.

Suppose X takes values in $\{7, 8, 9\}$ while Y takes values in $\{4, 5, 6\}$. So Z takes values in $\{11, \ldots, 15\}$.

Dependent Events Example (Cont'd)

Suppose the joint probability distribution of X and Y is given by

		Y			
		4	5	6	
	7	0.18	0.06	0.06	0.30
X	8	0.04	0.24	0.12	0.40
	9	0.06	0.09	0.15	0.30
		0.28	0.39	0.33	1.00

Because X and Y are *not independent,* we cannot compute the probability distribution of $Z = X + Y$ by convolving the two marginal distributions.

Dependent Events Example (Cont'd)

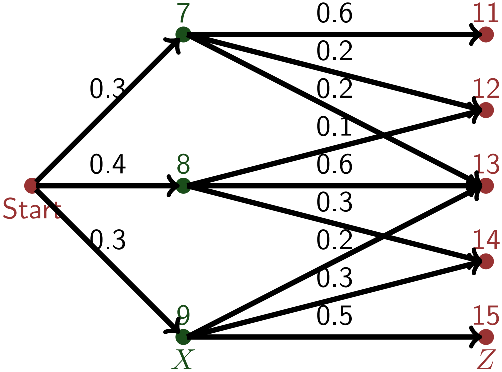
The marginal distribution of X by itself is $[0.3 \ \ 0.4 \ \ 0.3]$. Let us compute the matrix of *conditional probabilities* of Y for various outcomes of X.

		Y		
		4	5	6
	7	0.6	0.2	0.2
X	8	0.1	0.6	0.3
	9	0.2	0.3	0.5

This can be represented by an event tree as shown in the next slide. For simplicity the values of Y are not shown.

Note: This example is naturally suited for Bayesian networks.

Event Tree for Two-Stage Processing Example



Dependent Events Example (Cont'd)

There is one path each leading to $Z = 11$ and to $Z = 15$, two paths each leading to $Z = 12$ and to $Z = 14$, and three paths leading to $Z = 13$.

The probability of each path can either be computed using the table or conditional probabilities. For example

$$\begin{aligned} \Pr\{Z = 12\} &= \Pr\{X = 7 \& Y = 5\} + \Pr\{X = 8 \& Y = 4\} \\ &= 0.06 + 0.04 = 0.10. \end{aligned}$$

Dependent Events Example (Cont'd)

But we can also compute

$$\Pr\{X = 7 \& Y = 5\} = \Pr\{X = 7\} \cdot \Pr\{Y = 5 | X = 7\}$$

and similarly for the other term. Both approaches give the same answer.

Adding up all the probabilities gives the probability distribution of Z:

$$\phi_Z = [\ 0.18 \quad 0.10 \quad 0.36 \quad 0.21 \quad 0.15\ ].$$

It is possible to automate this process so that quite large event trees can be analyzed.

Outline

Decision Trees

Decision trees are essentially the same as event trees, except that at several nodes, some decision needs to be taken, such as the associated profit (or loss), the amount of investment needed, etc.

In short, the outcomes in decision trees are usually numerical, whereas in event trees they can also be "abstract" r.v.s.

Decision Tree Example

Consider again the two-stage process with dependent processing times shown earlier. Suppose the company receives a bonus or pays a penalty (in millions of dollars) depending on the processing time, as follows:

Outcome	Proc. Time	Bonus	Prob.
1	11	3	0.18
2	12	1	0.10
3	13	0	0.36
4	14	-1	0.21
5	15	-4	0.15

The expected value of the bonus/penalty is -0.17 million dollars.

Decision Tree Example (Cont'd)

Suppose you can replace the current processing station Y with another machine that costs 1 million, which can be used 10 times before it needs replacement. With this new machine, the second station takes less time. The joint probability distribution of X and Y is now given by

		Y			
		4	5	6	
	7	0.18	0.06	0.06	0.30
X	8	0.04	0.32	0.04	0.40
	9	0.06	0.12	0.12	0.30
		0.28	0.50	0.22	1.00

Decision Tree Example (Cont'd)

Now the probabiity distribution of $Z = X + Y$ and associated bonus are

Outcome	Proc. Time	Bonus	Prob.
1	11	3	0.18
2	12	1	0.10
3	13	0	0.44
4	14	-1	0.16
5	15	-4	0.12

The expected value of the bonus/penalty is zero, against -0.17 million. So it is worthwhile to do the replacement.

