Decision Theory: VE for Decision Networks, Sequential Decisions, Optimal Policies for Sequential Decisions

Alan Mackworth

UBC CS 322 – Decision Theory 3

April 3, 2013

Textbook §9.2.1, 9.3

Announcements (1)

- Assignment 4 was due today.
- The list of short questions for the final is online … please use it!
- Please submit suggested review topics on Connect for review lecture(s).
- Previous final has been posted.
- Additional review lecture(s) and TA hours will be scheduled before the final, if needed.
- TA hours to continue as scheduled during exam period, unless as posted otherwise to Connect.
- Exercise 12, for single-stage Decision Networks, and Exercise 13, for multi-stage Decision Networks, have been posted on the home page along with AIspace auxiliary files.

Announcements (2)

- Teaching Evaluations are online
  - You should have received a message about them
  - Secure, confidential, mobile access
- Your feedback is important!
  - Allows us to assess and improve the course material
  - I use it to assess and improve my teaching methods
  - The department as a whole uses it to shape the curriculum
  - Teaching evaluation results are important for instructors
    - Appointment, reappointment, tenure, promotion and merit, salary
  - UBC takes them very seriously (now)
  - Evaluations close at 11:59PM on April 9, 2013.
    - Before exam, but instructors can't see results until *after* we submit grades
  - Please do it!
- Take a few minutes and visit https://eval.olt.ubc.ca/science

Lecture Overview

- Recap: Single-Stage Decision Problems
  - Single-Stage decision networks
  - Variable elimination (VE) for computing the optimal decision
- Sequential Decision Problems
  - General decision networks
  - Policies
- Expected Utility and Optimality of Policies
- Computing the Optimal Policy by Variable Elimination
- Summary & Perspectives

Recap: Single vs. Sequential Actions

- Single Action (aka One-Off Decisions)
  - One or more primitive decisions that can be treated as a single macro decision to be made before acting

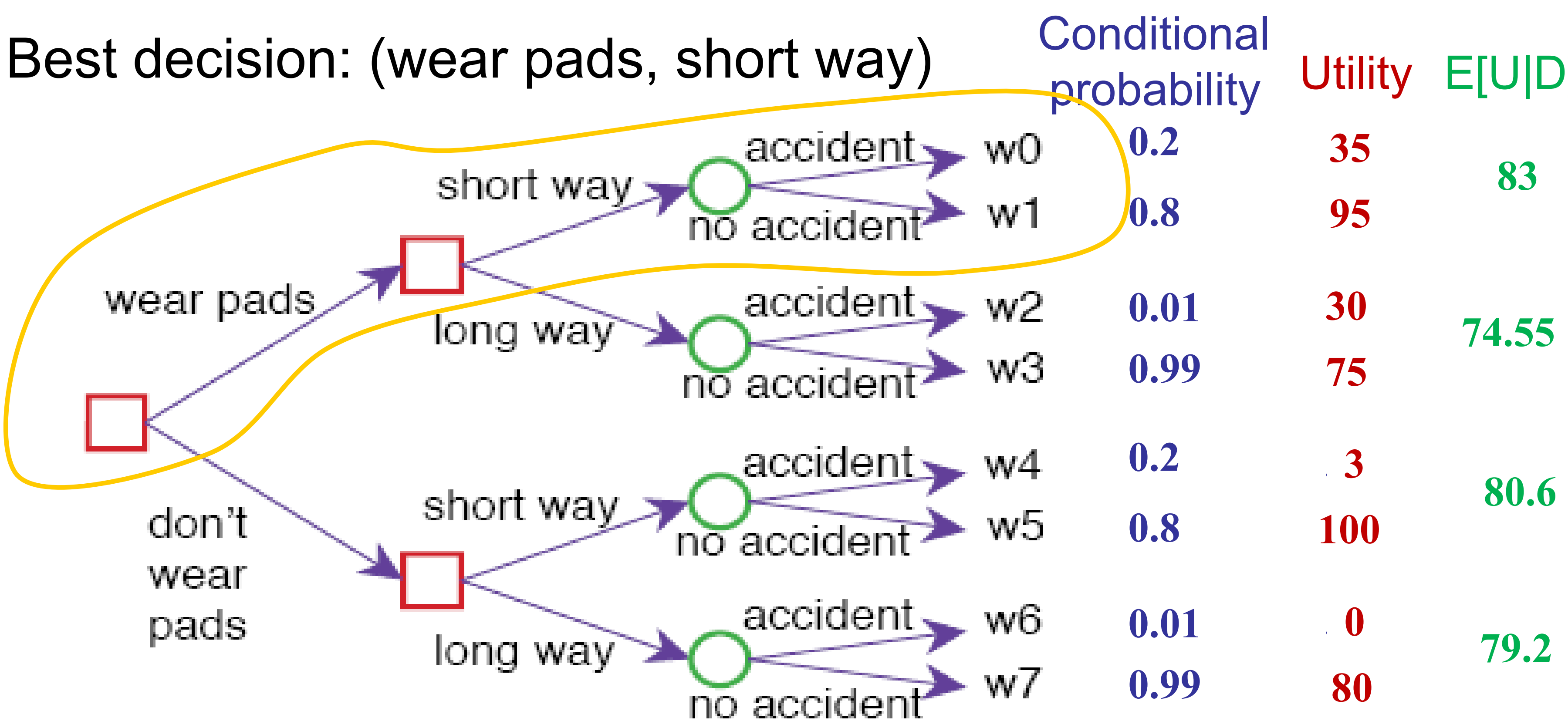
- Sequence of Actions (Sequential Decisions)
  - Repeat:
    - observe
    - act
  - Agent has to take actions not knowing what the future brings

