Lecture outline

- What is a policy?
  - Policy function approximations (PFAs)
  - Cost function approximations (CFAs)
  - Value function approximations (VFAs)
  - Lookahead policies
- Finding good policies
- Optimizing continuous parameters

What is a policy?

- Last time, we saw a few examples of "policies"
  - » Searching over a graph
  - » Learning when to sell an asset
- A policy is any rule/function that maps a state to an action.
  - » This is *the* reason why a state must be all the information you need to make a decision (now or in the future).
- Policies come in many forms, but these can be organized into major groups:
  - » Policy function approximations (PFAs)
  - » Policies based on cost function approximations (CFAs)
  - » Policies based on value function approximations (VFAs)
  - » Lookahead policies

What is a policy?

1) Policy function approximations (PFAs)

» Lookup table

- Recharge the battery between 2am and 6am each morning, and discharge as needed.

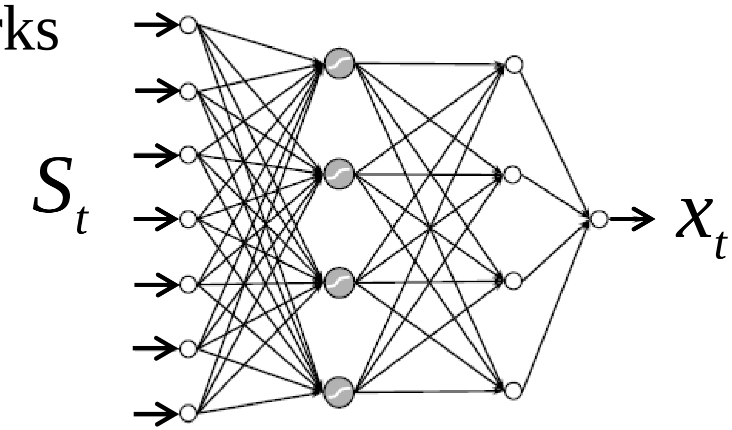
» Parameterized functions

- Recharge the battery when the price is below θ^{charge} and discharge when the price is above $\theta^{\text{discharge}}$

» Regression models

$$X^{PFA}(S_t \mid \theta) = \theta_0 + \theta_1 S_t + \theta_2 \left(S_t\right)^2$$

» Neural networks



What is a policy?

2) Cost function approximations (CFAs)

» Take the action that maximizes contribution (or minimizes cost) for just the current time period:

$$X^M(S_t) = \arg\max_{x_t} C(S_t, x_t)$$

» We can parameterize myopic policies with bonus and penalties to encourage good long-term behavior.

» We may use a ***cost function approximation:***

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

The cost function approximation $\bar{C}^{\pi}(S_t, x_t \mid \theta)$ may be designed to produce better long-run behaviors.

What is a policy?

3) Value function approximations (VFAS)

- Using the exact value function

$$X_t^{VFA}(S_t) = \arg\max_{x_t} \left(C(S_t, x_t) + \gamma V_{t+1}(S_{t+1}) \right)$$

This is how we solved the budgeting problem earlier.

- Or by approximating the value function in some way:

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

- This is what most people associate with "approximate dynamic programming" or "reinforcement learning"

What is a policy?

- Four fundamental classes of policies:
  - » 4) Lookahead policies
    - Plan over the next T periods, but implement only the action it tells you to do now.

$$X^M(S_t) = \arg\max_{x_t, x_{t+1}, \ldots, x_{t+T}} \sum_{t'=t}^{T} C(S_{t'}, x_{t'})$$

    - This strategy assumes that we forecast a perfect future, and solve the resulting deterministic optimization problem. There are more advanced strategies that explicitly model uncertainty in the future, but this is for advanced research groups.

Lecture outline

- What is a policy?
  - Policy function approximations (PFAs)
  - Cost function approximations (CFAs)
  - Value function approximations (VFAs)
  - Lookahead policies
- Finding good policies
- Optimizing continuous parameters

Policy function approximations

- Lookup tables
  - » When in discrete state S, take discrete action *a* (or *x*).
  - » These are popular with
    - Playing games (black jack,backgammon, Connect 4, ..)
    - Routing over graphs
    - … many others
  - » Black jack
    - State is cards that you are holding
    - Actions
      - – Double down?
      - – Take a card/hold
    - Let $A^\pi(S_t)$ be a proposed action for each state. This represents a policy. Fix the policy, and play the game many times.
    - Estimate the probability of winning from each state while following this "policy"

Policy function approximations

- Policy function approximation:
  - » Parametric functions
    - Example 1 – Our pricing problem.
      - – Sell if the price exceeds a smoothed estimate by a specific margin

$$X^{\pi}(S_t) = \begin{cases} 1 & \text{if } p_t > \bar{p}_t + \beta \\ 0 & \text{Otherwise} \end{cases}$$

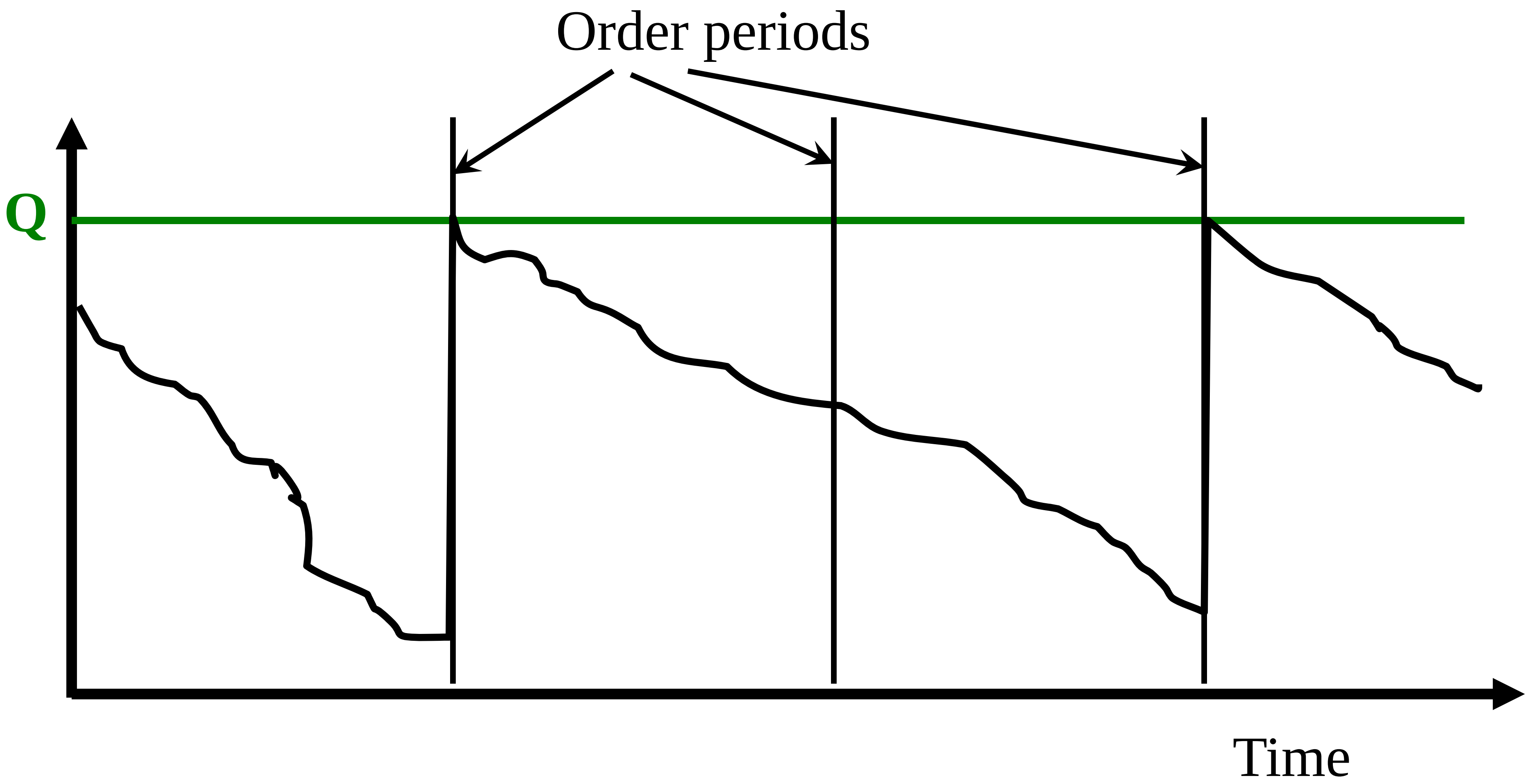
      - – We have to choose a parameter β that determines how much the price has risen over the long run average
    - Example 2 – Inventory ordering policies

$$X^{\pi}(S_t) = \begin{cases} Q - S_t & \text{If } S_t < q \\ 0 & \text{Otherwise} \end{cases}$$

      - – Need to determine (Q,q)

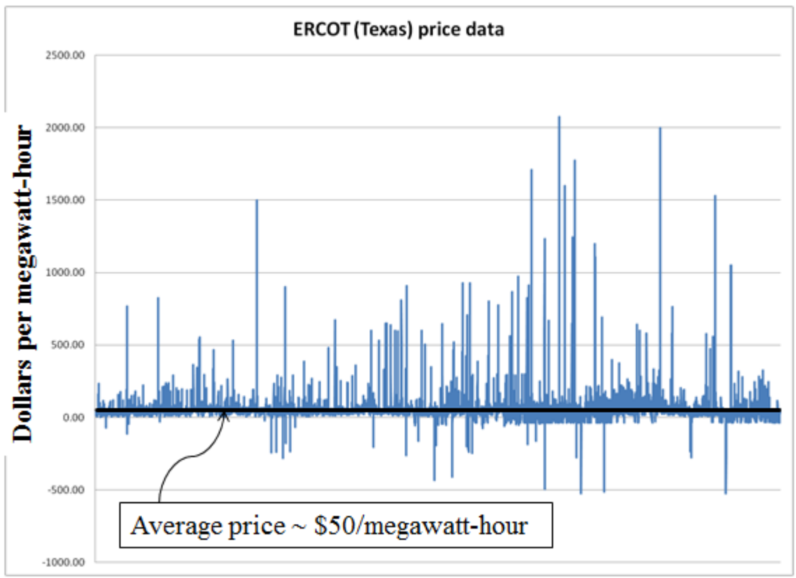
Policy function approximations

- In the presence of fixed order costs and under certain conditions (recall EOQ derivation), an optimal policy is to "order up to" a limit Q:



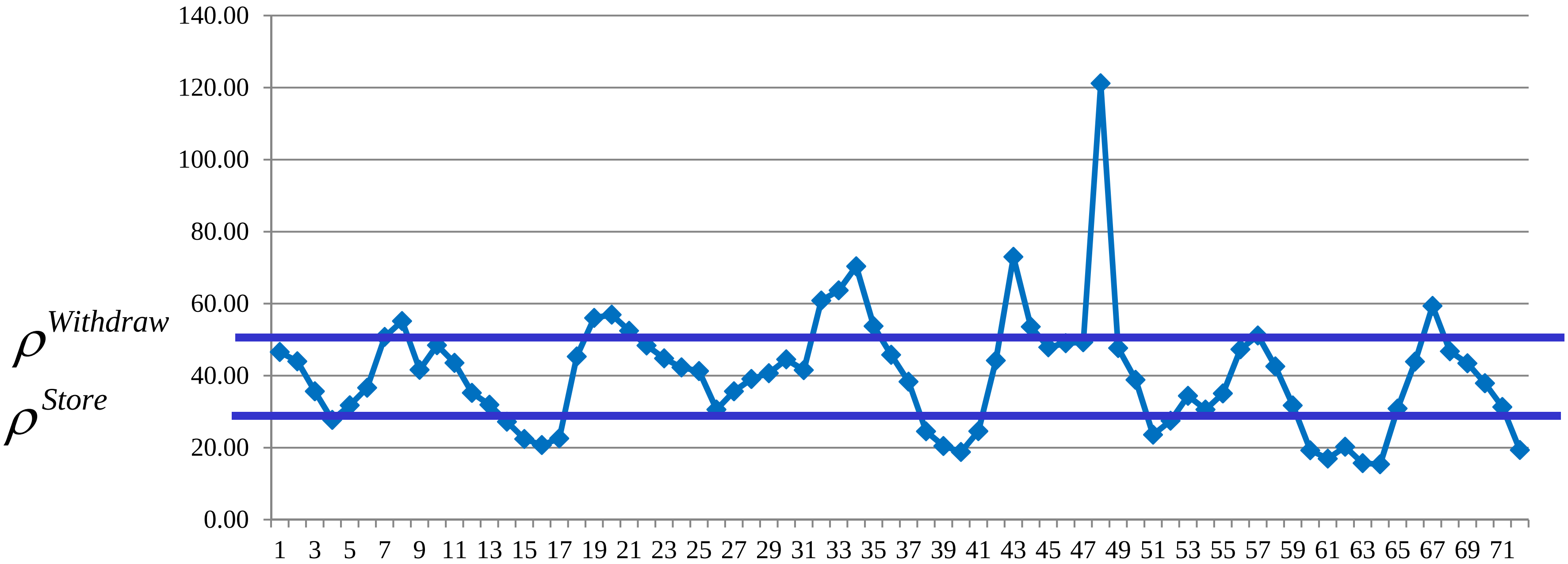
Policy function approximations

- Optimizing a policy for battery arbitrage



Policy function approximations

- We had to design a *simple*, *implementable* policy that did not cheat!



- We need to search for the best values of the parameters ρ^{Store} and $\rho^{Withdraw}$.

Lecture outline

- What is a policy?
  - Policy function approximations (PFAs)
  - Cost function approximations (CFAs)
  - Value function approximations (VFAs)
  - Lookahead policies
- Finding good policies
- Optimizing continuous parameters

Cost function approximation

- Myopic policy
  - » Let $C(s,x)$ be the cost of being in state s and taking action x. For example, this could be the cost of traversing link (i, j), we would choose the link with the lowest cost.
  - » In more complex situations, this means minimizing costs in one day, or month or year, ignoring the impact of decisions now on the future.
  - » We write this policy mathematically using:

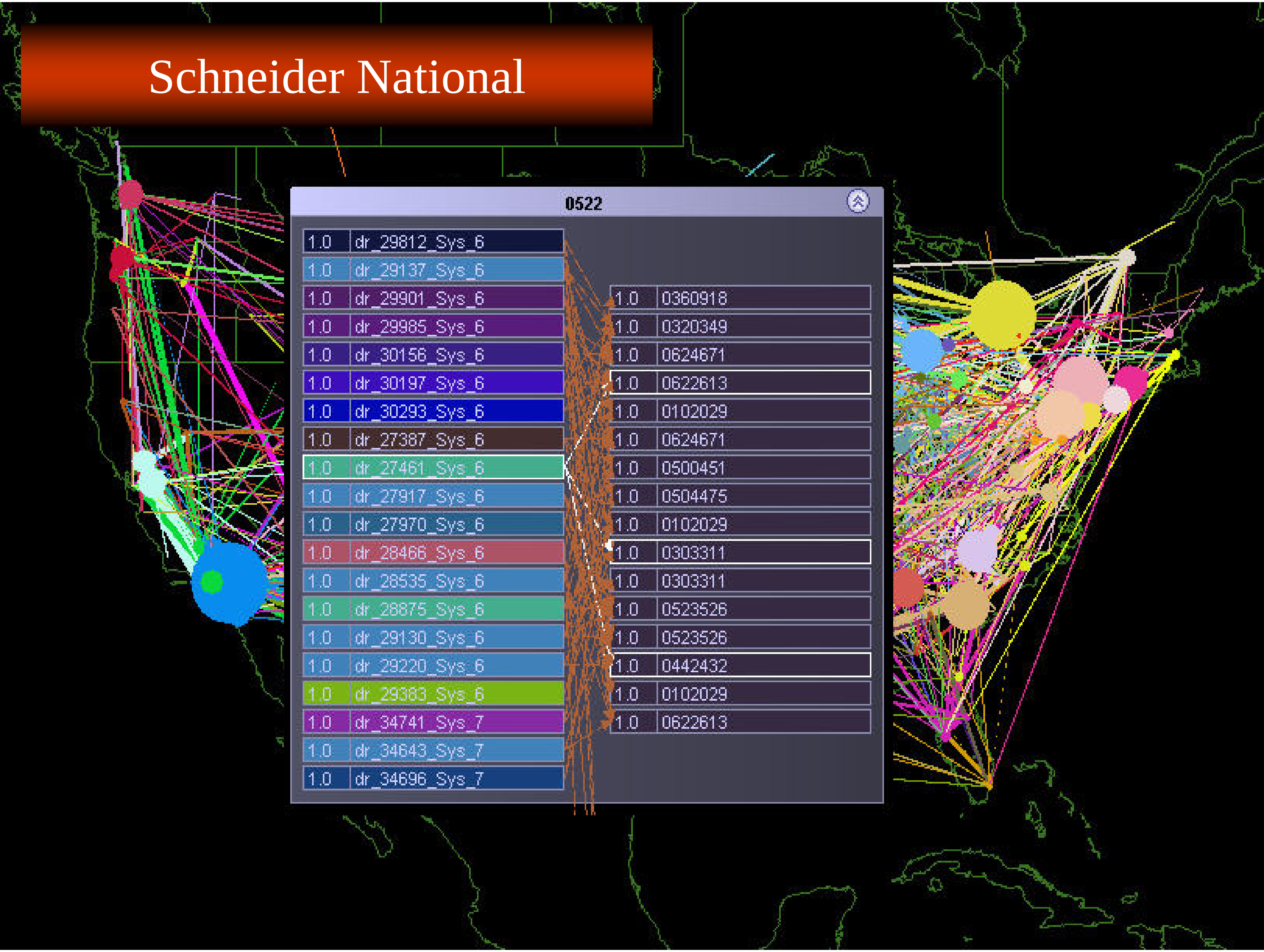
$$X(S_t) = \arg\min(or\max)_x C(S_t, x)$$

  - » Myopic policies can give silly results, but there are problems where it works perfectly well!

Cost function approximation

- Simple examples:
  - » Buying the cheapest laptop.
  - » Taking the job that offers the highest salary.
  - » In a medical emergency, choose the closest ambulance.

