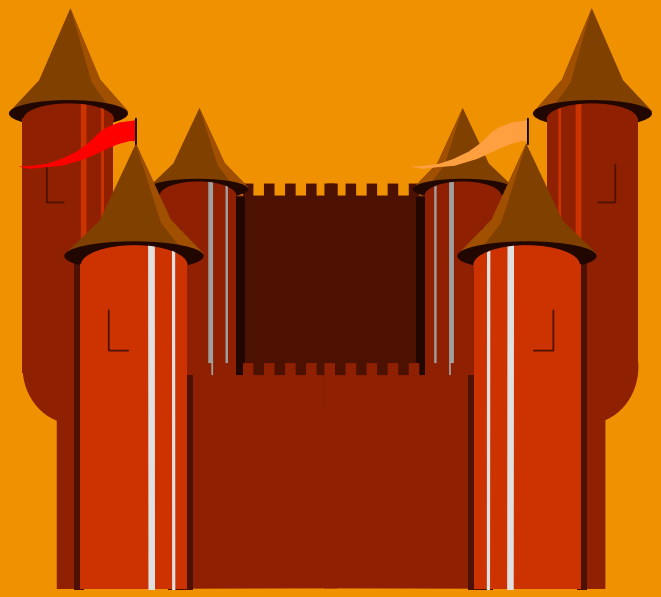
Tutorial: A Unified Modeling and Algorithmic Framework for Optimization under Uncertainty

StochMod 2018
Lancaster, UK

June 13, 2018

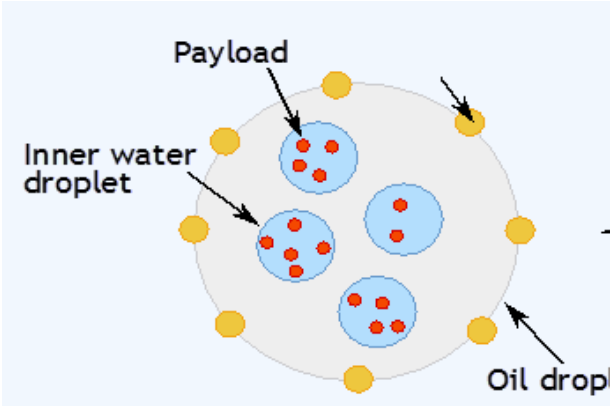


Warren B. Powell

Princeton University
Department of Operations Research and Financial Engineering

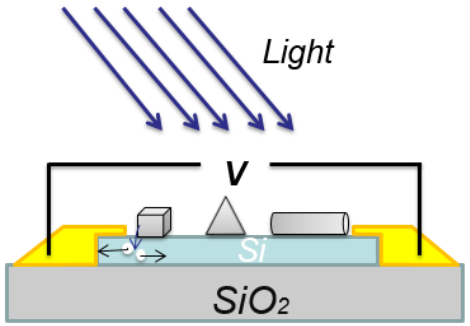
Learning problems

- Materials science
  - » Optimizing payloads: reactive species, biomolecules, fluorescent markers, …



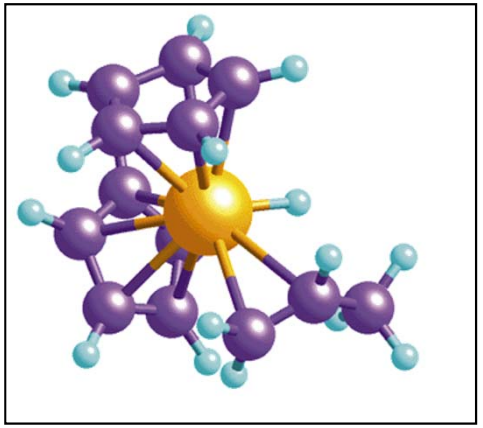
  - » Controllers for robotic scientist for materials science experiments

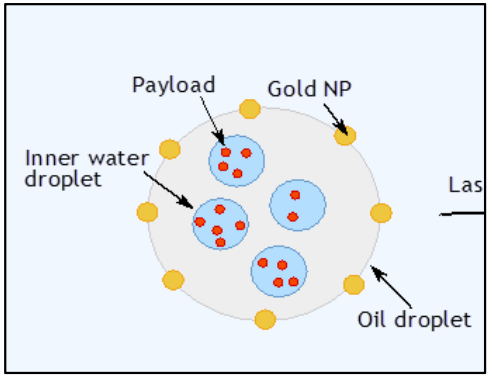
  - » Optimizing nanoparticles to maximize photoconductivity

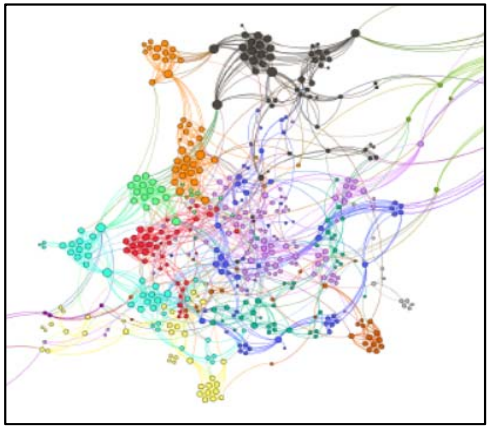


Learning problems

- Health sciences
  - » Sequential design of experiments for drug discovery
  - » Drug delivery – Optimizing the design of protective membranes to control drug release
  - » Medical decision making – Optimal learning for medical treatments.

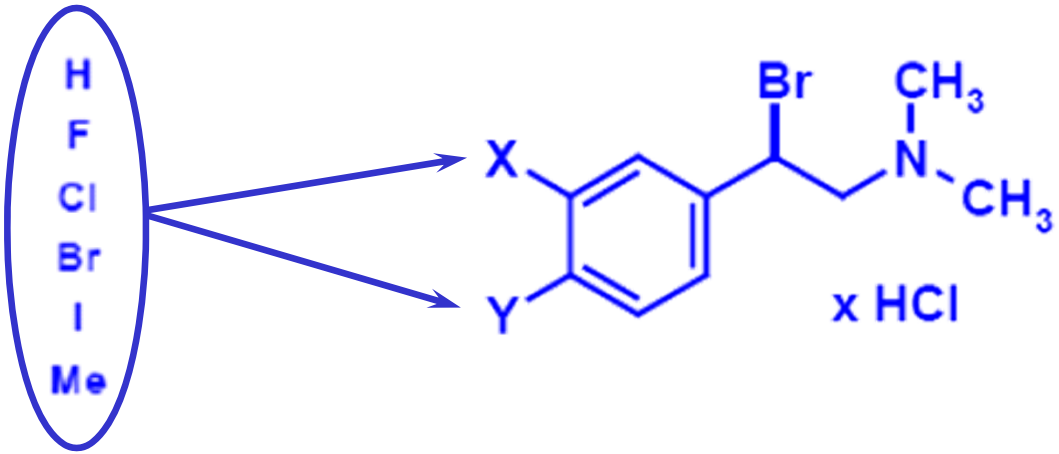


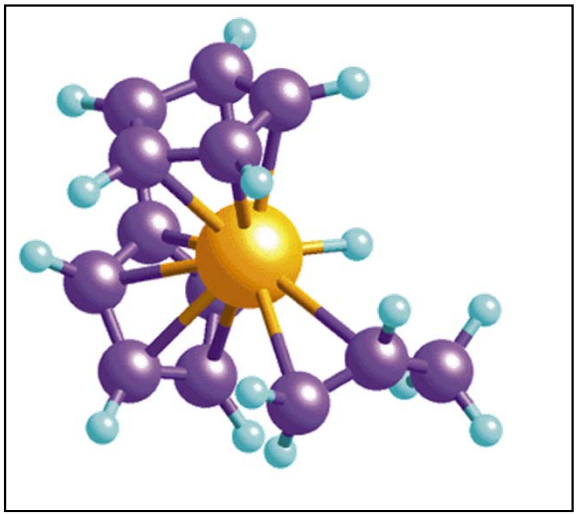




Drug discovery

- Designing molecules

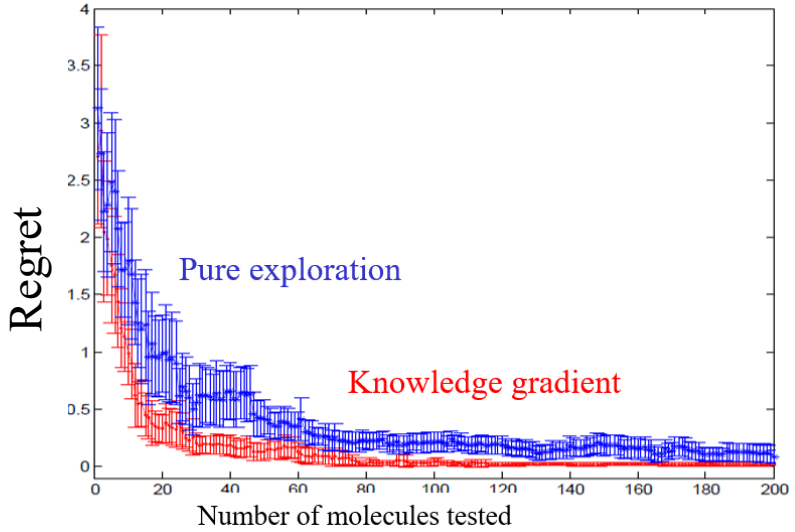




» X and Y are *sites* where we can hang *substituents* to change the behavior of the molecule. We approximate the performance using a linear belief model:

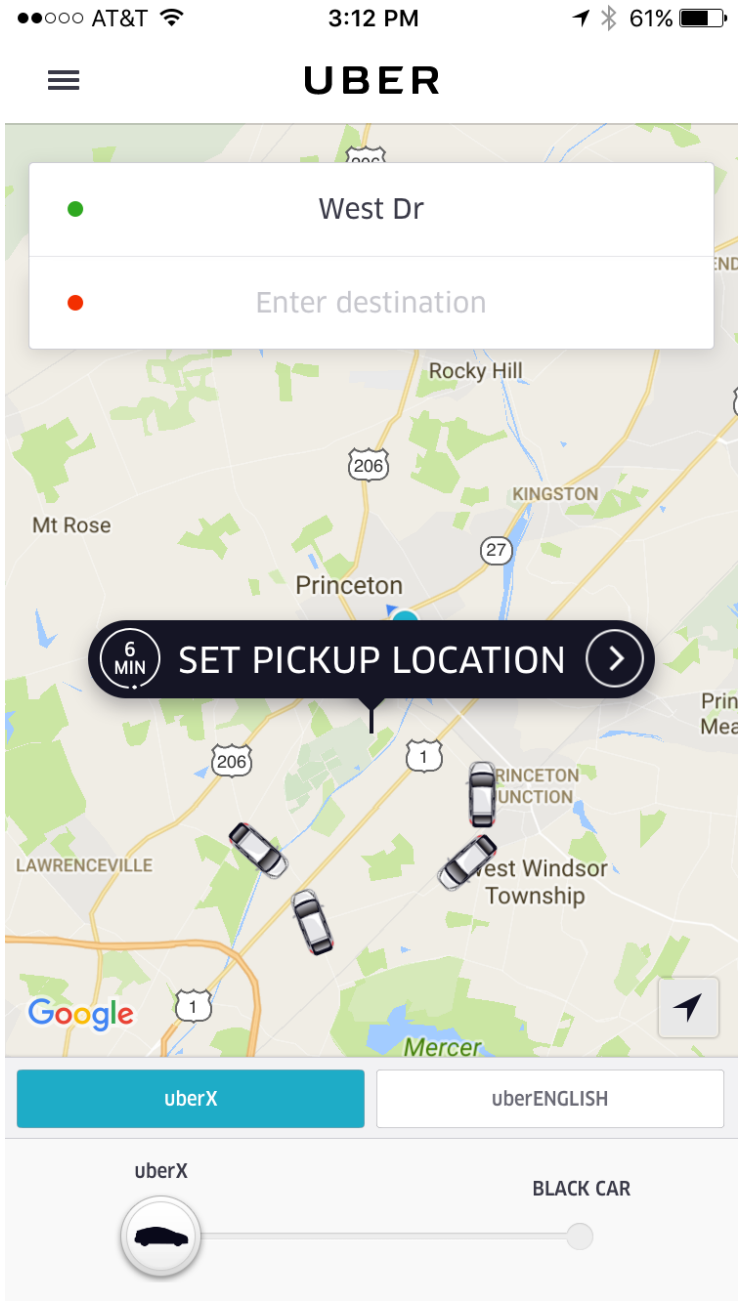
$$Y = \theta_0 + \sum_{\text{sites } i} \sum_{\text{substituents } j} \theta_{ij} X_{ij}$$

» How to sequence experiments to learn the best molecule as quickly as possible?

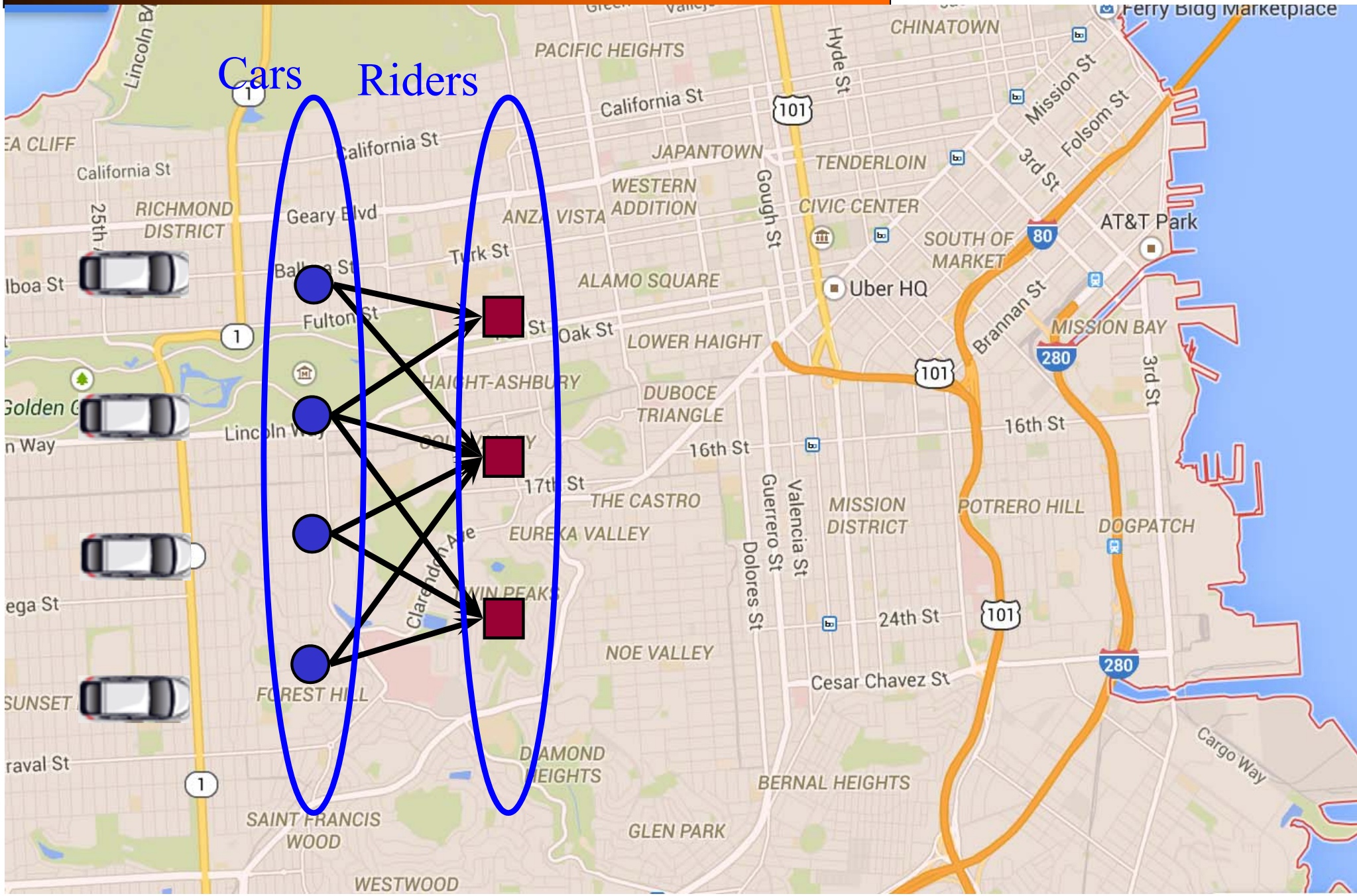


Ride sharing

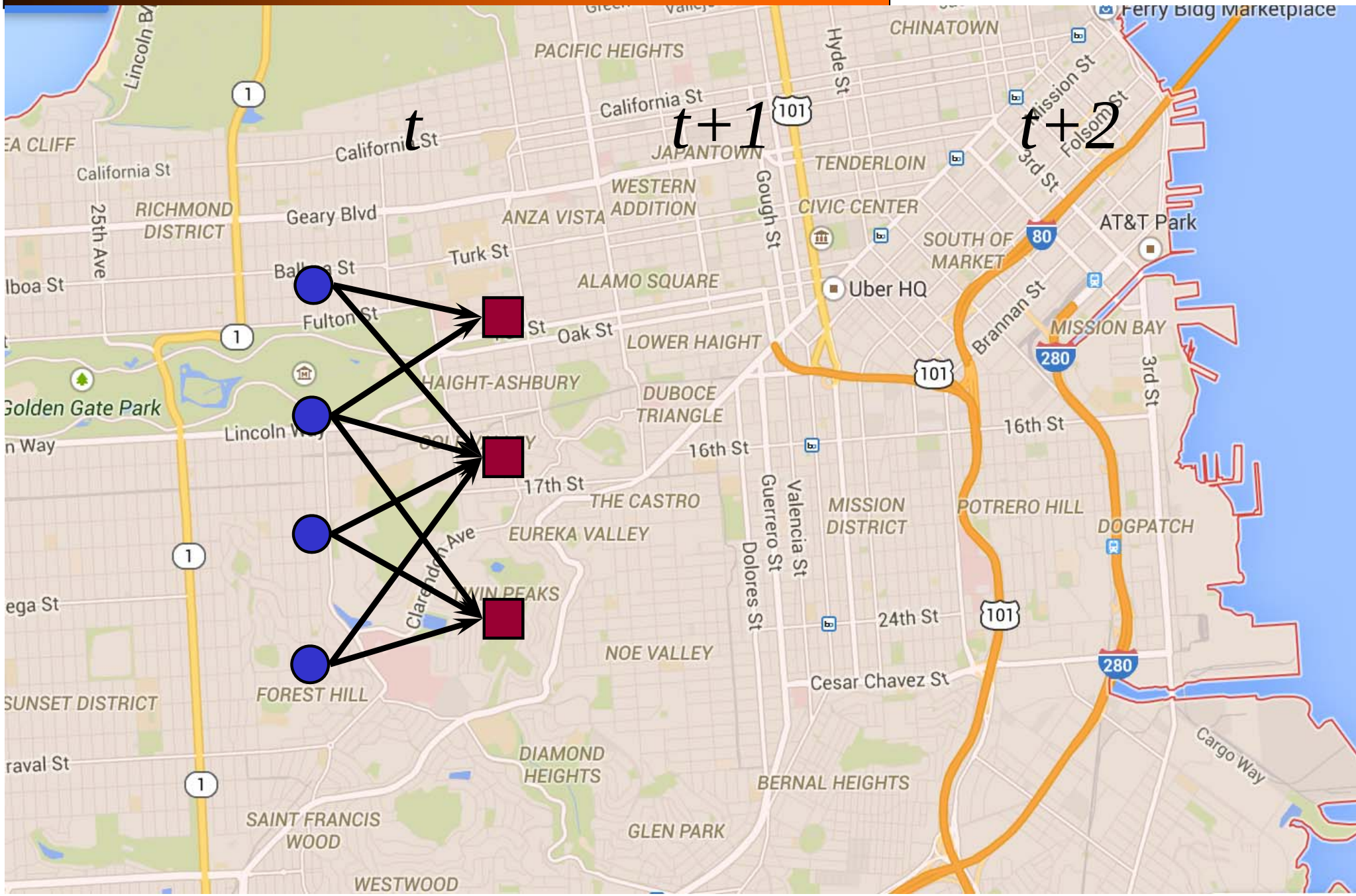
- Uber/Lyft
  - » Provides real-time, on-demand transportation.
  - » Drivers are encouraged to enter or leave the system using pricing signals and informational guidance.
- Decisions:
  - » How to price to get the right balance of drivers relative to customers.
  - » Real-time management of drivers.
  - » Policies (rules for managing drivers, customers, …)



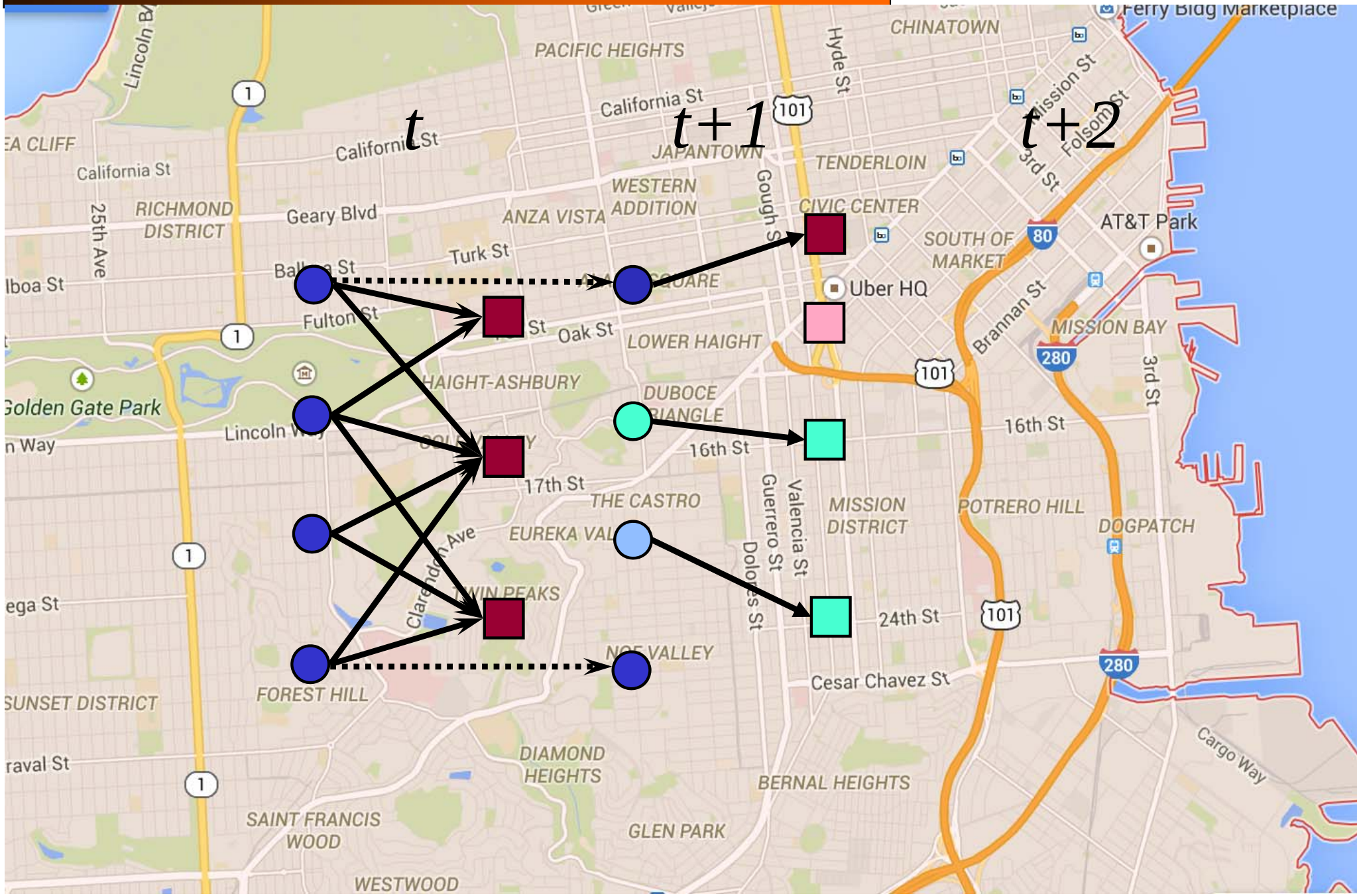
Ride sharing



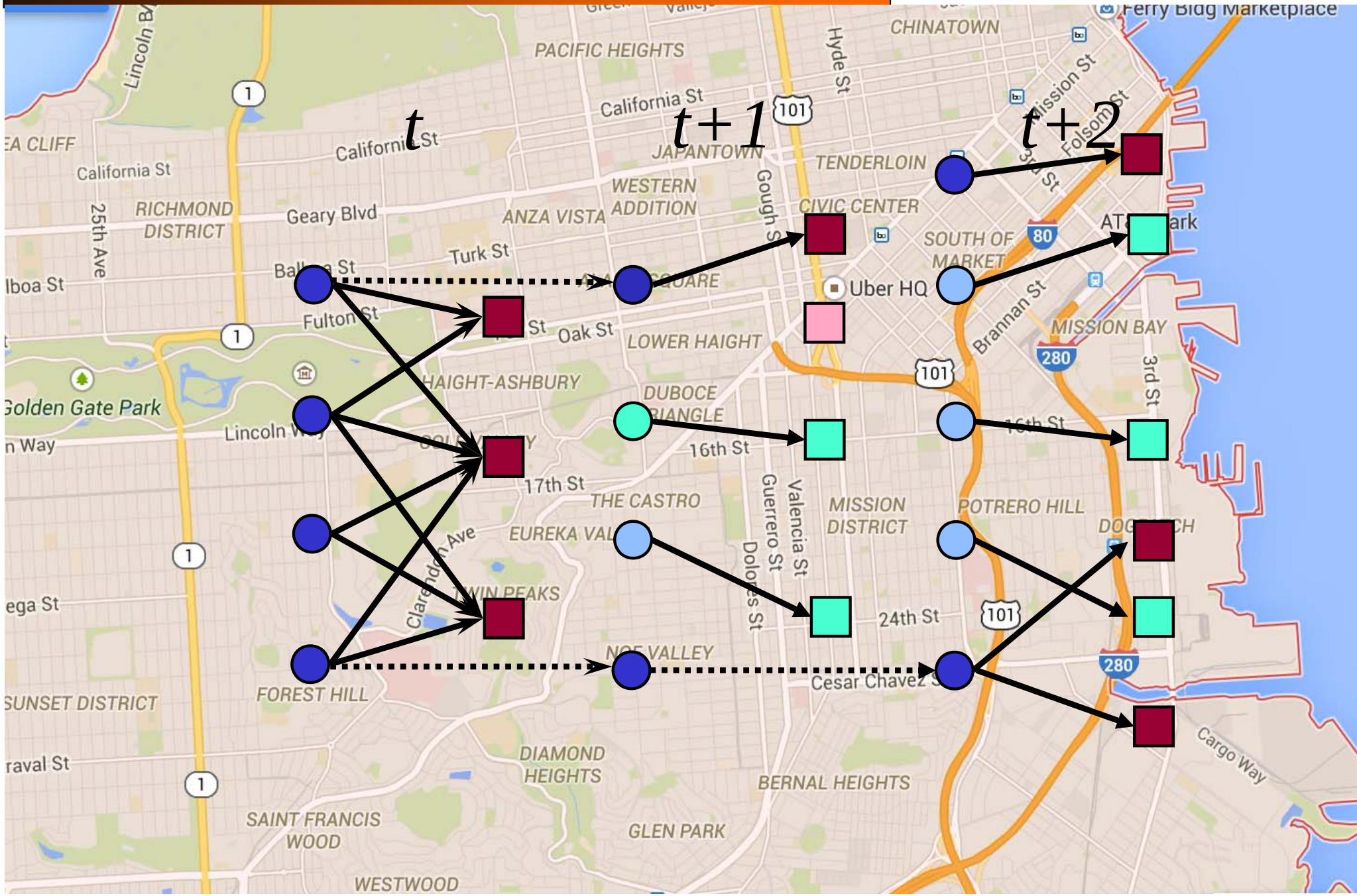
Ride sharing



Ride sharing



Ride sharing



Matching buyers with sellers

- Now we have a logistic curve for each origin-destination pair (i,j)

$$P^{Y}(p,a\,|\,\theta)=\frac{e^{\theta_{ij}^{0}+\theta_{ij}p+\theta_{ij}^{a}a}}{1+e^{\theta_{ij}^{0}+\theta_{ij}p+\theta_{ij}^{a}a}}$$

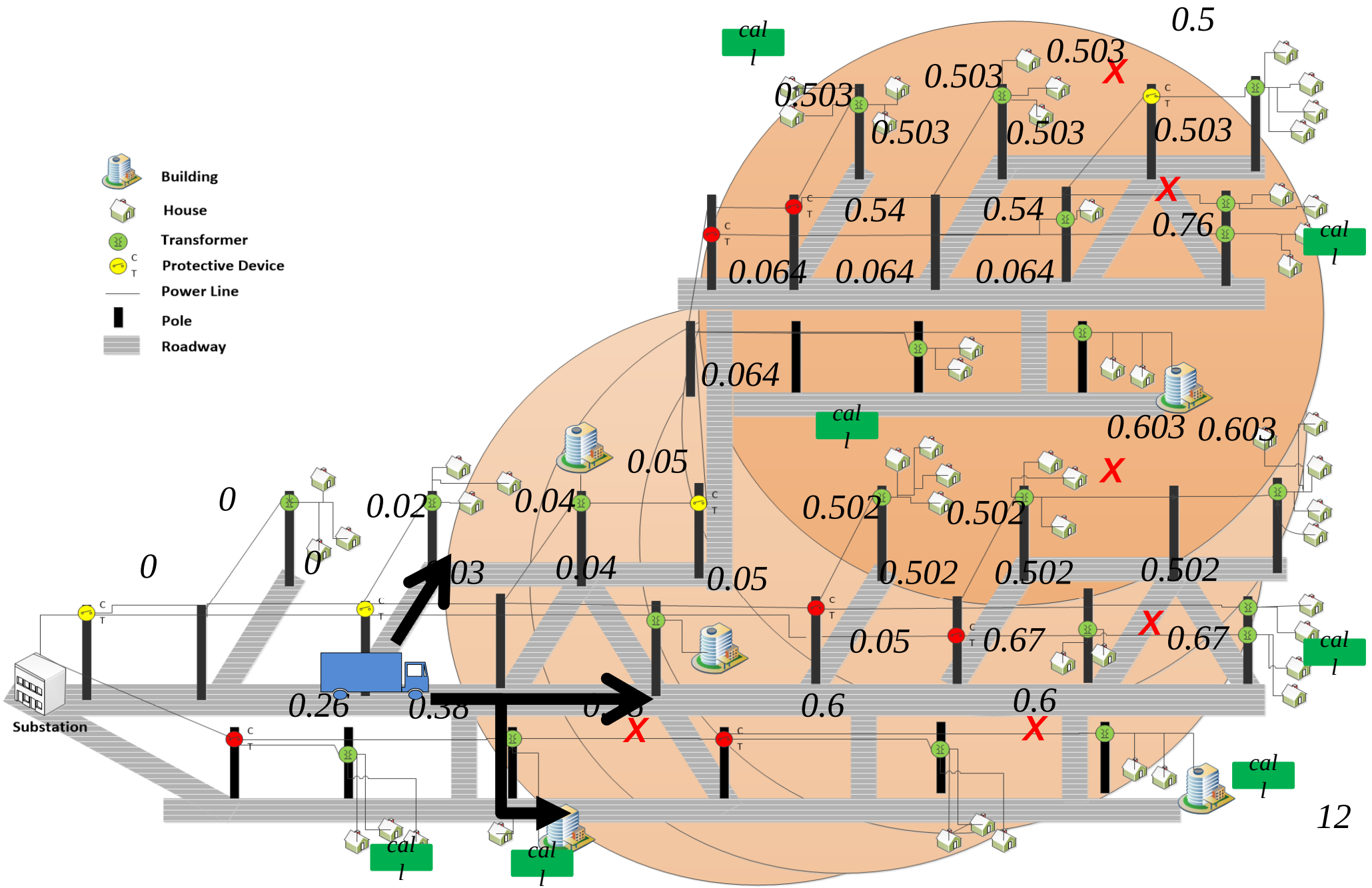
- Number of offers for each (i,j) pair is relatively small.
- Need to generalize the learning across hundreds to thousands of markets.

Emergency storm response

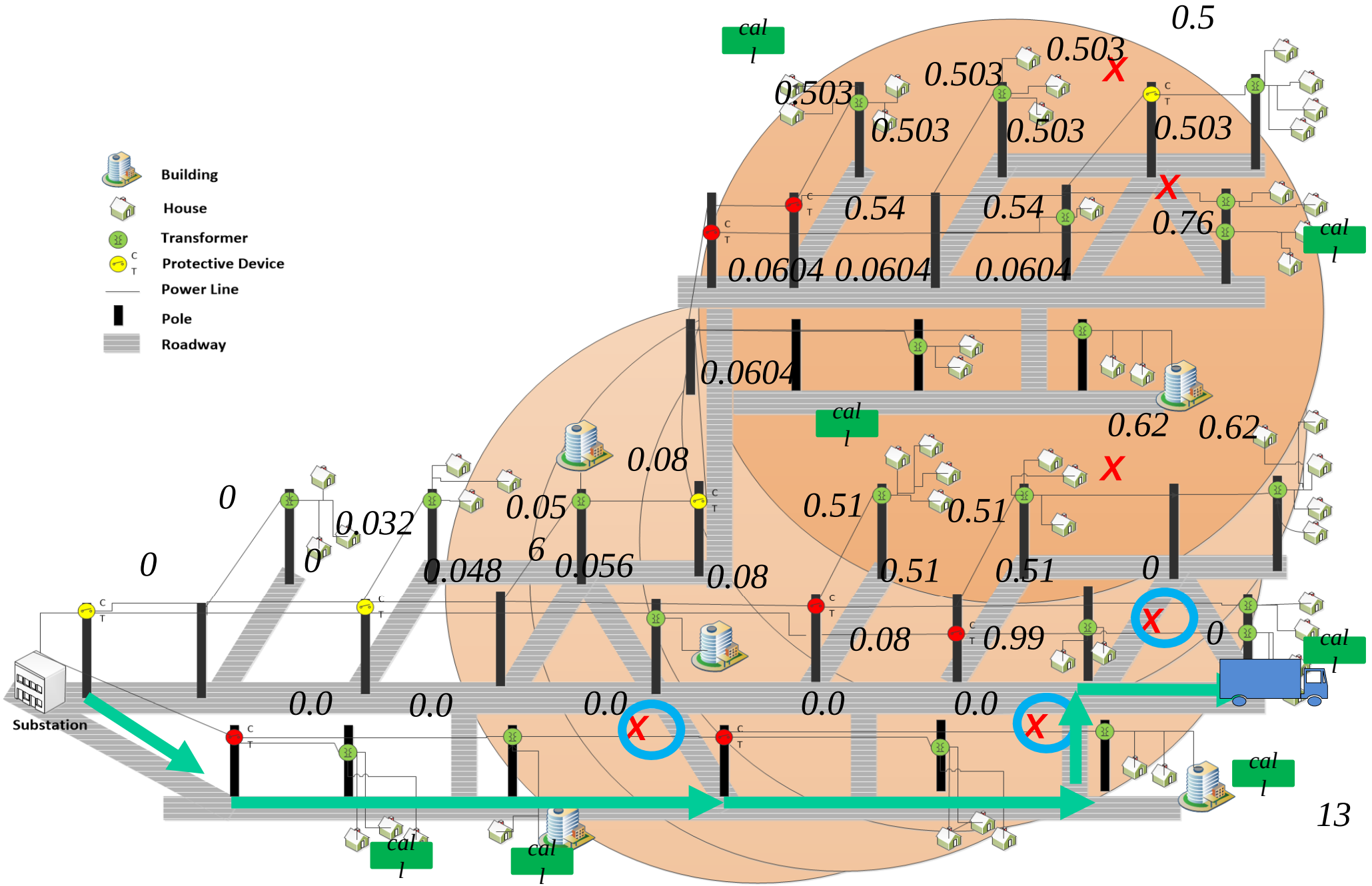
- Hurricane Sandy
  - » Once in 100 years?
  - » Rare convergence of events
  - » But, meteorologists did an amazing job of forecasting the storm.

- The power grid
  - » Loss of power creates cascading failures (lack of fuel, inability to pump water)
  - » How to plan?
  - » How to react?

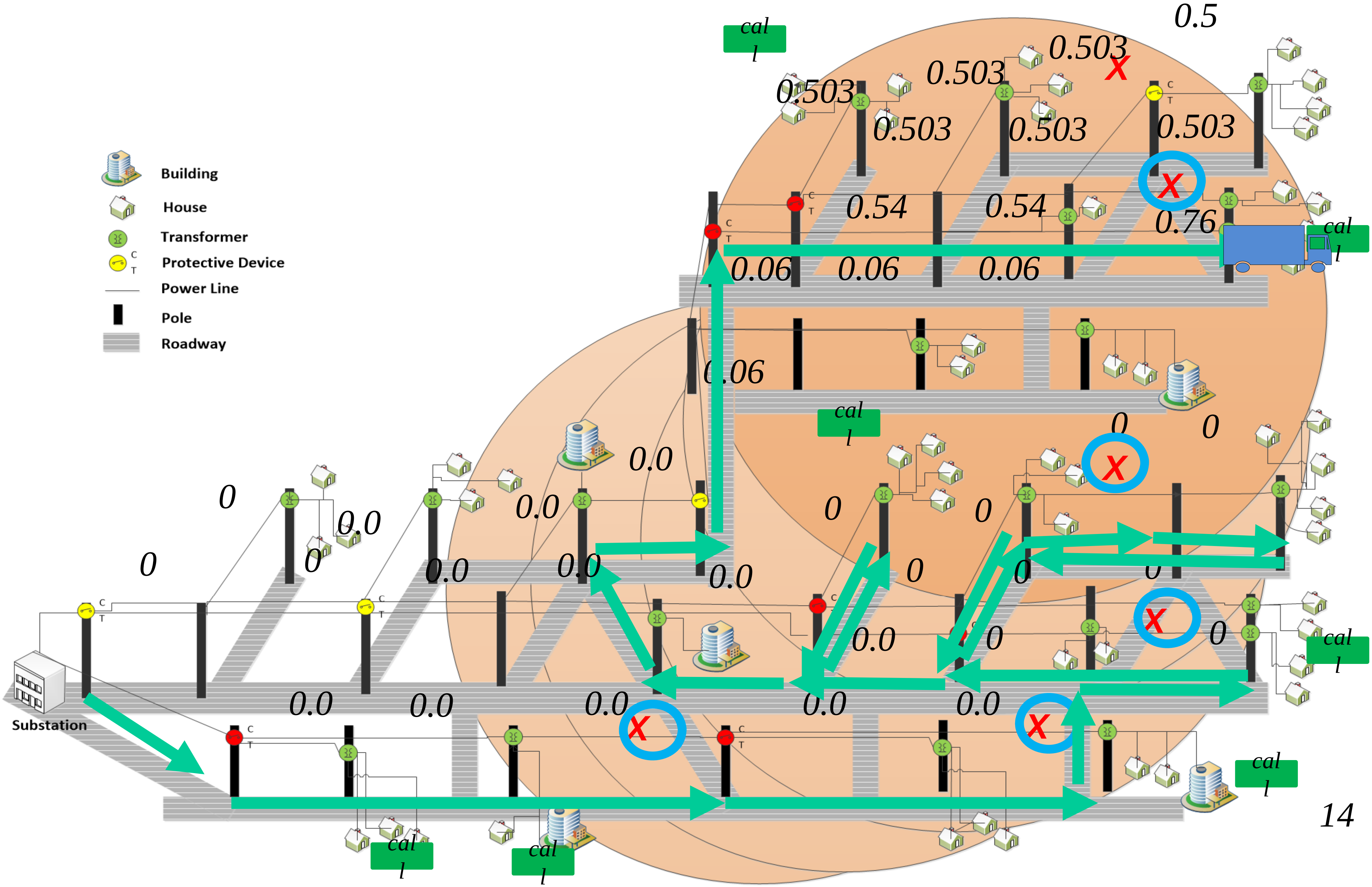
Emergency storm response



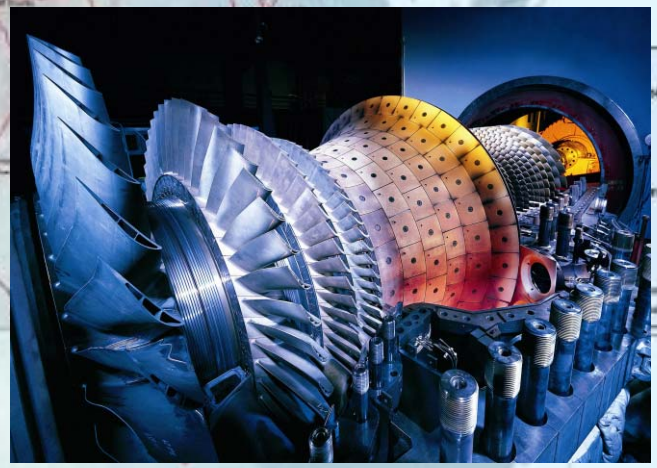
Emergency storm response



Emergency storm response



Meeting variability with *portfolios* of generation with mixtures of *dispatchability*



Storage applications

- How much energy to store in a battery to handle the volatility of wind and spot prices to meet demands?

