Lecture 24

Decision Networks and Sequential Decision Problems

Lecture Overview

- Recap

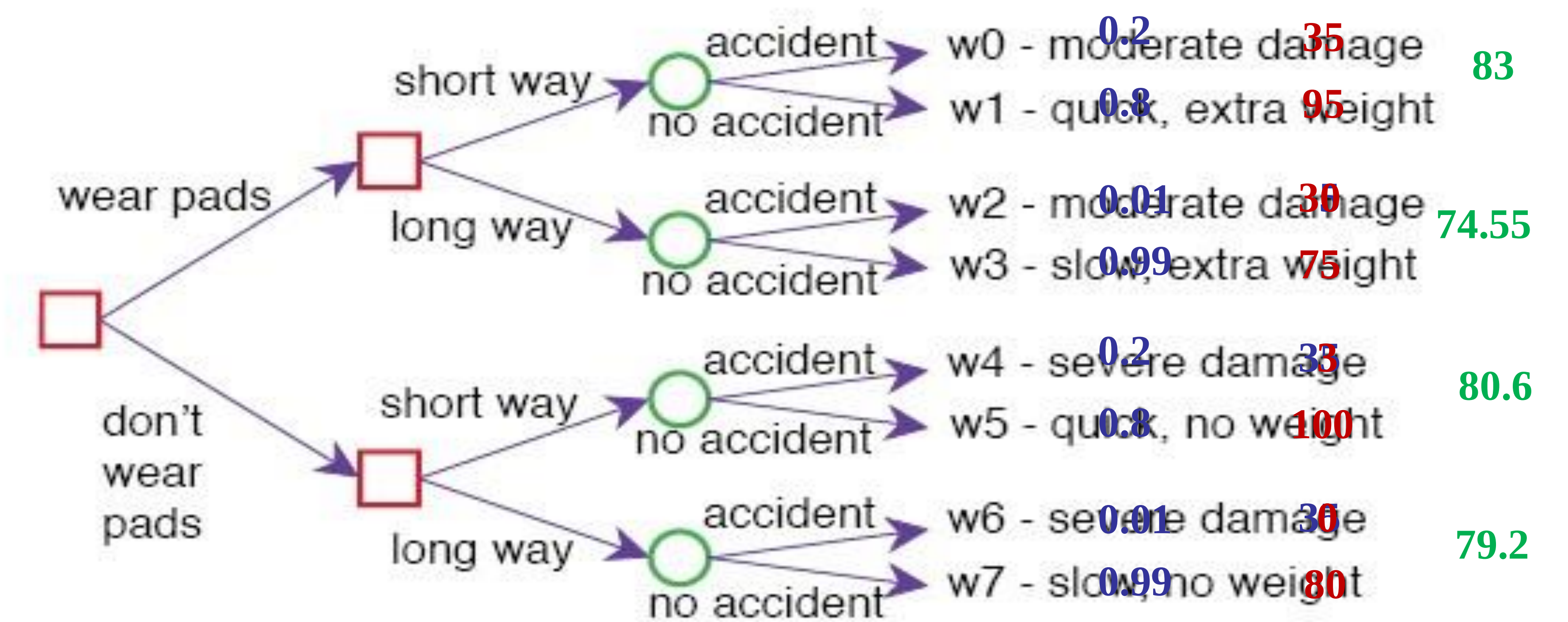
- Computing single-stage optimal decision
- Sequential Decision Problems
- Finding Optimal Policies with VE

Expected utility of a decision

- The expected utility of decision $D = d$ is

$$E(U \mid D = d) = \sum_{w \models (D = d)} P(w)\, U(w) = \mathbf{P(w_1) \times U(w_1) + \ldots + P(w_n) \times U(w_n)}$$

Probability Utility E[U|D]



Optimal single-stage decision

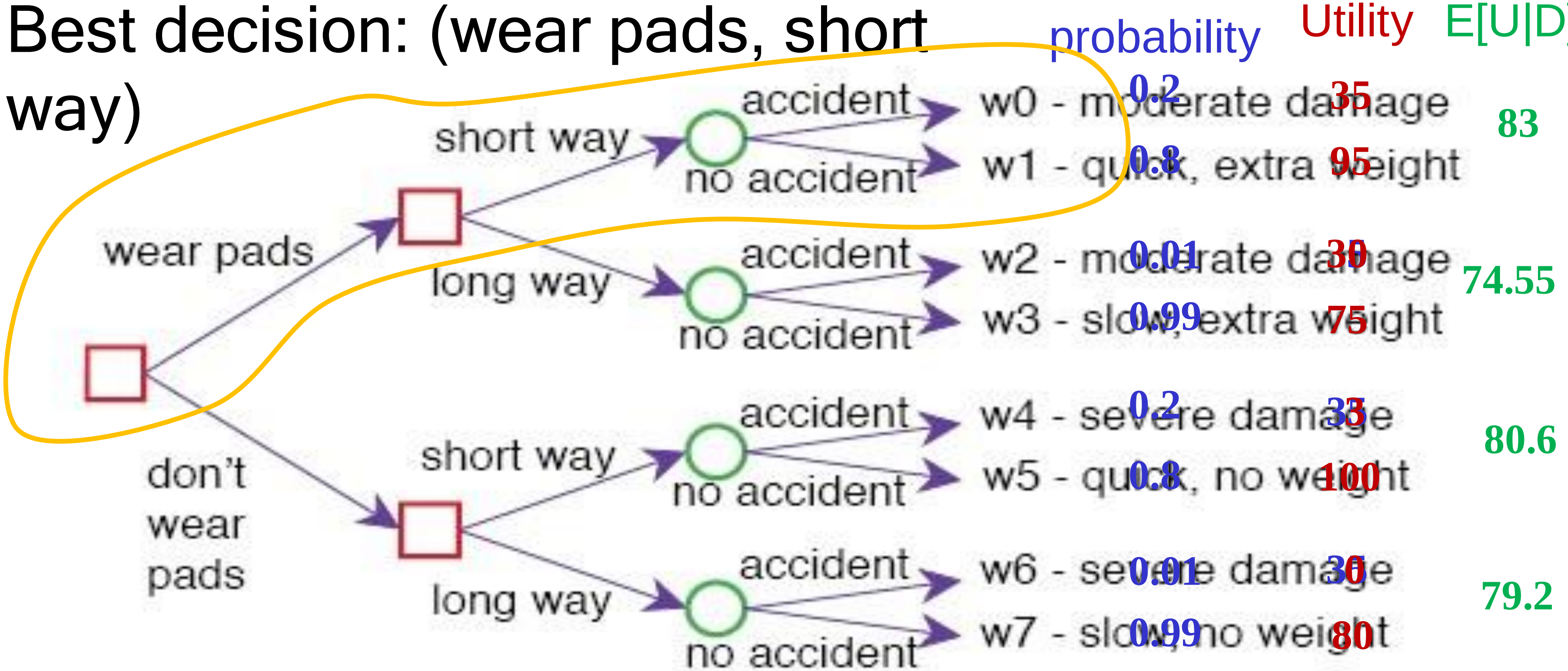
- Single Stage (aka One-Off) Decisions
- One or more primitive decisions that can be treated as a single macro decision to be made before acting
- Given a single (macro) decision variable D
- the agent can choose $D=d_i$ for any value $d_i \in dom(D)$

Definition (optimal single-stage decision)
An optimal single-stage decision is the decision $D=d_{max}$ whose expected value is maximal:

$$d_{max} \in \underset{d_i \in dom(D)}{\operatorname{argmax}} \; E[U|D=d_i]$$

Optimal decision in robot delivery example

Best decision: (wear pads, short way)



Definition (optimal single-stage decision)

An optimal single-stage decision is the decision $D{=}d_{max}$ whose expected value is maximal:

$$d_{max} \in \underset{d_i \in dom(D)}{\operatorname{argmax}} E[U|D{=}d_i]$$

Conditional

Single-Stage decision networks

Extend belief networks

Random variables: same as in Bayesian networks

- drawn as an ellipse
- Arcs into the node represent probabilistic dependence
- random variable is conditionally independent of its non-descendants gi its parents

Decision nodes, that the agent chooses the value for

- Parents: only other decision nodes allowed
  - ✓represent information available when the decision is made
- Domain is the set of possible actions
- Drawn as a rectangle Exactly one utility node

Explicitly shows dependencies. E.g., which variables affect the probability of an accident and the agent's utility?

- Parents: all random & decision variables on which the utility depends
- Specifies a utility for each instantiation of its parents
- Drawn as a diamond

Example Decision Network

Which Way W	Accident A	P(A|W)

Which way	Accident	Wear Pads	Utility
long	true	true	30
long	true	false	0
long	false	true	75
long	false	false	80
short	true	true	35
short	true	false	3
short	false	true	95
short	false	false	100

long long short short	true false true false	0.01 0.99 0.2 0.8

Which Way

t
f

Decision nodes simply list the available decisions.

Applet for Bayesian and Decision Networks

The Belief and Decision Networks we have seen previously allows you to load predefined Decision networks for various domains and run queries on them.

Select one of the available examples via “File -> Load Sample Problem

For Decision Networks

- Choose any of the examples below the blue line in the list that appears
- Right click on a node to perform any of these operations
- View the CPT/Decision table/Utility table for a chance/decision/utility node
- Make an observation for a chance variable (i.e., set it to one of its values)
- Query the current probability distribution for a chance node given the observations made

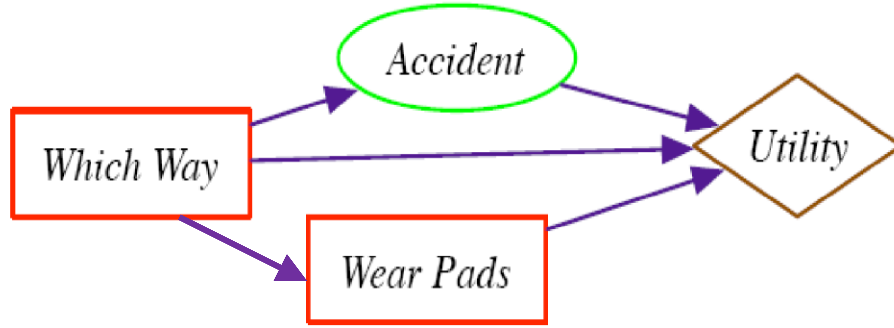
- A dialogue box will appear the first time you do this. Select “Always brief” at the bottom, and then click “Brief”.
- To compute the optimal decision (policy) click on the “Optimize Decision” button in the toolbar and select Brief in the dialogue box that will appear
- To see the actual policy, view the decision table for each decision node in the network

See available help pages and video tutorials for more details on how to use the Bayes applet (http://www.aispace.org/bayes/)

Lecture Overview

- Recap
- Computing single-stage optimal decision
- Sequential Decision Problems
- Finding Optimal Policies with VE

Computing the optimal decision: we can use VE



Denote

- the random variables as $X_1, \ldots, X_n$
- the decision variables as D
- the parents of node N as pa(N)

$$E(U) = \sum_{X_1,\ldots,X_n\, n} P(X_1,\ldots,X_n \mid D)\,U(pa(U))$$

$$= \sum_{X_1,\ldots,X_n} \prod_{i=1} P(X_i \mid pa(X_i))\,U(pa(U))$$

VE Example: Step 1, create initial factors

- To find the optimal decision we can use VE:

Includes decision vars

1. Create a factor for each conditional probability and for the utility
2. Sum out all random variables, one at a time
   1. This creates a factor on D that gives the expected utility for each d_i
3. Choose the d_i with the maximum value in the factor

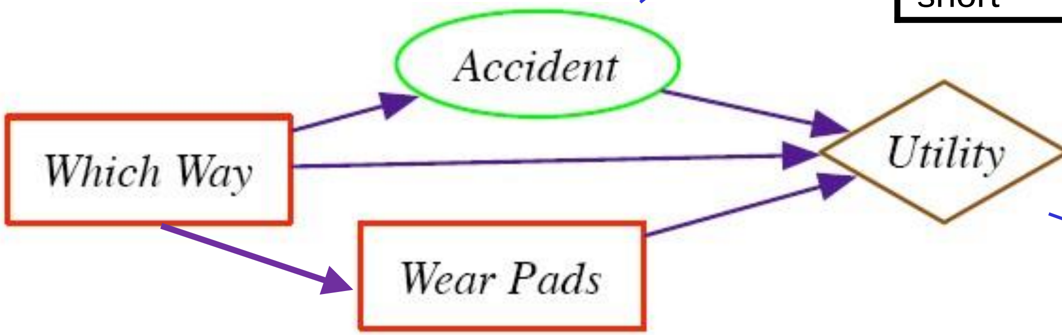
Which way W	Accident A	Pads P	Utility
long	true	true	30
long	true	false	0
long	false	true	75
long	false	false	80
short	true	true	35
short	true	false	3
short	false	true	95
short	false	false	100

Abbreviations:

W = Which Way
P = Wear Pads
A = Accident

$f_1(A,W)$

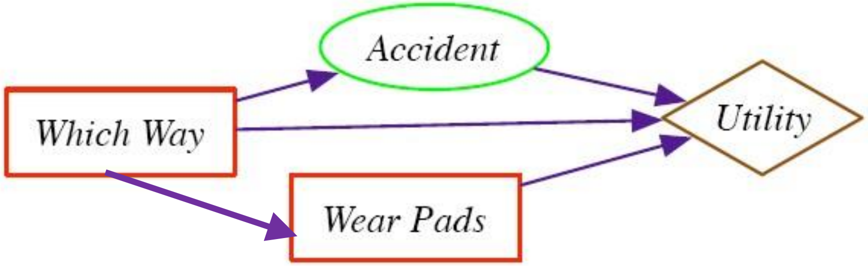
Which Way W	Accident A	P(A|W)
long	true	0.01
long	false	0.99
short	true	0.2
short	false	0.8



$f_2(A,W,P)$

$$E(U) = \sum_A P(A|W)\,U(A,W,P)$$

$$= \sum_A f_1(A,W)\, f_2(A,W,P)$$



VE example: step 2, sum out A

Step 2a: compute product
$f_1(A,W) \times f_2(A,W,P)$

What is the right form for the product $f_1(A,W) \times f_2(A,W,P)$?