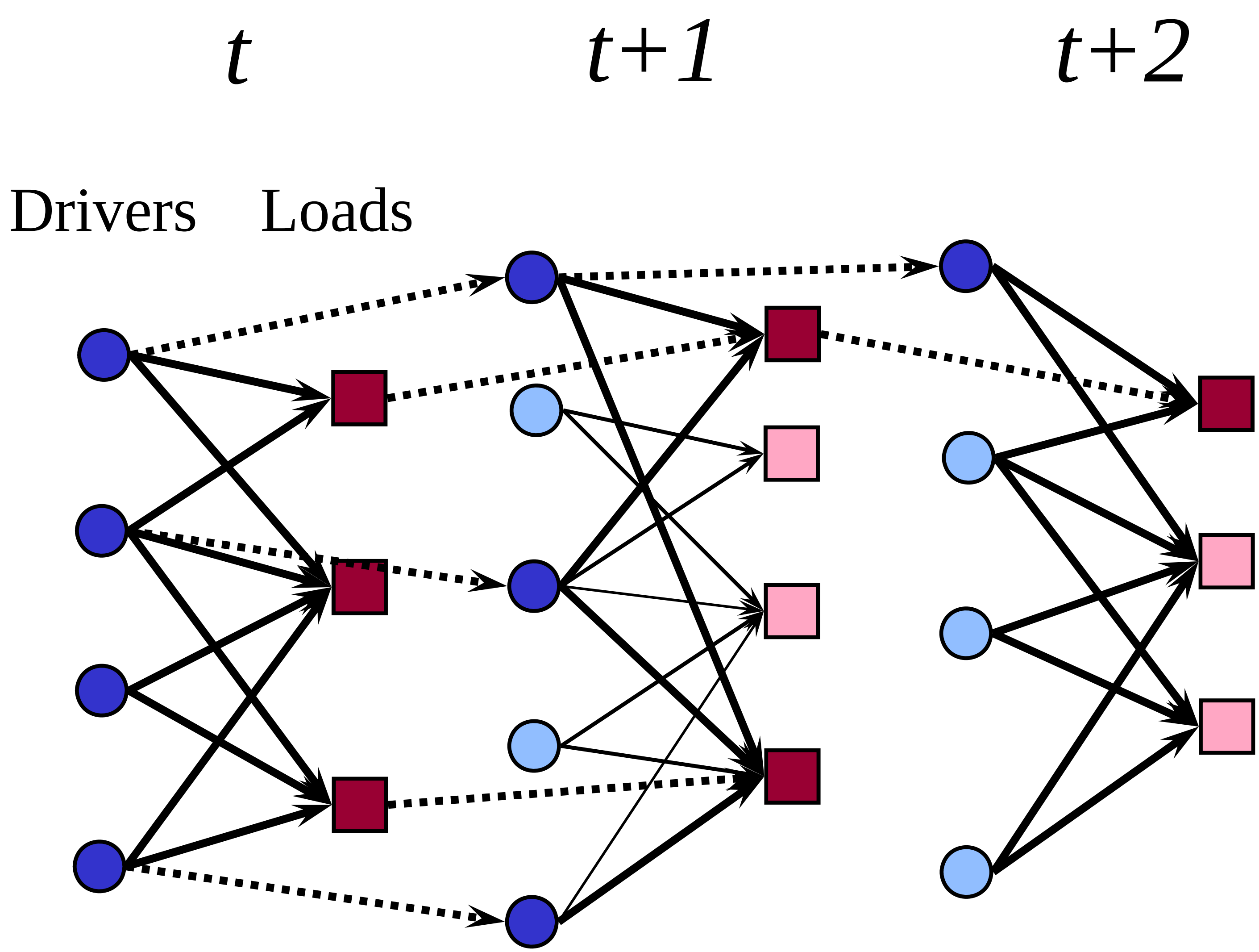
Schneider National




SCHNEIDER
Week 45
Miles 18 4 1 3 1
Working units 18 1
By industry
Auto Parts
Miles 8 27 1 9 33
Growth
SCHNEIDER

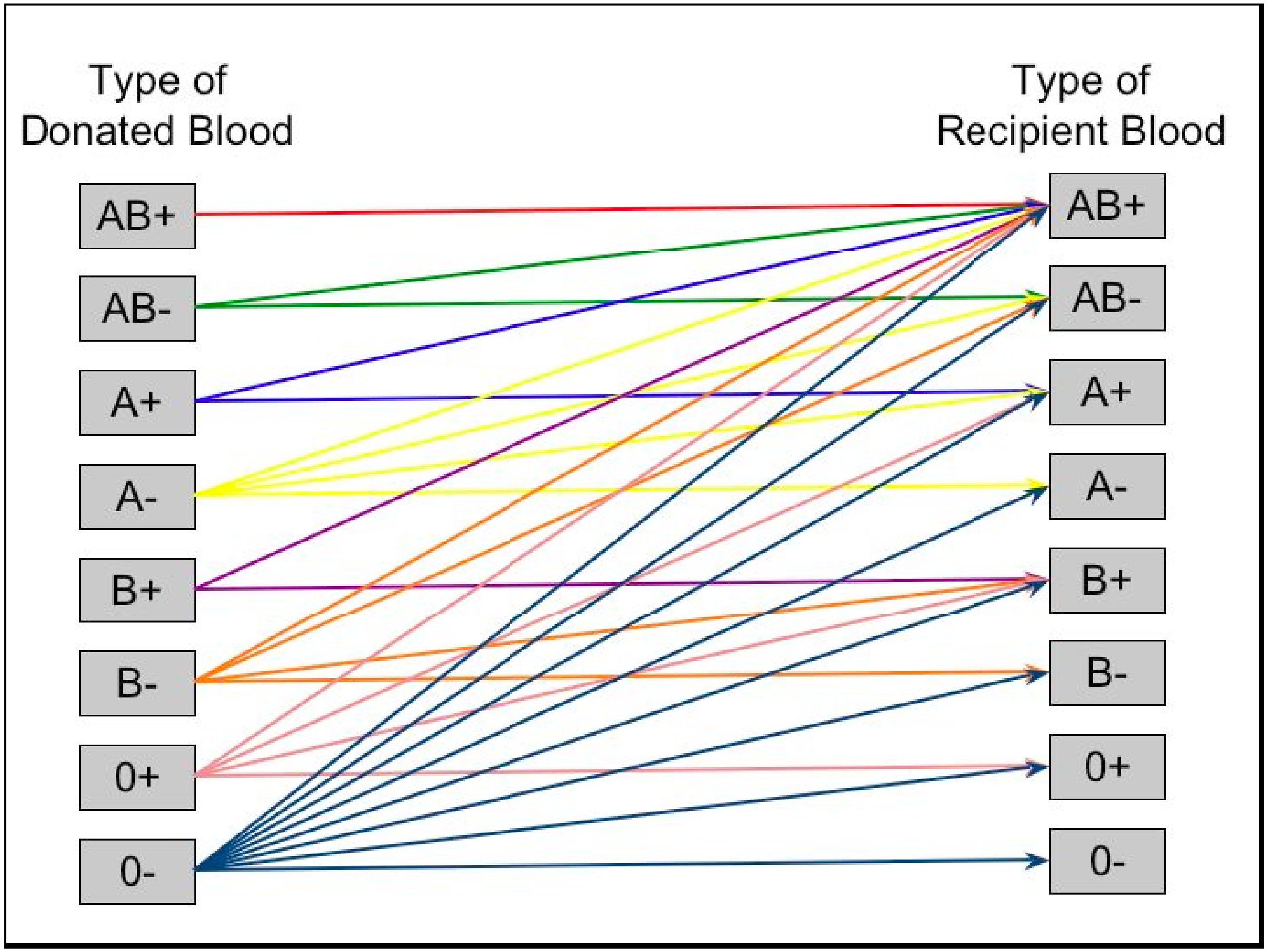
Cost function approximation

- Assigning drivers to loads over time.



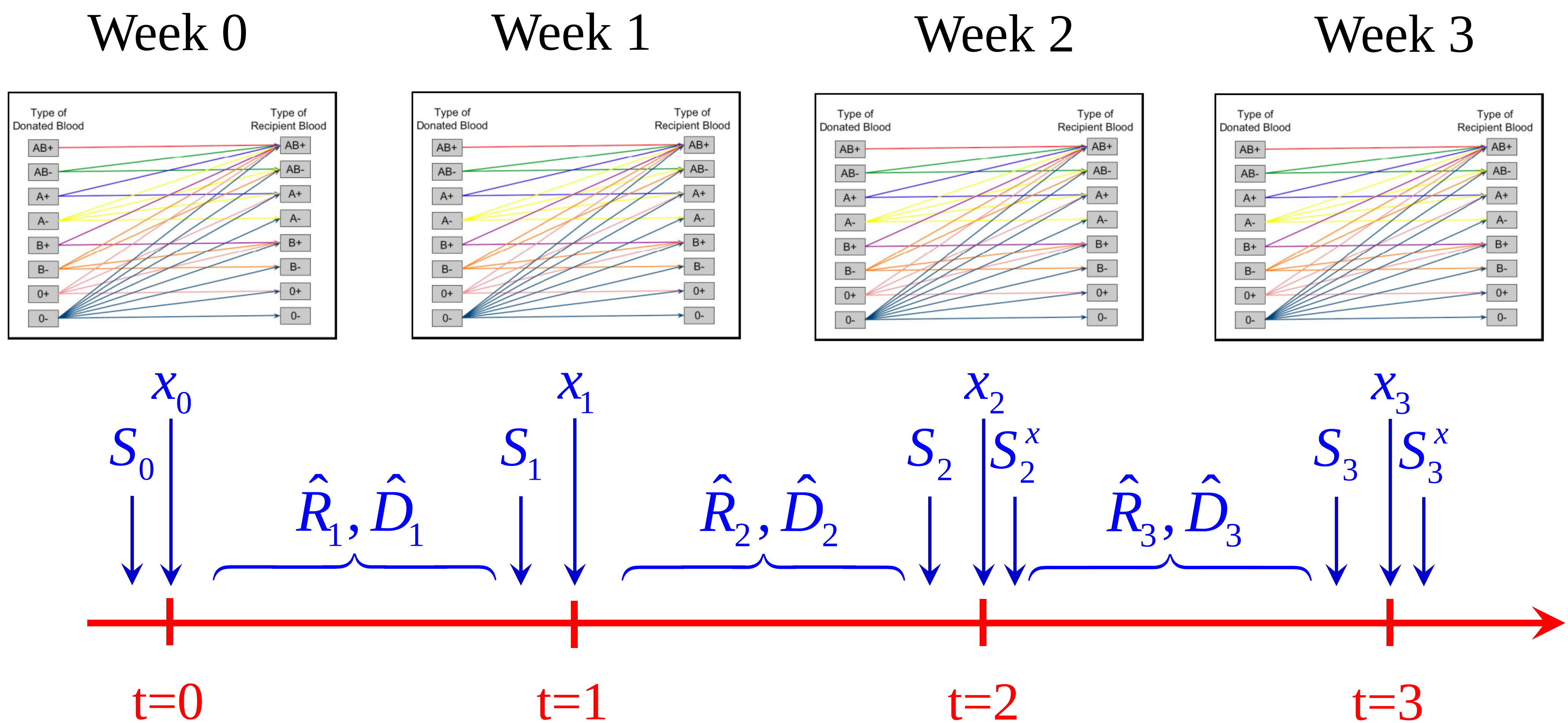
Cost function approximation

- Managing blood inventories



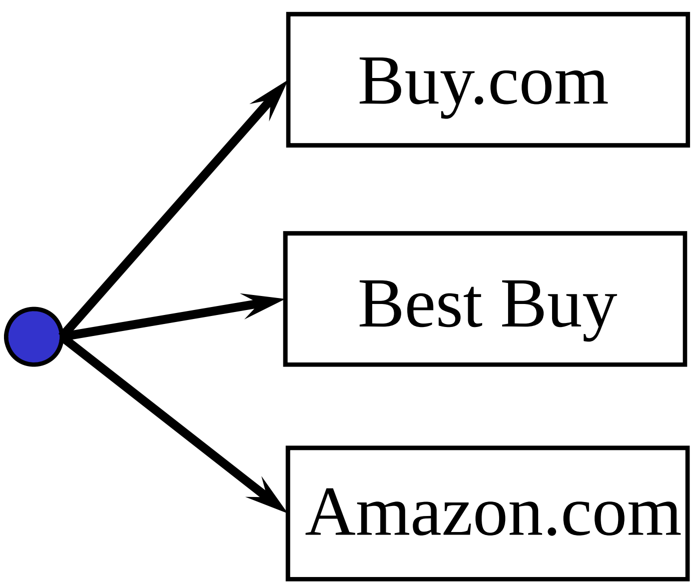
Cost function approximation

- Managing blood inventories over time



Cost function approximation

- Sometimes it is best to modify the cost function to obtain better performance over time
  - » Rather than buy the cheapest laptop over the internet, you purchase it from Best Buy so you can get their service plan. A higher cost now may lower costs in the future.



	Purchase cost	Service plan	Adjusted "cost"
Buy.com	$495	None	$495
Best Buy	$575	Geek squad	$474
Amazon.com	$519	None	$519

Cost function approximation

- Original objective function

$$F = \sum_{d \in \mathcal{D}} c_d x_d$$

$\mathcal{D} =$ Set of stores

$c_d =$ True purchase cost

- Cost function approximation

$$F = \sum_{d \in \mathcal{D}} c_d^{\pi} x_d$$

$\mathcal{D} =$ Set of stores

$c_d =$ True purchase cost

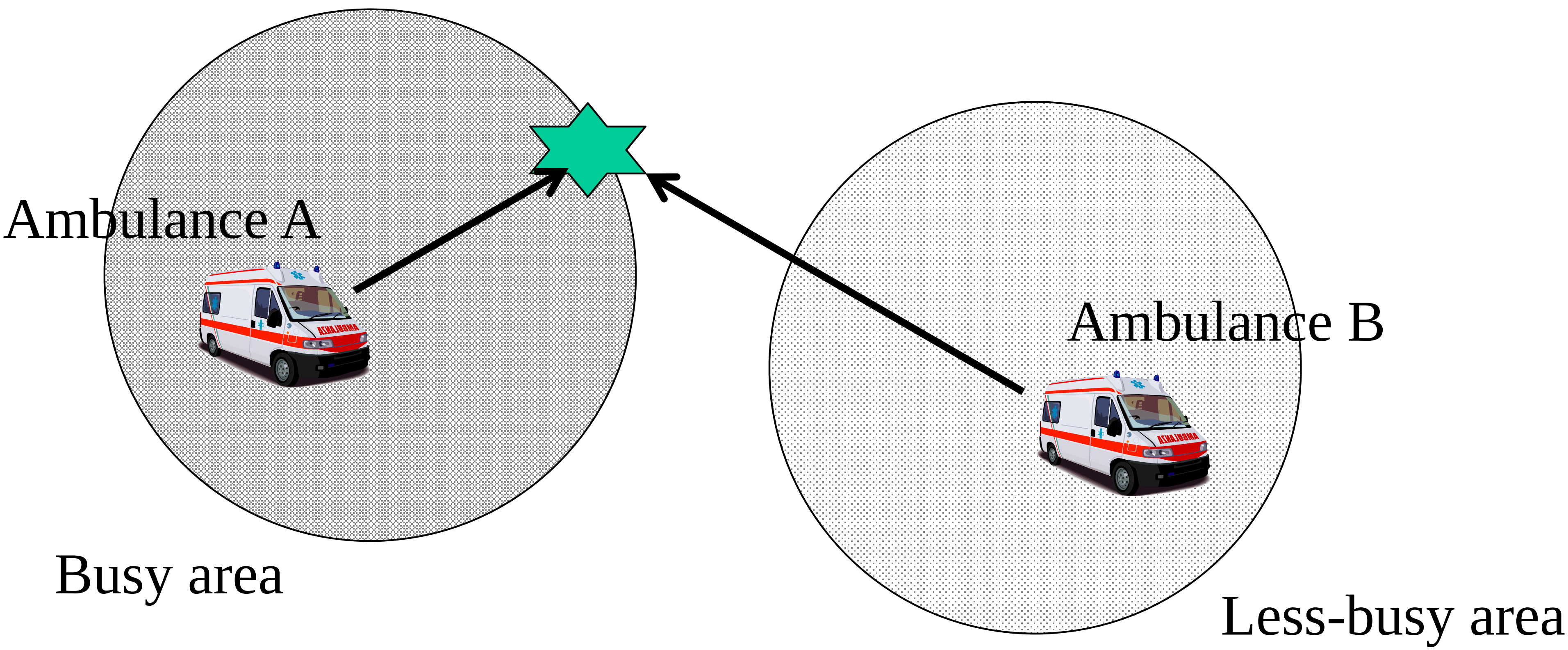
$c_d^{\pi} =$ Modified cost

$= c_d +$ Adjustment for service

The "policy" is captured by the adjustment.

Cost function approximation

- Other adjustments:
  - » Ambulance example
    - Instead of choosing the closest ambulance, we may need to make an adjustment to discourage pulling ambulances from areas which have few alternatives.



Lecture outline

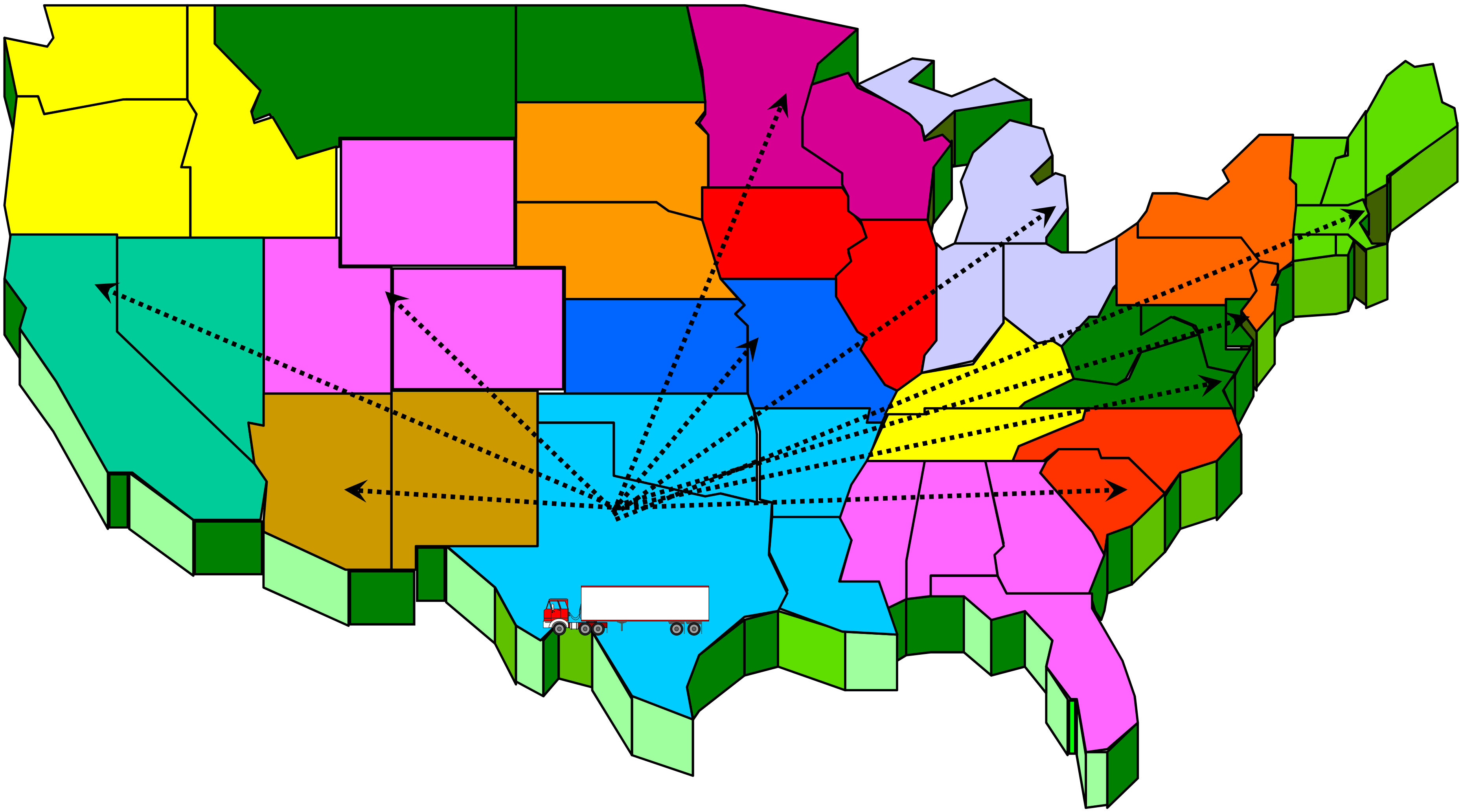
- What is a policy?
  - Policy function approximations (PFAs)
  - Cost function approximations (CFAs)
  - Value function approximations (VFAs)
  - Lookahead policies
- Finding good policies
- Optimizing continuous parameters

Value function approximations

- Basic idea
  - » Take an action, and identify the “state” that an action lands you in.
    - The state of the chess board.
    - The state of your resume from taking a job.
    - A physical location when the action involves moving from one place to another.