Outline

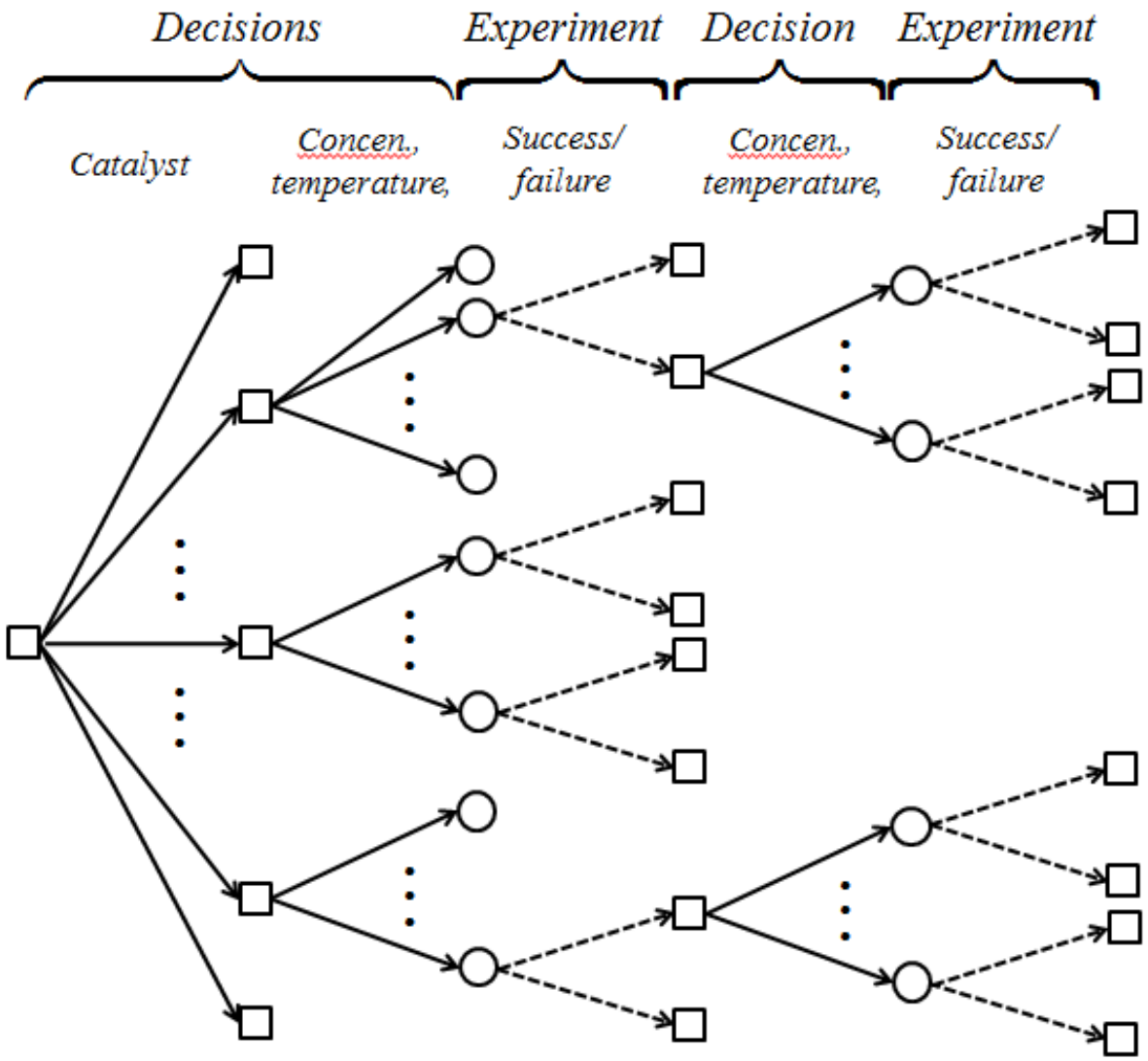
- Canonical problems
- Elements of a dynamic model
- An energy storage illustration
- Solution strategies and problem classes
- Modeling uncertainty
- Designing policies

Outline

- Canonical problems
- Elements of a dynamic model
- An energy storage illustration
- Solution strategies and problem classes
- Modeling uncertainty
- Designing policies

Canonical problems

- Decision trees



Canonical problems

- Stochastic search (derivative based)
  - » Basic problem:

$$\max_x \mathbb{E}F(x,W)$$

Manufacturing network (x=design)
Unit commitment problem (x=day ahead decisions)
Inventory system (x=design, replenishment policy)
Battery system (x=choice of material)
Patient treatment cost (x=drug, treatments)
Trucking company (x=fleet size and mix)

  - » Stochastic gradient

$$x^{n+1} = x^n + \alpha_n \nabla_x F(x^n, W^{n+1})$$

  - » Asymptotic convergence:

$$\lim_{n\to\infty} \mathbb{E}F(x^n,W) \to \mathbb{E}F(x^*,W)$$

  - » Finite time performance

$$\max_\pi \mathbb{E}F(x^{\pi,n},W)$$ where π is an algorithm (or policy)

Canonical problems

- Ranking and selection (derivative free)
  - » Basic problem:

$$\max_{x \in \{x_1, \ldots, x_M\}} \mathbb{E} F(x, W)$$

  - » We need to design a policy $X^{\pi}(S^n)$ that finds a design given by $x^{\pi,N}$

$$\max_{\pi} \mathbb{E} F\left(x^{\pi,N}, W\right)$$

  - » We refer to this objective as maximizing the *final reward.*

Canonical problems

- Multi-armed bandit problems
  - » We learn the reward from playing each "arm"
  - » We need to find a policy $X^{\pi}(S^n)$ for playing machine x that maximizes:

$$\max_{\pi} \mathbb{E} \sum_{n=0}^{N-1} F(X^{\pi}(S^n), W^{n+1})$$

where

W^{n+1} = "winnings" — New information

S^n = State of knowledge — What we know about each slot machine

$x^n = X^{\pi}(S^n)$ — Choose next "arm" to play

We refer to this problem as maximizing *cumulative reward.*

Canonical problems

- (Discrete) Markov decision processes
  - » Bellman's optimality equation

$$V_t(S_t) = \min_{a_t \in \mathcal{A}} \left(C(S_t, a_t) + \gamma \mathbb{E}\left\{ V_{t+1}(S_{t+1}) \mid S_t \right\} \right)$$

$$= \min_{a_t \in \mathcal{A}} \left(C(S_t, a_t) + \gamma \sum_{s'} p(S_{t+1} = s' \mid S_t, a_t) V_{t+1}(S_{t+1}) \right)$$

  - » This is also the same as solving

$$\min_{\pi} E\left\{ \sum_{t=0}^{T} C\left(S_t, X_t^{\pi}(S_t)\right) \mid S_0 \right\}$$

where the optimal policy has the form

$$X^{\pi}(S_t) = \arg\min_{x_t} \left(C(S_t, x_t) + \mathbb{E}\left\{ V_{t+1}(S_{t+1}) \mid S_t, x_t \right\} \right)$$

Canonical problems

- Optimal stopping I
  - » Model:
    - Exogenous process:

$$\omega = (p_1, p_2, ..., p_T) = \text{Sequence of stock prices}$$

    - Decision:

$$X_t(\omega) = \begin{cases} 1 & \text{If we stop and sell at time } t \\ 0 & \text{Otherwise} \end{cases}$$

    - Reward:

$$p_t = \text{Price received if we stop at time } t$$

  - » Optimization problem:

$$\max_\tau \mathbb{E} p_\tau X_\tau$$

where τ is a "stopping time" (or "F_t −measurable function")

Canonical problems

- Optimal stopping II
  - » Model:
    - Exogenous process:

$$\omega = \left(p_1, p_2, ..., p_T \right) = \text{Sequence of stock prices}$$

$$\bar{p}_t = (1-\alpha)\bar{p}_{t-1} + \alpha p_t$$

    - State:

$R_t = 1$ if we are holding asset, 0 otherwise.

$$S_t = (R_t, p_t, \bar{p}_t)$$

    - Policy:

$$X_t(S_t \mid \theta) = \begin{cases} 1 & p_t \geq \bar{p}_t + \theta \\ 0 & \text{Otherwise} \end{cases}$$

  - » Optimization problem:

$$\max_\pi \mathbb{E} \sum_{t=0}^{T} p_t X^\pi(S_t \mid \theta) = \max_\theta \mathbb{E} \sum_{t=0}^{T} p_t X^\pi(S_t \mid \theta)$$

Canonical problems

- Linear quadratic regulation (LQR)
  - » A popular optimal control problem in engineering involves solving:

$$\min_{u_0,\ldots,u_T} \mathbb{E}\sum_{t=0}^{T}\left((x_t)^T Q x_t + (u_t)^T R u_t\right)$$

  - » where:

$$x_t = \text{State at time } t$$

$$u_t = \text{Control at time } t \text{ (must be } F_t - \text{measurable)}$$

$$x_{t+1} = f(x_t, u_t) + w_t \quad (w_t \text{ is random at time } t)$$

  - » Possible to show that the optimal policy looks like:

$$U_t^*(x_t) = K_t x_t$$

where K_t is a complicated function of Q and R.

Canonical problems

- Stochastic programming
  - » A (two-stage) stochastic programming *problem*

$$\min_{x_0 \in X_0} c_0 x_0 + \mathbb{E} Q(x_0, \xi_1)$$

where

$$Q(x_0, \xi_1(\omega)) = \min_{x_1(\omega) \in X_1(\omega)} c_1(\omega) x_1(\omega)$$

  - » This is the canonical form of stochastic programming, which might also be written over multiple periods:

$$\min c_0 x_0 + \sum_{\omega \in \Omega} p(\omega) \sum_{t=1}^{T} c_t(\omega) x_t(\omega)$$

Canonical problems

- Stochastic programming
  - » A (two-stage) stochastic programming *policy*

$$X_t^{\pi}(S_t) = \arg\min_{x_t \in X_t} c_t x_t + \mathbb{E}Q(x_t, \xi_{t+1})$$

where

$$Q(x_t, \xi_{t+1}(\omega)) = \min_{x_{t+1}(\omega) \in X_{t+1}(\omega)} c_{t+1}(\omega) x_{t+1}(\omega)$$

  - » This is the canonical form of stochastic programming, which might also be written over multiple periods:

$$\min c_t x_t + \sum_{\omega_t \in \Omega_t} p(\omega_t) \sum_{t'=t+1}^{t+H} c_{tt'}(\omega) x_{tt'}(\omega)$$

Canonical problems

- A robust optimization *problem* would be written

$$\min_{x \in X} \max_{w \in \mathcal{W}(\theta)} F(x, w)$$

  - » This means finding the best design x for the worst outcome w in an "uncertainty set" $\mathcal{W}(\theta)$

  - » This has been adapted to multiperiod problems

$$X_t^{\pi}(S_t) = \arg\min_{x_t, \ldots, x_{t+H}} \max_{(w_t, \ldots, w_{t+H}) \in \mathcal{W}(\theta)} \sum_{t'=t}^{t+H} c_{t'}(w_{t'}) x_{t'}$$

Canonical problems

- Why do we need a unified framework?
  - » The classical frameworks and algorithms are *fragile*.
  - » Small changes to problems invalidate optimality conditions, or make algorithmic approaches intractable.
  - » Practitioners need robust approaches that will provide high quality solutions for all problems.

Outline

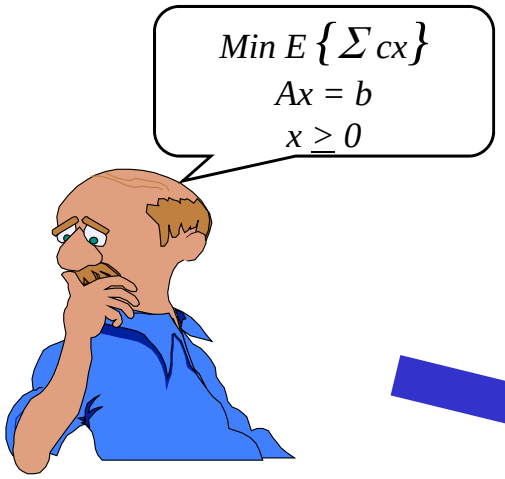
- Canonical problems
- Elements of a dynamic model
- An energy storage illustration
- Solution strategies problem classes
- Modeling uncertainty
- Designing policies

Modeling

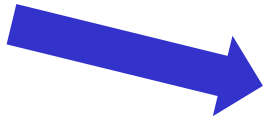
- How much energy to store in a battery to handle the volatility of wind and spot prices to meet demands?

Modeling

- Before we can *solve* complex problems, we have to know how to *think* about them.



Mathematician



Software

- The biggest challenge when making decisions under uncertainty is *modeling*.

Modeling

- For deterministic problems, we speak the language of mathematical programming
  - » Linear programming:

$$\min_x cx$$
$$Ax = b$$
$$x \geq 0$$

  - » For time-staged problems

$$\min_{x_0,\ldots,x_T} \sum_{t=0}^{T} c_t x_t$$
$$A_t x_t - B_{t-1} x_{t-1} = b_t$$
$$D_t x_t \leq u_t$$
$$x_t \geq 0$$

Arguably Dantzig's biggest contribution, more so than the simplex algorithm, was his articulation of optimization problems in a standard format, which has given algorithmic researchers a common language.


Stochastic programming
Robust optimization
Decision
Approximate dynamic programming
Simulation optimization
analysis
Optimal learning
Model predictive control
Dynamic Programming and control
Stochastic search
Bandit problems
Optimal control
Online computation
Reinforcement learning
Markov decision processes
Stochastic control
Simulation optimization


John R. Birge
François Louveaux
Introduction to Stochastic Programming
Second Edition
Michael C. Fu Editor
Handbook of Simulation Optimization
Princeton Series in Applied Mathematics
Robust Optimization
Introduction to Decision Analysis
A Practitioner's Guide to Improving Decision Quality
SECOND EDITION
Approximate Dynamic Programming
Solving the Curses of Dimensionality
Warren B. Powell
VOLUME 2 • 4th EDITION
Dynamic Programming and Optimal Control
APPROXIMATE DYNAMIC PROGRAMMING
Optimal Learning
SECOND EDITION
Model Predictive Control
OPTIMAL CONTROL
Dimitri P. Bertsekas
Athena Scientific
INTRODUCTION TO STOCHASTIC SEARCH AND OPTIMIZATION
Estimation, Simulation, and Control
JAMES C. SPALL
WILEY
MULTI-ARMED BANDIT ALLOCATION INDICES
SECOND EDITION
John Gittins, Kevin Glazebrook and Richard Weber
Reinforcement Learning
Introduction
Richard S. Sutton and Andrew G. Barto
Markov Decision Processes
Discrete Stochastic Dynamic Programming
MARTIN L. PUTERMAN
Online Computation and Competitive Analysis
Allan Borodin
Ran El-Yaniv
STOCHASTIC SIMULATION OPTIMIZATION
An Optimal Computing Budget Allocation
Chun-Hung Chen • Loo Hay Lee
Vol. 1
43
Jiongmin Yong
Xun Yu Zhou
Stochastic Controls
Hamiltonian Systems and HJB Equations

Modeling

- We lack a standard language for modeling sequential, stochastic decision problems.
  - » In the slides that follow, we propose to model problems along five fundamental dimensions:
    - State variables
    - Decision variables
    - Exogenous information
    - Transition function
    - Objective function
  - » This framework draws heavily from Markov decision processes and the control theory communities, but it is not the standard form used anywhere.

Modeling dynamic problems

- The state variable:

Controls community

x_t = "Information state"

Operations research/MDP/Computer science

$S_t = (R_t, I_t, B_t)$ = System state, where:

R_t = Resource state (physical state)

Location/status of truck/train/plane

Energy in storage

I_t = Information state

Prices

Weather

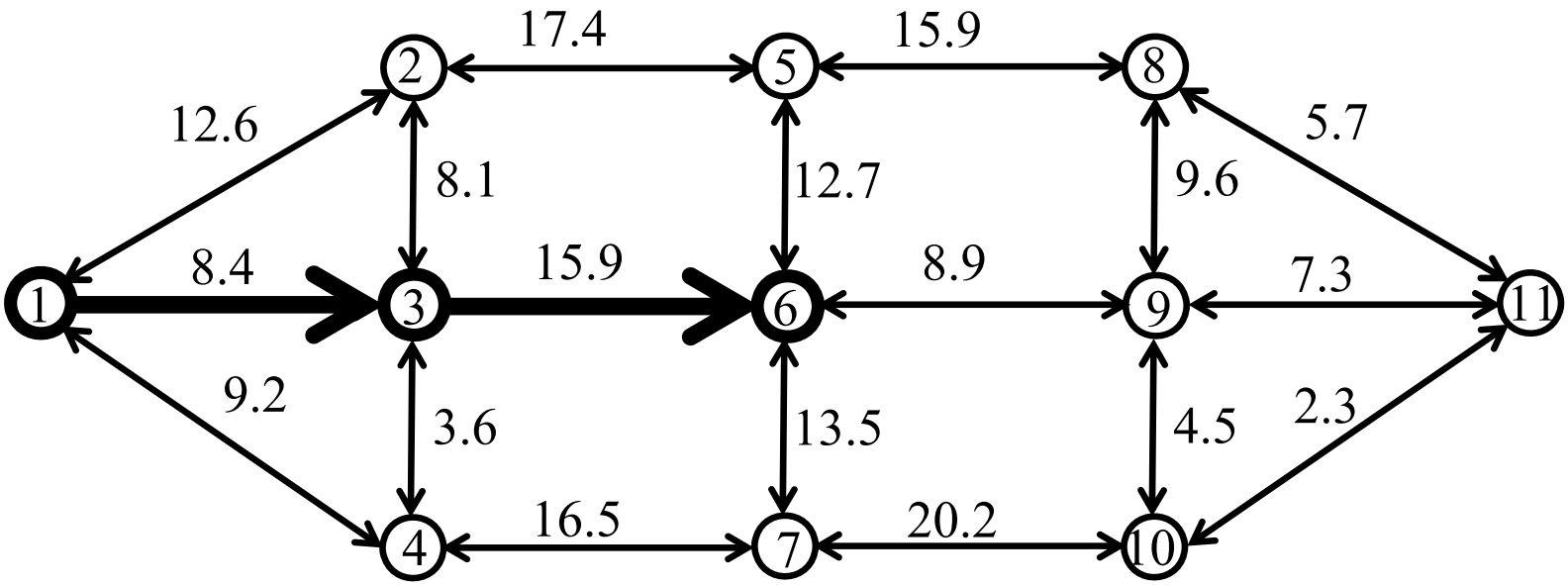
B_t = Belief state ("state of knowledge")

Belief about traffic delays

Belief about the status of equipment

The state variable

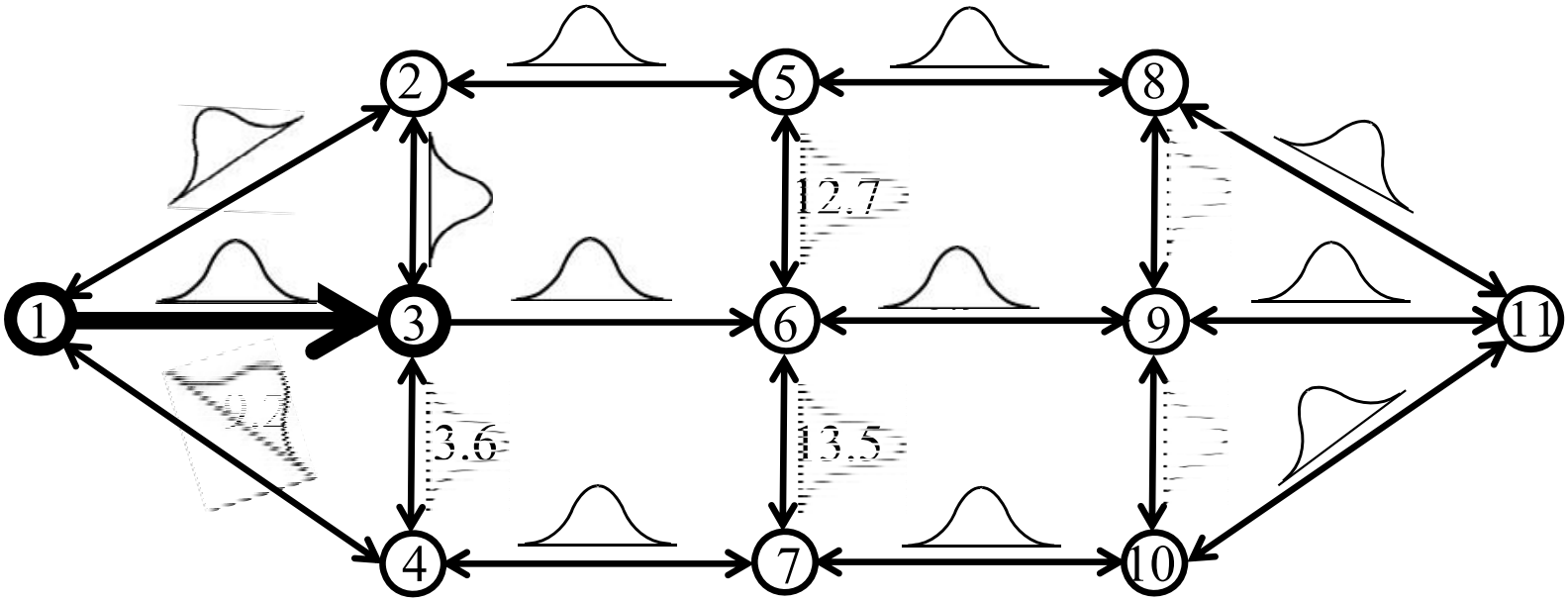
- Illustrating state variables
  - » A deterministic graph



$$S_t = (N_t) = 6$$

The state variable

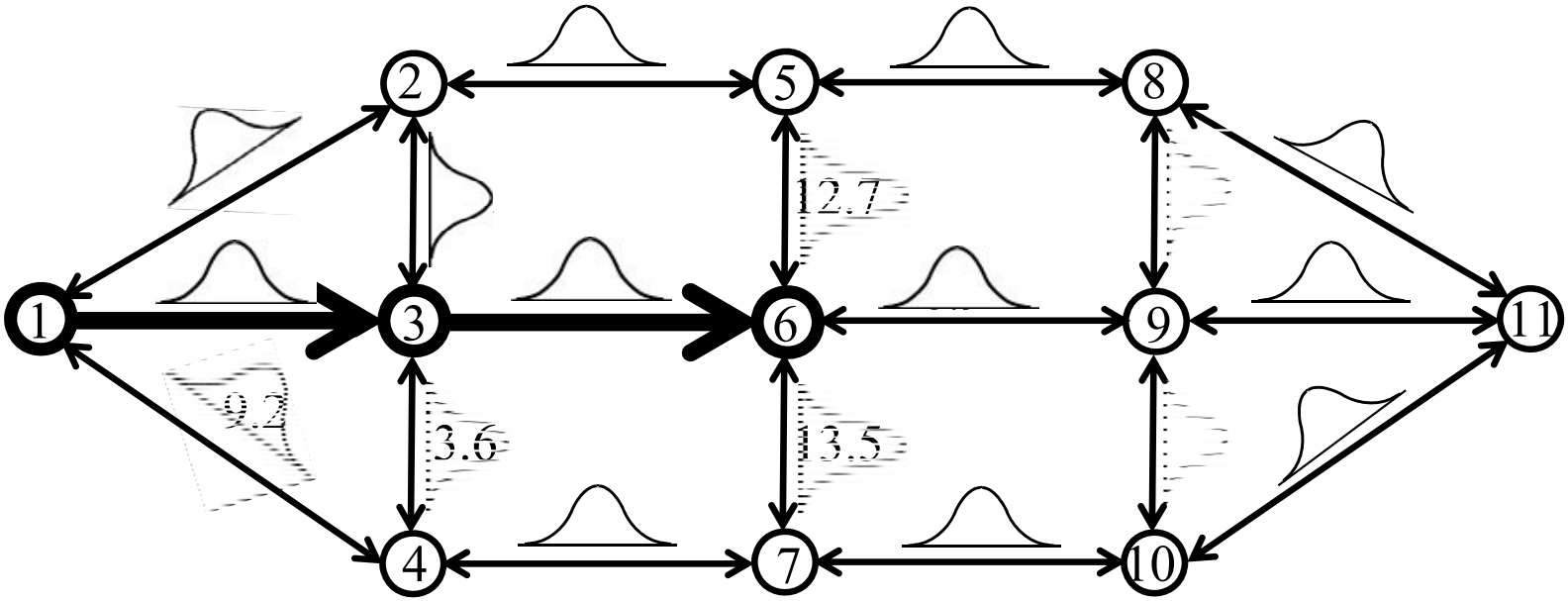
- Illustrating state variables
  - » A stochastic graph



$S_t = ?$

The state variable

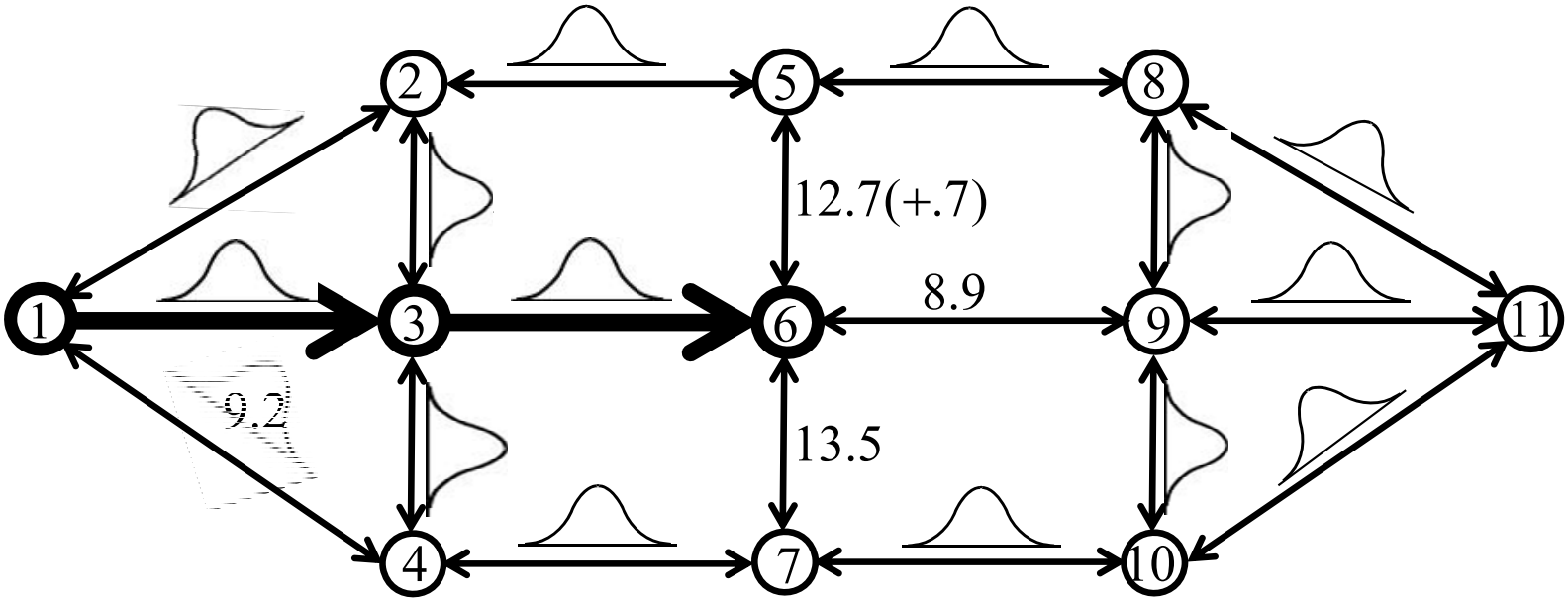
- Illustrating state variables
  - » A stochastic graph



$$S_t = \left(\underbrace{N_t}_{R_t}, \underbrace{\left(c_{t,N_t,j} \right)_j}_{I_t} \right) = \left(6, \left(12.7, 8.9, 13.5 \right) \right)$$

The state variable

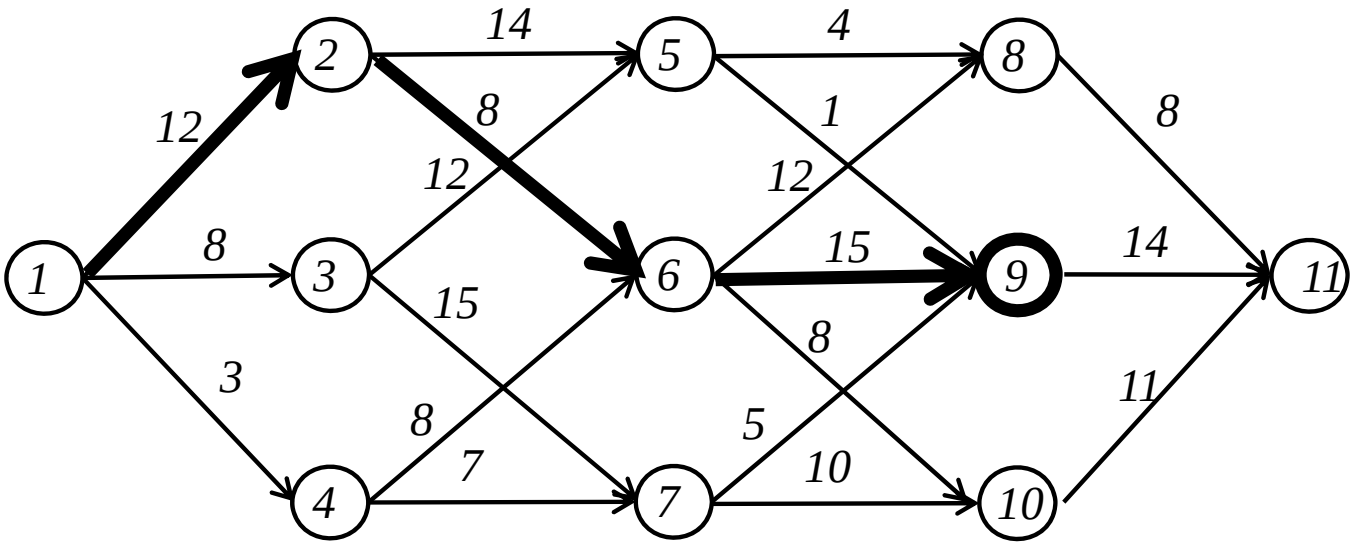
- Illustrating state variables
  - » A stochastic graph with left turn penalties



$$S_t = \left(\underbrace{N_t}_{R_t}, \underbrace{\left(c_{t,N_t,j} \right)_j, N_{t-1}}_{I_t} \right) = \left(6, \left(12, 7, 8.9, 13.5 \right), 3 \right)$$

The state variable

- Variant of problem in Puterman (2005):
  - » Find best path from 1 to 11 that minimizes the *second highest arc cost* along the path:



  - » If the traveler is at node 9, what is her state?

$$S_t = (N_t, \text{highest}, \text{second highest}) = (9,15,12)$$

The state variables

- What is a state variable?
  - » Bellman's classic text on dynamic programming (1957) describes the state variable with:
    - "… we have a physical system characterized at any stage by a small set of parameters, the *state variables*."
  - » The most popular book on dynamic programming (Puterman, 2005, p.18) "defines" a state variable with the following sentence:
    - "At each decision epoch, the system occupies a *state*."
  - » Wikipedia:
    - A state variable is one of the set of variables that are used to describe the mathematical 'state' of a dynamical system

The state variable

- My definition of a state variable:

Definition 9.3.1 *A* **state variable** *is:*

a) Policy-dependent version *A function of history that, combined with the exogenous information (and a policy), is necessary and sufficient to compute the cost/contribution function, the decision function (the policy), and any information required to model the evolution of information needed in the cost/contribution and decision functions.*

b) Optimization version *A function of history that is necessary and sufficient to compute the cost/contribution function, the constraints, and any information required to model the evolution of information needed in the cost/contribution function and the constraints.*

  - » The first depends on a policy. The second depends only on the problem (and includes the constraints).
  - » Using either definition, ***all properly modeled problems are Markovian!***

Modeling dynamic problems

- Decisions:

Markov decision processes/Computer science

a_t = Discrete action

Control theory

u_t = Low-dimensional continuous vector

Operations research

x_t = Usually a discrete or continuous but high-dimensional vector of decisions.

At this point, we do not specify *how* to make a decision. Instead, we define the function $X^\pi(s)$ (or $A^\pi(s)$ or $U^\pi(s)$), where π specifies the type of policy. "π" carries information about the type of function f, and any tunable parameters $\theta \in \Theta^f$.

The decision variables

- Styles of decisions
  - » Binary

    $x \in X = \{0,1\}$
  - » Finite

    $x \in X = \{1,2,...,M\}$
  - » Continuous scalar

    $x \in X = [a,b]$
  - » Continuous vector

    $x = (x_1,...,x_K), \quad x_k \in \mathbb{R}$
  - » Discrete vector

    $x = (x_1,...,x_K), \quad x_k \in \mathbb{Z}$
  - » Categorical

    $x = (a_1,...,a_I), \quad a_i$ is a category (e.g. red/green/blue)

Modeling dynamic problems

- Exogenous information:

W_t = New information that first became known at time t

$= \left(\hat{R}_t, \hat{D}_t, \hat{p}_t, \hat{E}_t\right)$

$\hat{R}_t$ = Equipment failures, delays, new arrivals
New drivers being hired to the network

$\hat{D}_t$ = New customer demands

$\hat{p}_t$ = Changes in prices

$\hat{E}_t$ = Information about the environment (temperature, ...)

Note: Any variable indexed by t is known at time t. This convention, which is not standard in control theory, dramatically simplifies the modeling of information.

Below, we let ω represent a sequence of actual observations $W_1, W_2, \ldots$.

$W_t(\omega)$ refers to a sample realization of the random variable W_t.

Modeling dynamic problems

- The transition function

$$S_{t+1} = S^M(S_t,\ x_t,\ W_{t+1})$$

$$R_{t+1} = R_t + x_t + \hat{R}_{t+1} \qquad \text{Inventories}$$

$$p_{t+1} = p_t + \hat{p}_{t+1} \qquad \text{Spot prices}$$

$$D_{t+1} = D_t + \hat{D}_{t+1} \qquad \text{Market demands}$$

Also known as the:

"System model"	"Transfer function"
"State transition model"	"Transformation function"
"Plant model"	"Law of motion"
"Plant equation"	"Model"
"State equation"	"transition function"

For many applications, these equations are unknown. This is known as "model-free" dynamic programming.

Modeling stochastic, dynamic problems

- Objective functions

  » Cumulative reward ("online learning")

$$\max_{\pi} \mathbb{E}\left\{\sum_{t=0}^{T} C_t\left(S_t, X_t^{\pi}(S_t), W_{t+1}\right) \mid S_0\right\}$$

  - Policies have to work well *over time*, which means fast convergence, and possibly fast learning (if there is a belief state).

  » Final reward ("offline learning")

$$\max_{\pi} \mathbb{E}\left\{F(x^{\pi,N}, \hat{W}) \mid S_0\right\}$$

  - We only care about how well the final decision $x^{\pi,N}$ works, not how well we do while finding it.

Modeling stochastic, dynamic problems

- The complete model:
  - » Objective function
    - Cumulative reward ("online learning")

$$\max_{\pi} \mathbb{E}\left\{\sum_{t=0}^{T} C_t\left(S_t, X_t^{\pi}(S_t), W_{t+1}\right) \mid S_0\right\}$$

    - Final reward ("offline learning")

$$\max_{\pi} \mathbb{E}\left\{F(x^{\pi,N}, \hat{W}) \mid S_0\right\}$$

  - » Transition function:

$$S_{t+1} = S^M\left(S_t, x_t, W_{t+1}(\omega)\right)$$

  - » Exogenous information:

$$\left(S_0, W_1, W_2, ..., W_T\right)$$

Modeling

- Deterministic
  - » Objective function

$$\min_{x_0,\ldots,x_T} \sum_{t=0}^{T} c_t x_t$$

  - » Decision variables:

$$(x_0,\ldots,x_T)$$

  - » Constraints:
    - at time t

$$\left.\begin{array}{l} A_t x_t = R_t \\ x_t \geq 0 \end{array}\right\} \mathcal{X}_t$$

    - Transition function

$$R_{t+1} = b_{t+1} + B_t x_t$$

- Stochastic
  - » Objective function

$$\max_{\pi} E^{\pi} \left\{ \sum_{t=0}^{T} C_t\left(S_t, X_t^{\pi}(S_t), W_{t+1}\right) \mid S_0 \right\}$$

  - » Policy

$$X^{\pi} : S \mapsto \mathcal{X}$$

  - » Constraints at time t

$$x_t = X_t^{\pi}(S_t) \in \mathcal{X}_t$$

  - » Transition function

$$S_{t+1} = S^M\left(S_t, x_t, W_{t+1}\right)$$

  - » Exogenous information

$$(S_0, W_1, W_2, \ldots, W_T)$$

Outline

- Canonical problems
- Elements of a dynamic model
- An energy storage illustration
- Solution strategies and problem classes
- Modeling uncertainty
- Designing policies

An energy storage problem

- Consider a basic energy storage problem:

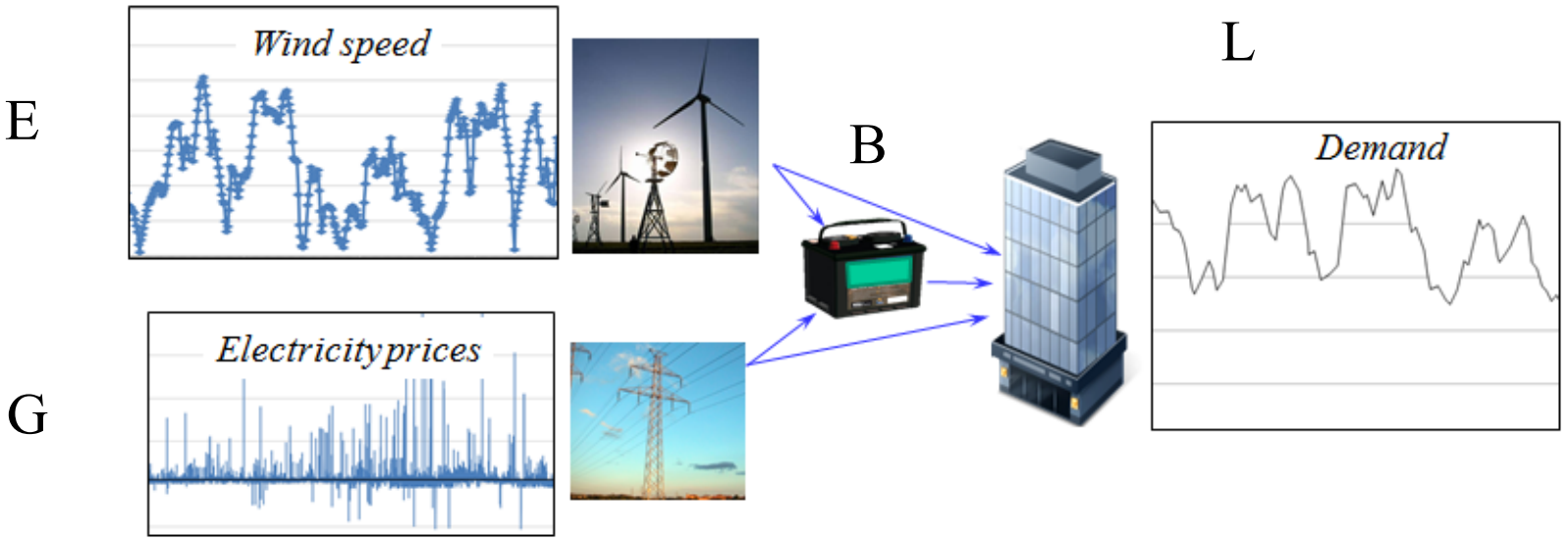
» We are going to show that with minor variations in the characteristics of this problem, we can make *each* class of policy work best.

