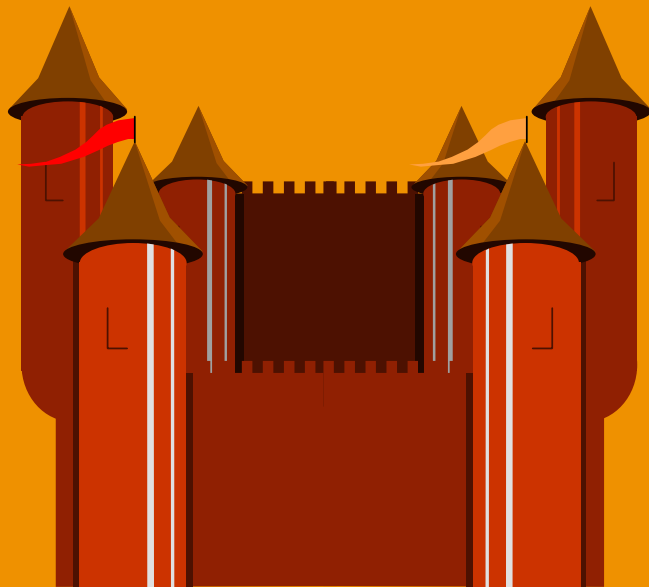


*ORF 544*

*Stochastic Optimization and Learning*

*Spring, 2019*



*Warren Powell*  
*Princeton University*  
*<http://www.castlelab.princeton.edu>*

# Week 12

Direct lookahead



# Direct lookahead

---

## ● 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.

# Direct lookahead

- An optimal policy:

- » The base model:

$$\max_{\pi} E \left\{ \sum_{t=0}^T C \left( S_t, X_t^{\pi} (S_t | \theta) \right) \mid S_0 \right\} \quad \left. \vphantom{\max_{\pi}} \right\} \text{Base model}$$

- » The optimal lookahead policy:

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

- » If you can solve this exactly, you do not need any tunable parameters!



# Direct lookahead

- The ultimate lookahead policy is optimal

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

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

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

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

# Direct lookahead

- Direct lookahead policies

- » With rare exception, direct lookahead policies are trying to solve approximate lookahead models, where we might write the policy as

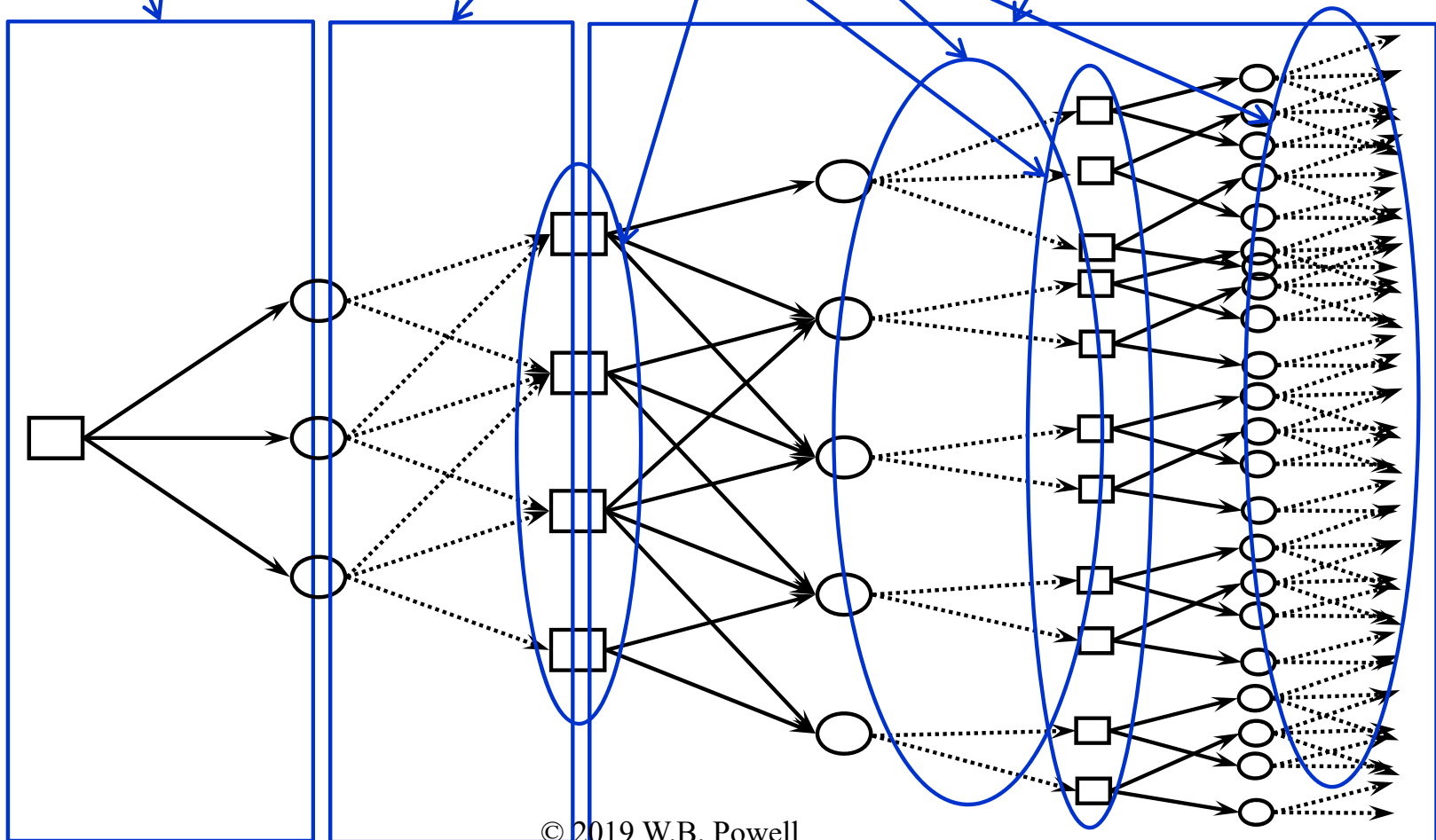
$$X_t^{DLA}(S_t) = \arg \max_{x_t} \left( C(S_t, x_t) + \tilde{\mathbb{E}} \left\{ \max_{\tilde{\pi} \in \tilde{\Pi}} \left\{ \tilde{\mathbb{E}} \sum_{t'=t+1}^{t+H} C(\tilde{S}_{t'}, \tilde{X}_{t'}^{\tilde{\pi}}(\tilde{S}_{t'})) \mid \tilde{S}_{t,t+1} \right\} \mid \tilde{S}_{tt}, x_t \right\} \right)$$

- » The challenge is creating a tractable set of approximations of the lookahead model.

# Direct lookahead

- The ultimate lookahead policy is optimal

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



# Direct lookahead

---

- The lookahead policy

- » This can be any of the four classes of policies, but optimality is less important, and simplicity is more important.

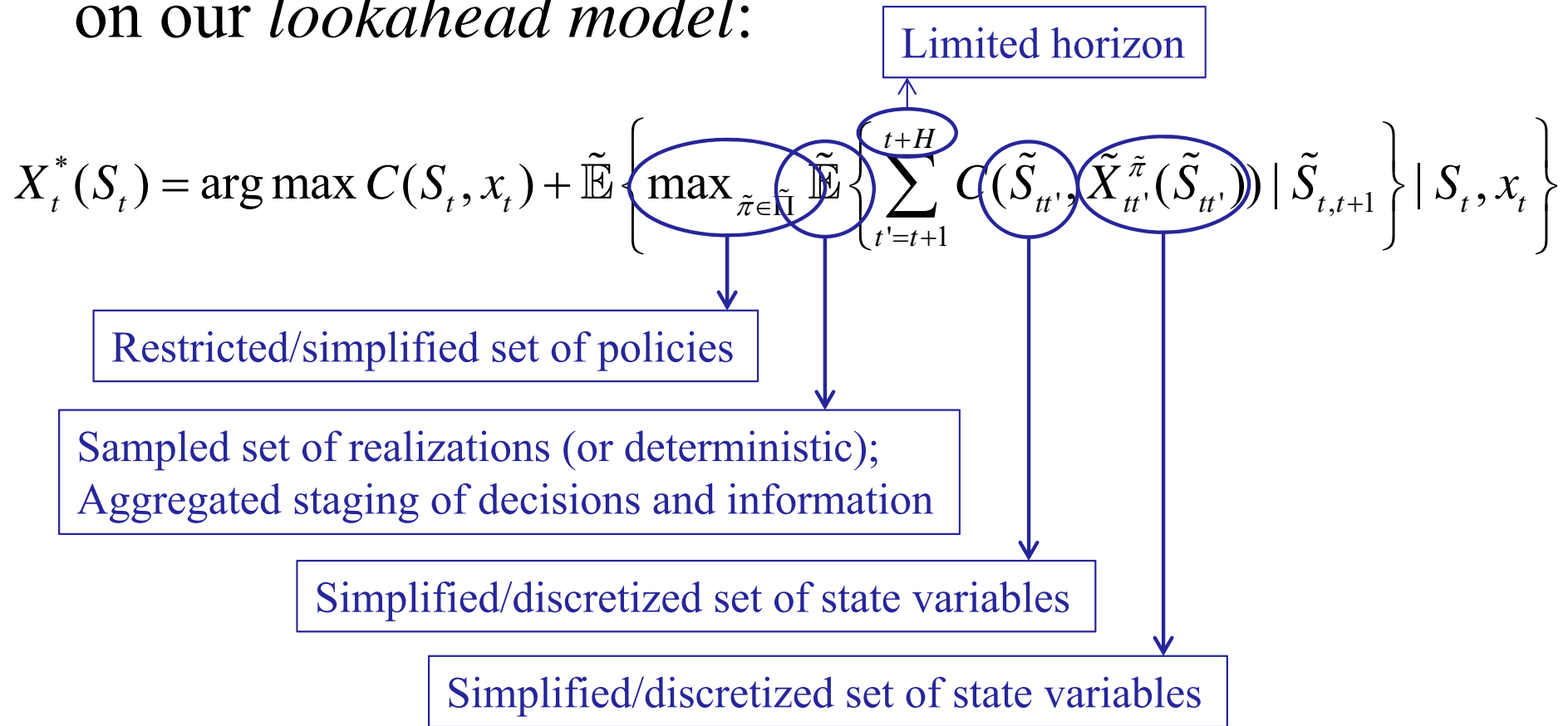
- Some options:

- » Parameterized policy

- E.g. “order up to”, or “sell when price goes above some point”
- VFA-based policy – We could solve a lookahead DP (exact or with ADP)
- Deterministic lookahead
- Stochastic lookahead
  - Monte Carlo tree search

# Direct lookahead

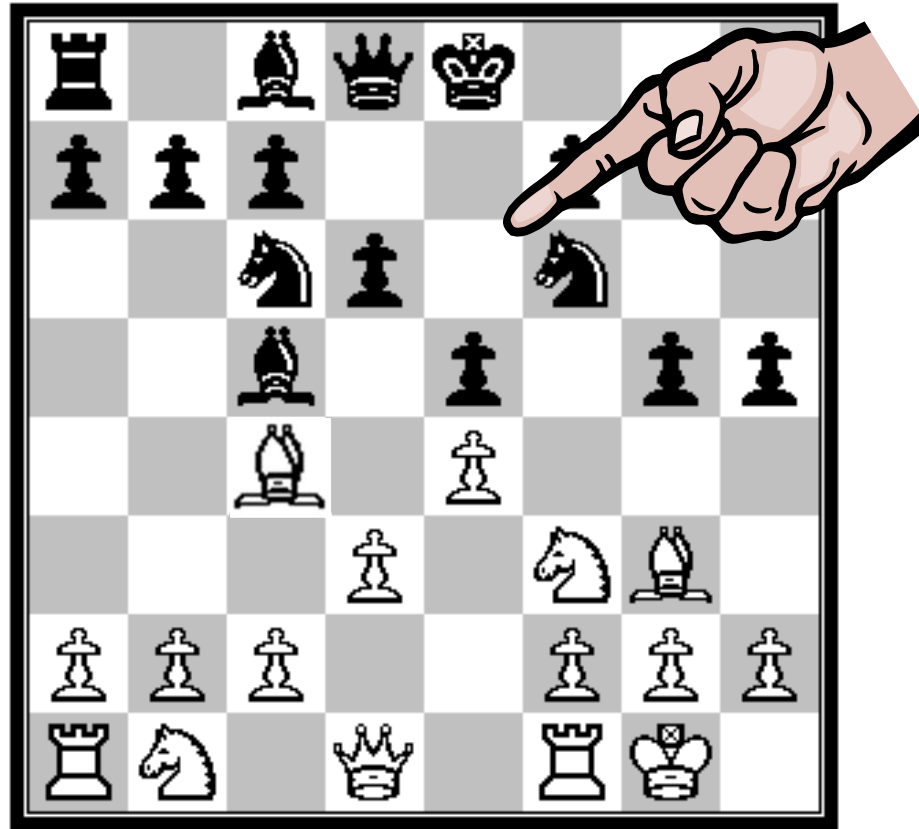
- We can use this notation to create a policy based on our *lookahead model*:



» Simplest lookahead is deterministic.

# Direct lookahead

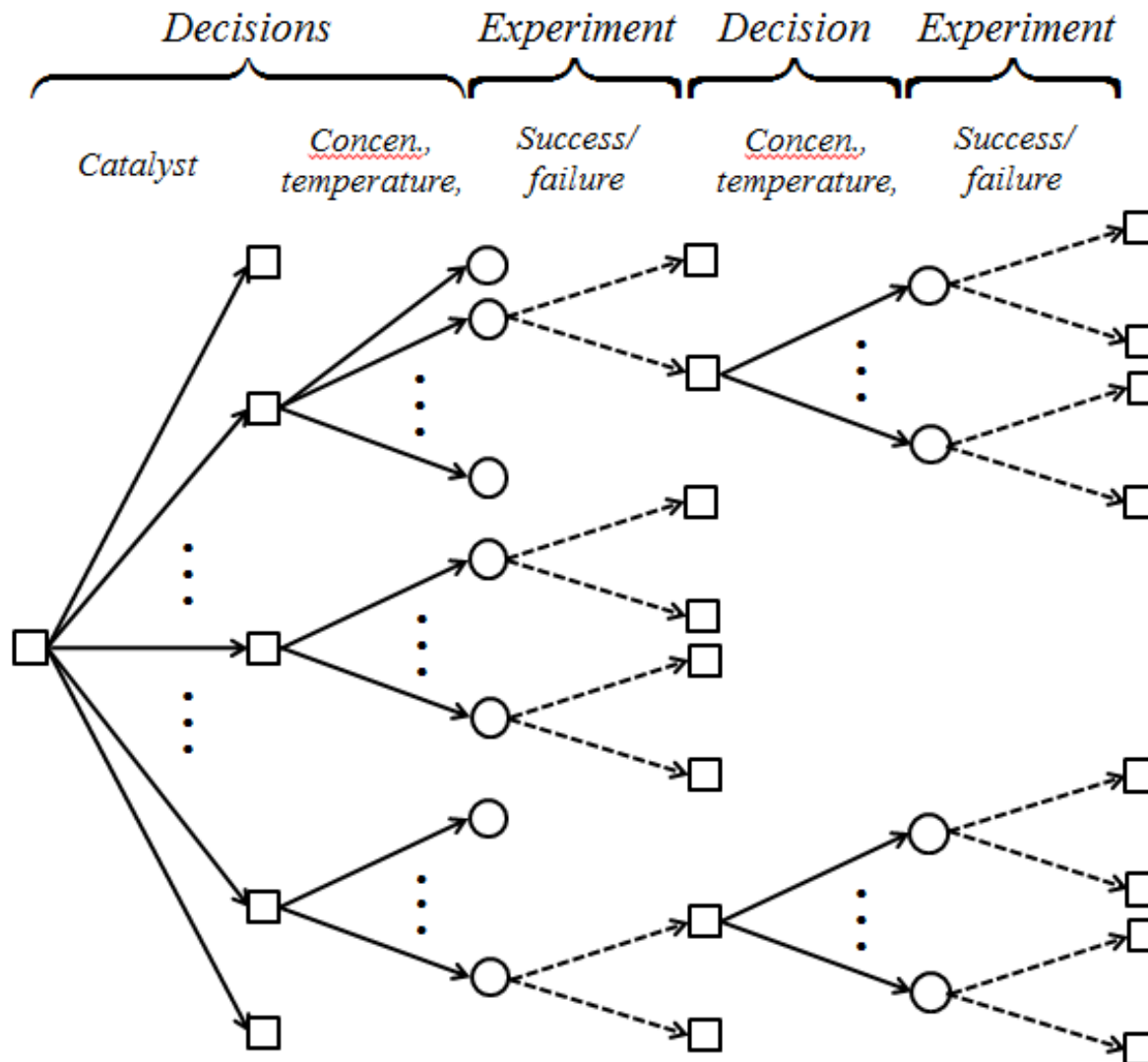
- Planning your next chess move:



- » 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.

# Direct lookahead

## ● Decision trees:



# Direct lookahead

---

- 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).



# Direct lookahead

- 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  $t$ ,  $t' = t, t + 1, \dots, t + H$

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

$\tilde{W}_{t,t'}$  = 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).

# Direct lookahead policies

Deterministic approximations

“Model predictive control”

Rolling horizon procedure

Receding horizon procedure

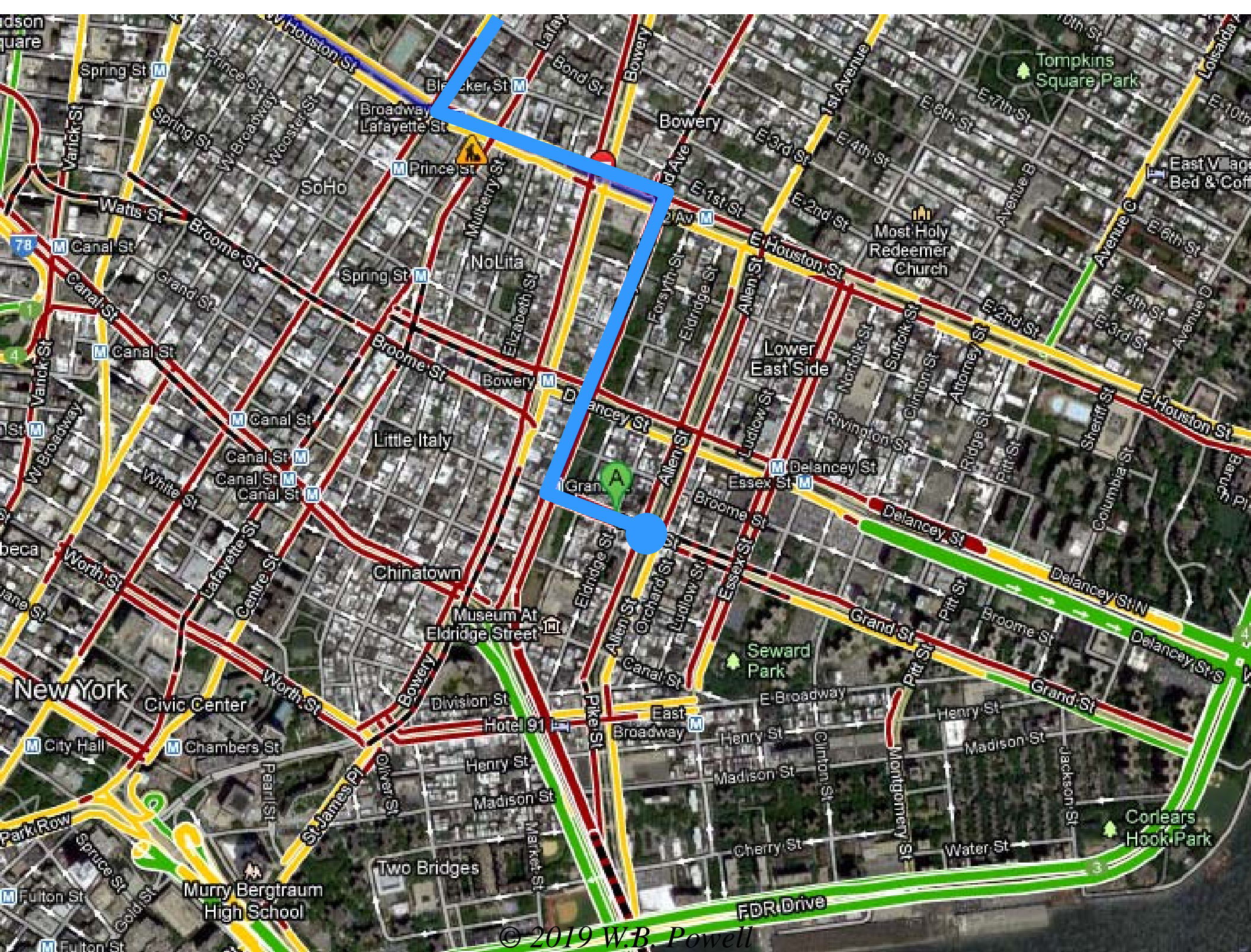
# Deterministic lookahead

## ● Deterministic lookahead policies

- » When we eliminate uncertainty, our lookahead model looks like:

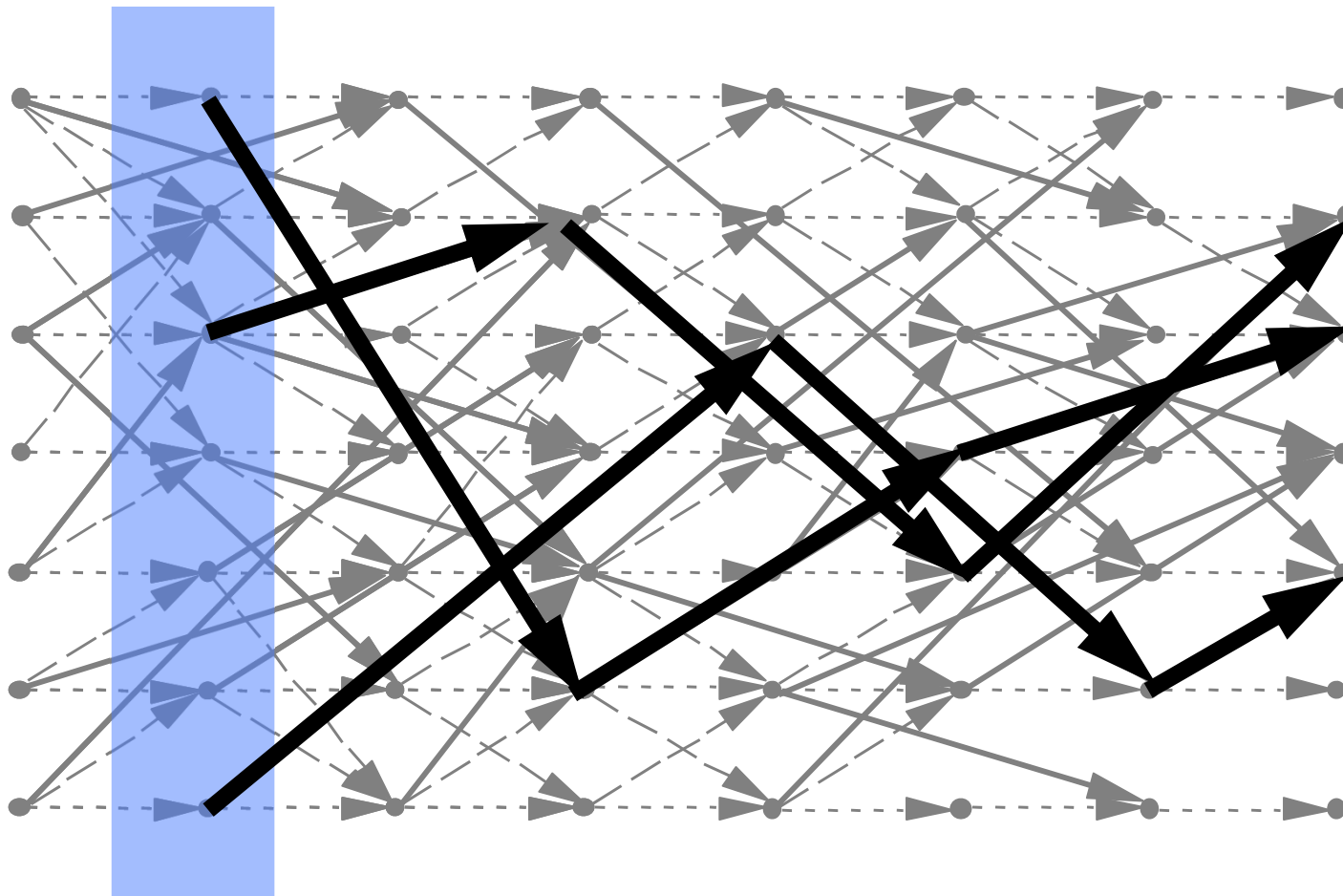
$$\begin{aligned} X_t^{DLA}(S_t) &= \arg \max_{x_t} \left( C(S_t, x_t) + \left\{ \max_{\tilde{x}_{t,t+1}, \dots, \tilde{x}_{t,t+H}} \left\{ \sum_{t'=t+1}^{t+H} C(\tilde{S}_{t'}, \tilde{x}_{t'}) \mid \tilde{S}_{t,t+1} \right\} \mid \tilde{S}_{t,t} \right\} \right) \\ &= \arg \max_{x_t, \tilde{x}_{t,t+1}, \dots, \tilde{x}_{t,t+H}} \left( C(S_t, x_t) + \sum_{t'=t+1}^{t+H} C(\tilde{S}_{t'}, \tilde{x}_{t'}) \right) \end{aligned}$$

- » Now we just have to pick an optimal action in the future (think of a shortest path problem).