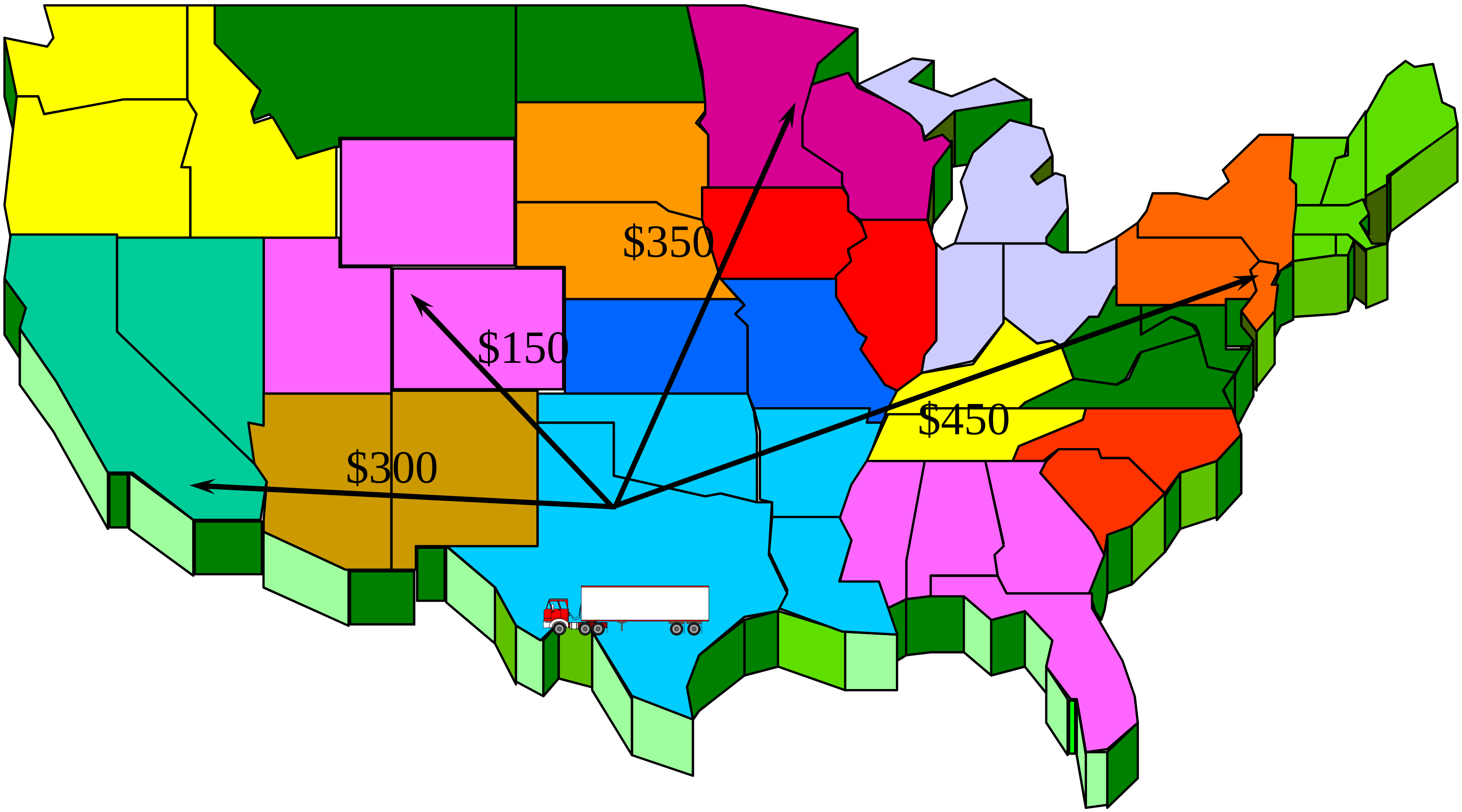
Value function approximations

- The previous post-decision state: trucker in Texas



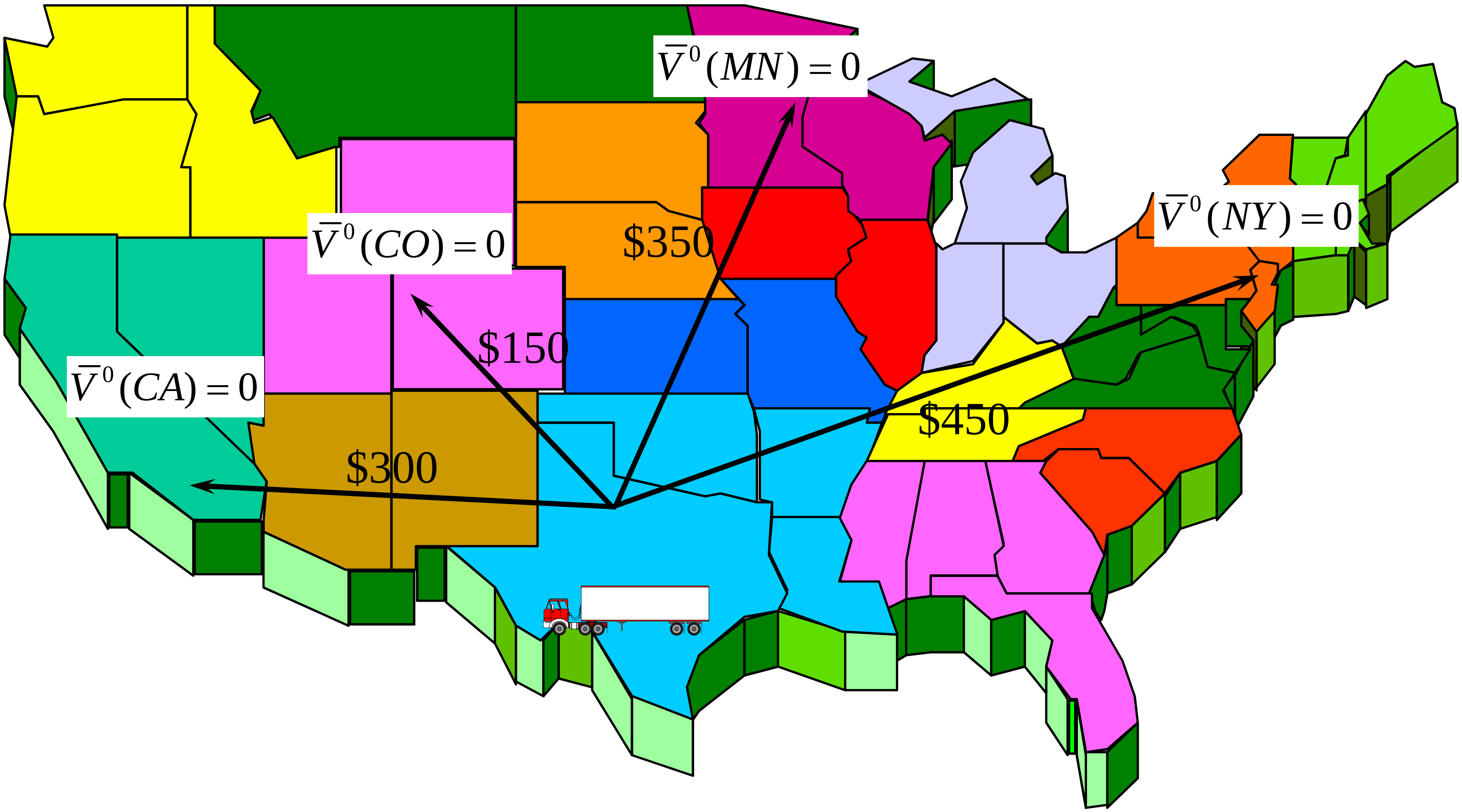
Value function approximations

- Pre-decision state: we see the demands



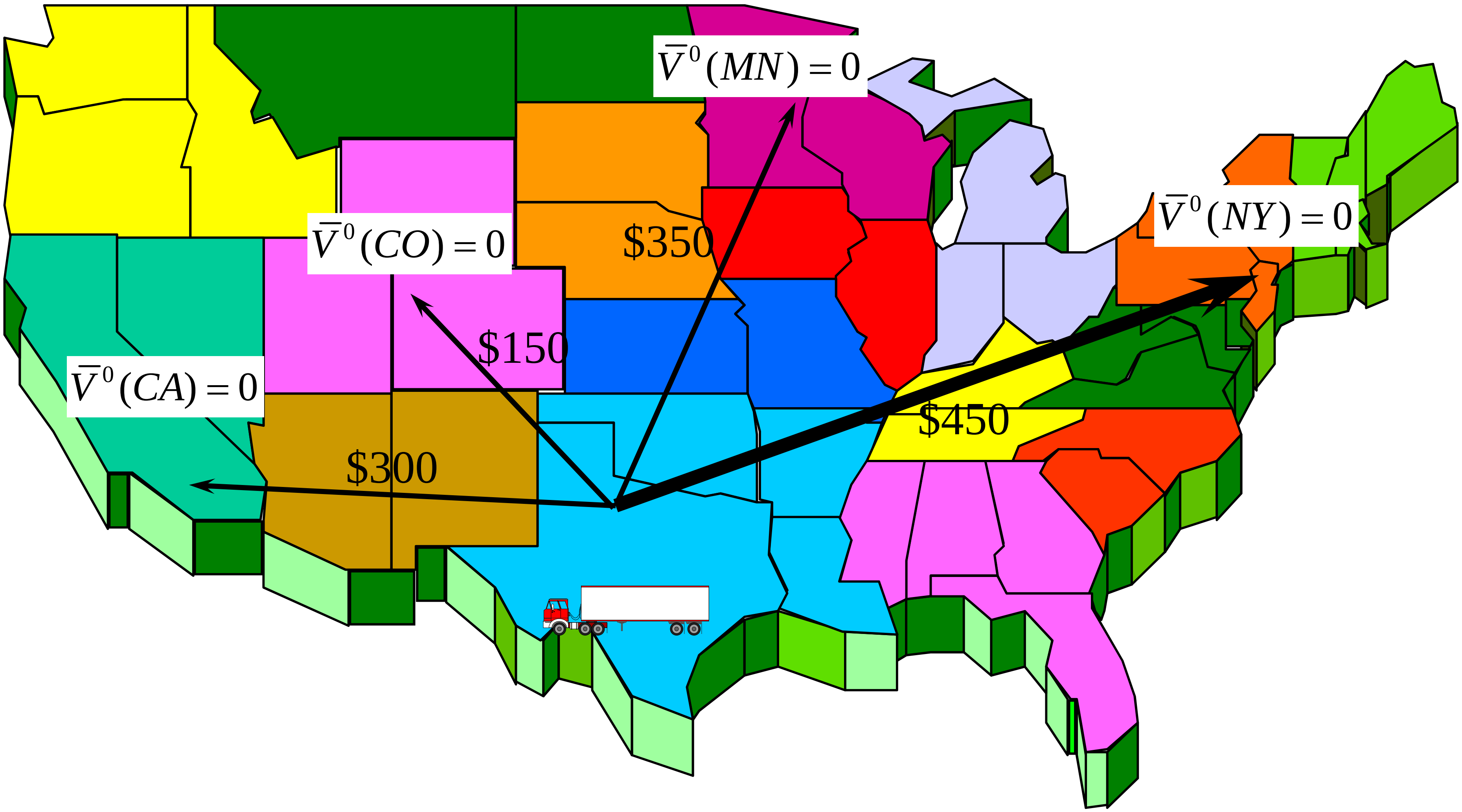
Value function approximations

- We use initial value function approximations…



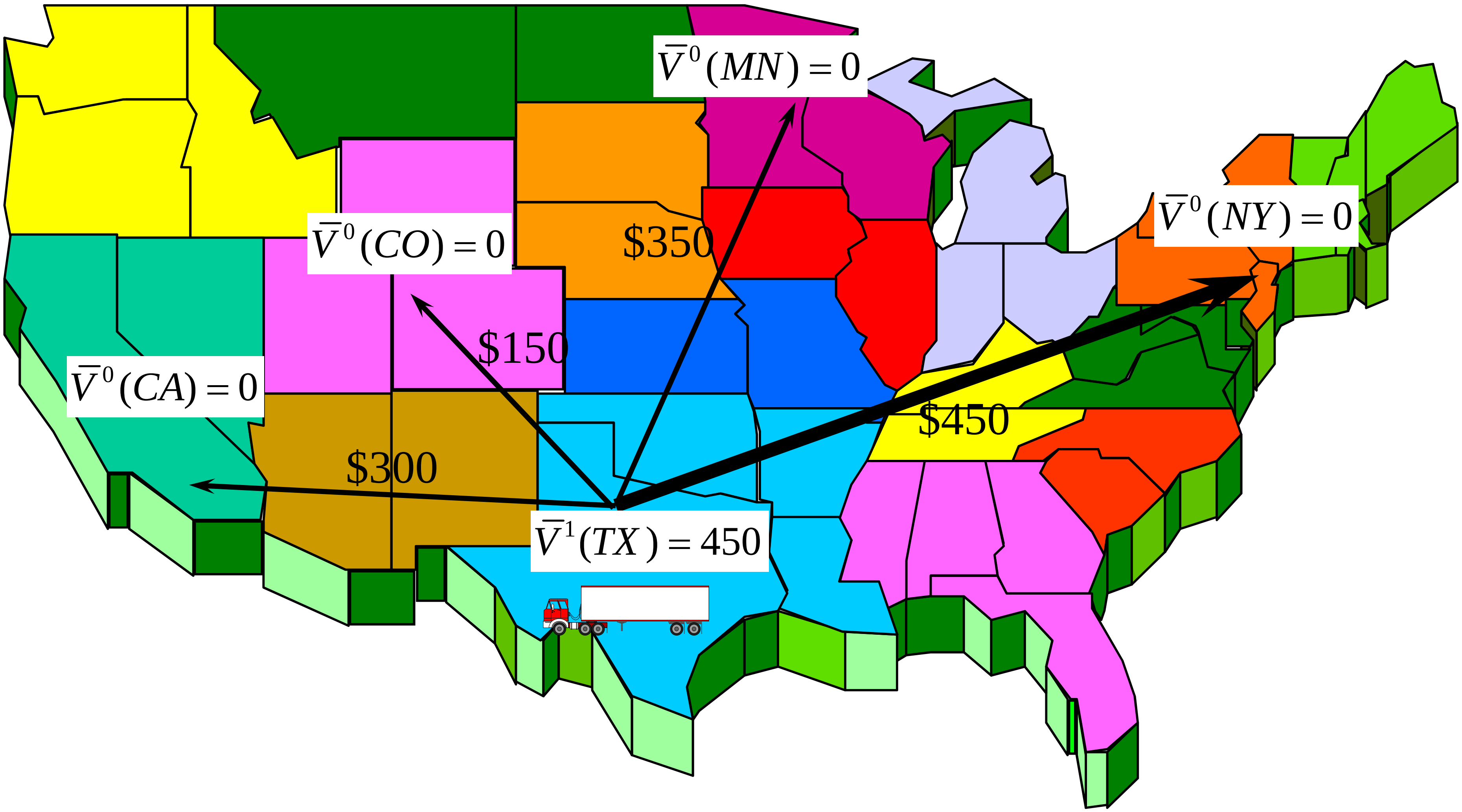
Value function approximations

- … and make our first choice: x^1



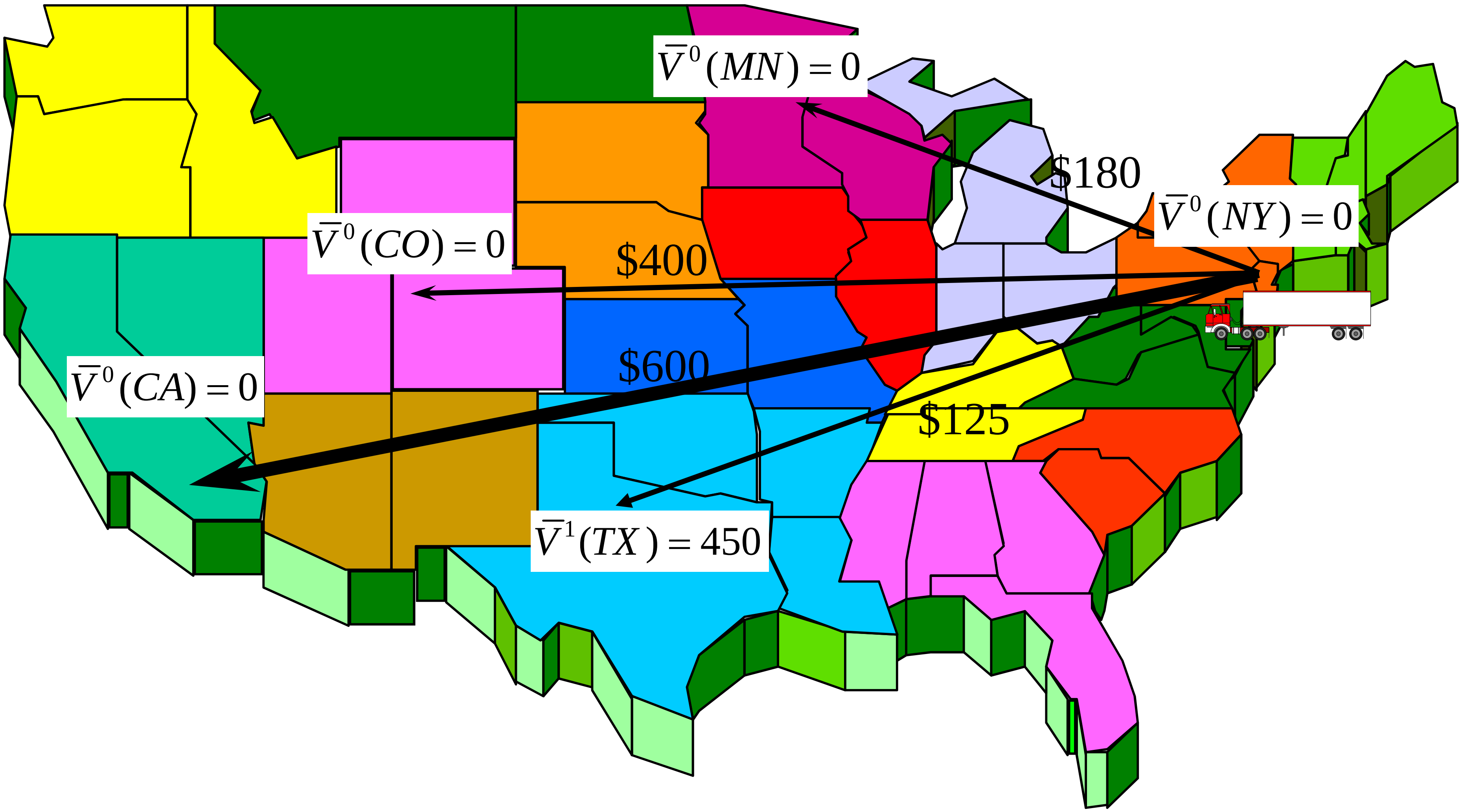
Value function approximations

- Update the value of being in Texas.