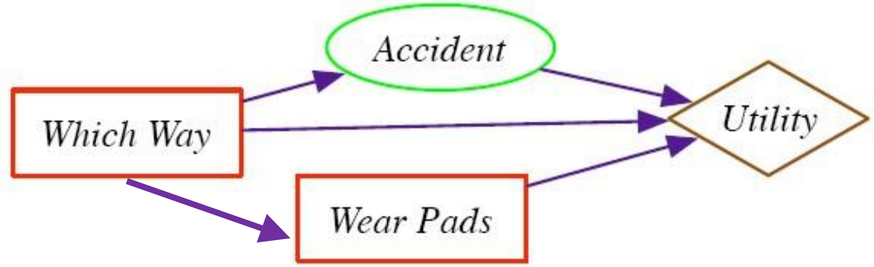
- It is $f(A,P,W)$:

  the domain of the product is the union of the multiplicands' domains

- $f(A=a,P=p,W=w) = f_1(A=a,W=w) \times f_2(A=a,W=w,P=p)$

Which way W	Accident A	Pads P	$f_2(A,W,P)$

long	true	true	30
long	true	false	0
long	false	true	75
long	false	false	80
short	true	true	35
short	true	false	3
short	false	true	95
short	false	false	100



VE example: sum out A step 2,

Which way W	Accident A	$f_1(A,W)$
long	true	0.01
long	false	0.99
short	true	0.2
short	false	0.8

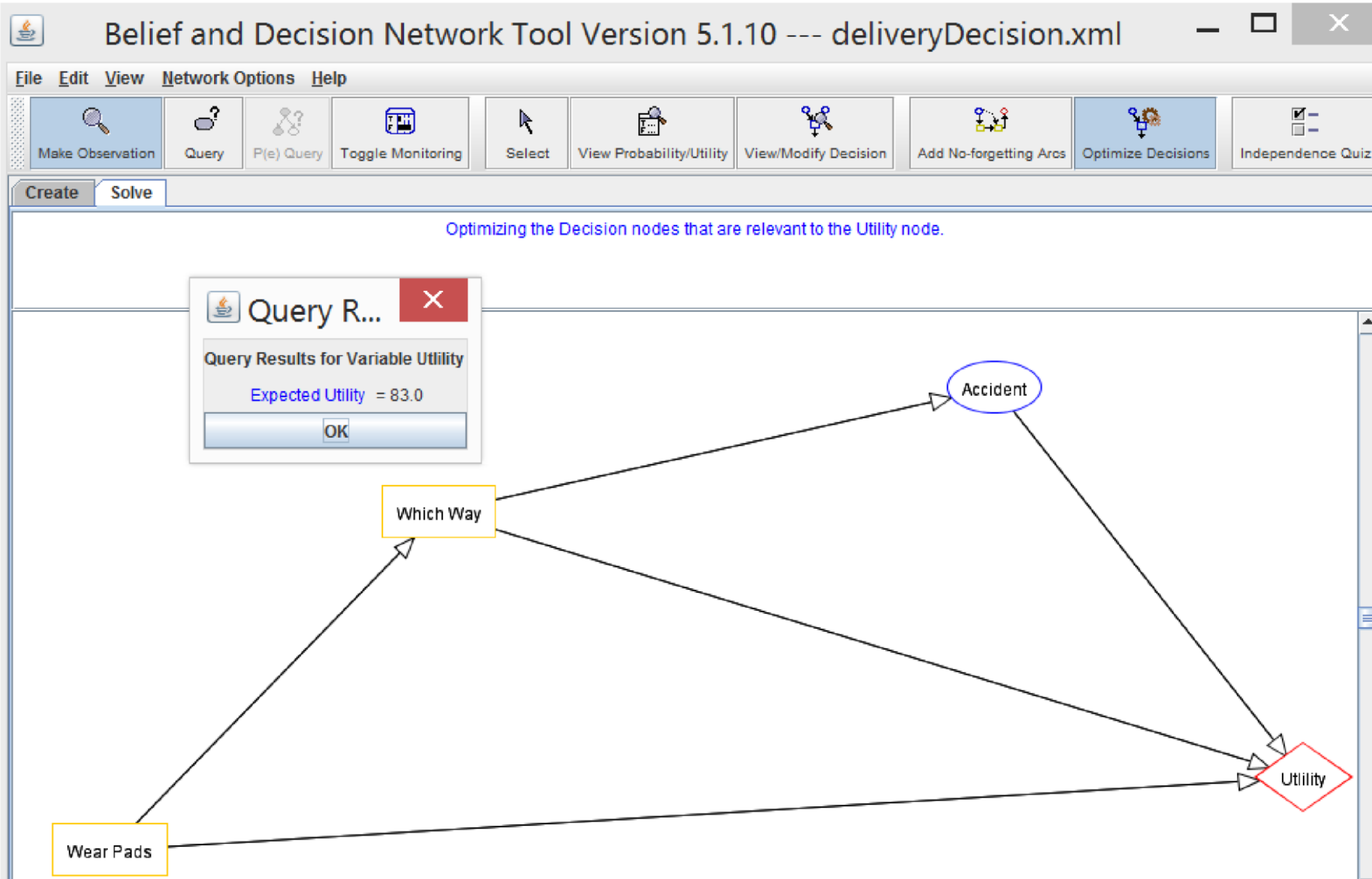
Step 2a: compute product $f_1(A,W) \times f_2(A,W,P)$

$$f(A=a,P=p,W=w) = f_1(A=a,W=w) \times f_2(A=a,W=w,P=p)$$

Which way W	Accident A	Pads P	f(A,W,P)
long	true	true	
long	true	false	
long	false	true	

long	false	false	????
short	true	true	
short	true	false	
short	false	true	
short	false	false	

Getting the outcome with the applet



Select “optimize decision” in the menu bar

Lecture Overview

- Recap
- Computing single-stage optimal decision
- Sequential Decision Problems
- Finding Optimal Policies with VE

Sequential Decision Problems

- Under uncertainty, a typical scenario is that an agent observes, acts, observes, acts, …
- New observations are taken into account for acting

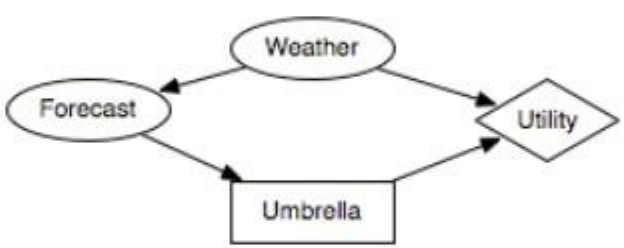
- Subsequent actions can depend on what is observed
- What is observed often depends on previous actions
- Often the sole reason for carrying out an action is to provide information for future actions ✓For example: diagnostic tests

- General Decision networks:

- Just like single-stage decision networks, with one exception: the parents of decision nodes can include random variables

Sequential decisions : Simplest possible

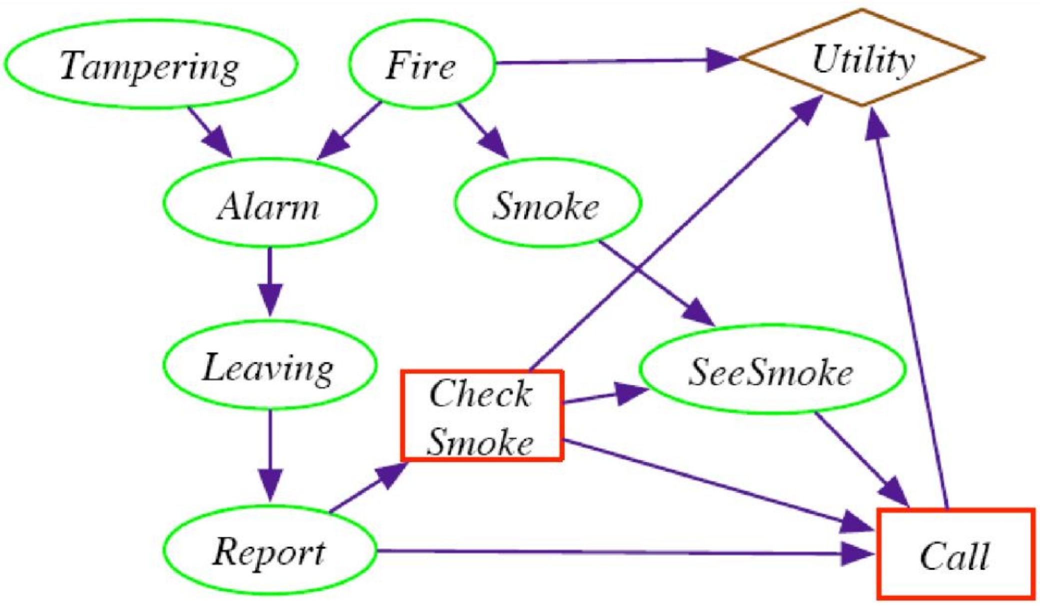
- Only one decision! (but different from one-off decisions)
- Early in the morning. Shall I take my umbrella today, based on the weather forecast? (I'll have to go for a long walk at noon)
- Relevant Random Variables?



Sequential Decision Problems: Example

- In our Fire Alarm domain
- If there is a report you can decide to call the fire department
- Before doing that, you can decide to check if you can see smoke, but this takes time and will delay calling
- A decision (e.g. Call) can depend on a random variable

(e.g. SeeSmoke)



Decision node: Agent decides

Chance node: Chance decides

Each decision D_i has an information set of variables $pa(D_i)$, whose value will be known at the time decision D_i is made

- pa(CheckSmoke) = {Report}
- pa(Call) = {Report, CheckSmoke, See Smoke}

The no-forgetting property

- A decision network has the no-forgetting property if
- Decision variables are totally ordered: $D_1, \ldots, D_m$
- If a decision D_i comes before D_j ,then

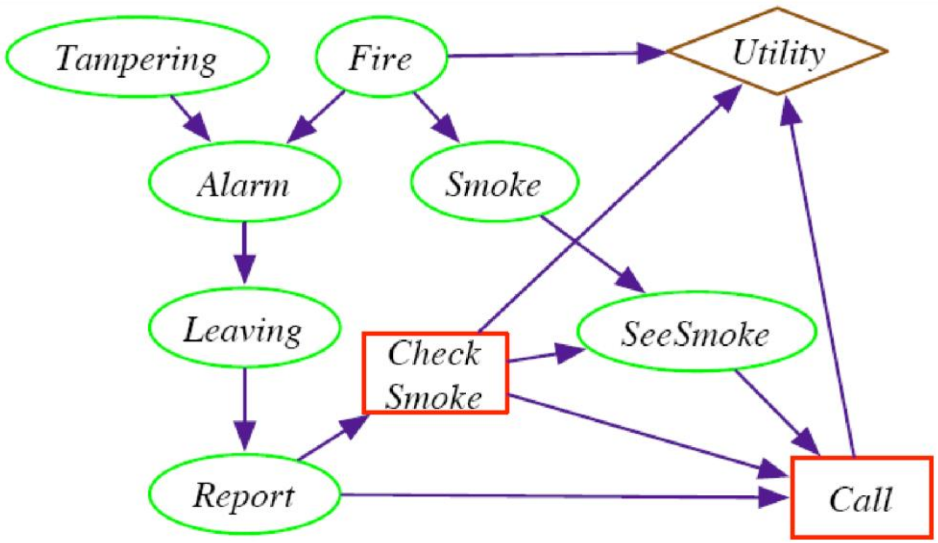
✓D_i is a parent of D_j

✓any parent of D_i is a parent of D_j

pa(CheckSmoke) = {Report}

pa(Call) = {Report, CheckSmoke, See Smoke}

Sequential Decision Problems



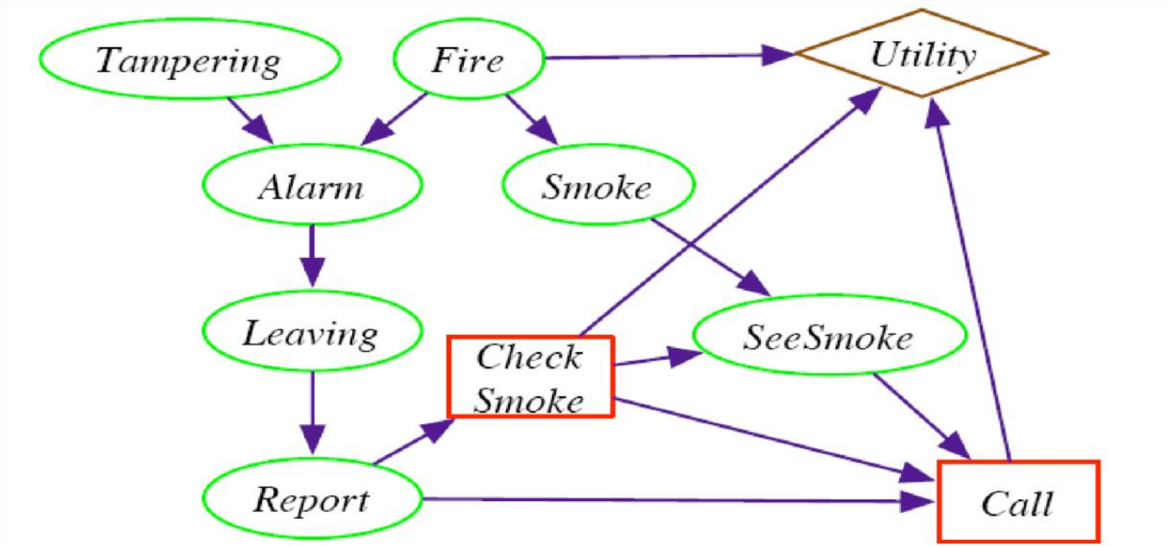
- What should an agent do?

- Subsequent actions can depend on what is observed

✓What is observed often depends on previous actions

The agent needs a conditional plan of what it will do given every possible set of circumstances

✓This conditional plan is referred to as a policy

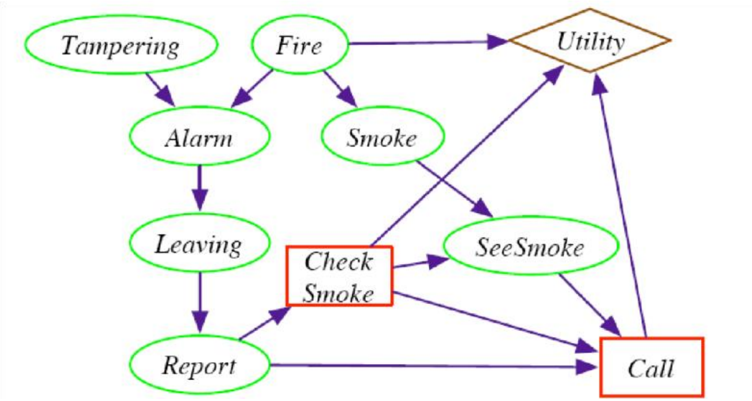


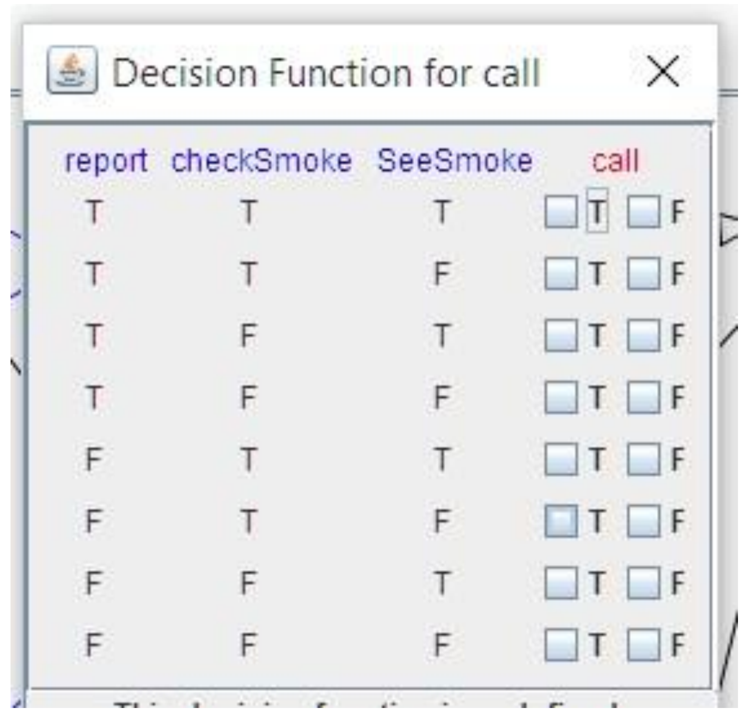
Policies for Sequential Decision Problems