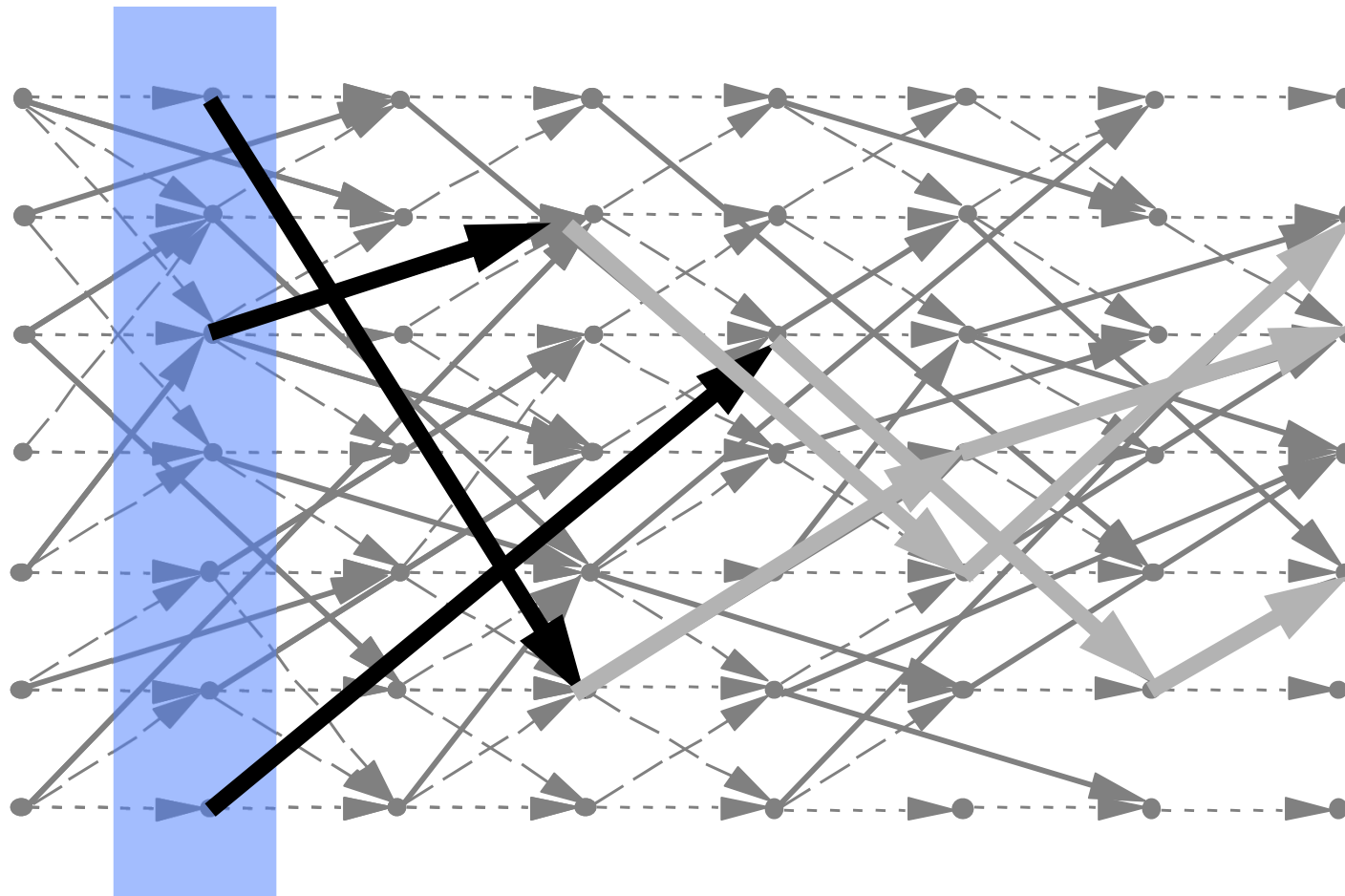
# Deterministic lookahead

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



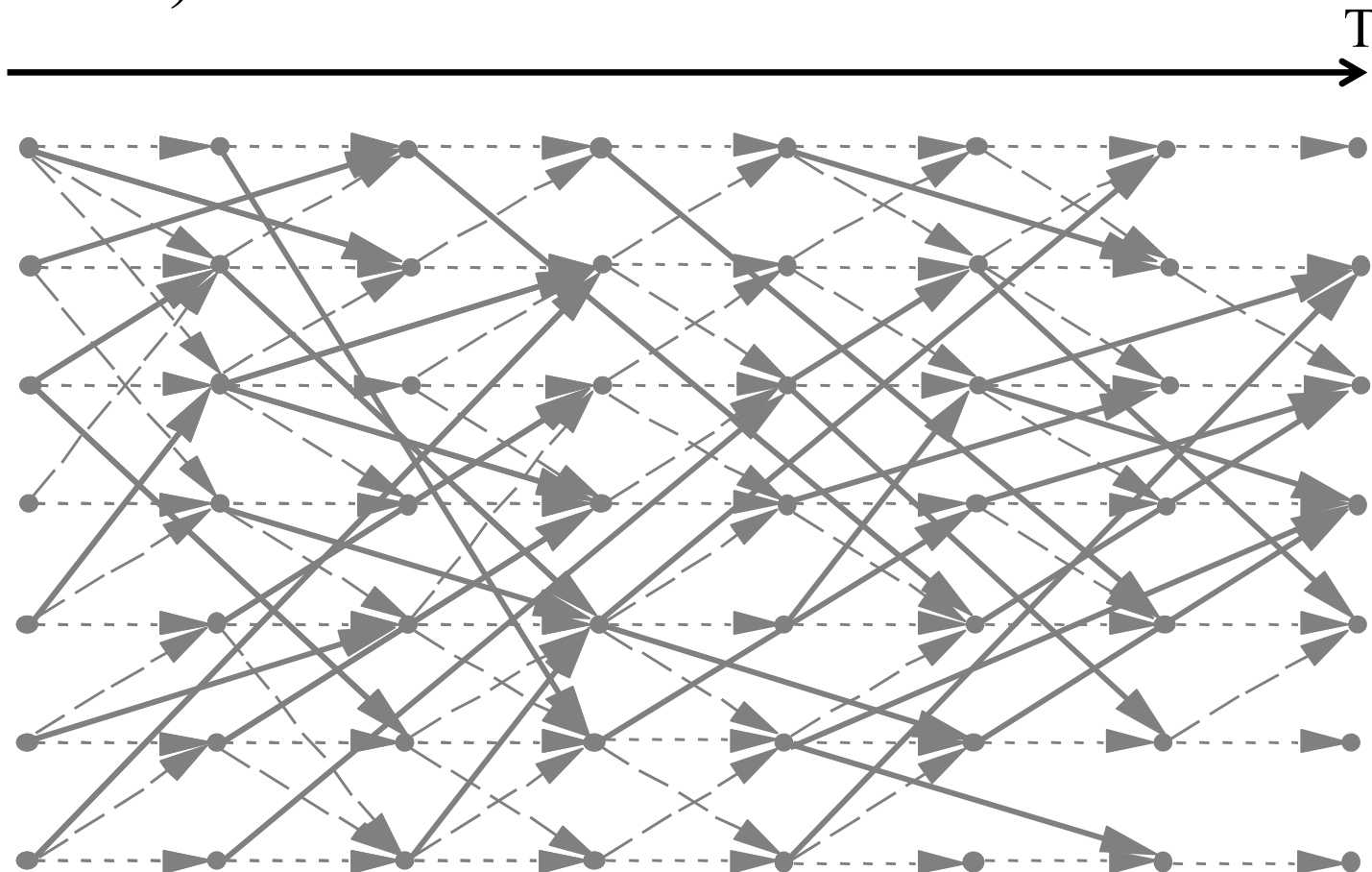
# Deterministic lookahead

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



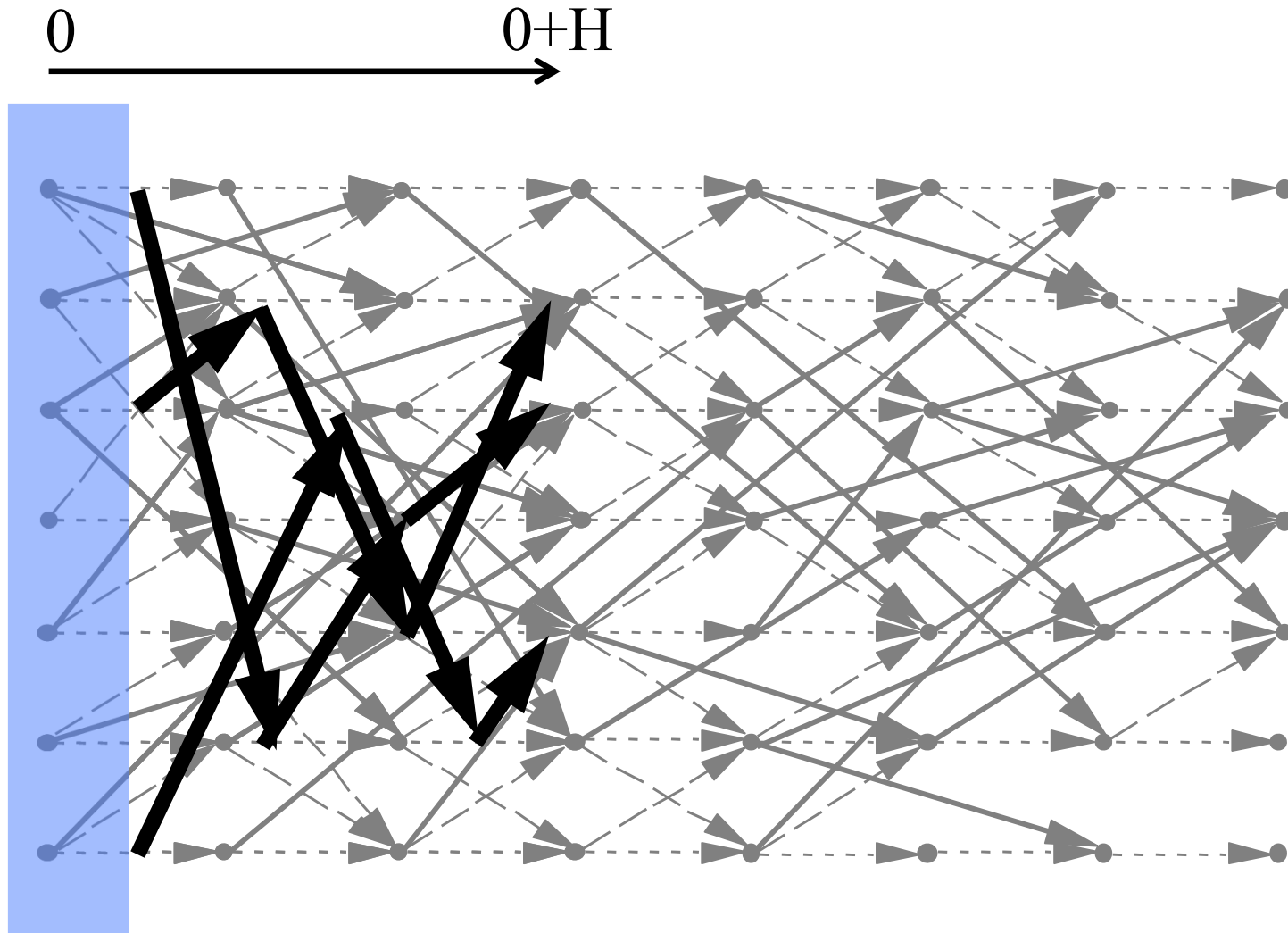
# Deterministic lookahead

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



# Deterministic lookahead

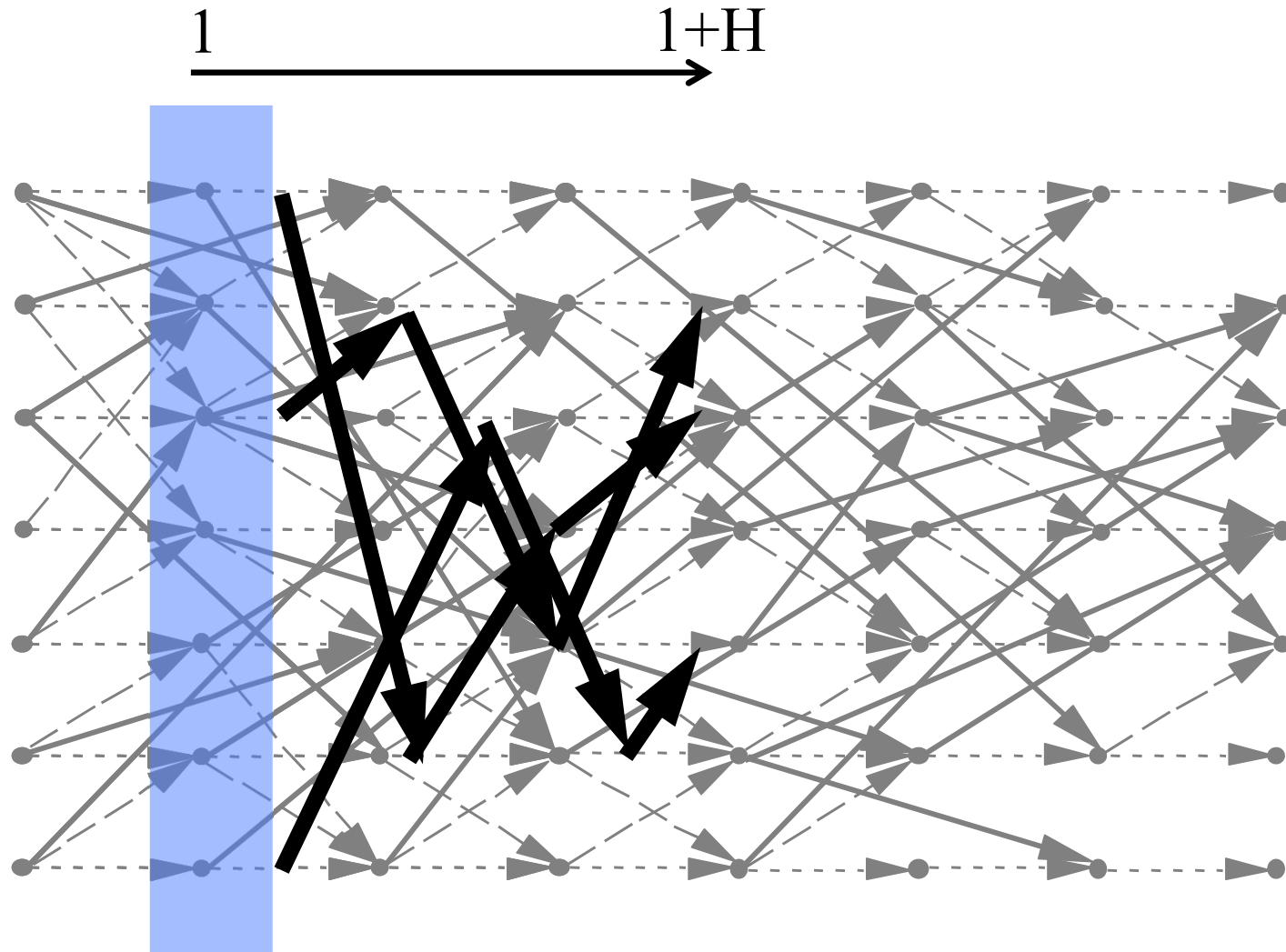
- But we solve a smaller lookahead model (from  $t$  to  $t+H$ )





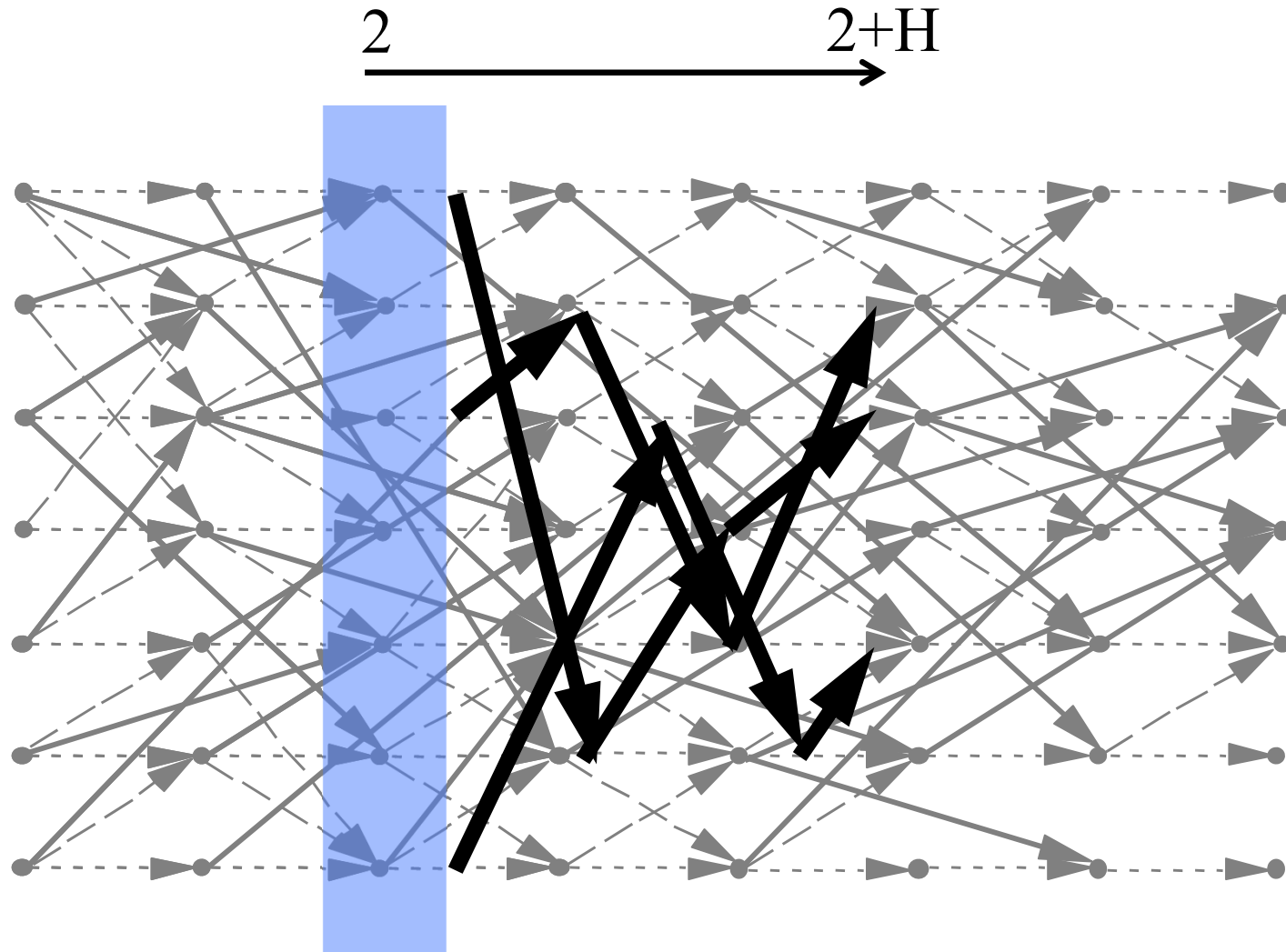
# Deterministic lookahead

## ■ Following a lookahead policy



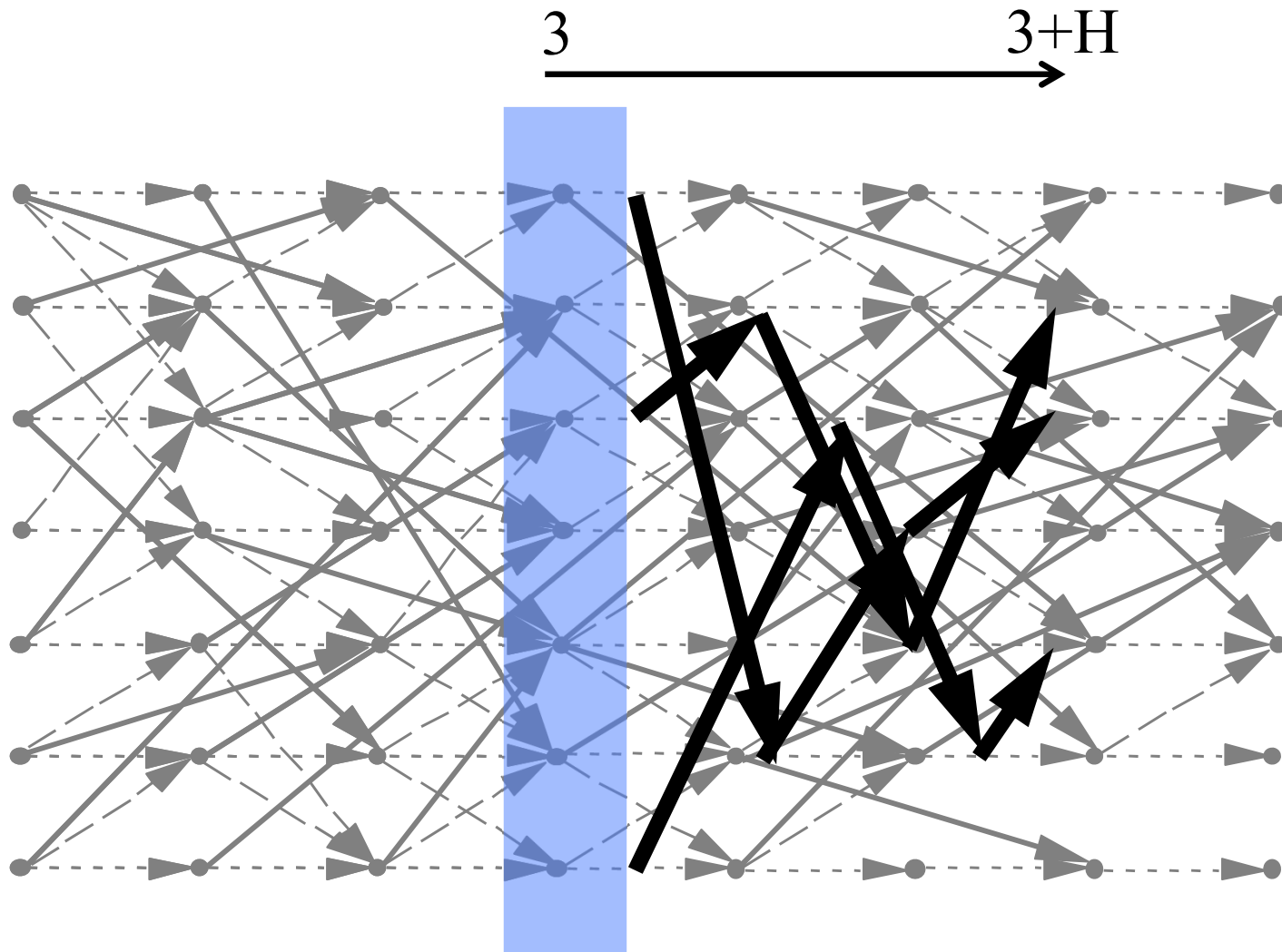
# Deterministic lookahead

- ... which rolls forward in time.