Recap: Optimal single-stage decisions

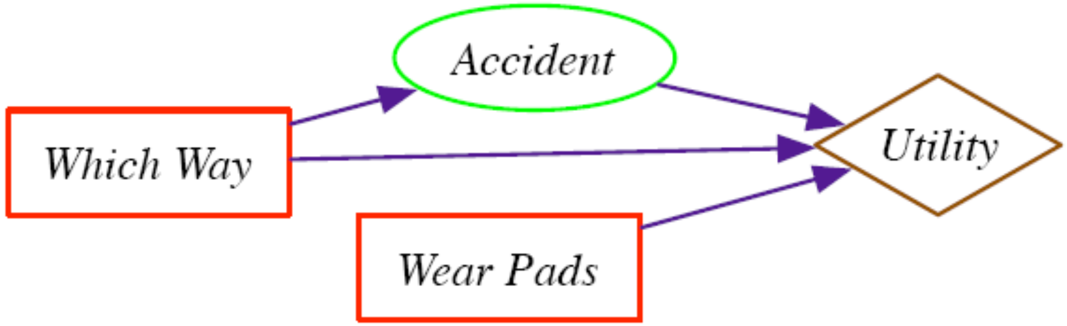
Definition (optimal single-stage decision)
An optimal single-stage decision is the decision $D{=}d_{max}$ whose expected value is maximal:

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

Best decision: (wear pads, short way)



Recap: Single-Stage decision networks

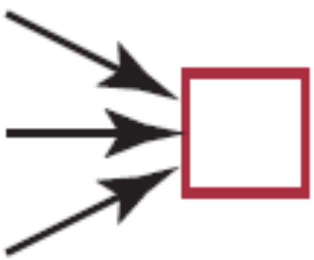


- Compact and explicit representation
  - Compact: each random/decision variable only occurs once
  - Explicit: dependences are made explicit
    - e.g., which variables affect the probability of an accident?

- Extension of Bayesian networks with
  - Decision variables
  - A single utility node

Recap: Types of nodes in decision networks

- A random variable is drawn as an ellipse.
  - Parents pa(X): encode dependence
    Conditional probability p(X | pa(X))
    Random variable X is conditionally independent of its non-descendants given its parents
  - Domain: the values it can take at random



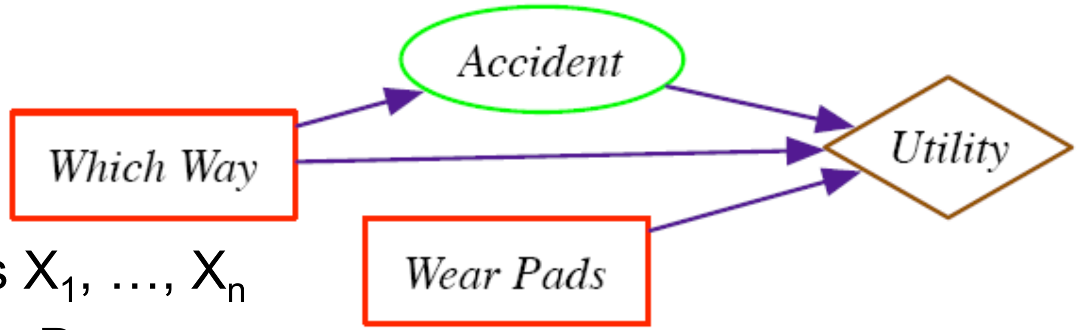
- A decision variable is drawn as an rectangle.
  - Parents pa(D)
    information available when decision D is made
    - Single-stage: pa(D) only includes decision variables
  - Domain: the values the agents can choose (actions)

- A utility node is drawn as a diamond.
  - Parents pa(U): variables utility directly depends on
    - utility U(pa(U)) for each instantiation of its parents
  - Domain: does not have a domain!

Lecture Overview

- Recap: Single-Stage Decision Problems
  - Single-Stage decision networks
  - Variable elimination (VE) for computing the optimal decision
- Sequential Decision Problems
  - General decision networks
  - Policies
- Expected Utility and Optimality of Policies
- Computing the Optimal Policy by Variable Elimination
- Summary & Perspectives

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)

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

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

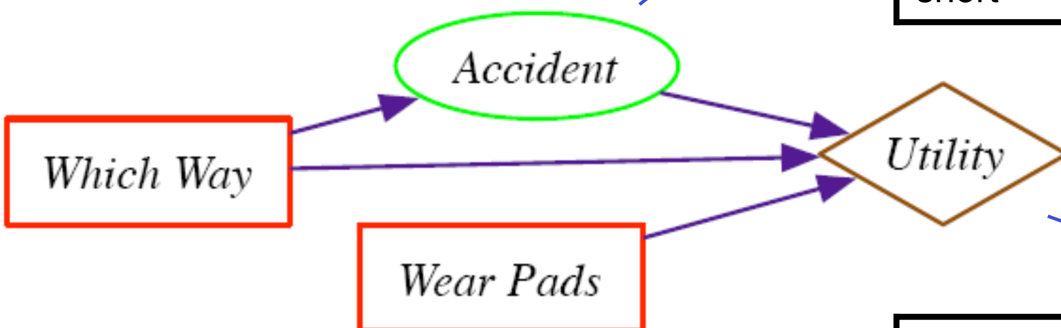
- To find the optimal decision we can use VE:
  1. Create a factor for each conditional probability and for the utility
  2. Sum out all random variables, one at a time
     - This creates a factor on D that gives the expected utility for each d_i
  3. Choose the d_i with the maximum value in the factor

VE Example: Step 1, create initial factors

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

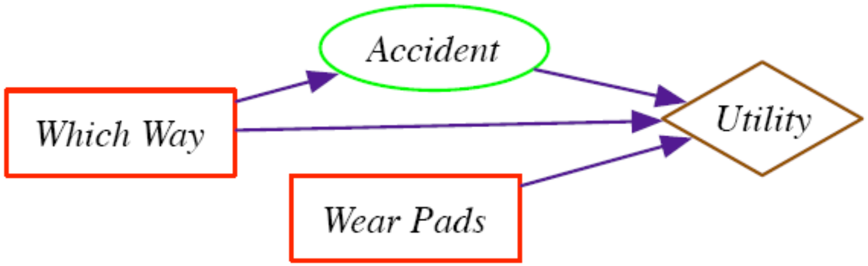


$f_2(A,W,P)$

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

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

VE example: step 2, sum out A

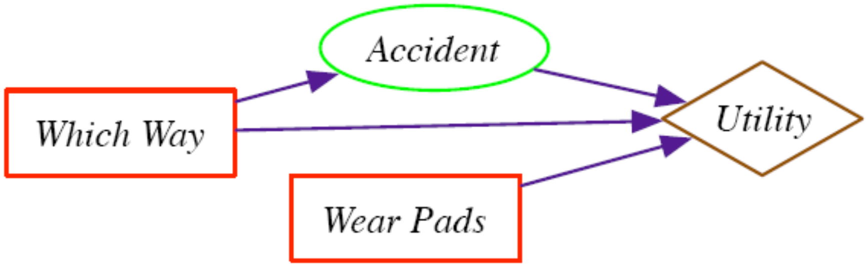


Step 2a: compute product
$f_1(A,W) \times f_2(A,W,P)$

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

f (A,W)   f(A,P)   f(A)   f(A,P,W)