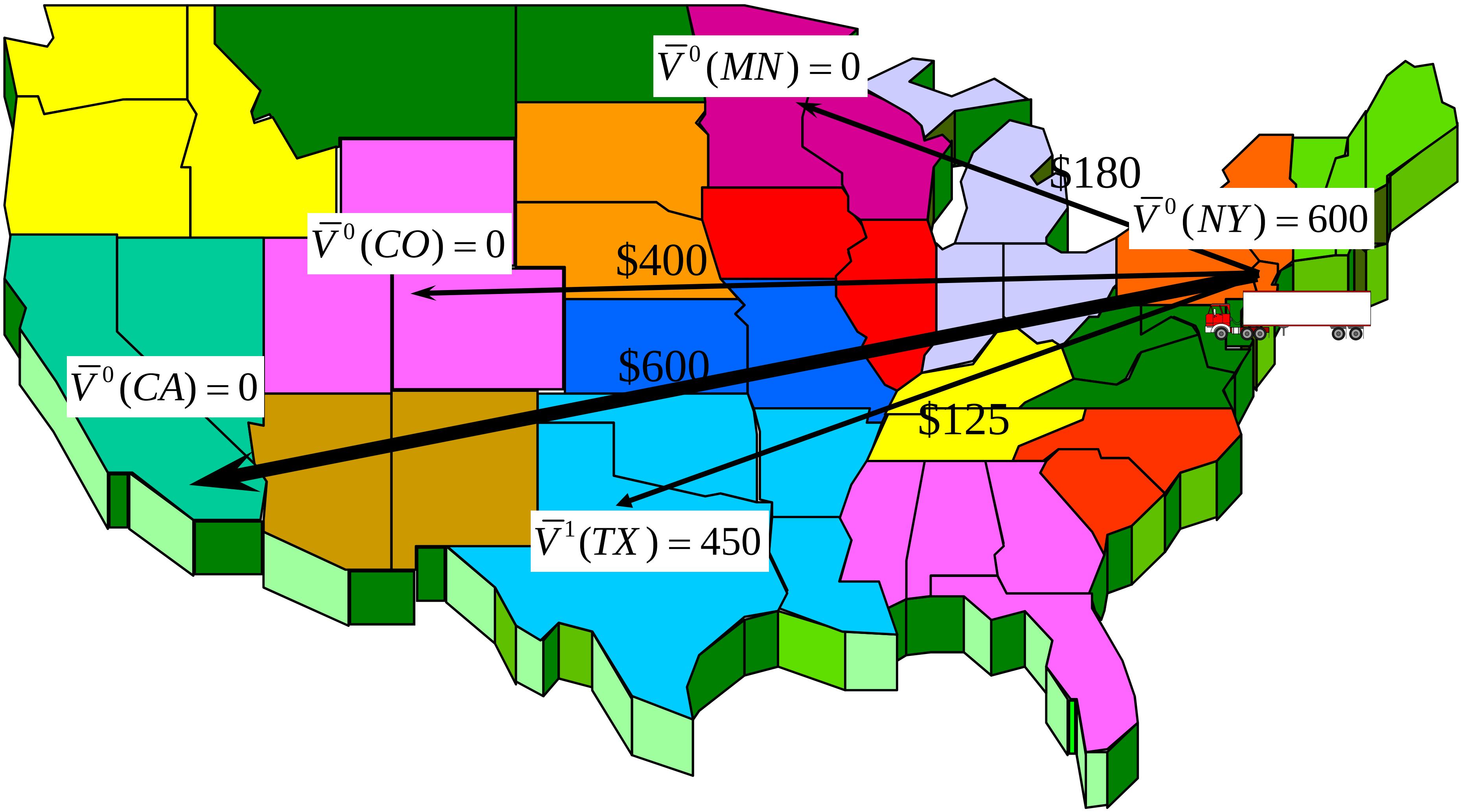
Value function approximations

- Now move to the next state, sample new demands and make a new decision



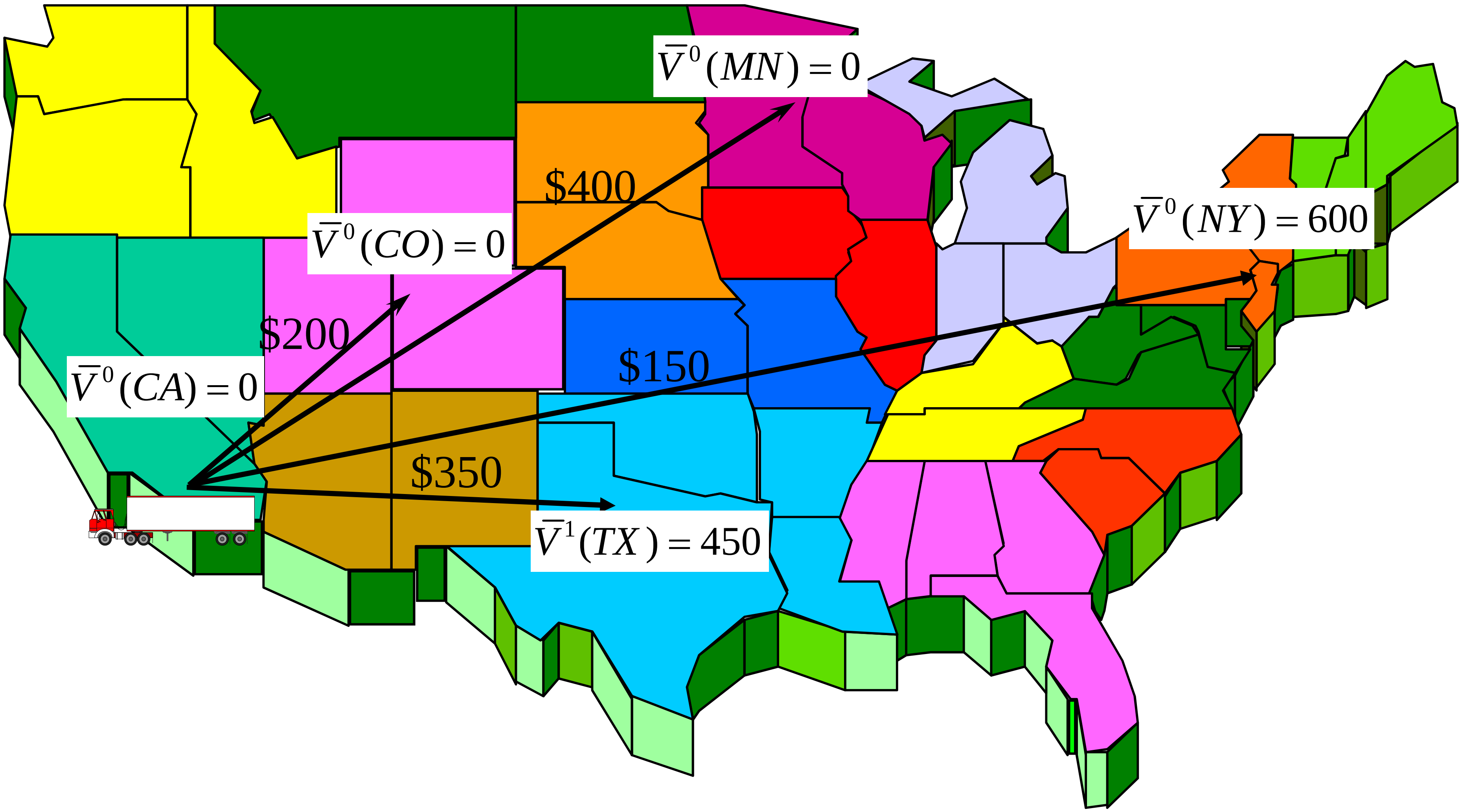
Value function approximations

- Update value of being in NY



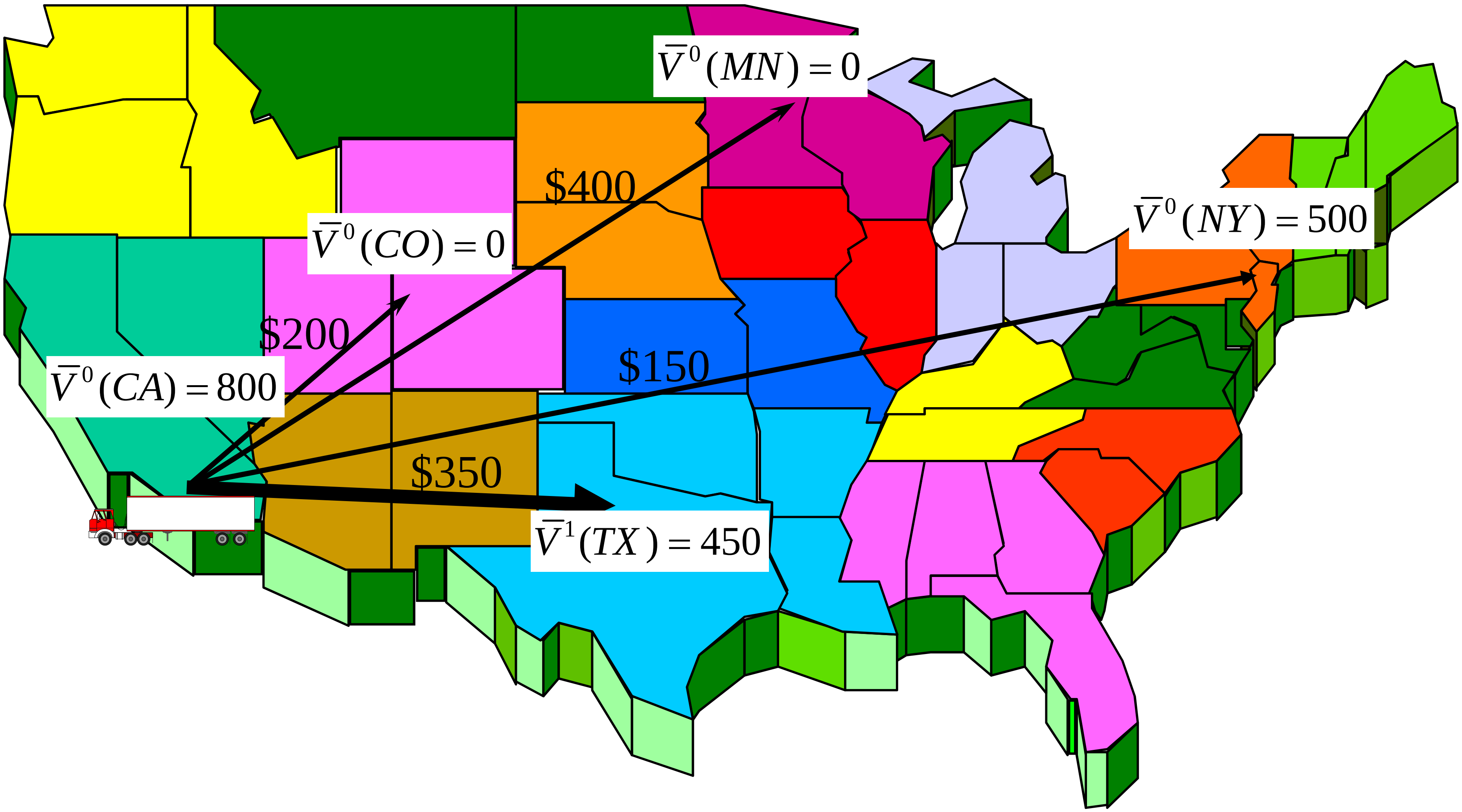
Value function approximations

- Move to California.



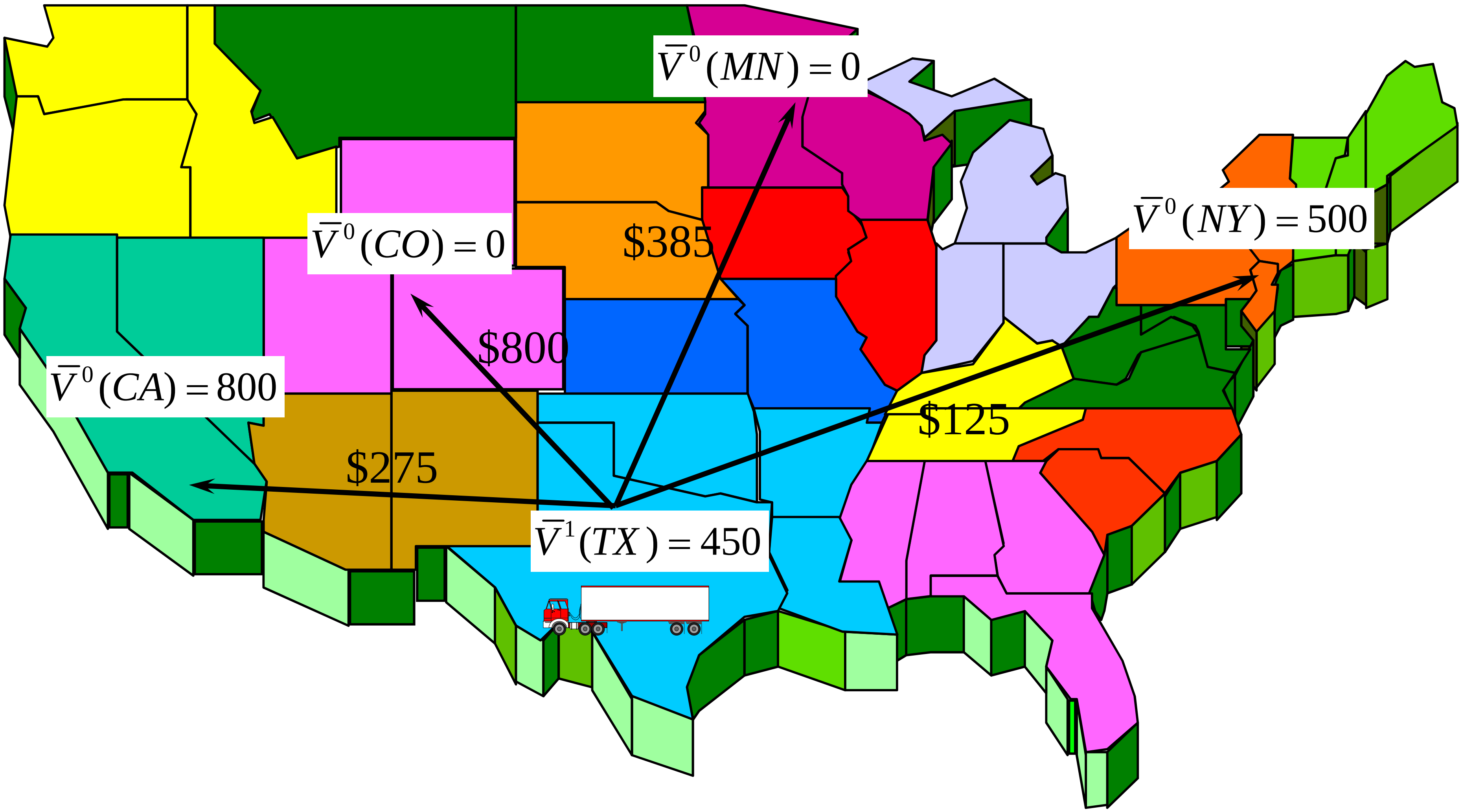
Value function approximations

- Make decision to return to TX and update value of being in CA



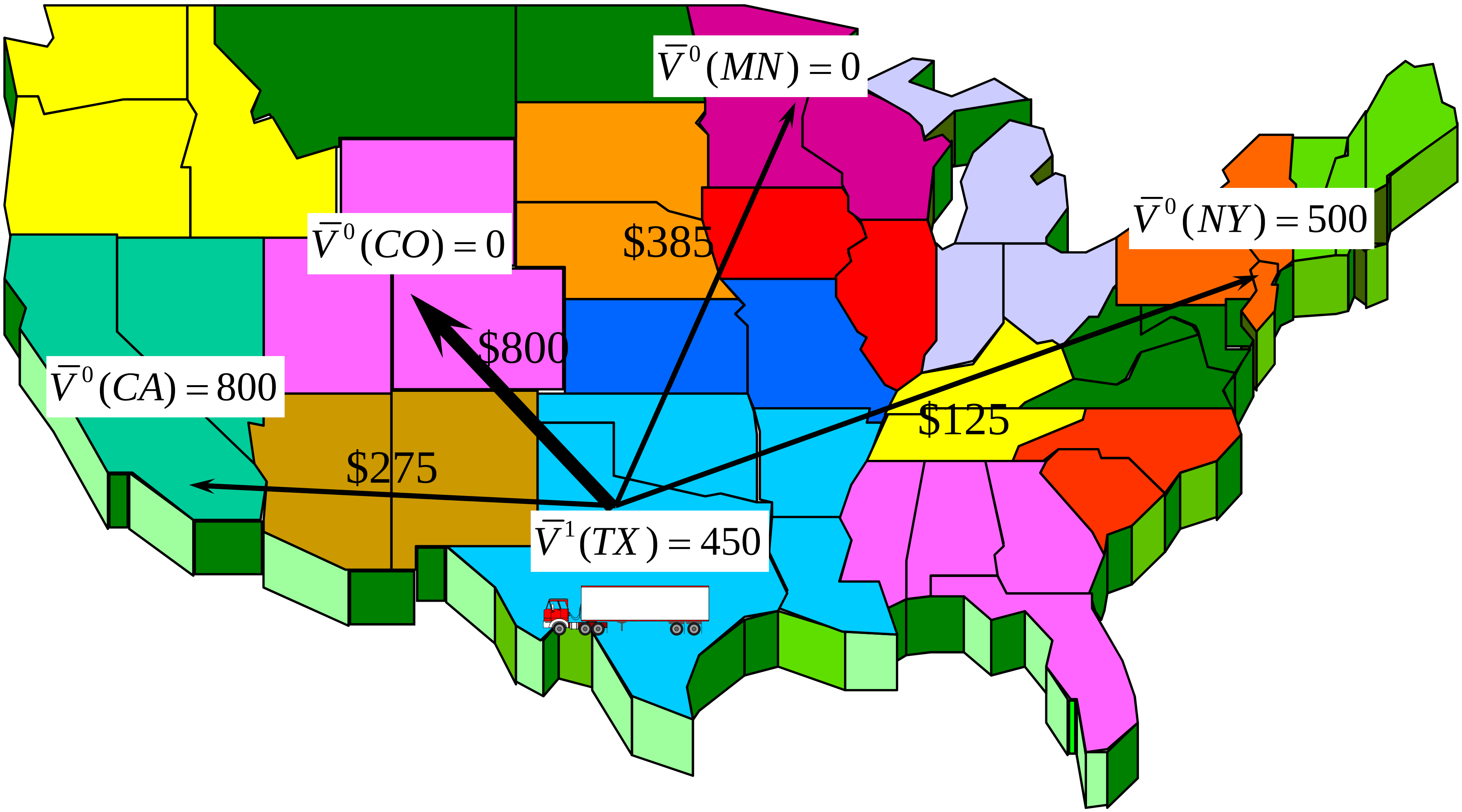
Value function approximations

- Back in TX, we repeat the process, observing a different set of demands.