An energy storage problem

- A model of our problem
  - » State variables
  - » Decision variables
  - » Exogenous information
  - » Transition function
  - » Objective function

An energy storage problem

- State variables

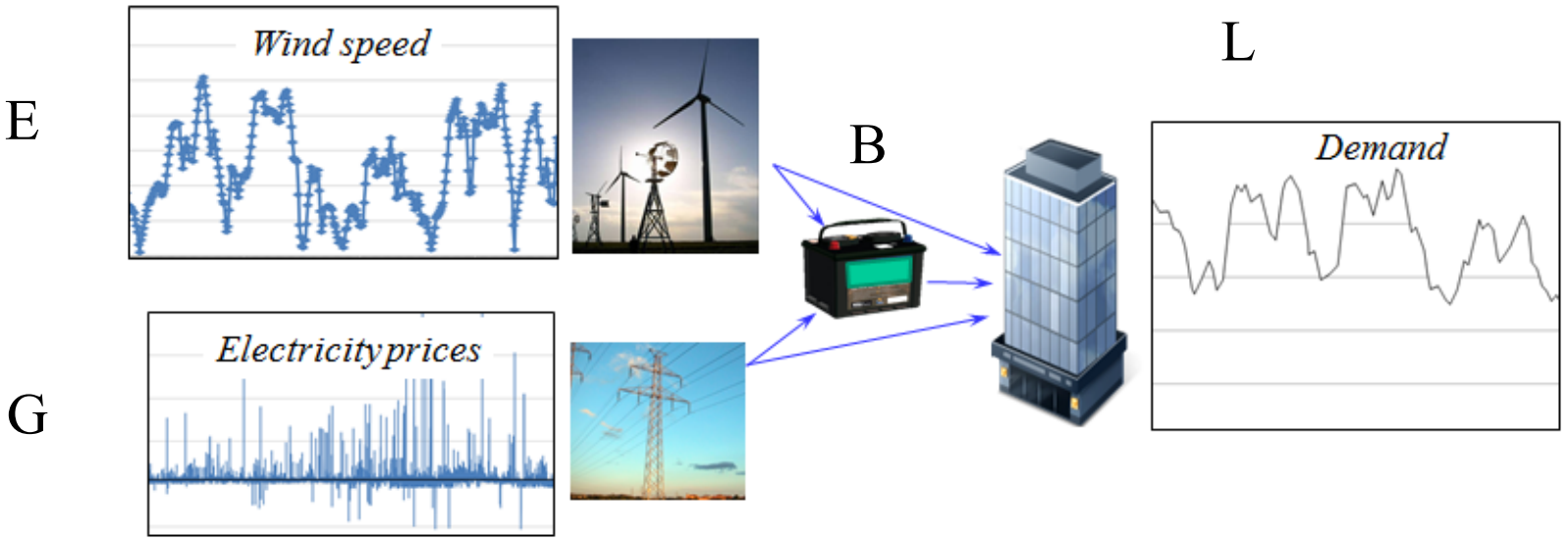


» We will present the full model, accumulating the information we need in the state variable.

» We will highlight information we need as we proceed. This information will make up our state variable.

An energy storage problem

- Decision variables



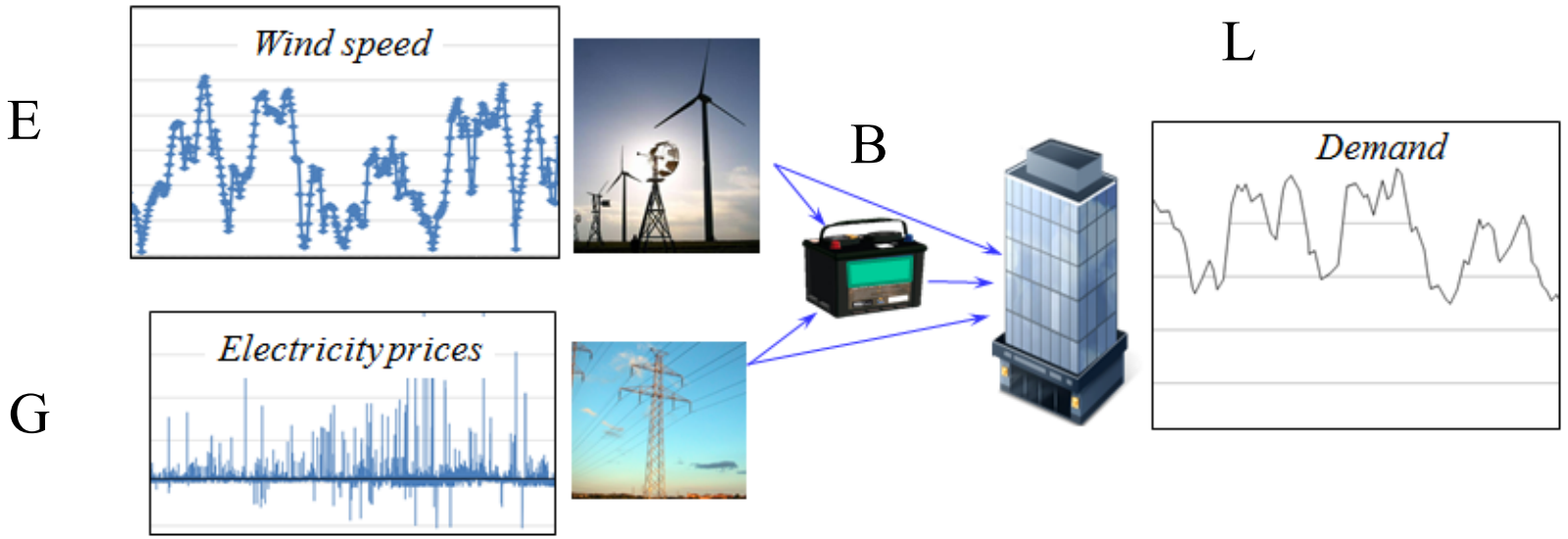
$$x_t = \left(x_t^{EL}, x_t^{EB}, x_t^{GL}, x_t^{GB}, x_t^{BL}, \right)$$

» Constraints;

$$\begin{aligned} x_t^{EL} + x_t^{EB} &\leq E_t, \\ (x_t^{GL} + x_t^{EL} + x_t^{BL}) &= L_t, \\ x_t^{BL} &\leq R_t, \end{aligned}$$

An energy storage problem

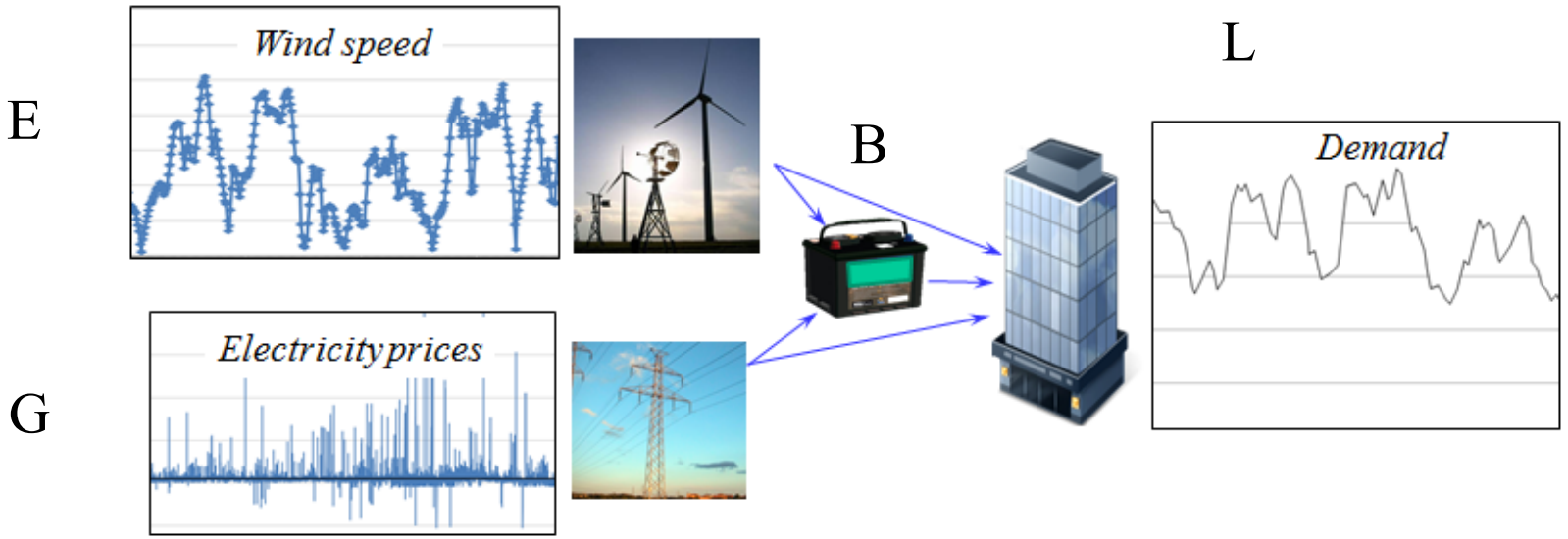
- Exogenous information



$$W_t = \begin{cases} \hat{E}_t = \text{Change in energy from wind between } t-1 \text{ and } t \\ \varepsilon_t^p = \text{Noise in the price process between } t-1 \text{ and } t \\ f_{tt'}^D = \text{Forecast of demand } D_{t'} \text{ provided by vendor at time } t \\ f_t^D = \left(f_{tt'}^D \right)_{t'>t} \quad \text{Provided exogenously} \\ \varepsilon_t^D = \text{Difference between actual demand and forecast} \end{cases}$$

An energy storage problem

- Transition function



$$E_{t+1} = E_t + \hat{E}_{t+1}$$

$$p_{t+1} = \bar{\theta}_{t0} p_t + \bar{\theta}_{t1} p_{t-1} + \bar{\theta}_{t2} p_{t-2} + \varepsilon^p_{t+1} = (\bar{\theta}_t)^T \bar{p}_t + \varepsilon^p_{t+1} \quad \bar{p}_t = \begin{pmatrix} p_t \\ p_{t-1} \\ p_{t-2} \end{pmatrix}$$

$$D_{t+1} = f^D_{t,t+1} + \varepsilon^D_{t+1}$$

$$R^{battery}_{t+1} = R^{battery}_t + x_t$$

Learning in stochastic optimization

- Updating the demand parameter
  - » Let p_{t+1} be the new price and let

$$\bar{F}_t^{price}(\bar{p}_t \mid \bar{\theta}_t) = (\bar{\theta}_t)^T \bar{p}_t = \bar{\theta}_{t0} p_t + \bar{\theta}_{t1} p_{t-1} + \bar{\theta}_{t2} p_{t-2}$$

  - » We update our estimate $\bar{\theta}_t$ using our recursive least squares equations:

$$\bar{\theta}_{t+1} = \bar{\theta}_t - \frac{1}{\gamma_{t+1}} B_t \bar{p}_t \varepsilon_{t+1}$$

$$\varepsilon_{t+1} = \bar{F}_t^{price}(\bar{p}_t \mid \bar{\theta}_t) - p_{t+1},$$

$$B_{t+1} = B_t - \frac{1}{\gamma_{t+1}} \left(B_t \bar{p}_t (\bar{p}_t)^T B_t \right)$$

$$\gamma_{t+1} = 1 + (\bar{p}_t)^T B_t \bar{p}_t$$

An energy storage problem

- Types of learning:
  - » No learning ($\theta's$ are known)

$$p_{t+1} = \theta_0 p_t + \theta_1 p_{t-1} + \theta_2 p_{t-2} + \varepsilon^p_{t+1}$$

  - » Passive learning (learn θs from price data)

$$p_{t+1} = \bar{\theta}_{t0} p_t + \bar{\theta}_{t1} p_{t-1} + \bar{\theta}_{t2} p_{t-2} + \varepsilon^p_{t+1}$$

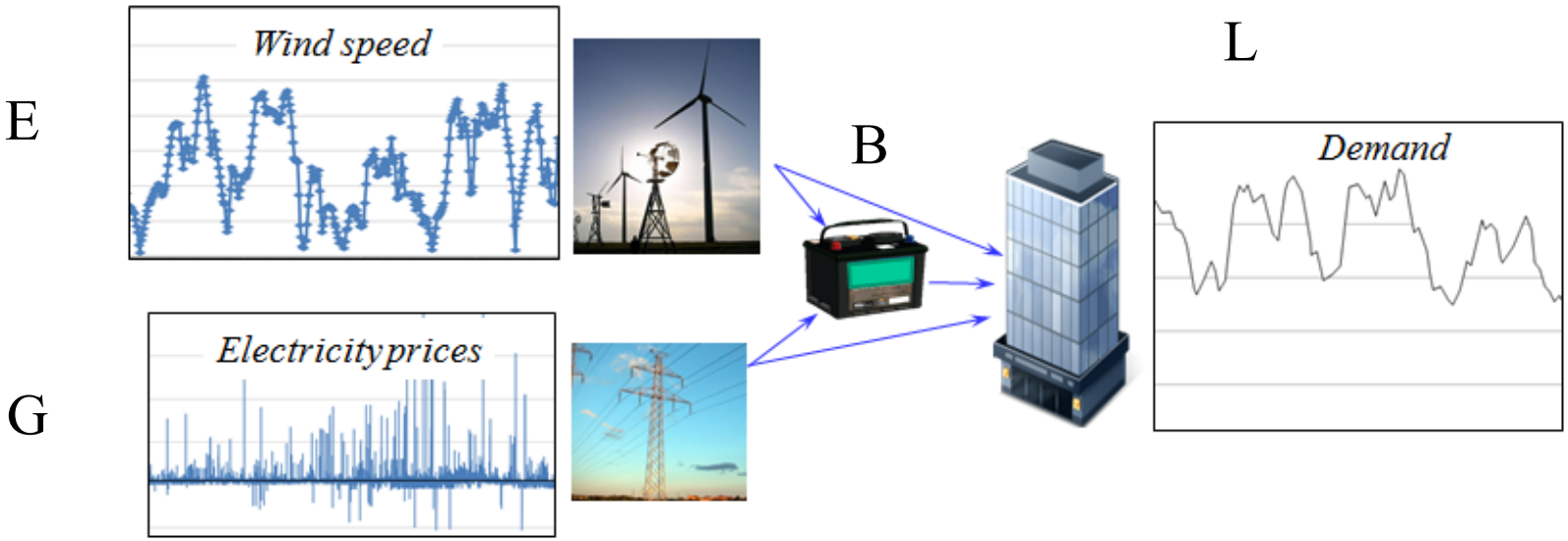
  - » Active learning ("bandit problems")

$$p_{t+1} = \bar{\theta}_{t0} p_t + \bar{\theta}_{t1} p_{t-1} + \bar{\theta}_{t2} p_{t-2} + \bar{\theta}_{t3} x^{GB}_t + \varepsilon^p_{t+1}$$

Buy/sell decisions

An energy storage problem

- Objective function



$$C(S_t, x_t) = p_t \left(x_t^{GB} + x_t^{GL} \right)$$

$$\min_\pi \mathbb{E} \left\{ \sum_{t=0}^{T} C_t(S_t, X_t^\pi(S_t), W_{t+1}) \mid S_0 \right\}$$

An energy storage problem

- State variables
  - » Cost function

    p_t = Price of electricity
  - » Decision function

    Constraints:

$$S_t = \left(\underbrace{E_t, L_t, R_t}, \underbrace{(p_t, p_{t-1}, p_{t-2})}, f_t^D, \underbrace{(\bar{\theta}_t, B_t)} \right)$$

$$\begin{aligned} x_t^{EL} + x_t^{EB} &\leq E_t \\ (x_t^{GL} + x_t^{EL} + x_t^{BL}) &= L_t \\ x_t^{BL} &\leq R_t \end{aligned}$$

  - » Transition function

$$p_{t+1} = \bar{\theta}_{t0} p_t + \bar{\theta}_{t1} p_{t-1} + \bar{\theta}_{t2} p_{t-2} + \varepsilon_{t+1}^p$$

$$D_{t+1} = f_{t,t+1}^D + \varepsilon_{t+1}^D$$

$$\bar{\theta}_{t+1} = \bar{\theta}_t - \frac{1}{\gamma_{t+1}} B_t \phi(x_t) \varepsilon_{t+1}$$

Outline

- Elements of a dynamic model
- An energy storage illustration
- Solution strategies and problem classes
- Modeling uncertainty
- Designing policies

Solution strategies and problem classes

- Special structure
  - » There are special cases where we can solve

$$\max_x \mathbb{E}F(x,W)$$

    exactly. But not very many.
- Sampled problems (SAA, scenario trees)
  - » If the only problem is that we cannot compute the expectation, we might solve a sampled approximation

$$\max_x \hat{\mathbb{E}}F(x,W)$$

- Adaptive learning algorithms
  - » This is what we have to turn to for most problems, and is the focus of this tutorial.

Solution strategies and problem classes

- State independent problems
  - » The *problem* does not depend on the state of the system.

$$\max_x \mathbb{E}F(x,W) = \mathbb{E}\left\{p\min(x,W) - cx\right\}$$

  - » The only state variable is what we know (or believe) about the unknown function $\mathbb{E}F(x,W)$, called the belief state B_t, so $S_t = B_t$.
- State dependent problems
  - » Now the *problem* may depend on what we know at time *t*:

$$\max_{0\le x\le R_t} \mathbb{E}C(S,x,W) = \mathbb{E}\left\{p_t \min(x,W) - cx\right\}$$

  - » Now the state is $S_t = (R_t, p_t, B_t)$

Solution strategies and problem classes

- Offline (final reward)
  - » We can iteratively search for the best solution, but only care about the final answer.
  - » Asymptotic formulation:

$$\max_x \mathbb{E} F(x, W)$$

  - » Finite horizon formulation:

$$\max_\pi \mathbb{E} F(x^{\pi,N}, W)$$

"ranking and selection" or "stochastic search"

- Online (cumulative reward)
  - » We have to learn as we go

$$\max_\pi \mathbb{E} \sum_{n=0}^{N-1} F(X^\pi(S^n), W^{n+1})$$

Solution strategies and problem classes

	Offline Terminal reward	Online Cumulative reward
State independent problems	$\max_\pi \mathbb{E}\{F(x^{\pi,N}, W)\|S_0\}$ Stochastic search (1)	$\max_\pi \mathbb{E}\{\sum_{n=0}^{N-1} F(X^\pi(S^n), W^{n+1})\|S_0\}$ Multiarmed bandit problem (2)
State dependent problems	$\max_{\pi^{lrn}} \mathbb{E}\{C(S, X^{\pi^{impl}}(S\|\theta^{imp}), W)\|S_0\}$ Offline dynamic programming (4)	$\max_\pi \mathbb{E}\{\sum_{t=0}^{T} C(S_t, X^\pi(S_t), W_{t+1})\|S_0\}$ Online dynamic programming (3)

Solution strategies and problem classes

	Offline Terminal reward	Online Cumulative reward
State independent problems	$\max_\pi \mathbb{E}\{F(x^{\pi,N}, W)\|S_0\}$ Stochastic search (1)	$\max_\pi \mathbb{E}\{\sum_{n=0}^{N-1} F(X^\pi(S^n), W^{n+1})\|S_0\}$ Multiarmed bandit problem (2)
State dependent problems	$\max_{\pi^{lrn}} \mathbb{E}\{C(S, X^{\pi^{impl}}(S\|\theta^{imp}), W)\|S_0\}$ Offline dynamic programming (4)	$\max_\pi \mathbb{E}\{\sum_{t=0}^{T} C(S_t, X^\pi(S_t), W_{t+1})\|S_0\}$ Online dynamic programming (3)

$$\max_{\pi^{lrn}} \mathbb{E}_{S^0} \mathbb{E}^{\pi^{imp}}_{(W_t^n)_{t=0}^T, n=1,\ldots,N|S^0} \left(\mathbb{E}^{\pi^{imp}}_{(\widehat{W}_t)_{t=0}^T|S^0} \frac{1}{T} \sum_{t=0}^{T-1} C(S_t, X^{\pi^{imp}}(S_t|\theta^{imp}), \widehat{W}_{t+1}) \right)$$

Solution strategies and problem classes

	Offline Terminal reward	Online Cumulative reward
State independent problems	$\max_\pi \mathbb{E}\{F(x^{\pi,N}, W)\|S_0\}$ Stochastic search (1)	$\max_\pi \mathbb{E}\{\sum_{n=0}^{N-1} F(X^\pi(S^n), W^{n+1})\|S_0\}$ Multiarmed bandit problem (2)
State dependent problems	$\max_{\pi^{lrn}} \mathbb{E}\{C(S, X^{\pi^{impl}}(S\|\theta^{imp}), W)\|S_0\}$ Offline dynamic programming (4)	$\max_\pi \mathbb{E}\{\sum_{t=0}^{T} C(S_t, X^\pi(S_t), W_{t+1})\|S_0\}$ Online dynamic programming (3)

Learning policies:

- Approximate dynamic programming
- Q-learning
- SDDP
- …

Solution strategies and problem classes

	Offline Terminal reward	Online Cumulative reward
State independent problems	$\max_\pi \mathbb{E}\{F(x^{\pi,N}, W)\|S_0\}$ Stochastic search (1)	$\max_\pi \mathbb{E}\{\sum_{n=0}^{N-1} F(X^\pi(S^n), W^{n+1})\|S_0\}$ Multiarmed bandit problem (2)
State dependent problems	$\max_{\pi^{lrn}} \mathbb{E}\{C(S, X^{\pi^{impl}}(S\|\theta^{imp}), W)\|S_0\}$ Offline dynamic programming (4)	$\max_\pi \mathbb{E}\{\sum_{t=0}^{T} C(S_t, X^\pi(S_t), W_{t+1})\|S_0\}$ Online dynamic programming (3)

"Online" (cumulative reward) dynamic programming is recognized as the "dynamic programming problem," but the entire literature on solving dynamic programs describes class (4) problems. This appears to be an open problem.

Outline

- Canonical problems
- Elements of a dynamic model
- An energy storage illustration
- Solution strategies and problem classes
- Modeling uncertainty
- Designing policies

Modeling uncertainty

- Observational uncertainty
- Prognostic uncertainty (forecasting)
- Experimental noise/variability
- Transitional uncertainty
- Inferential uncertainty
- Model uncertainty
- Systematic exogenous uncertainty
- Control/implementation uncertainty
- Algorithmic noise
- Goal uncertainty

Modeling uncertainty in the context of stochastic optimization is a relatively untapped area of research.

Outline

- Canonical problems
- Elements of a dynamic model
- An energy storage illustration
- Solution strategies and problem classes
- Modeling uncertainty
- Designing policies

Designing policies

- We have to start by describing what we mean by a policy.
  - » Definition:

    A policy is a mapping from a state to an action.

    … any mapping.

- How do we search over an arbitrary space of policies?

Designing policies

- Two fundamental strategies:

1) Policy search – Search over a class of functions for making decisions to optimize some metric.

$$\max_{\pi=(f\in F,\theta^f\in\Theta^f)} E\left\{\sum_{t=0}^{T} C\left(S_t, X_t^{\pi}(S_t\,|\,\theta)\right)\,|\,S_0\right\}$$

2) Lookahead approximations – Approximate the impact of a decision now on the future.

$$X_t^*(S_t) = \arg\max_{x_t}\left(C(S_t,x_t) + \mathbb{E}\left\{\max_{\pi\in\Pi}\left\{\mathbb{E}\sum_{t'=t+1}^{T} C(S_{t'}, X_{t'}^{\pi}(S_{t'}))\,|\,S_{t+1}\right\}|\,S_t, x_t\right\}\right)$$

Designing policies

- Policy search:

1a) Policy function approximations (PFAs) $x_t = X^{PFA}(S_t \mid \theta)$

- Lookup tables
  - "when in this state, take this action"
- Parametric functions
  - Order-up-to policies: if inventory is less than s, order up to S.
  - Affine policies - $x_t = X^{PFA}(S_t \mid \theta) = \sum_{f \in F} \theta_f \phi_f(S_t)$
  - Neural networks
- Locally/semi/non parametric
  - Requires optimizing over local regions

1b) Cost function approximations (CFAs)

- Optimizing a deterministic model modified to handle uncertainty (buffer stocks, schedule slack)

$$X^{CFA}(S_t \mid \theta) = \arg\max_{x_t \in \bar{X}_t^{\pi}(\theta)} \bar{C}^{\pi}(S_t, x_t \mid \theta)$$

Designing policies

- Lookahead approximations – Approximate the impact of a decision now on the future:
  - » An optimal policy (based on looking ahead):

$$X_t^*(S_t) = \arg\max_{x_t} \left(C(S_t, x_t) + \mathbb{E}\left\{ \max_{\pi \in \Pi} \left\{ \mathbb{E} \sum_{t'=t+1}^{T} C(S_{t'}, X_{t'}^{\pi}(S_{t'})) \mid S_{t+1} \right\} \mid S_t, x_t \right\} \right)$$

2a) Approximating the value of being in a downstream state using machine learning ("value function approximations")

$$X_t^*(S_t) = \arg\max_{x_t} \left(C(S_t, x_t) + \mathbb{E}\left\{ V_{t+1}(S_{t+1}) \mid S_t, x_t \right\} \right)$$

$$X_t^{VFA}(S_t) = \arg\max_{x_t} \left(C(S_t, x_t) + \mathbb{E}\left\{ \bar{V}_{t+1}(S_{t+1}) \mid S_t, x_t \right\} \right)$$

$$= \arg\max_{x_t} \left(C(S_t, x_t) + \bar{V}_t^x(S_t^x) \right)$$

Designing policies

- Lookahead approximations – Approximate the impact of a decision now on the future:
  - » An optimal policy (based on looking ahead):

$$X_t^*(S_t) = \arg\max_{x_t} \left(C(S_t, x_t) + \mathbb{E}\left\{ \max_{\pi \in \Pi} \left\{ \mathbb{E} \sum_{t'=t+1}^{T} C(S_{t'}, X_{t'}^{\pi}(S_{t'})) \mid S_{t+1} \right\} \mid S_t, x_t \right\} \right)$$

2a) Approximating the value of being in a downstream state using machine learning ("value function approximations")

$$X_t^*(S_t) = \arg\max_{x_t} \left(C(S_t, x_t) + \mathbb{E}\left\{ V_{t+1}(S_{t+1}) \mid S_t, x_t \right\} \right)$$

$$X_t^{VFA}(S_t) = \arg\max_{x_t} \left(C(S_t, x_t) + \mathbb{E}\left\{ \bar{V}_{t+1}(S_{t+1}) \mid S_t, x_t \right\} \right)$$

$$= \arg\max_{x_t} \left(C(S_t, x_t) + \bar{V}_t^x(S_t^x) \right)$$

Designing policies

- Lookahead approximations – Approximate the impact of a decision now on the future:
  - » An optimal policy (based on looking ahead):

$$X_t^*(S_t) = \arg\max_{x_t} \left(C(S_t, x_t) + \mathbb{E}\left\{ \max_{\pi \in \Pi} \left\{ \mathbb{E} \sum_{t'=t+1}^{T} C(S_{t'}, X_{t'}^{\pi}(S_{t'})) \mid S_{t+1} \right\} \mid S_t, x_t \right\} \right)$$

2a) Approximating the value of being in a downstream state using machine learning ("value function approximations")

$$X_t^*(S_t) = \arg\max_{x_t} \left(C(S_t, x_t) + \mathbb{E}\left\{ V_{t+1}(S_{t+1}) \mid S_t, x_t \right\} \right)$$

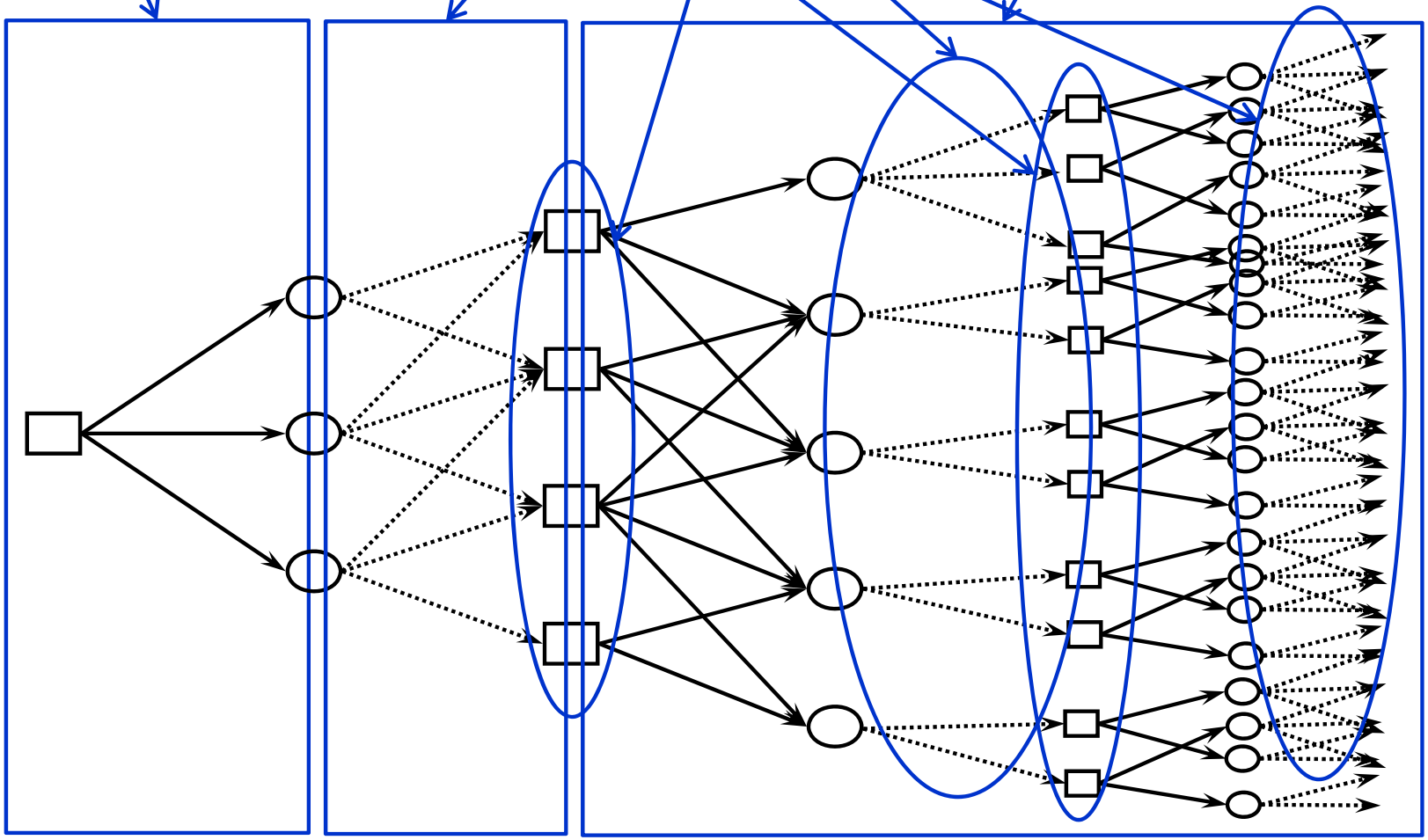
$$X_t^{VFA}(S_t) = \arg\max_{x_t} \left(C(S_t, x_t) + \mathbb{E}\left\{ \bar{V}_{t+1}(S_{t+1}) \mid S_t, x_t \right\} \right)$$

$$= \arg\max_{x_t} \left(C(S_t, x_t) + \bar{V}_t^x(S_t^x) \right)$$

Designing policies

- The ultimate lookahead policy is optimal

$$X_t^*(S_t) = \arg\max_{x_t} \left(C(S_t, x_t) + \mathbb{E}\left\{ \max_{\pi \in \Pi} \left\{ \mathbb{E} \sum_{t'=t+1}^{T} C(S_{t'}, X_{t'}^{\pi}(S_{t'})) \mid S_{t+1} \right\} \mid S_t, x_t \right\} \right)$$



Designing policies

- The ultimate lookahead policy is optimal

$$X_t^*(S_t) = \arg\max_{x_t} \left(C(S_t, x_t) + \mathbb{E}\left\{ \max_{\pi \in \Pi} \left\{ \mathbb{E} \sum_{t'=t+1}^{T} C(S_{t'}, X_{t'}^{\pi}(S_{t'})) \mid S_{t+1} \right\} \mid S_t, x_t \right\} \right)$$

Maximization that we cannot compute

Expectations that we cannot compute

Designing policies

- The ultimate lookahead policy is optimal

$$X_t^*(S_t) = \arg\max_{x_t}\left(C(S_t,x_t) + \mathbb{E}\left\{\max_{\pi\in\Pi}\left\{\mathbb{E}\sum_{t'=t+1}^{T} C(S_{t'}, X_{t'}^{\pi}(S_{t'}))\,|\,S_{t+1}\right\}|\,S_t,x_t\right\}\right)$$

» 2b) Instead, we have to solve an approximation called the *lookahead model:*

$$X_t^*(S_t) = \arg\max_{x_t}\left(C(S_t,x_t) + \tilde{\mathbb{E}}\left\{\max_{\tilde{\pi}\in\tilde{\Pi}}\left\{\tilde{\mathbb{E}}\sum_{t'=t+1}^{t+H} C(\tilde{S}_{tt'}, \tilde{X}_{tt'}^{\pi}(\tilde{S}_{tt'}))\,|\,\tilde{S}_{t,t+1}\right\}|\,S_t,x_t\right\}\right)$$

» *A lookahead policy* works by approximating the *lookahead model*.

Designing policies

- Types of lookahead approximations
  - » One-step lookahead – Widely used in pure learning policies:
    - Bayes greedy/naïve Bayes
    - Expected improvement
    - Value of information (knowledge gradient)
  - » Multi-step lookahead
    - Deterministic lookahead, also known as model predictive control, rolling horizon procedure
    - Stochastic lookahead:
      - Two-stage (widely used in stochastic linear programming)
      - Multistage
        - » Monte carlo tree search (MCTS) for discrete action spaces
        - » Multistage scenario trees (stochastic linear programming) – typically not tractable.

Four (meta)classes of policies

Policy search

1) **Policy function approximations (PFAs)**

» Lookup tables, rules, parametric/nonparametric functions

2) **Cost function approximation (CFAs)**

» $X^{CFA}(S_t \mid \theta) = \arg\max_{x_t \in \bar{X}_t^{\pi}(\theta)} \bar{C}^{\pi}(S_t, x_t \mid \theta)$

Lookahead approximations

3) **Policies based on value function approximations (VFAs)**

» $X_t^{VFA}(S_t) = \arg\max_{x_t} \left(C(S_t, x_t) + \bar{V}_t^{x} \left(S_t^{x}(S_t, x_t) \right) \right)$

4) **Direct lookahead policies (DLAs)**

» *Deterministic lookahead/rolling horizon proc./model predictive control*

$$X_t^{LA-D}(S_t) = \arg\max_{\tilde{x}_{tt}, \ldots, \tilde{x}_{t,t+H}} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}, \tilde{x}_{tt'})$$

» *Chance constrained programming*

$$P[A_t x_t \le f(W)] \le 1 - \delta$$

» *Stochastic lookahead /stochastic prog/Monte Carlo tree search*

$$X_t^{LA-S}(S_t) = \arg\max_{\tilde{x}_{tt}, \tilde{x}_{t,t+1}, \ldots, \tilde{x}_{t,t+T}} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{\tilde{\omega} \in \tilde{\Omega}_t} p(\tilde{\omega}) \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}(\tilde{\omega}), \tilde{x}_{tt'}(\tilde{\omega}))$$

» *"Robust optimization"*

$$X_t^{LA-RO}(S_t) = \arg\max_{\tilde{x}_{tt}, \ldots, \tilde{x}_{t,t+H}} \min_{w \in W_t(\theta)} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}(w), \tilde{x}_{tt'}(w))$$

Four (meta)classes of policies

Function approx.

1) **Policy function approximations (PFAs)**
   - » Lookup tables, rules, parametric/nonparametric functions
2) **Cost function approximation (CFAs)**
   - » $X^{CFA}(S_t \mid \theta) = \arg\max_{x_t \in \bar{X}_t^{\pi}(\theta)} \bar{C}^{\pi}(S_t, x_t \mid \theta)$
3) **Policies based on value function approximations (VFAs)**
   - » $X_t^{VFA}(S_t) = \arg\max_{x_t} \left(C(S_t, x_t) + \bar{V}_t^{x} \left(S_t^{x}(S_t, x_t) \right) \right)$
4) **Direct lookahead policies (DLAs)**
   - » *Deterministic lookahead/rolling horizon proc./model predictive control*

$$X_t^{LA-D}(S_t) = \arg\max_{\tilde{x}_{tt}, \ldots, \tilde{x}_{t,t+H}} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}, \tilde{x}_{tt'})$$

   - » *Chance constrained programming*

$$P[A_t x_t \le f(W)] \le 1 - \delta$$

   - » *Stochastic lookahead /stochastic prog/Monte Carlo tree search*

$$X_t^{LA-S}(S_t) = \arg\max_{\tilde{x}_{tt}, \tilde{x}_{t,t+1}, \ldots, \tilde{x}_{t,t+T}} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{\tilde{\omega} \in \tilde{\Omega}_t} p(\tilde{\omega}) \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}(\tilde{\omega}), \tilde{x}_{tt'}(\tilde{\omega}))$$

   - » *"Robust optimization"*

$$X_t^{LA-RO}(S_t) = \arg\max_{\tilde{x}_{tt}, \ldots, \tilde{x}_{t,t+H}} \min_{w \in W_t(\theta)} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}(w), \tilde{x}_{tt'}(w))$$

Four (meta)classes of policies

1) **Policy function approximations (PFAs)**

» Lookup tables, rules, parametric/nonparametric functions

2) **Cost function approximation (CFAs)**

» $X^{CFA}(S_t \mid \theta) = \arg\max_{x_t \in \bar{X}_t^{\pi}(\theta)} \bar{C}^{\pi}(S_t, x_t \mid \theta)$

3) **Policies based on value function approximations (VFAs)**

» $X_t^{VFA}(S_t) = \arg\max_{x_t} \left(C(S_t, x_t) + \bar{V}_t^{x} \left(S_t^{x}(S_t, x_t) \right) \right)$

4) **Direct lookahead policies (DLAs)**

» *Deterministic lookahead/rolling horizon proc./model predictive control*

$$X_t^{LA-D}(S_t) = \arg\max_{\tilde{x}_{tt}, \ldots, \tilde{x}_{t,t+H}} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}, \tilde{x}_{tt'})$$

» *Chance constrained programming*

$$P[A_t x_t \le f(W)] \le 1 - \delta$$

» *Stochastic lookahead /stochastic prog/Monte Carlo tree search*

$$X_t^{LA-S}(S_t) = \arg\max_{\tilde{x}_{tt}, \tilde{x}_{t,t+1}, \ldots, \tilde{x}_{t,t+T}} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{\tilde{\omega} \in \tilde{\Omega}_t} p(\tilde{\omega}) \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}(\tilde{\omega}), \tilde{x}_{tt'}(\tilde{\omega}))$$

» *"Robust optimization"*

$$X_t^{LA-RO}(S_t) = \arg\max_{\tilde{x}_{tt}, \ldots, \tilde{x}_{t,t+H}} \min_{w \in W_t(\theta)} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}(w), \tilde{x}_{tt'}(w))$$

Imbedded optimization

Learning problems

- Classes of learning problems in stochastic optimization

  1) Approximating the objective

  $$\bar{F}(x|\theta) \approx \mathbb{E}F(x, W).$$

  2) Designing a policy $X^{\pi}(S|\theta)$.