VE example: step 2, sum out A



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

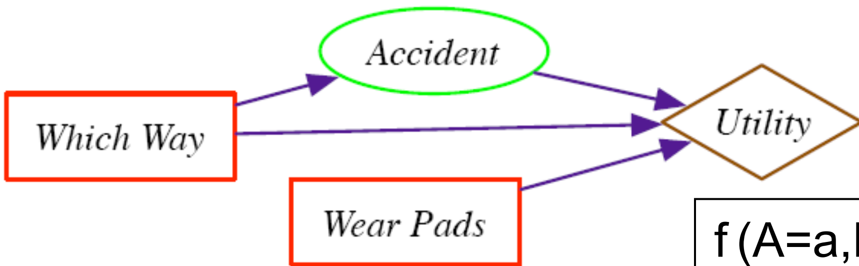
What is the right form for the product $f_1(A,W) \times f_2(A,W,P)$?

•It is $f(A,P,W)$:
the domain of the product is the union of the multiplicands' domains

•$f(A,P,W) = f_1(A,W) \times f_2(A,W,P)$

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

VE example: step 2, sum out A



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

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

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

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

Which way W	Accident A	Pads P	f(A,W,P)
long	true	true	0.01 * 30
long	true	false	
long	false	true	
long	false	false	???
short	true	true	
short	true	false	
short	false	true	
short	false	false	

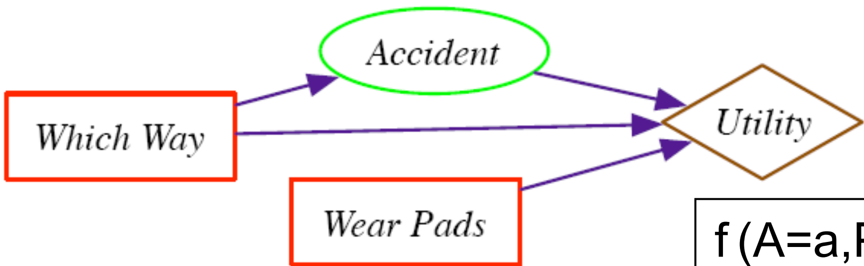
0.99 * 30

0.01 * 80

0.99 * 80

0.8 * 30

VE example: step 2, sum out A



Step 2a: compute product

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

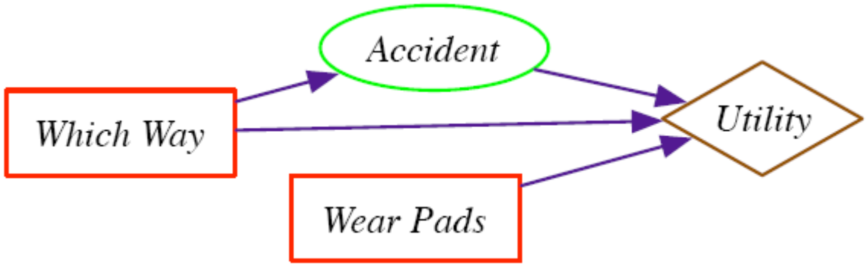
$f(A=a,P=p,W=w) = f_1(A=a,W=w) \times f_2(A=a,W=w,P=p)$

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

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

Which way W	Accident A	Pads P	f(A,W,P)
long	true	true	0.01 * 30
long	true	false	0.01*0
long	false	true	0.99*75
long	false	false	0.99*80
short	true	true	0.2*35
short	true	false	0.2*3
short	false	true	0.8*95
short	false	false	0.8*100

VE example: step 2, sum out A



Step 2b: sum A out of the product f(A,W,P):

$$f_3(W,P) = \sum_A f(A,W,P)$$

Which way W	Pads P	$f_3(W,P)$
long	true	0.01*30+0.99*75=74.55
long	false	
short	true	??
short	false	

Which way W	Accident A	Pads P	f(A,W,P)
long	true	true	0.01 * 30
long	true	false	0.01*0
long	false	true	0.99*75
long	false	false	0.99*80
short	true	true	0.2*35
short	true	false	0.2*3
short	false	true	0.8*95
short	false	false	0.8*100

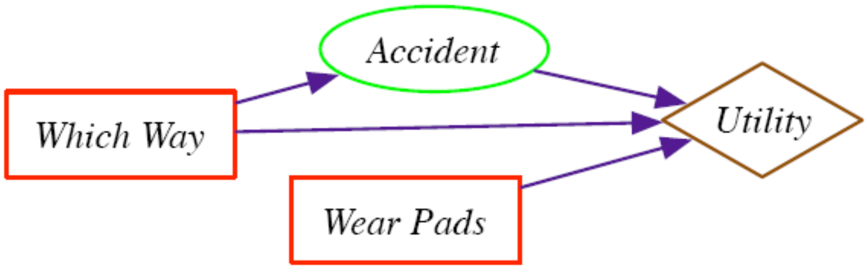
0.2*35 + 0.2*0.3

0.2*35 + 0.8*95

0.99*80 + 0.8*95

0.8 * 95 + 0.8*100

VE example: step 2, sum out A



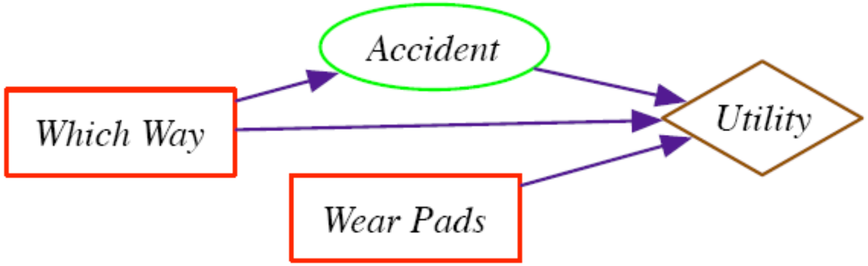
Step 2b: sum A out of the product f(A,W,P):

$$f_3(W,P) = \sum_A f(A,W,P)$$

Which way W	Pads P	$f_3(W,P)$
long	true	0.01*30+0.99*75=74.55
long	false	0.01*0+0.99*80=79.2
short	true	0.2*35+0.8*95=83
short	false	0.2*3+0.8*100=80.6