Definition (Policy)

A policy specifies, for each decision node, which value it should take for every possible combination of values for its parents

For instance, in our Alarm problem, specifying a policy means selecting specific values for the two decision nodes, given all possible combinations of values of their parents



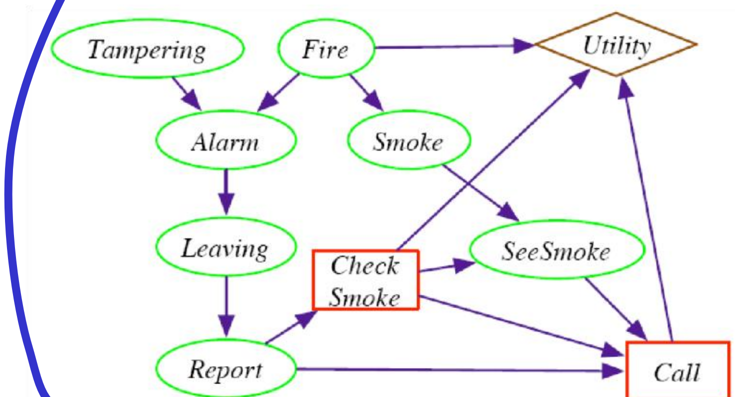


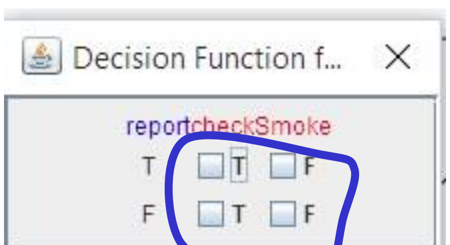
Policies for Sequential Decision Problems

Definition (Policy)

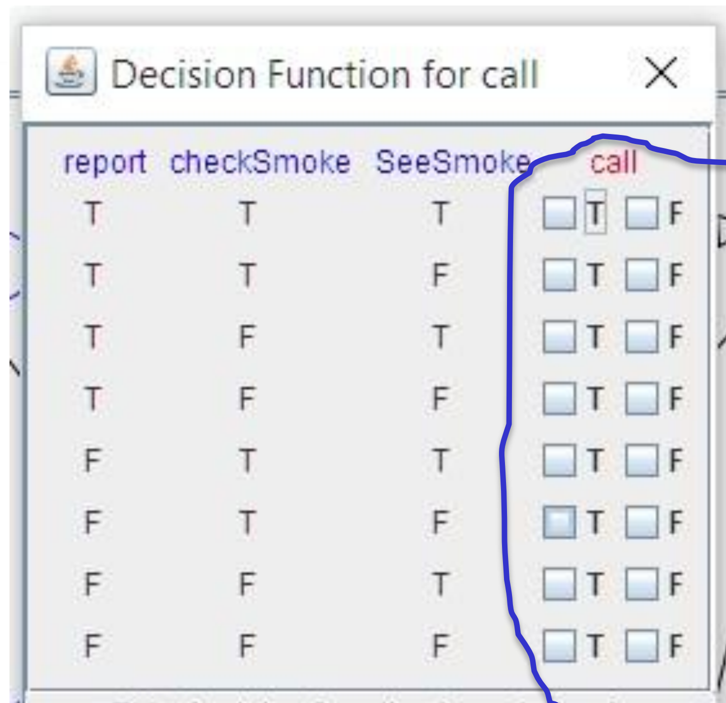
A policy specifies, for each decision node, which value it should take for every possible combination of values for its parents

That is, selecting a policy means selecting either T or F for each of these entries





report	checkSmoke		
T		T	F
F		T	F

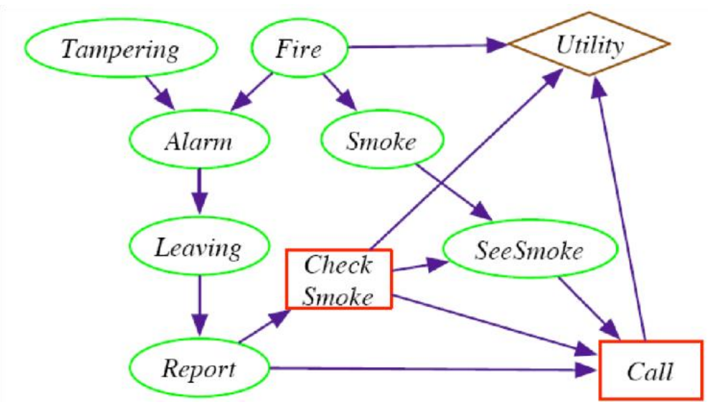


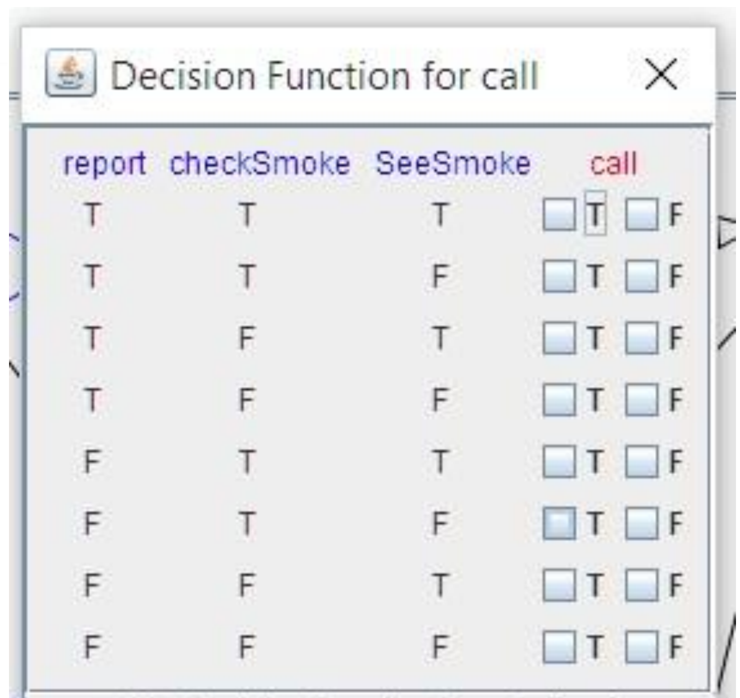
report	checkSmoke	SeeSmoke	call	
T	T	T	T	F
T	T	F	T	F
T	F	T	T	F
T	F	F	T	F
F	T	T	T	F
F	T	F	T	F
F	F	T	T	F
F	F	F	T	F

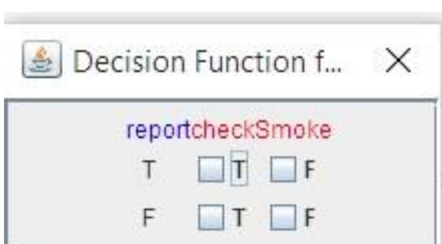
Policies for Sequential Decision Problems

Definition (Policy)
A policy specifies, for each decision node, which value it should take for every possible combination of values for its parents

Why do we want to do that? Because we want to enable an agent to know what to do under every "possible world" that can be observed







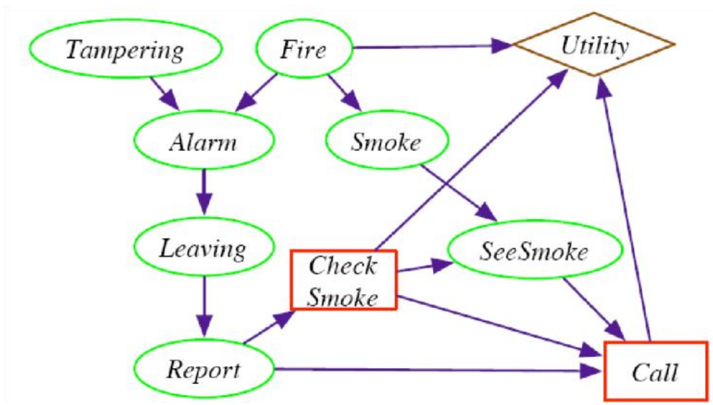
Policies for Sequential Decision Problems

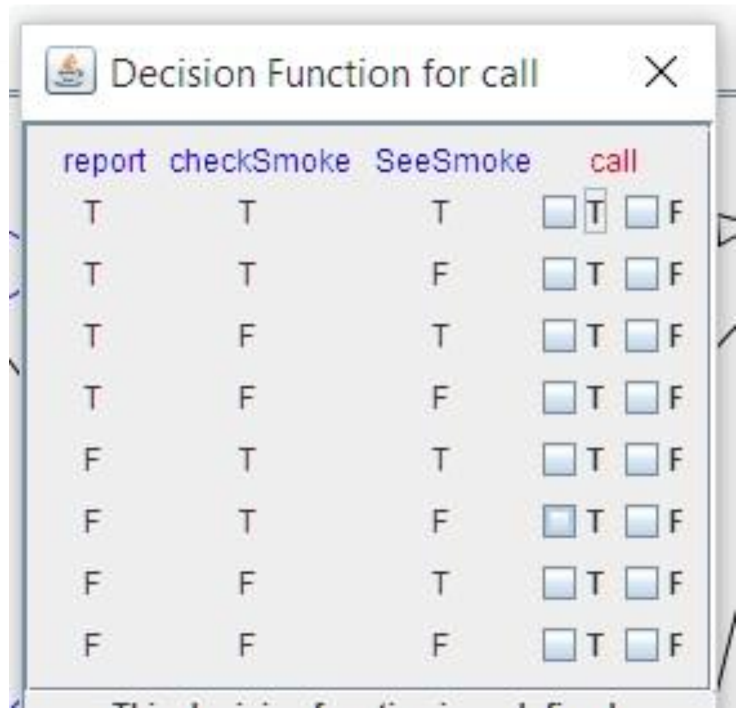
Policy: Formal Definition

A policy is a sequence of $\delta_1, \ldots, \delta_n$ decision functions δ_i :

$$\mathrm{dom}(\mathrm{pa}(D_i)) \rightarrow \mathrm{dom}(D_i)$$

This policy means that when the agent has observed $o \in \mathrm{dom}(\mathrm{pa}(D_i))$, it will do $\delta_i(o)$





Policies for Sequential Decision Problems

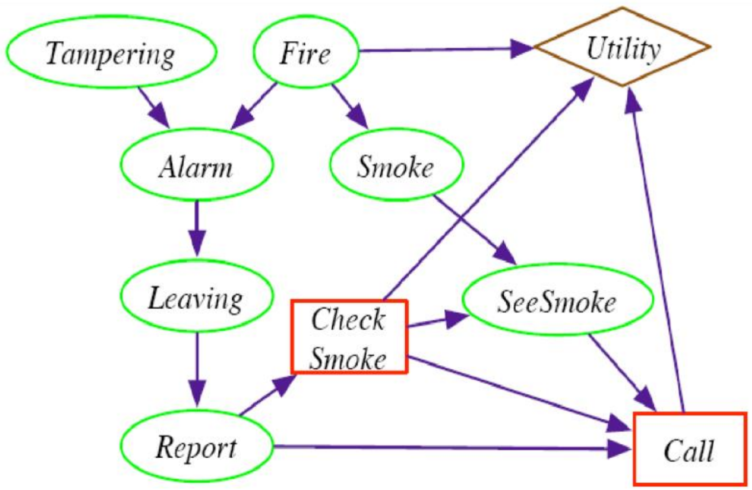
Definition (Policy)
A policy specifies, for each decision node, which value it should take for every possible combination of values for its parents

Why do we want to do that? Because we want to enable an agent to know what to do under every "possible world" that can be observed

Policy: Formal Definition
A policy is a sequence of $\delta_1, \ldots, \delta_n$ decision functions δ_i :

$$\text{dom}(\text{pa}(D_i)) \rightarrow \text{dom}(D_i)$$

This policy means that when the agent has observed $o \in \text{dom}(\text{pa}(D_i))$, it will do $\delta_i(o)$



Complexity of policy finding

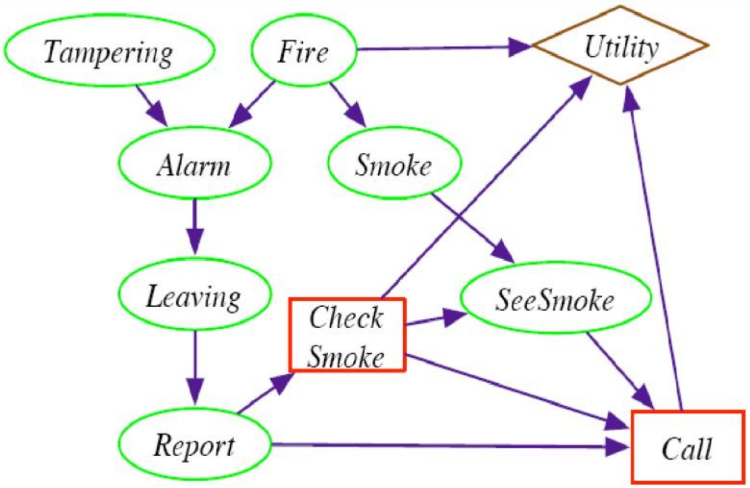
- There are 2^2=4 possible ways to assign what to do for the decision to check smoke
- Let's identify each of these assignments with the symbol δ_{cs} followed by a specific number

CheckSmoke

Report	$\delta_{cs}1$	$\delta_{cs}2$	$\delta_{cs}3$	$\delta_{cs}4$
T				
F				

Definition (Policy)

A policy π is a sequence of δ_1 ,…..,δ_n decision functions δ_i : $dom(pa(D_i)) \rightarrow dom(D_i)$



Complexity of policy finding

- There are 2^2=4 possible ways to assign what to do for the decision to check smoke
- Let's identify each of these assignments with the symbol δ_{cs} followed by a specific number

CheckSmoke

Report	$\delta_{cs}1$	$\delta_{cs}2$	$\delta_{cs}3$	$\delta_{cs}4$
T	T	T	F	F
F	T	F	T	F

Definition (Policy)

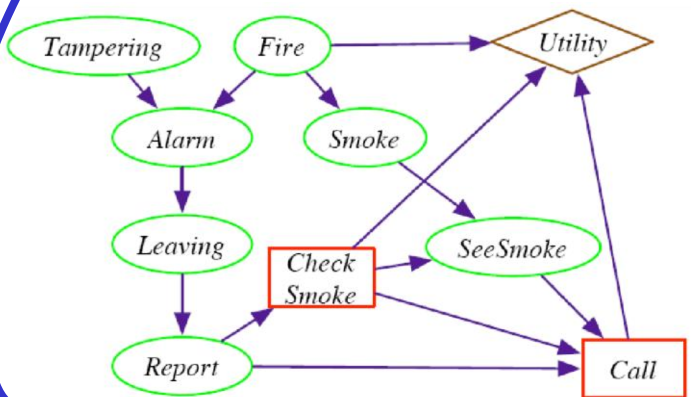
A policy π is a sequence of δ_1 ,…..,δ_n decision functions δ_i : dom(pa(D_i)) → dom(D_i)

Policies for Sequential Decision Problems

Definition (Policy)

A policy specifies, for each decision node, which value it should take for every possible combination of values for its parents

That is, selecting a policy means selecting either T or F for each of these entries



Decision Function for call

report	checkSmoke	SeeSmoke	call
T	T	T	☐ T ☐ F
T	T	F	☐ T ☐ F
T	F	T	☐ T ☐ F
T	F	F	☐ T ☐ F
F	T	T	☐ T ☐ F
F	T	F	☐ T ☐ F
F	F	T	☐ T ☐ F
F	F	F	☐ T ☐ F

Decision Function f...