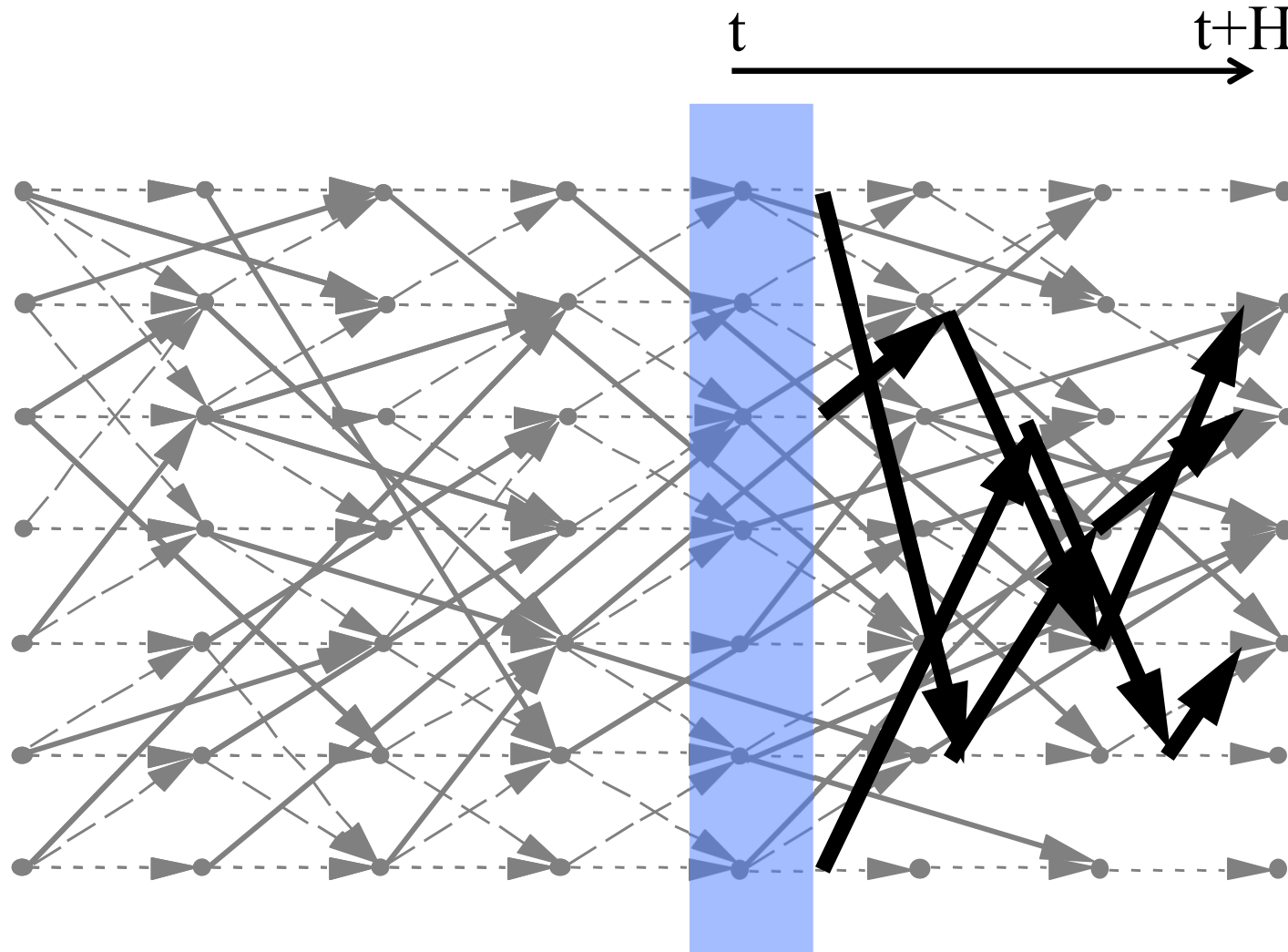
# Deterministic lookahead

- ... which rolls forward in time.



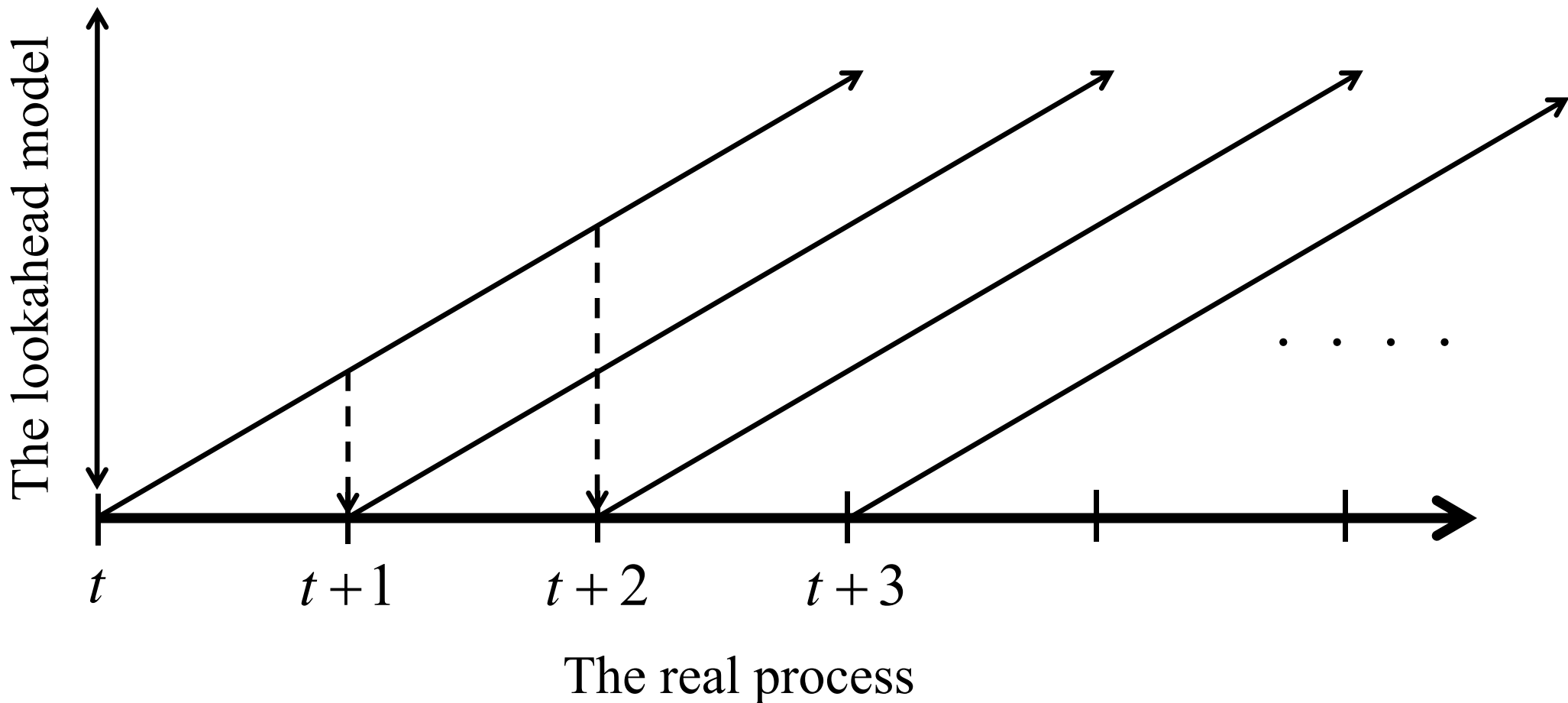
# Deterministic lookahead

- ... which rolls forward in time.



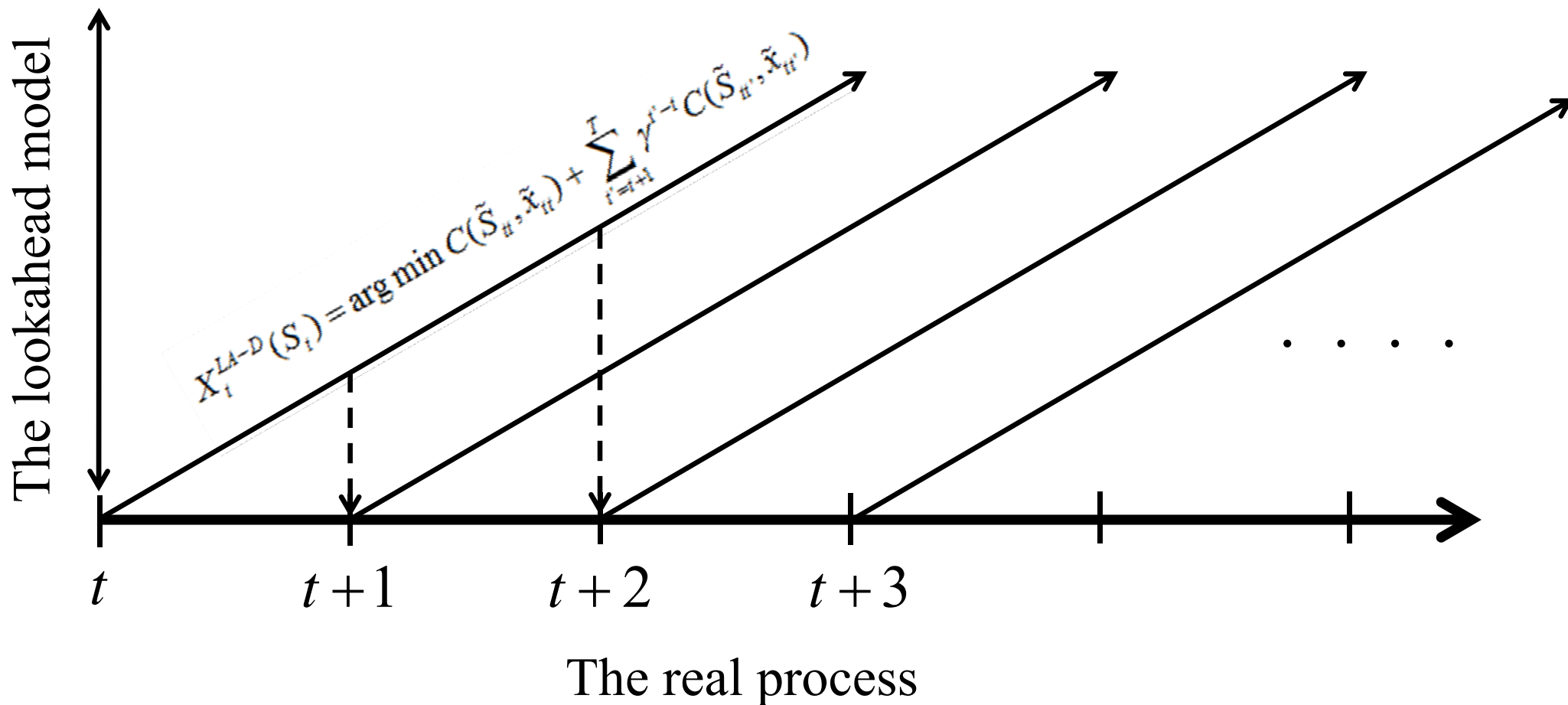
# Deterministic lookahead

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



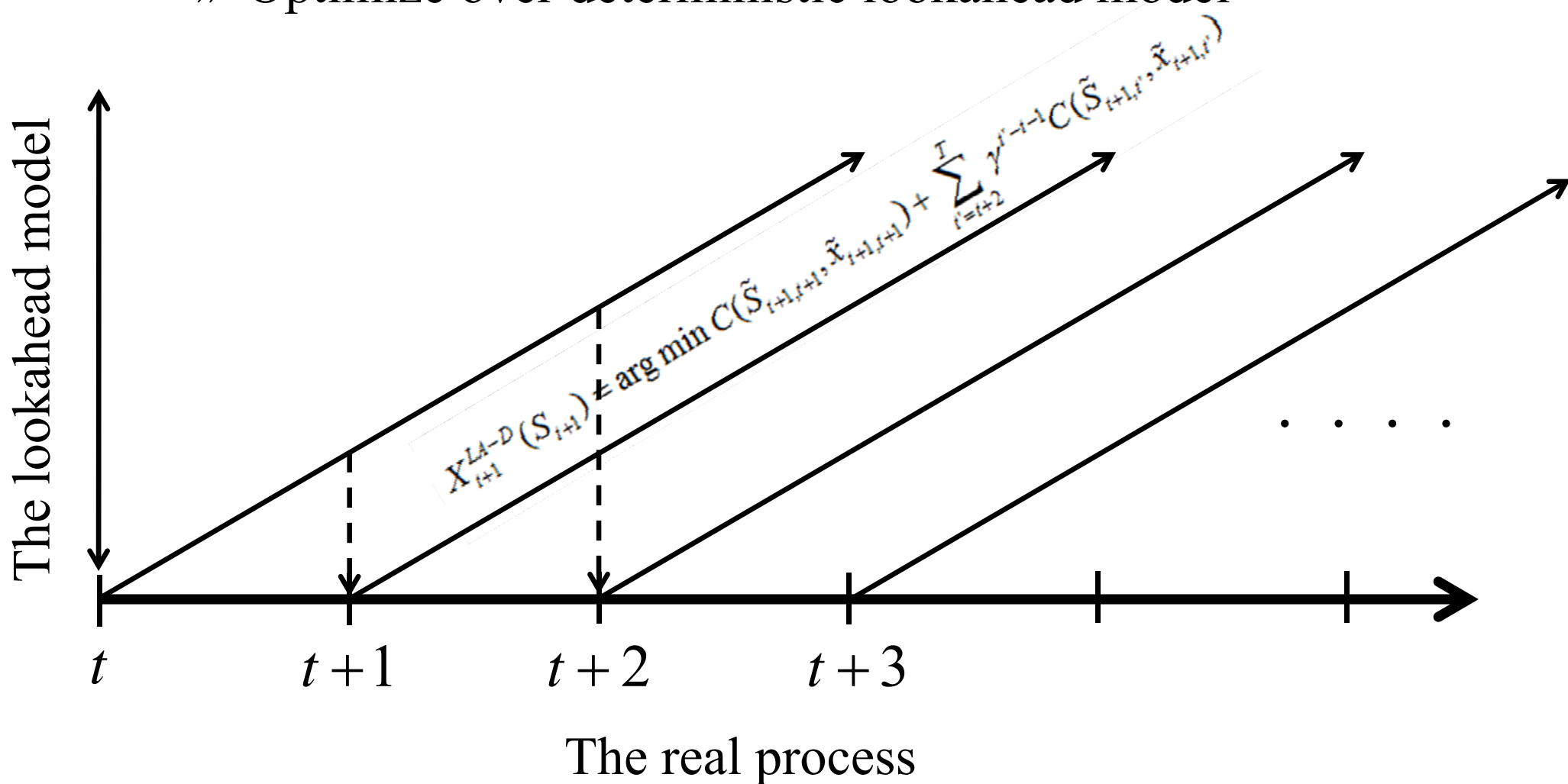
# Deterministic lookahead

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



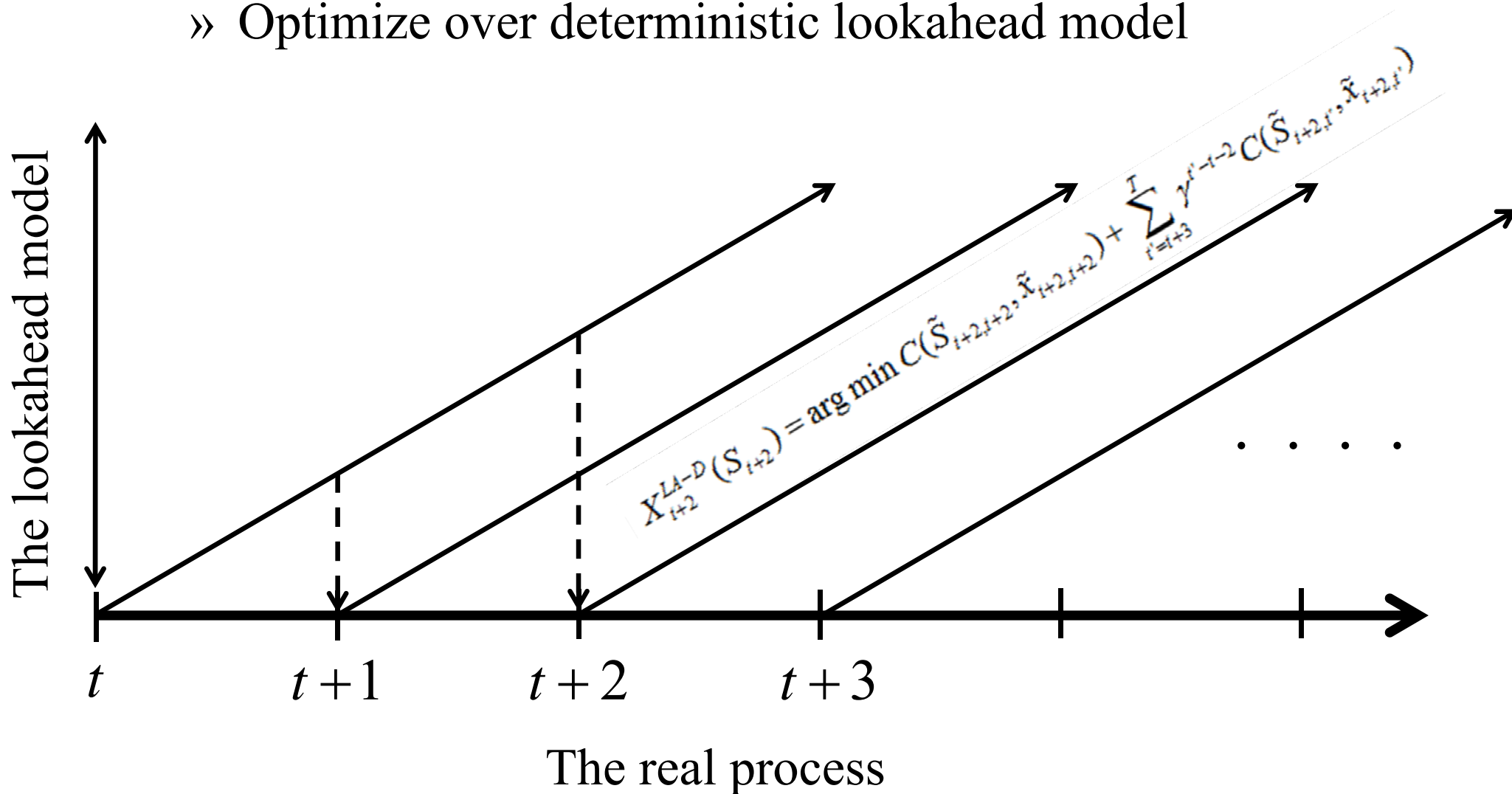
# Deterministic lookahead

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



# Deterministic lookahead

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





# Deterministic lookahead

---

## ● Notes

- » It is common in a deterministic lookahead to introduced a tunable parameter, as we did with our energy storage problem.
- » In effect, we are compensating for the simplicity of our lookahead with a tunable parameter.

Direct lookahead

Energy planning problem

# Parameterized deterministic lookahead

---

- Recall from the CFA lecture:
  - » We can do a lookahead, but parameterize it to handle uncertainty.
  - » In the energy storage problem, we multiplied the forecasts times a coefficient to accommodate uncertainty....

# Parameterized deterministic lookahead

## ● An energy storage problem:



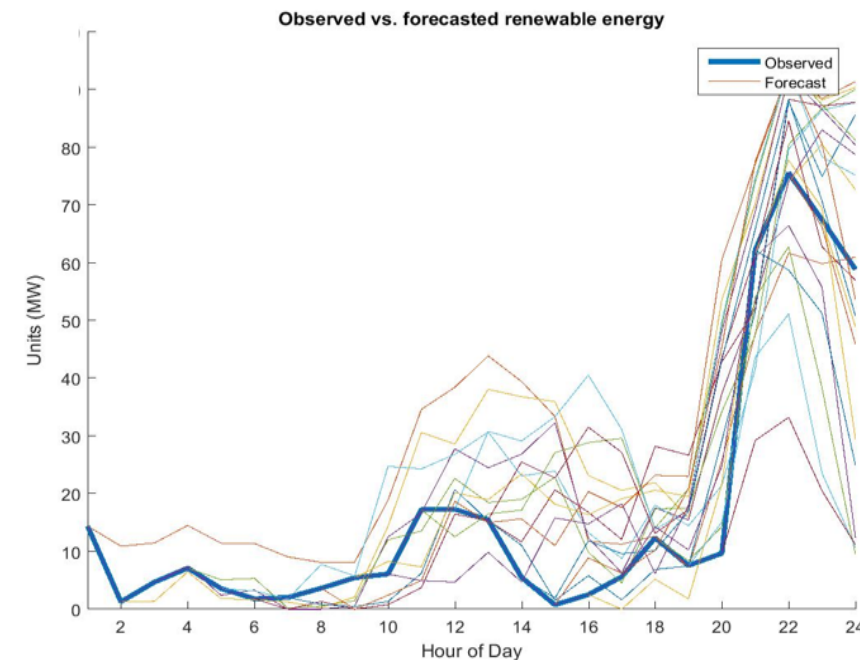
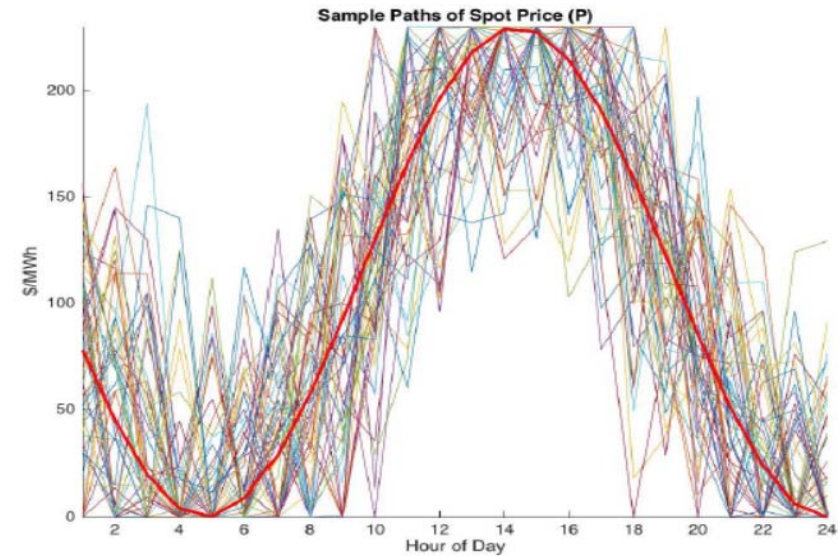
The state of the system can be represented by

$r_t$ ,

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

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



# Parameterized deterministic lookahead

- Benchmark policy – Deterministic lookahead

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

$$\tilde{x}_{tt'}^{wd} + \beta \tilde{x}_{tt'}^{rd} + \tilde{x}_{tt'}^{gd} \leq f_{tt'}^D$$

$$\tilde{x}_{tt'}^{gd} + \tilde{x}_{tt'}^{gr} \leq f_{tt'}^G$$

$$\tilde{x}_{tt'}^{rd} + \tilde{x}_{tt'}^{rg} \leq \tilde{R}_{tt'}$$

$$\tilde{x}_{tt'}^{wr} + \tilde{x}_{tt'}^{gr} \leq R^{\max} - \tilde{R}_{tt'}$$

$$x_{tt'}^{wr} + x_{tt'}^{wd} \leq \theta_{t'-t} f_{tt'}^E$$

$$\tilde{x}_{tt'}^{wr} + \tilde{x}_{tt'}^{gr} \leq \gamma^{\text{charge}}$$

$$\tilde{x}_{tt'}^{rd} + \tilde{x}_{tt'}^{rg} \leq \gamma^{\text{discharge}}$$

# Parameterized deterministic lookahead

- Parametric cost function approximations

» Now we need to find the best parameterization:

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

» We evaluate the policy via simulation:

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

# Parameterized deterministic lookahead

## ● Designing the policy

### » The structure of the policy

- Should the coefficients  $\theta$  be indexed by:
  - Time  $t$ , as in  $\theta_t f_{tt'}$
  - Or number of time periods in the future  $\theta_{t'-t} f_{tt'}$ .
- Should we have additive terms?
- This is the art of parametric modeling.