Which way W	Accident A	Pads P	f(A,W,P)
long	true	true	0.01 * 30
long	true	false	0.01*0
long	false	true	0.99*75
long	false	false	0.99*80
short	true	true	0.2*35
short	true	false	0.2*3
short	false	true	0.8*95
short	false	false	0.8*100

VE example: step 3, choose decision with max E(U)



Step 2b: sum A out of the product f(A,W,P):

$$f_3(W,P) = \sum_A f(A,W,P)$$

Which way W	Accident A	Pads P	f(A,W,P)
long	true	true	0.01 * 30
long	true	false	0.01*0
long	false	true	0.99*75
long	false	false	0.99*80
short	true	true	0.2*35
short	true	false	0.2*3
short	false	true	0.8*95
short	false	false	0.8*100

Which way W	Pads P	f_3(W,P)
long	true	0.01*30+0.99*75=74.55
long	false	0.01*0+0.99*80=79.2
short	true	0.2*35+0.8*95=83
short	false	0.2*3+0.8*100=80.6

The final factor encodes the expected utility of each decision

- Thus, taking the short way but wearing pads is the best choice, with an expected utility of 83

Variable Elimination for Single-Stage Decision Networks: Summary

1. Create a factor for each conditional probability and for the utility
2. Sum out all random variables, one at a time
   – 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

This is Algorithm OptimizeSSDN, in P&M, Section 9.2.1, p.387

Learning Goals So Far For 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

Lecture Overview

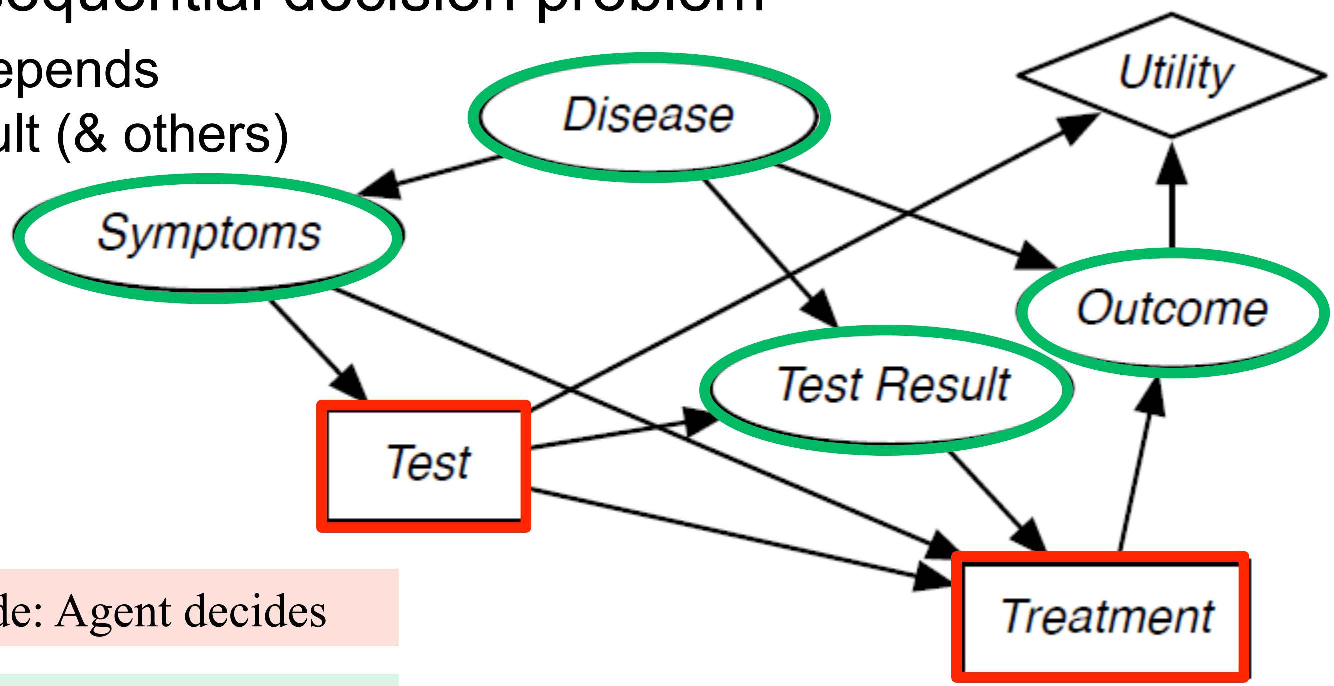
- Recap: Single-Stage Decision Problems
  - Single-Stage decision networks
  - Variable elimination (VE) for computing the optimal decision
- Sequential Decision Problems
  - General decision networks
  - Policies
- Expected Utility and Optimality of Policies
- Computing the Optimal Policy by Variable Elimination
- Summary & Perspectives

Sequential Decision Problems

- An intelligent agent doesn't make a multi-step decision and carry it out blindly
  - It would take new observations it makes into account
- A more typical scenario:
  - The agent observes, acts, observes, acts, …
- 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, spying
- General Decision networks:
  - Just like single-stage decision networks, with one exception: the parents of decision nodes can include random variables

Sequential Decision Problems: Example

- Example for sequential decision problem
  - Treatment depends on Test Result (& others)