Decision Trees: Useful URL

A detailed discussion of decision trees and several examples can be found at the following URL:

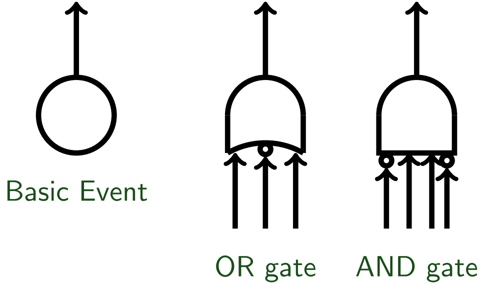
> *http://orms.pef.czu.cz/text/game-theory/DecisionTheory.html*

Outline

General Philosophy

- Event trees and decision trees are *forward-looking* – they start with initiating events, follow up on subsequent events, and end at the set of final outcomes.
- Fault trees are *backward-looking* – they start with the failure state, and work backwards to see what all events can lead to the failure. Then they compute (or estimate) the probability of at least one of these events occurring.
- Unlike event trees, fault trees use *Boolean algebra* to analyze failures.
    - Every random variable in the system is taken as binary, with an appropriate probability of equalling 0 or 1.
    - Every intermediate variable is an 'and' or 'or' of two or more input variables (or their negations).

Standard Symbolism for Fault Trees



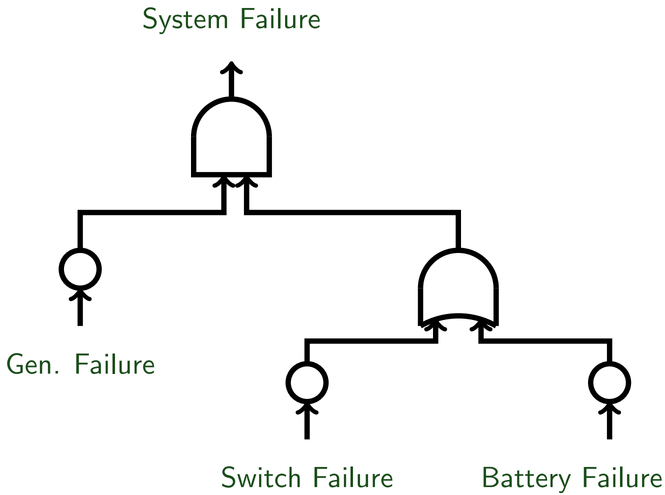
Each gate can have multiple inputs, some of which are “negated,” but only one output.

Simple Example: Generator Back-Up System

- When a generator fails, the system switches over to a battery back-up.
- But if the switch or the battery fails, then the back-up also fails.

How do we represent this graphically?

Fault Tree Representation



Boolean Representation

We can write the event "System Failure" (F) as a Boolean variable in terms of the "basic" Boolean variables "Generator Failure" (G), "Switch Failure" (S) and "Battery Failure" (B).

Recall the following standard facts about Boolean algebra:

$$X \wedge Y \ (X \text{ and } Y) \ = XY \text{ or } X \cdot Y.$$

$$X \vee Y \ (X \text{ or } Y) \ = X + Y.$$

So the generator back-up system can be represented as

$$F = G(S + B).$$

Minimal Cutsets

A **minimal cutset** is a set of Boolean variables such that

- Setting every variable to "True" causes the top event to be "True."
- Setting any proper subset to "True" does not necessarily cause the top event to be "True."

Minimal Cutsets of Back-Up Example

Because $F = G(S + B)$, there are two minimal cutsets: $\{G, S\}$ and $\{G, B\}$.

So the failure probability is the total probability that either the generator and switch fail together, or that the generator and battery fail together.

If we assume that all these events are independent, then the total failure probability is $P(G)P(S) + P(G)P(B)$.

Even if the events are not necessarily independent, this gives a quick upper bound.

Identifying Minimal Cutsets in Large Fault Trees

Even in very large fault trees, it is possible to identify minimal cutsets efficiently.

- Start from the top event.
- If it is an "AND" gate, each input must be true (or false if one input is negated).
- If it is an "OR" gate, any one input must be true.
- Repeat all the way down.

This approach can be automated, making this a very practical approach.

Constructing Fault Trees

One of the major advantages of fault trees is that it is possible to construct models for individual subsystems, and then “aggregate” them to get a fault tree for the overall system.