### » Tuning the parameters

- We can use any of the algorithms in chapters 5 (derivative-based) or 7 (derivative-free).
- This is the science of parametric modeling.

# Direct lookahead

Dynamic shortest path problem

Deterministic lookahead



# Dynamic shortest paths

---

- The problem

- » Finding the best path through a stochastic dynamic network.

- The policy

- » We can try to solve a stochastic lookahead model (perhaps using approximate dynamic programming).
- » We can solve a deterministic lookahead model using point estimates of travel times that are updated from time to time.
- » We can solve a parameterized lookahead model, where we use the  $\theta$ -percentile of the travel time on each link.

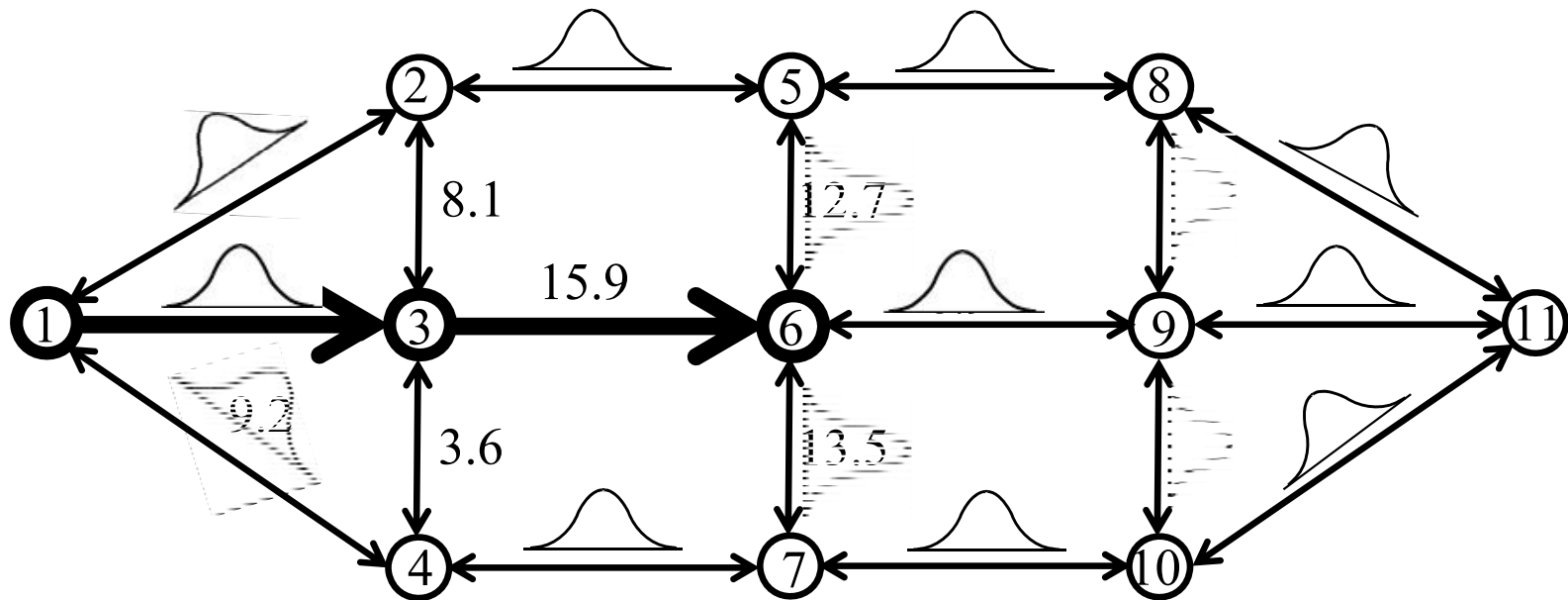
# Dynamic shortest paths

---

- Flavors of stochastic networks
  - » Arc costs are known after we make a decision – This is a deterministic shortest path problem that we can solve as a linear program or a classical deterministic shortest path problem.
  - » Arc costs are known before we make a decision – Now we would have to use approximate dynamic programming to solve the lookahead model. This then has to be re-optimized as the mean arc costs are updated.

# Dynamic shortest paths

- A stochastic network, costs revealed as we arrive to a node:



# Dynamic shortest paths

## ● Modeling:

### » Objective function

- Cost per period

$$\begin{aligned}C(S_t, X_t^\pi(S_t)) &= (X_t^\pi(S_t))^T \hat{c}_t \\ &= \sum_j x_{t,i_t,j}^\pi \hat{c}_{tij} \\ &= \text{Costs incurred at time } t.\end{aligned}$$

- Total costs:

$$\min_{\pi} \mathbb{E} \sum_{t=0}^T C(S_t, X_t^\pi(S_t))$$

### » This is the base model.

# Dynamic shortest paths

## ● A policy based on a lookahead model

- » At each time  $t$  we are going to optimize over an estimate of the network that we are going to call the *lookahead model*.
- » Notation: all variables in the lookahead model have tilde's, and two time indices.
  - First time index,  $t$ , is the time at which we are making a decision. This determines the information content of all parameters (e.g. costs) and decisions.
  - A second time index,  $t'$ , is the time within the lookahead model.
- » Decisions
  - $\tilde{x}_{tij} = 1$  if we plan on traversing link  $(i, j)$  in the lookahead model.
  - $\tilde{c}_{tij} =$  Estimated cost at time  $t$  of traversing link  $(i, j)$  in the lookahead model.

# Dynamic shortest paths

- Imagine that the lookahead is just a black box:

» Solve the optimization problem

$$X_t^\pi(S_t^n) = \arg \min \sum_{i \in N} \sum_{j \in N_i^+} \tilde{c}_{tij}^n \tilde{x}_{tij}$$

» subject to

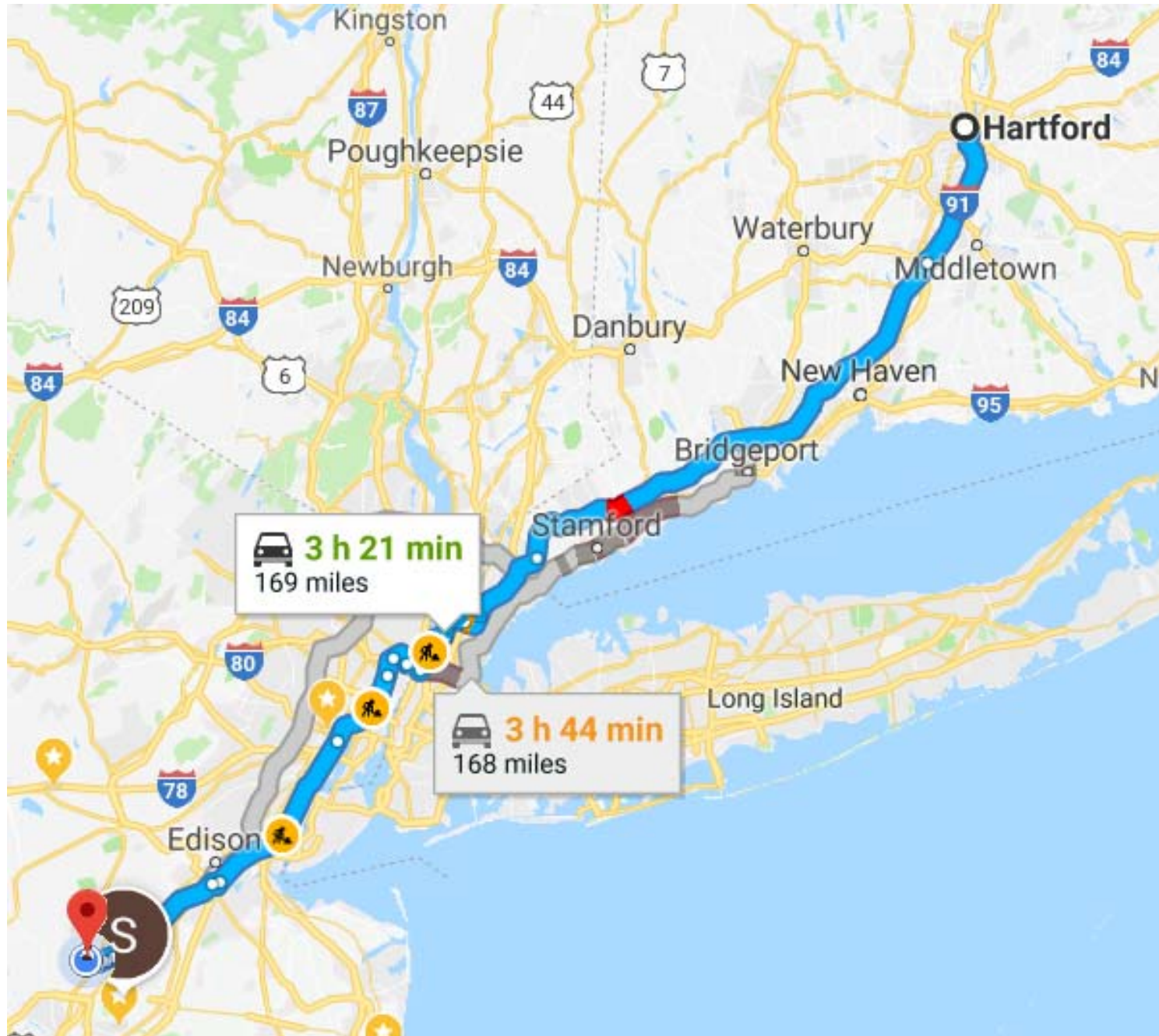
$$\sum_j \tilde{x}_{i^n, j} = 1 \quad \text{Flow out of current node where we are located}$$

$$\sum_i \tilde{x}_{i, r} = 1 \quad \text{Flow into destination node } r$$

$$\sum_i \tilde{x}_{i, j} - \sum_k \tilde{x}_{j, k} = 0 \quad \text{for all other nodes.}$$

» This is a deterministic shortest path problem that we could solve using Bellman's equation, but for now we will just view it as a black box optimization problem.