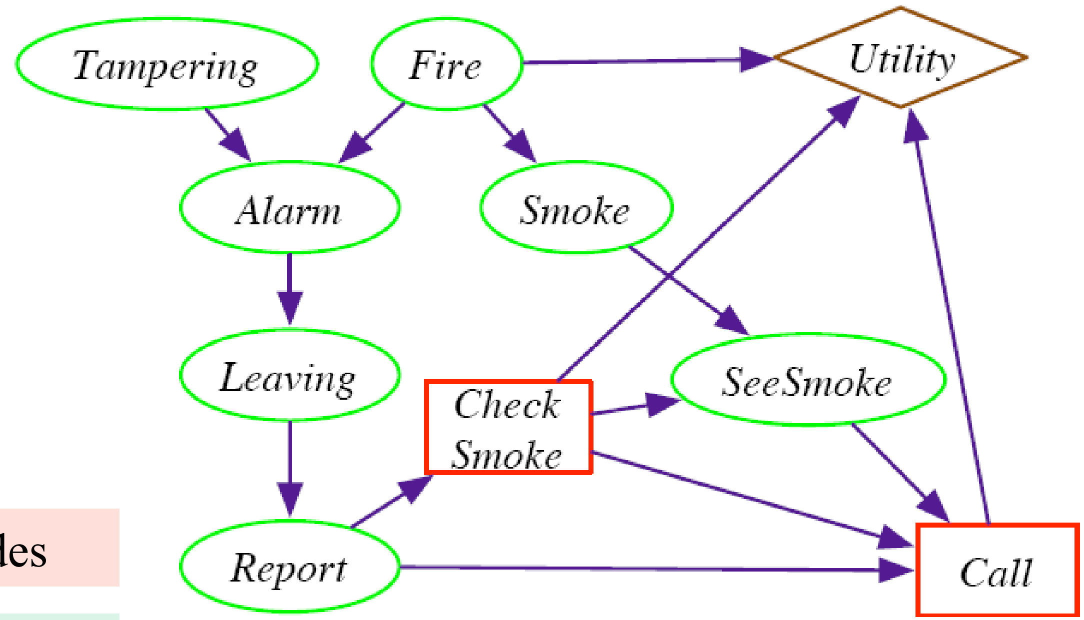
Decision node: Agent decides

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(Test) = {Symptoms}
  - pa(Treatment) = {Test, Symptoms, TestResult}

Sequential Decision Problems: Example

- Another example for sequential decision problems
  - Call depends on Report and SeeSmoke (and on CheckSmoke)

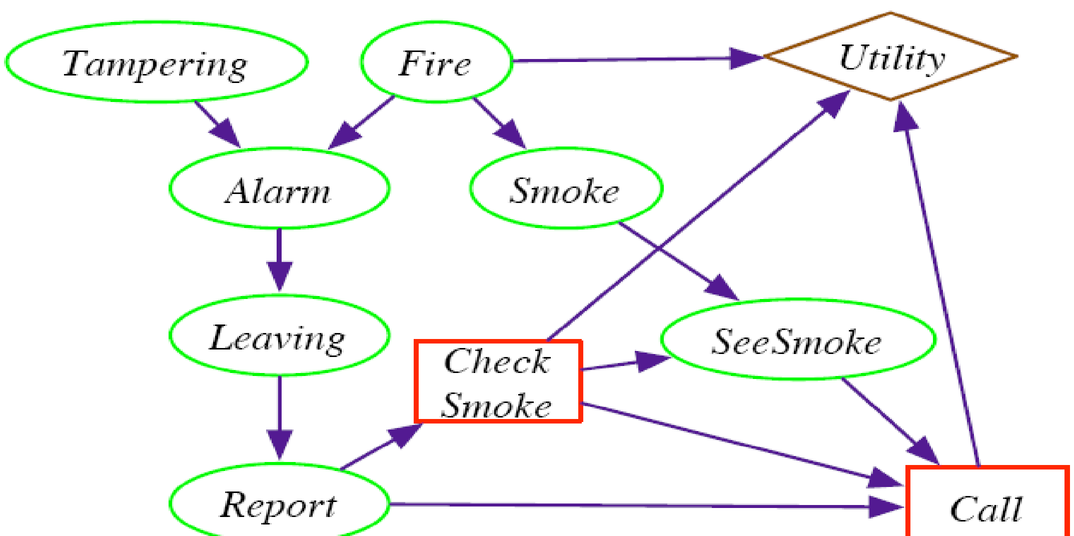


Decision node: Agent decides

Chance node: Chance decides

Sequential Decision Problems

- What should an agent do?
  - What an agent should do depends on what it will do in the future
    - E.g. agent only needs to check for smoke if that will affect whether it calls
  - What an agent does in the future depends on what it did before
    - E.g. when making the decision it needs to know whether it checked for smoke
  - We will get around this problem as follows
    - The agent has a conditional plan of what it will do in the future
    - We will formalize this conditional plan as a policy

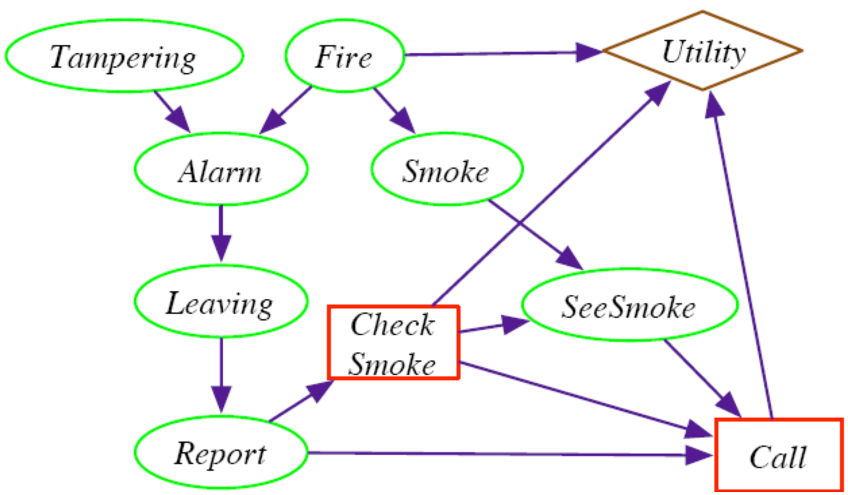


Policies for Sequential Decision Problems

Definition (Policy)
A policy is a sequence of $\delta_1, \ldots\ldots, \delta_n$ decision functions

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

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



There are $2^2=4$ possible decision functions δ_{cs} for Check Smoke:

- Decision function needs to specify a value for each instantiation of parents

CheckSmoke

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

Call

Policies for Sequential Decision Problems

Definition (Policy)
A policy π is a sequence of δ_1 ,….., δ_n decision functions

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

I.e., when the agent has observed $o \in \mathrm{dom}(pD_i)$, it will do $\delta_i(o)$