  3) A value function approximation

  $$\bar{V}_t(S_t|\theta) \approx V_t(S_t).$$

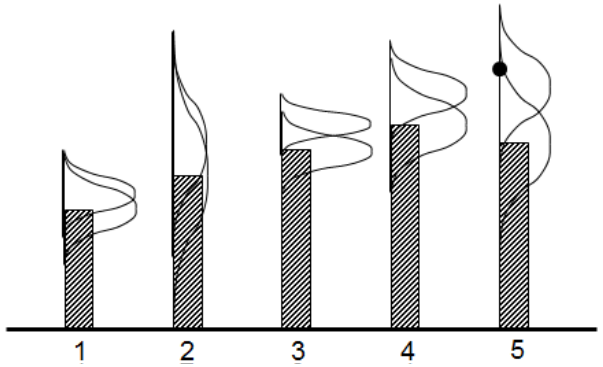
  4) Designing a cost function approximation:
     - The objective function $\bar{C}^{\pi}(S_t, x_t|\theta)$.
     - The constraints $X^{\pi}(S_t|\theta)$

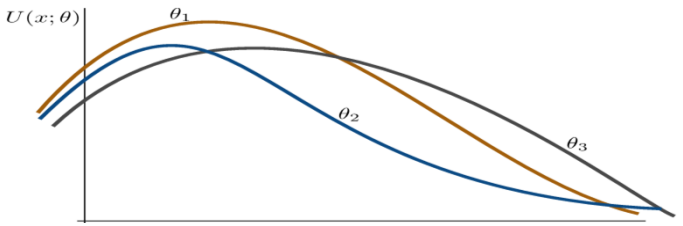
  5) Approximating the transition function

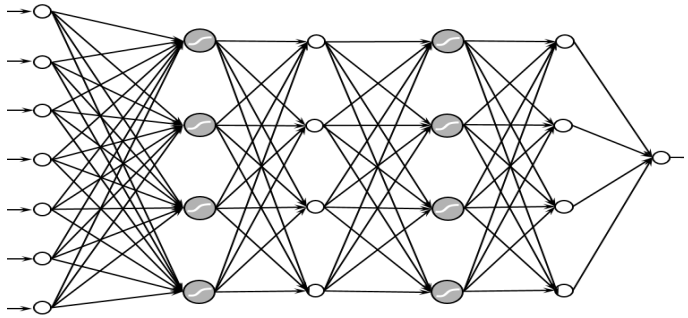
  $$\bar{S}^M(S_t, x_t, W_{t+1}|\theta) \approx S^M(S_t, x_t, W_{t+1})$$

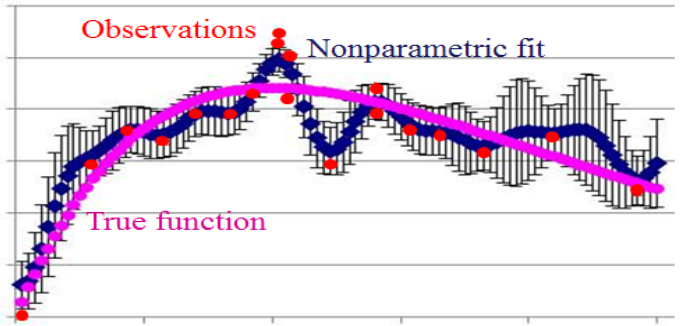
Approximation strategies

- Approximation strategies
  - » Lookup tables
    - Independent beliefs
    - Correlated beliefs
  - » Linear parametric models
    - Linear models
    - Sparse-linear
    - Tree regression
  - » Nonlinear parametric models
    - Logistic regression
    - Neural networks
  - » Nonparametric models
    - Gaussian process regression
    - Kernel regression
    - Support vector machines
    - Deep neural networks









Designing policies

- Finding the best policy
  - » We have to first articulate our classes of policies

$$f \in \mathcal{F} = \{PFAs, CFAs, VFAs, DLAs\}$$

$$\theta \in \Theta^f = \text{Parameters that characterize each family.}$$

  - » So minimizing over $\pi \in \Pi$ means:

$$\Pi = \left\{ f \in \mathcal{F}, \theta \in \Theta^f \right\}$$

  - » We then have to pick an objective such as

$$\max_\pi \mathbb{E} \sum_{t=0}^{T} C\left(S_t, X^\pi(S_t \mid \theta)\right) = \mathbb{E} \sum_{t=0}^{T} F\left(X^\pi(S_t \mid \theta), W_{t+1}\right)$$

or

$$\max_\pi \mathbb{E} C\left(S_T, X_T^\pi\right) = \mathbb{E} F(X_T^\pi, W)$$

Outline

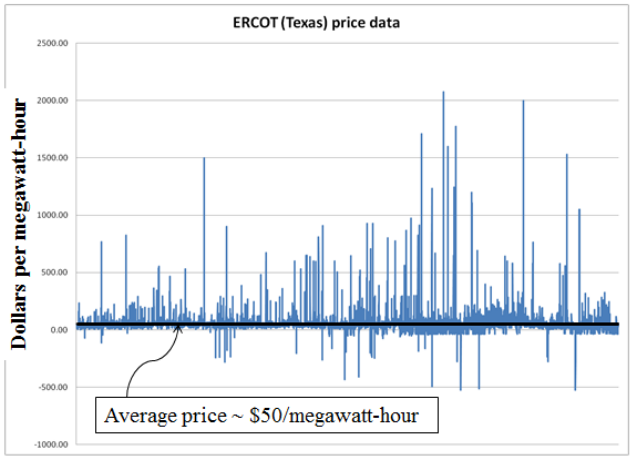
- The four classes of policies
  - » Policy function approximations (PFAs)
  - » Cost function approximations (CFAs)
  - » Value function approximations (VFAs)
  - » Direct lookahead policies (DLAs)
  - » A hybrid lookahead/CFA

Outline

- The four classes of policies
  - » Policy function approximations (PFAs)
  - » Cost function approximations (CFAs)
  - » Value function approximations (VFAs)
  - » Direct lookahead policies (DLAs)
  - » A hybrid lookahead/CFA

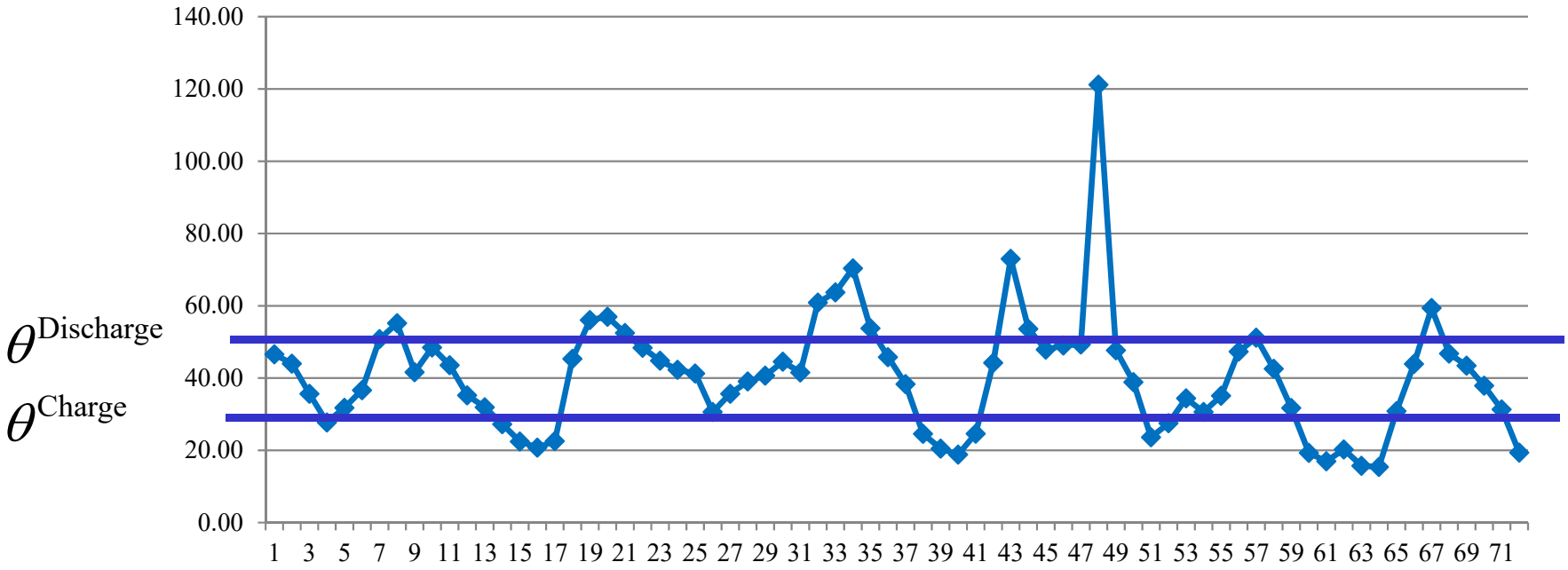
Policy function approximations

- Battery arbitrage – When to charge, when to discharge, given volatile LMPs



Policy function approximations

- Grid operators require that batteries bid charge and discharge prices, an hour in advance.

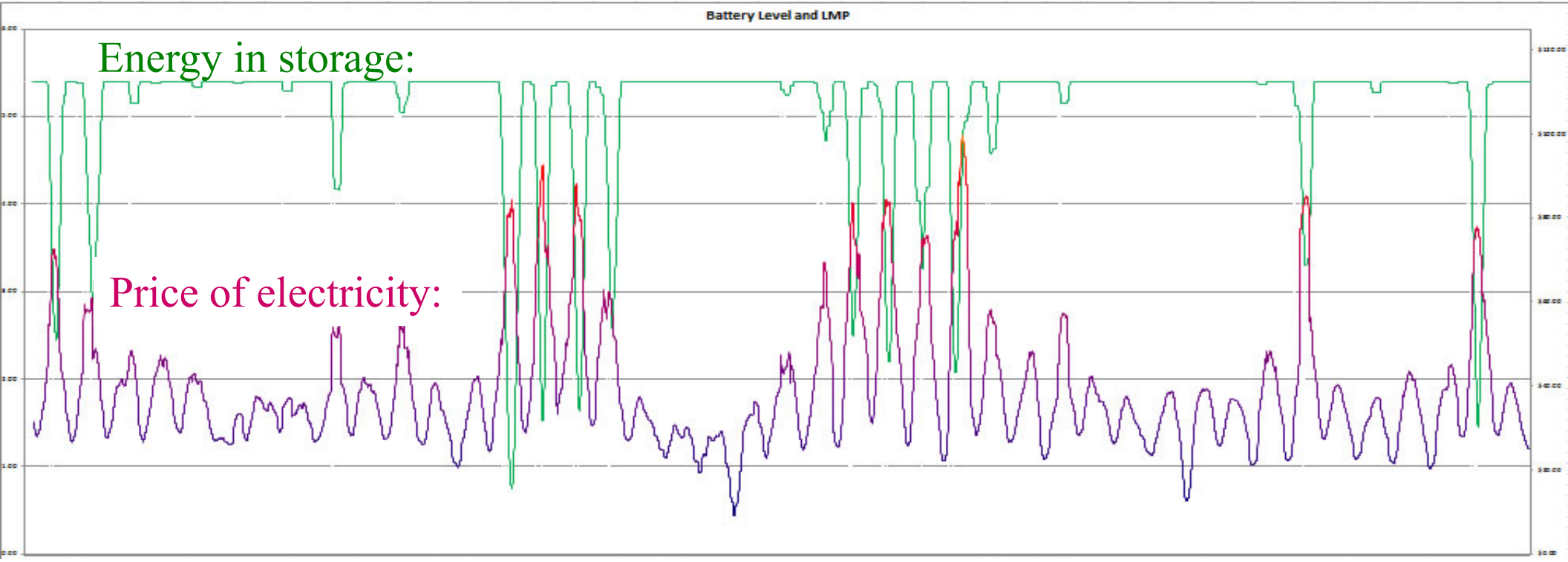


- We have to search for the best values for the policy parameters θ^{Charge} and $\theta^{\text{Discharge}}$.

Policy function approximations

- Our policy function might be the parametric model (this is nonlinear in the parameters):

$$X^{\pi}(S_t \mid \theta) = \begin{cases} +1 & \text{if } p_t < \theta^{\text{charge}} \\ 0 & \text{if } \theta^{\text{charge}} < p_t < \theta^{\text{discharge}} \\ -1 & \text{if } p_t > \theta^{\text{charge}} \end{cases}$$

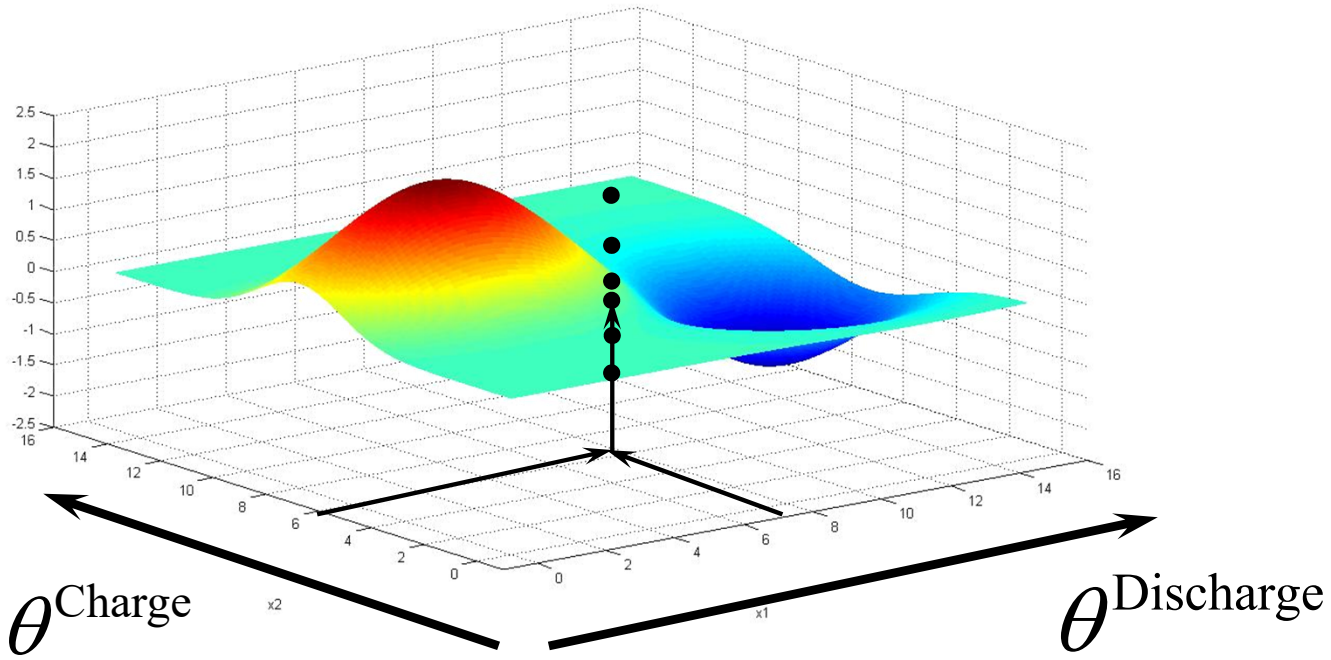


Policy function approximations

- Finding the best policy
  - » We need to maximize

$$\max_\theta F(\theta) = \mathbb{E}\sum_{t=0}^{T} \gamma^t C\left(S_t, X_t^\pi(S_t \mid \theta)\right)$$

  - » We cannot compute the expectation, so we run simulations:



Outline

- The four classes of policies
  - » Policy function approximations (PFAs)
  - » Cost function approximations (CFAs)
  - » Value function approximations (VFAs)
  - » Direct lookahead policies (DLAs)
  - » A hybrid lookahead/CFA

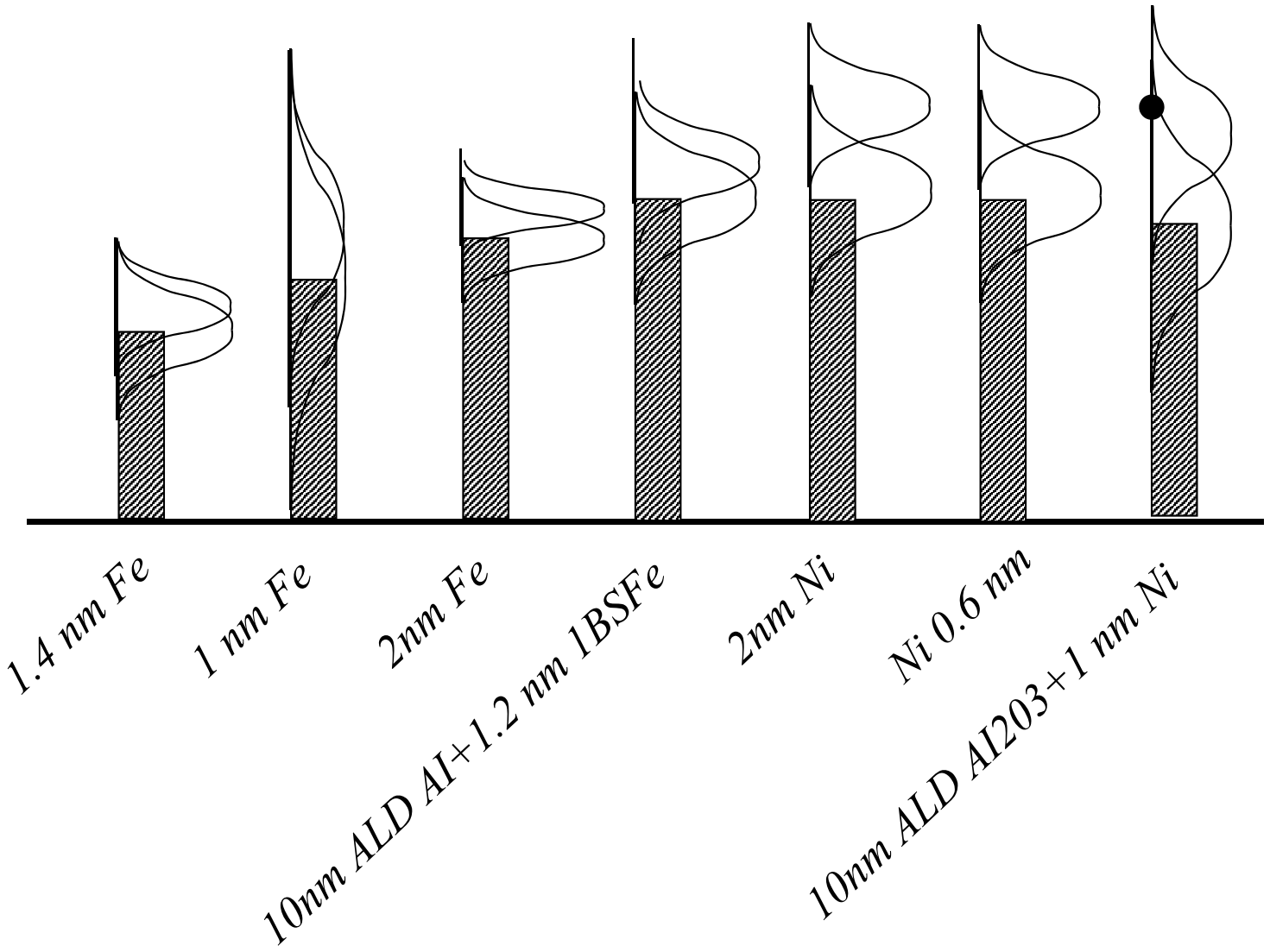
Cost function approximations

- Lookup table
  - » We can organize potential catalysts into groups
  - » Scientists using domain knowledge can estimate correlations in experiments between similar catalysts.

	1.4 nm Fe	1 nm Fe	2nm Fe	10nm ALD Al2O3+1.2 nm IBS Fe	2 nm Ni	Ni 0.6 nm	10nm ALD Al2O3+1 nm Ni
1.4 nm Fe	1	0.7	0.7	0.6	0.4	0.4	0.2
1 nm Fe	0.7	1	0.7	0.6	0.4	0.4	0.2
2nm Fe	0.7	0.7	1	0.6	0.4	0.4	0.2
10nm ALD Al2O3+1.2 nm IBS Fe	0.6	0.6	0.6	1	1	0.3	0
2 nm Ni	0.4	0.4	0.4	1	1	0.7	0.6
Ni 0.6 nm	0.4	0.4	0.4	0.3	0.7	1	0.6
10nm ALD Al2O3+1 nm Ni	0.2	0.2	0.2	0	0.6	0.6	1

Cost function approximations

- Correlated beliefs: Testing one material teaches us about other materials

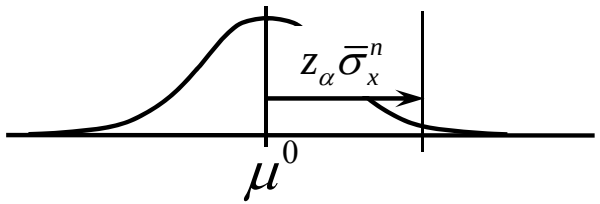


Cost function approximations

- Cost function approximations (CFA)
  - » Upper confidence bounding

$$X^{UCB}(S^n \mid \theta^{UCB}) = \arg\max_x \left(\bar{\mu}_x^n + \theta^{UCB} \sqrt{\frac{\log n}{N_x^n}} \right)$$

  - » Interval estimation



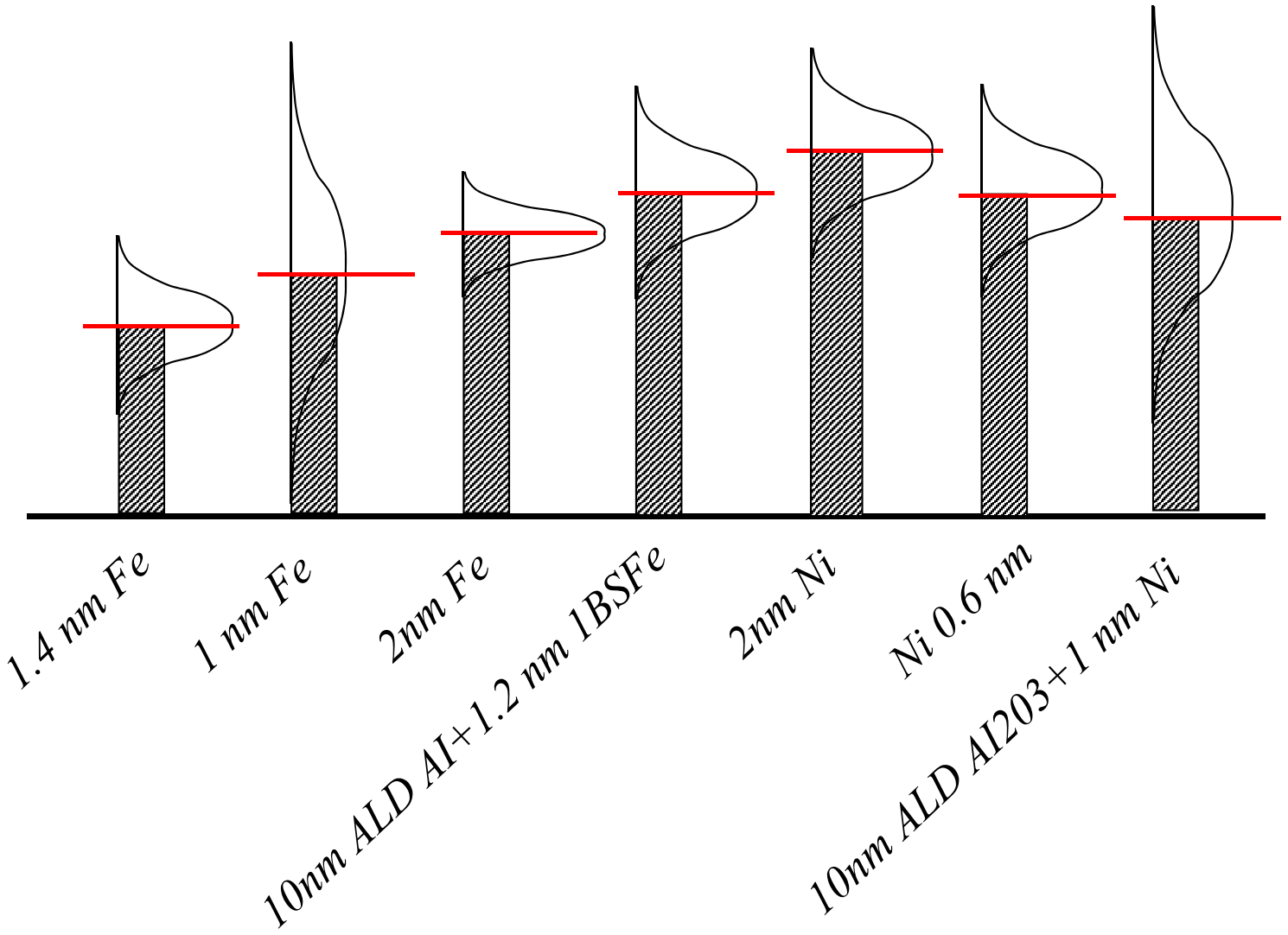
$$X^{IE}(S^n \mid \theta^{IE}) = \arg\max_x \left(\bar{\mu}_x^n + \theta^{IE} \bar{\sigma}_x^n \right)$$

  - » Boltzmann exploration ("soft max")
    - Choose x with probability: $P_x^n(\theta) = \dfrac{e^{\theta \bar{\mu}_x^n}}{\sum_{x'} e^{\theta \bar{\mu}_{x'}^n}}$

$$X^{Boltz}(S^n|\theta) = \arg\max_x \{x | P_x^n(\theta) \leq U\}.$$

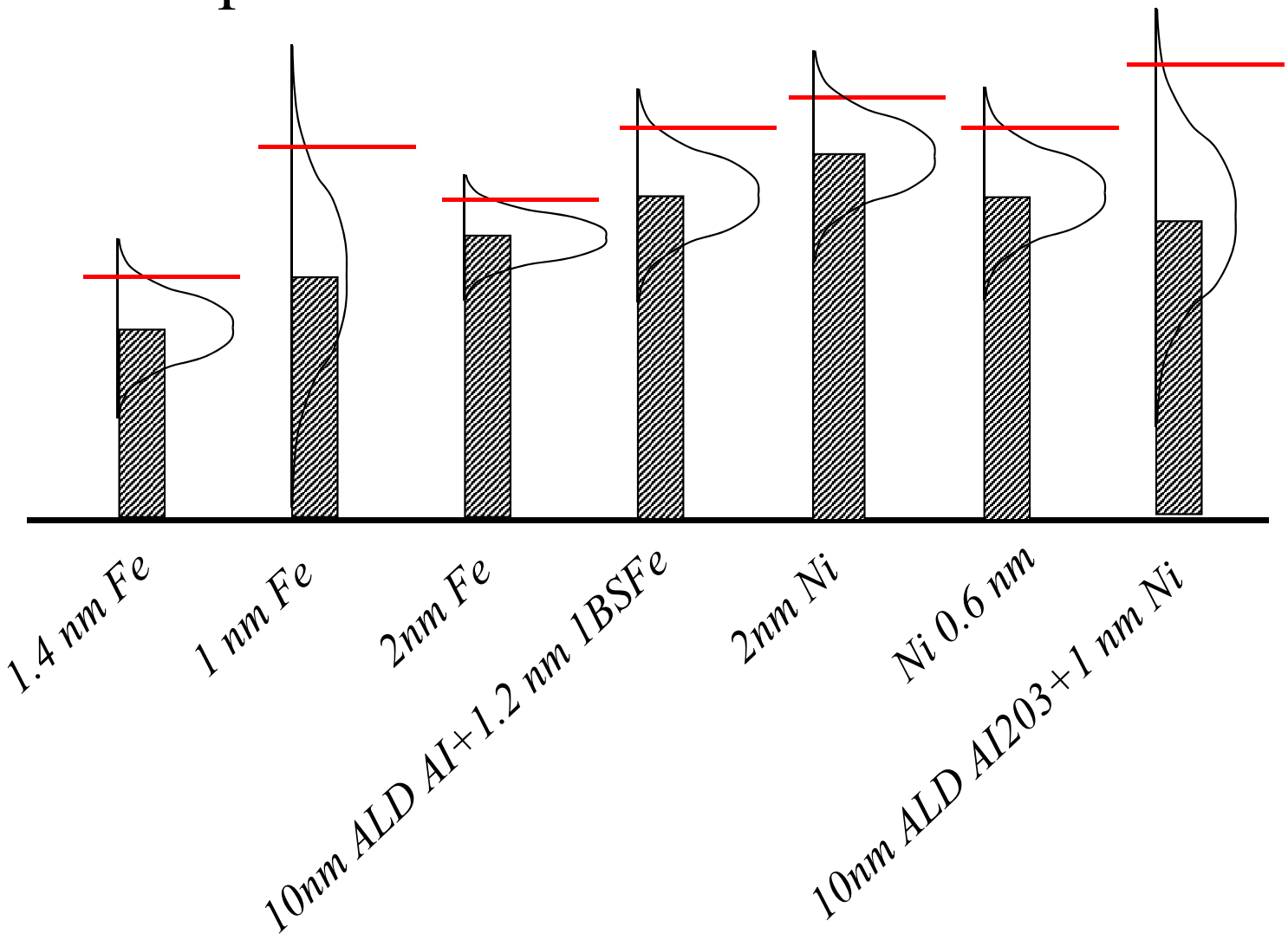
Cost function approximations

- Picking $\theta^{IE} = 0$ means we are evaluating each choice at the mean.



Cost function approximations

- Picking $\theta^{IE} = 2$ means we are evaluating each choice at the 95th percentile.



Cost function approximations

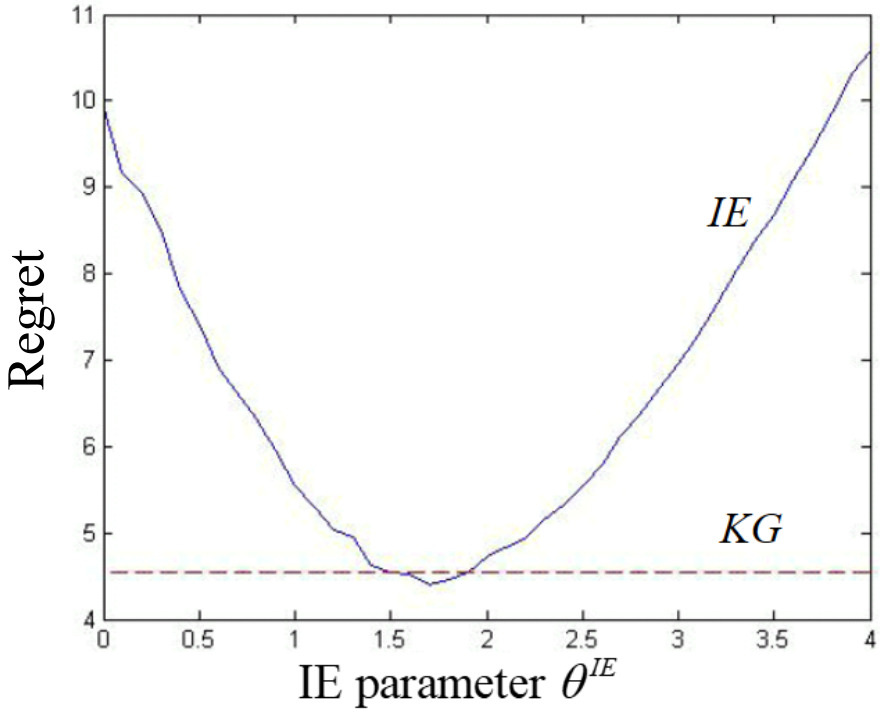
- Optimizing the policy
  - » We optimize θ^{IE} to maximize:

$$\max_{\theta^{IE}} F(\theta^{IE}) = \mathbb{E}F\left(x^{\pi,N}, W\right)$$

where

$$x^n = X^{IE}(S^n \mid \theta^{IE}) = \arg\max_x \left(\bar{\mu}_x^n + \theta^{IE}\bar{\sigma}_x^n\right) \qquad S^n = (\bar{\mu}_x^n, \bar{\sigma}_x^n)$$

- Notes:
  - » This can handle any belief model, including correlated beliefs, nonlinear belief models.
  - » All we require is that we be able to simulate a policy.



Cost function approximations

- Inventory management

  - » How much product should I order to anticipate future demands?

  - » Need to accommodate different sources of uncertainty.
    - Market behavior
    - Transit times
    - Supplier uncertainty
    - Product quality

Cost function approximations

- Imagine that we want to purchase parts from different suppliers. Let x_{tp} be the amount of product we purchase at time t from supplier p to meet forecasted demand D_t. We would solve

$$X_t^{\pi}(S_t) = \arg\min_{x_t} \sum_{p \in P} c_p x_{tp}$$

subject to

$$\left.\begin{aligned} \sum_{p \in P} x_{tp} &\geq D_t \\ x_{tp} &\leq u_p \\ x_{tp} &\geq 0 \end{aligned}\right\} \mathcal{X}_t$$

» This assumes our demand forecast D_t is accurate.

Cost function approximations

- Imagine that we want to purchase parts from different suppliers. Let x_{tp} be the amount of product we purchase at time t from supplier p to meet forecasted demand D_t. We would solve

$$X_t^{\pi}(S_t \mid \theta) = \arg\min_{x_t \in \mathcal{X}^{\pi}(\theta)} \sum_{p \in P} c_p x_{tp}$$

subject to

$$\sum_{p \in P} x_{tp} \geq \theta^{\text{Reserve}} D_t$$

$$x_{tp} \leq u_p$$

$$x_{tp} \geq \theta^{\text{buffer}}$$

$\mathcal{X}_t^{\pi}(\theta)$

Reserve

Buffer stock

» This is a "parametric cost function approximation"

Cost function approximations

- A general way of creating CFAs:
  - » Define our policy:

$$X_t^\pi(\theta) = \arg\min_x \bar{C}^\pi(S_t, x_t \mid \theta)$$

Parametrically modified costs

subject to

$$Ax = \bar{b}^\pi(\theta)$$

Parametrically modified constraints

  - » We tune θ by optimizing:

$$\min_\theta F^\pi(\theta) = \mathbb{E}\left\{\sum_{t=0}^{T} C(S_t, X_t^\pi(\theta)) \mid S_0\right\}$$

Outline

- The four classes of policies
  - » Policy function approximations (PFAs)
  - » Cost function approximations (CFAs)
  - » Value function approximations (VFAs)
  - » Direct lookahead policies (DLAs)
  - » A hybrid lookahead/CFA

Value function approximations

- Q-learning (for discrete actions)

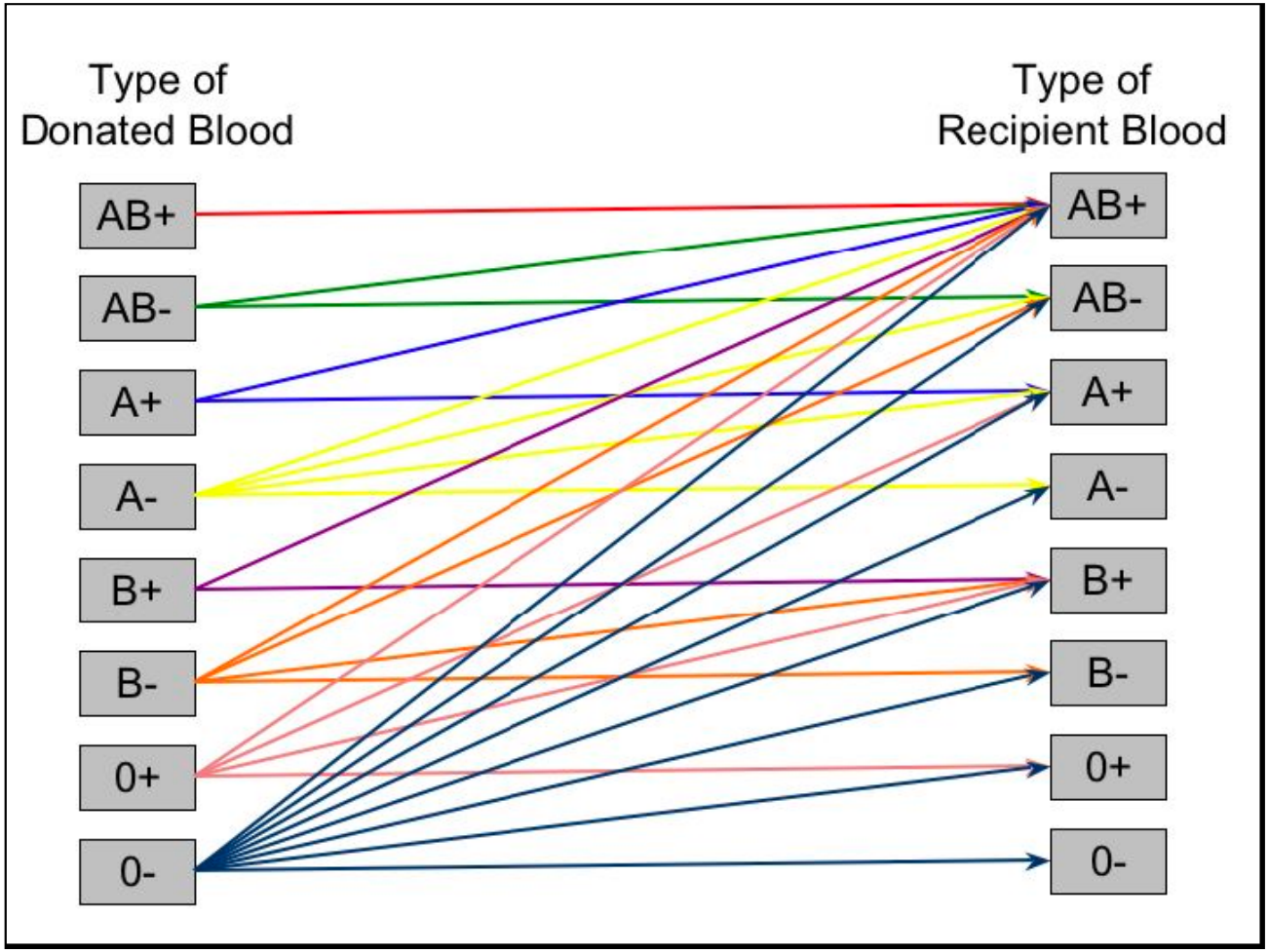
$$\hat{q}^n(s^n, a^n) = r(s^n, a^n) + \gamma \max_{a'} \bar{Q}^{n-1}(s', a')$$

$$\bar{Q}^n(s^n, a^n) = (1 - \alpha_{n-1})\bar{Q}^{n-1}(s^n, a^n) + \alpha_{n-1}\hat{q}^n(s^n, a^n)$$

» But what if the action a is a vector?