# Dynamic shortest paths





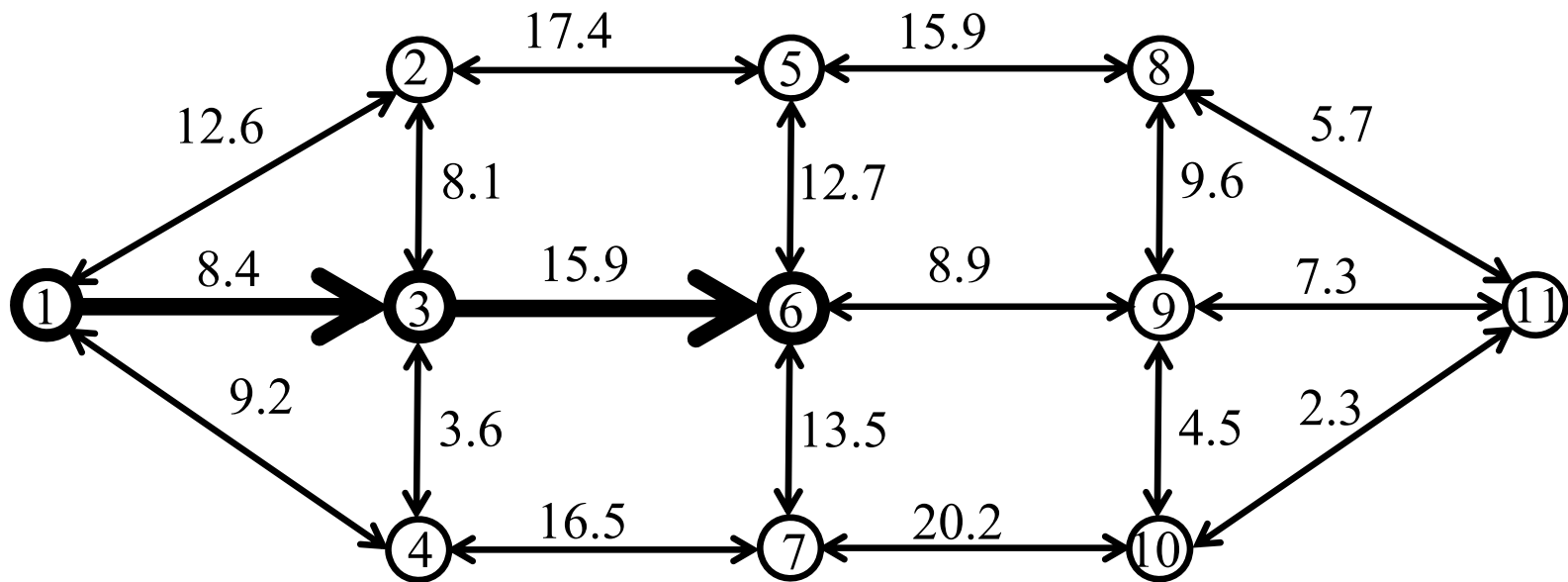
# Dynamic shortest paths





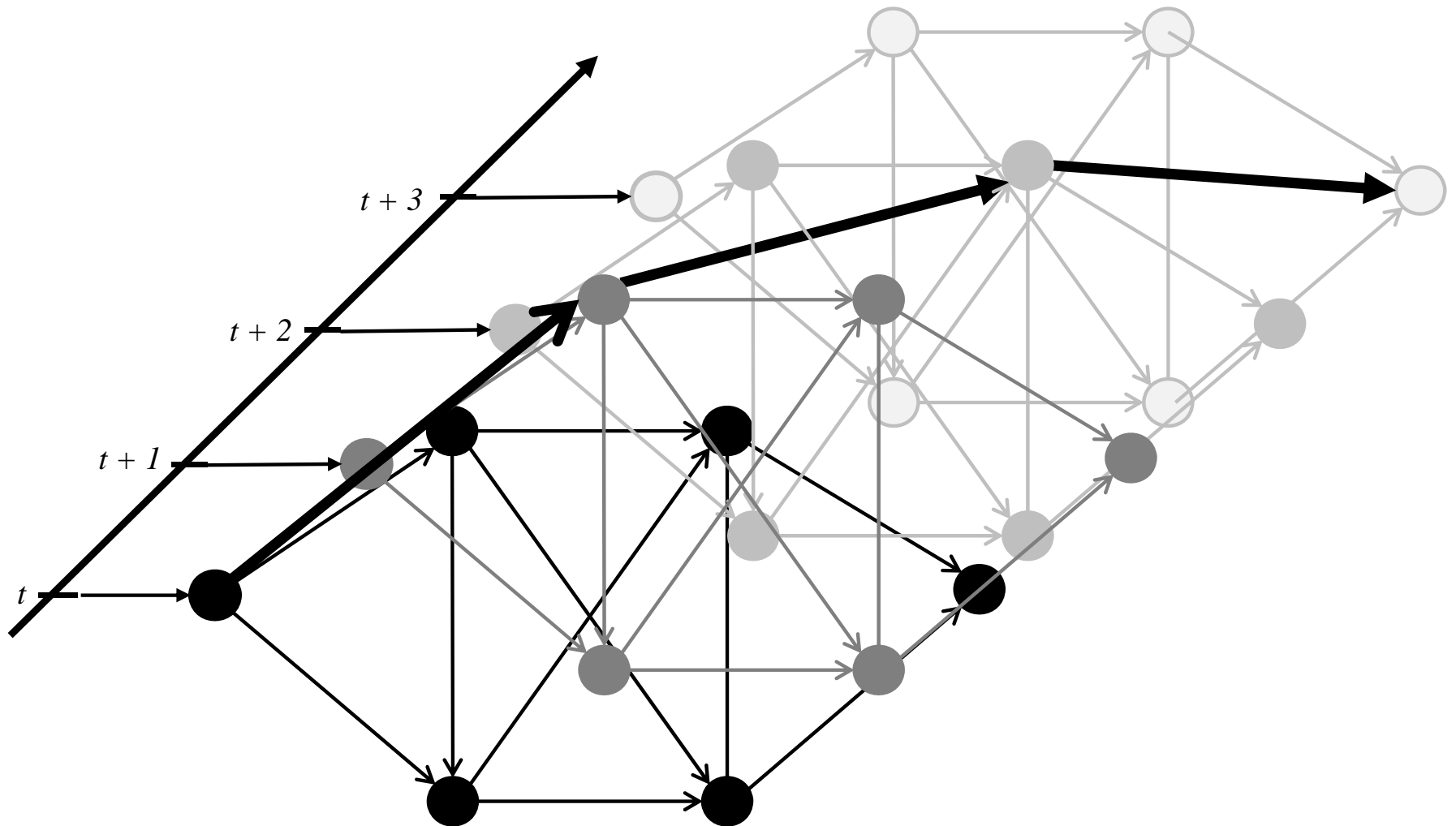
# Dynamic shortest paths

- A static, deterministic network



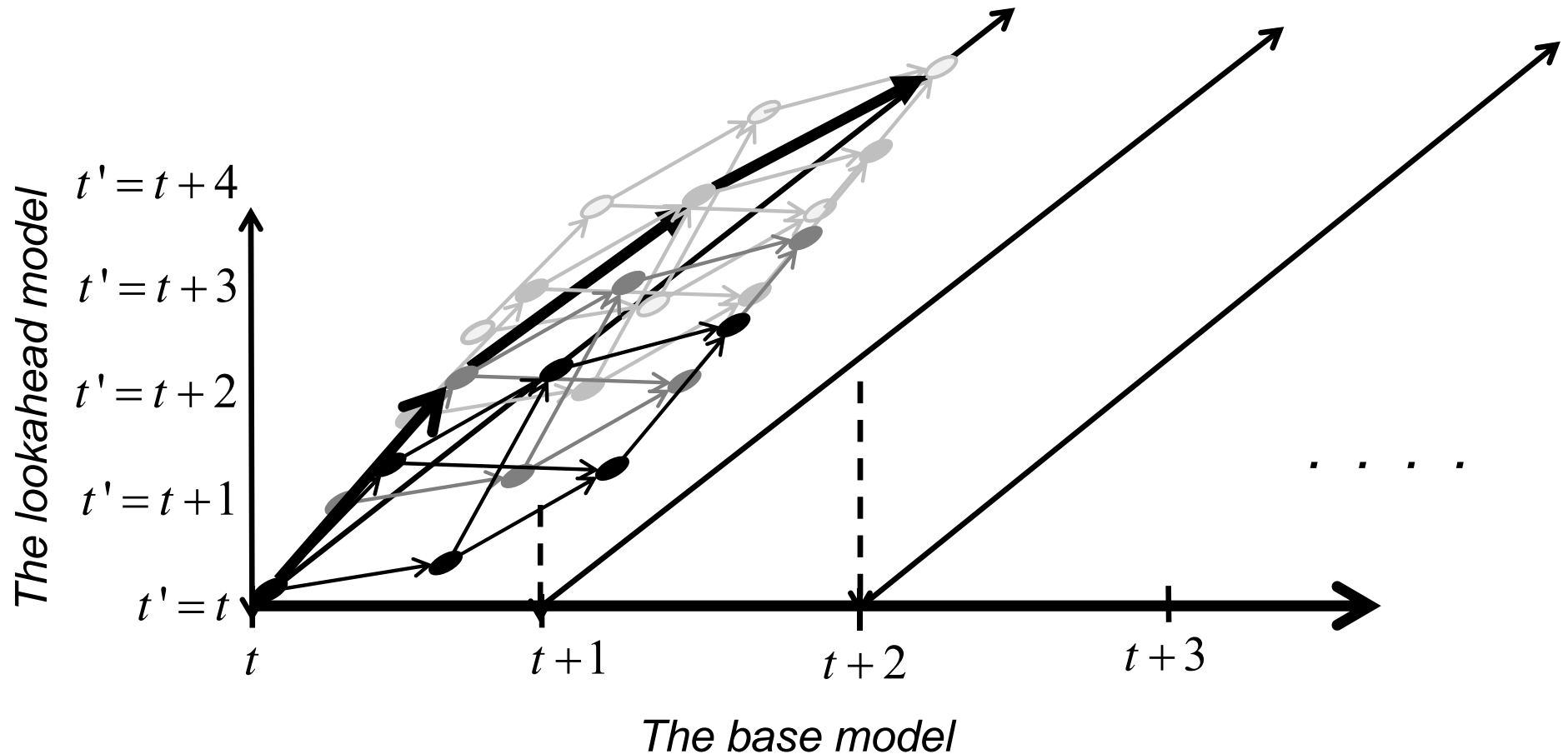
# Dynamic shortest paths

- A time-dependent, deterministic network



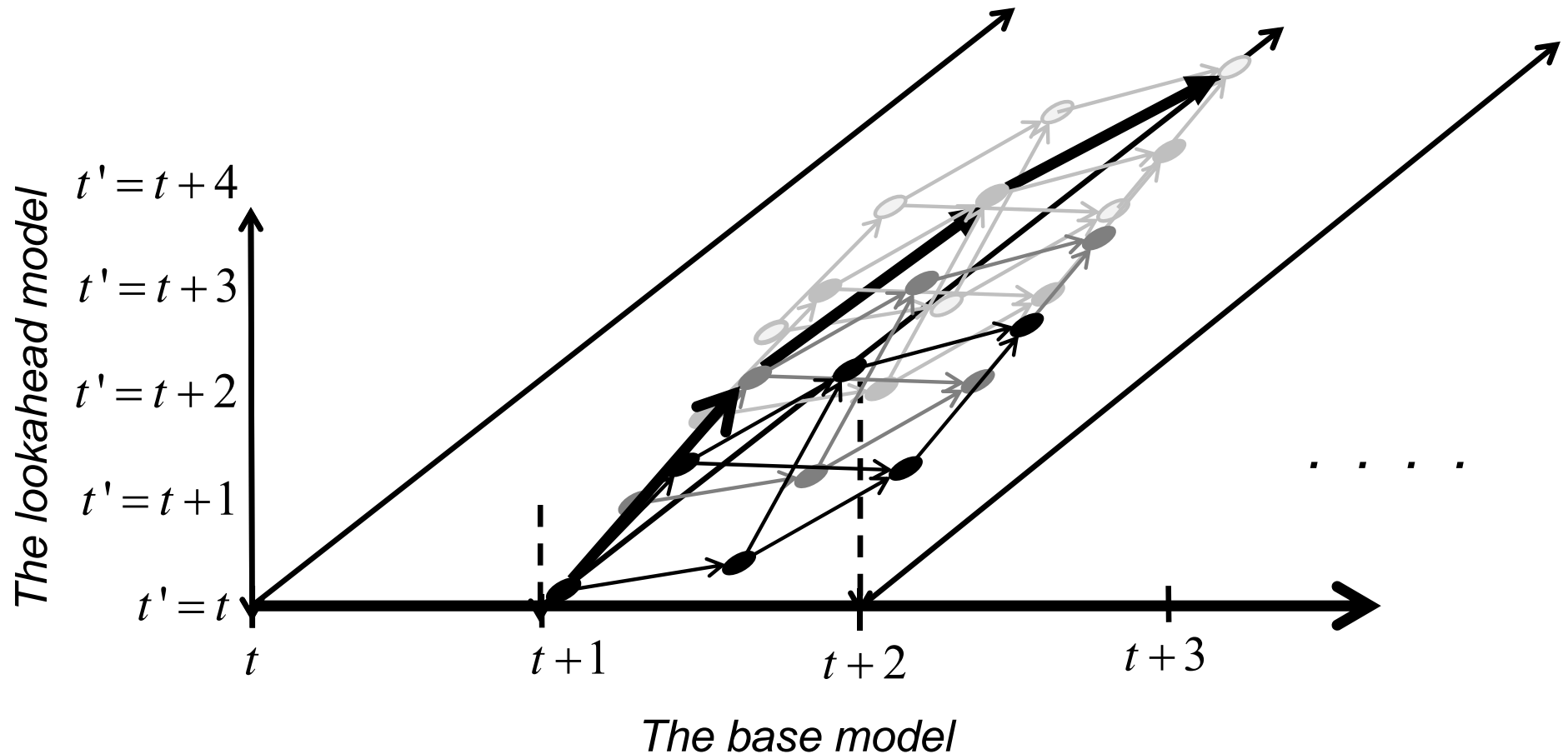
# Dynamic shortest paths

- A time-dependent, deterministic lookahead network



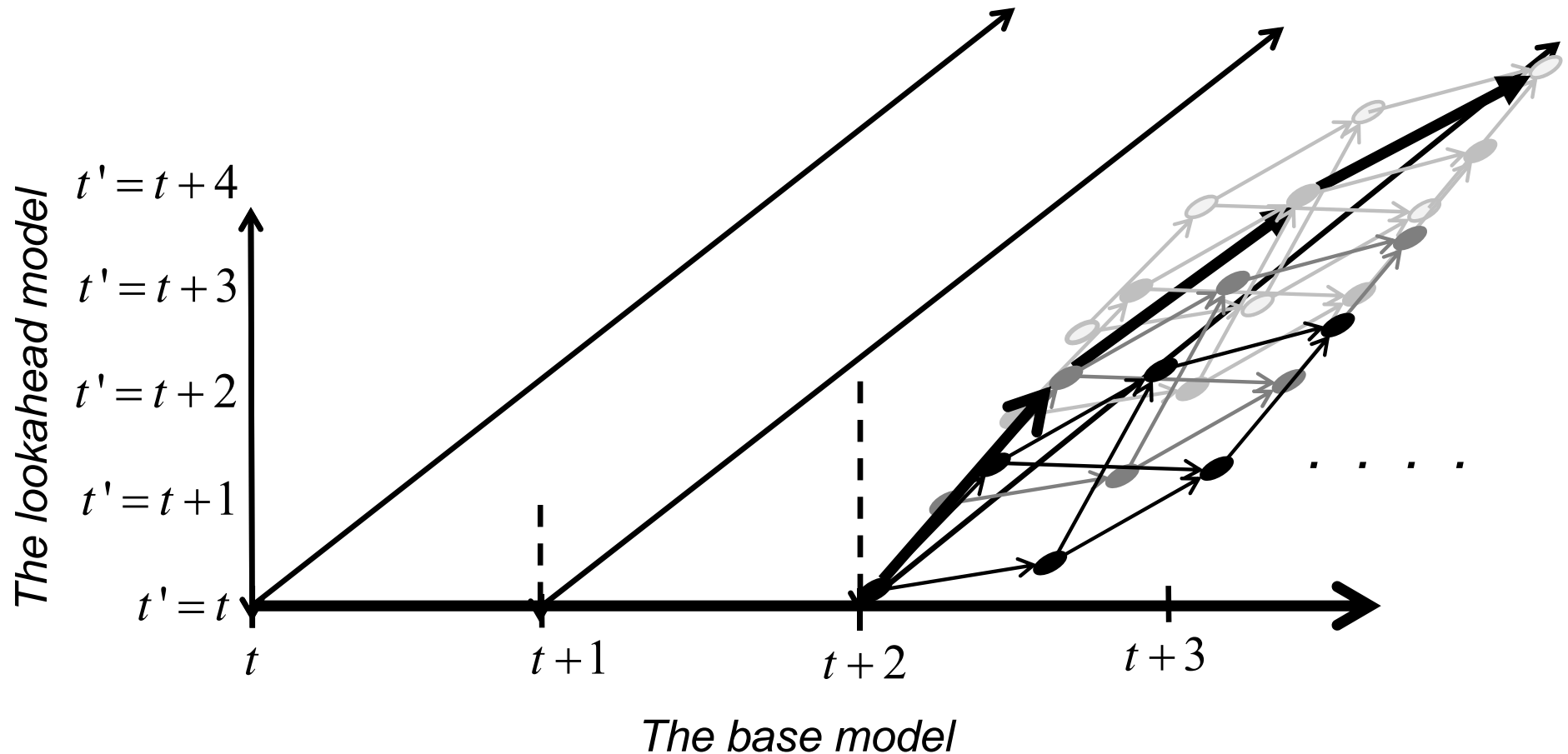
# Dynamic shortest paths

- A time-dependent, deterministic lookahead network



# Dynamic shortest paths

- A time-dependent, deterministic lookahead network



# Dynamic shortest paths

- Simulating a lookahead policy

We would like to compute

$$F^\pi = \mathbb{E} \sum_{t=0}^T \sum_{i,j} X_{t,ij}^\pi(S_t) \hat{c}_{t,ij}$$

but this is intractable.

Let  $\omega$  be a sample realization of costs

$$\hat{c}_{t,t',ij}(\omega), \hat{c}_{t+1,t',ij}(\omega), \hat{c}_{t+2,t',ij}(\omega), \dots$$

Now simulate the policy

$$\hat{F}^\pi(\omega^n) = \sum_{t=0}^T \sum_{i,j} X_{t,ij}^\pi(S_t(\omega^n)) \hat{c}_{t,ij}(\omega^n)$$

Talk through how this works.

Finally, get the average performance

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

# Dynamic shortest paths

---

## ● Discuss:

- » The lookahead model uses forecasted costs  $\bar{c}_{tij}$ . These are estimates made at time  $t$ .
- » The policy is simulated using sampled costs  $\hat{c}_{tij}$ .
- » Both are updated with time and from one iteration to another.

# Direct lookahead

Dynamic shortest path problem

Parameterized deterministic lookahead



# Dynamic shortest paths

- Notes:

- » The deterministic lookahead is still a policy for a stochastic problem.
- » Can we make it better?

- Idea:

- » Instead of using the expected cost, what about using a percentile.

- » Use pdf of  $\hat{c}_{ij}$  to find  $\theta$  percentile (e.g.  $\theta = .8$ ). Let

$$\tilde{c}_{ij}^p(\theta) = \text{The } \theta \text{ -percentile of } \hat{c}_{ij}$$

- » Which means  $Prob \left[ \hat{c}_{ij} \leq \tilde{c}_{ij}^p(\theta) \right] = \theta$ .

# Parameterized deterministic lookahead

- The  $\theta$  –percentile policy.

- » Solve the linear program (shortest path problem):

$$X_t^\pi(S_t^n | \theta) = \arg \min \sum_{i \in N} \sum_{j \in N_i^+} \tilde{c}_{tij}^p(\theta) \tilde{x}_{tij} \quad (\text{Vector with } x_{tij} = 1 \text{ if decision is to take } (i, j))$$

- » subject to

$$\sum_j \tilde{x}_{t, i^n, j} = 1 \quad \text{Flow out of current node where we are located}$$

$$\sum_i \tilde{x}_{tir} = 1 \quad \text{Flow into destination node } r$$

$$\sum_i \tilde{x}_{tij} - \sum_k \tilde{x}_{tjk} = 0 \quad \text{for all other nodes.}$$

- » This is a deterministic shortest path problem that we could solve using Bellman's equation, but for now we will just view it as a black box optimization problem.

# Parameterized deterministic lookahead

- Simulating a lookahead policy

Let  $\omega$  be a sample realization of costs

$$\hat{c}_{t,t',ij}(\omega), \hat{c}_{t+1,t',ij}(\omega), \hat{c}_{t+2,t',ij}(\omega), \dots$$

Now simulate the policy

$$\hat{F}^\pi(\omega^n) = \sum_{t=0}^T \sum_{i,j} \hat{c}_{t,t',ij}(\omega) X_t^\pi(S_t(\omega^n) | \theta)$$

Finally, get the average performance

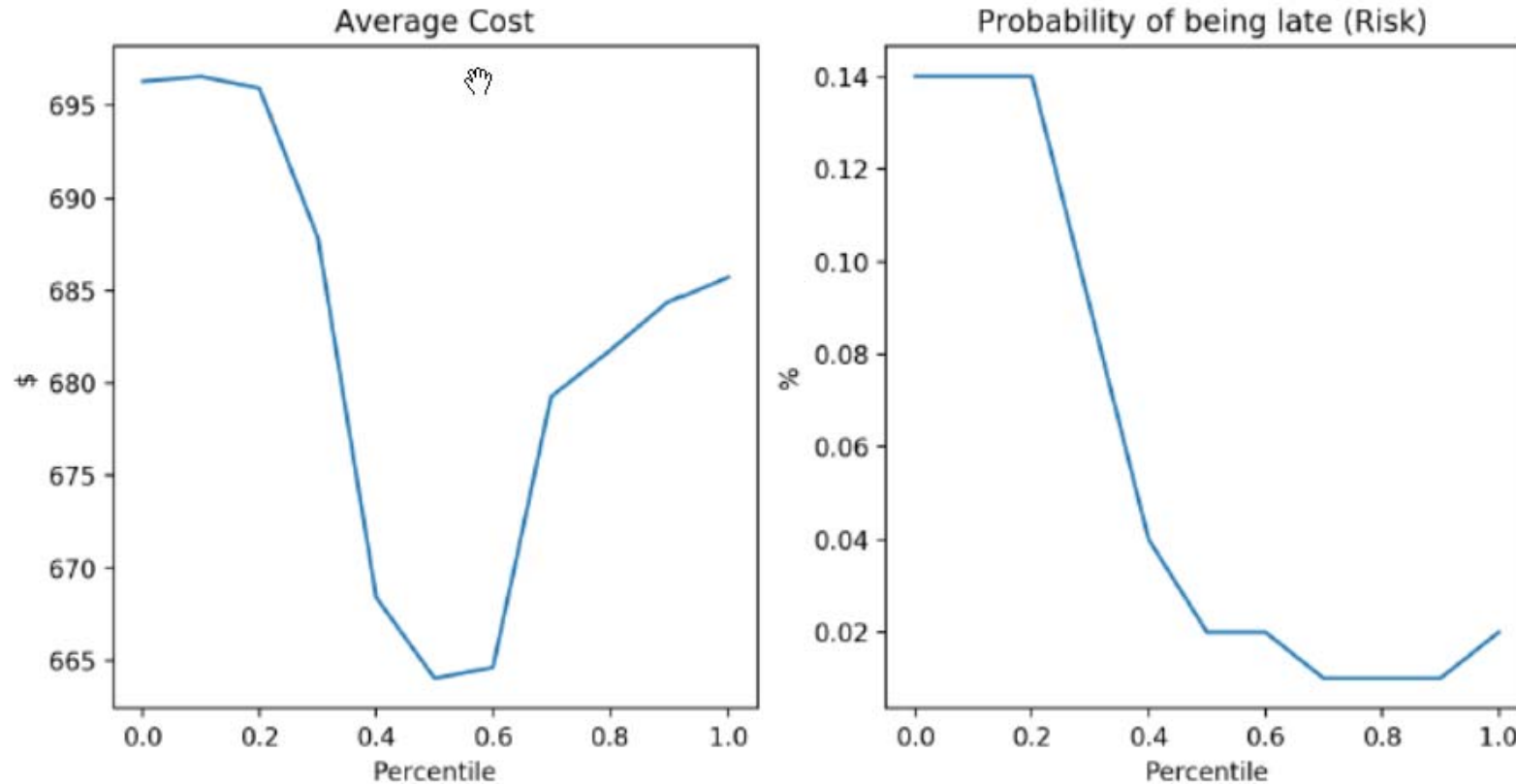
$$\bar{F}^\pi(\theta) = \frac{1}{N} \sum_{n=1}^N \hat{F}^\pi(\omega^n)$$

# Parameterized deterministic lookahead

## ● Policy tuning

» Cost vs. lateness (risk)

Comparison of  $\theta^{\text{cost}}$  - origin 0, destination 24, dist 6 - deadline 780.0 and number of iterations 100



Direct lookahead

Monte Carlo tree search

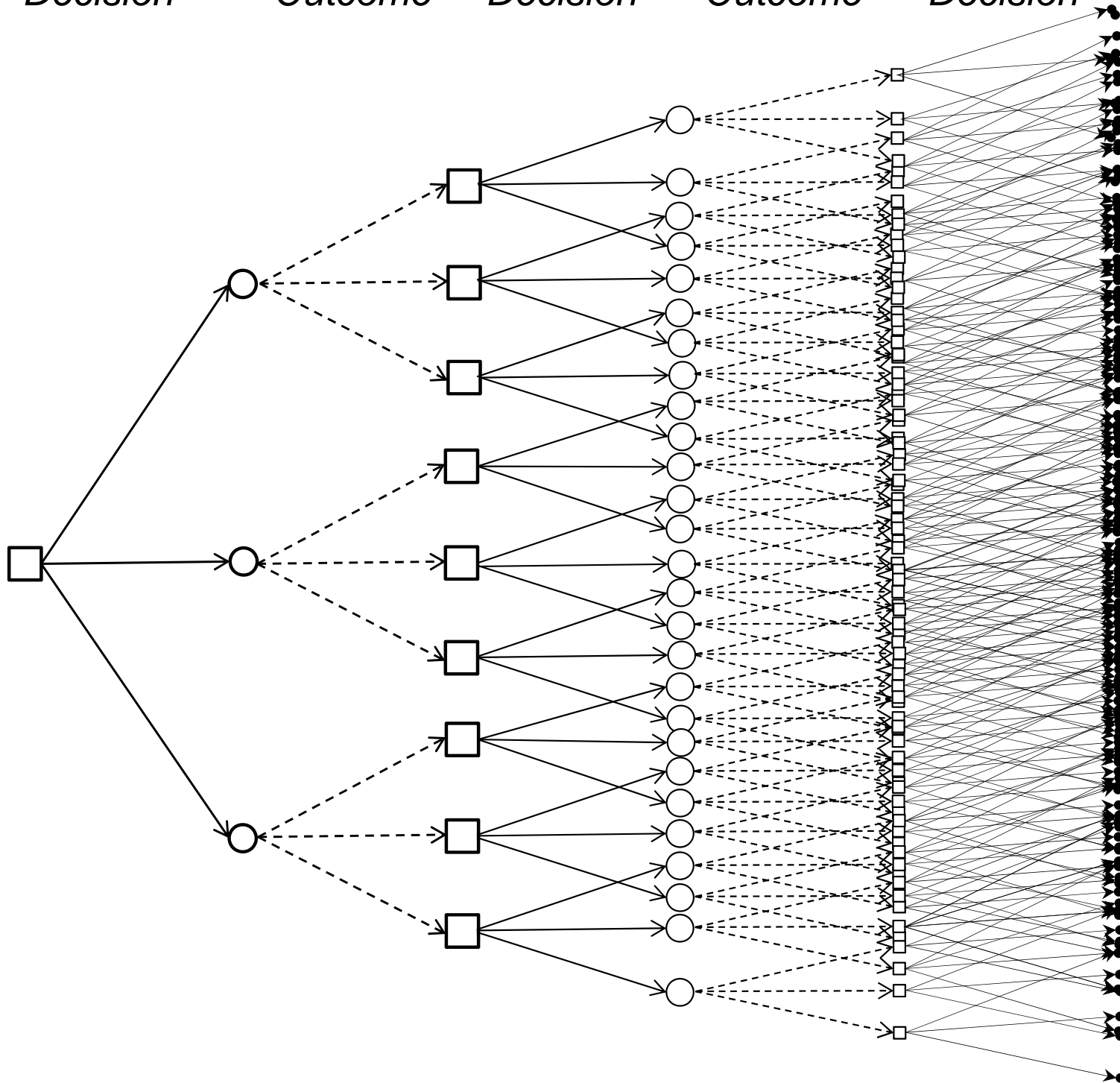
*Decision*

*Outcome*

*Decision*

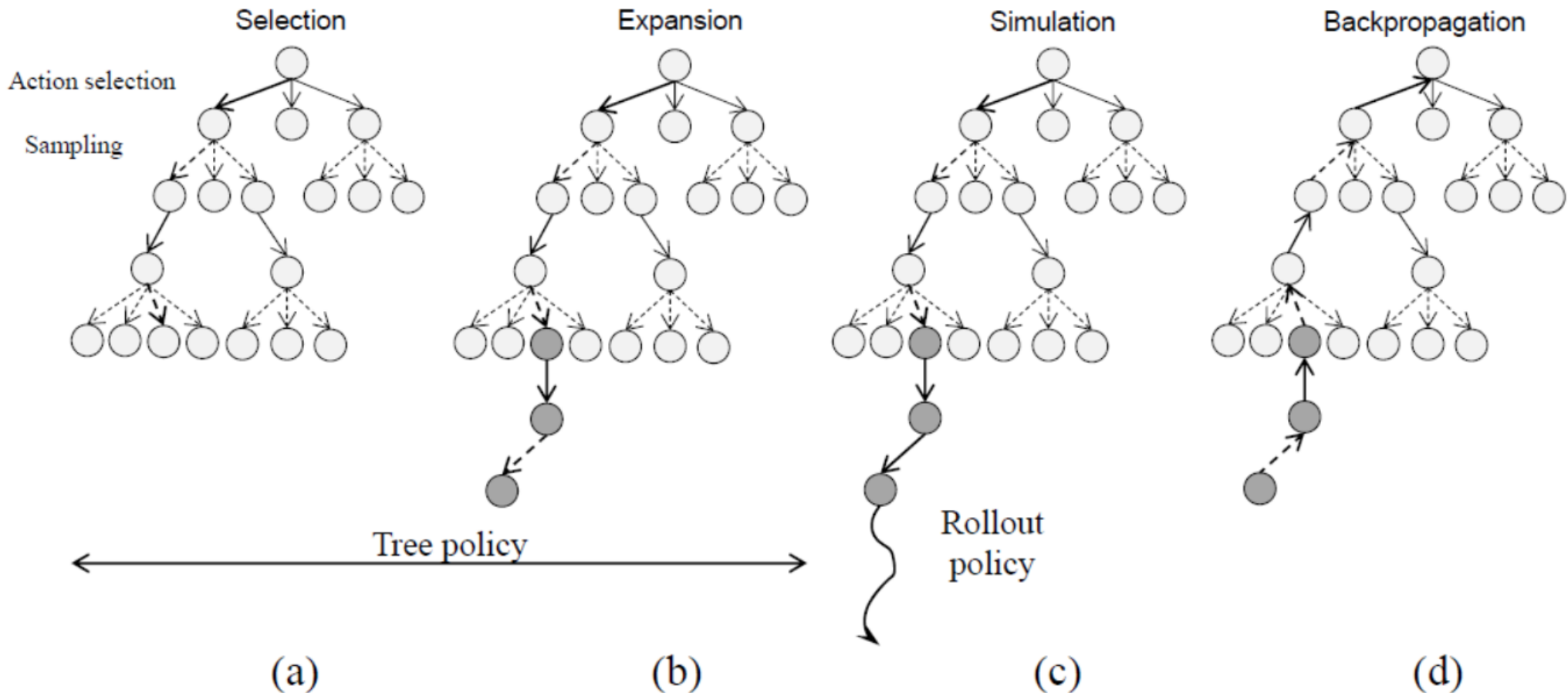
*Outcome*

*Decision*



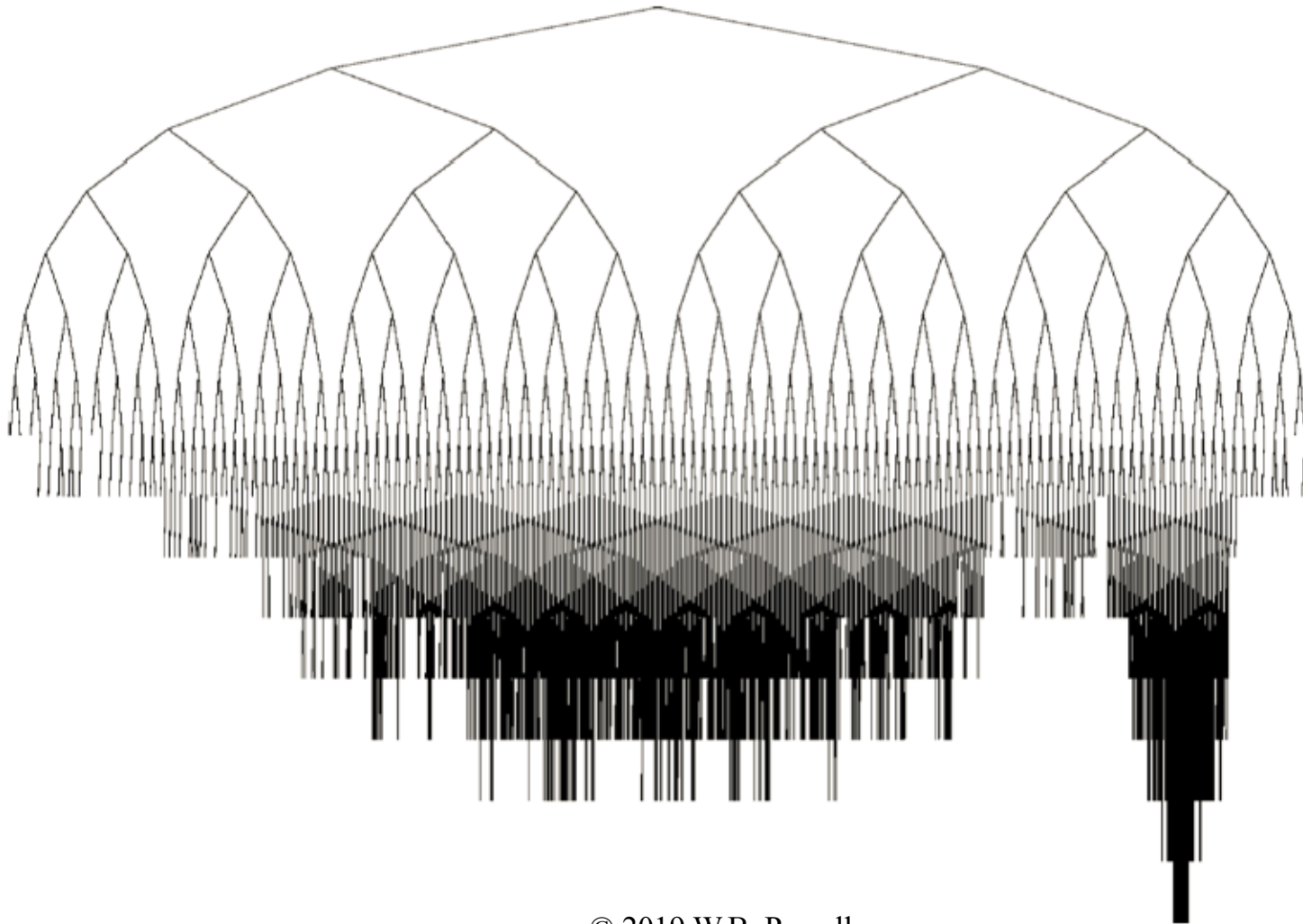
# Monte Carlo tree search

## ● Monte Carlo tree search:



# Monte Carlo tree search

- Monte Carlo tree search
  - » Explores some nodes more than others.





# Monte Carlo tree search

## ● Elements of MCTS:

» Selection – Choose decision and random outcome:

- Decision – Use “upper confidence bounding fore trees”, UCT, to decide which action to take

$$A_{u'}^{UCT}(\tilde{S}_{u'}|\theta^{UCT}) = \arg \max_{\tilde{a} \in \mathcal{A}_{u'}} \left( (C(\tilde{S}_{u'}, \tilde{a}) + \tilde{V}_{u'}^{\tilde{a}}(\tilde{S}_{u'}^{\tilde{a}})) + \theta^{UCT} \sqrt{\frac{\ln N(\tilde{S}_{u'})}{N(\tilde{S}_{u'}, \tilde{a}_{u'})}} \right)$$

- Requires enumerating all actions.
- Depends on having estimate of the value of the downstream state.
- Sampling the outcome
  - Use Monte Carlo simulation to sample our way to the next pre-decision state.