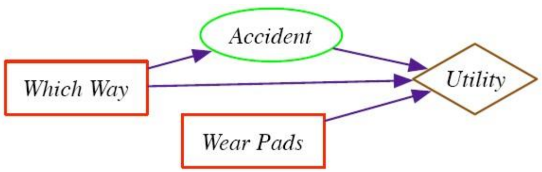
report	checkSmoke
T	☐ T ☐ F
F	☐ T ☐ F

Policies for Sequential Decision Problems

- Policies for sequential decisions are the counterpart of Single

Decisions for One-Off decisions

- They are just much more complex, because they tell the agent what to do for any step of the decision sequence given any possible combination of available information



Sample decision

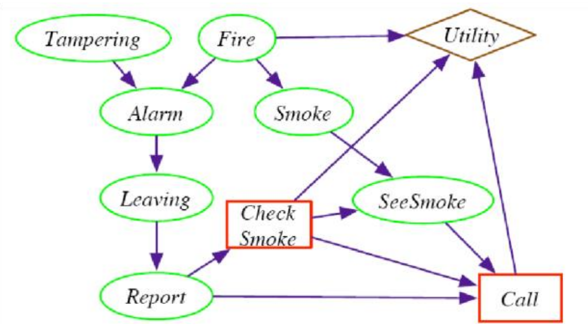
	Which way	Wear Pads
d1	long	true
d2	long	false
d3	short	true
d4	short	false

Sample policy

Sample policy

Report	$\delta_{cs}1$	$\delta_{cs}2$	$\delta_{cs}3$	$\delta_{cs}4$
T	T	T	F	F
F	T	F	T	F

CheckSmoke



Call

Report	CheckS	SeeS	$\delta_{call}1$	… …	$\delta_{call}k$	…..	$\delta_{call}n$
T	T	T	T		T		T
T	T	F	F		T		T
T	F	T	T	…	T		T
T	F	F	F	…	F		T
F	T	T	T	…	T		T
F	T	F	F	….	F		T
F	F	T	T		T		T
F	F	F	F		F		T
							T

How many policies are there?

If we have a problem with d decision variables, each with b possible values, and k binary parents,

- there are 2^k different assignments of values to the parents
- if there are b possible value for a decision variable with k binary parents
- There are b^{2^k} different decision functions for that variable
- because there are 2^k possible instantiations for the parents and for each such instantiation, the decision function could pick any of the b values
- in total, there are $(b^{2^k})^d$ different policies
  - because there are b^{2^k} possible decision functions for each decision, and a policy is a combination of d such decision functions

Lecture Overview

- Recap
- Computing single-stage optimal decision
- Sequential Decision Problems
- Finding Optimal Policies with VE

Expected Value of a Policy

- Like for One-Off decisions, policies are selected based on their expected utility
- Each possible world has a probability P(w) and a utility U(w)
- The expected utility of policy π is

$$\sum_{w \models \pi} = P(w) * U(w)$$

Possible worlds that satisfy the policy

- The optimal policy is one with the maximum expected utility.

Optimality of a policy

Definition (expected utility of a policy)

The expected utility $E[\pi]$ of a policy π is:

$$E[\pi] = \sum_{w \models \pi} P(w)\, U(w)$$

$w \models \pi$ indicates a possible world w that satisfies a policy π,

Definition (optimal policy)

An optimal policy π_{max} is a policy whose expected utility is maximal among all possible policies $\prod$:

$$\pi_{max} \in \underset{\pi \in \prod}{\operatorname{argmax}} E[\pi]$$

Possible worlds satisfying a policy

Definition (Satisfaction of a policy)

A possible world w satisfies a policy π, written $w \models \pi$, if the value of each decision variable in w is the value selected

by its decision function in policy π (when applied to w)

Possible worlds satisfying a policy

Definition (Satisfaction of a policy)

A possible world w satisfies a policy π, written w ⊧π, if the value of each decision variable in w is the value selected

by its decision function in policy π (when applied to w)

VARs	
Fire	true
Tampering	false
Alarm	true
Leaving	true
Report	true
Smoke	true
SeeSmoke	true
CheckSmoke	true
Call	true

Report	Check Smoke
true	true
false	false

w2Decision function

δ_{cs}2

Policy π

Decision function δ_{call}20

Report	. . .	CheckSmoke	SeeSmoke	Call
true		true	true true	false
true	false			false
true		false	true	true
true		false	false	false
false		true	true	true
				alse
				alse
				alse

w2 $\models_{\pi}$?

Definition (Satisfaction of a policy)

A possible world w satisfies a policy π, written w $\models \pi$, if the value of each decision variable in w is the value selected by its decision function in policy π (when applied to w)

VARs	
Fire	true
Tampering	
Alarm	false
Leaving	true
Report	true
Smoke	true
SeeSmoke	true
CheckSmoke	true
Call	true

$w2 \models_{\pi} ?\ w2$

Decision function $\delta_{cs}2$

…

Report	Check Smoke
true false	true false

Policy π

Decision function $\delta_{call}20$

…

Definition (Satisfaction of a policy)

A possible world w satisfies a policy π, written w ⊧π, if the value of each decision variable in w is the value selected by its decision function in policy π (when applied to w)

Report	CheckSmoke	SeeSmoke	Call
true	true	true true	false
true	false		false
true	false	true	true
true	false	false	false
false	true	true	true
false	true	false	false
false	false true false		false
false	false		false

$w2 \models$? NO

Decision function $\delta_{cs}2$ …

Report	Check Smoke
true	true
false	false

Policy π

w2

VARs	
Fire	true
Tampering	false
Alarm	true
Leaving	true
Report	true
Smoke	true
SeeSmoke	true
CheckSmoke	true
Call	true

Decision function $\delta_{call}20$

Report	CheckSmoke	SeeSmoke	Call
true	true	true	false
true	true	false	false
true	false	true	true
true	false	false	false
false	true	true	true
false	true	false	false
false	false	true	false
false	false	false	false

To apply VE we need one last operation on factors

- Maxing out a variable: similar to marginalization
- But instead of taking the sum of some values, we take the max

$$\left(\max_{X_1} f\right)(X_2, \ldots, X_j) = \max_{x \in dom(X_1)} f(X_1 = x, X_2, \ldots, X_j)$$

B	A	C	$f_3(A,B,C)$
t	t	t	0.03
t	t	f	0.07
f	t	t	0.54
f	t	f	0.36
t	f	t	0.06
t	f	f	0.14

f	f	t	0.48
f	f	f	0.32

$$\boxed{\max}_B\ \boxed{f}_3\boxed{(A,B,C) = f}_4\boxed{(A,C)}$$

A	C	$f_4(A,C)$
t	t	0.54
t	f	
f	t	
f	f	

To apply VE we need one last operation on factors

- Maxing out a variable: similar to marginalization
- But instead of taking the sum of some values, we take the max

$$\left(\max_{X_1} f\right)(X_2, \ldots, X_j) = \max_{x \in dom(X_1)} f(X_1 = x, X_2, \ldots, X_j)$$

B	A	C	$f_3(A,B,C)$
t	t	t	0.03
t	t	f	0.07
f	t	t	0.54
f	t	f	0.36
t	f	t	0.06
t	f	f	0.14

f	f	t	0.48
f	f	f	0.32

$$\boxed{\max}_B \boxed{f}_3\boxed{(A,B,C) = f}_4\boxed{(A,C)}$$

A	C	$f_4(A,C)$
t	t	0.54
t	f	?
f	t	
f	f	

A. 0.36

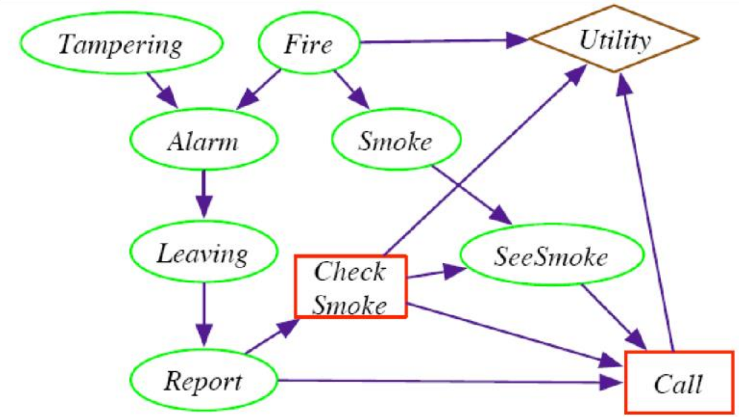
B. 0.48

C 0.32

D 0.8

Definition (optimal policy)
An optimal policy π_{max} is a policy whose expected utility is maximal among all possible policies $\prod$:

$$\pi_{max} \in \underset{\pi \in \prod}{\operatorname{argmax}} E[\pi]$$



Idea for finding optimal policies with VE

Consider the last decision D to be made

✓Find optimal decision D=d for each instantiation of D's parents

- For each instantiation of D's parents, this is just a single-stage decision problem

✓Create a factor of these maximum values: max out D

- I.e., for each instantiation of the parents, what is the best utility I can achieve by making this last decision optimally?

Recursive call to find optimal policy for reduced network (now with one less decision)

Finding optimal policies with VE

1. Create a factor for each CPT and a factor for the utility
2. While there are still decision variables
   - 2a: Sum out random variables that are not parents of a decision node.
   - 2b: Max out last decision variable D in the total ordering
     - ✓ Keep track of decision function
3. Sum out any remaining variable:
   this is the expected utility of the optimal policy.

This is Algorithm VE_DN in P&M, Section 9.3.3, p. 393

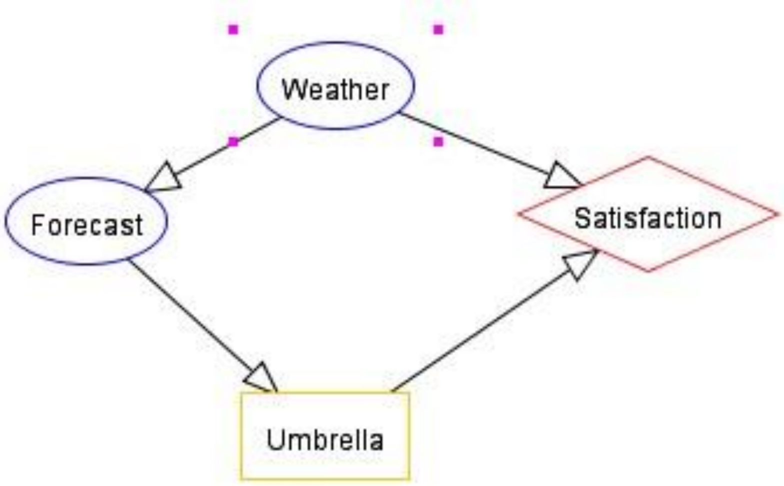
Finding optimal policies with VE

1. Create a factor for each CPT and a factor for the utility
2. While there are still decision variables

- 2a: Sum out random variables that are not parents of a decision node.
- 2b: Max out last decision variable D in the total ordering
  - ✓ Keep track of decision function

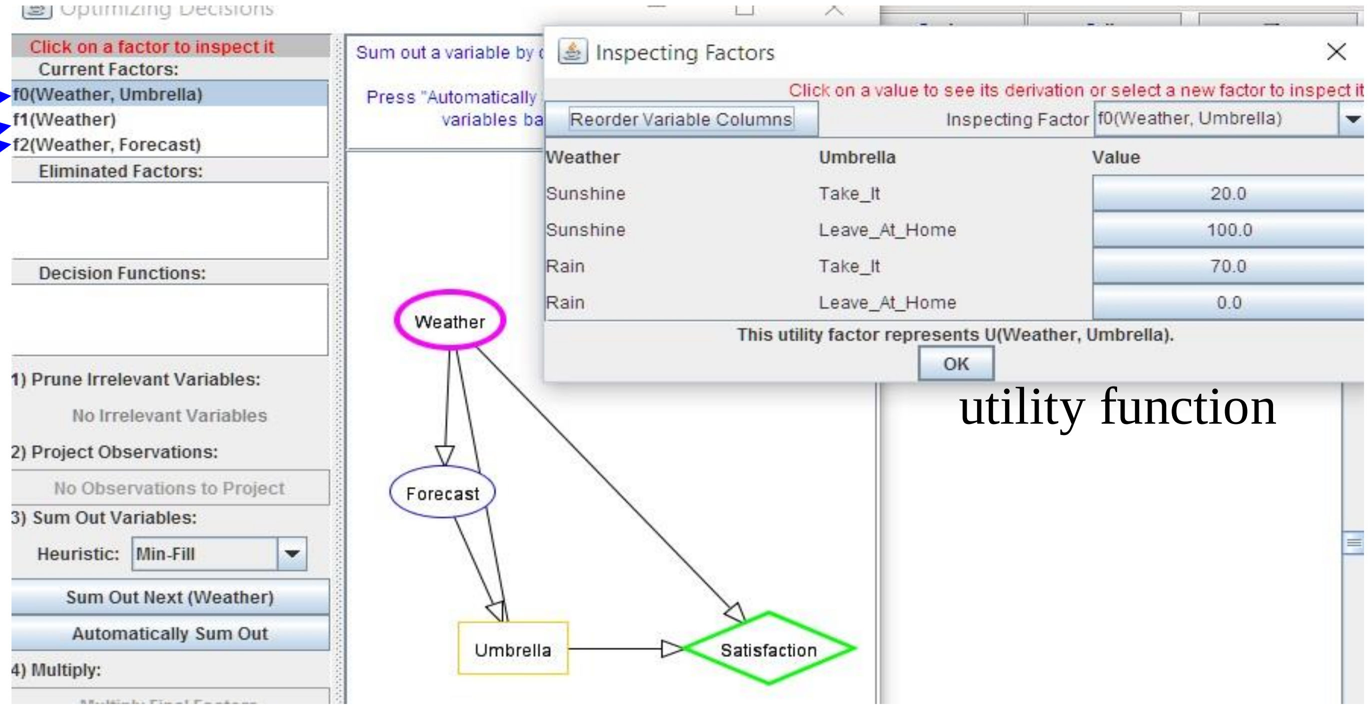
3. Sum out any remaining variable:
   this is the expected utility of the optimal policy.