This is highly desirable because usually *no one* has a thorough understanding of every single component.

Also, there exist algorithms for constructing minimal cutsets for quite large fault trees, thus enhancing the applicability of the approach.

Combining Fault Trees and Event Trees

Often event trees are very simplified representations of complex systems, where each node represents a fairly complex subsystem. Fault trees can be used to construct very elaborate representations of each subsystem, and also to provide worst case failure probabilities of each subsystem. These numbers can then be used in the event tree analysis.

Outline

Bayesian Networks: Philosophy

Bayesian networks are a way of representing *conditional dependence* among random variables.

Each node represents one of the possible values of a random variable. Edges pointing into that node are weighted by the *conditional probability* of the present node given the previous node.

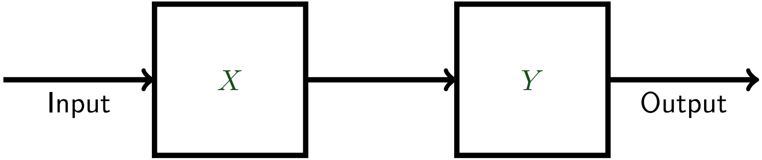
So the weight of an edge from the node $X = a$ to the node $Y = b$ is $\Pr\{Y = b | X = a\}$.

In explicitly permitting random variables to be dependent, Bayesian networks are more general than event trees or fault trees.

However, even Bayesian networks permit only "one-step" dependence as we shall see.

Bayesian Network Example 1

Consider the two-stage processing example studied earlier:



Suppose X takes values in $\{7, 8, 9\}$ while Y takes values in $\{4, 5, 6\}$.

Bayesian Network Example 1 (Cont'd)

Suppose the joint probability distribution of X and Y is given by

		Y			
		4	5	6	
	7	0.18	0.06	0.06	0.30
X	8	0.04	0.24	0.12	0.40
	9	0.06	0.09	0.15	0.30
		0.28	0.39	0.33	1.00

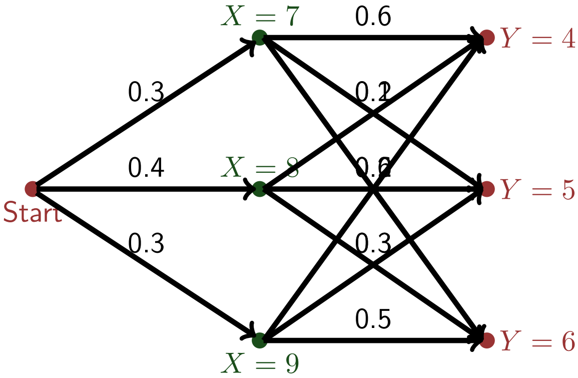
Bayesian Network Example 1 (Cont'd)

The marginal distribution of X by itself is $[0.3 \;\; 0.4 \;\; 0.3]$, and the matrix of *conditional probabilities* of Y for various outcomes of Y:

		Y		
		4	5	6
	7	0.6	0.2	0.2
X	8	0.1	0.6	0.3
	9	0.2	0.3	0.5

This can be represented by a Bayesian Network as shown in the next slide.

Bayesian Network Example 1 (Cont'd)

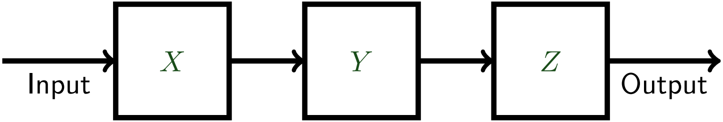


But $Z = X + Y$ cannot be efficiently represented by a Bayesian network because Z depends on both X and Y.

Bayesian Network Example 2

Bayesian networks are capable of capturing only *one-step memory.*

Consider the three-stage manufacturing process below.



Bayesian networks can accommodate the case where Y depends on X and Z depends on Y *alone,* but not the case where Y depends on X and Z depends on *both* Y *and* X.

Bayesian Belief Networks

Bayesian networks are very popular in artificial intelligence (AI) circles, because the actual conditional probabilities can be replaced by one's "beliefs" as to what those conditional probabilities ought to be.

By treating these "beliefs" as actual conditional probabilities, the beliefs can be "propagated" along the network.

Since different human experts are knowledgeable about different parts of the network (and can thus give realistic "beliefs"), while being ignorant about everything else, Bayesian networks provide a simple and mathematically valid procedure for combining beliefs from different humans.