# Monte Carlo tree search

- Choosing an action:

$$x_t = \arg \max_x \left[ \underbrace{\left( C(s_t, x) + \hat{V}_{t+1}(s_{t+1}) \right)}_{\text{Downstream value}} + \theta \sqrt{\frac{\log N(s_t)}{N(s_t, x)}} \right]$$

No. times visited node  $s_t$

No. times tested action  $x$  from  $s_t$

$\hat{V}_{t+1}(s_{t+1})$

**Expansion**

The diagram illustrates the expansion step of Monte Carlo tree search. A tree structure is shown with nodes represented by circles. A node  $s_t$  is highlighted with a solid arrow pointing to it from a box labeled "No. times visited node  $s_t$ ". From  $s_t$ , an action  $x_t$  is chosen, leading to a new node  $s_{t+1}$ , which is highlighted with a solid arrow from a box labeled "No. times tested action  $x$  from  $s_t$ ". The node  $s_{t+1}$  is then expanded into a subtree of nodes, shown with dashed lines. A dashed arrow points from  $s_{t+1}$  to a terminal state, with the value  $\hat{V}_{t+1}(s_{t+1})$  indicated by a dashed line.

- » Downstream value  $\hat{V}_{t+1}(s_{t+1})$  is obtained by simulating some policy (not too complicated) to “peek” into the future.

# Monte Carlo tree search

---

## ● Elements of MCTS:

### » Expansion

- If action taken goes to a state we have already generated, then update the value of being in the state we are searching from.
- If action takes us to a new state, then we add the post-decision node, and sample our way to the next pre-decision node.
- Use our rollout policy to get an initial estimate of the value of being in a state.

### » Simulation

- We assume we have access to some default policy that we can use to simulate our way forward to get an estimate of the value of being in a state.
- This is known under different names, but we use the term “rollout policy” first introduced by Bertsekas.
- Rollout policy is typically suboptimal, but it is possible to produce an optimistic estimate by optimizing over a sampled future.

# Monte Carlo tree search

---

## ● Elements of MCTS:

### » Backpropagation

- After using our rollout policy to step forward, we then do a backward traversal (“backpropagation”) to update all upstream value functions.
- Use standard smoothing to update values.

# Monte Carlo tree search

## ● Notes:

- » Even with discrete actions, decision trees explode in size extremely quickly.
- » Monte Carlo tree search is a method for generating the most promising parts of the trees.
- » The key is the availability of a simple rollout policy that does a “good” job of approximating an optimal policy.
- » The UCB policy guarantees asymptotic optimality because it ensures that each action will be tested infinitely often:

$$x_t = \arg \max_x \left[ \left( C(s_t, x) + \hat{V}_{t+1}(s_{t+1}) \right) + \theta \sqrt{\frac{\log N(s_t)}{N(s_t, x)}} \right]$$

... this keeps growing.  
If we do not try  $x$ ...

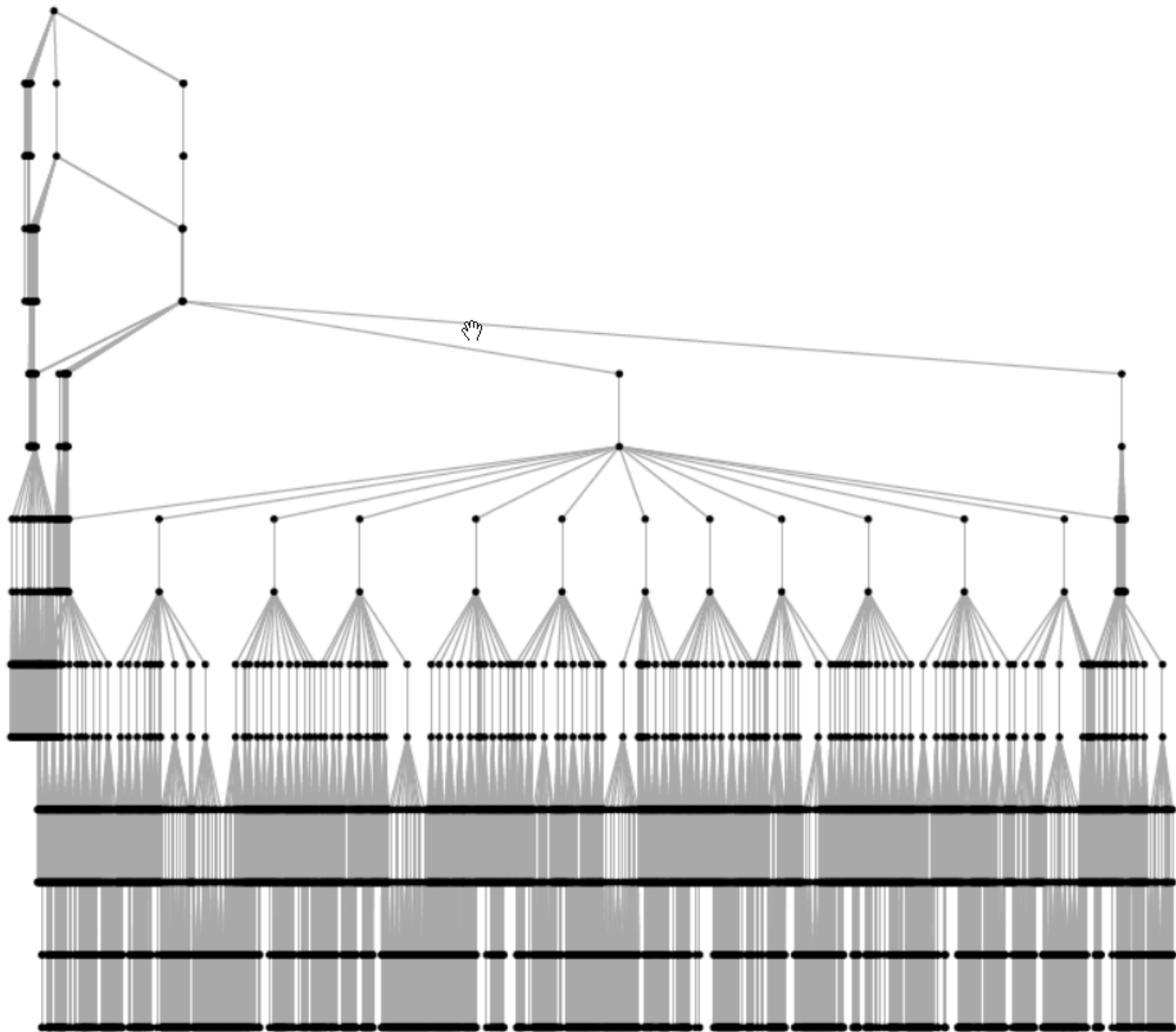
- » ... but MCTS does not work well with large sets of actions.

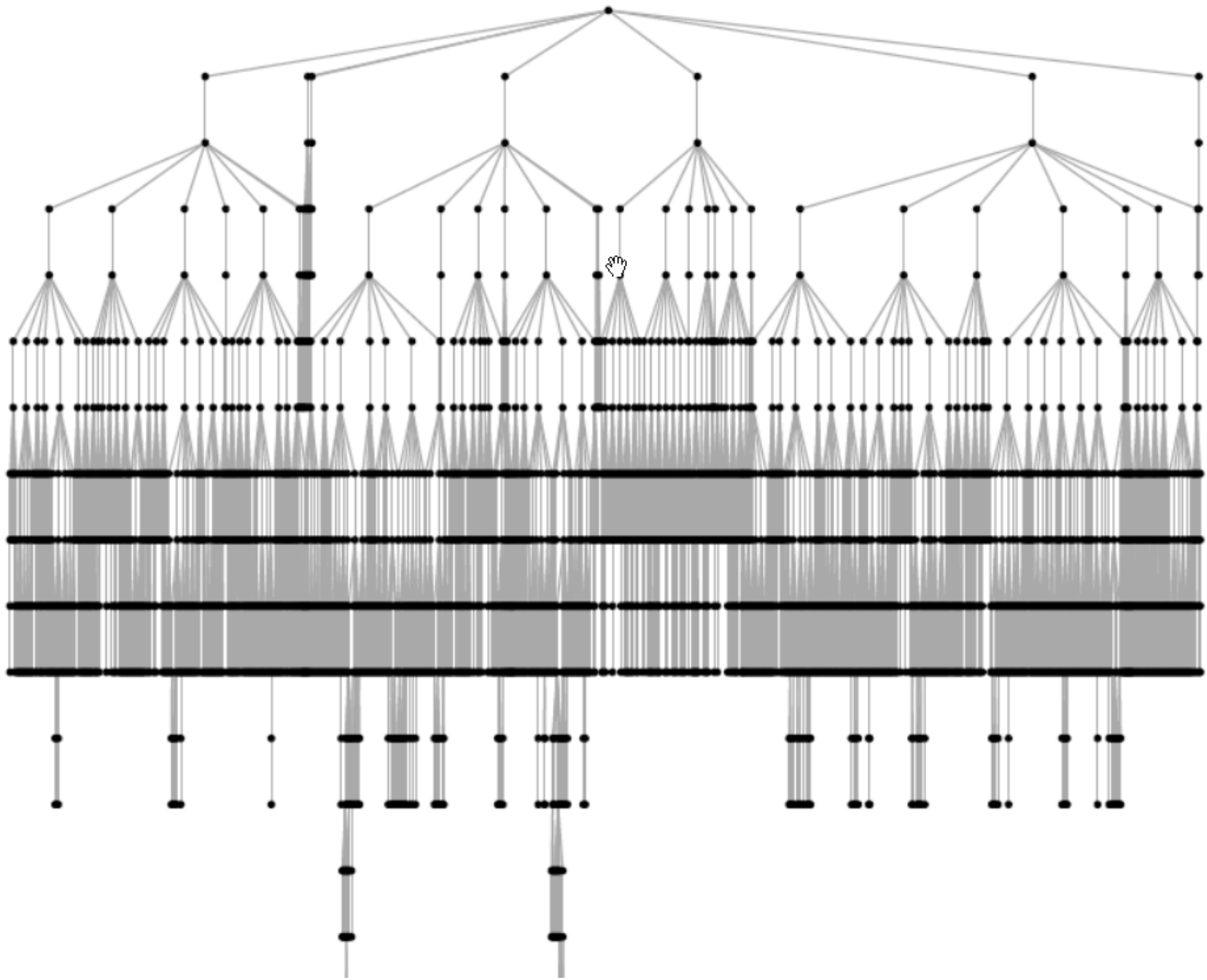
# Monte Carlo tree search

---

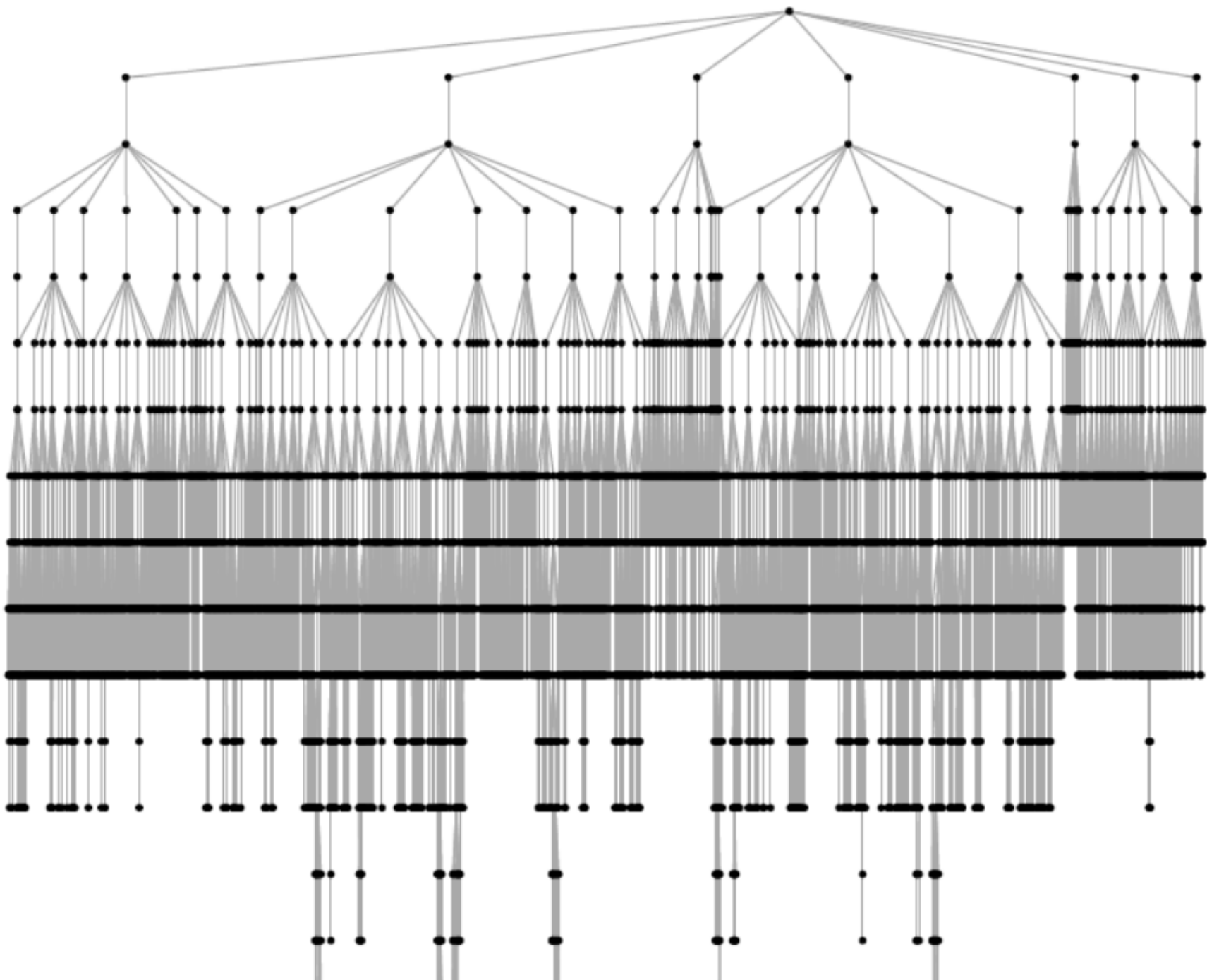
## ● Notes:

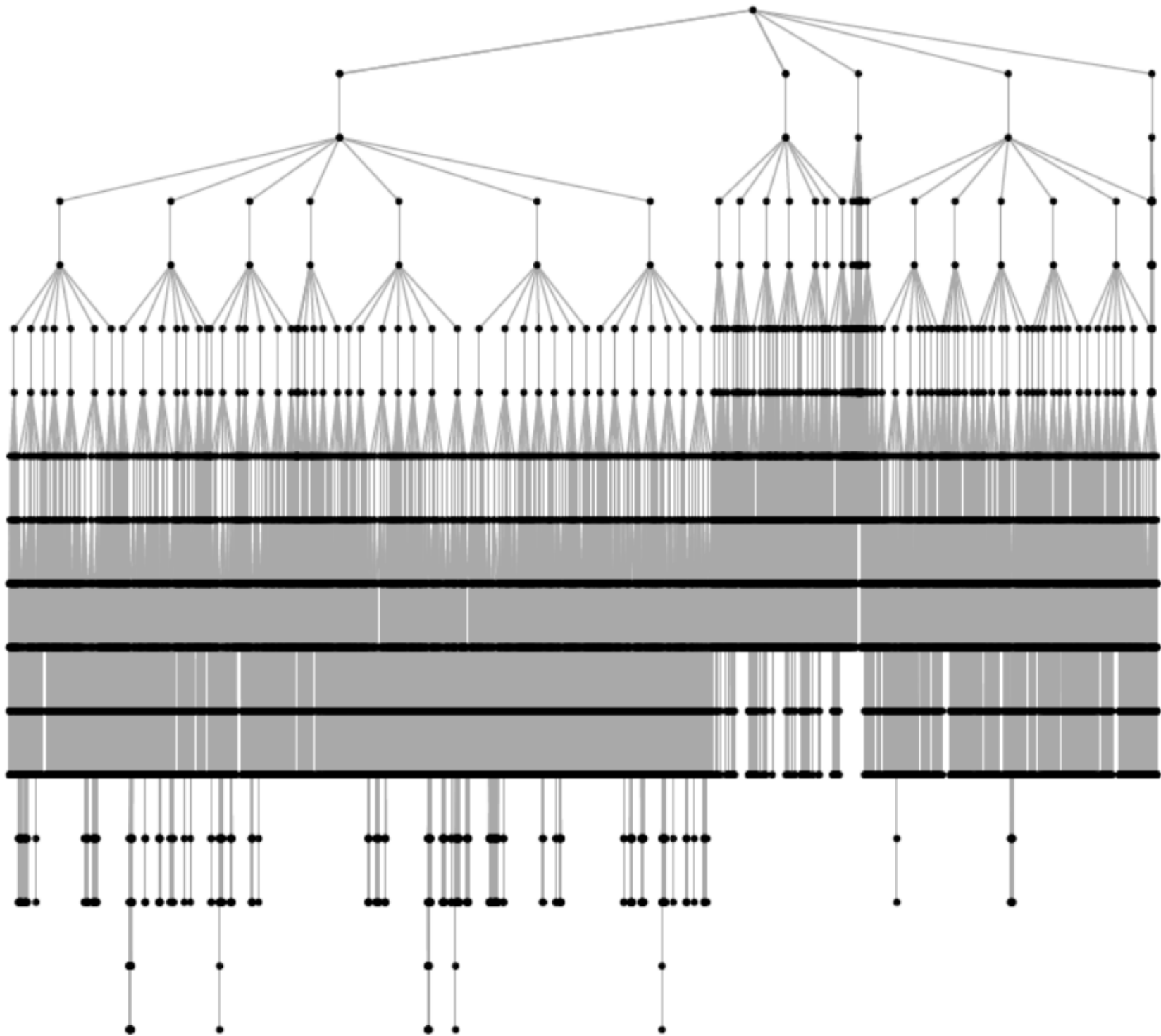
- » MCTS is immune to large state spaces. This means that it can easily handle the very large belief state.
- » MCTS is sensitive to large numbers of decisions.
- » You can use action-sampling strategies to handle large action sets...
- » ... or try “sampled lookahead” policies (sample the future, and solve a larger optimization problem).
- » Since it is exploring a substantial part of the tree, calculations need to be quite fast, which means that Bayesian updating within the search may be too expensive.











# Monte Carlo tree search

Case application – emergency storm  
response

# Case study: MCTS for storm response

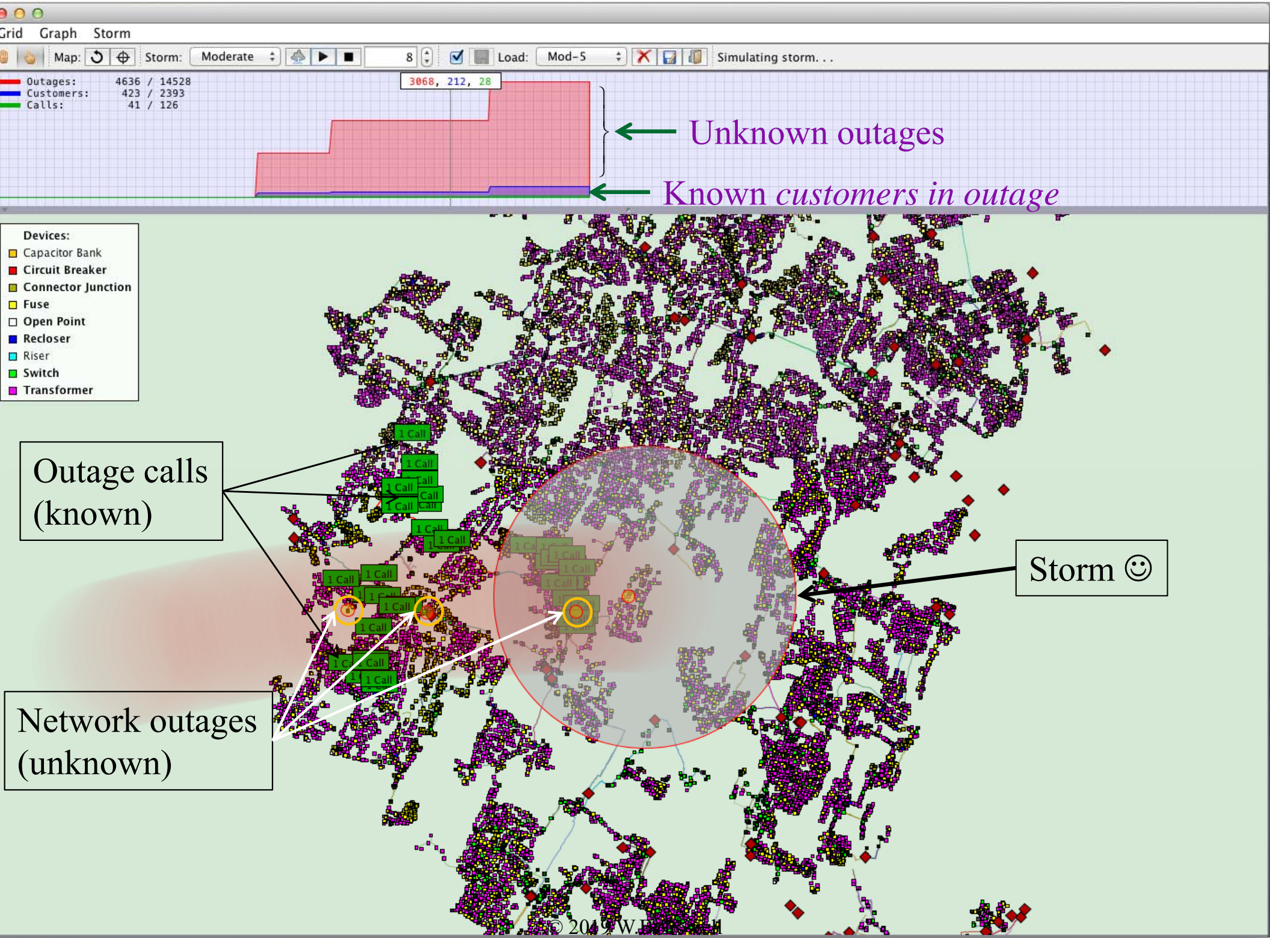


- Hurricane Sandy
  - » Once in 100 years?
  - » Rare convergence of events
  - » But, meteorologists did an amazing job of forecasting the storm.