Let's start with a simple example: only one decision node

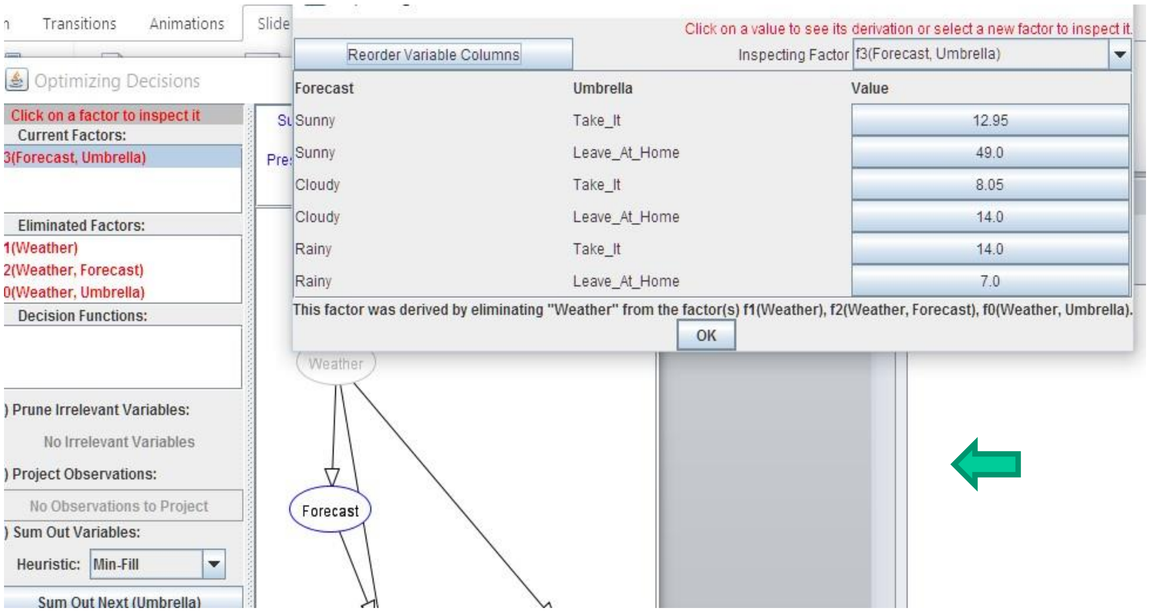


1: Create a factor for

- the utility
- each CPT

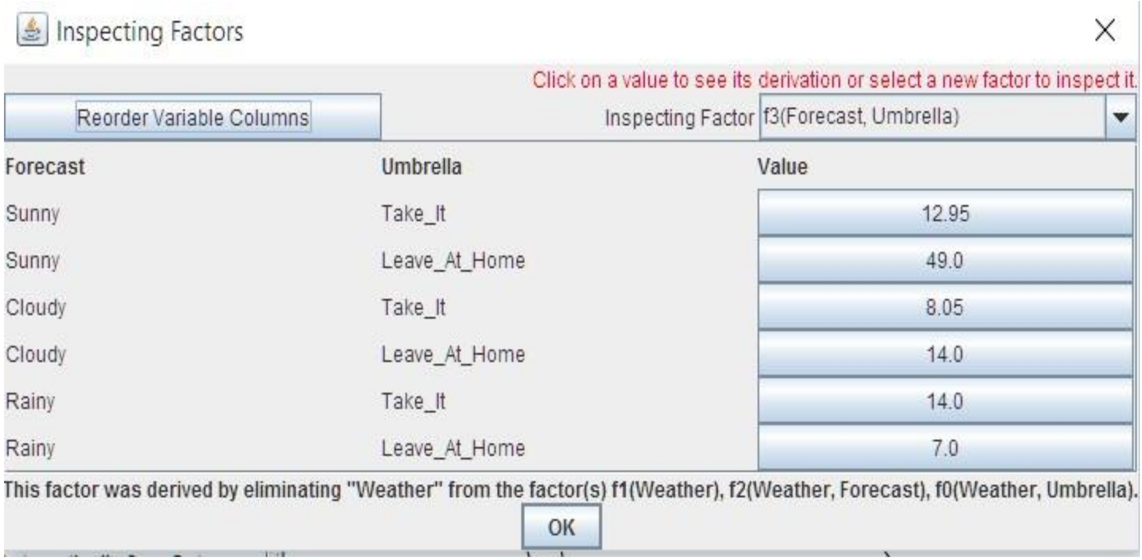


utility function

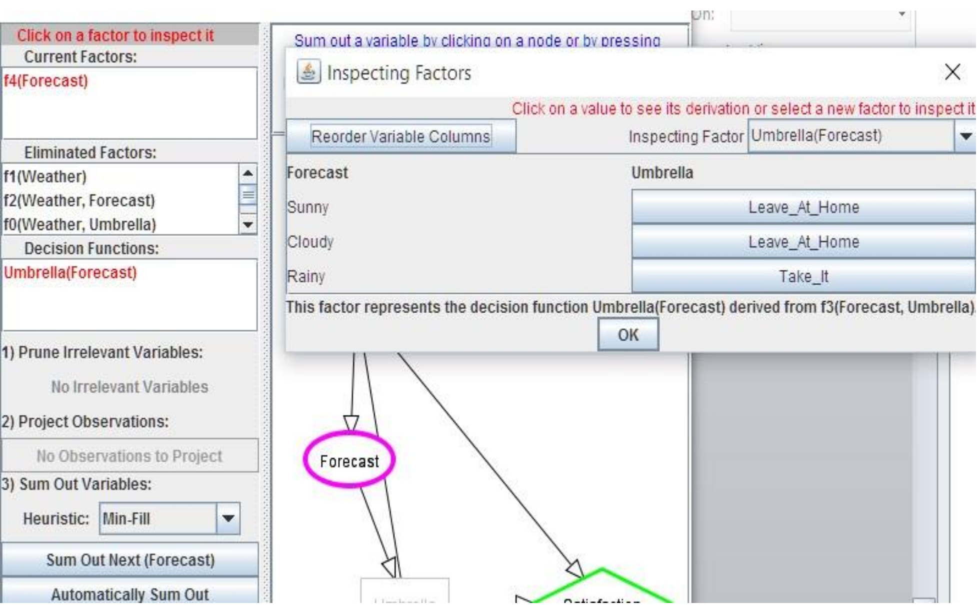


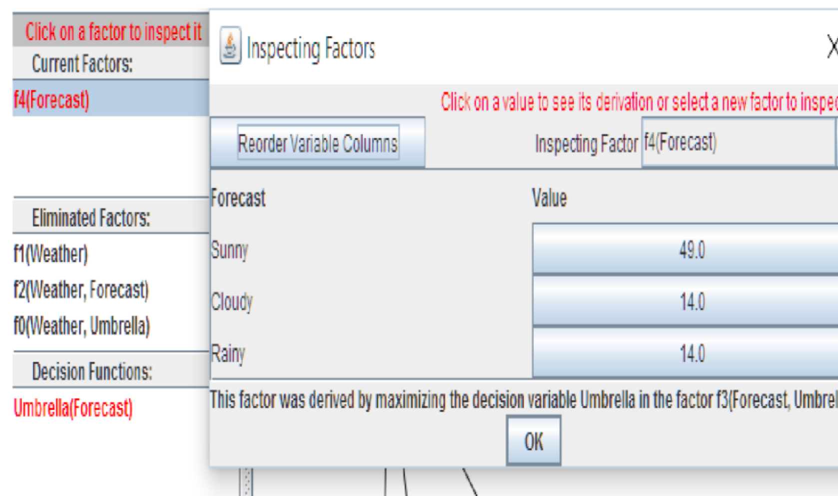
2a: Sum out random variables that are not parents of a decision node

Weather

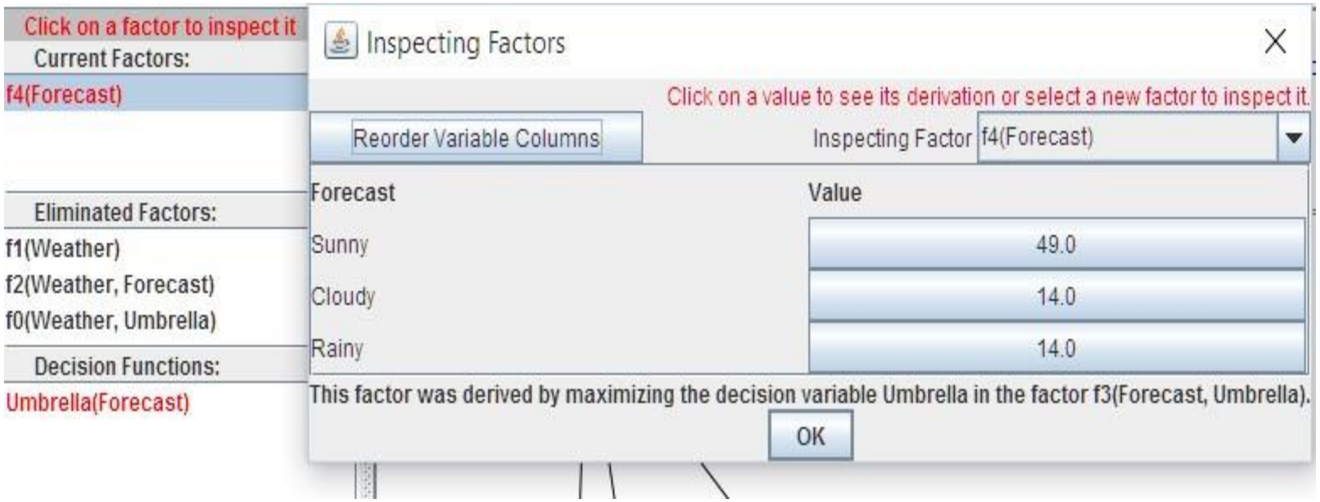


2b Max out last decision variable D in the total ordering

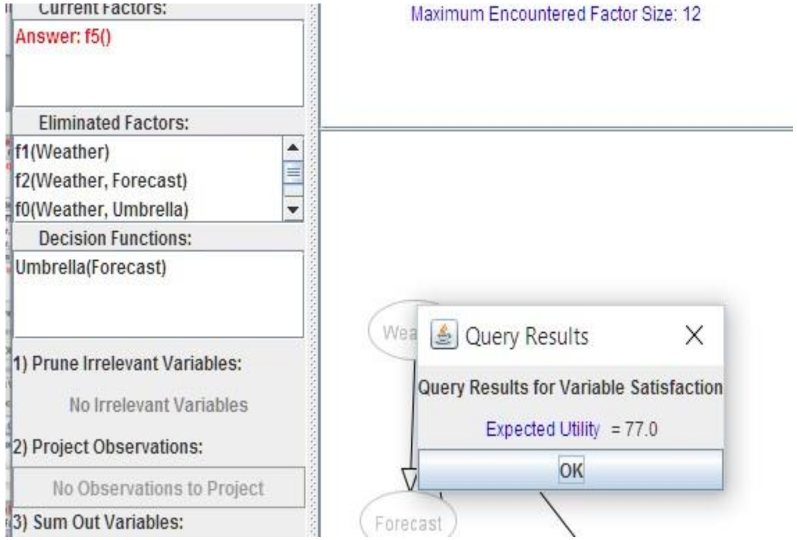




Actual utility of each decision value for Umbrella



this is the expected utility of the optimal policy. Sum out any remaining variable:

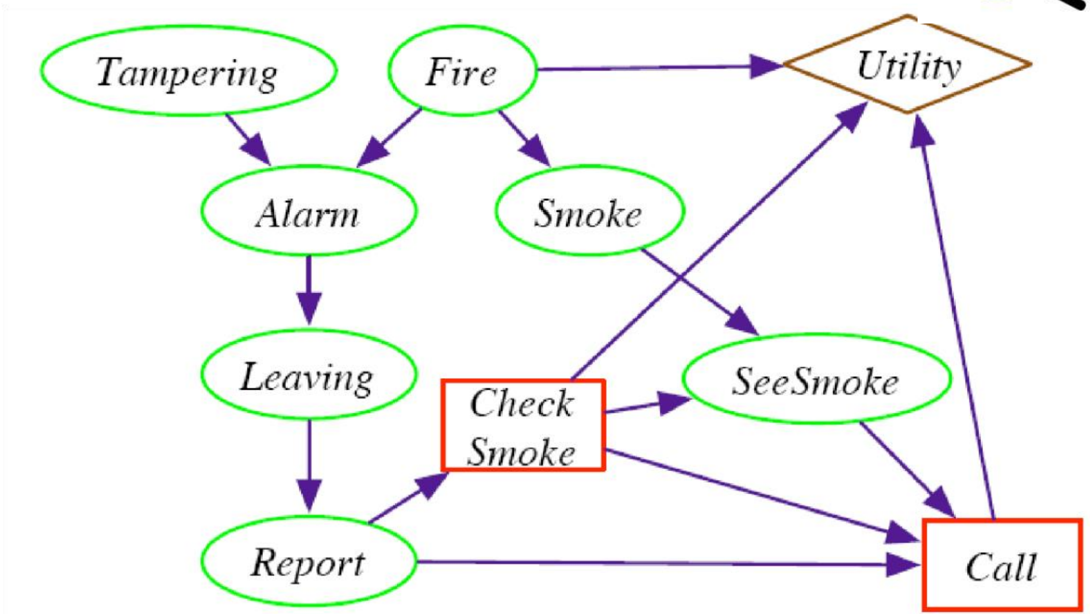


Finding optimal policies with VE

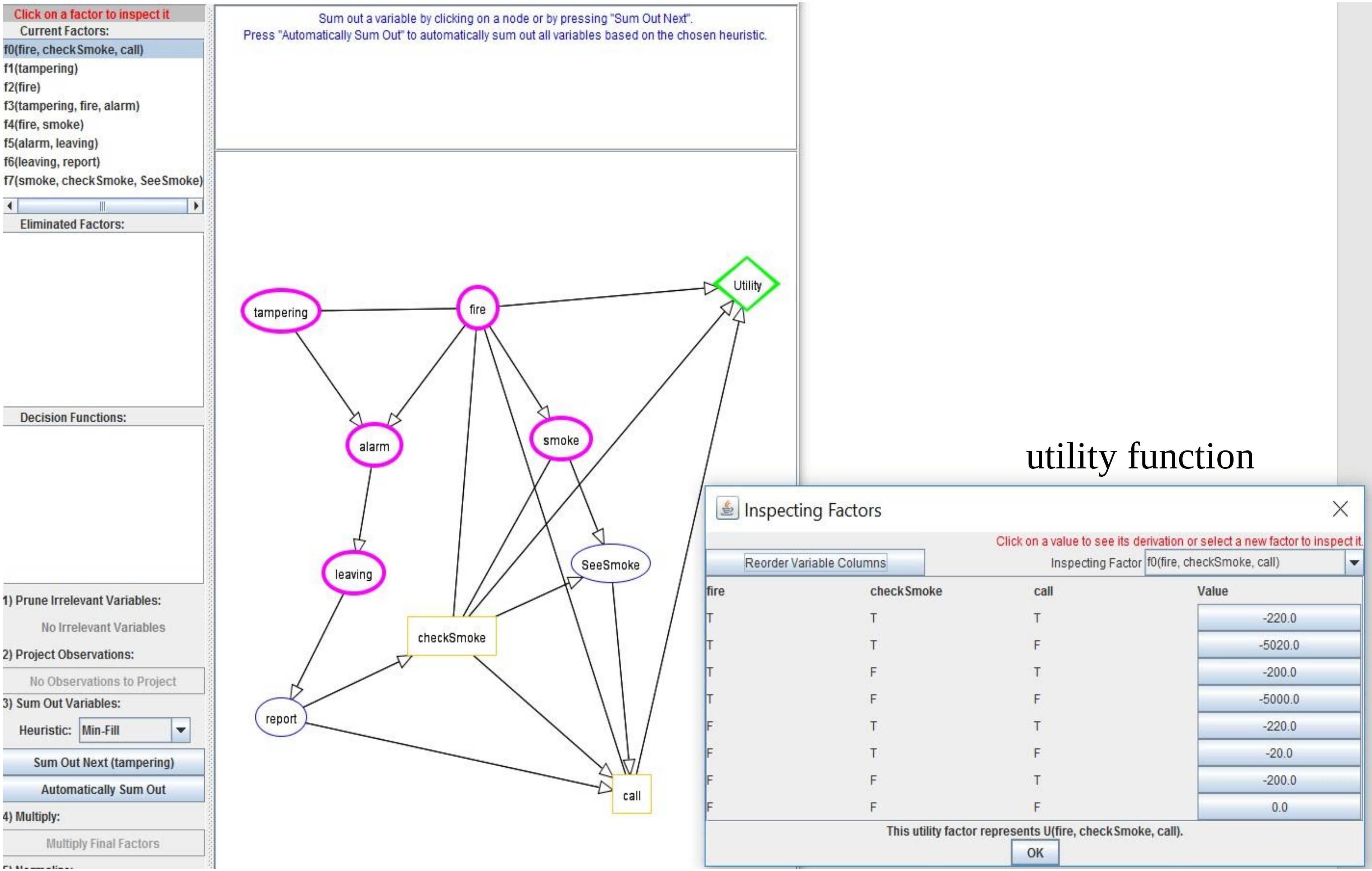
1. Create a factor for each CPT and a factor for the utility
2. While there are still decision variables