Value function approximations

- We get a different decision and a new estimate of the value of being in TX



Value function approximations

- Updating the value function:

  Old value:

  $$\bar{V}^1(TX) = \$450$$

  New estimate:

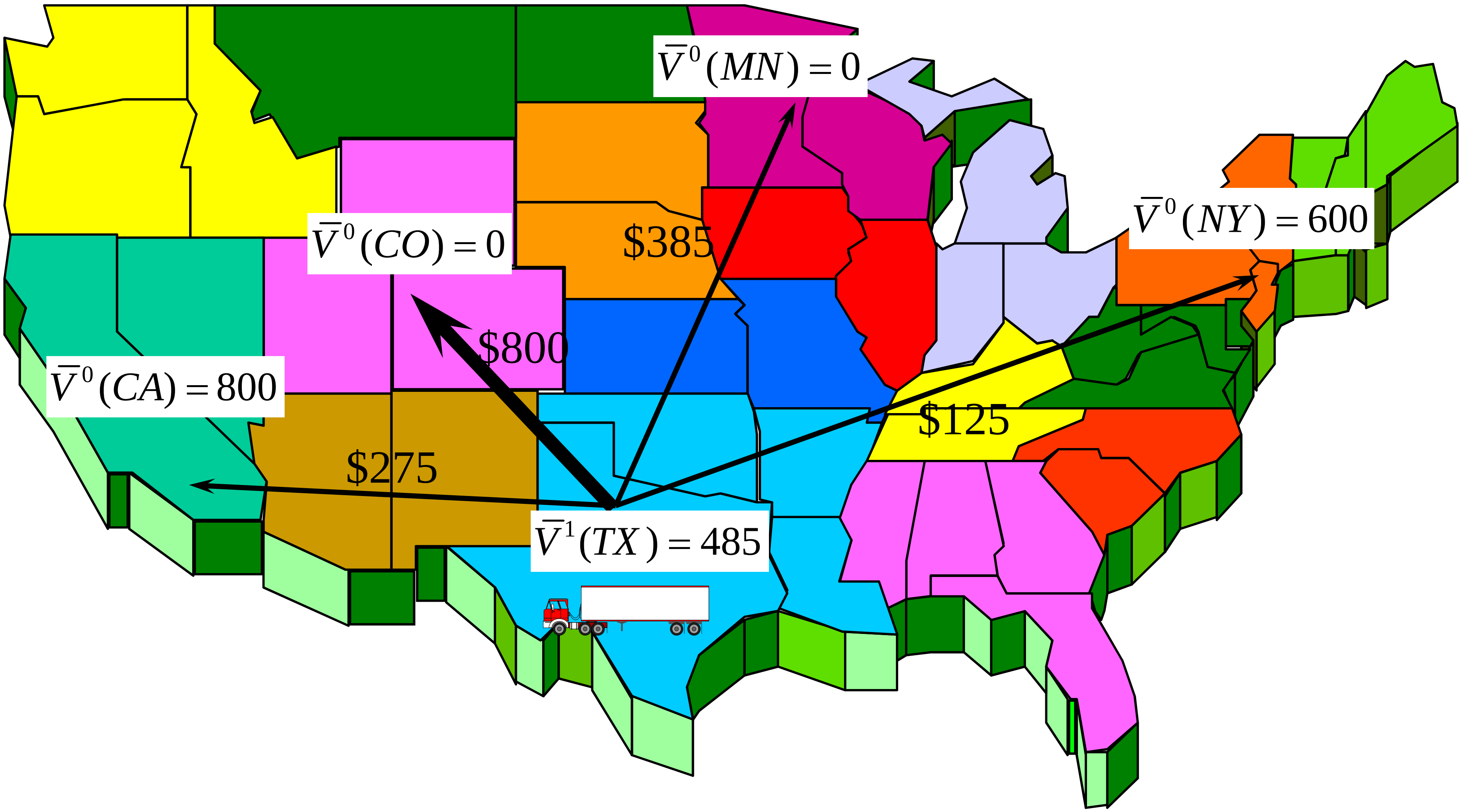
  $$\hat{v}^2(TX) = \$800$$

  How do we merge old with new?

  $$\bar{V}^2(TX) = (1-\alpha)\bar{V}^1(TX) + (\alpha)\hat{v}^2(TX)$$
  $$= (0.90)\$450+(0.10)\$800$$
  $$= \$485$$

Value function approximations

- An updated value of being in TX



Value function approximation

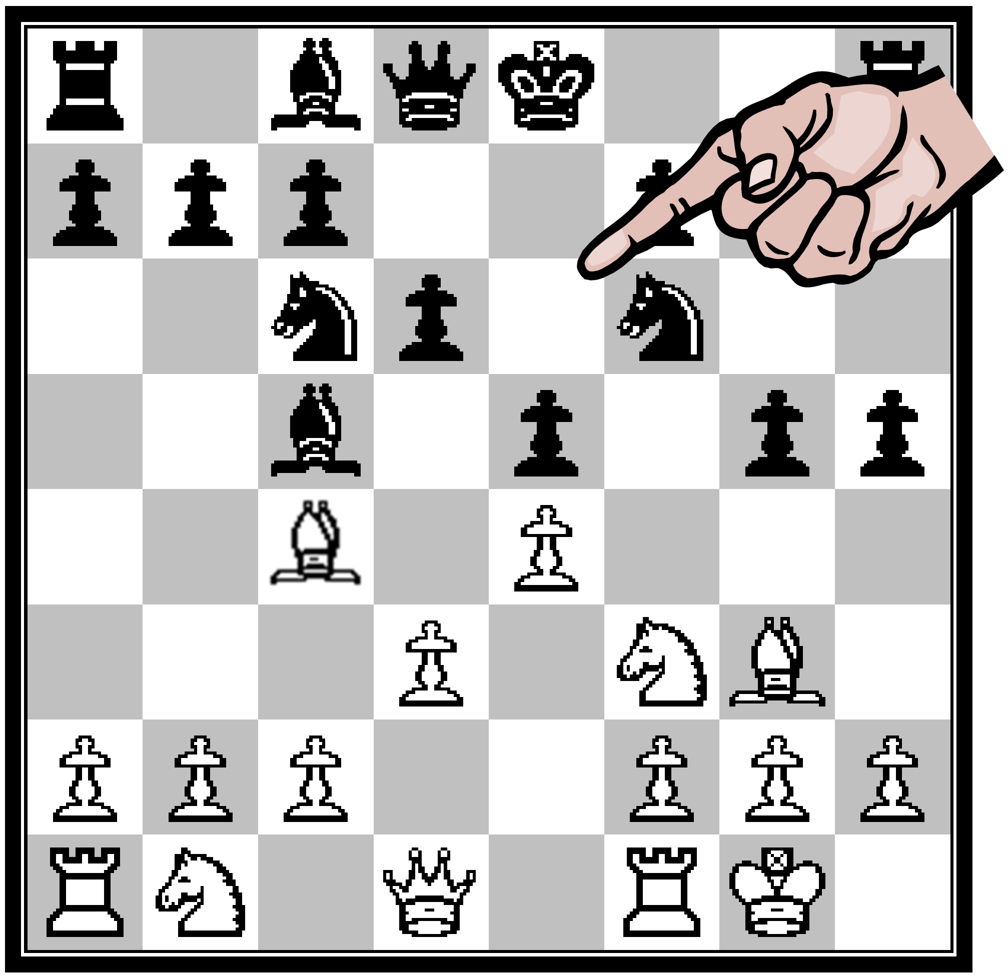
- Notes:
  - » At each step, our truck driver makes a decision based on *previously computed estimates* of the value of being in each location.
  - » Using these value function approximations, decisions which capture (approximately) downstream impacts become quite easy.
  - » But you have to trust the quality of your approximation.
  - » There is an entire field of research that focuses on how to approximate value functions known as "approximate dynamic programming."

Lecture outline

- What is a policy?
  - Policy function approximations (PFAs)
  - Cost function approximations (CFAs)
  - Value function approximations (VFAs)
  - Lookahead policies
- Finding good policies
- Optimizing continuous parameters

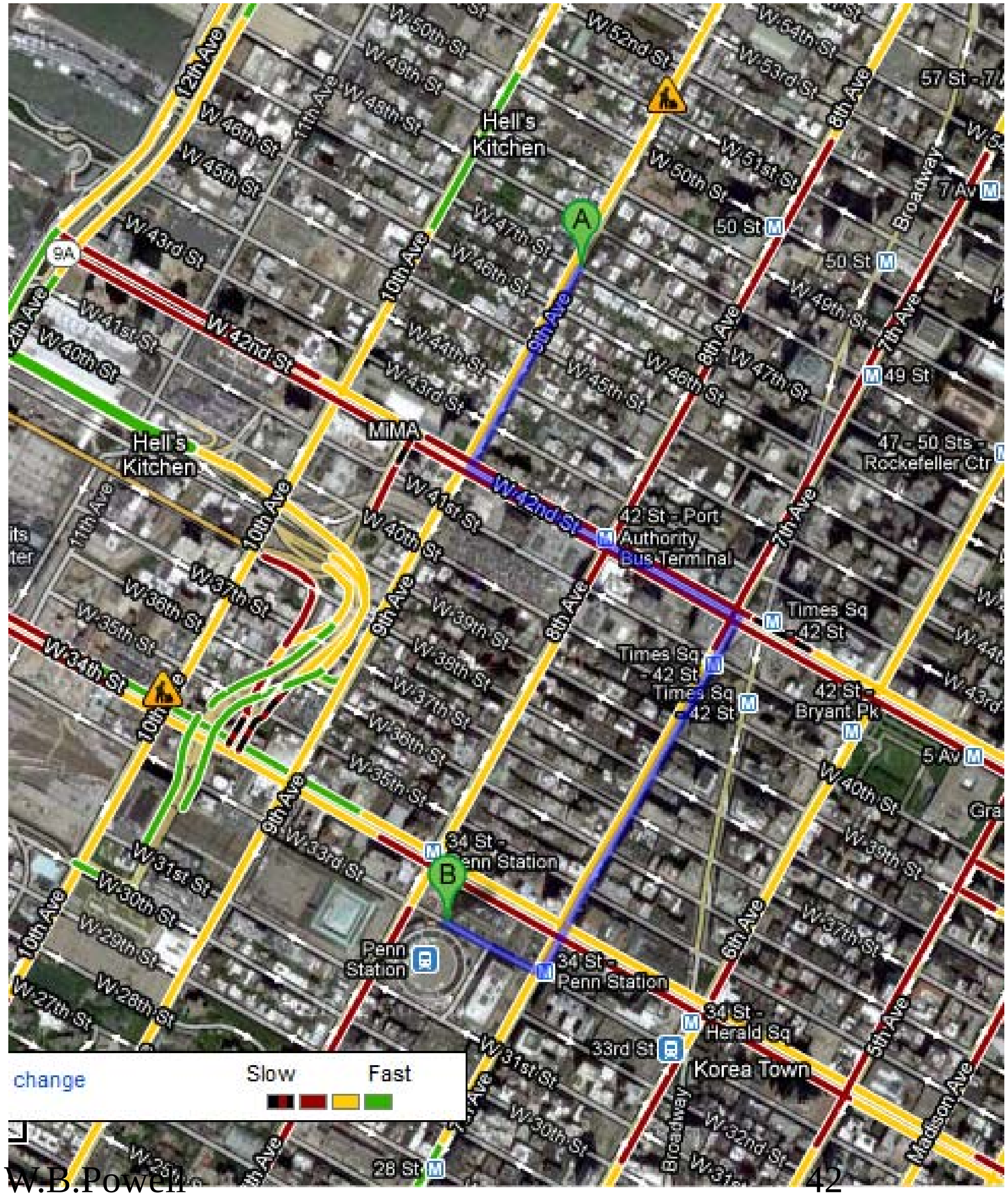
Lookahead policies

- It is common to “peek” into the future:



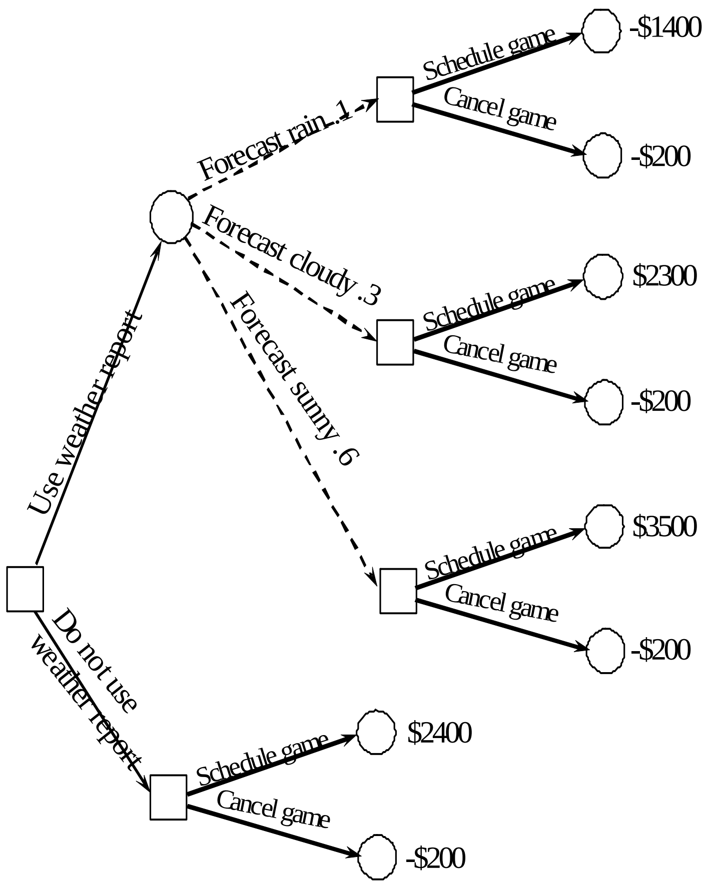
Lookahead policies

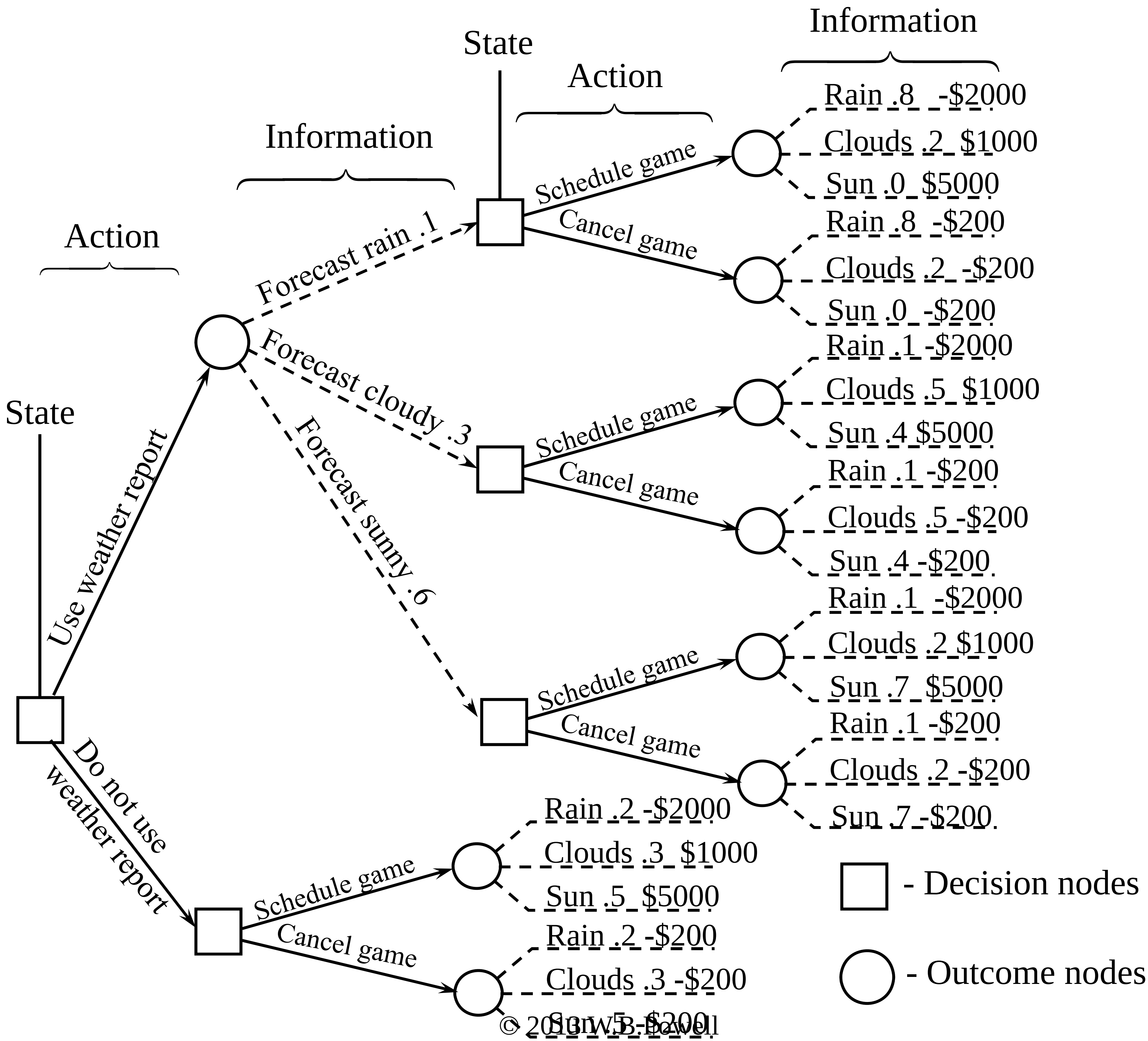
- Shortest path problems
  - » Solve shortest path to destination to figure out the next step. We solve the shortest path using a point estimate of the future.
  - » As car advances, Google updates traffic estimations (or you may react to traffic as you see it).
  - » As the situation changes, we recalculate the shortest path to find an updated route.



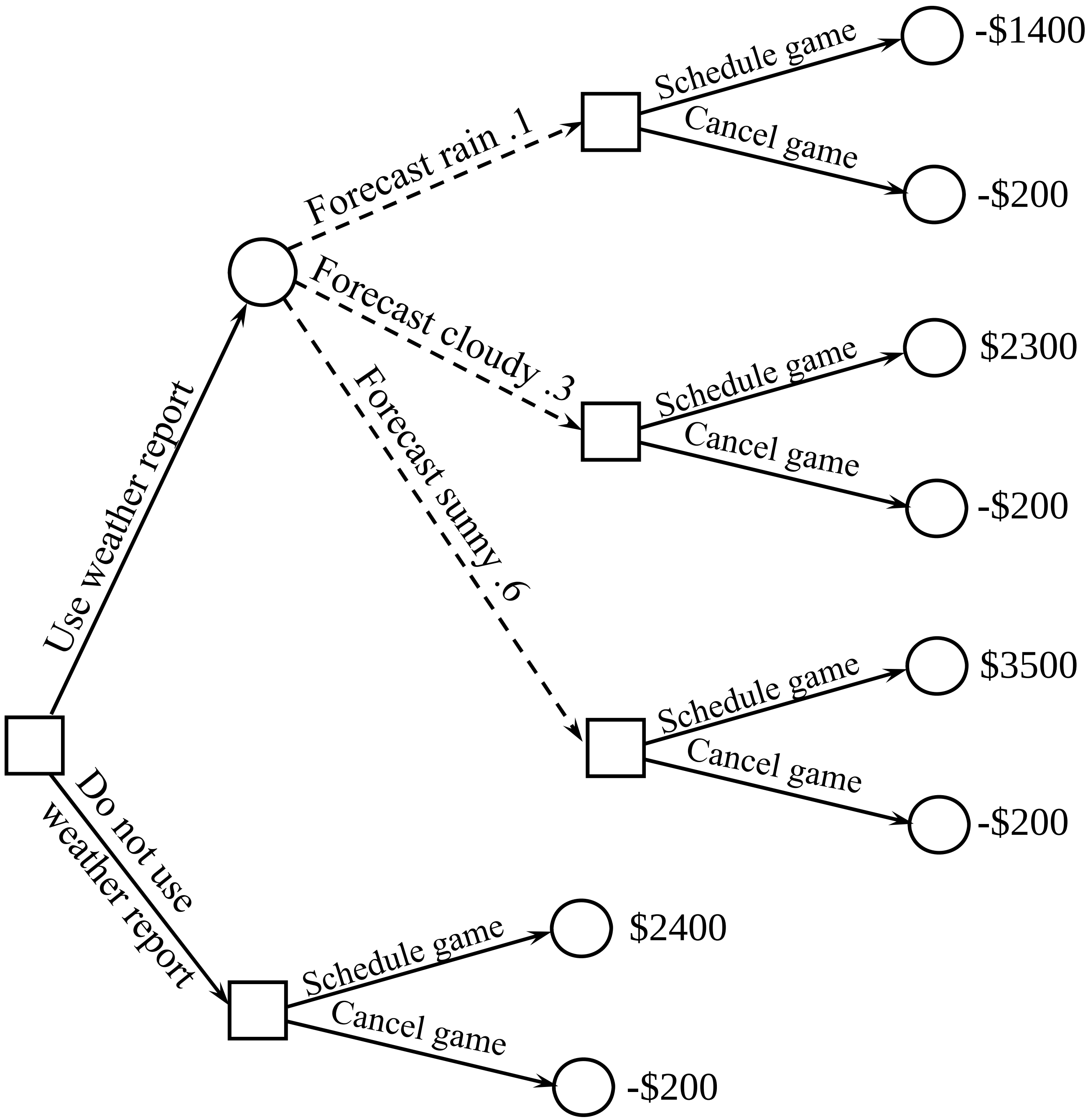
Lookahead policies

- Decision trees
  - » A form of lookup table representation
    - Square nodes – Make a decision
    - Circles – Outcome nodes
      - – Represents state-action pairs
  - » Solving decision trees means finding the value at each outcome node.