There are 2^8=256 possible decision functions δ_{cs} for Call:

	R=t, CS=t, SS=t	*R=t, CS=t, SS=f*	*R=t, CS=f, SS=t*	*R=t, CS=f, SS=f*	*R=f, CS=t, SS=t*	*R=f, CS=t, SS=f*	*R=f, CS=f, SS=t*	*R=f, CS=f, SS=f*
δ_{call}1(R)	T	T	T	T	T	T	T	T
δ_{call}2(R)	T	T	T	T	T	T	T	F
δ_{call}3(R)	T	T	T	T	T	T	F	T
δ_{call}4(R)	T	T	T	T	T	T	F	F
δ_{call}5(R)	T	T	T	T	T	F	T	T
…	…	…	…	…	…	…	…	…
δ_{call}256(R)	F	F	F	F	F	F	F	F

How many policies are there?

- If a decision D has k binary parents, how many assignments of values to the parents are there?

$2k$ $2+k$ k^2 2^k

How many policies are there?

- If a decision D has k binary parents, how many assignments of values to the parents are there?
  - 2^k

- If there are b possible value for a decision variable, how many different decision functions are there for it if it has k binary parents?

2^{kp} | $b*2^k$ | b^{2^k} | 2^{k^b}

How many policies are there?

- If a decision D has k binary parents, how many assignments of values to the parents are there?
  - 2^k
- If there are b possible value for a decision variable, how many different decision functions are there for it if it has k binary parents?
  - b^{2^k}, because there are 2^k possible instantiations for the parents and for every instantiation of those parents, the decision function could pick any of b values
- If there are *d* decision variables, each with *k* binary parents and *b* possible actions, how many policies are there?

db^k $\quad$ b^{dk} $\quad$ $d(b^{2^k})$ $\quad$ $(b^{2^k})^d$

How many policies are there?

- If a decision D has k binary parents, how many assignments of values to the parents are there?
  - 2^k
- If there are b possible value for a decision variable, how many different decision functions are there for it if it has k binary parents?
  - b^{2^k}, because there are 2^k possible instantiations for the parents and for every instantiation of those parents, the decision function could pick any of b values
- If there are *d* decision variables, each with *k* binary parents and *b* possible actions, how many policies are there?
  - $(b^{2^k})^d$, 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: Single-Stage Decision Problems
  - Single-Stage decision networks
  - Variable elimination (VE) for computing the optimal decision
- Sequential Decision Problems
  - General decision networks
  - Policies
- Expected Utility and Optimality of Policies
- Computing the Optimal Policy by Variable Elimination
- Summary & Perspectives

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)

- Consider our previous example policy:
  - Check smoke (i.e. set CheckSmoke=true) if and only if Report=true
  - Call if and only if Report=true, CheckSmoke=true, SeeSmoke=true
- Does the following possible world satisfy this policy?

  ¬tampering, fire, alarm, leaving, report, smoke, checkSmoke, seeSmoke, call

  Yes No

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)

- Consider our previous example policy:
  - Check smoke (i.e. set CheckSmoke=true) if and only if Report=true
  - Call if and only if Report=true, CheckSmoke=true, SeeSmoke=true

- Do the following possible worlds satisfy this policy?

  ¬tampering, fire, alarm, leaving, report, smoke, checkSmoke, seeSmoke, call
    - Yes! Conditions are satisfied for each of the policy's decision functions

  ¬tampering, fire, alarm, leaving, report, smoke, checkSmoke, seeSmoke, ¬call

  Yes No

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)

- Consider our previous example policy:
  - Check smoke (i.e. set CheckSmoke=true) if and only if Report=true
  - Call if and only if Report=true, CheckSmoke=true, SeeSmoke=true
- Do the following possible worlds satisfy this policy?

  ¬tampering, fire, alarm, leaving, report, smoke, checkSmoke, seeSmoke, call
  - Yes! Conditions are satisfied for each of the policy's decision functions

  ¬tampering, fire, alarm, leaving, report, smoke, checkSmoke, seeSmoke, ¬call
  - No! The policy says to call if Report and CheckSmoke and SeeSmoke all true

  ¬tampering,fire,alarm,leaving,¬report,¬smoke,¬checkSmoke,¬seeSmoke,¬call
  - Yes! Policy says to neither check smoke nor call when there is no report

Yes No

Expected utility of a policy

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

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

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

$$= \sum_{X_1,\ldots,X_n,D_1,\ldots,D_m} \prod_{i=1}^{n} P(X_i | pa(X_i)) \prod_{j=1}^{m} (\delta_j(pa(D_j)) = D_j)\ U(pa(U))$$

This term is zero if D_j's value does not agree with what the policy dictates given D_j's parents.

Optimality of a policy

Definition (expected utility of a policy)

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

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

Definition (optimal policy)

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

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

Lecture Overview

- Recap: Single-Stage Decision Problems
  - Single-Stage decision networks
  - Variable elimination (VE) for computing the optimal decision
- Sequential Decision Problems
  - General decision networks
  - Policies
- Expected Utility and Optimality of Policies
- Computing the Optimal Policy by Variable Elimination
- Summary & Perspectives