Blood management

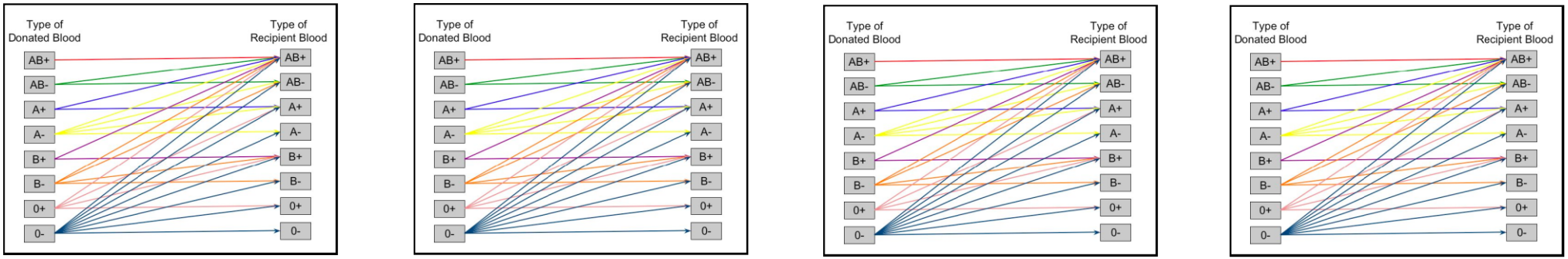
- Managing blood inventories

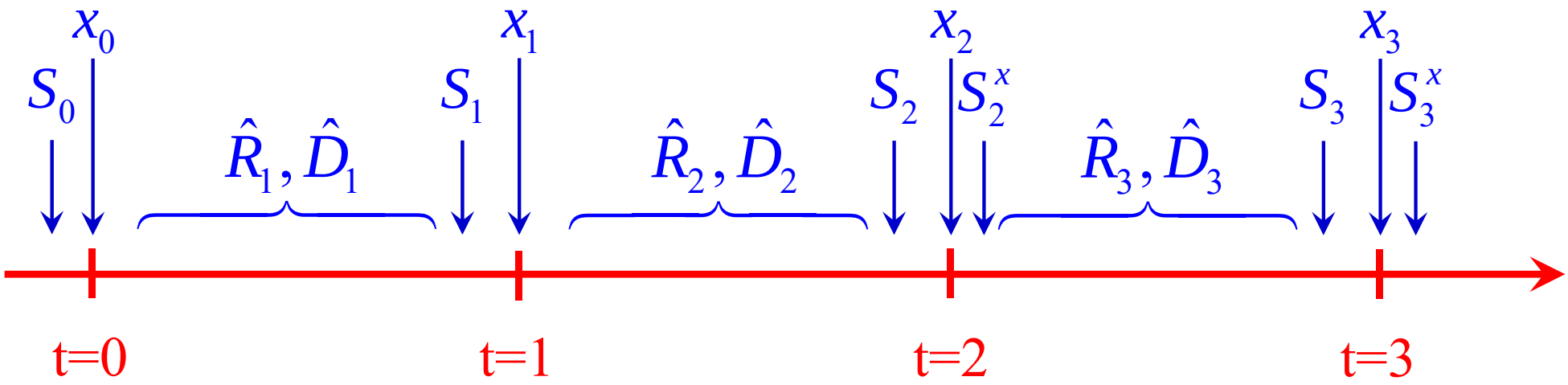


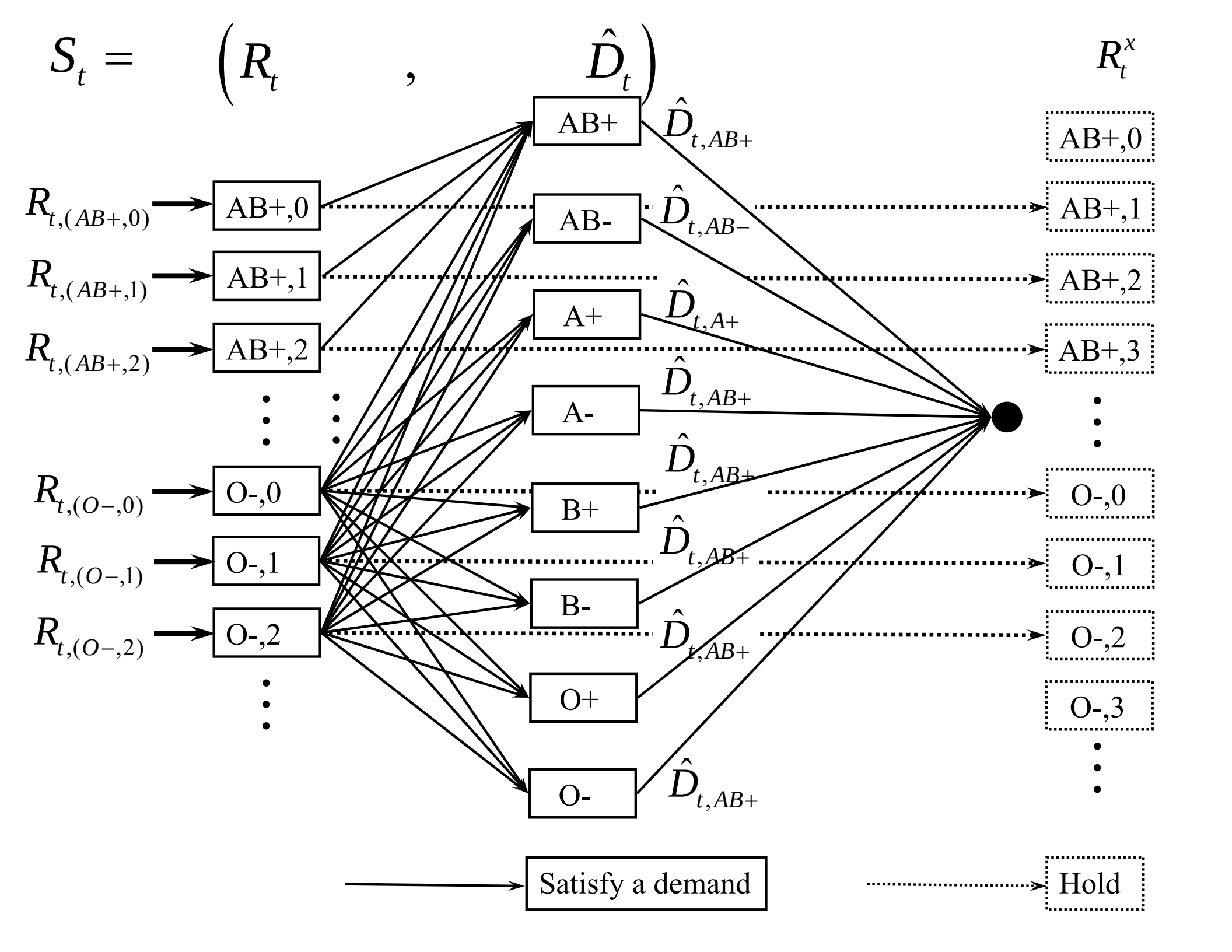
Blood management

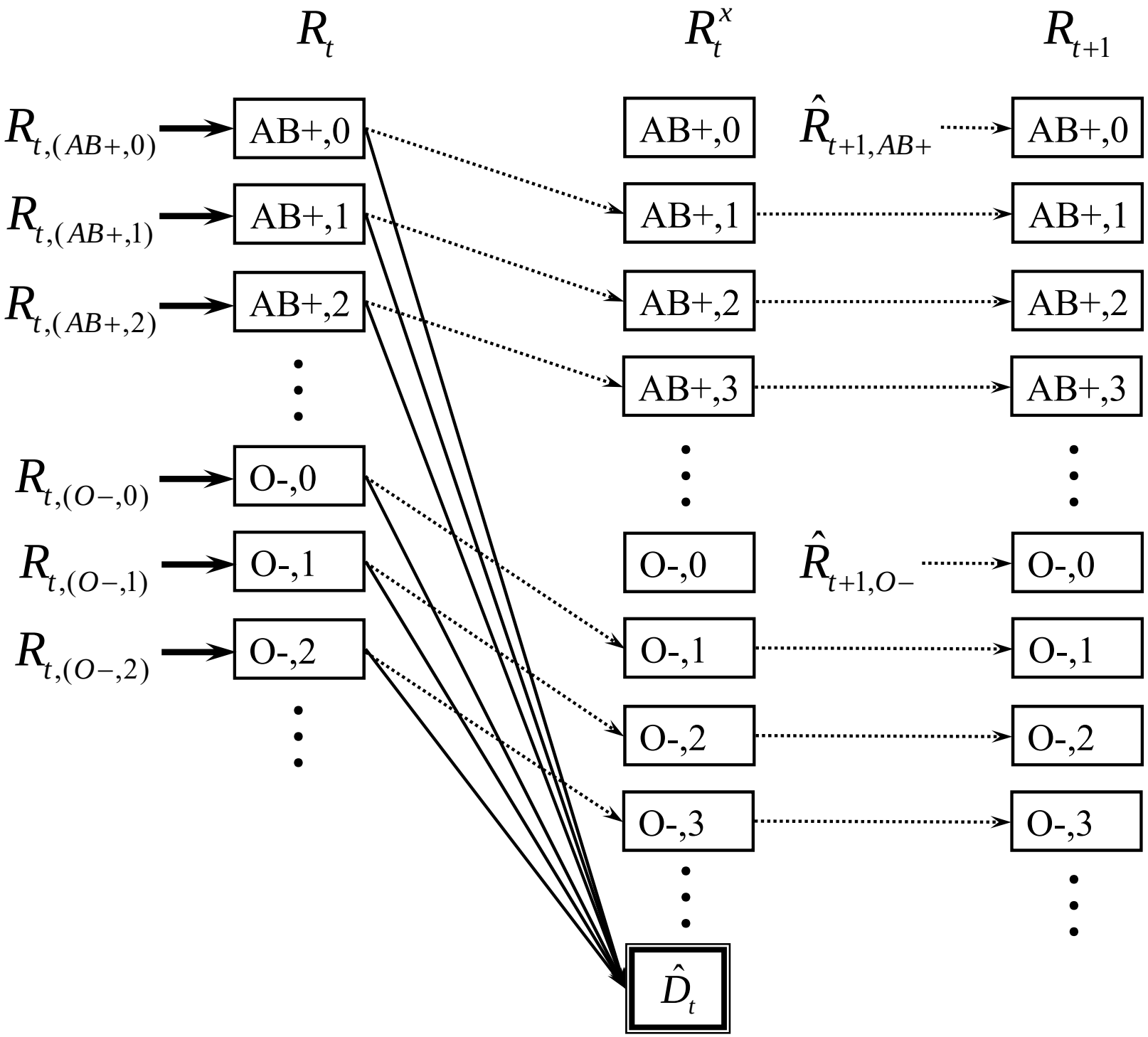
- Managing blood inventories over time

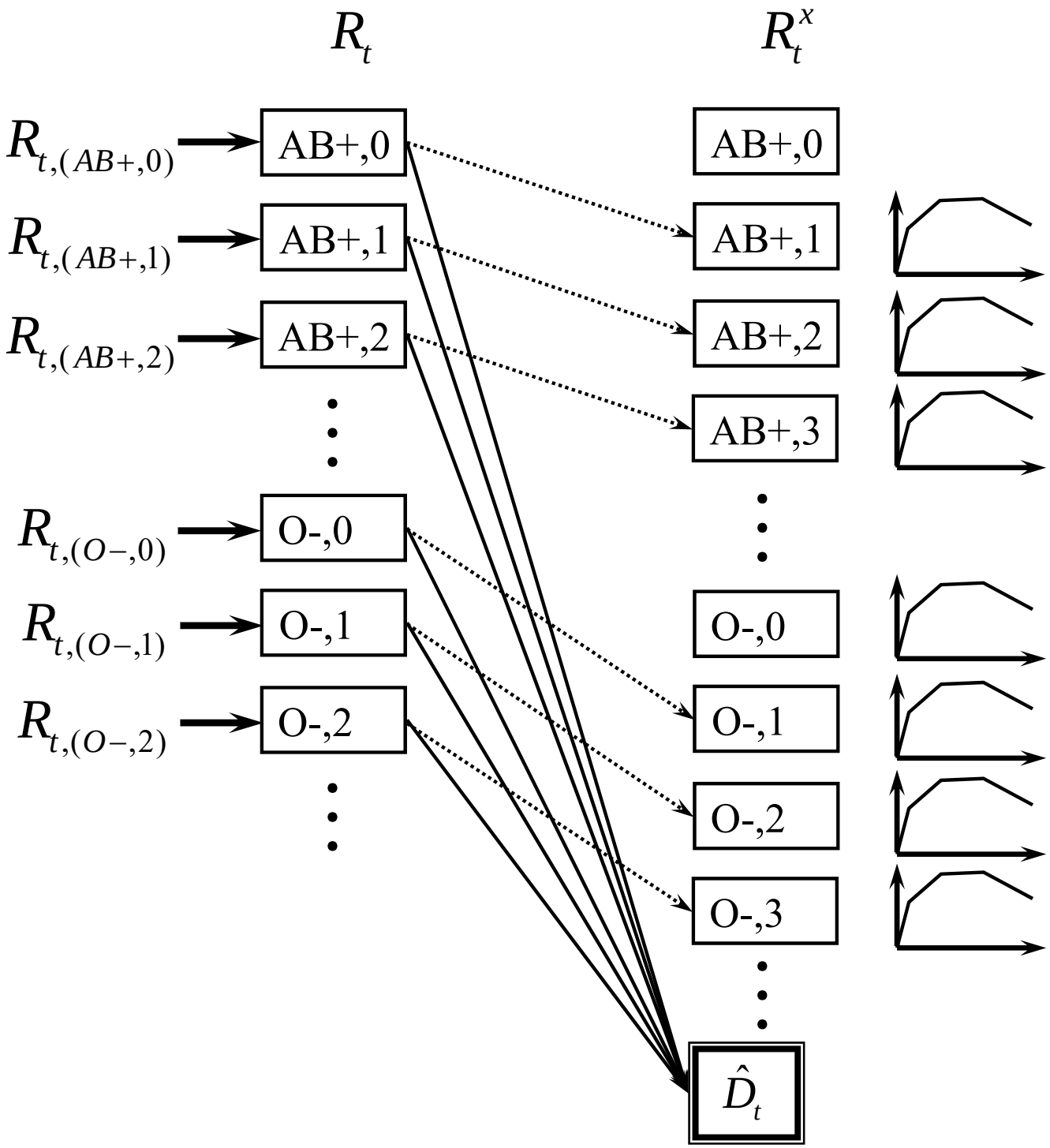
Week 0 Week 1 Week 2 Week 3

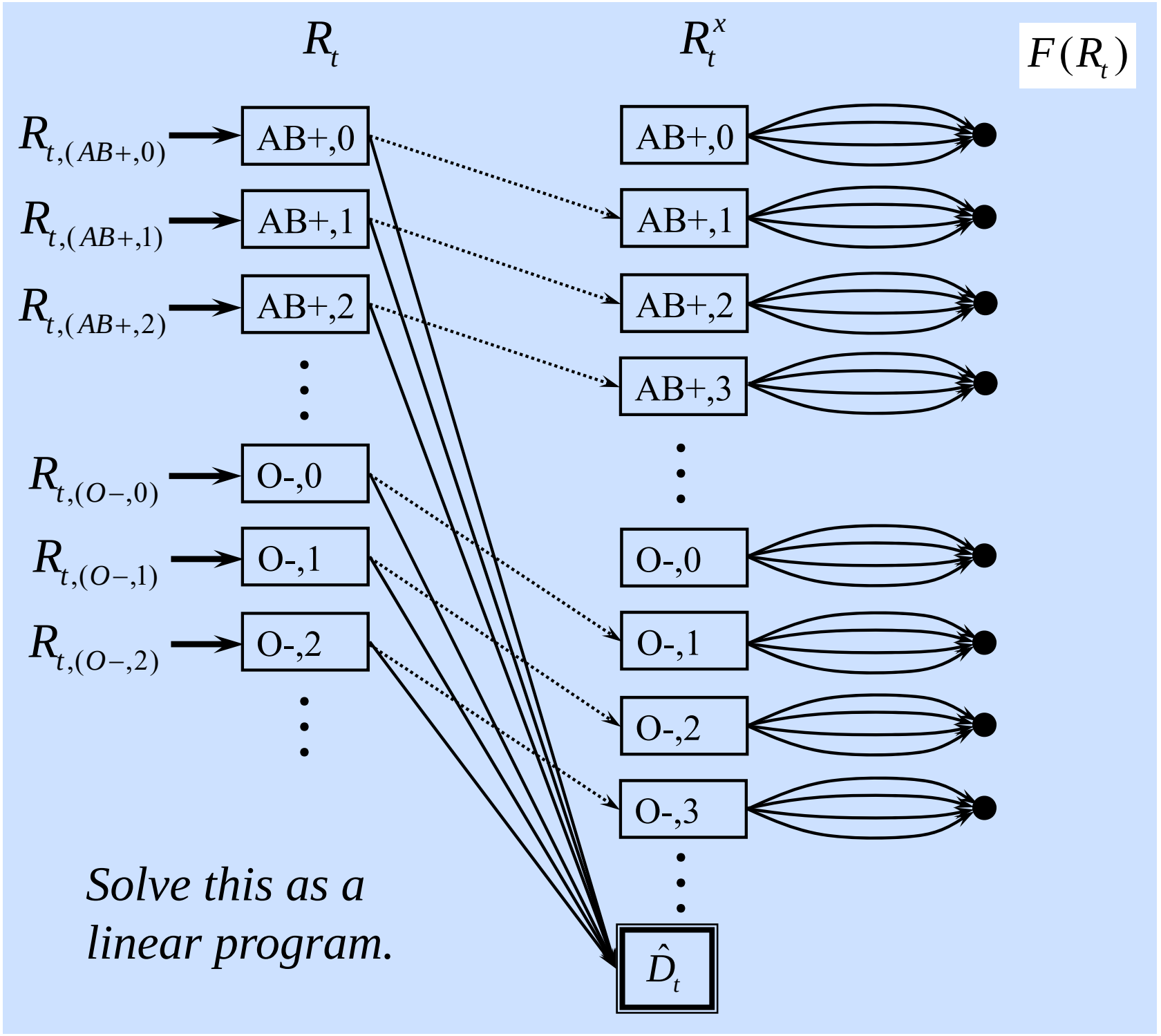


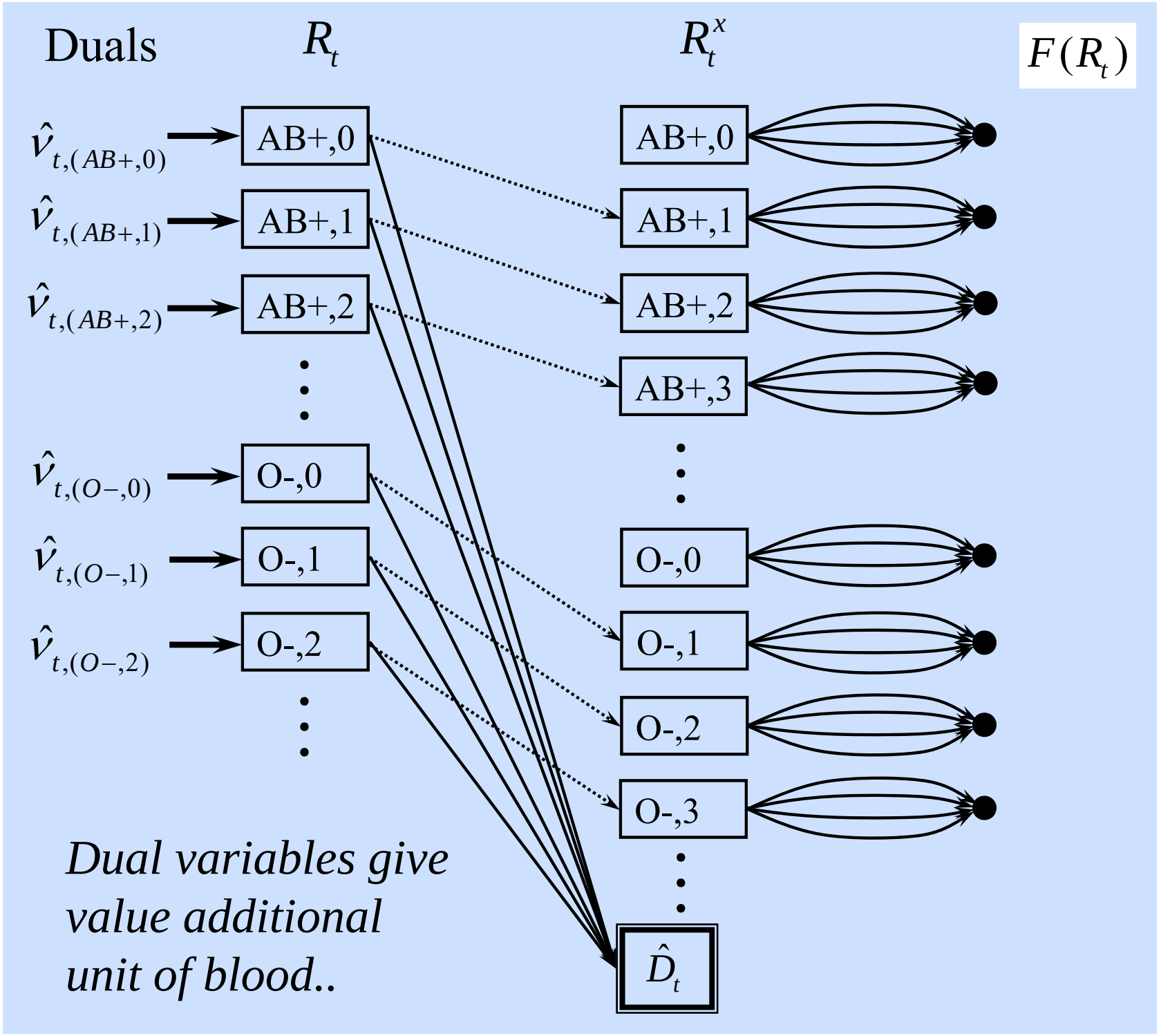



$S_t = \left(R_t \quad , \quad \hat{D}_t \right)$
R_t^x
AB+
AB-
A+
A-
B+
B-
O+
O-
$\hat{D}_{t,AB+}$
$\hat{D}_{t,AB-}$
$\hat{D}_{t,A+}$
$\hat{D}_{t,AB+}$
$\hat{D}_{t,AB+}$
$\hat{D}_{t,AB+}$
$\hat{D}_{t,AB+}$
$\hat{D}_{t,AB+}$
AB+,0
AB+,1
AB+,2
AB+,3
O-,0
O-,1
O-,2
O-,3
$R_{t,(AB+,0)}$
$R_{t,(AB+,1)}$
$R_{t,(AB+,2)}$
$R_{t,(O-,0)}$
$R_{t,(O-,1)}$
$R_{t,(O-,2)}$
AB+,0
AB+,1
AB+,2
O-,0
O-,1
O-,2
Satisfy a demand
Hold


R_t
R_t^x
R_{t+1}
$R_{t,(AB+,0)}$
$R_{t,(AB+,1)}$
$R_{t,(AB+,2)}$
$R_{t,(O-,0)}$
$R_{t,(O-,1)}$
$R_{t,(O-,2)}$
AB+,0
AB+,1
AB+,2
O-,0
O-,1
O-,2
AB+,0
AB+,1
AB+,2
AB+,3
O-,0
O-,1
O-,2
O-,3
$\hat{D}_t$
$\hat{R}_{t+1,AB+}$
$\hat{R}_{t+1,O-}$
AB+,0
AB+,1
AB+,2
AB+,3
O-,0
O-,1
O-,2
O-,3

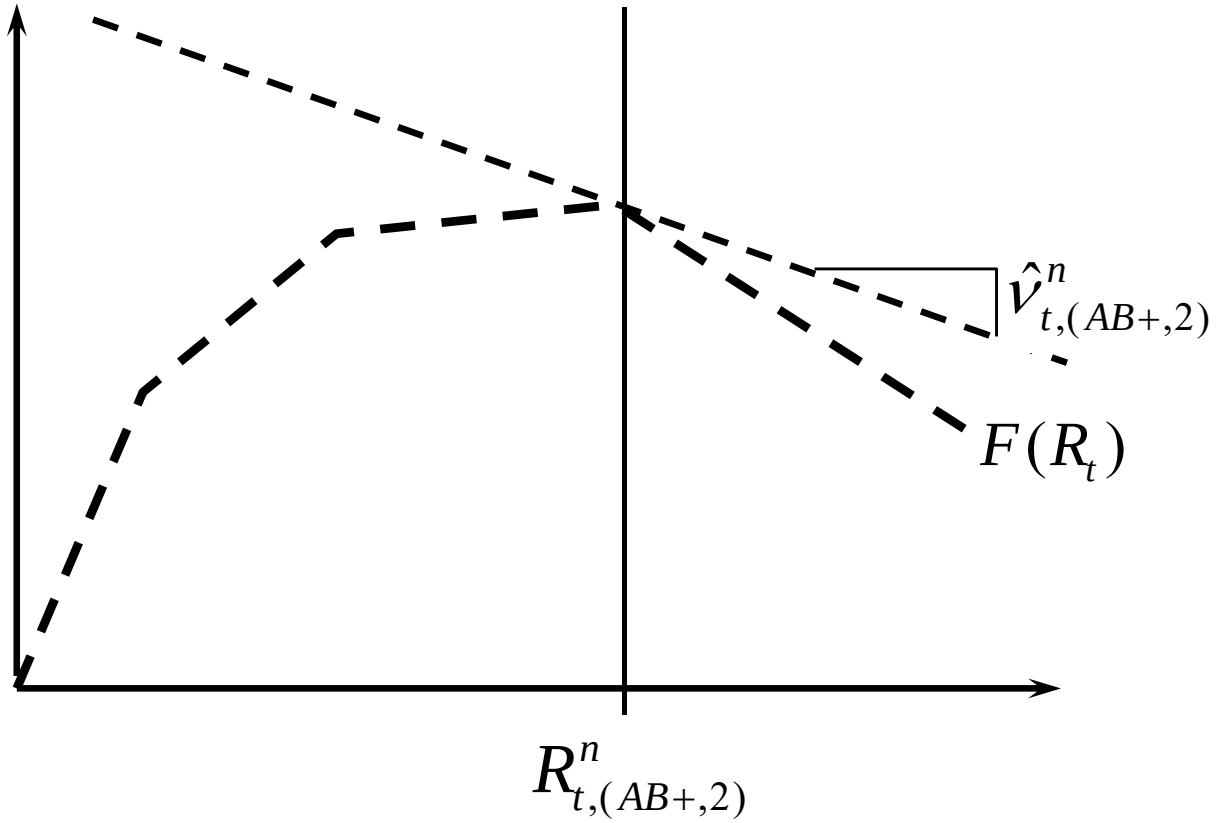

R_t
R_t^x
$R_{t,(AB+,0)}$
$R_{t,(AB+,1)}$
$R_{t,(AB+,2)}$
$R_{t,(O-,0)}$
$R_{t,(O-,1)}$
$R_{t,(O-,2)}$
AB+,0
AB+,1
AB+,2
O-,0
O-,1
O-,2
AB+,0
AB+,1
AB+,2
AB+,3
O-,0
O-,1
O-,2
O-,3
$\hat{D}_t$


R_t
R_t^x
$F(R_t)$
$R_{t,(AB+,0)}$
$R_{t,(AB+,1)}$
$R_{t,(AB+,2)}$
$R_{t,(O-,0)}$
$R_{t,(O-,1)}$
$R_{t,(O-,2)}$
AB+,0
AB+,1
AB+,2
O-,0
O-,1
O-,2
AB+,0
AB+,1
AB+,2
AB+,3
O-,0
O-,1
O-,2
O-,3
$\hat{D}_t$
Solve this as a linear program.


Duals
R_t
R_t^x
$F(R_t)$
$\hat{\nu}_{t,(AB+,0)}$
$\hat{\nu}_{t,(AB+,1)}$
$\hat{\nu}_{t,(AB+,2)}$
$\hat{\nu}_{t,(O-,0)}$
$\hat{\nu}_{t,(O-,1)}$
$\hat{\nu}_{t,(O-,2)}$
AB+,0
AB+,1
AB+,2
O-,0
O-,1
O-,2
AB+,0
AB+,1
AB+,2
AB+,3
O-,0
O-,1
O-,2
O-,3
$\hat{D}_t$
Dual variables give value additional unit of blood..

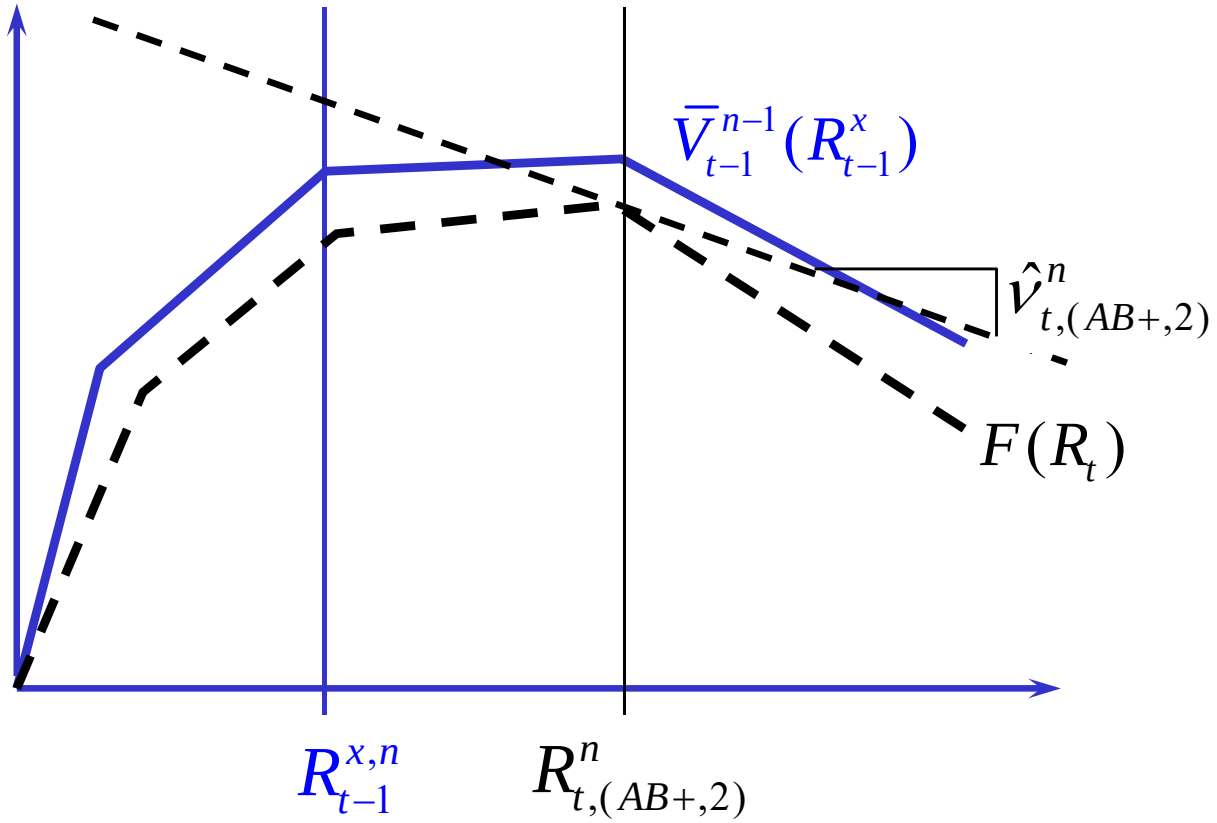
Updating the value function approximation

- Estimate the gradient at R_t^n



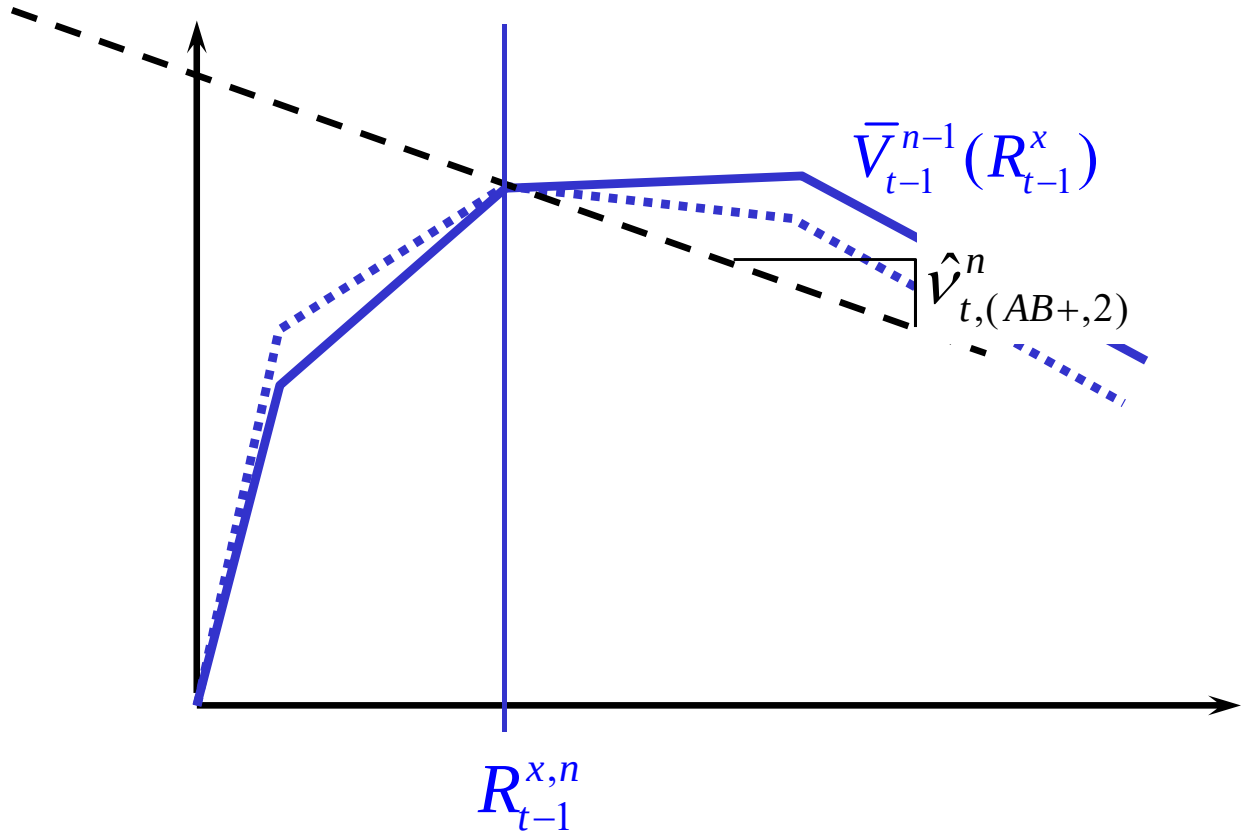
Updating the value function approximation

- Update the value function at $R_{t-1}^{x,n}$



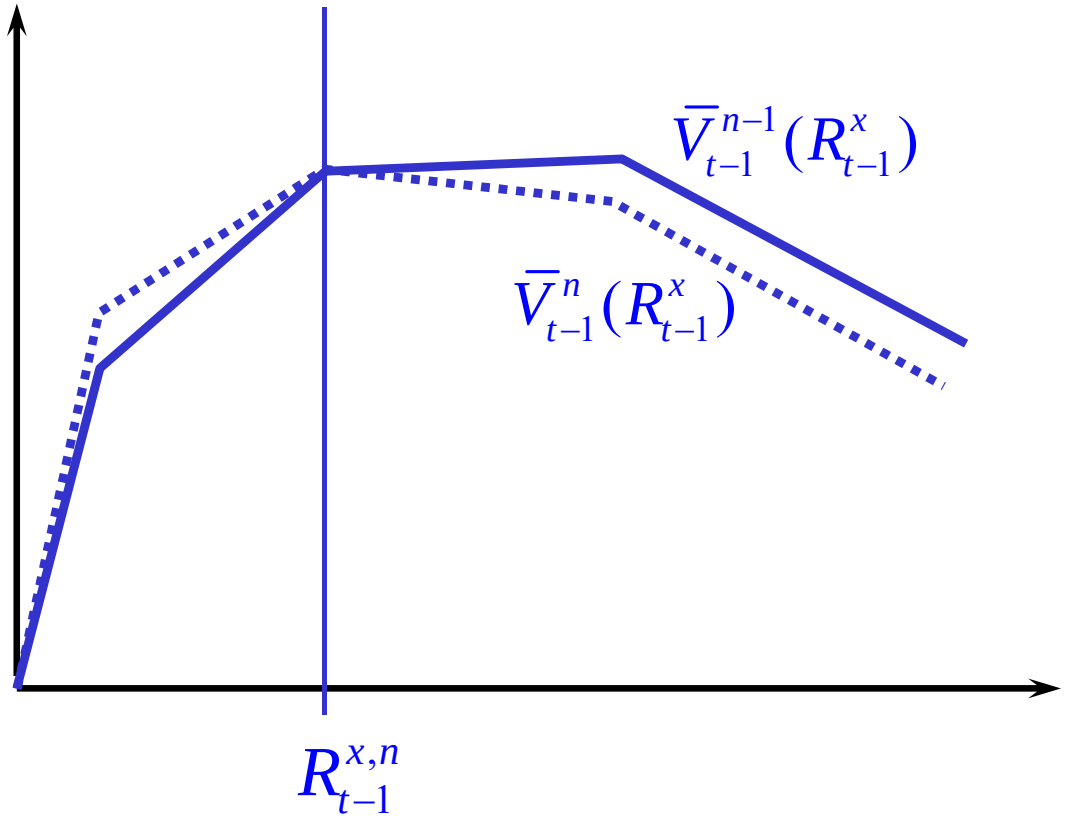
Updating the value function approximation

- Update the value function at $R_{t-1}^{x,n}$



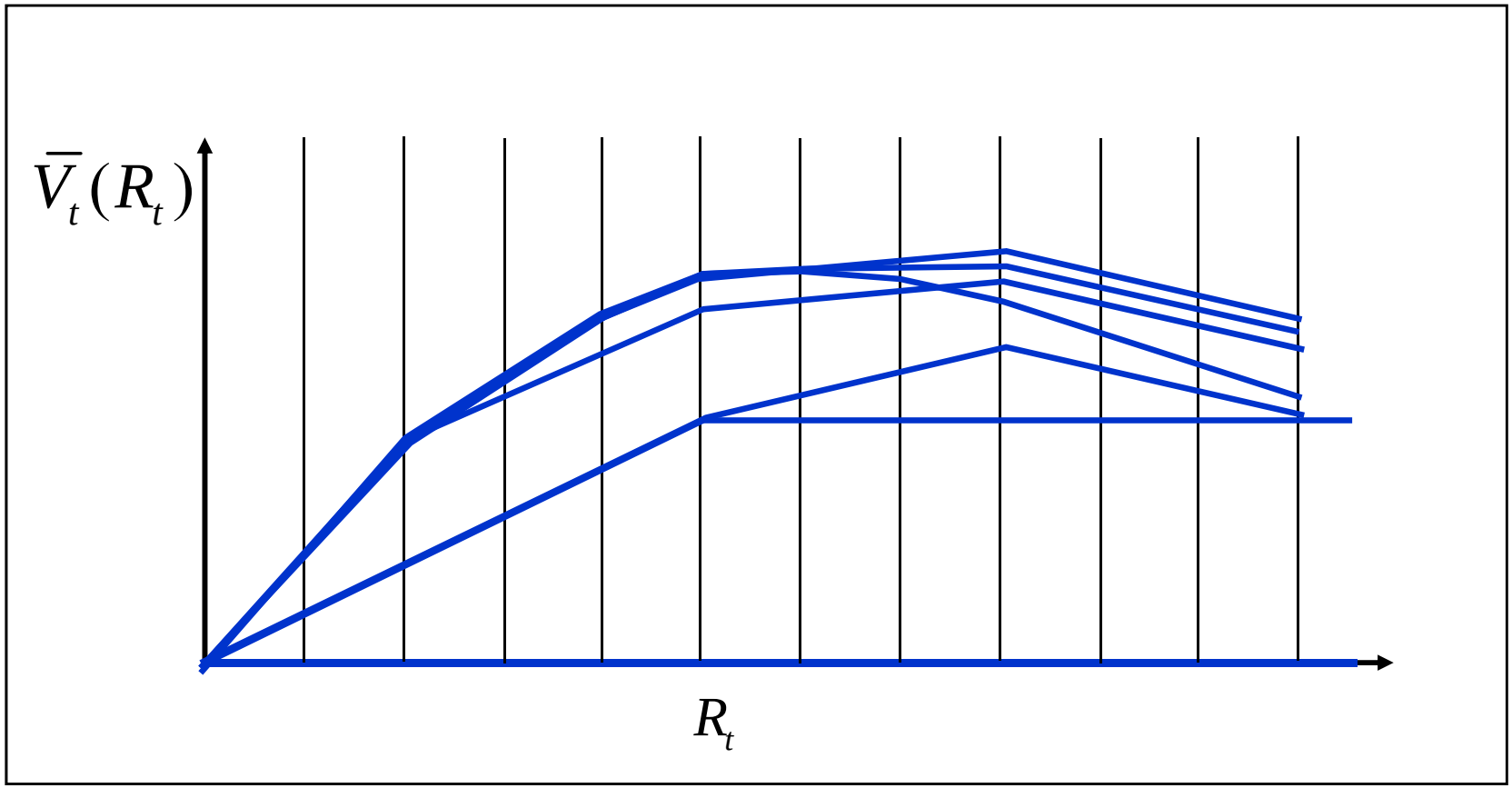
Updating the value function approximation

- Update the value function at $R_{t-1}^{x,n}$



Exploiting concavity

- Derivatives are used to estimate a piecewise linear approximation



Approximate value iteration

Step 1: Start with a pre-decision state S_t^n

Step 2: Solve the deterministic optimization using an approximate value function:

$$\max_x \left(C_t(S_t^n, x_t) + \bar{V}_t^{n-1}(S^{M,x}(S_t^n, x_t)) \right)$$

to obtain x_t^n and dual variables $\left(\hat{v}_{ti}^n\right)$.