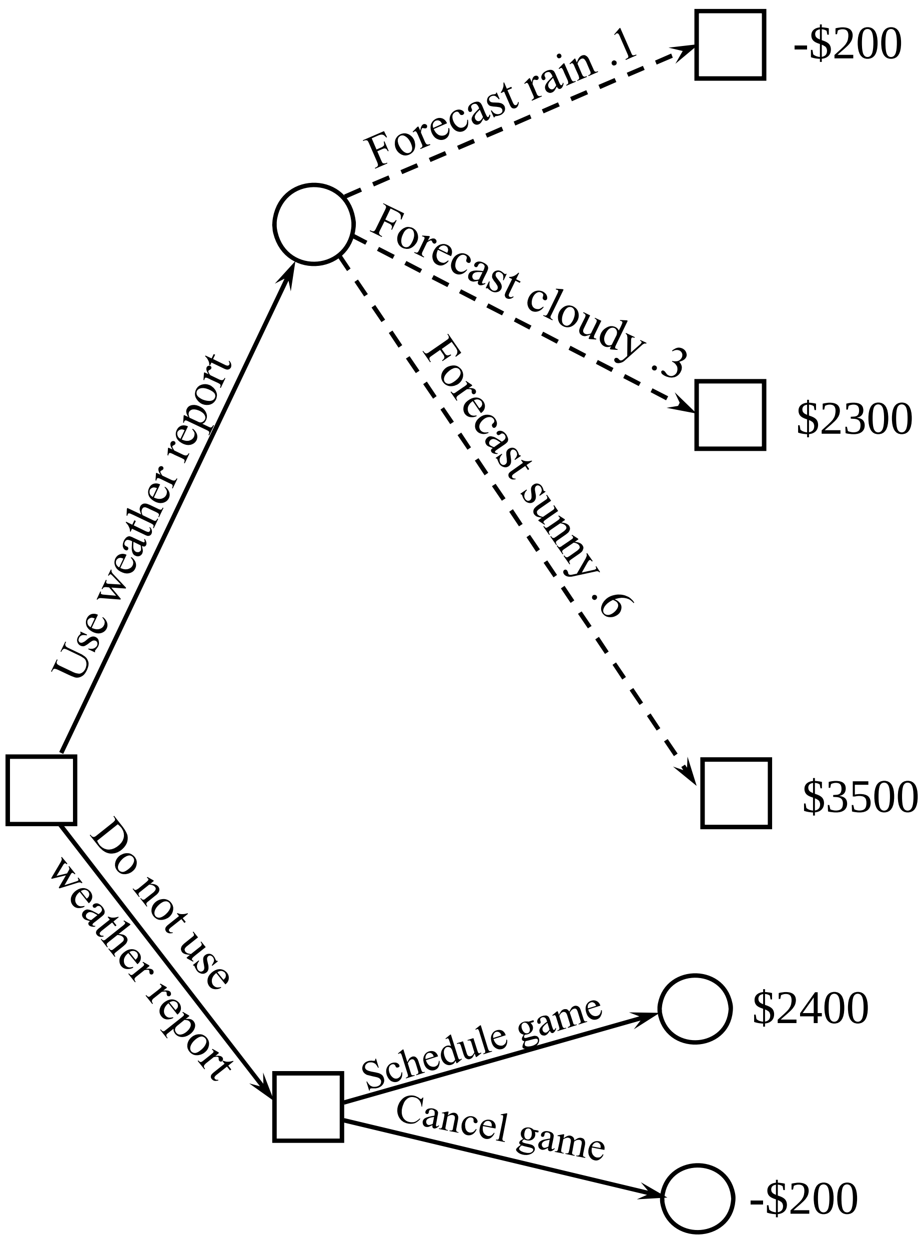


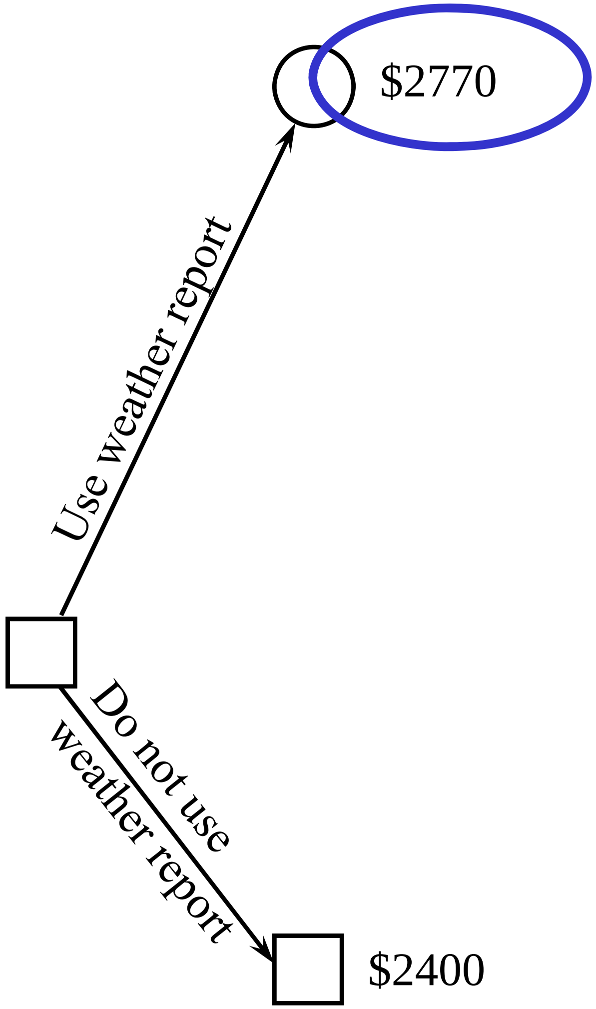


State
Action
Information
State
Action
Information
Use weather report
Do not use weather report
Forecast rain .1
Forecast cloudy .3
Forecast sunny .6
Schedule game
Cancel game
Rain .8 -$2000
Clouds .2 $1000
Sun .0 $5000
Rain .8 -$200
Clouds .2 -$200
Sun .0 -$200
Schedule game
Cancel game
Rain .1 -$2000
Clouds .5 $1000
Sun .4 $5000
Rain .1 -$200
Clouds .5 -$200
Sun .4 -$200
Schedule game
Cancel game
Rain .1 -$2000
Clouds .2 $1000
Sun .7 $5000
Rain .1 -$200
Clouds .2 -$200
Sun .7 -$200
Schedule game
Cancel game
Rain .2 -$2000
Clouds .3 $1000
Sun .5 $5000
Rain .2 -$200
Clouds .3 -$200
Sun .5 -$200
- Decision nodes
- Outcome nodes



Use weather report
Forecast rain .1
Schedule game
-$1400
Cancel game
-$200
Forecast cloudy .3
Schedule game
$2300
Cancel game
-$200
Forecast sunny .6
Schedule game
$3500
Cancel game
-$200
Do not use weather report
Schedule game
$2400
Cancel game
-$200


Forecast rain .1
-$200
Forecast cloudy .3
$2300
Forecast sunny .6
$3500
Use weather report
Do not use weather report
Schedule game
$2400
Cancel game
-$200



Approximate value of being in this state

After rolling back, we use the value at each node to make the best decision. This value captures the effect of all future information and decisions.

Lookahead policies

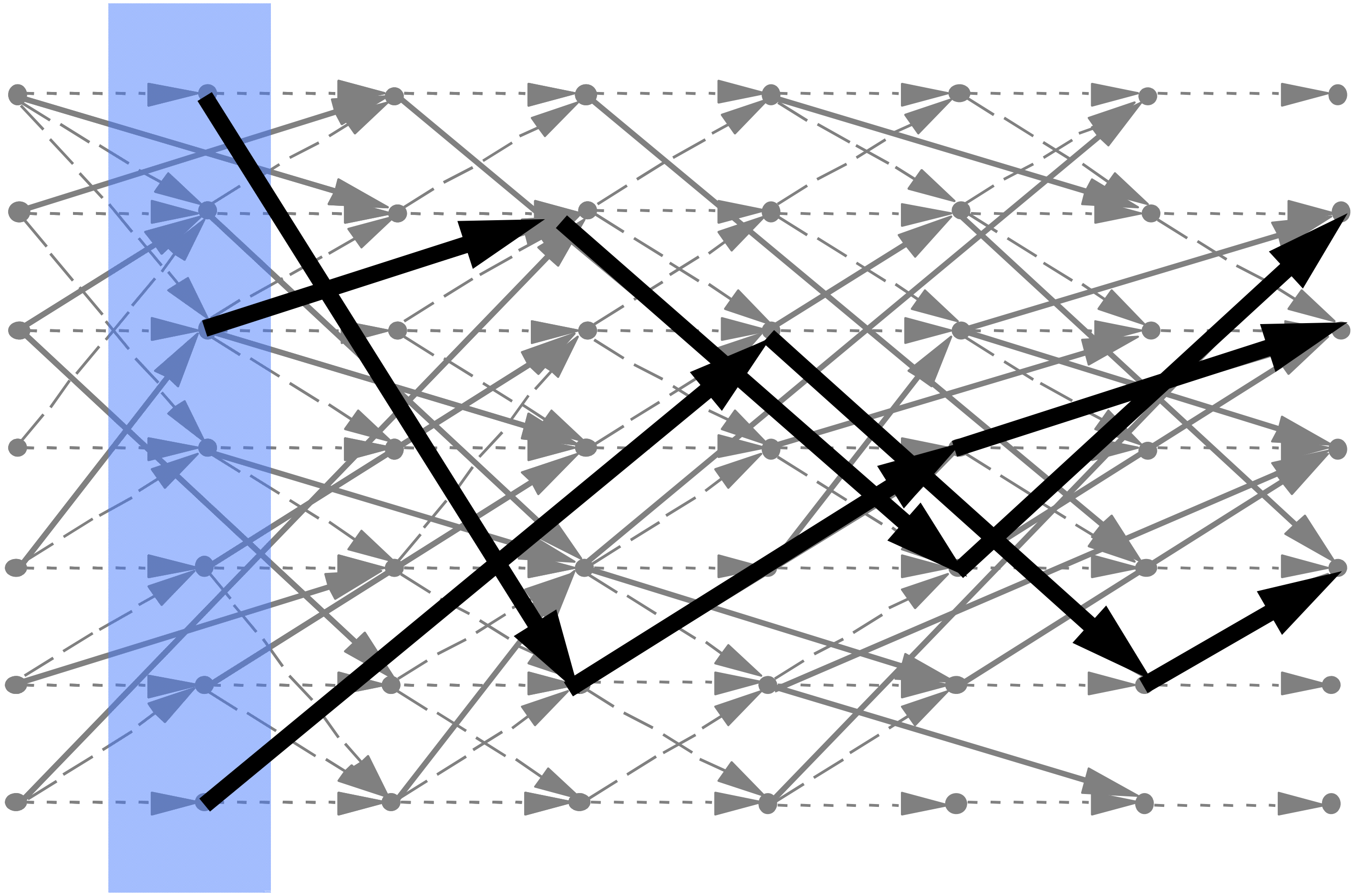
- Sometimes, our lookahead policy involves solving a linear program over multiple time periods:

$$X(S_t) = \arg\min_{x_t, x_{t+1}, \ldots, x_{t+T}} \sum_i c_{ti} x_{ti} + \underbrace{\sum_{t'=t+1}^{T} \sum_i c_{t'i} x_{t'i}}_{\text{Optimizing into the future}}$$

  - » This strategy requires that we pretend we know everything that will happen in the future, and then optimize deterministically.

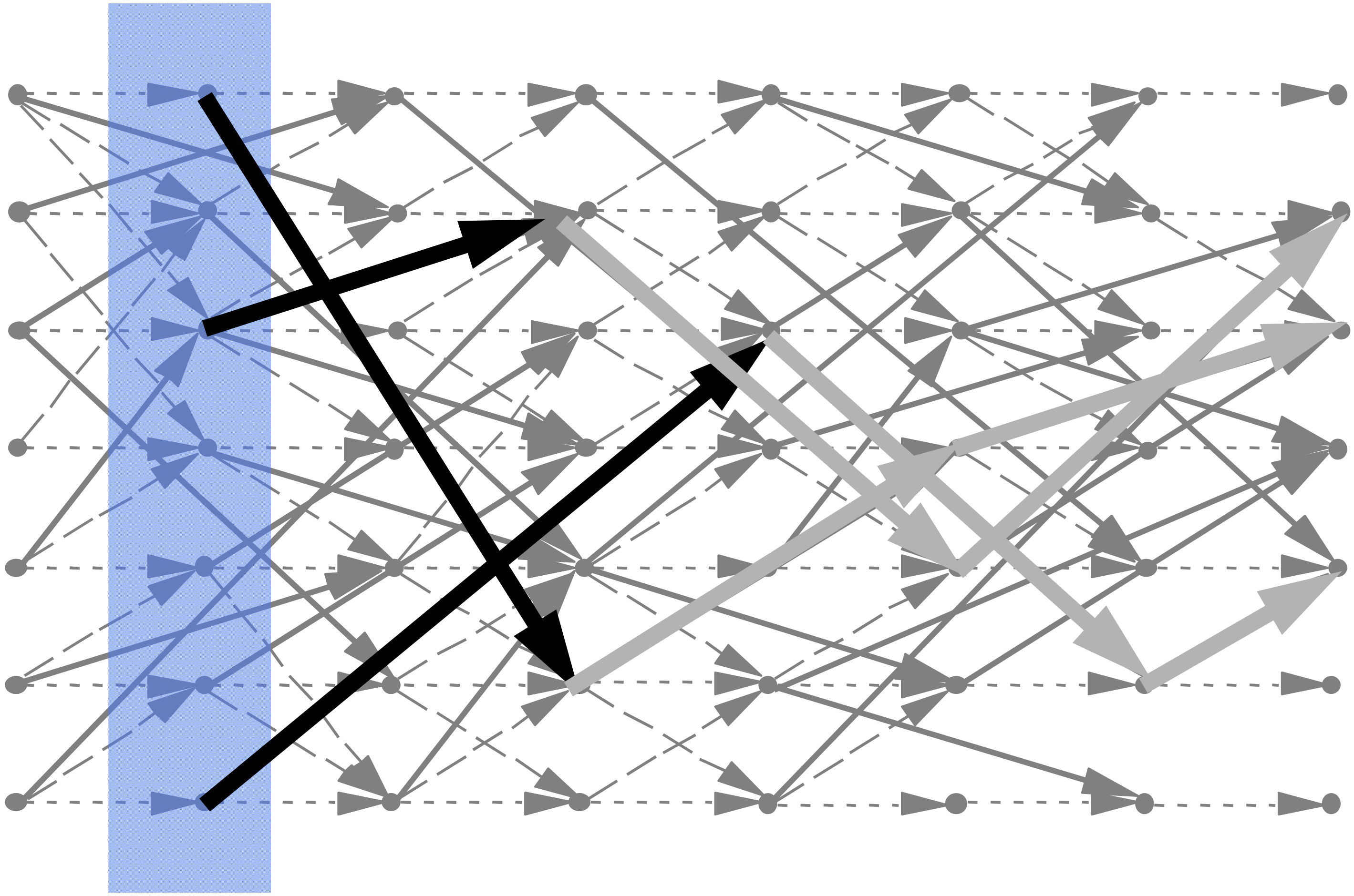
Lookahead policies

- We can handle vector-valued decisions by solving linear (or integer) programs over a horizon.



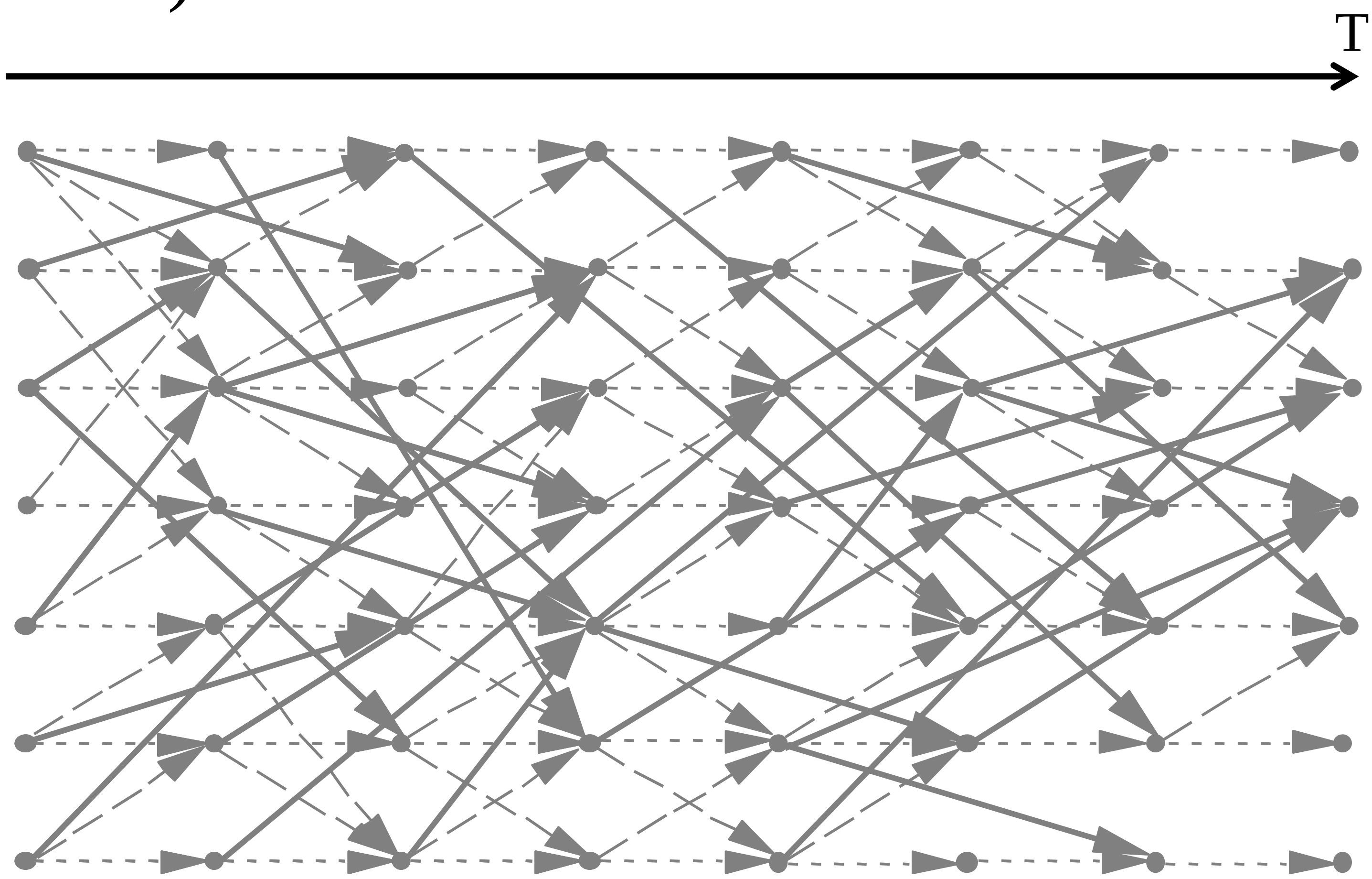
Lookahead policies

- We optimize into the future, but then ignore the decisions that would not be implemented until later.