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







Outages: 4636 / 14528  
Customers: 423 / 2393  
Calls: 41 / 126

3068, 212, 28

Unknown outages

Known customers in outage

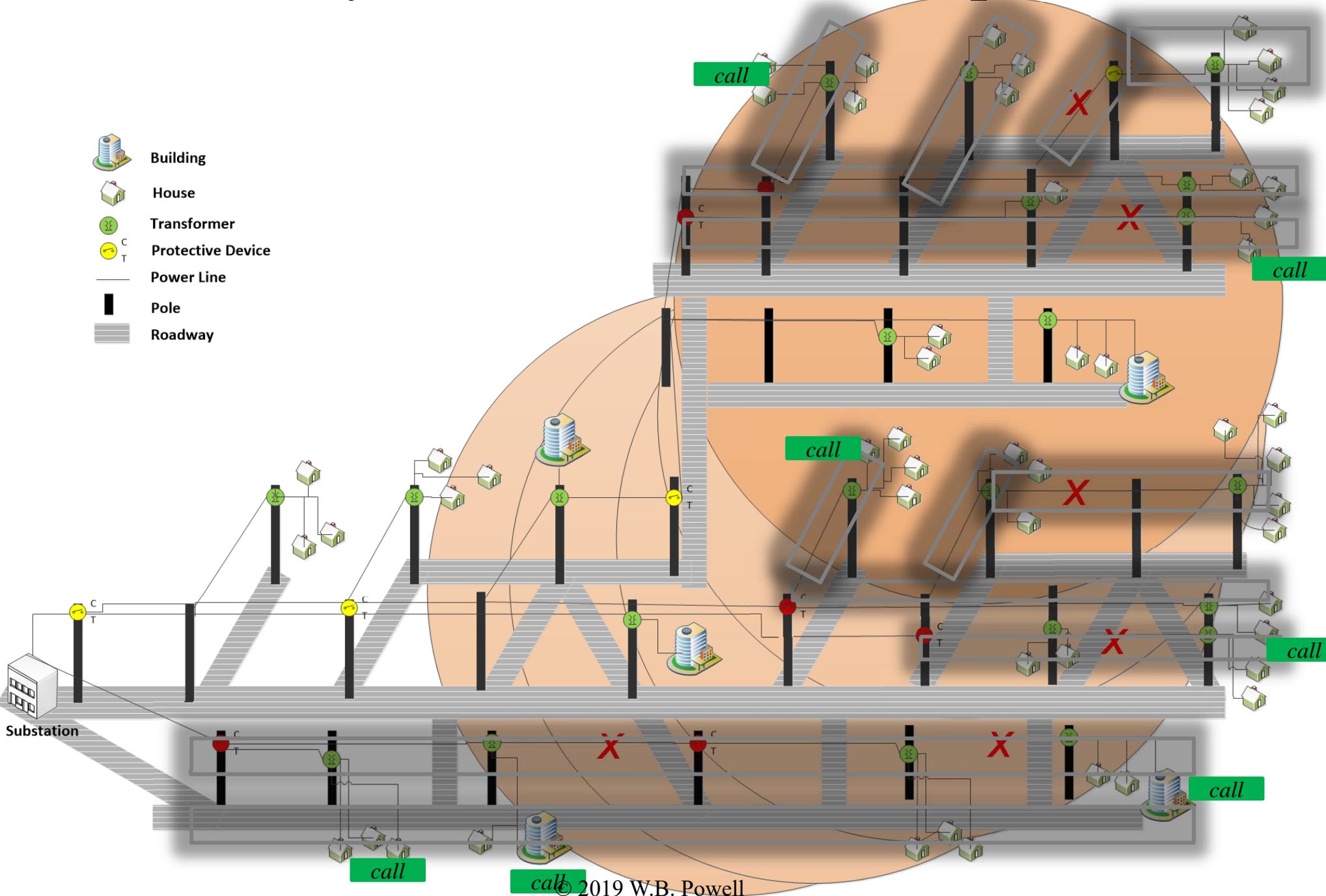
- Devices:
- Capacitor Bank
  - Circuit Breaker
  - Connector Junction
  - Fuse
  - Open Point
  - Recloser
  - Riser
  - Switch
  - Transformer

Outage calls (known)

Network outages (unknown)








Storm ☺

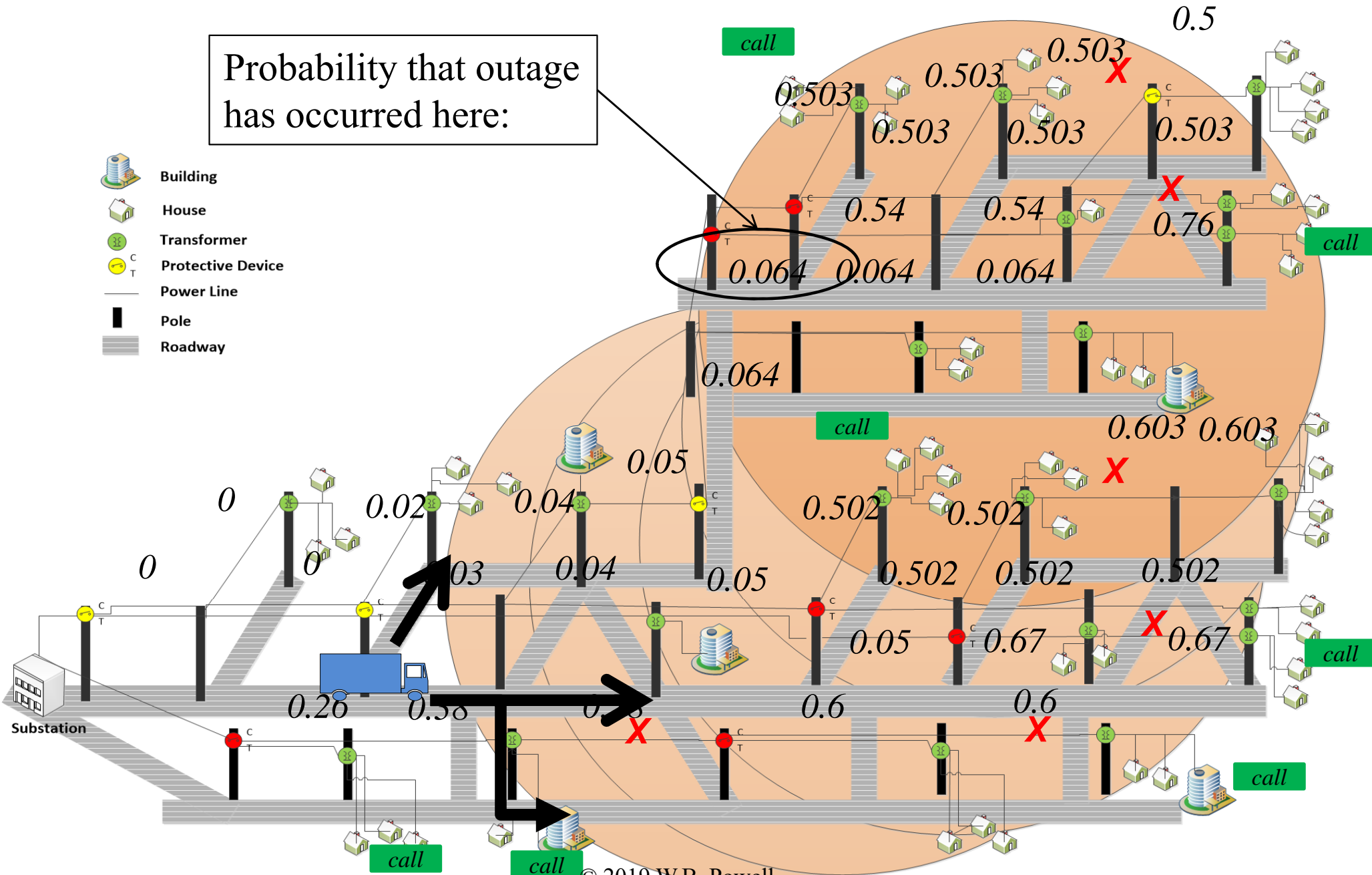
# Case study: MCTS for storm response



# Case study: MCTS for storm response

Probability that outage has occurred here:








-  Building
-  House
-  Transformer
-  Protective Device
-  Power Line
-  Pole
-  Roadway

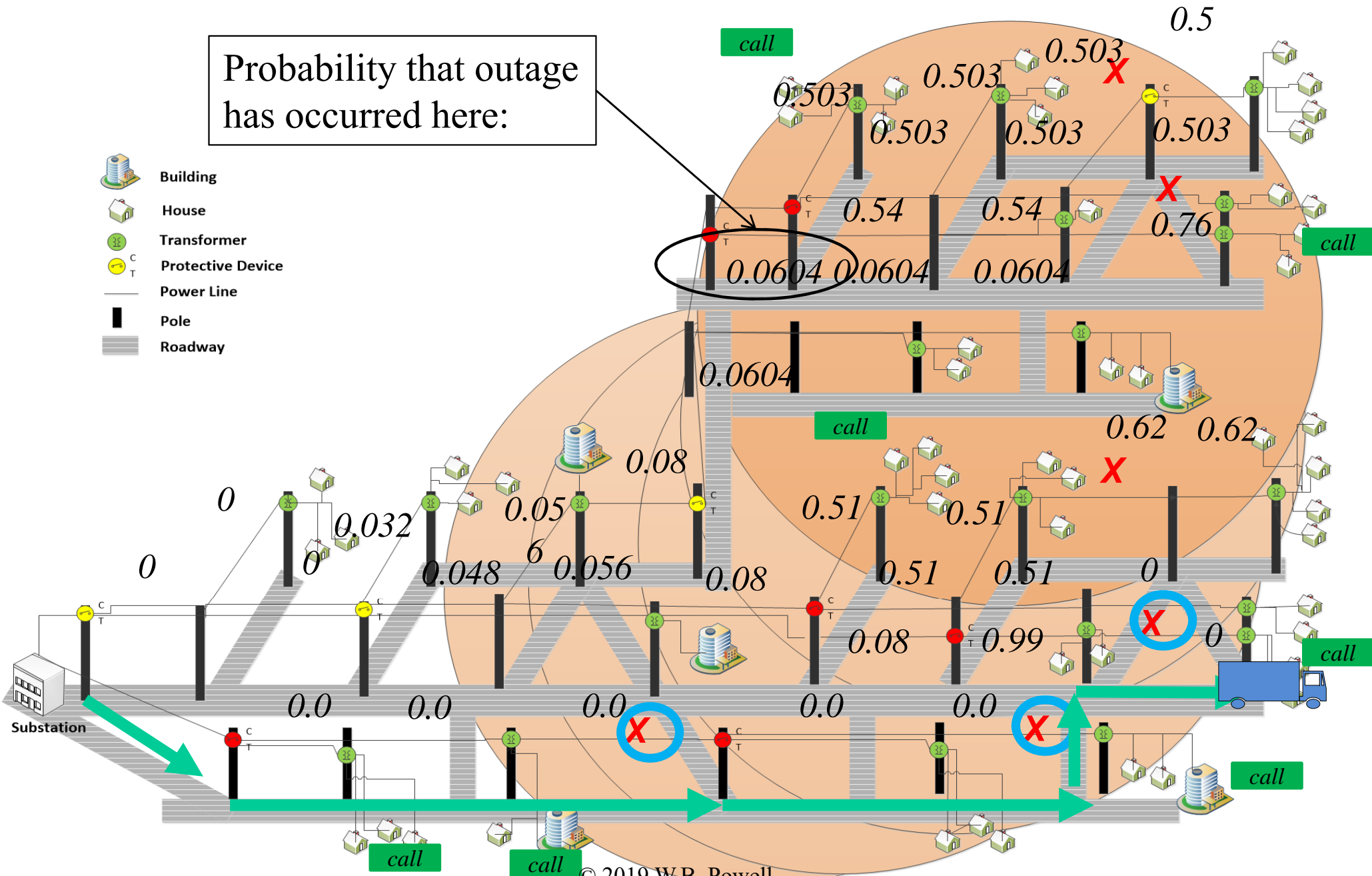




# Case study: MCTS for storm response

Probability that outage has occurred here:








-  Building
-  House
-  Transformer
-  Protective Device
-  Power Line
-  Pole
-  Roadway

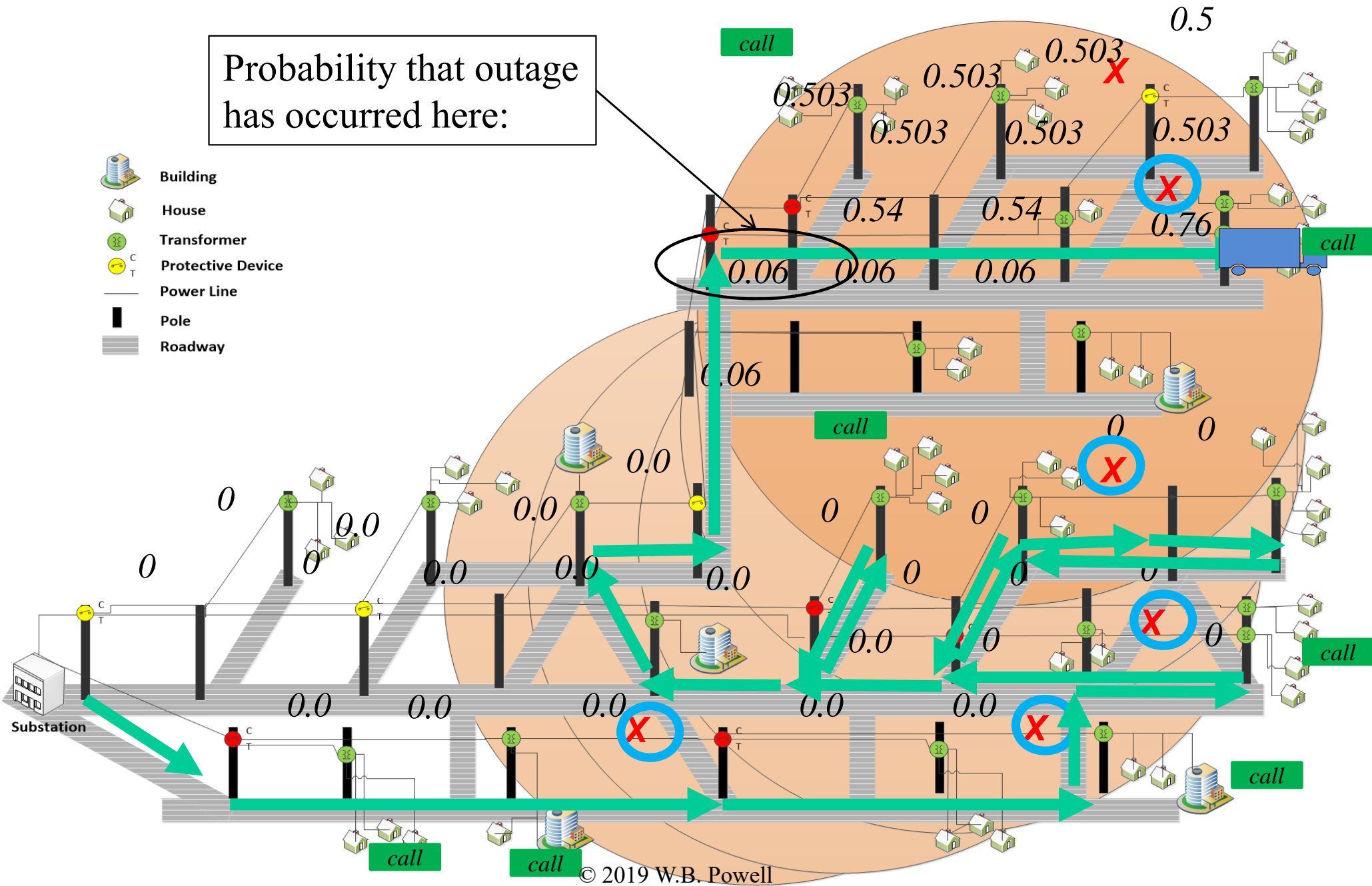




# Case study: MCTS for storm response

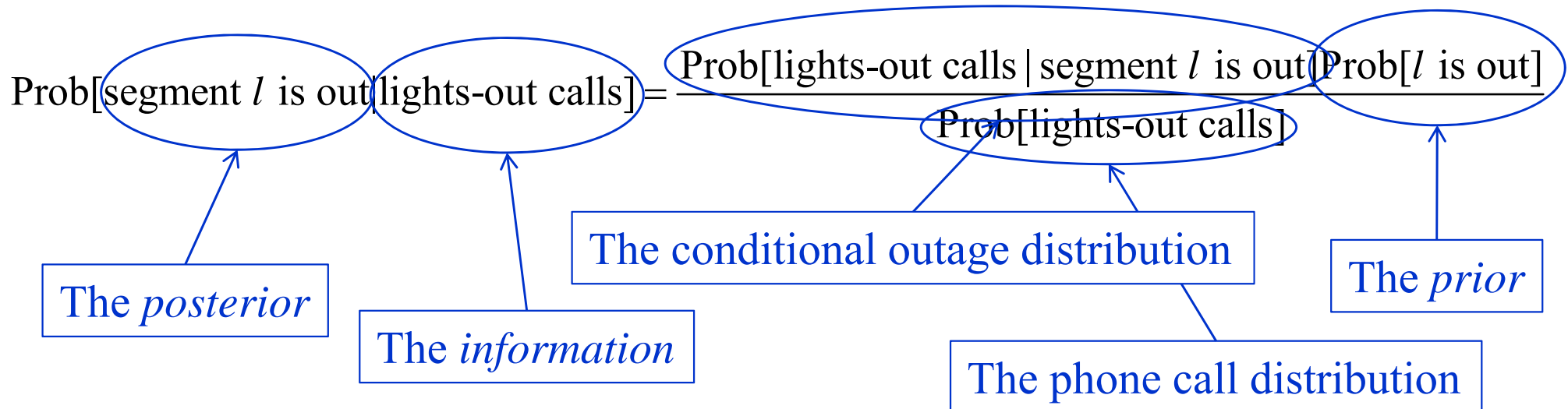
Probability that outage has occurred here:

-  Building
-  House
-  Transformer
-  Protective Device
-  Power Line
-  Pole
-  Roadway



# Case study: MCTS for storm response

- Exploiting the information from phone calls
  - » We have to blend the following....
    - What we knew before the phone calls came in – this is called the *prior belief*.
    - The outage calls – this is called *information*.
  - » ...to produce the updated estimates of outage – this is called *the posterior*.
- To compute this we have to use *Bayes theorem*:



# Case study: MCTS for storm response

---

## ● State variables:

### » Physical state

- Location/status of the truck
- Known state of the grid (from visiting segments)

### » Other information

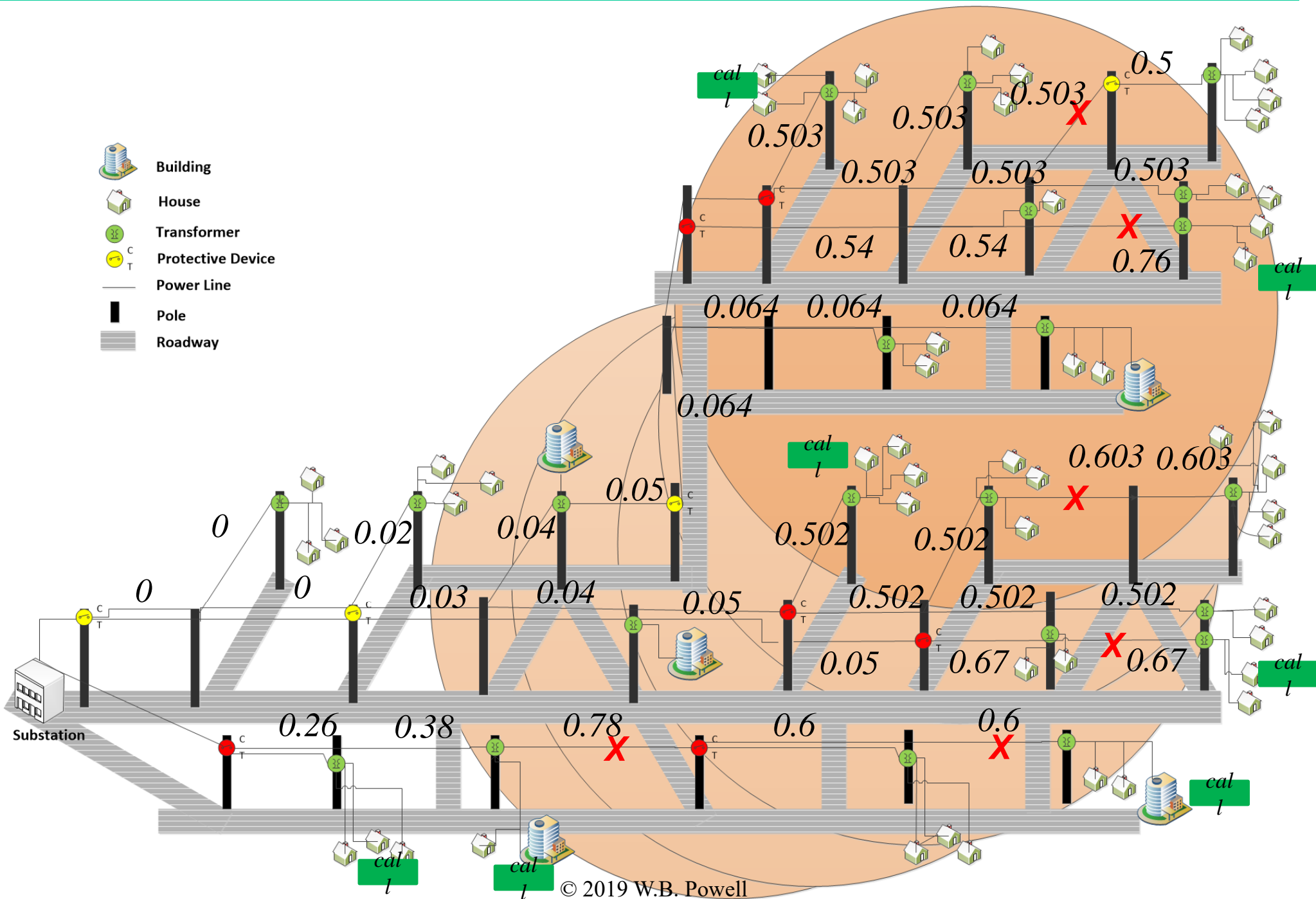
- Might be phone calls, weather forecast

### » Belief state

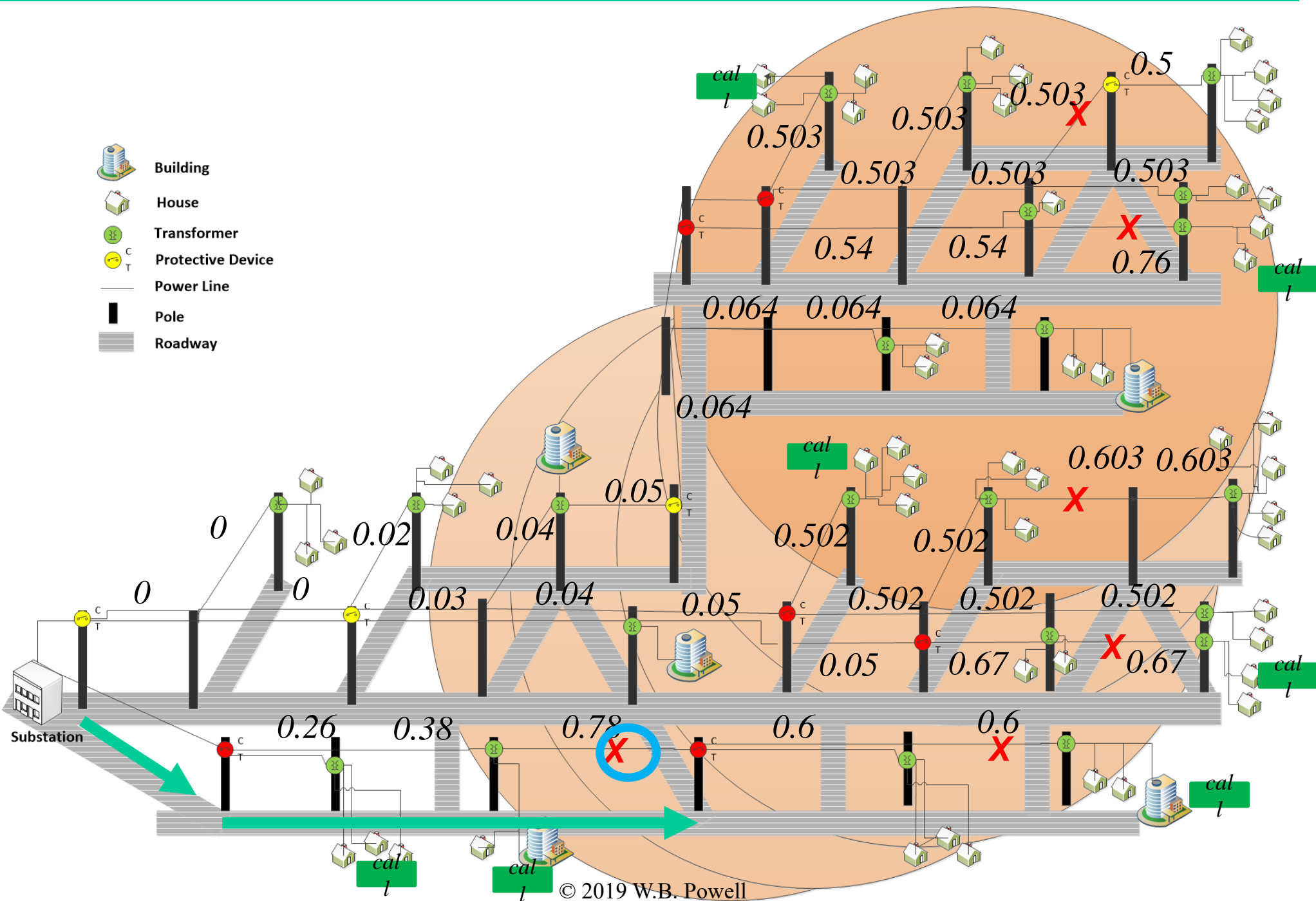
- Probabilities of outages on each unobserved link.