One last operation on factors: maxing out a variable

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

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

$\max_B f_3(A,B,C) = f_4(A,C)$

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

A	C	$f_4(A,C)$
t	t	0.54
t	f	0.36
f	t	?
f	f	

0.32 0.06 0.48 0.14

One last operation on factors: maxing out a variable

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

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

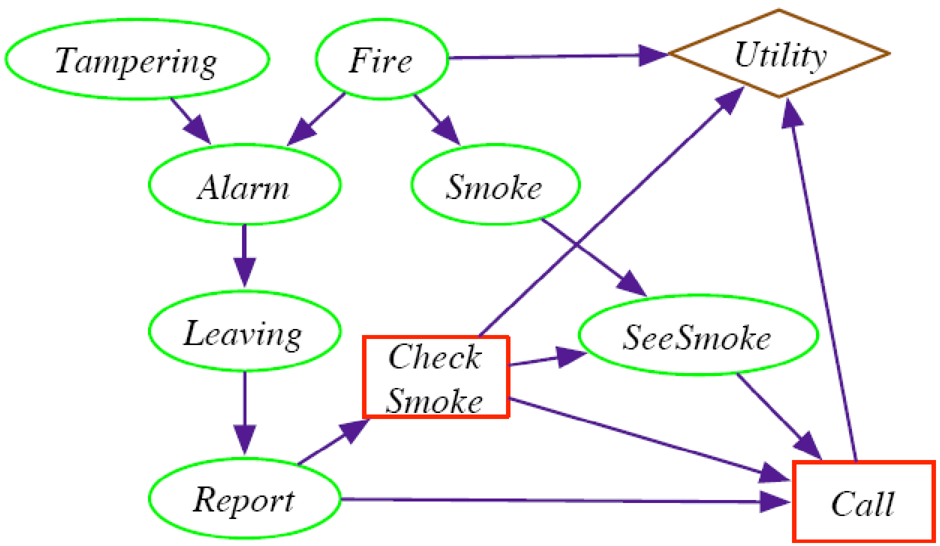
$$\boxed{\max_B f_3(A,B,C) = f_4(A,C)}$$

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

A	C	$f_4(A,C)$
t	t	0.54
t	f	0.36
f	t	0.48
f	f	0.32

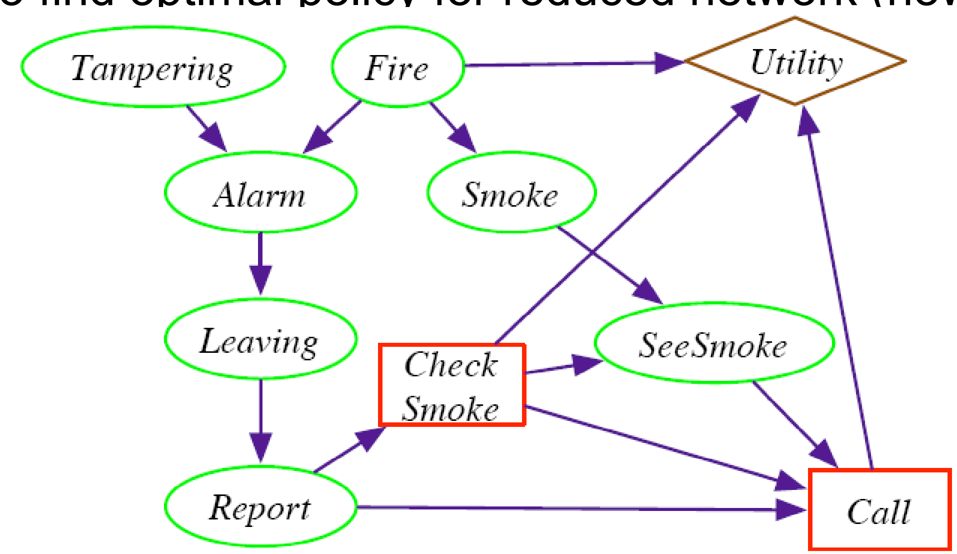
The no-forgetting property

- 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



Idea for finding optimal policies with VE

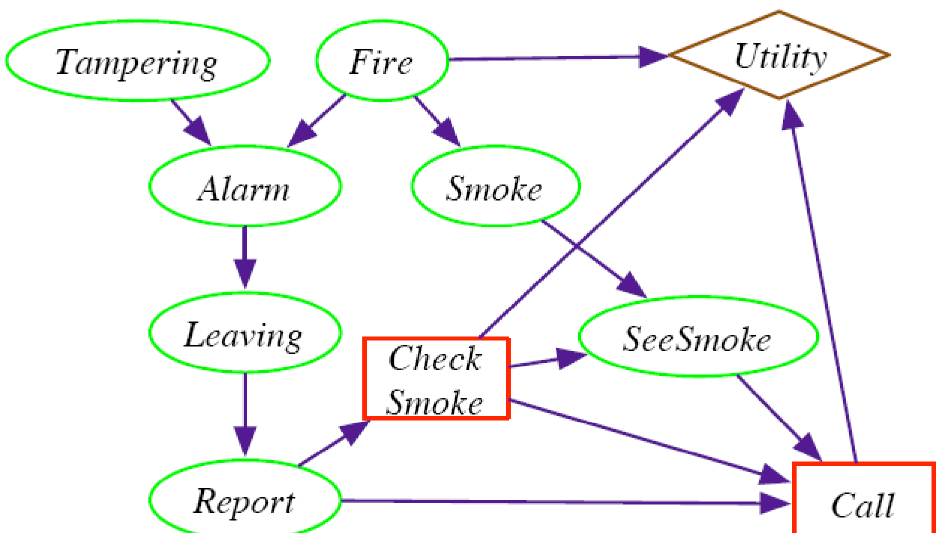
- Idea for finding optimal policies with variable elimination (VE): Dynamic programming: precompute optimal future decisions
  - Consider the last decision D to be made
    - Find optimal decision D=d for each instantiation of D's parents
      - For each instantiation of D's parents, this is just a 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?
    - Recurse to find optimal policy for reduced network (now 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.
     - E.g Tampering, Fire, Alarm, Smoke, Leaving
   - 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