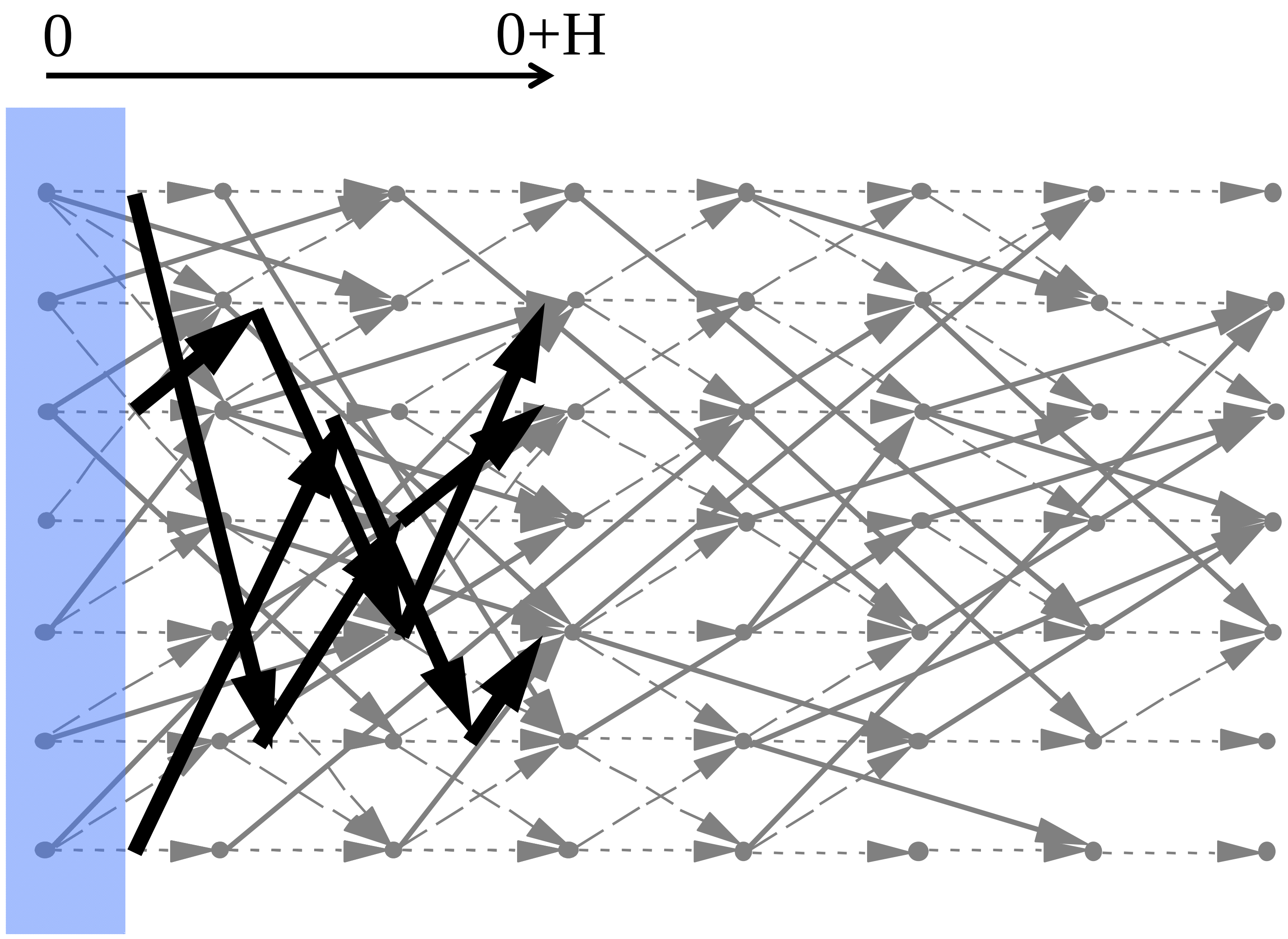
Lookahead policies

- Assume that this is the full model (over T time periods)



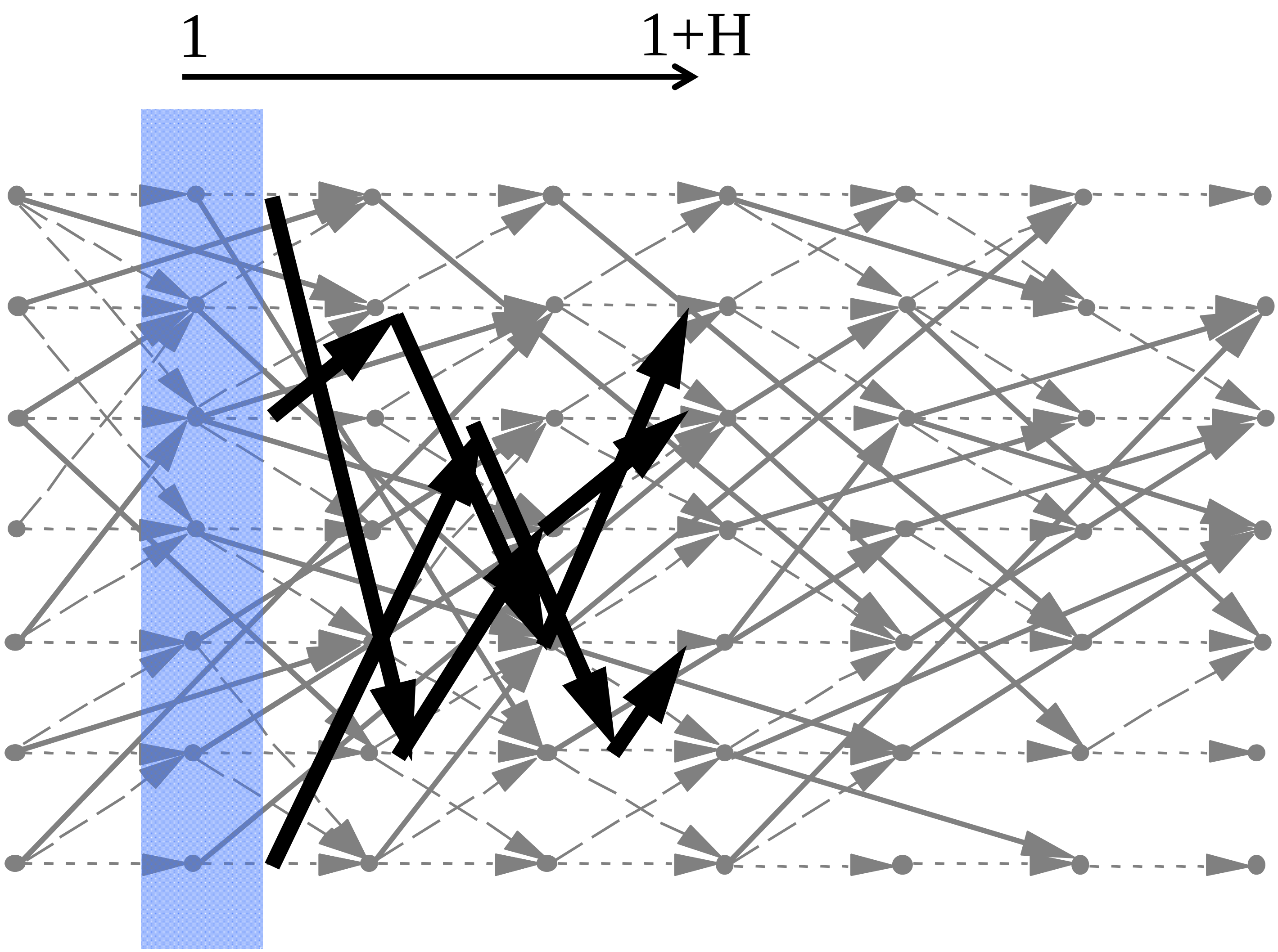
Lookahead policies

- But we solve a smaller lookahead model (from t to t+H)



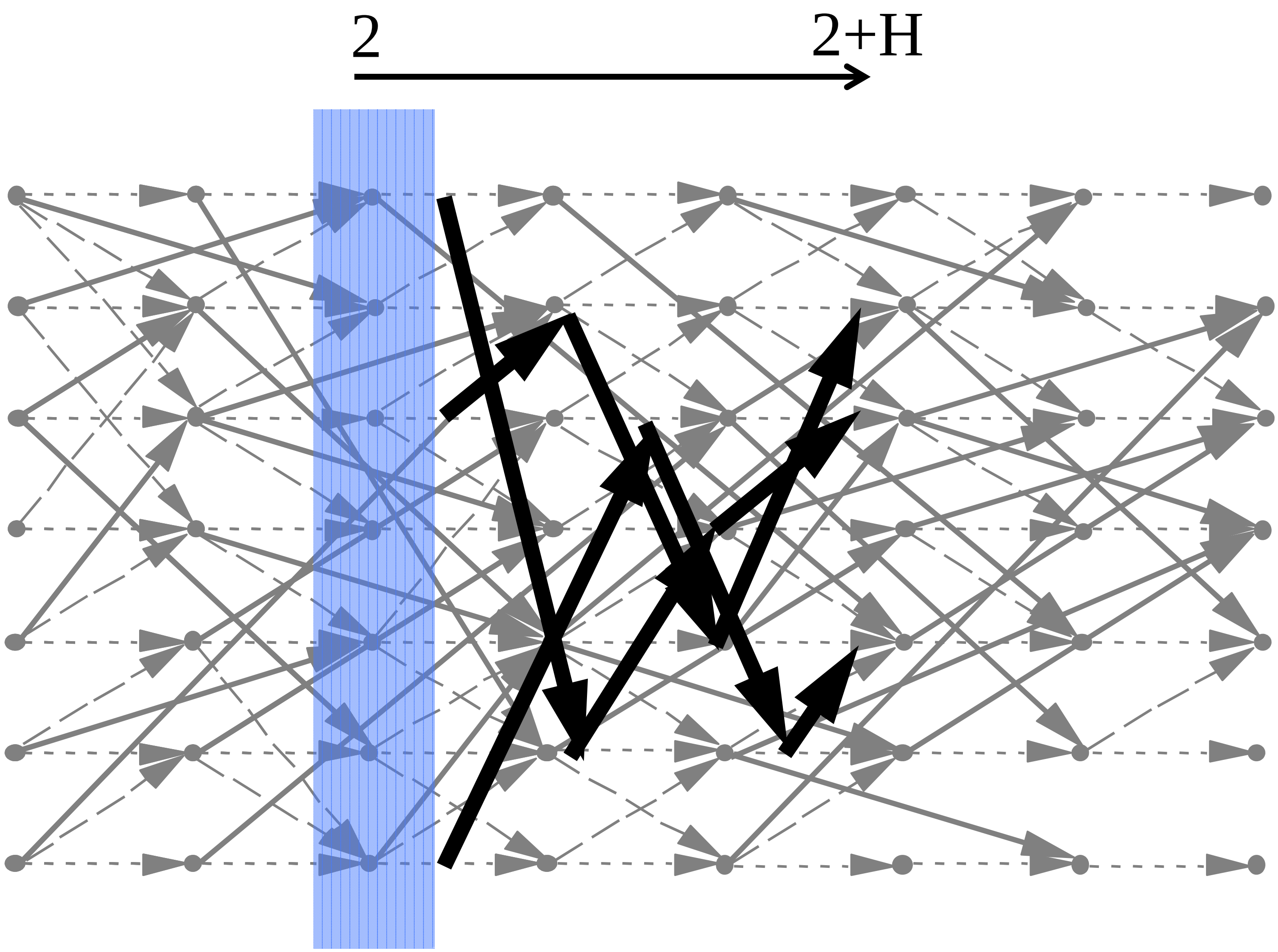
Lookahead policies

- Following a lookahead policy



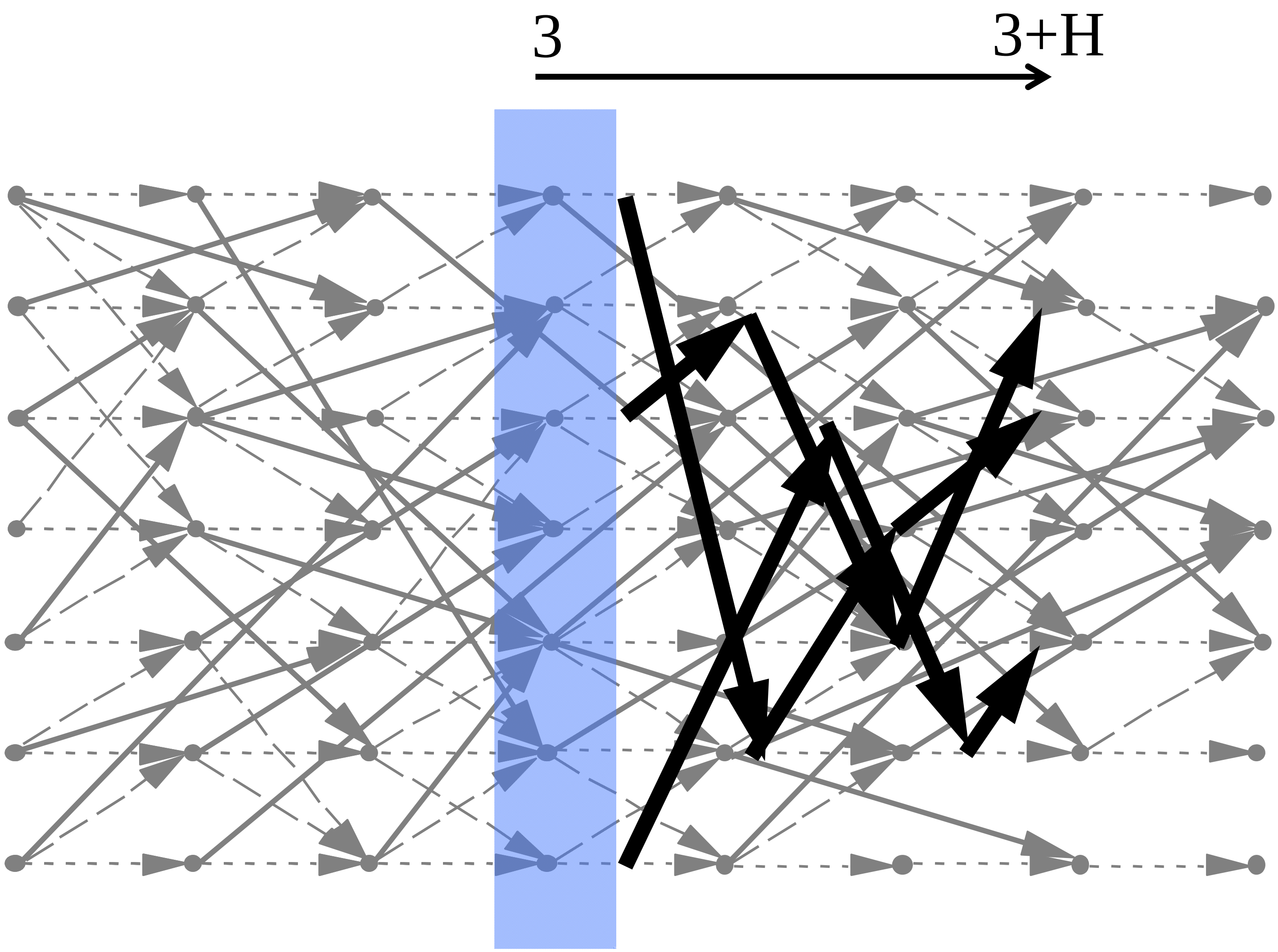
Lookahead policies

- … which rolls forward in time.



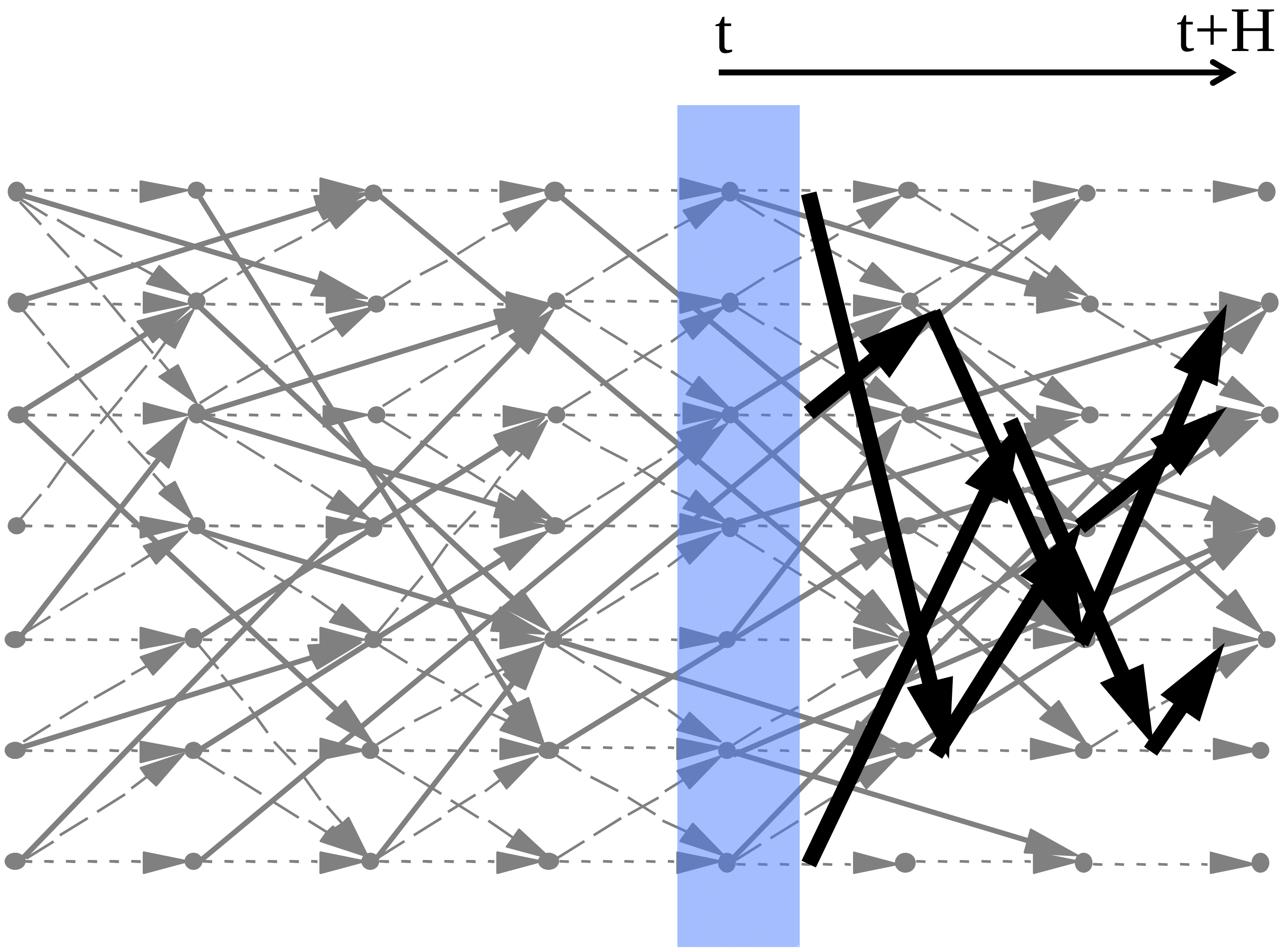
Lookahead policies

- … which rolls forward in time.



Lookahead policies

- … which rolls forward in time.



Lookahead policies

- Notes on lookahead policies:
  - » They construct the value of being in a state in the future "on the fly," which allows the calculation to take into account many other variables (e.g. the status of the entire chess board).
  - » Lookahead policies are brute force – searching the tree of all possible outcomes and decisions can get expensive. Compute times grow exponentially with the length of the horizon.
  - » But, they are simple to understand.

Lecture outline

- What is a policy?
  - Myopic cost function approximations
  - Lookahead policies
  - Policies based on value function approximations
  - Policy function approximations
- Finding good policies
- Optimizing continuous parameters

Finding good policies

- The process of searching for a good policy depends on the nature of the policy space:

  » 1) Small number of discrete policies

  » 2) Single, continuous parameter

  » 3) Two or more continuous parameters

  » 4) Finding the best of a subset

  » …. other more complicated stuff.

Finding good policies

- Evaluating policies
  - » We learned we can write our objective function as

$$\min_{\pi \in \Pi} E\left\{\sum_{t}\sum_{ij} C\left(S_t, X^{\pi}(S_t)\right)\right\}$$

- We now have to deal with:
  - » How do we design a policy?
    - Choose the best type of policy (PFA, CFA, VFA, Look-ahead, hybrid)
    - Tune the parameters of the policy
  - » How do we search for the best policy?