Deterministic optimization

Step 3: Update the value function approximation

$$\bar{V}_{t-1}^n(S_{t-1}^{x,n}) = (1-\alpha_{n-1})\bar{V}_{t-1}^{n-1}(S_{t-1}^{x,n}) + \alpha_{n-1}\hat{v}_t^n$$

Recursive statistics

Step 4: Obtain Monte Carlo sample of $W_t(\omega^n)$ and compute the next pre-decision state:

$$S_{t+1}^n = S^M(S_t^n, x_t^n, W_{t+1}(\omega^n))$$

Simulation

Step 5: Return to step 1.

Approximate value iteration

Step 1: Start with a pre-decision state S_t^n

Step 2: Solve the deterministic optimization using an approximate value function:

$$\max_x \left(C_t(S_t^n, x_t) + \bar{V}_t^{n-1}(S^{M,x}(S_t^n, x_t)) \right)$$

to obtain x_t^n and dual variables $\left(\hat{v}_{ti}^n\right)$.

Deterministic optimization

Approximate value iteration

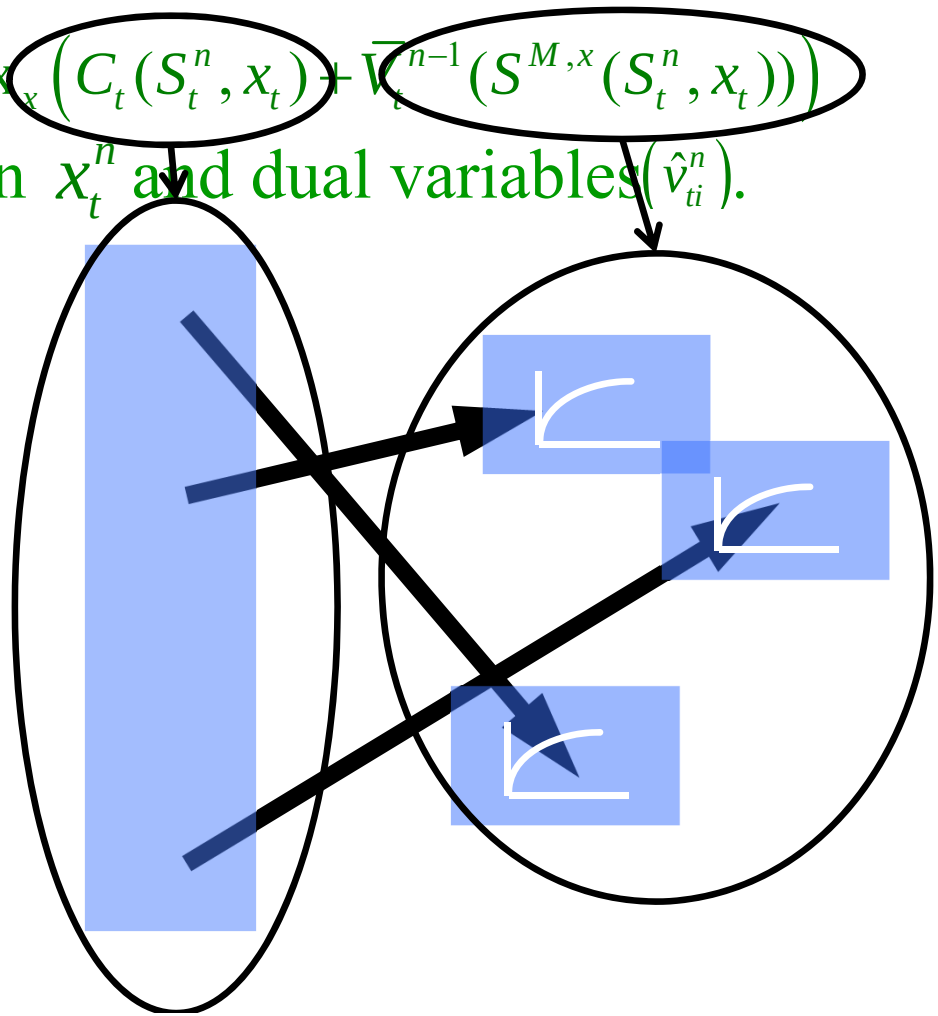
Step 1: Start with a pre-decision state S_t^n

Step 2: Solve the deterministic optimization using an approximate value function:

$$\max_x \left(C_t(S_t^n, x_t) + \overline{V}_t^{n-1}(S^{M,x}(S_t^n, x_t)) \right)$$

to obtain x_t^n and dual variables $\left(\hat{v}_{ti}^n\right)$.

Deterministic optimization



Approximate value iteration

Step 1: Start with a pre-decision state S_t^n

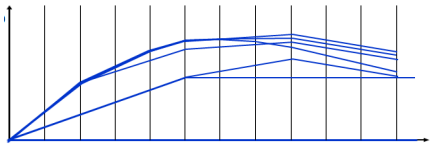
Step 2: Solve the deterministic optimization using an approximate value function:

$$\max_x \left(C_t(S_t^n, x_t) + \bar{V}_t^{n-1}(S^{M,x}(S_t^n, x_t)) \right)$$

to obtain x_t^n and dual variables $\left(\hat{v}_{ti}^n\right)$.

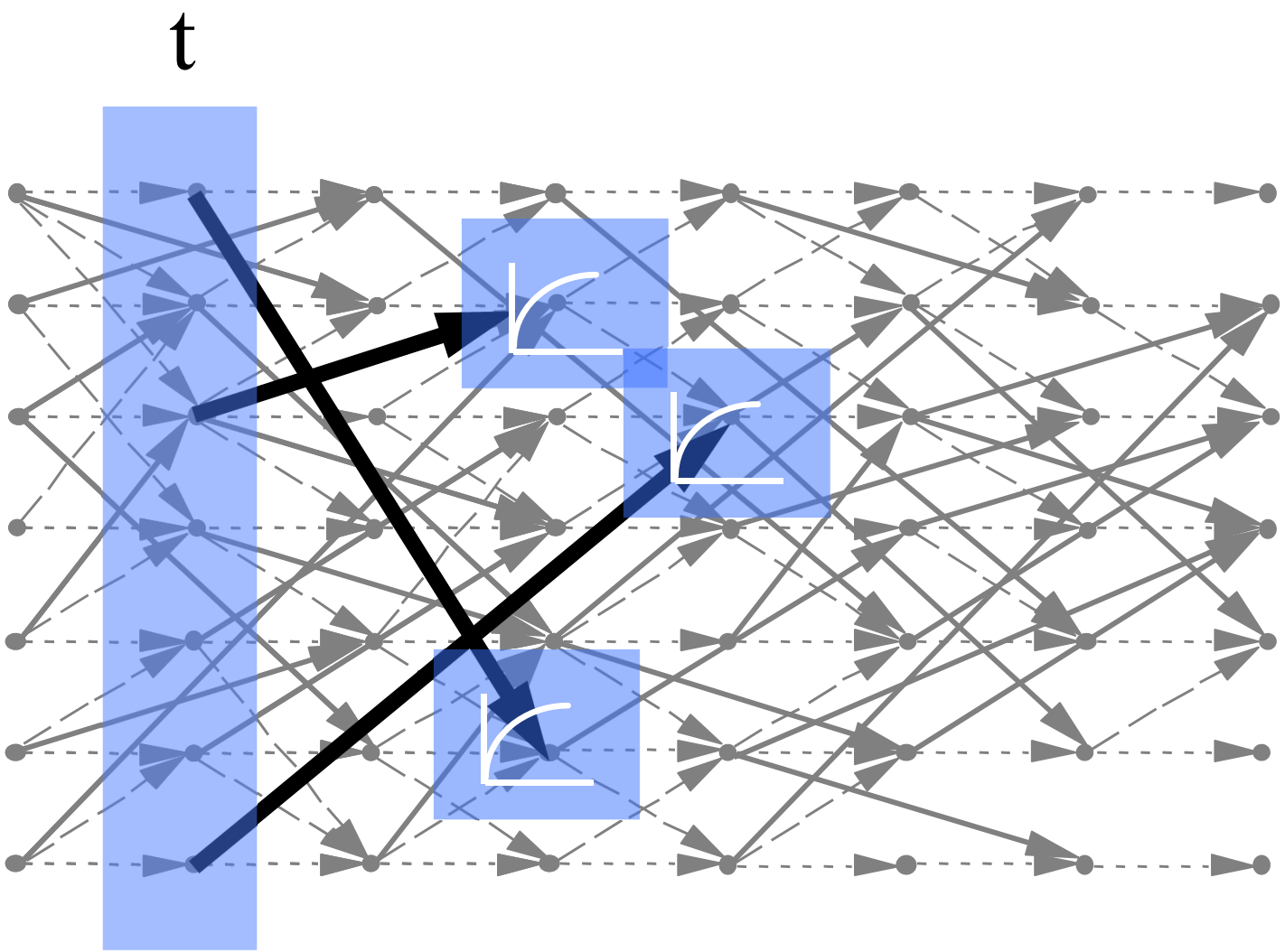
Deterministic optimization

Step 3: Update the value function approximation

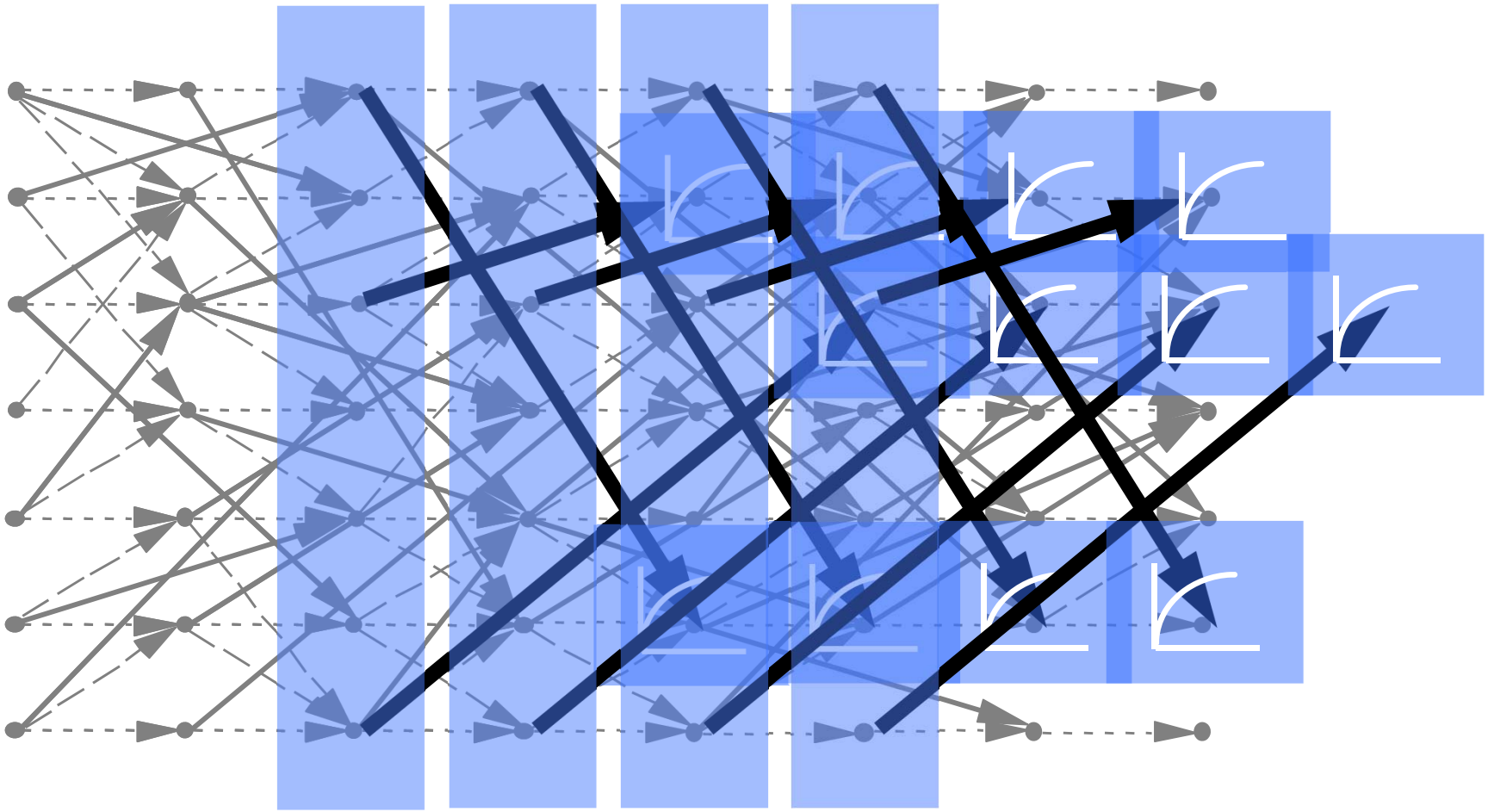


Recursive statistics

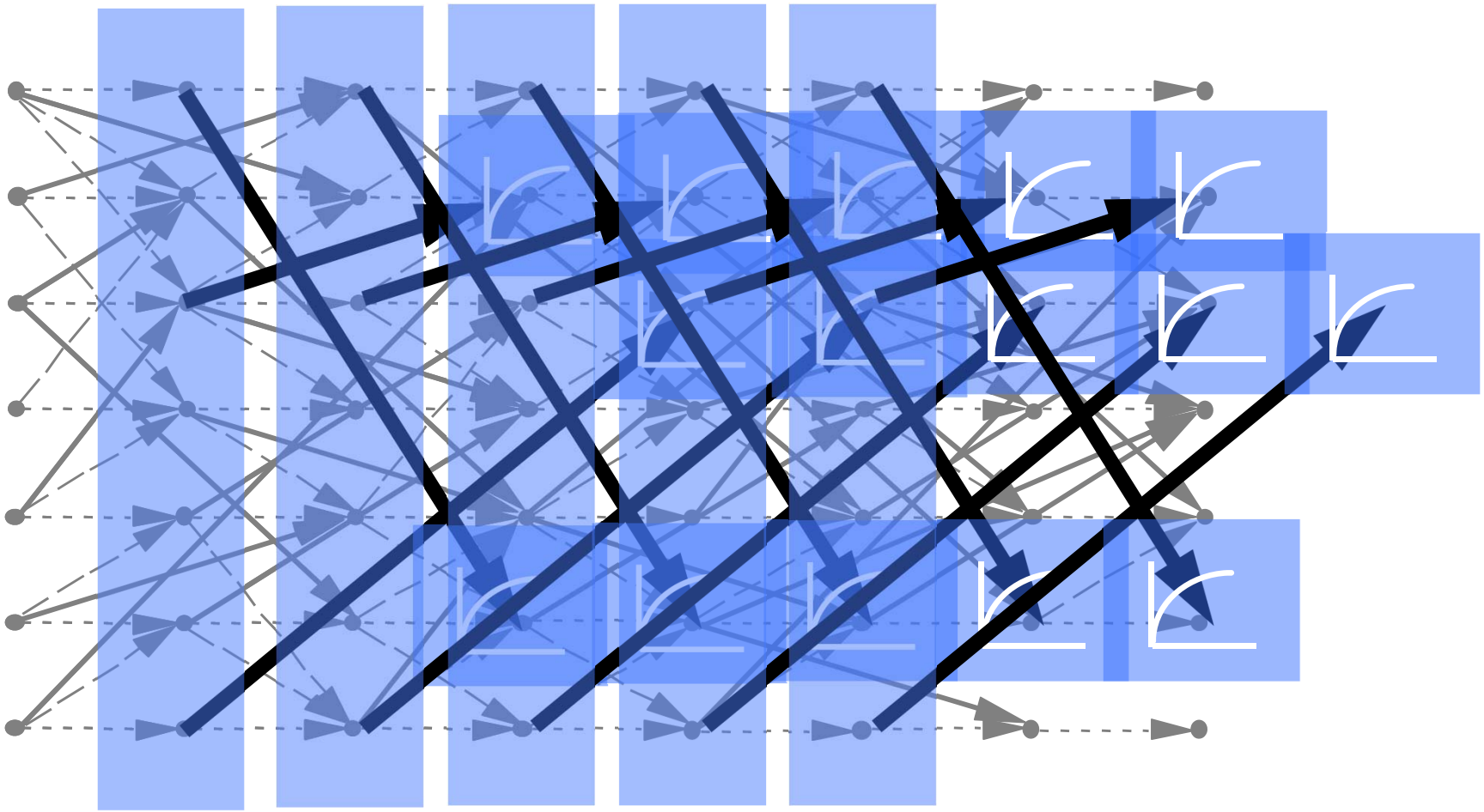
Iterative learning



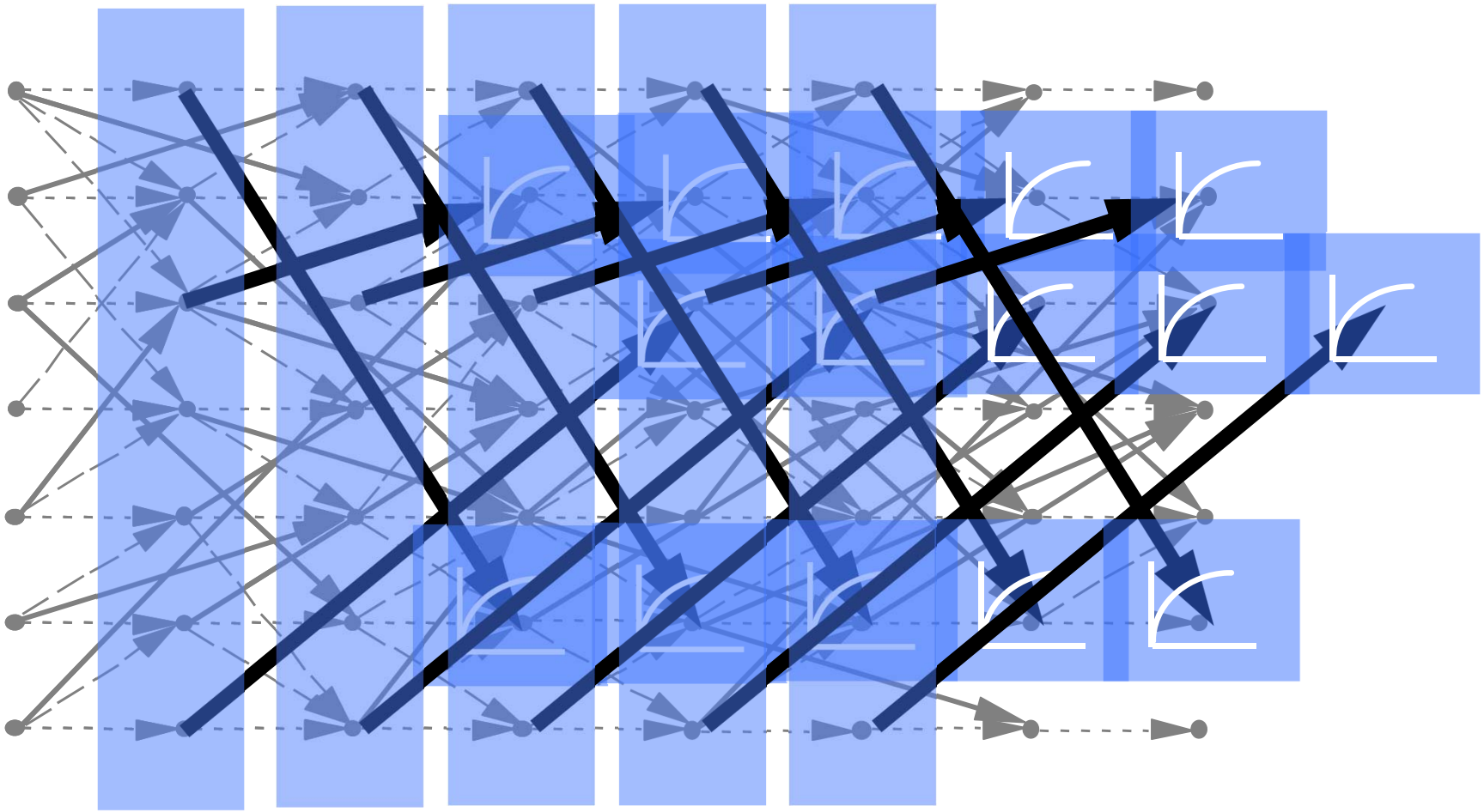
Iterative learning



Iterative learning

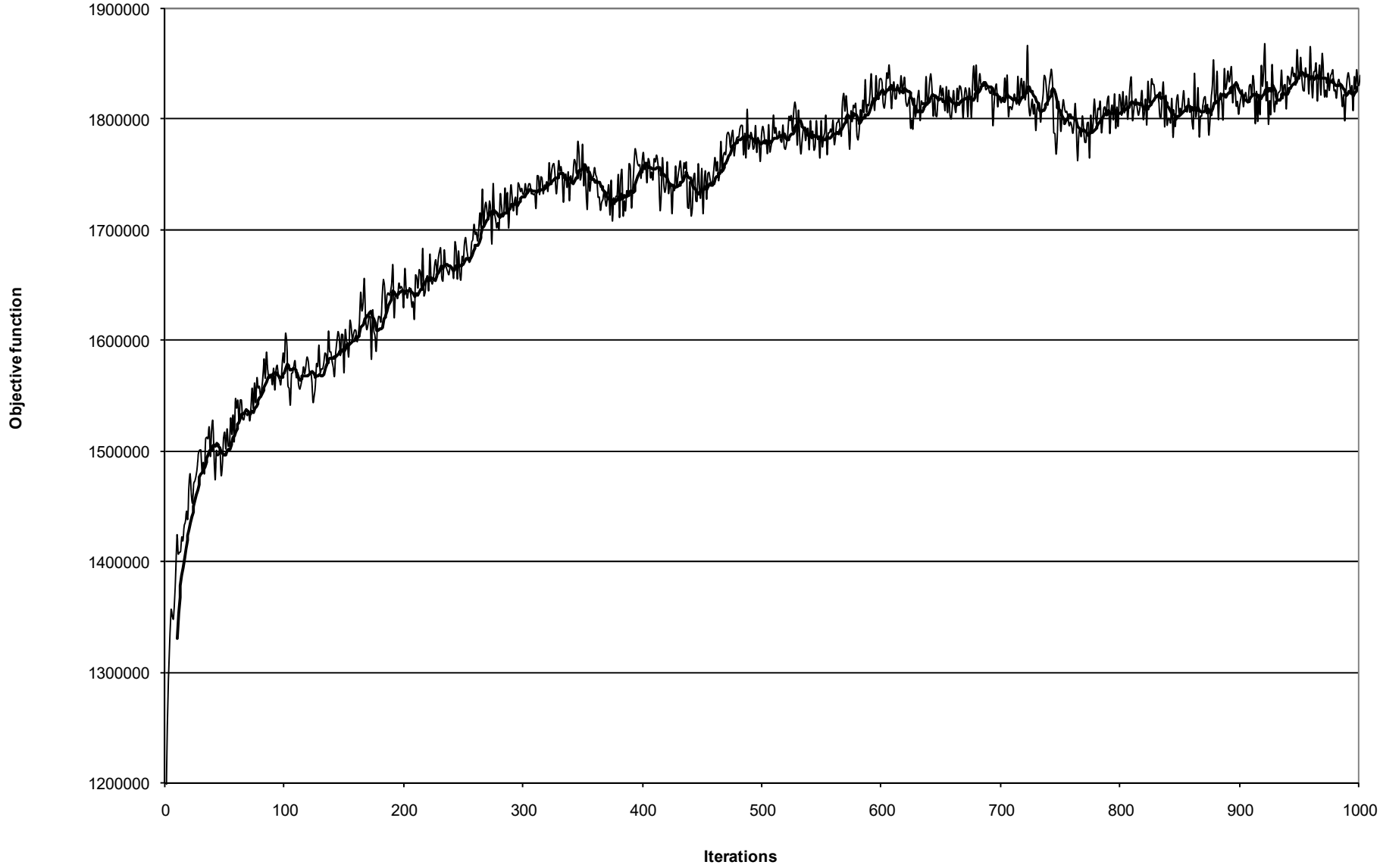


Iterative learning



Approximate dynamic programming

- … a typical performance graph.



A Comparison of Approximate Dynamic Programming Techniques on Benchmark Energy Storage Problems: Does Anything Work?

Daniel R. Jiang, Thuy V. Pham, Warren B. Powell, Daniel F. Salas, and Warren R. Scott

Approximate value functions can work very well, but you need structure to guide the learning process. ADP needs benchmarks and careful tuning.

***Abstract*— like solar ar problem of is becoming problems are when the state space becomes large, traditional (exact) techniques such as backward induction, policy iteration, or value iteration quickly become computationally intractable. Approximate dynamic programming (ADP) thus becomes a natural solution technique for solving these problems to near-optimality using significantly fewer computational resources. In this paper, we compare the performance of the following: various approximation architectures used approximate policy iteration (API), approximate value iteration (AVI) with structured lookup table, and direct policy search on an energy storage problem, for which optimal benchmarks exist.**

lookup table
he additional
emely effec-
ore advanced

statistical estimation methods.

This paper reports on the performance of a variety of approximation methods that have been developed in the approximate dynamic programming community, tested using a series of optimal benchmark problems drawn from a relatively simple energy storage application. These suggest that methods based on Bellman error minimization, using both approximate value iteration and approximate policy iteration, work surprisingly poorly if we use approximation methods drawn from machine learning. Pure table lookup also works poorly. By contrast,

"I think you give a too rosy a picture of ADP…."
Andy Barto, in comments on a paper (2009)

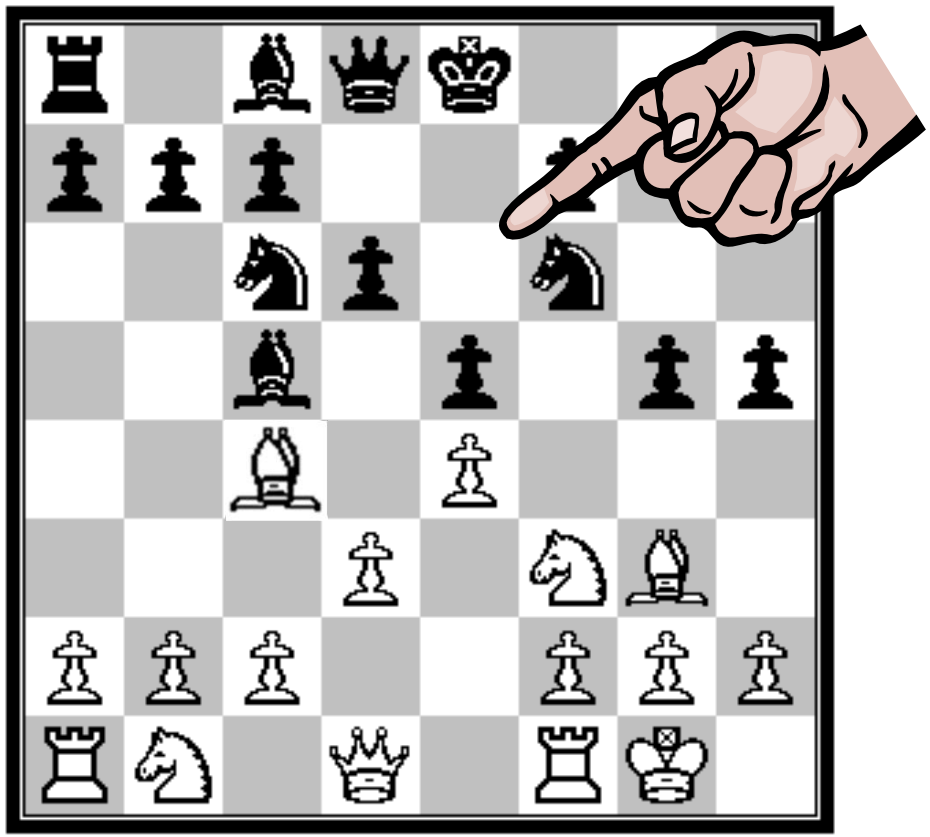
"Is the RL glass half full, or half empty?"
Rich Sutton, NIPS workshop, (2014)

Outline

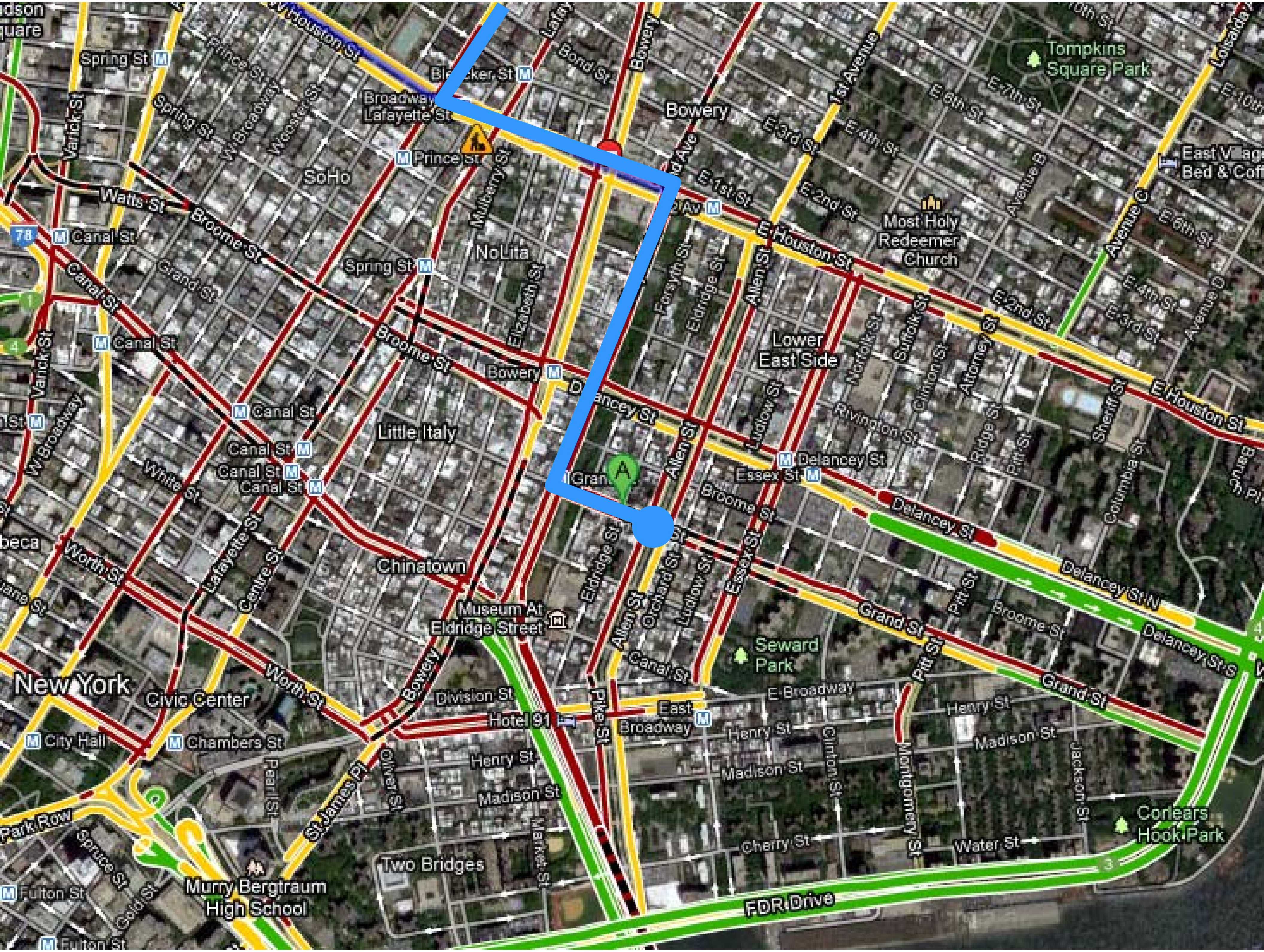
- The four classes of policies
  - » Policy function approximations (PFAs)
  - » Cost function approximations (CFAs)
  - » Value function approximations (VFAs)
  - » Direct lookahead policies (DLAs)
  - » A hybrid lookahead/CFA

Lookahead policies

- Planning your next chess move:



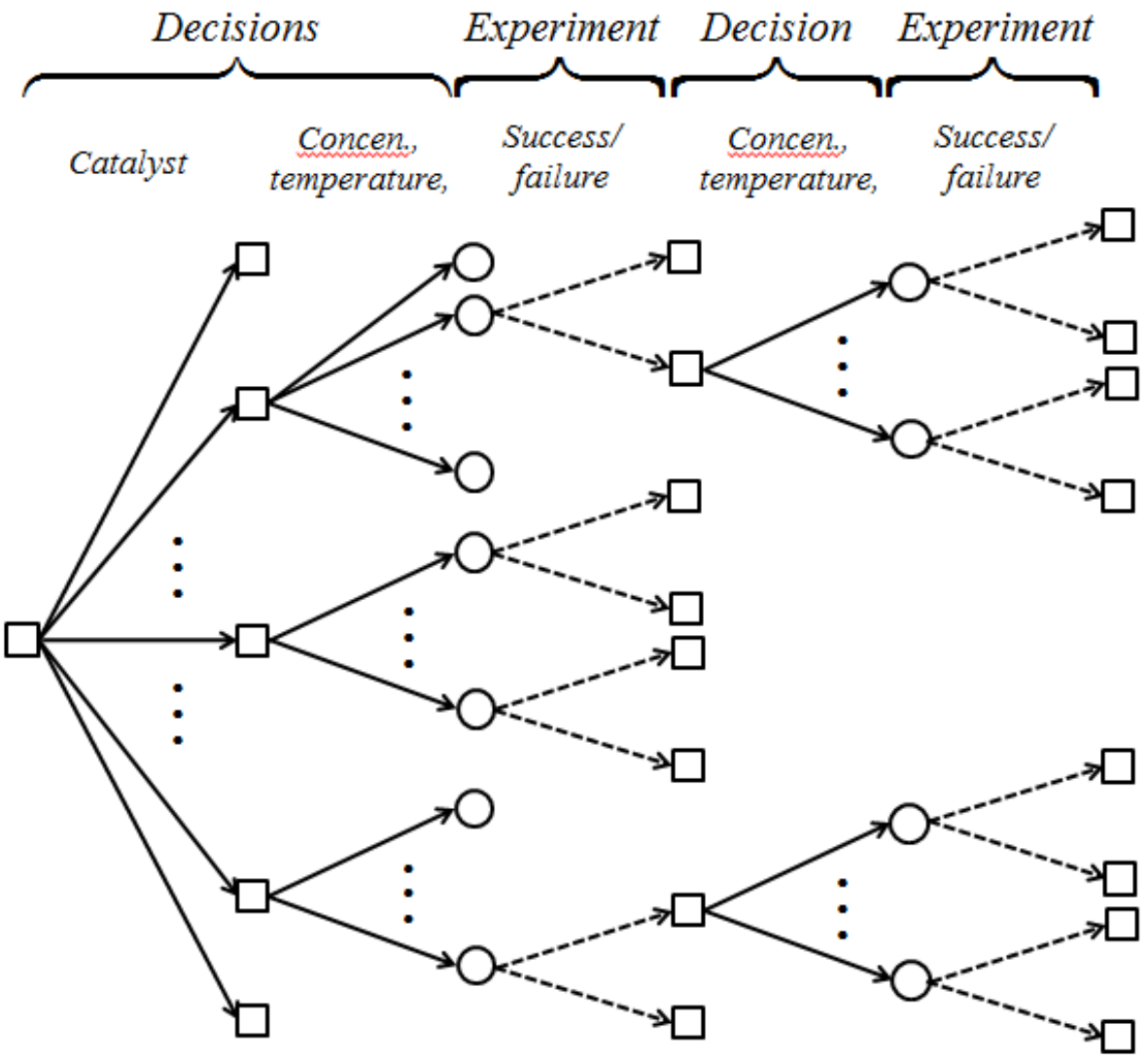
» You put your finger on the piece while you think about moves into the future. This is a lookahead policy, illustrated for a problem with discrete actions.



Tompkins Square Park
Bowery
SoHo
NoLita
Little Italy
Lower East Side
Chinatown
Museum At Eldridge Street
Seward Park
New York
Civic Center
Two Bridges
Murry Bergtraum High School
Corlears Hook Park
Most Holy Redeemer Church
FDR Drive
E Houston St
Delancey St
Grand St
Canal St
Broome St
E Broadway
Hotel 91

Lookahead policies

- Decision trees:



Lookahead policies

- Modeling lookahead policies
  - » Lookahead policies solve a *lookahead model*, which is an approximation of the future.
  - » It is important to understand the difference between the:
    - Base model – this is the model we are trying to solve by finding the best policy. This is usually some form of simulator.
    - The lookahead model, which is our approximation of the future to help us make better decisions now.
  - » The base model is typically a simulator, or it might be the real world.

Lookahead policies

- Lookahead models use five classes of approximations:
  - » Horizon truncation – Replacing a longer horizon problem with a shorter horizon
  - » Stage aggregation – Replacing multistage problems with two-stage approximation.
  - » Outcome aggregation/sampling – Simplifying the exogenous information process
  - » Discretization – Of time, states and decisions
  - » Dimensionality reduction – We may ignore some variables (such as forecasts) in the lookahead model that we capture in the base model (these become *latent* variables in the lookahead model).