- 2a: Sum out random variables that are not parents of a decision node.
- 2b: Max out last decision variable D in the total ordering
  - ✓ Keep track of decision function

3. Sum out any remaining variable:
   this is the expected utility of the optimal policy.



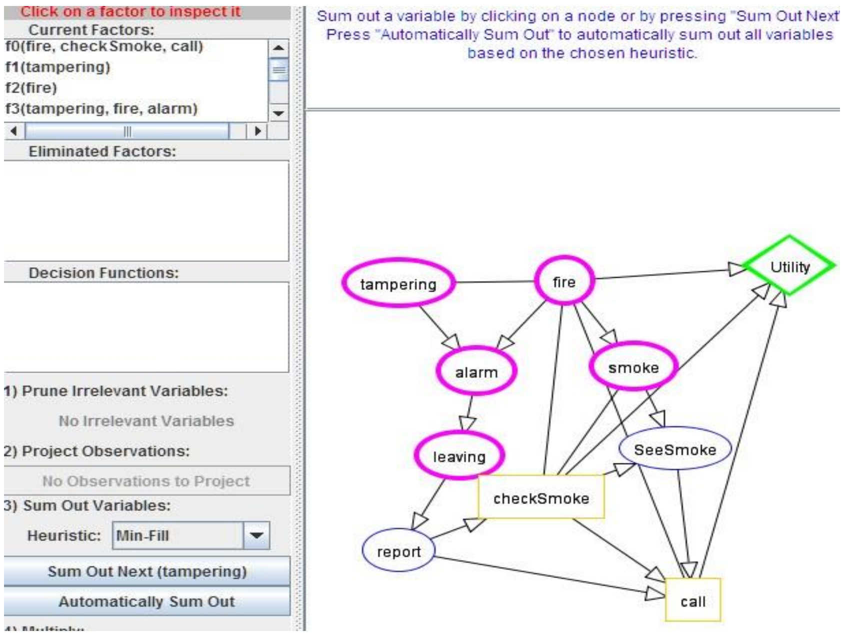
Now let's look at a more complex example



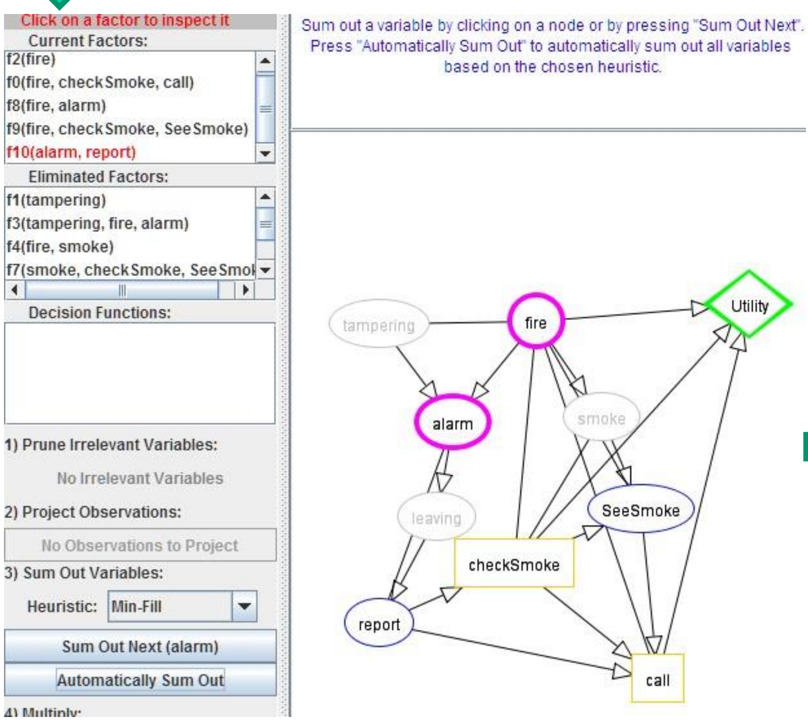
utility function

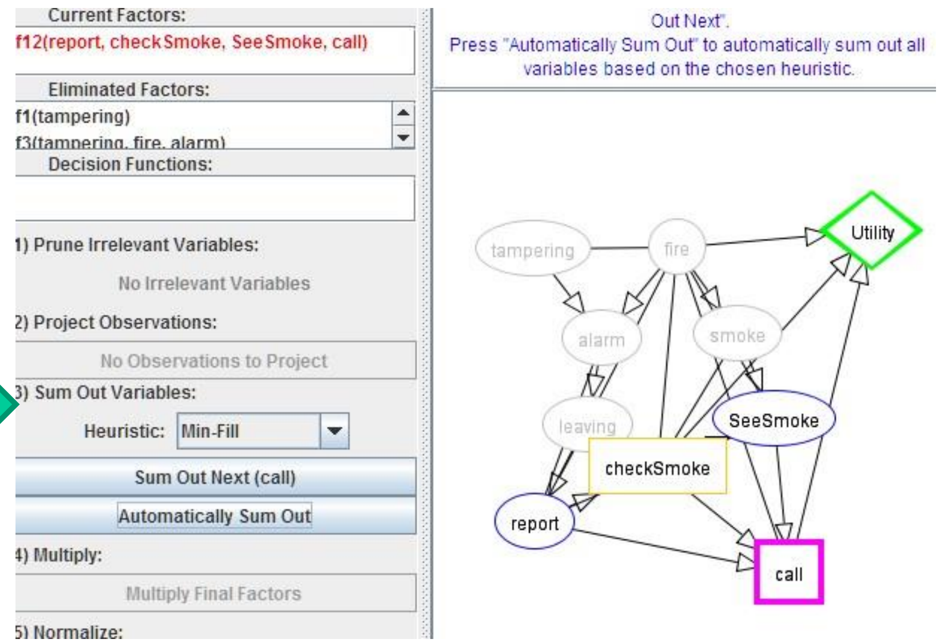
1. Create a factor for each CPT and a factor for the utility

2a: Sum out random variables that are not parents of a decision node (pink nodes in the applet)



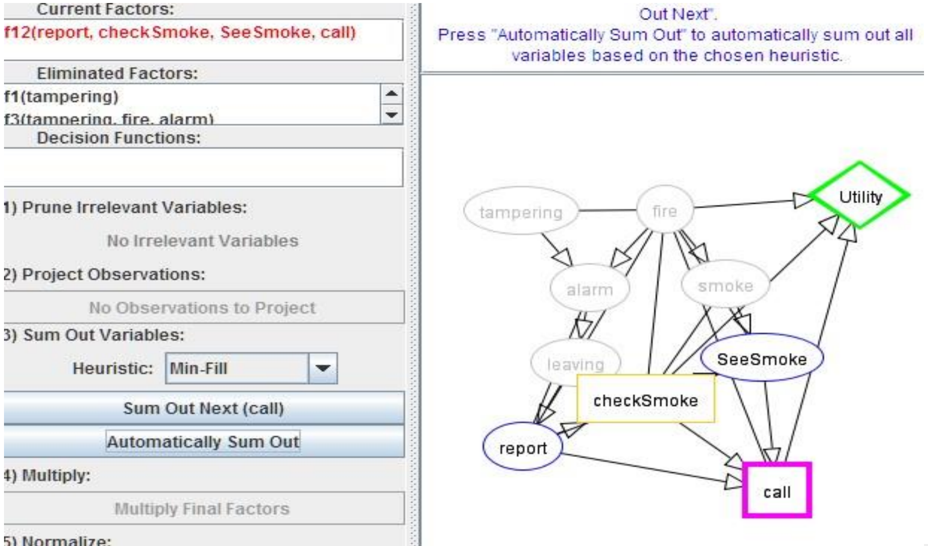
After summing out leaving, tampering and smoke





Max out last decision variable: step 2b details

- Select a variable D that corresponds to the latest decision to be made
- this variable will appear in a factor that only contains that variable and (some of) its parents (for the noforgetting condition)
- Eliminate Dby maximizing. This returns:
- The optimal decision function for D, arg $\max_D f$
- A new factor to use in VE, $\max_D f$
- Repeat till there are no more decision nodes.



2b

Select a variable D that corresponds to the latest decision to be made

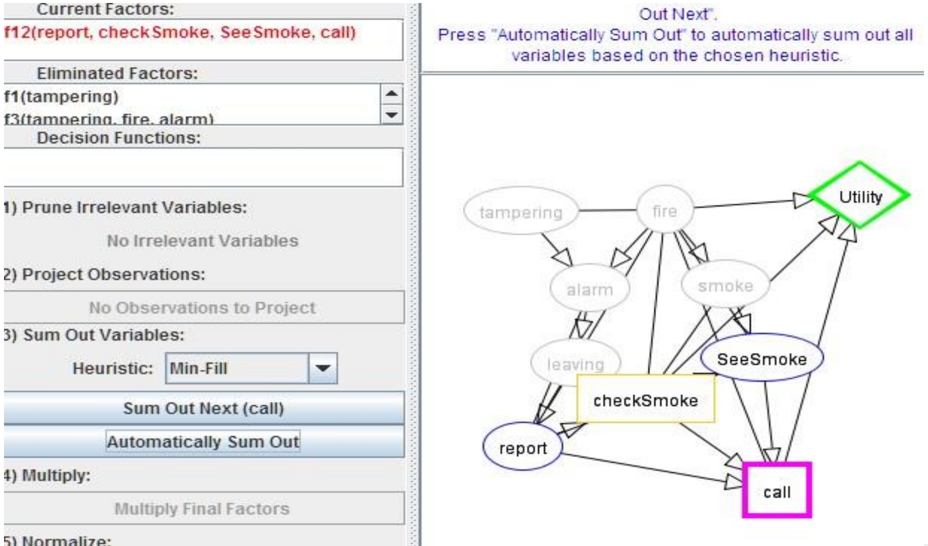
this variable will appear in a factor that only contains that variable and (some of) its parents (for the no-forgetting condition), no children

Here ???

Reorder Variable Columns — Inspecting Factor f12(report, checkSmoke, SeeSmoke, call)

report	checkSmoke	SeeSmoke	call	Value
T	T	T	T	-1.33129
T	T	T	F	-29.29547
T	T	F	T	-4.85646
T	T	F	F	-3.6831
T	F	T	T	0.0
T	F	T	F	0.0
T	F	F	T	-5.62523
T	F	F	F	-32.41604
F	T	T	T	-2.82671
F	T	T	F	-16.08253
F	T	F	T	-210.98554
F	T	F	F	-20.9389
F	F	T	T	0.0
F	F	T	F	0.0
F	F	F	T	-194.37477
F	F	F	F	-17.58396

This factor was derived by eliminating "fire" from the factor(s) f2(fire), f0(fire, checkSmoke, call), f9(fire, checkSmoke, SeeSmoke), f11(fire, report).



2b

Select a variable D that corresponds to the latest decision to be made

this variable will appear in a factor that only contains that variable and (some of) its parents (for the no-forgetting condition), no children

Here, "call"

Reorder Variable Columns

Inspecting Factor f12(report, checkSmoke, SeeSmoke, call)

report	checkSmoke	SeeSmoke	call	Value
T	T	T	T	-1.33129
T	T	T	F	-29.29547
T	T	F	T	-4.85646
T	T	F	F	-3.6831
T	F	T	T	0.0
T	F	T	F	0.0
T	F	F	T	-5.62523
T	F	F	F	-32.41604
F	T	T	T	-2.82671
F	T	T	F	-16.08253
F	T	F	T	-210.98554
F	T	F	F	-20.9389
F	F	T	T	0.0
F	F	T	F	0.0
F	F	F	T	-194.37477
F	F	F	F	-17.58396

This factor was derived by eliminating "fire" from the factor(s) f2(fire), f0(fire, checkSmoke, call), f9(fire, checkSmoke, SeeSmoke), f11(fire, report).