Computational complexity of VE for finding optimal policies

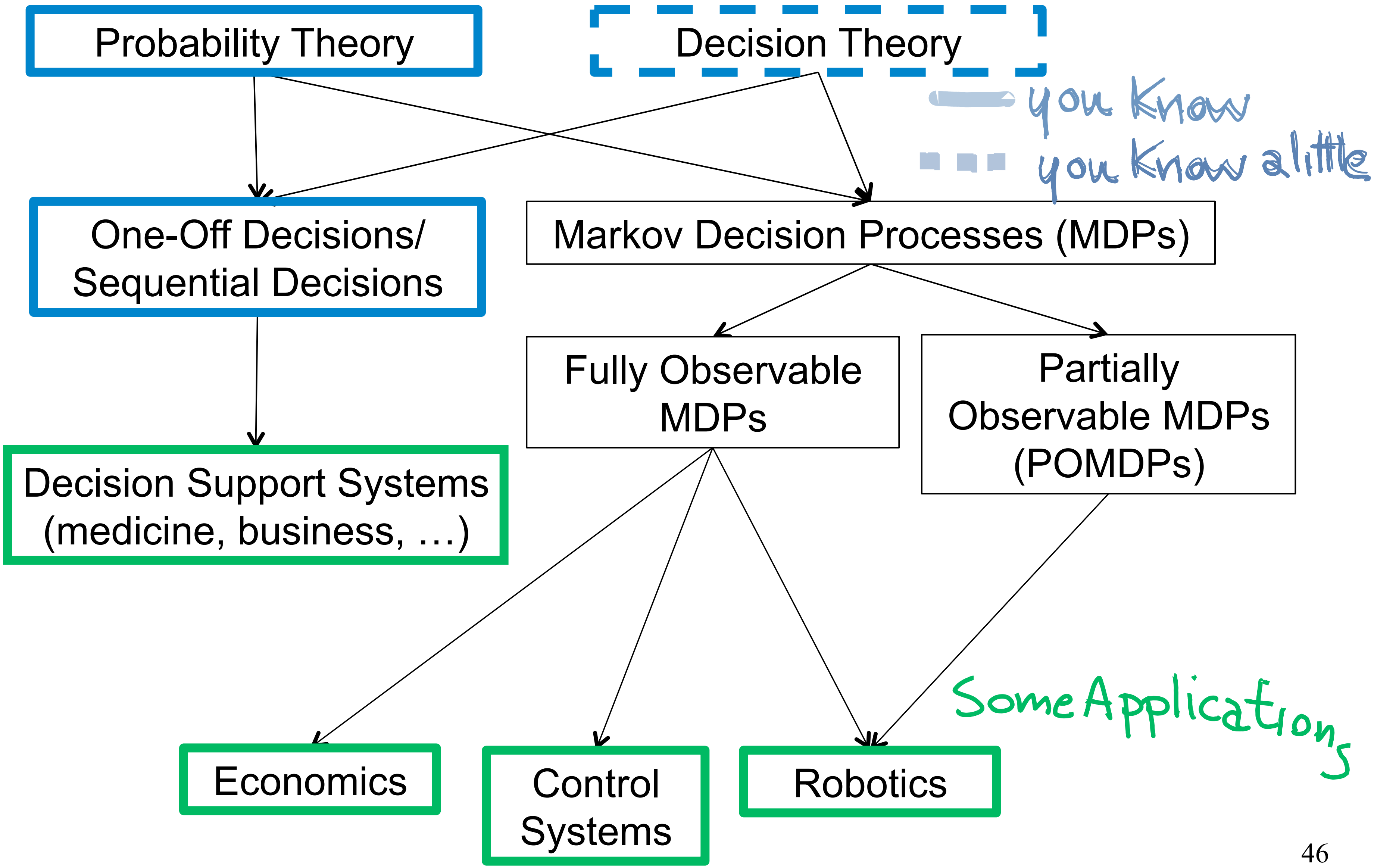
- We saw:
  For *d* decision variables (each with *k* binary parents and *b* possible actions), there are $(b^{2^k})^d$ policies
  - All combinations of (b^{2^k}) decision functions per decision

- Variable elimination saves the final exponent:
  - Dynamic programming: consider each decision functions only once
  - Resulting complexity: $O(d * b^{2^k})$
  - Much faster than enumerating policies (or search in policy space), but still doubly exponential
  - CS422: approximation algorithms for finding optimal policies

Lecture Overview

- Recap: Single-Stage Decision Problems
  - Single-Stage decision networks
  - Variable elimination (VE) for computing the optimal decision
- Sequential Decision Problems
  - General decision networks
  - Policies
- Expected Utility and Optimality of Policies
- Computing the Optimal Policy by Variable Elimination
- Summary & Perspectives

Big Picture: Planning under Uncertainty



Decision Theory: Decision Support Systems

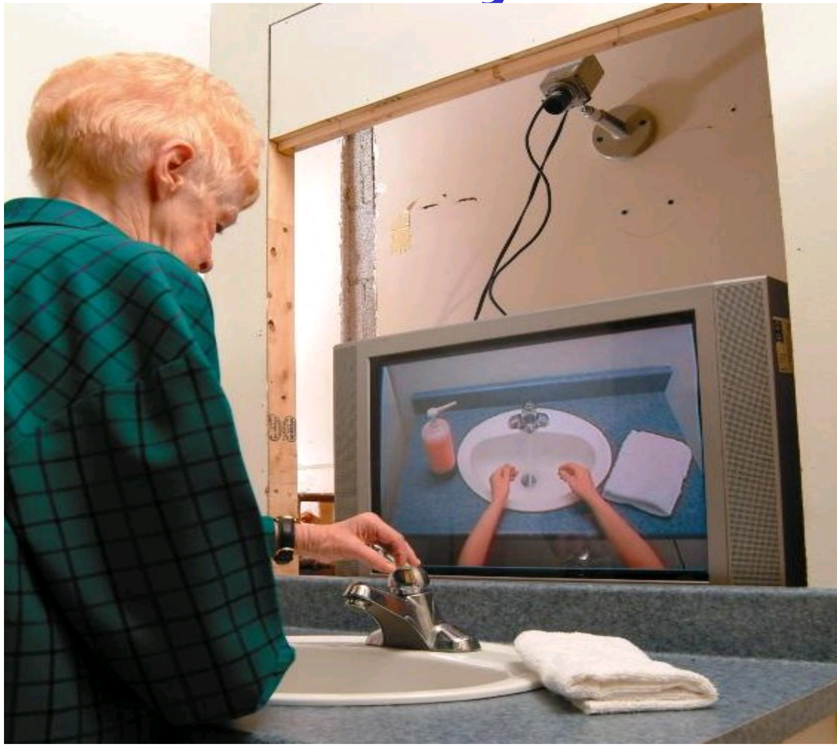
E.g., Computational Sustainability

- 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
- Often constraint optimization problems. E.g.
  - Energy: when are 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 POMDP models of Patient-Caregiver Interactions During Activities of Daily Living

- 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*

Planning Under Uncertainty

Helicopter control: MDP, reinforcement learning

(states: all possible positions, orientations, velocities and angular velocities)

Source: *Andrew Ng*

Planning Under Uncertainty

Autonomous driving: DARPA Urban Challenge – Stanford’s Junior

Source: *Sebastian Thrun*

Learning Goals For Today's Class

- 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