Lookahead policies

- Lookahead policies are the trickiest to model:
  - » We create "tilde variables" for the lookahead model:

    $\tilde{S}_{t,t'}$ = Approximated state variable (e.g coarse discretization)

    $\tilde{x}_{t,t'}$ = Decision we plan on implementing at time t' when we are planning at time t, $t' = t, t+1, ..., t+H$

    $\tilde{x}_t = \left(\tilde{x}_{t,t}, \tilde{x}_{t,t+1}, ..., \tilde{x}_{t,t+H} \right)$

    $\tilde{W}_{t,t'}$ = Approximation of information process

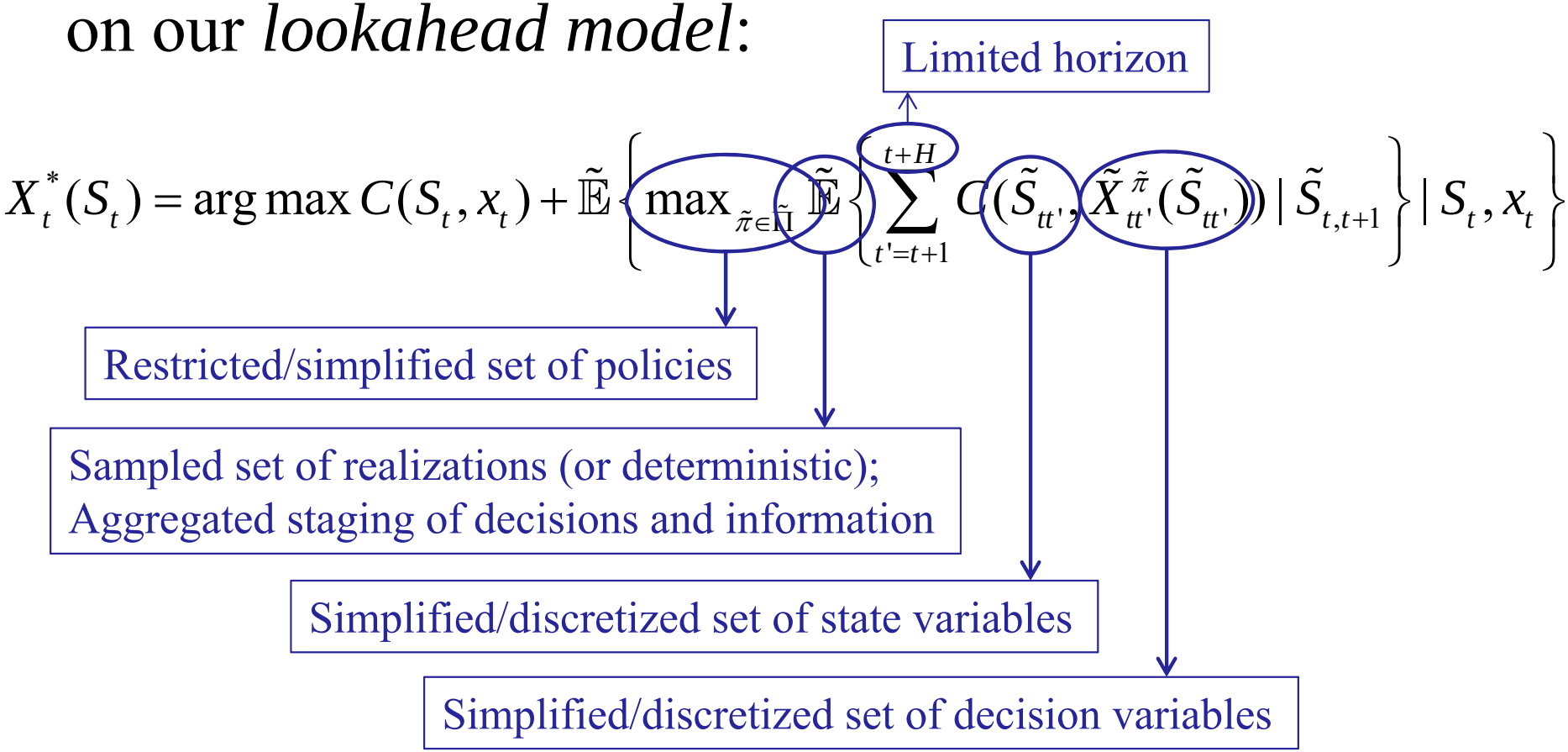
    $\tilde{c}_{t,t'}$ = Forecast of costs at time t' made at time t

    $\tilde{b}_{t,t'}$ = Forecast of right hand sides for time t' made at time t

  - » All variables are indexed by t (when the lookahead model is being generated) and t' (the time within the lookahead model).

Lookahead policies

- We can use this notation to create a policy based on our *lookahead model*:



» Simplest lookahead is deterministic.

Lookahead policies

- Deterministic lookahead

$$X_t^{LA-D}(S_t) = \arg\max_{\tilde{x}_{tt}, \tilde{x}_{t,t+1}, \ldots, \tilde{x}_{t,t+T}} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}, \tilde{x}_{tt'})$$

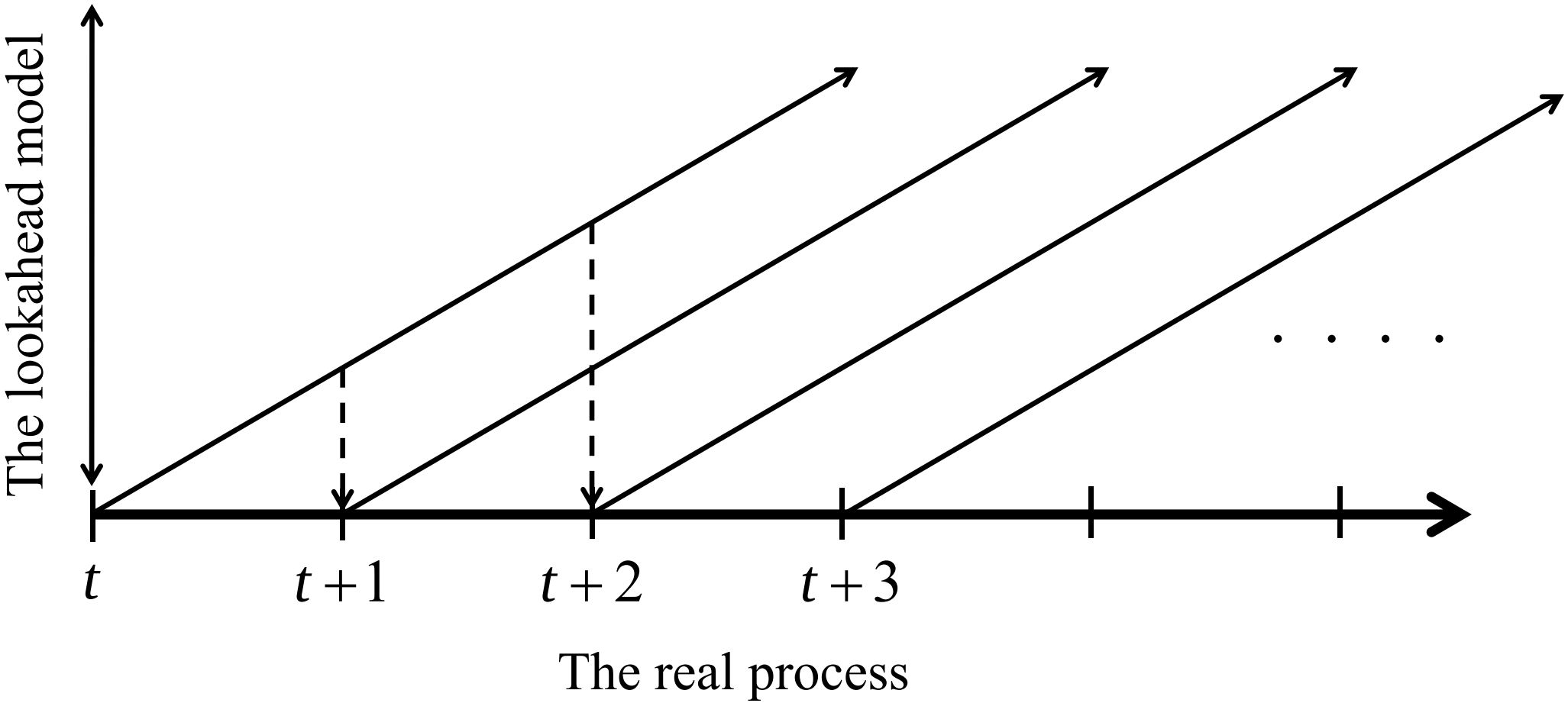
- Stochastic lookahead (with two-stage approximation)

$$X_t^{LA-S}(S_t) = \arg\max_{\tilde{x}_{tt}, \tilde{x}_{t,t+1}, \ldots, \tilde{x}_{t,t+T}} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \sum_{\tilde{\omega} \in \tilde{\Omega}_t} p(\tilde{\omega}) \sum_{t'=t+1}^{T} C(\tilde{S}_{tt'}(\tilde{\omega}), \tilde{x}_{tt'}(\tilde{\omega}))$$

Scenario trees

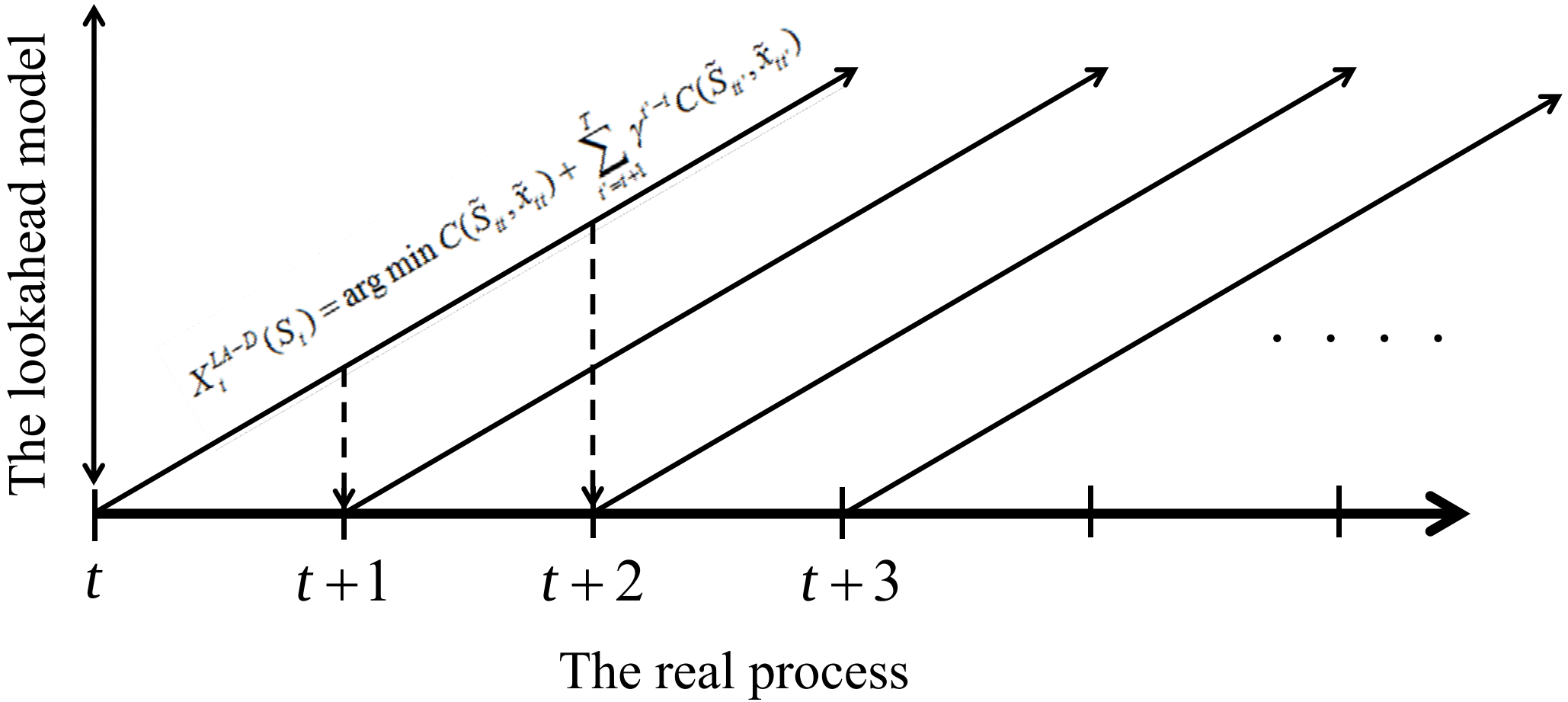
Lookahead policies

- Lookahead policies peek into the future
  - » Optimize over deterministic lookahead model



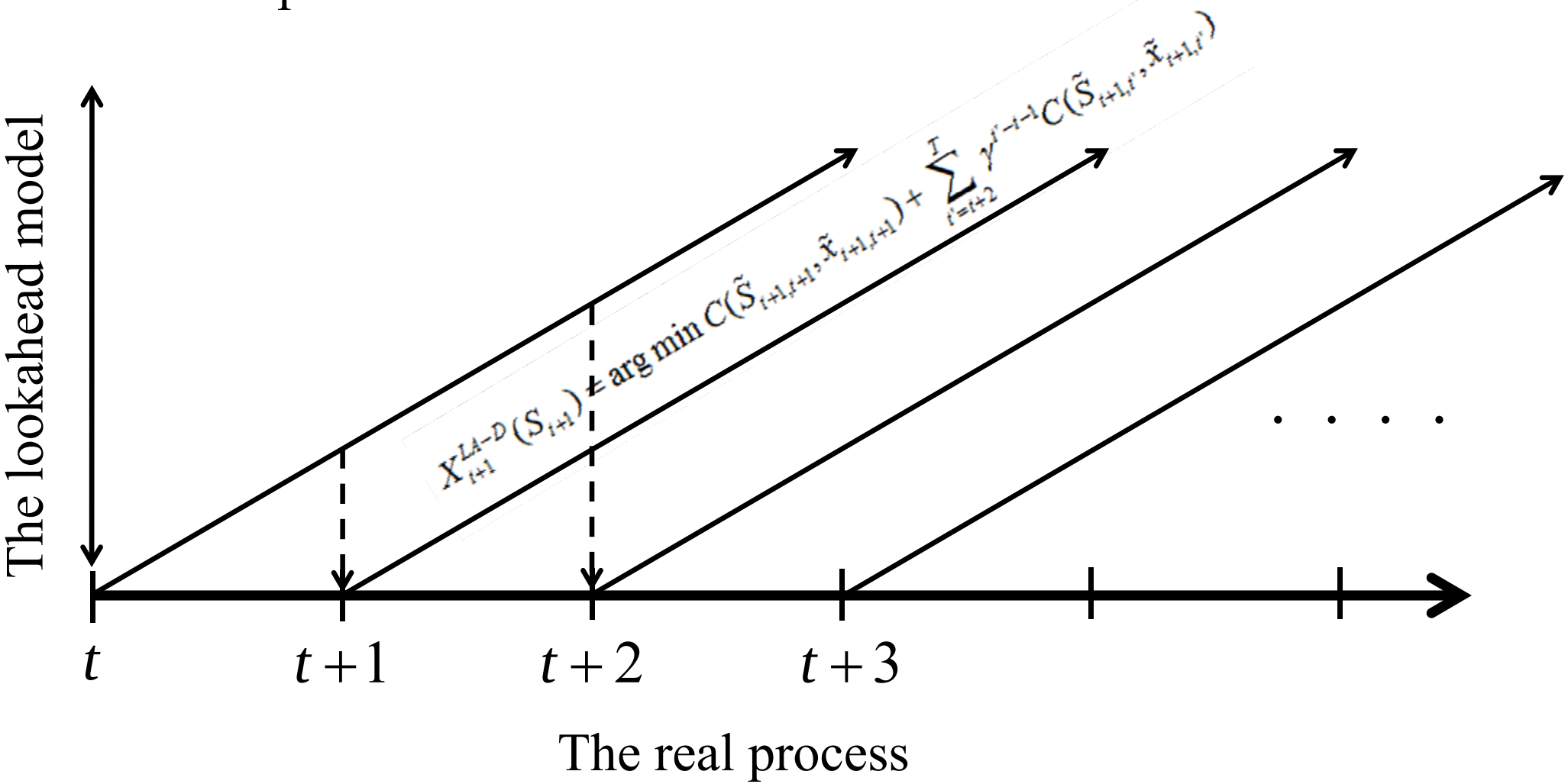
Lookahead policies

- Lookahead policies peek into the future
  - » Optimize over deterministic lookahead model



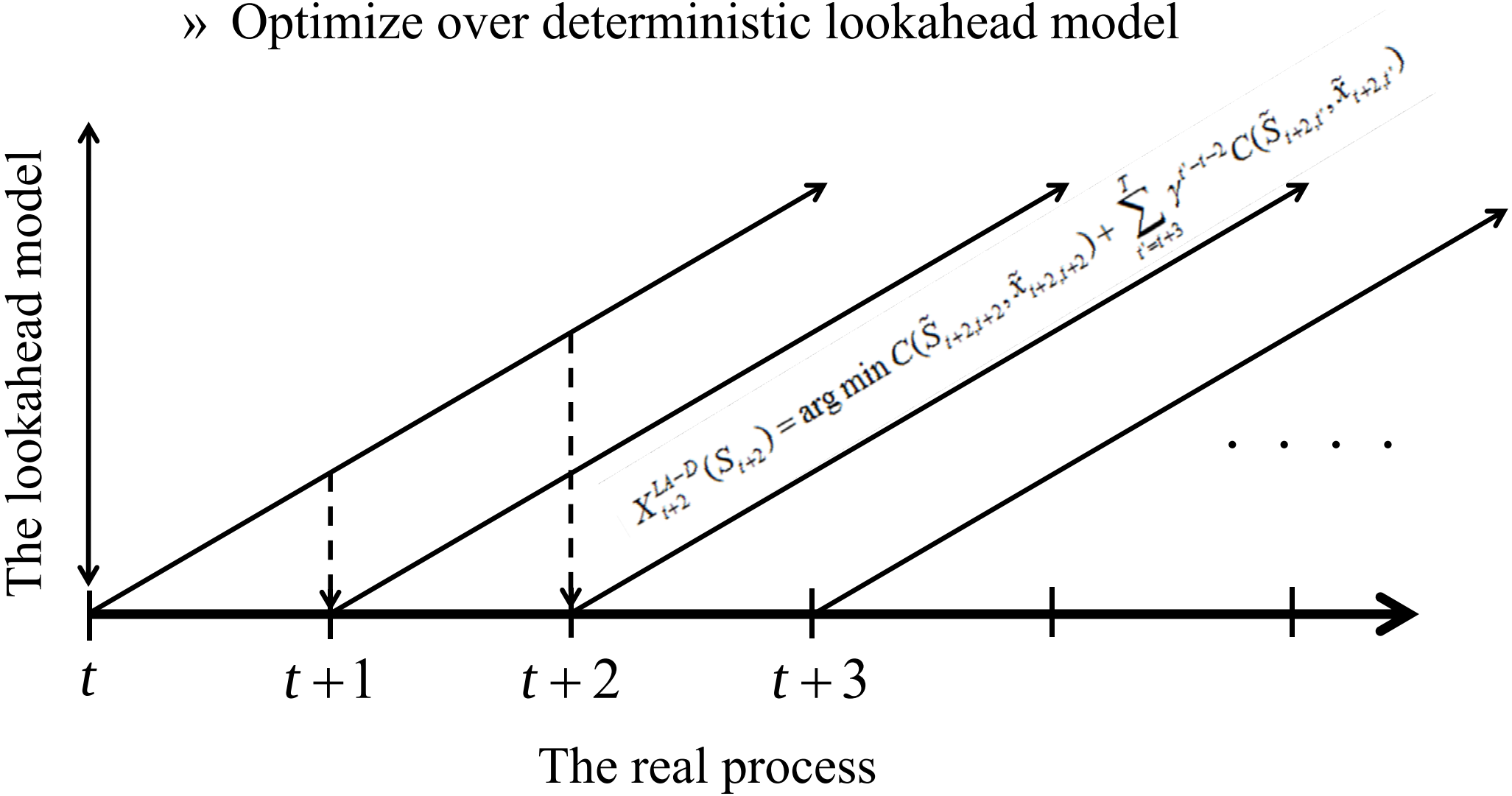
Lookahead policies

- Lookahead policies peek into the future
  - » Optimize over deterministic lookahead model



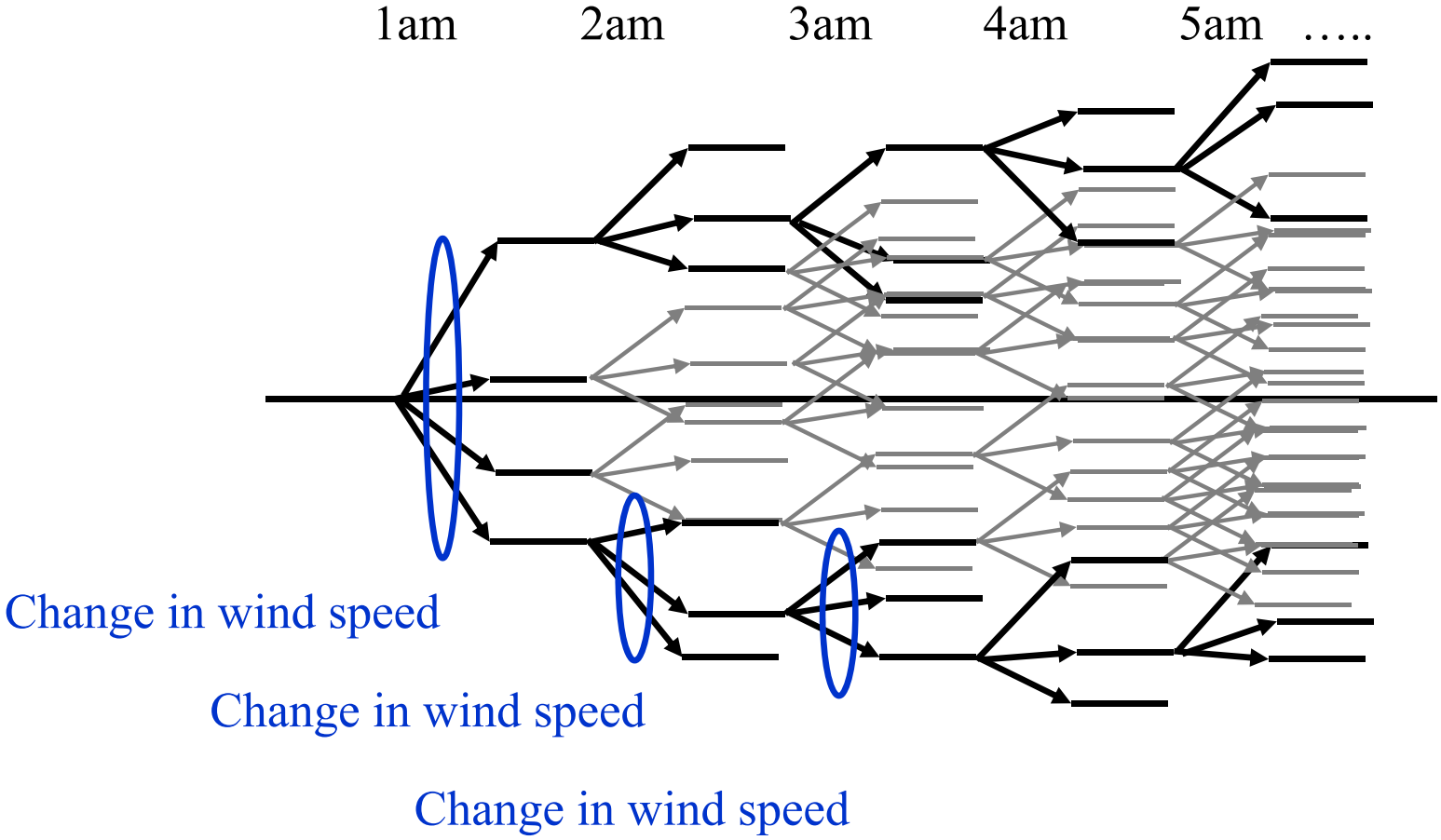
Lookahead policies

- Lookahead policies peek into the future
  - » Optimize over deterministic lookahead model



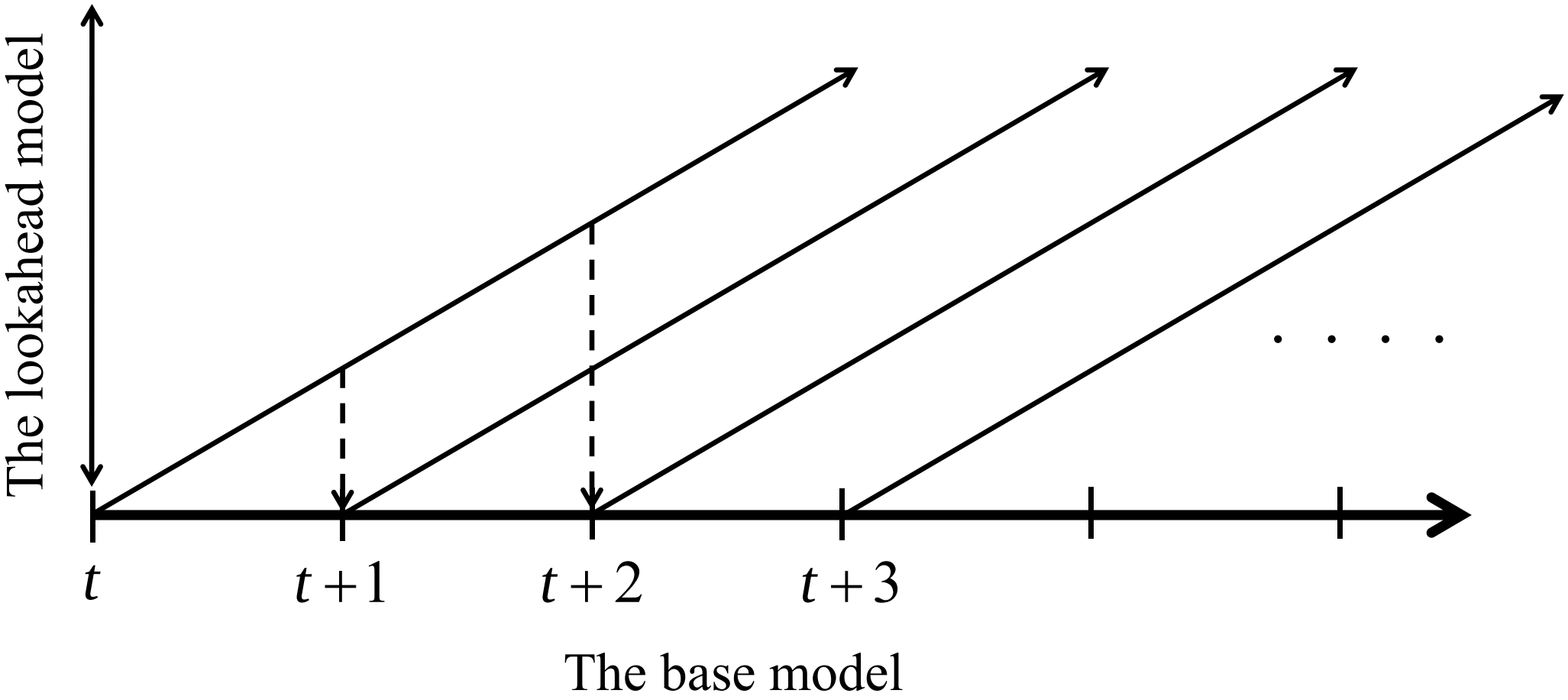
Lookahead policies

- Stochastic lookahead
  - » Here, we approximate the information model by using a Monte Carlo sample to create a scenario tree:



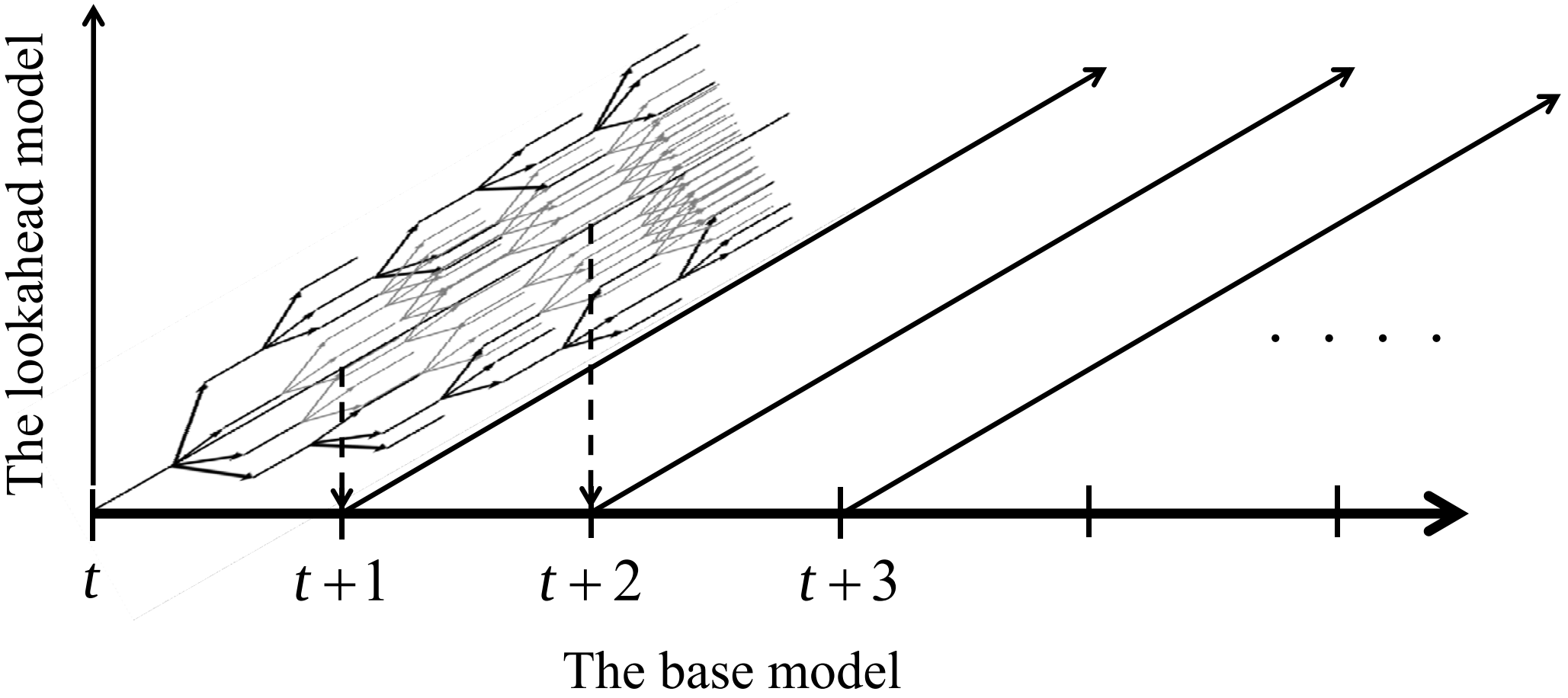
Lookahead policies

- We can then simulate this *lookahead policy* over time:



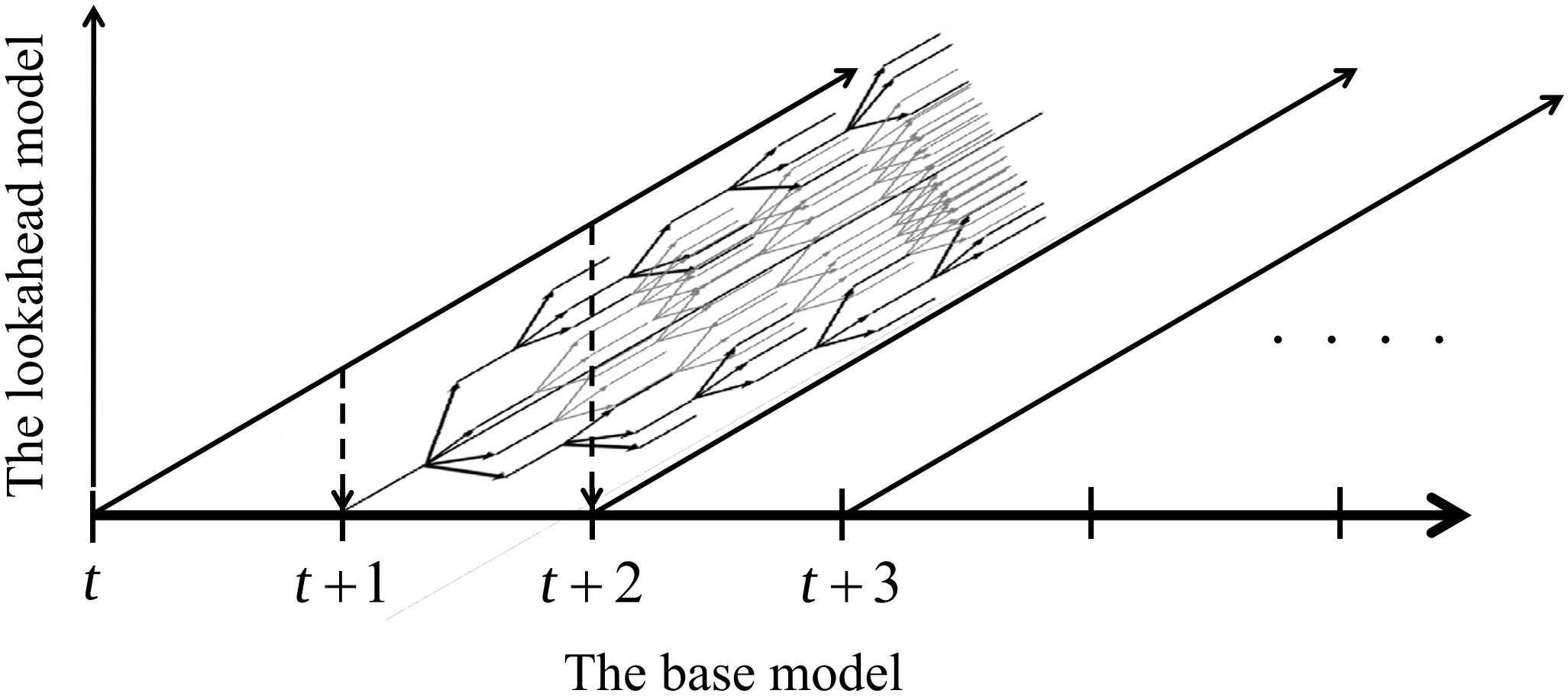
Lookahead policies

- We can then simulate this *lookahead policy* over time:



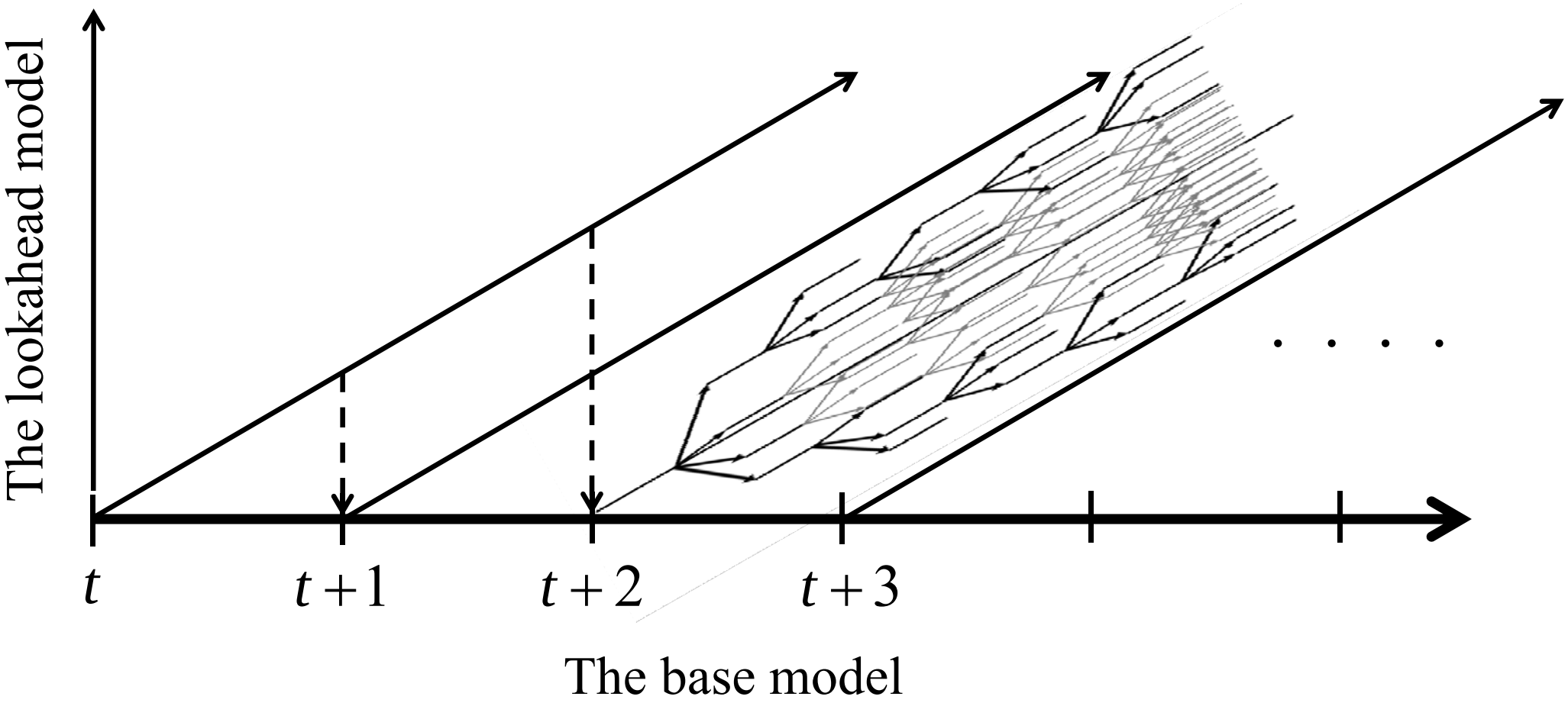
Lookahead policies

- We can then simulate this *lookahead policy* over time:



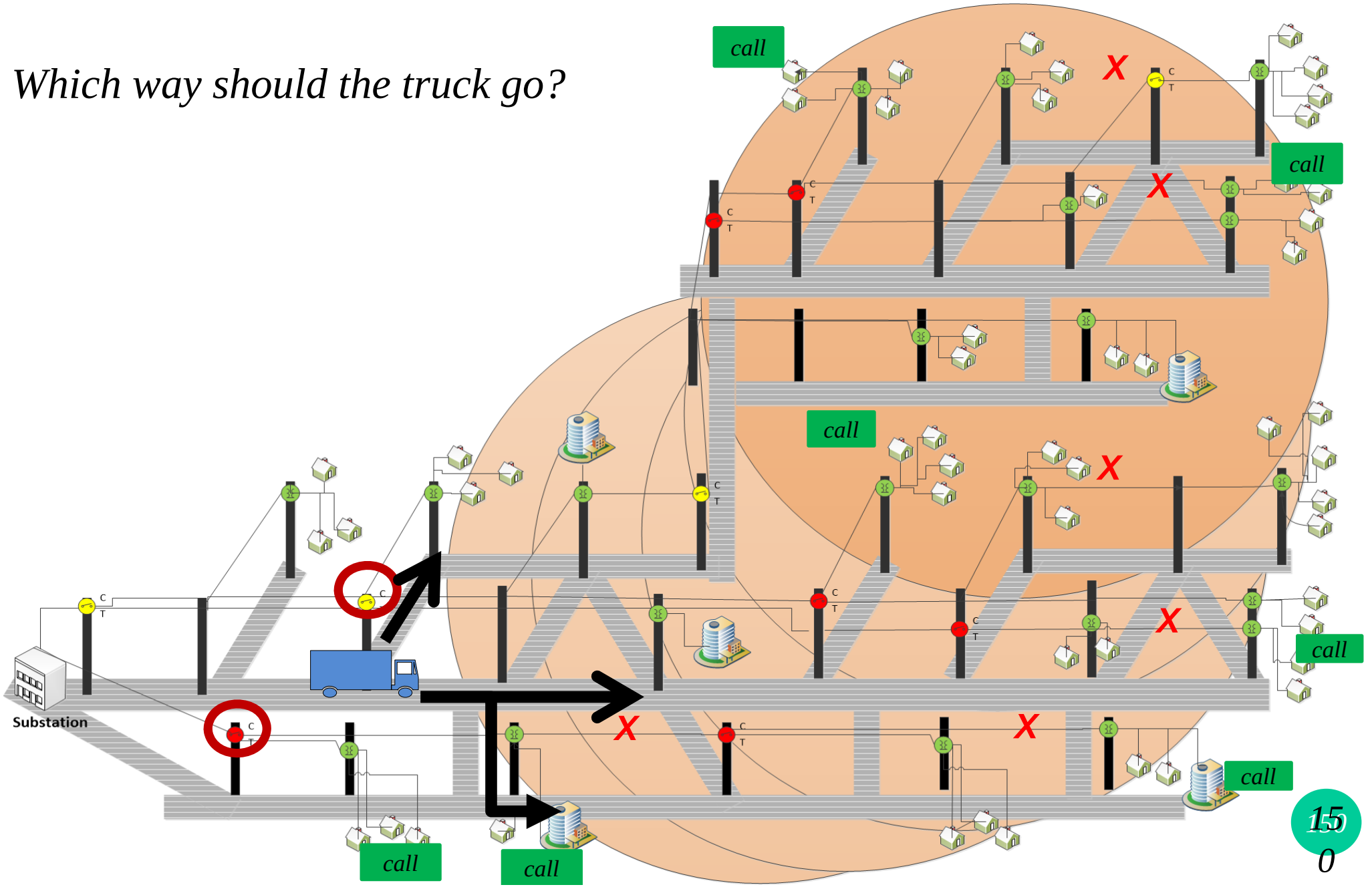
Lookahead policies

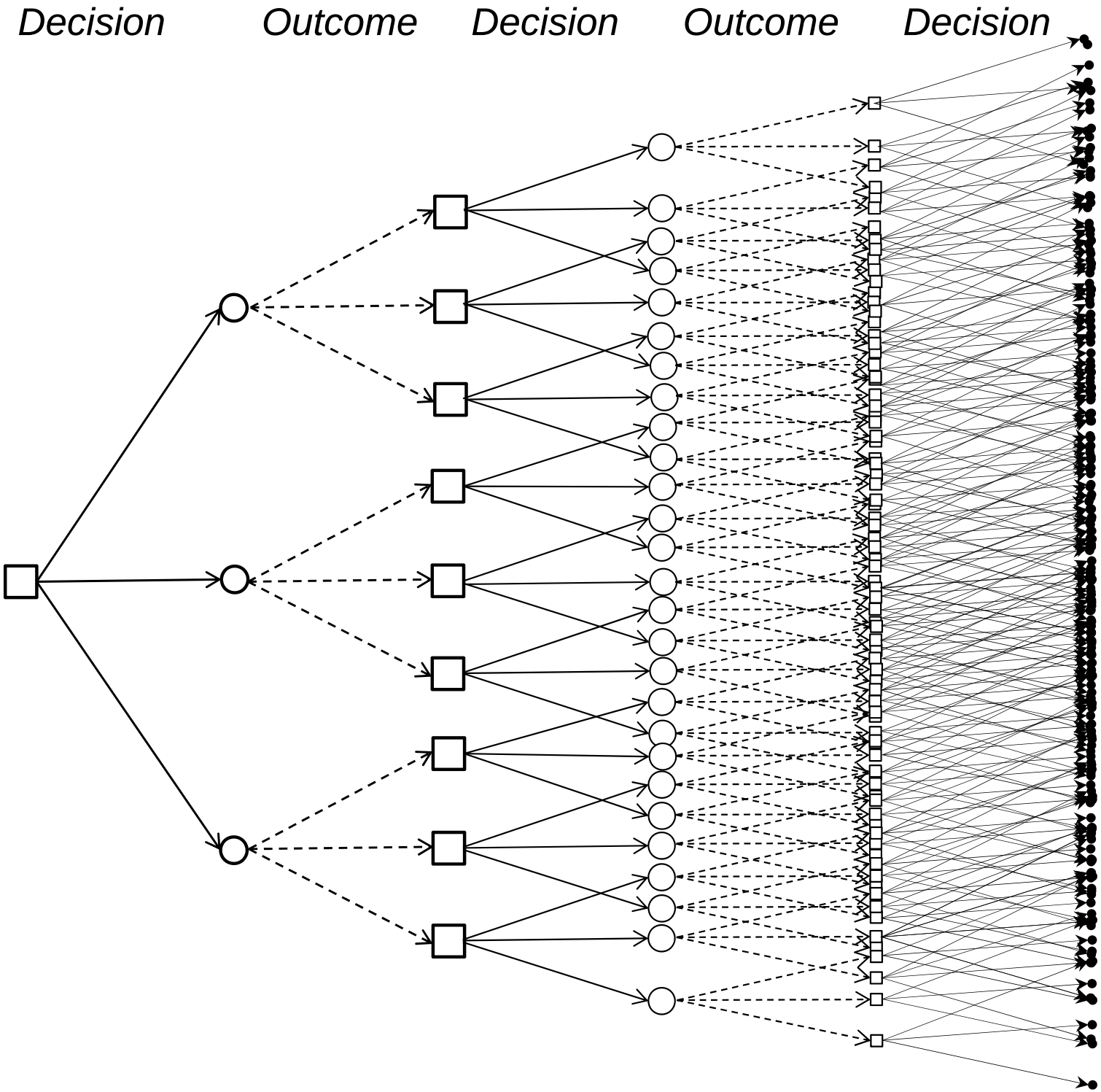
- We can then simulate this *lookahead policy* over time:



Learning damaged networks

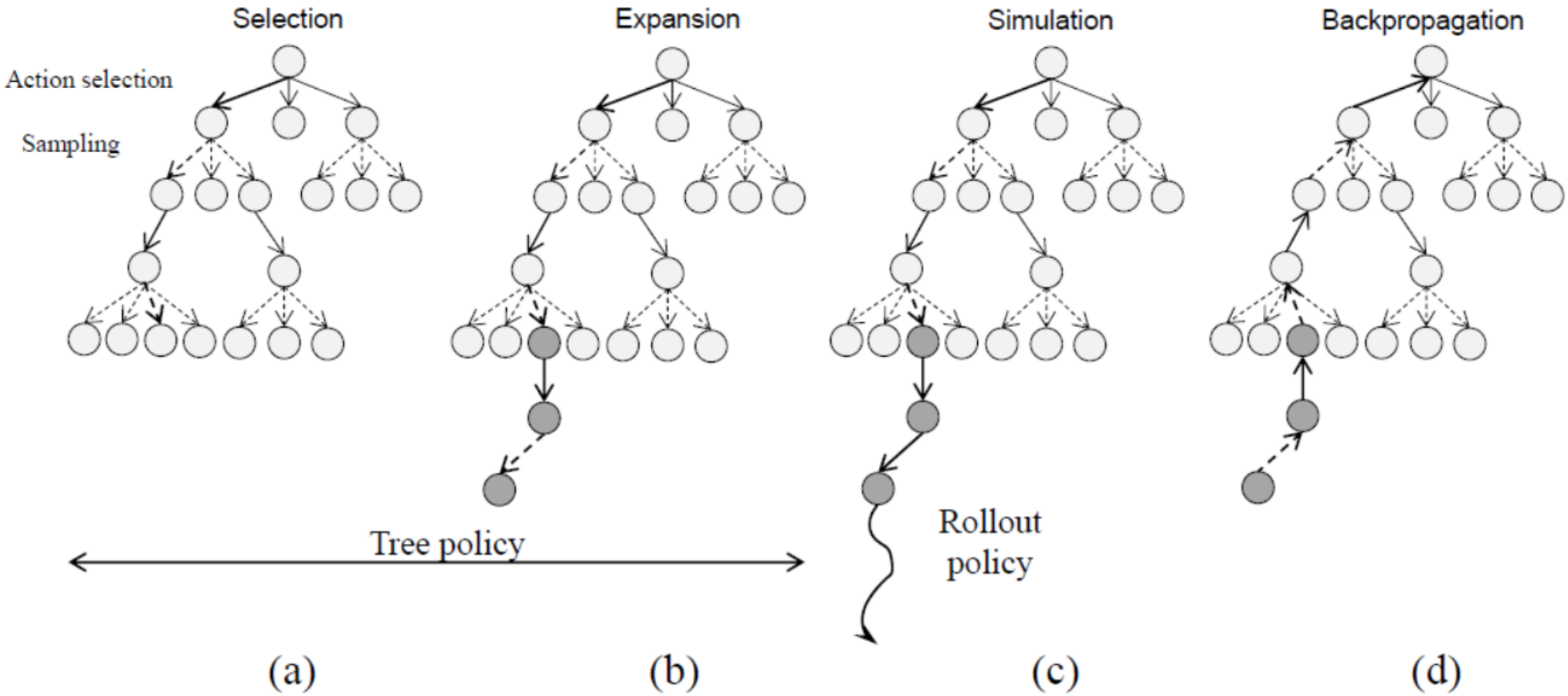
Which way should the truck go?



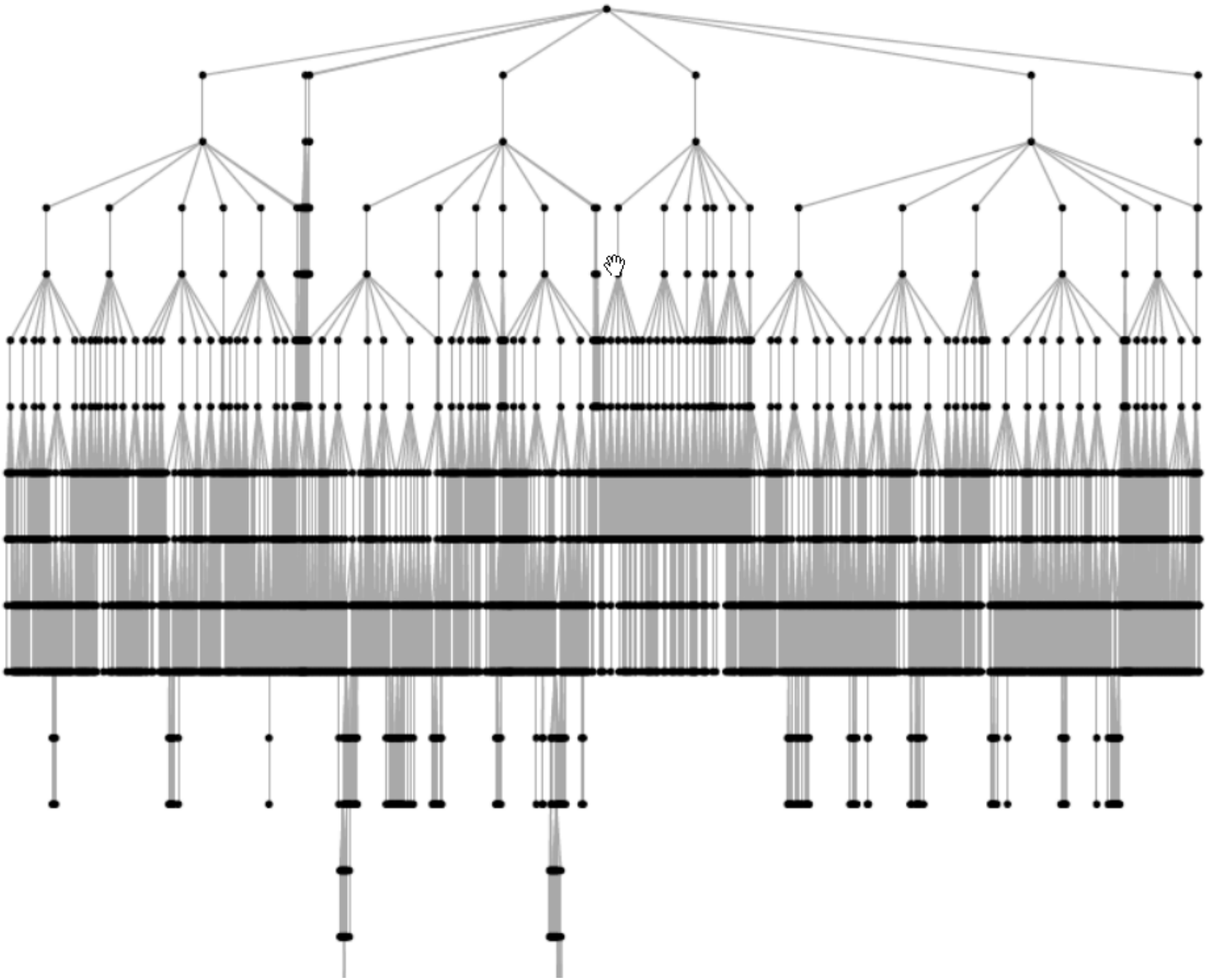

Decision
Outcome
Decision
Outcome
Decision

Lookahead policies

- Monte Carlo tree search:



C. Browne, E. Powley, D. Whitehouse, S. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis and S. Colton, "A survey of Monte Carlo tree search methods," IEEE Transactions on Computational Intelligence and AI in Games, vol. 4, no. 1, pp. 1–49, March 2012.



Lookahead policies

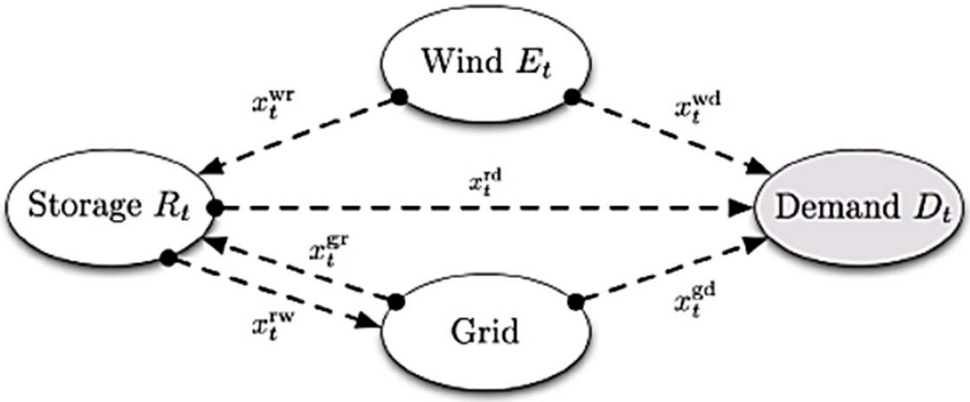
- Notes:
  - » Solving stochastic lookahead policies can be hard!
  - » … but this is still just a lookahead policy which is a class of rolling horizon heuristic.
  - » Even if solving the lookahead model is hard, an optimal solution of a lookahead model (even a stochastic one) is (with rare exceptions) not an optimal policy.

Outline

- The four classes of policies
  - » Policy function approximations (PFAs)
  - » Cost function approximations (CFAs)
  - » Value function approximations (VFAs)
  - » Direct lookahead policies (DLAs)
  - » A hybrid lookahead/CFA

Parametric cost function approximation

- An energy storage problem:



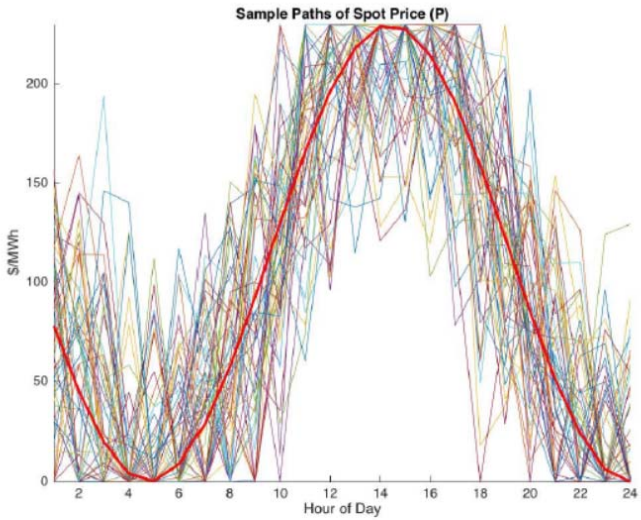


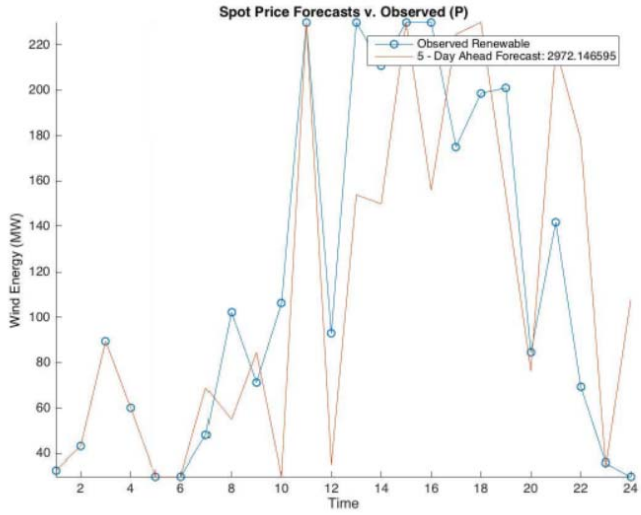
Figure: Sample paths of spot prices (P_t)

The state of the system can be represented by the following five dimensional vector,

$$S_t = (R_t, E_t, P_t, D_t, G_t)$$

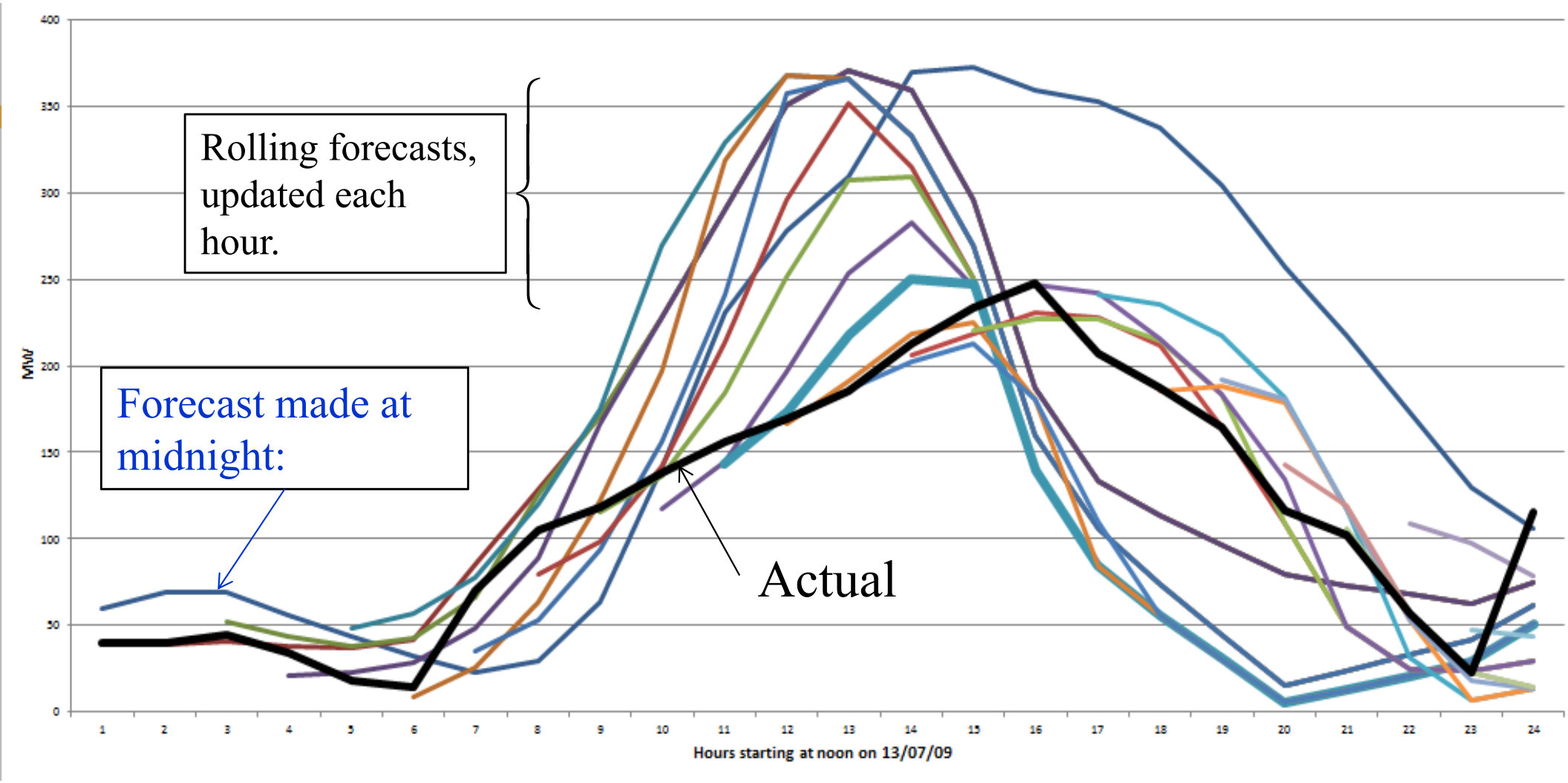
where

- $R_t \in [0, R_{\max}]$ is the level of energy in storage at time t
- E_t is the amount of energy available from wind
- P_t is the spot price of electricity
- D_t is the power demand
- G_t is the energy available from the grid



Parametric cost function approximation

- Forecasts evolve over time as new information arrives:



Parametric cost function approximation

- Benchmark policy – Deterministic lookahead

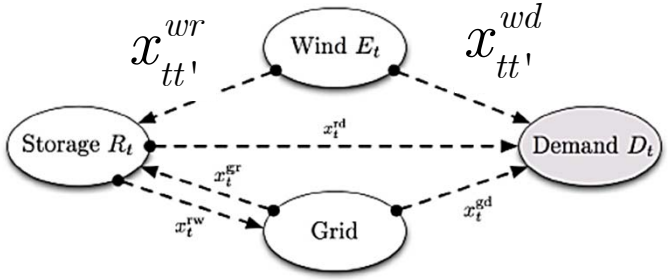
$$X_t^{\text{D-LA}}(S_t) = \underset{x_t,(\tilde{x}_{tt'},t'=t+1,\ldots,t+H)}{\text{argmin}} \left(C(S_t,x_t) + \left[\sum_{t'=t+1}^{t+H} \tilde{c}_{tt'}\tilde{x}_{tt'} \right] \right)$$

$$\begin{aligned}
\tilde{x}_{tt'}^{wd} + \beta\tilde{x}_{tt'}^{rd} + \tilde{x}_{tt'}^{gd} &\le f_{tt'}^{D} \\
\tilde{x}_{tt'}^{gd} + \tilde{x}_{tt'}^{gr} &\le f_{tt'}^{G} \\
\tilde{x}_{tt'}^{rd} + \tilde{x}_{tt'}^{rg} &\le \tilde{R}_{tt'} \\
\tilde{x}_{tt'}^{wr} + \tilde{x}_{tt'}^{gr} &\le R^{\max} - \tilde{R}_{tt'} \\
\tilde{x}_{tt'}^{wr} + \tilde{x}_{tt'}^{wd} &\le f_{tt'}^{E} \\
\tilde{x}_{tt'}^{wr} + \tilde{x}_{tt'}^{gr} &\le \gamma^{charge} \\
\tilde{x}_{tt'}^{rd} + \tilde{x}_{tt'}^{rg} &\le \gamma^{discharge}
\end{aligned}$$

Parametric cost function approximation

- Parametric cost function approximations
  - » Replace the constraint

$$\tilde{x}^{wr}_{tt'} + \tilde{x}^{wd}_{tt'} \leq f^{E}_{tt'}$$



with:

  - » Lookup table modified forecasts (one adjustment term for each time $\tau = t' - t$ in the future):

$$x^{wr}_{tt'} + x^{wd}_{tt'} \leq \theta_{t'-t} f^{E}_{tt'}$$

  - » Exponential function for adjustments (just two parameters)

$$x^{wr}_{tt'} + x^{wd}_{tt'} \leq \theta_1 e^{\theta_2 (t'-t)} f^{E}_{tt'}$$

  - » Constant adjustment (one parameter)

$$x^{wr}_{tt'} + x^{wd}_{tt'} \leq \theta f^{E}_{tt'}$$

Parametric cost function approximation

- Optimizing the CFA:
  - » Let $\bar{F}(\theta,\omega)$ be a simulation of our policy given by

$$\bar{F}(\theta,\omega)=\sum_{t=0}^{T}C\left(S_t(\omega),X_t^{\pi}(S_t(\omega)\mid\theta)\right)$$

  - » We then compute the gradient with respect to θ

$$\nabla_\theta F(\theta,\omega)=\nabla_\theta \bar{F}(\theta,\omega)$$

  - » The parameter θ is found using a classical stochastic gradient algorithm:

$$\theta^{n+1}=\theta^n+\alpha_n\nabla_\theta F(\theta^n,\omega^{n+1})$$

We tested several stepsize formulas and found that ADAGRAD worked best:

$$\alpha_n=\frac{\eta}{\sqrt{G_t+\varepsilon}}\qquad G_t=\sum_{t'=0}^{t}\left(\nabla_x F(x_t,W_{t+1})^2\right)$$

Parametric cost function approximation

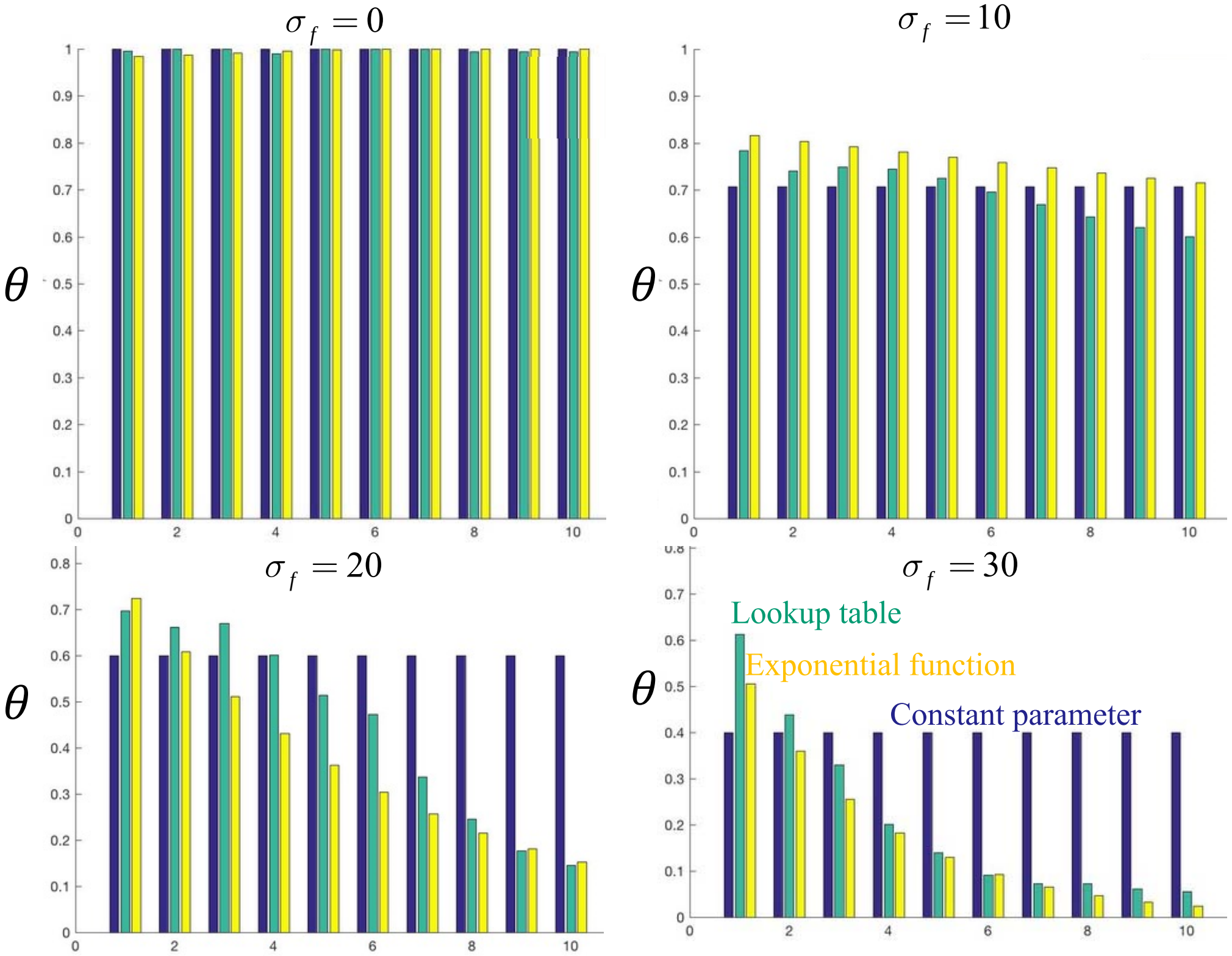
- Optimizing the CFA:
  - » We compute the gradient by applying the chain rule

$$\nabla_\theta \bar{F} = \left(\frac{\partial C_0}{\partial X_0}\cdot\frac{\partial X_0}{\partial \theta}\right)+\sum_{t'=1}^{T}\left[\left(\frac{\partial C_{t'}}{\partial S_{t'}}\cdot\frac{\partial S_{t'}}{\partial \theta}\right)+\left(\frac{\partial C_{t'}}{\partial X_{t'}(S_t|\theta)}\cdot\left(\frac{\partial X_{t'}(S_t|\theta)}{\partial S_{t'}}\cdot\frac{\partial S_{t'}}{\partial \theta}+\frac{\partial X_{t'}(S_t|\theta)}{\partial \theta}\right)\right)\right],$$

where the interaction from one time period to the next is captured using

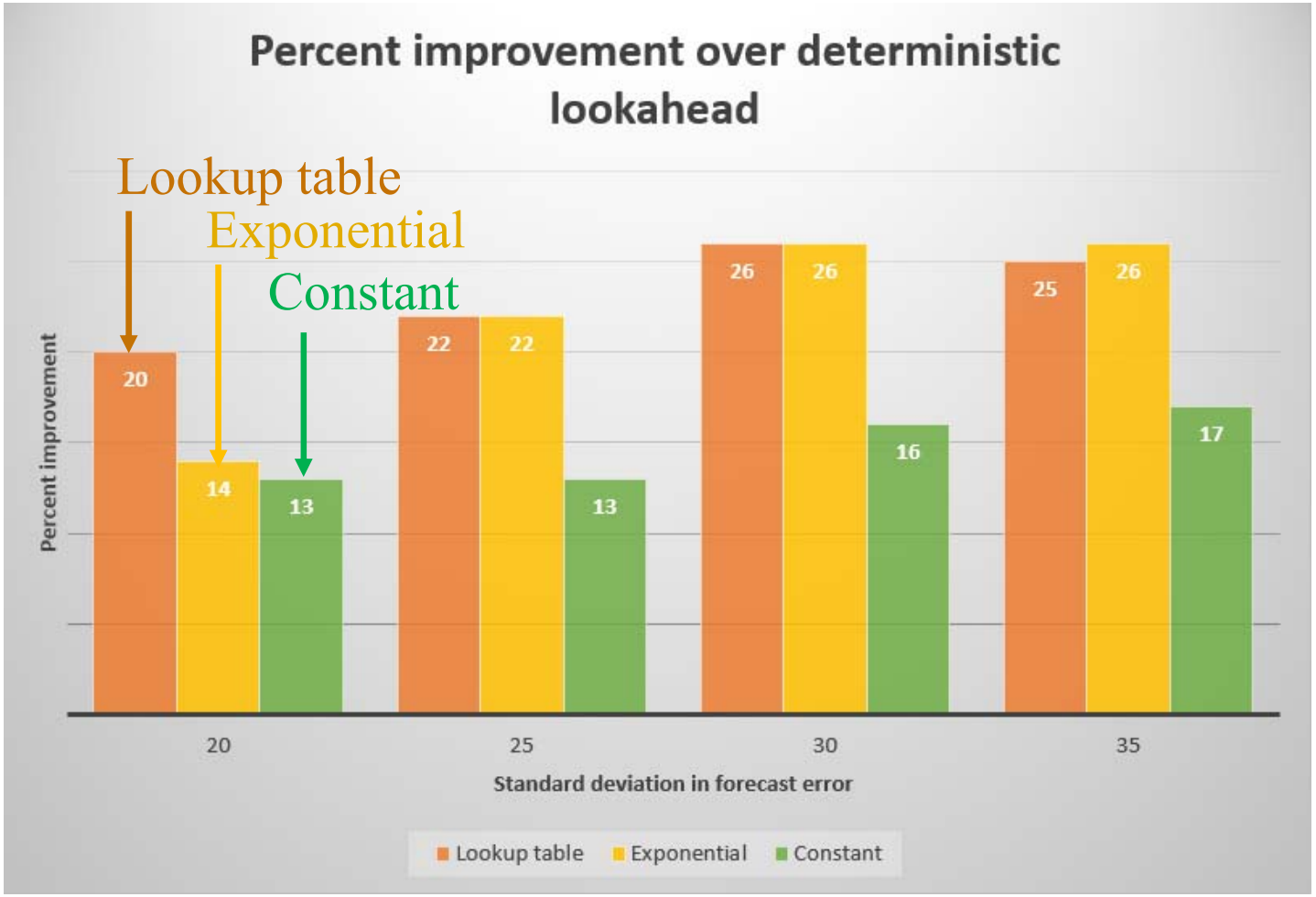
$$\frac{\partial S_{t'}}{\partial \theta} = \frac{\partial S_{t'}}{\partial S_{t'-1}}\cdot\frac{\partial S_{t'-1}}{\partial \theta}+\frac{\partial S_{t'}}{\partial X_{t'-1}(S_{t-1}|\theta)}\cdot\left[\frac{\partial X_{t;-1}(S_{t-1}|\theta)}{\partial S_{t'-1}}\cdot\frac{\partial S_{t'-1}}{\partial \theta}+\frac{\partial X_{t'-1}(S_{t-1}|\theta)}{\partial \theta}\right].$$

  - » Assuming there are no integer variables, these equations are quite easy to compute.


$\sigma_f = 0$
$\sigma_f = 10$
$\sigma_f = 20$
$\sigma_f = 30$
θ
Lookup table
Exponential function
Constant parameter

Parametric cost function approximation

- Improvement over deterministic benchmark:



Parametric cost function approximation

- The parametric CFA represents a fundamental rethinking of the modeling of stochastic programming problems:
  - » From thinking of the lookahead model as the objective function:

$$\max c_0 x_0 + \sum_{\omega \in \Omega} p(\omega) \sum_{t=1}^{T} c_t(\omega) x_t(\omega)$$

  - » To acknowledging that the lookahead model is a policy for solving the base model…

$$\max_{\pi=\theta} E^{\pi} \left\{ \sum_{t=0}^{T} C_t \left(S_t, X_t^{\pi}(S_t \mid \theta), W_{t+1} \right) \mid S_0 \right\}$$

…. which is a simulator where we do not have to make any of the standard approximations required in stochastic programming.

An energy storage problem

- Consider a basic energy storage problem:

» We are going to show that with minor variations in the characteristics of this problem, we can make *each* class of policy work best.

An energy storage problem

- We can create distinct flavors of this problem:
  - » Problem class 1 – Best for PFAs
    - Highly stochastic (heavy tailed) electricity prices
    - Stationary data
  - » Problem class 2 – Best for CFAs
    - Stochastic prices and wind (but not heavy tailed)
    - Stationary data
  - » Problem class 3 - Best for VFAs
    - Stochastic wind and prices (but not too random)
    - Time varying loads, but inaccurate wind forecasts
  - » Problem class 4 – Best for deterministic lookaheads
    - Relatively low noise problem with accurate forecasts
  - » Problem class 5 – A hybrid policy worked best here
    - Stochastic prices and wind, nonstationary data, noisy forecasts.

An energy storage problem

- The policies
  - » The PFA:
    - Charge battery when price is below p1
    - Discharge when price is above p2
  - » The CFA
    - Optimize over a horizon H; maintain upper and lower bounds (u, l) for every time period except the first (note that this is a hybrid with a lookahead).
  - » The VFA
    - Piecewise linear, concave value function in terms of energy, indexed by time.
  - » The lookahead (deterministic)
    - Optimize over a horizon H (only tunable parameter) using forecasts of demand, prices and wind energy
  - » The lookahead CFA
    - Use a lookahead policy (deterministic), but with a tunable parameter that improves robustness.

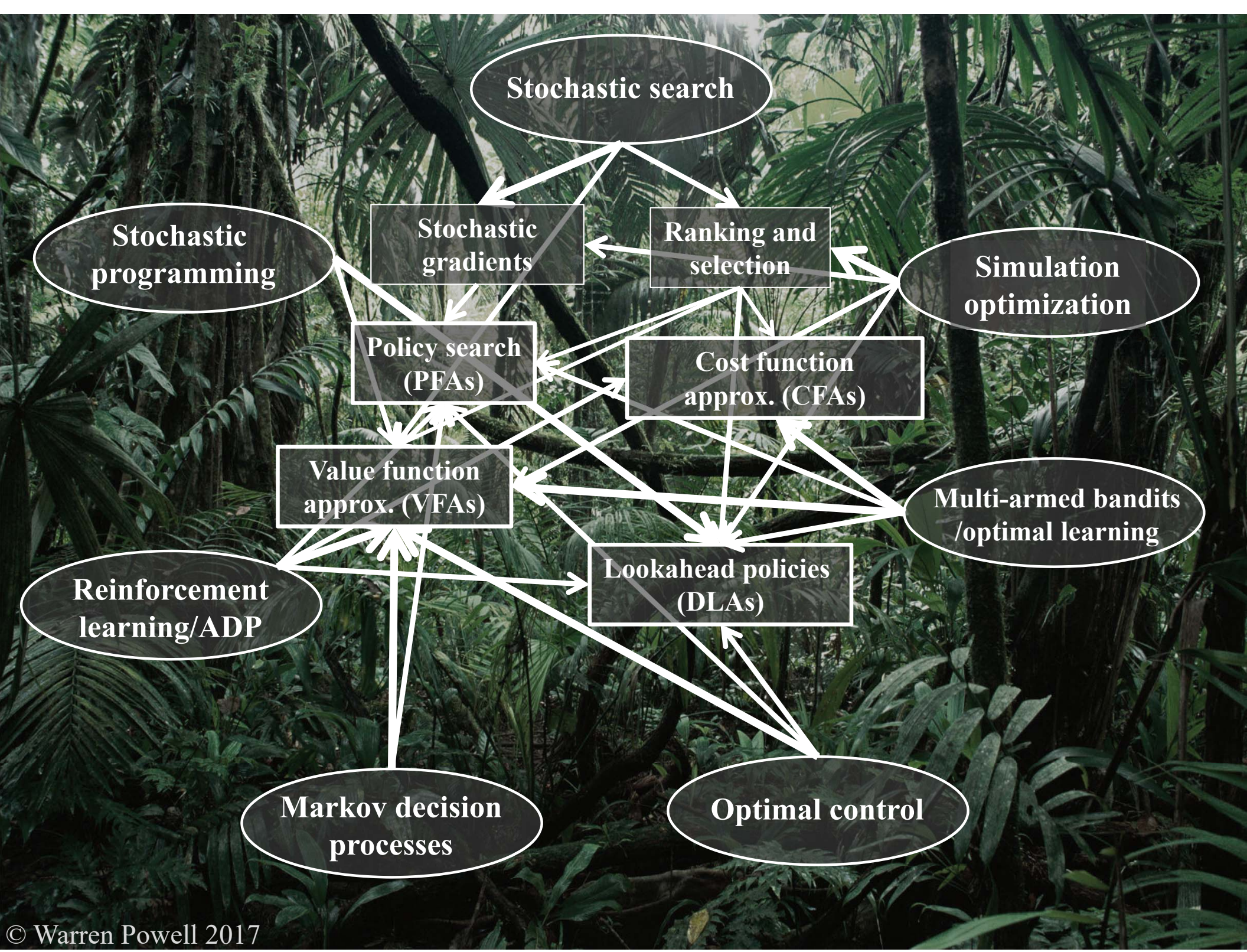
An energy storage problem

- Each policy is best on certain problems
  - » Results are percent of *posterior* optimal solution

Problem:	Problem description	PFA	CFA Error correction	VFA	Determ. Lookahead	CFA Lookahead
A	A stationary problem with heavy-tailed prices, relatively low noise, moderately accurate forecasts.	**0.959**	**0.839**	**0.936**	**0.887**	**0.887**
B	A time-dependent problem with daily load patterns, no seasonalities in energy and price, relatively low noise, less accurate forecasts.	**0.714**	**0.752**	**0.712**	**0.746**	**0.746**
C	A time-dependent problem with daily load, energy and price patterns, relatively high noise, forecast errors increase over horizon.	**0.865**	**0.590**	**0.914**	**0.886**	**0.886**
D	A time-dependent problem, relatively low noise, very accurate forecasts.	**0.962**	**0.749**	**0.971**	**0.997**	**0.997**
E	Same as (C), but the forecast errors are stationary over the planning horizon.	**0.865**	**0.590**	**0.914**	**0.922**	**0.934**

  - » … any policy might be best depending on the data.

Joint research with Prof. Stephan Meisel, University of Muenster, Germany.



Thank you!

http://www.castlelab.princeton.edu/

A tutorial on this topic is available at the top of

http://www.castlelab.princeton.edu/jungle/