Eliminate the decision Variables: step 2b details

- Eliminate D("call" here) by maximizing. This returns:
- The optimal decision function for D, arg $\max_D f$

Reorder Variable Columns   Inspecting Factor f12(report, checkSmoke, SeeSmoke, call)

report	checkSmoke	SeeSmoke	call	Value
T	T	T	T	-1.33129
T	T	T	F	-29.29547
T	T	F	T	-4.85646
T	T	F	F	-3.6831
T	F	T	T	0.0
T	F	T	F	0.0
T	F	F	T	-5.62523
T	F	F	F	-32.41604
F	T	T	T	-2.82671
F	T	T	F	-16.08253
F	T	F	T	-210.98554
F	T	F	F	-20.9389
F	F	T	T	0.0
F	F	T	F	0.0
F	F	F	T	-194.37477
F	F	F	F	-17.58396

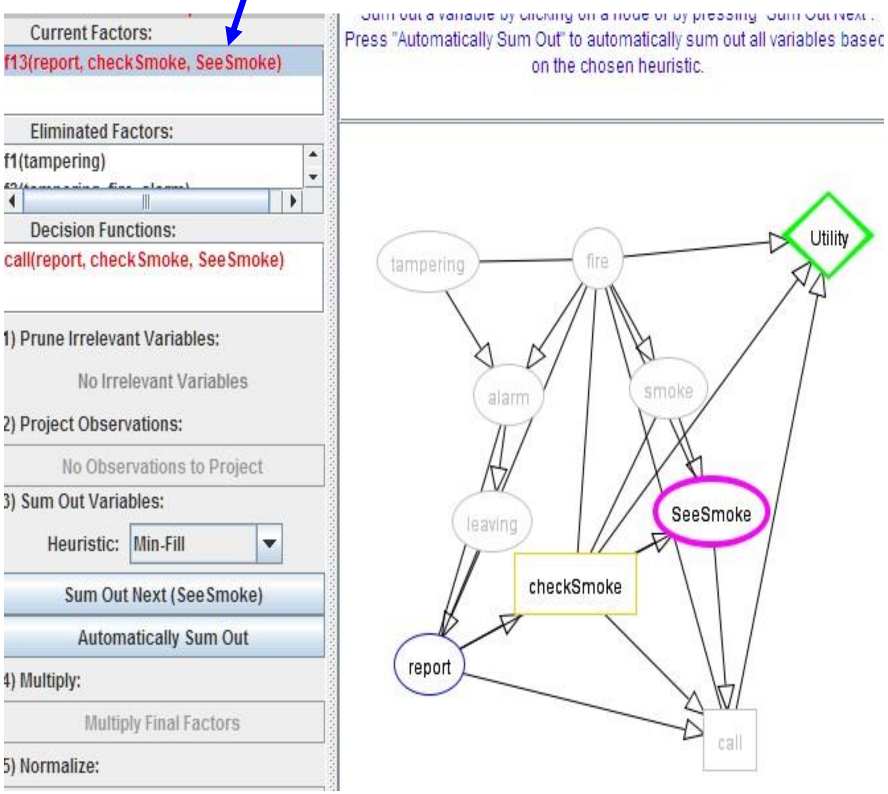
This factor was derived by eliminating "fire" from the factor(s) f2(fire), f0(fire, checkSmoke, call), f9(fire, checkSmoke, SeeSmoke), **f11(fire, report)**.

- A new factor to use in VE, $\max_D f$

Eliminate D (call here) by maximizing. This returns:

The optimal decision function for D, arg $\max_D f$

A new factor to use in VE, $\max_D f$



Inspecting Factors

Click on a value to see its derivation or select a new factor to inspect it.

Reorder Variable Columns | Inspecting Factor call(report, checkSmoke, SeeSmoke)

report	checkSmoke	SeeSmoke	call
T	T	T	T
T	T	F	F
T	F	T	T
T	F	F	T
F	T	T	T
F	T	F	F
F	F	T	T
F	F	F	F

This factor represents the decision function call(report, checkSmoke, SeeSmoke) derived from f12(report, checkSmoke, SeeSmoke, call).

OK

Inspecting Factors

Click on a value to see its derivation or select a new factor to inspect i

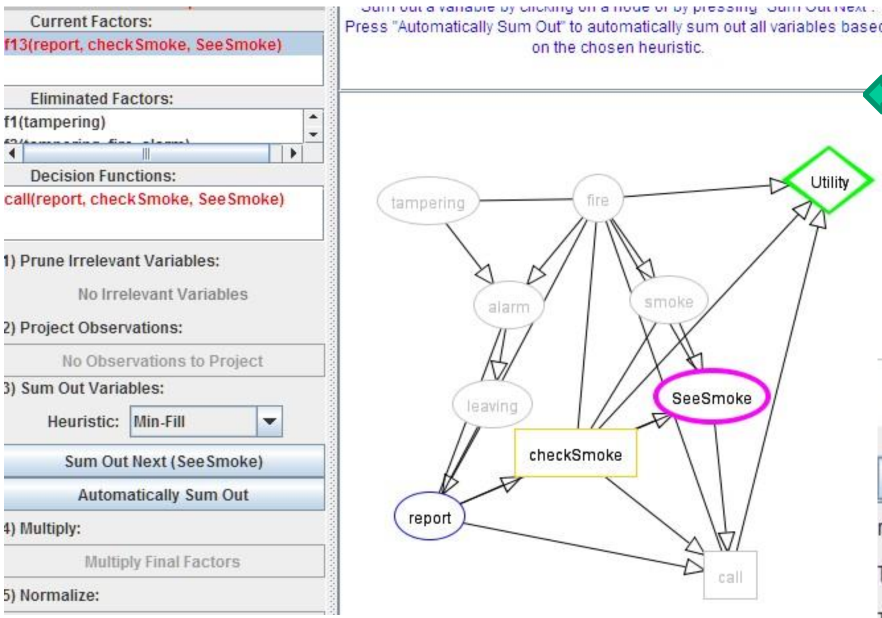
Reorder Variable Columns | Inspecting Factor f13(report, checkSmoke, SeeSmoke)

report	checkSmoke	SeeSmoke	Value
T	T	T	-1.33129
T	T	F	-3.6831
T	F	T	0.0
T	F	F	-5.62523
F	T	T	-2.82671
F	T	F	-20.9389
F	F	T	0.0
F	F	F	-17.58396

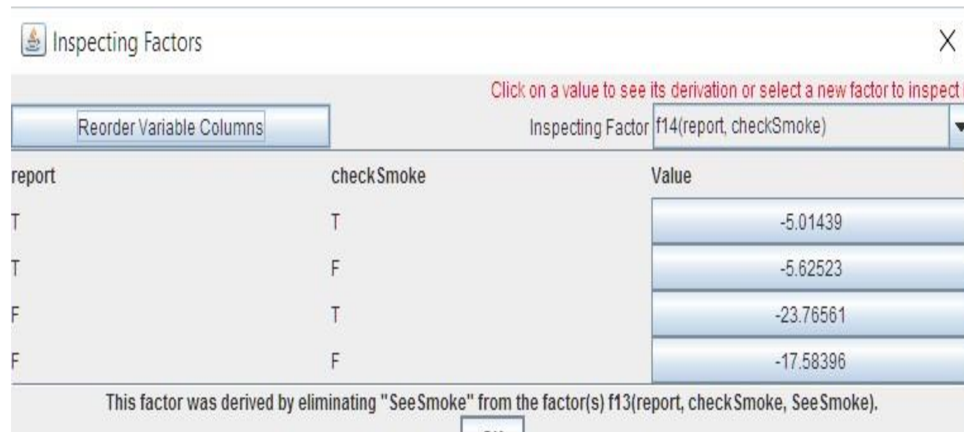
This factor was derived by maximizing the decision variable call in the factor f12(report, checkSmoke, SeeSmoke, call).

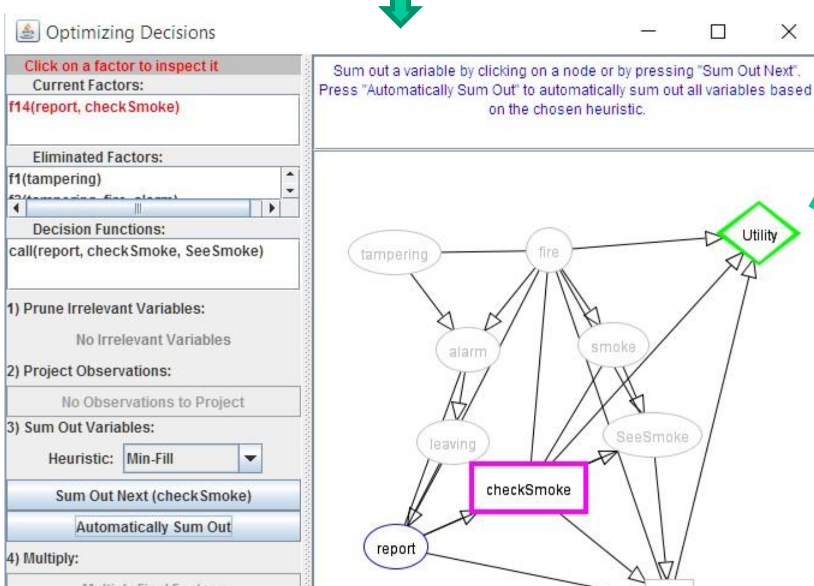
Eliminate the decision Variables: step 2b

details



2a: Sum out random variables that are not parents of a decision node





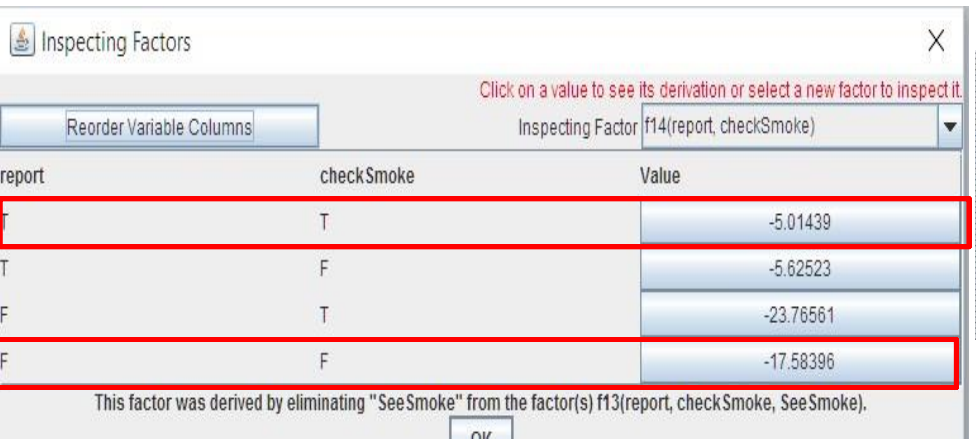
2b

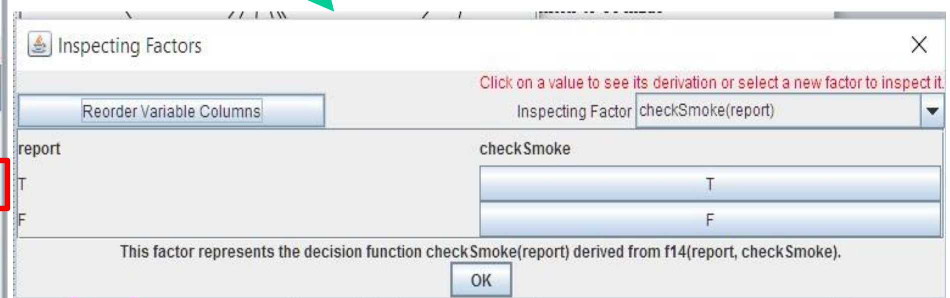
Select a variable D that corresponds to the latest decision to be made:

this variable will appear in a factor that only contains that variable and (some of) its parents (for the no-forgetting

Eliminate the decision Variables: step 2b details

- Eliminate D(here “checkSmoke”) by maximizing. This returns:

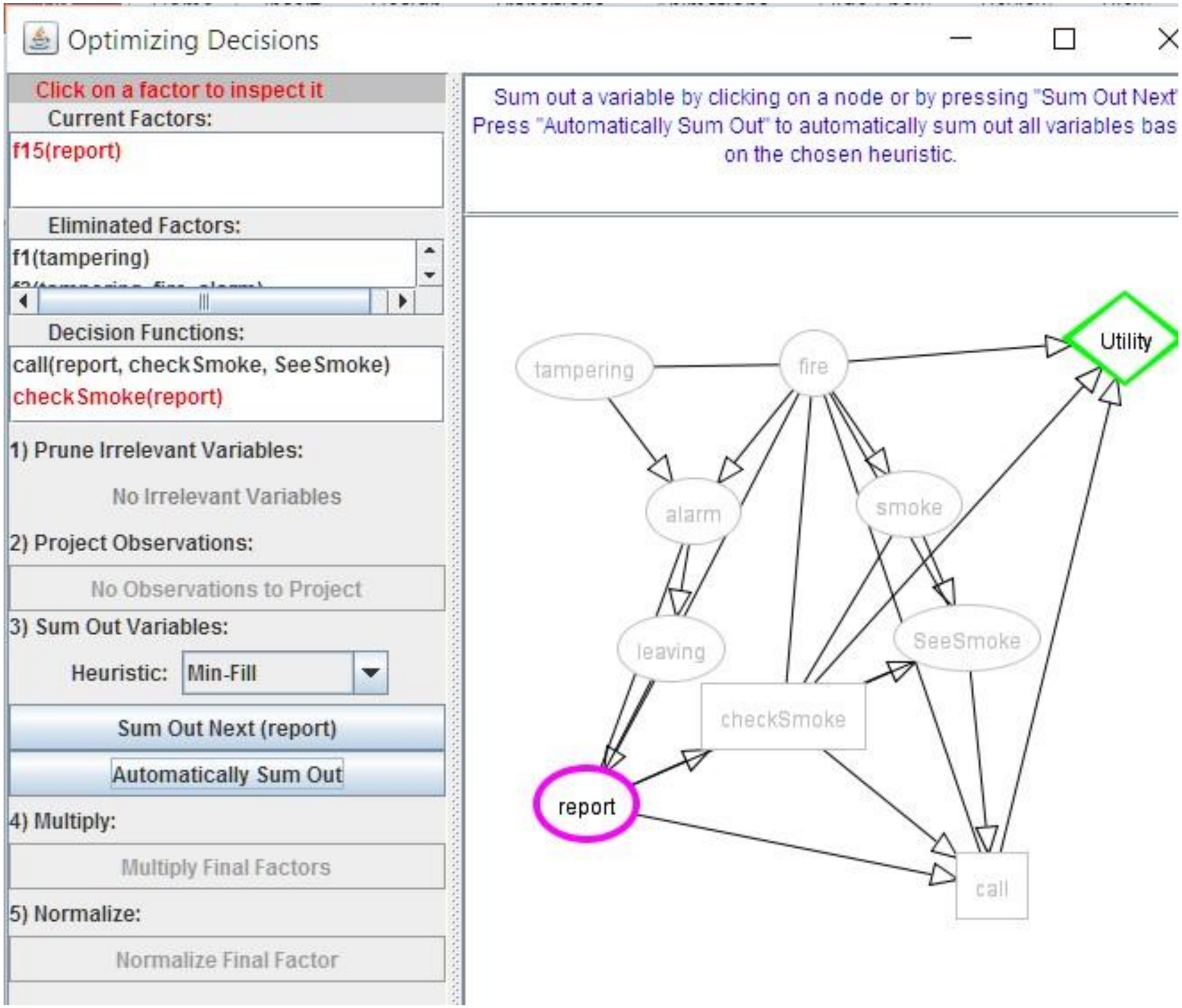


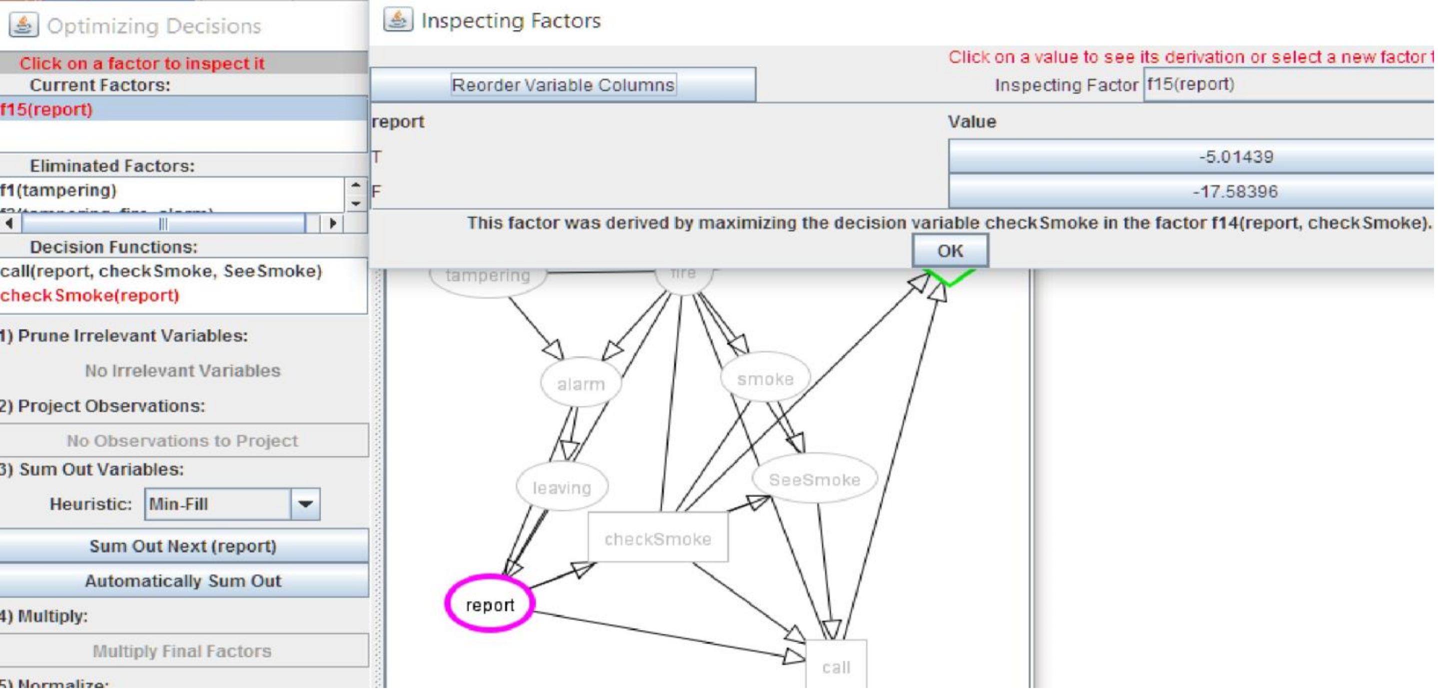


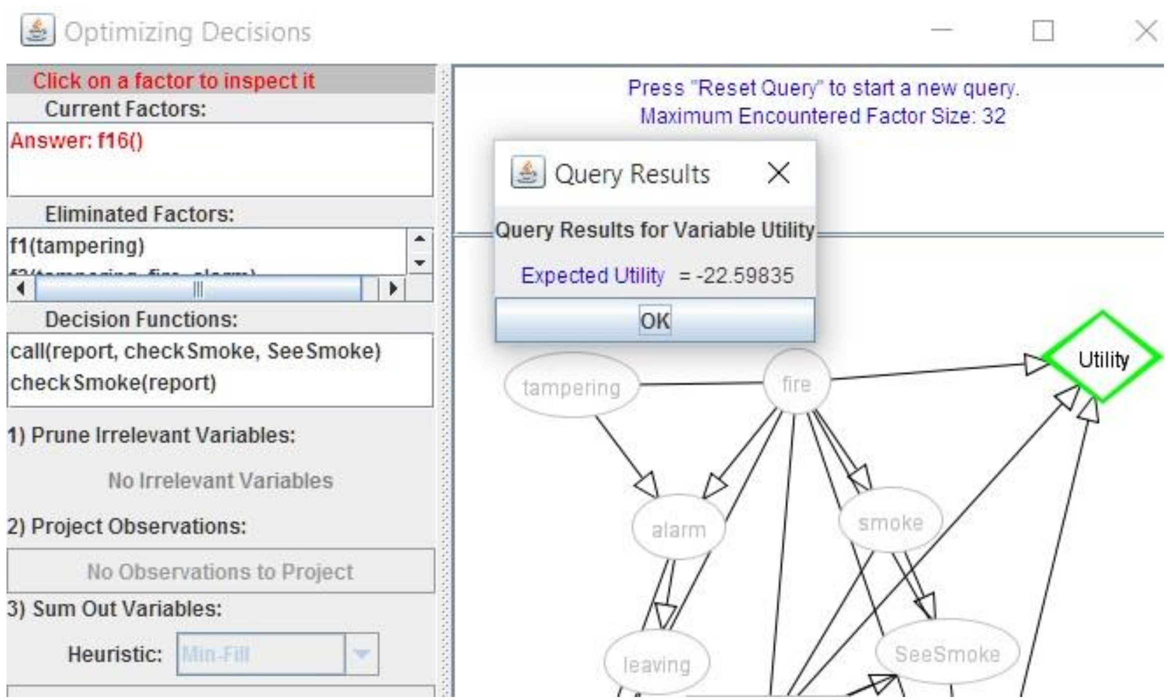
- The optimal decision function for D, arg $\max_D f$ • A new factor to use in VE, $\max_D f$

No more decision variables

No more decision variables




Optimizing Decisions
Click on a factor to inspect it
Current Factors:
f15(report)
Eliminated Factors:
f1(tampering)
Decision Functions:
call(report, checkSmoke, SeeSmoke)
checkSmoke(report)
1) Prune Irrelevant Variables:
No Irrelevant Variables
2) Project Observations:
No Observations to Project
3) Sum Out Variables:
Heuristic: Min-Fill
Sum Out Next (report)
Automatically Sum Out
4) Multiply:
Multiply Final Factors
Inspecting Factors
Reorder Variable Columns
Inspecting Factor f15(report)
report
Value
T
-5.01439
F
-17.58396
This factor was derived by maximizing the decision variable checkSmoke in the factor f14(report, checkSmoke).
OK
tampering
alarm
smoke
leaving
SeeSmoke
checkSmoke
report
call


Optimizing Decisions
Click on a factor to inspect it
Current Factors:
Answer: f16()
Eliminated Factors:
f1(tampering)
Decision Functions:
call(report, checkSmoke, SeeSmoke)
checkSmoke(report)
1) Prune Irrelevant Variables:
No Irrelevant Variables
2) Project Observations:
No Observations to Project
3) Sum Out Variables:
Heuristic: Min-Fill
Press "Reset Query" to start a new query.
Maximum Encountered Factor Size: 32
Query Results
Query Results for Variable Utility
Expected Utility = -22.59835
OK
Utility
tampering
fire
alarm
smoke
leaving
SeeSmoke

this is the expected utility of the optimal policy.

Computational complexity of VE for finding optimal policies

- We saw that for d decision variables (each with k binary parents and b possible actions)
- There are $(b^{2^k})^d$ policies
- All combinations of (b^{2^k}) decision functions per decision

- Variable elimination saves the final exponent:
- Consider each decision function only once
- Resulting complexity: $O(d * b^{2^k})$

- Much faster than enumerating policies (or search in policy space), but still a double exponential • Solution: approximation algorithms for finding optimal policies (beyond the scope of this course)

Learning Goals For Decision Under Uncertainty

One-Off decisions

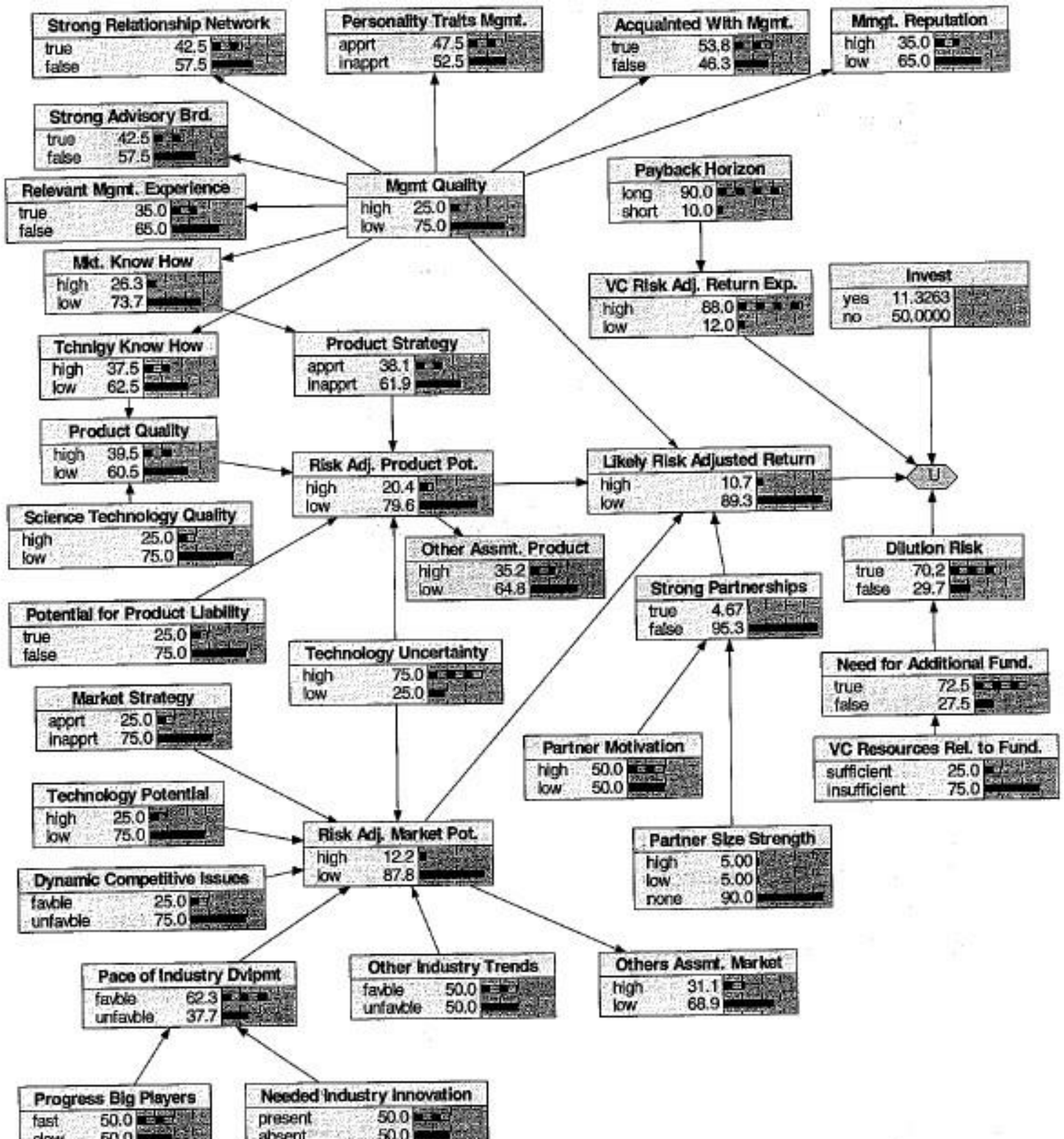
- Compare and contrast stochastic single-stage (one-off) decisions vs. multistage (sequential) decisions
- Define a Utility Function on possible worlds
- Define and compute optimal one-off decisions
- Represent one-off decisions as single stage decision networks
- Compute optimal decisions by Variable Elimination Sequential decision networks

- Represent sequential decision problems as decision networks
- Explain the non-forgetting property Policies
- Verify whether a possible world satisfies a policy
- Define the expected utility of a policy
- Compute the number of policies for a decision problem
- Compute the optimal policy by Variable Elimination for a One Off Decision and for sequential decision problems with one decision variable
- Compute the optimal policy by Variable Elimination for sequential decisions in

(just general ideas)

Decision Theory: Decision Support Systems

Support for management: e.g. hiring



Source: *R.E. Neapolitan, 2007*

Decision Theory: Decision Support Systems

Computational Sustainability: New interdisciplinary field, AI is a key component

- Models and methods for decision making concerning the management and allocation of resources
- to solve most challenging problems related to sustainability, E.g.
  - ✓Energy: when and where to produce green energy most economically?
  - ✓Which parcels of land to purchase to protect endangered species?
  - ✓Urban planning: how to use budget for best development in 30 years?

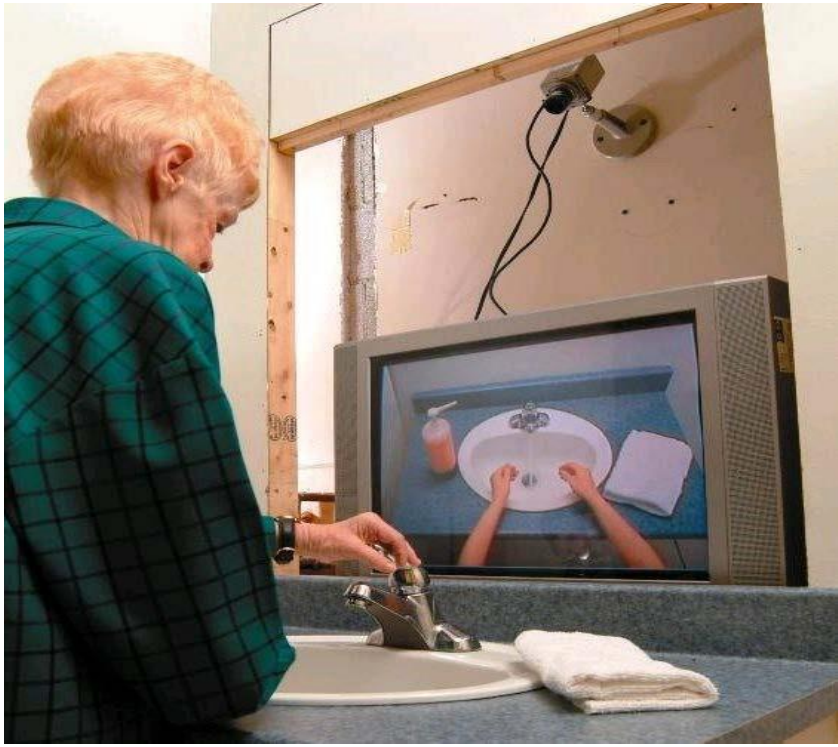
Source: http://www.computational-sustainability.org/

Planning Under Uncertainty

- Learning and Using models of Patient-Caregiver Interactions During Activities of Daily Living

- by using POMDP (an extension of decision networks that model the temporal evolution of the world)

- Goal: Help older adults living with cognitive disabilities (such as Alzheimer's) when they:
- forget the proper sequence of tasks that need to be completed
- lose track of the steps that they have already completed



Source: *Jesse Hoey UofT 2007*

We are done!

Representation

Environment

Problem Type Deterministic Arc Stochastic ReasoningTechnique

Consistency

Constraint Satisfaction ***VarsConstraints+*** Search

Static ***Belief Nets***

Query ***Logics*** Variable

Search

Elimination

Sequential ***STRIPS Decision Nets*** Variable Planning Search

Elimination