Finding good policies

- Finding the best of two policies
  - » We simulate a policy N times and take an average:

$$\bar{F}^{\pi} = \frac{1}{N}\sum_{n=1}^{N} F^{\pi}(\omega^n)$$

  - » If we simulate policies π_1 and π_2, we would like to conclude that π_1 is better than π_2 if

$$\bar{F}^{\pi_1} > \bar{F}^{\pi_2}$$

  - » How big should N be (or, is N big enough)?
    - Have to compute confidence intervals. The variance of an estimate of the value of a policy is given by the usual formula:

$$s^{2,\pi} = \frac{1}{N}\left(\frac{1}{N-1}\sum_{n=1}^{N}\left(F^{\pi}(\omega^n) - \bar{F}^{\pi}\right)^2\right)$$

Finding good policies

- Now construct confidence interval for the difference:
  - $\bar{\delta} = \bar{F}^{\pi_1} - \bar{F}^{\pi_2} =$ Point estimate of difference
  - Assume that the estimates of the value of each policy were performed independently. The variance of the difference is then
    - $s_\delta^2 = s^{2,\pi_1} + s^{2,\pi_2}$
  - Now construct a confidence interval around the difference:
    - $\left(\bar{\delta} - z_{\alpha/2} s_\delta, \bar{\delta} + z_{\alpha/2} s_\delta\right)$

Finding good policies

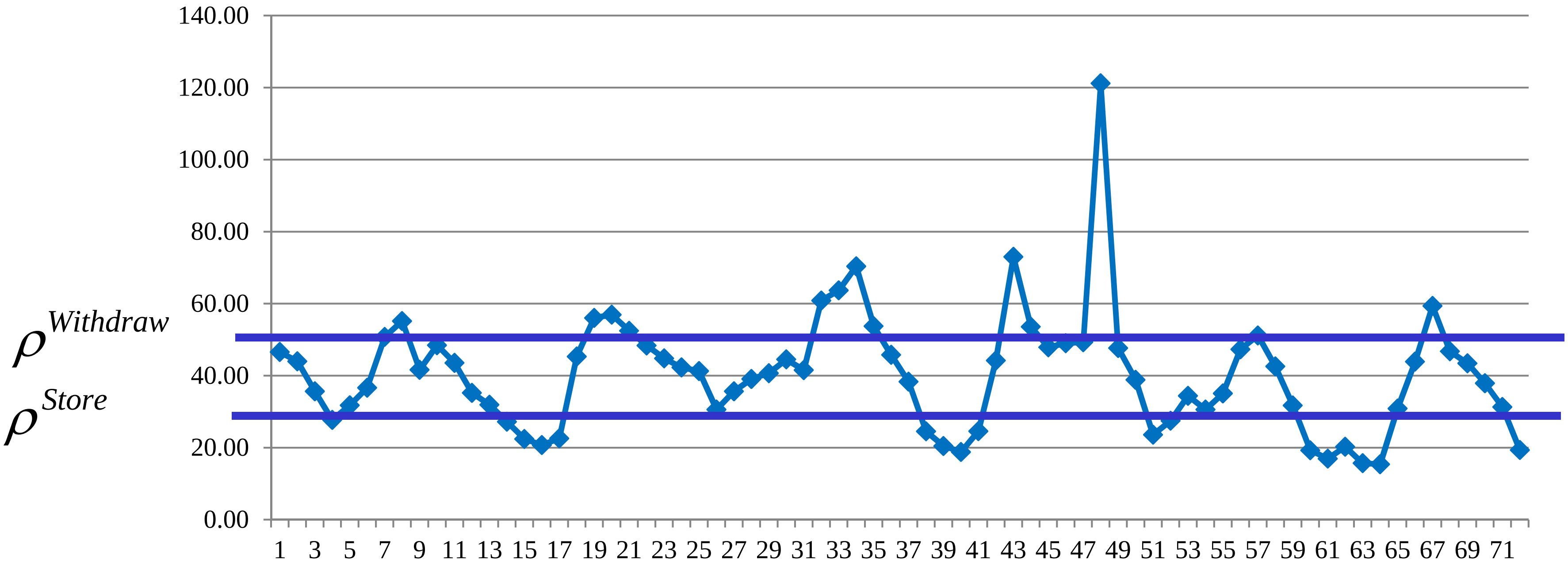
- Better way:
  - » Evaluate each policy using the same set of random variables (the same sample path)
  - » Compute a sample realization of the difference:
    - $$\delta(\omega) = F^{\pi_1}(\omega) - F^{\pi_2}(\omega)$$
      $$\bar{\delta} = \frac{1}{N}\sum_{n=1}^{N}\delta(\omega^n)$$
      $$s_\delta^2 = \frac{1}{N}\left(\frac{1}{N-1}\sum_{n=1}^{N}\left(\delta(\omega^n) - \bar{\delta}\right)^2\right)$$
    - Now compute confidence interval in the usual way.

Finding good policies

- Notes:
  - » First method requires 2N simulations
  - » Second method requires N simulations, but they have to be coordinated (e.g. run in parallel).
  - » There is another method which further minimizes how many simulations are needed. Will describe this later in the course.

Finding good policies

- We had to design a *simple*, *implementable* policy that did not cheat!

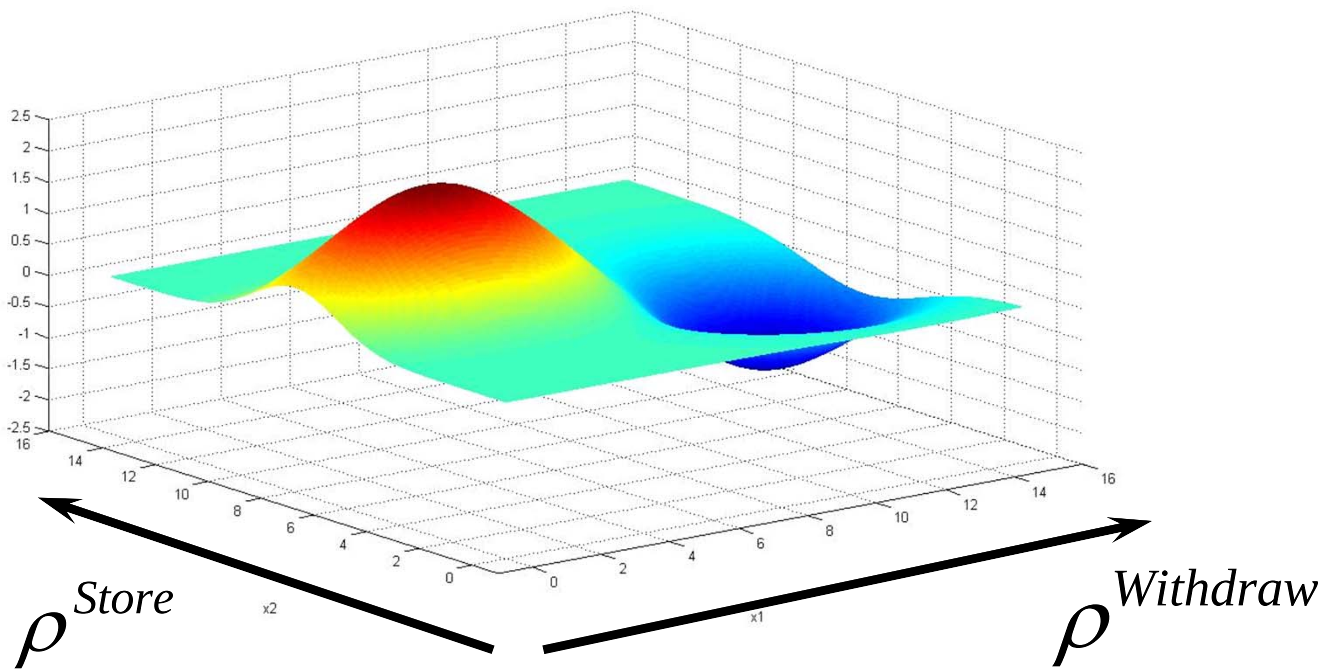


- We need to search for the best values of the parameters ρ^{Store} and $\rho^{Withdraw}$.

Finding good policies

- Finding the best policy ("policy search")
  - » Let $X^{\pi}(S_t \mid \rho^{store}, \rho^{withdraw})$ be the "policy" that chooses the actions.
  - » We wish to maximize the function

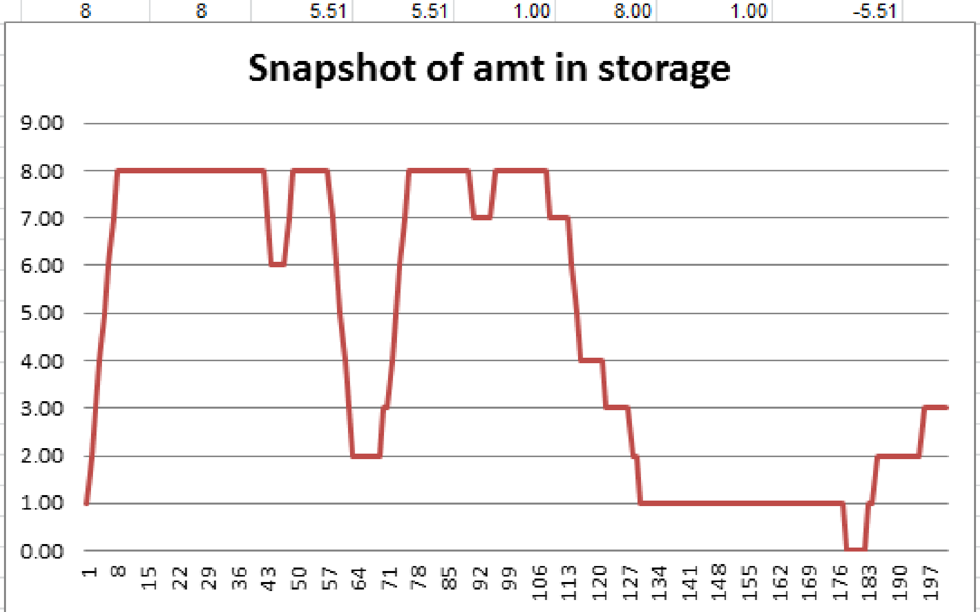
$$\min_{\rho} \mathbb{E}F(\rho, W) = \mathbb{E}\sum_{t=0}^{T} \gamma^t C\left(S_t, X^{\pi}(S_t \mid \rho)\right)$$

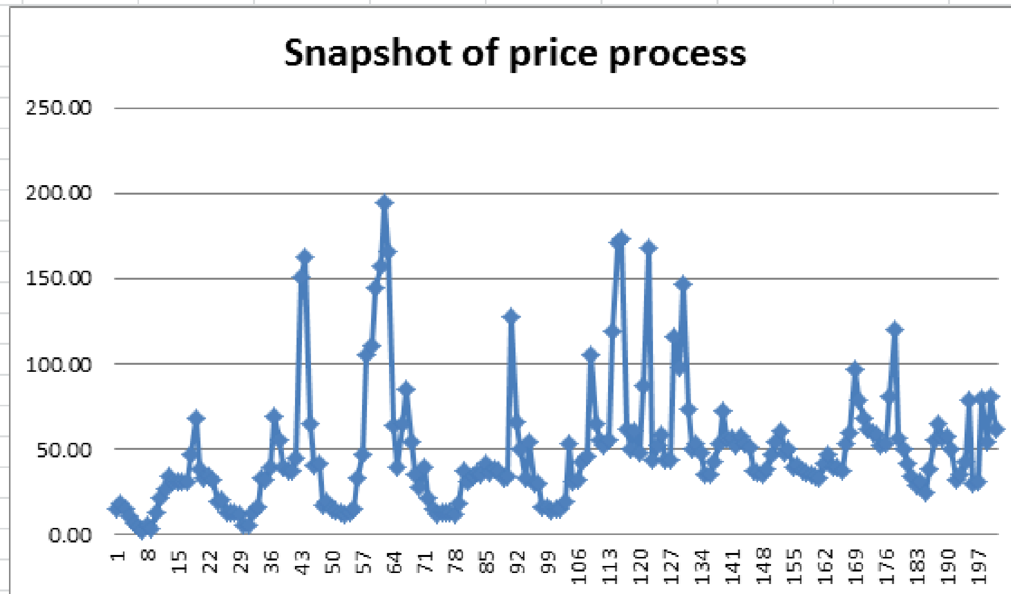


Finding good policies

- Illustration of policy search

loss	0.70				Battery siz	8.00	Total profit/h	0.00	Average	Variance	Sampled from N(50,sqrt(3400))				
Smoothing	1.00		Buy		30.00	Amt in						Mean=30			Mean=
			Sell		100.00	storage	Bought/sold	Revenue		Mean	30			50	
						0.00				Std	58.30952			58.30952	
1/1/05	1	1	14.88	14.88	1.00	1.00	1.00	-14.88	14.88		-35.898	-35.90		46.81959	46
1/1/05	2	2	18.47	18.47	1.00	2.00	1.00	-18.47	16.675	6.44405	42.95748	3.52972	3109.097	-45.4948	0.662
1/1/05	3	3	14.43	14.43	1.00	3.00	1.00	-14.43	15.92667	4.902033	-32.3674	-8.43599	1984.083	108.735	36.68
1/1/05	4	4	10.58	10.58	1.00	4.00	1.00	-10.58	14.59	10.41473	62.51413	9.301541	2581.202	137.7898	61.96
1/1/05	5	5	6.54	6.54	1.00	5.00	1.00	-6.54	12.98	20.77155	20.13827	11.46889	1959.388	68.34944	63.2
1/1/05	6	6	3.86	3.86	1.00	6.00	1.00	-3.86	11.46	30.47964	-13.7702	7.262367	1673.68	-6.94179	51.54
1/1/05	7	7	2.54	2.54	1.00	7.00	1.00	-2.54	10.18571	36.76633	-1.67054	5.986237	1406.133	-33.0701	39.45
1/1/05	8	8	5.51	5.51	1.00	8.00	1.00	-5.51	9.60125	34.24678	4.045646	5.743663	1205.727	159.6181	54.47
1/1/05															
1/1/05															
1/1/05															
1/1/05															
1/1/05															
1/1/05															
1/1/05															
1/1/05															
1/1/05															
1/1/05															
1/1/05															
1/1/05															
1/1/05															
1/1/05															
1/1/05															
1/1/05															
1/2/05															
1/2/05															
1/2/05															
1/2/05	28	4	12.80	12.80	1.00	8.00	0.00	0.00	22.42536	221.7175	70.25475	23.11226	2006.045	64.34963	73.63
1/2/05	29	5	11.25	11.25	1.00	8.00	0.00	0.00	22.04	218.1056	67.63873	24.64767	2004.695	27.51886	72.0





Finding good policies

- SMART-Solar
  - » See http://energysystems.princeton.edu/smartfamily.htm

SMART-Solar: Co-optimization of solar and storage

Regular Charge @ $	**56**
Regular Discharge @ $	**71**
Cold Charge @ $	**95**
Cold Discharge @ $	**115**
Hot Charge @ $	**100**
Hot Discharge @ $	**118**
Grid Charge @ $	**60**
Grid Discharge @ $	**120**
Temp (F)	
Hot	75
Cold	20
Inverter loss	**20%**
Total battery size (MWh)	**8**
Rate (MW)	**4**
Upper boundary	**90%**
Lower boundary	**10%**
Interconnection approval (MW)	8.5
Battery size for ES (MWh)	6
Battery size for FR (MWh)	2
Battery in use	**Yes**
Policy/CPLEX	**Policy**

Parameters that control the behavior of the policy.

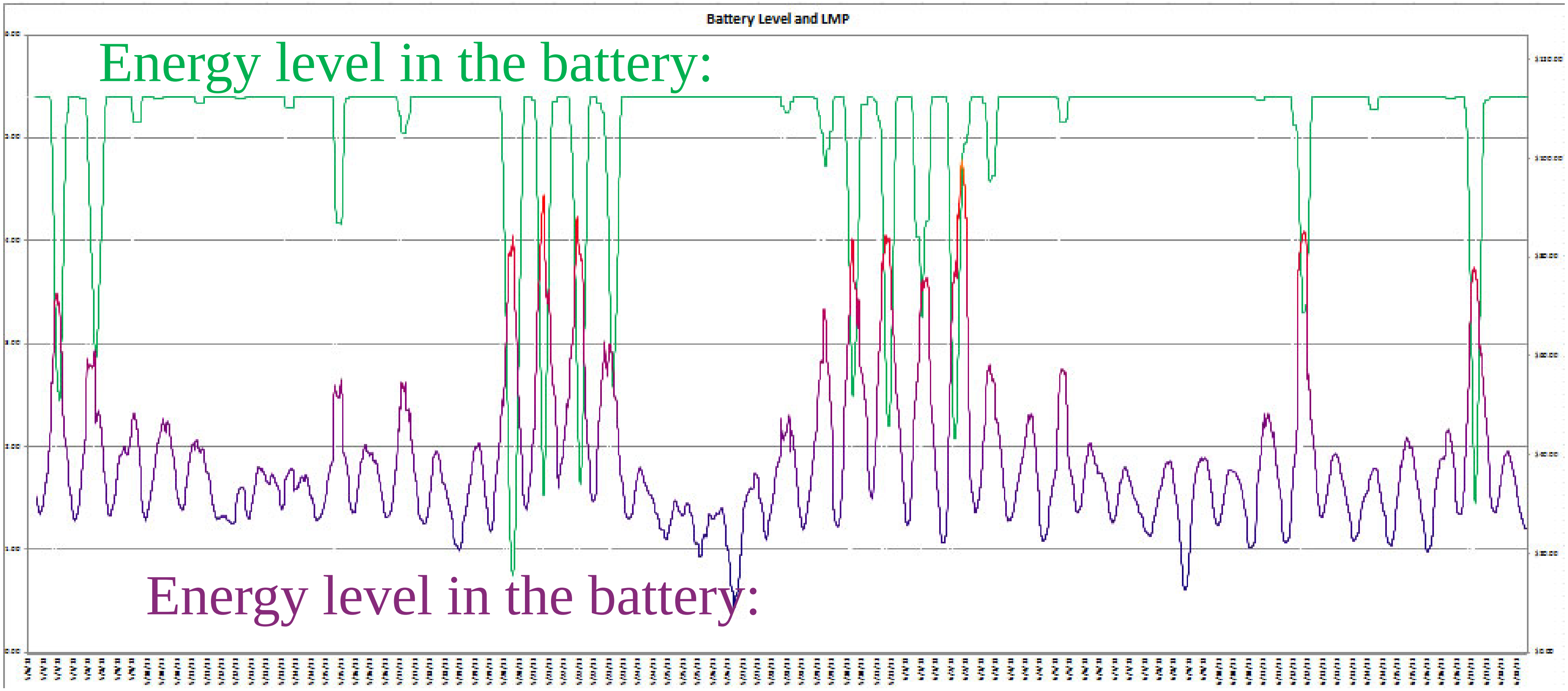
Payback Years	5.58
Net Revenue 5yr	($867,078.29)
Net Revenue 10yr	$4,229,776.26
Net Revenue 15yr	$8,345,079.40
Net Revenue 20yr	$11,578,315.69
NPV 5yr	($2,533,117.84)
NPV 10yr	$985,043.93
NPV 15yr	$3,174,578.03
NPV 20yr	$4,522,756.86
ROI 5yr	-6.14%
ROI 10yr	29.96%
ROI 15yr	59.10%
ROI 20yr	82.00%
Revenue from ES (Annual, Yr 1)	$80,772.68
Revenue from FR (Annual, Yr 1)	$1,012,252.39
Revenue from Solar (Annual, Yr 1)	$346,068.10
Revenue from SREC (Annual, Yr 1)	$543,422.98

Help

All bold values can be changed

Finding good policies

- The control policy determines when the battery is charged or discharged.



» Different values of the charge/discharge prices are simulated to determine which works the best. This is a form of *policy search.*

Lecture outline

- What is a policy?
  - Myopic cost function approximations
  - Lookahead policies
  - Policies based on value function approximations
  - Policy function approximations
- Finding good policies
- Optimizing continuous parameters

Optimizing continuous parameters

- The problem of finding the best policy can be written as a classic stochastic search problem:

$$\min_x E\left\{F(x,W)\right\}$$

  - » … where x is a vector of continuous parameters
  - » W represents all the random variables involved in evaluating a policy.

Optimizing continuous parameters

- We can find *x* using a classic stochastic gradient algorithm
  - » Let

$$F(x) = E\left\{F(x,W)\right\}$$

  - » Now assume that we can find the derivative with respect to each parameter in the policy (not always true). We would write this as

$$g(x,\omega) = \nabla F(x, W(\omega))$$

  - » The stochastic gradient algorithm is then

$$x^n = x^{n-1} - \alpha_{n-1} g(x^{n-1}, \omega^n)$$

  - » We then use x^n for iteration n+1 (for sample path ω^n)

Optimizing continuous parameters

- Notes:
  - » If we are maximizing, we use

$$x^n = x^{n-1} + \alpha_{n-1} g(x^{n-1}, \omega^n)$$

  - » This algorithm is provably convergent if we use a stepsize such as

$$\alpha_n = \frac{\alpha_0}{a+n-1} \qquad n = 1, 2, \ldots$$

  - » Need to choose α_0 to solve the difference in units between the derivative and the parameters.

Optimizing continuous parameters

- Computing a gradient generally requires some insight into the structure of the problem.
- An alternative is to use a finite difference.
- Assume that *x* is a scalar. We can find a gradient using

$$g(x,\omega) = F(x+\delta, W(\omega)) - F(x, W(\omega))$$

  - » Very important: note that we are running the simulation twice using the same sample path.