» State variable is high-dimensional, and continuous, but MCTS is not sensitive to the complexity of the state space. It is sensitive to the action space.

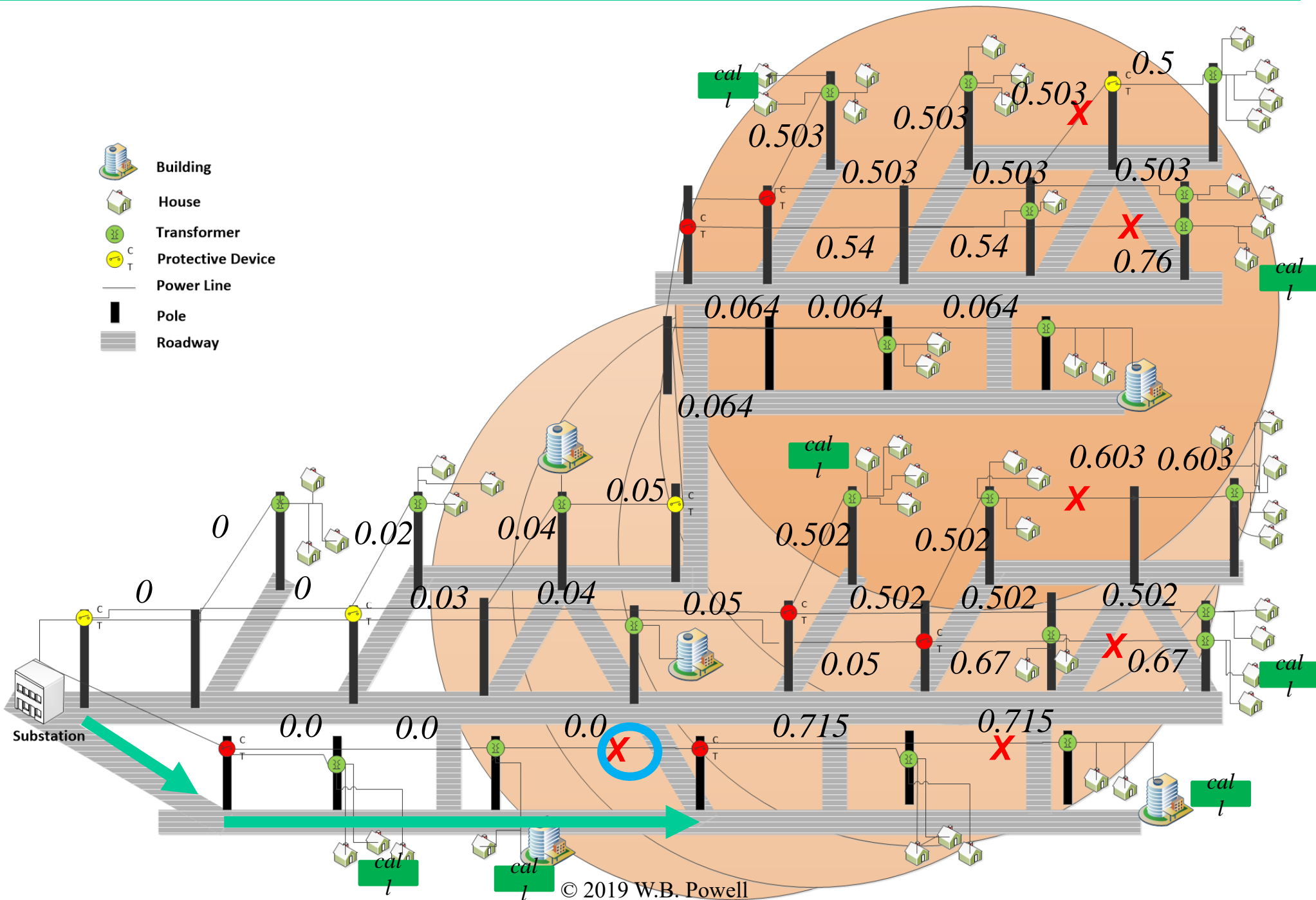
# Case study: MCTS for storm response



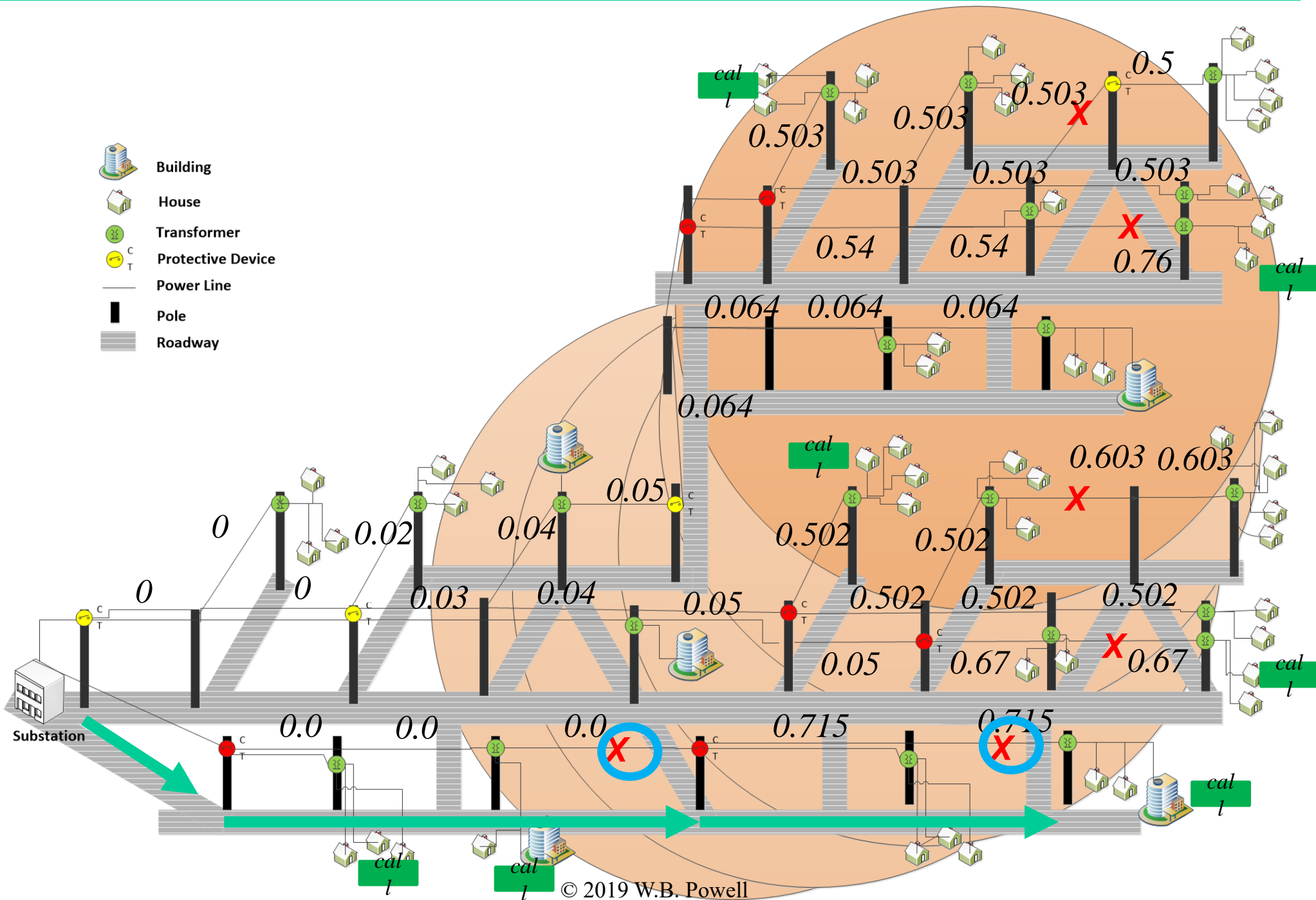
# Case study: MCTS for storm response



# Case study: MCTS for storm response

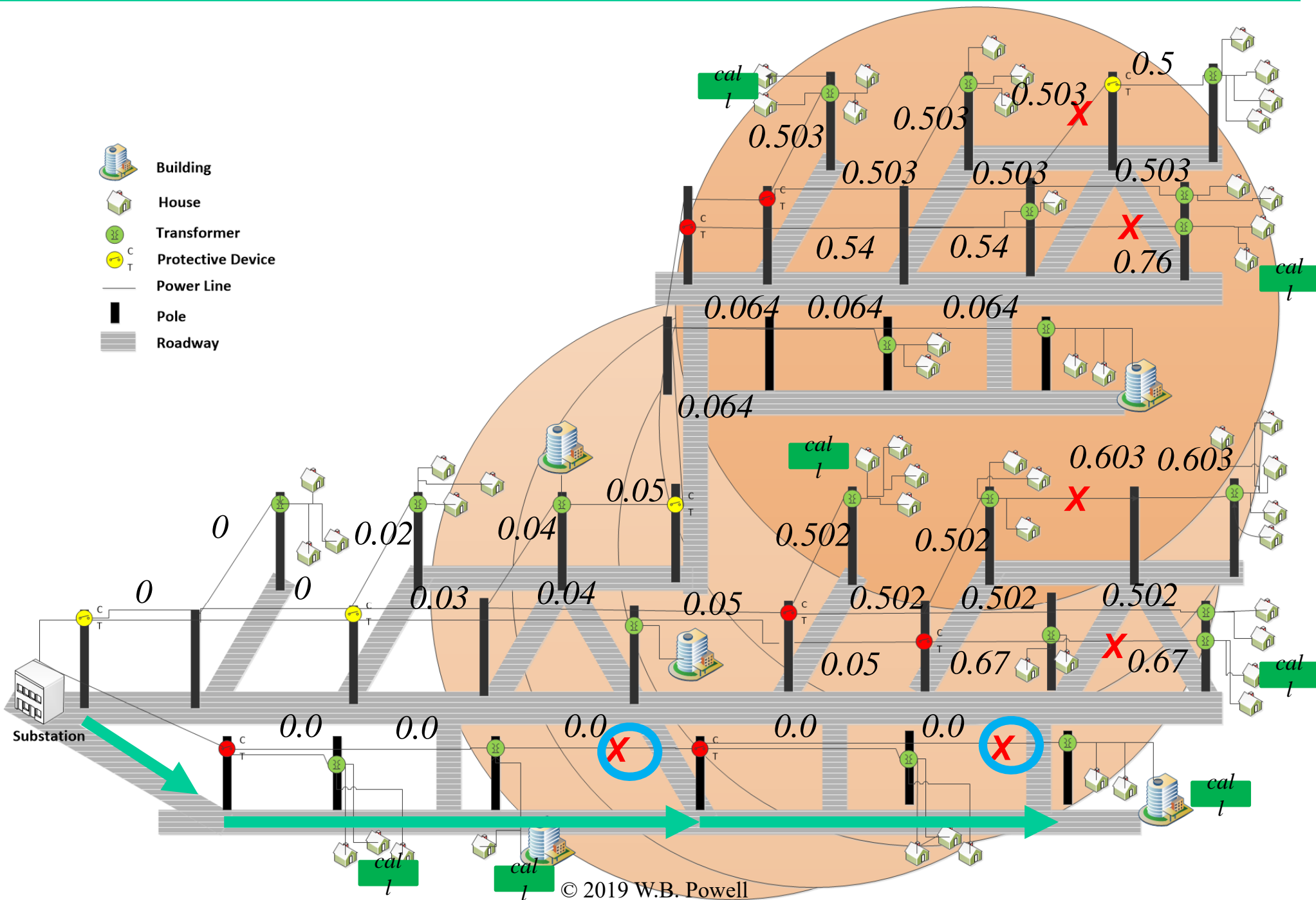


# Case study: MCTS for storm response



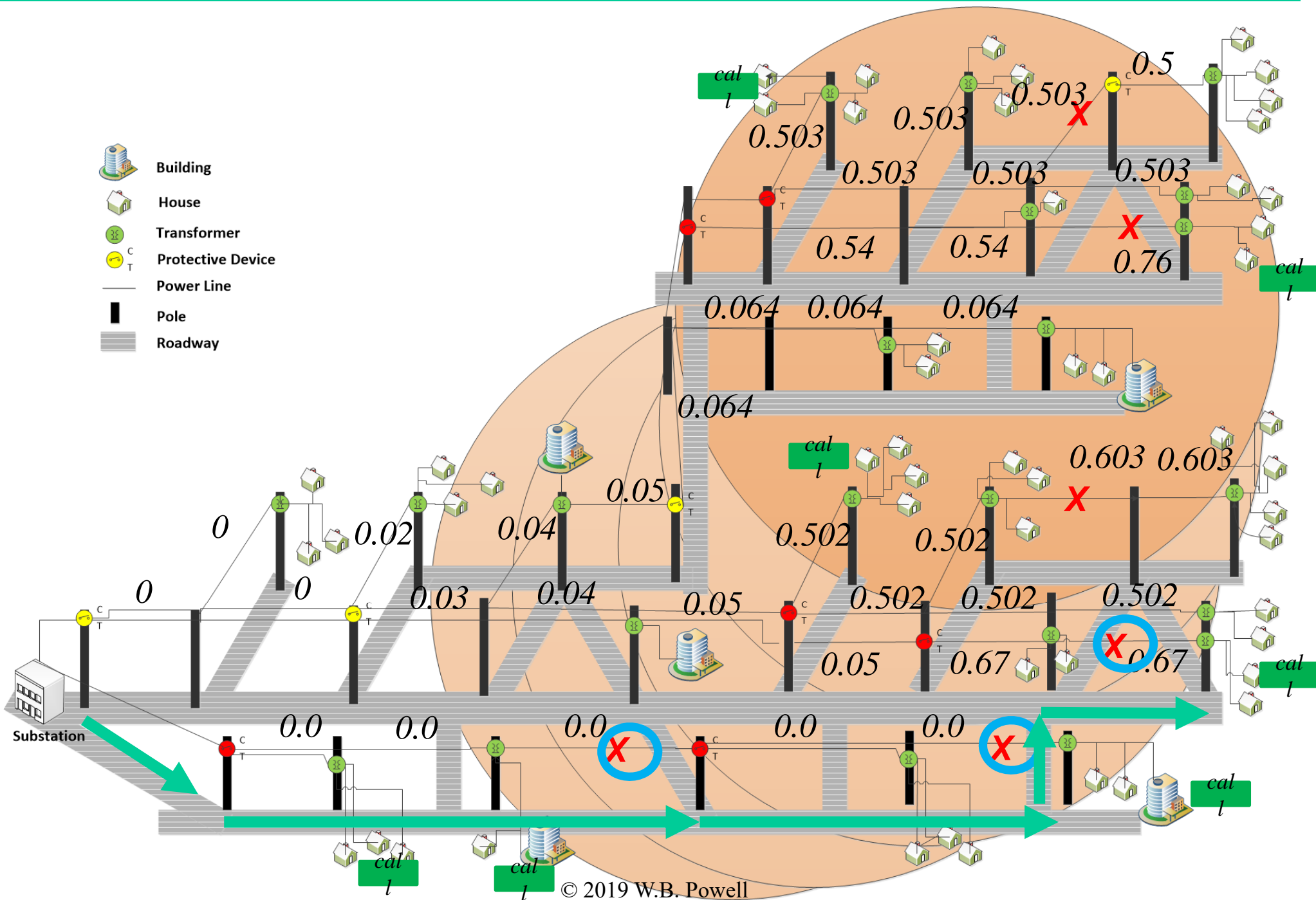


# Case study: MCTS for storm response

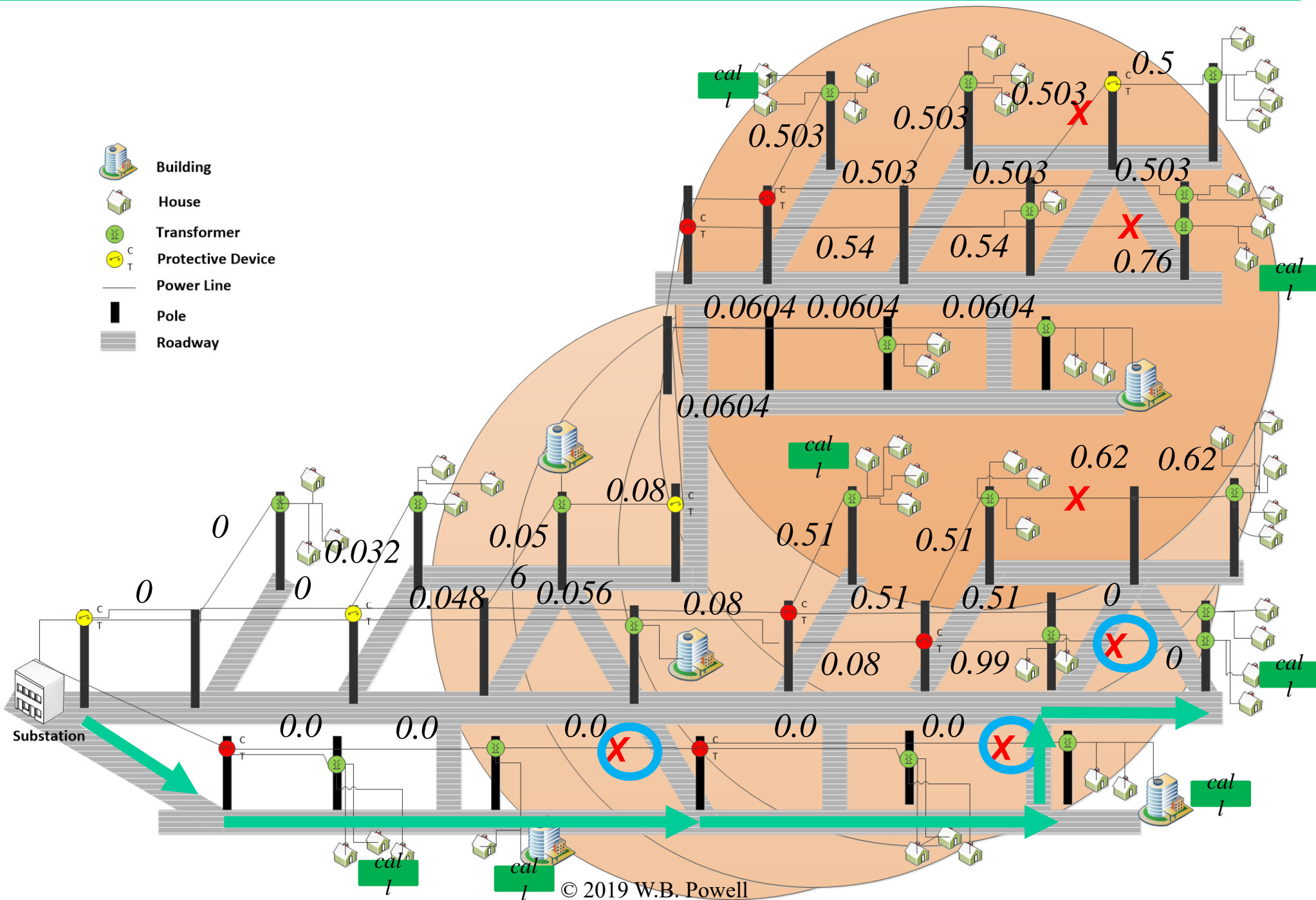




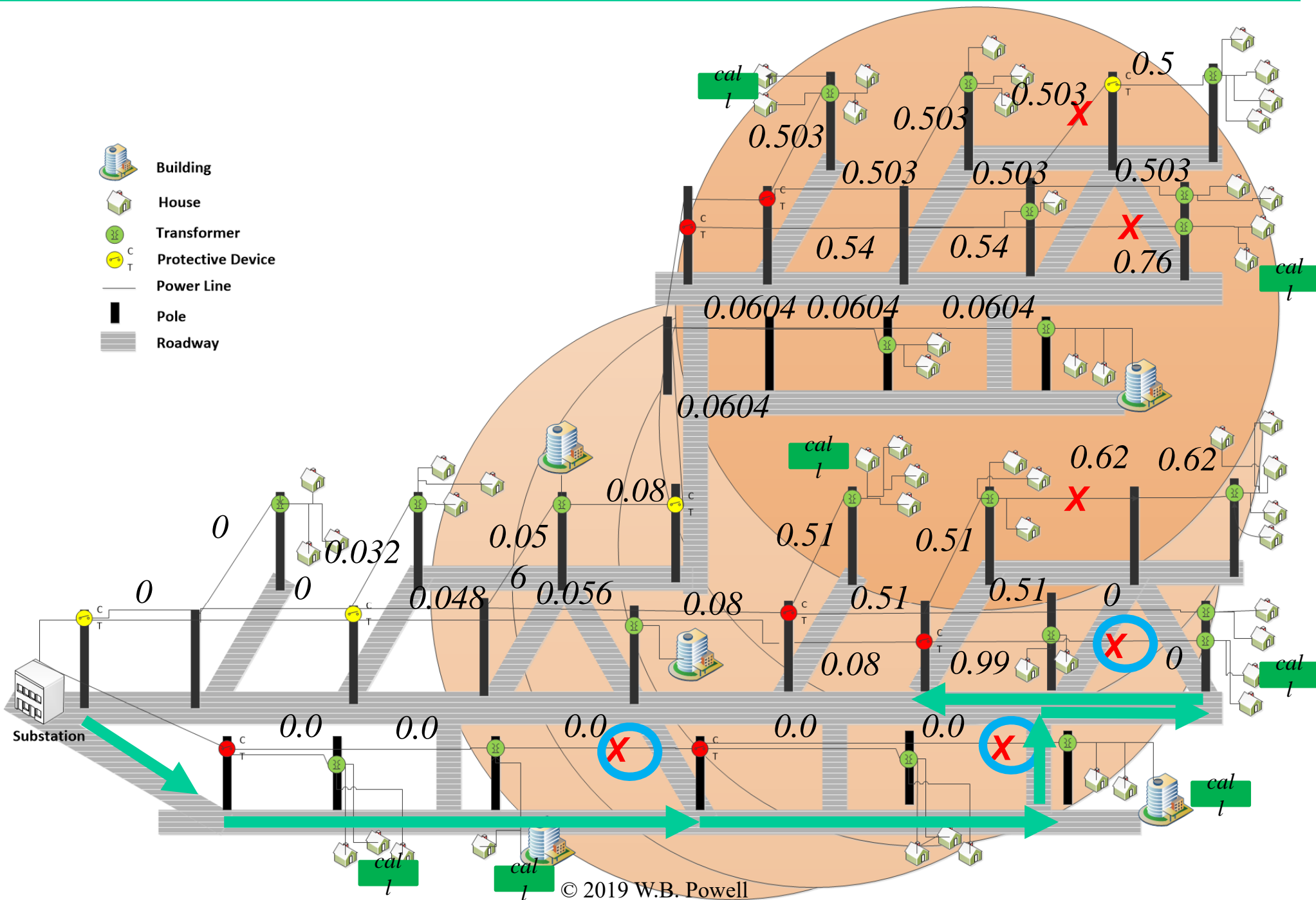
# Case study: MCTS for storm response



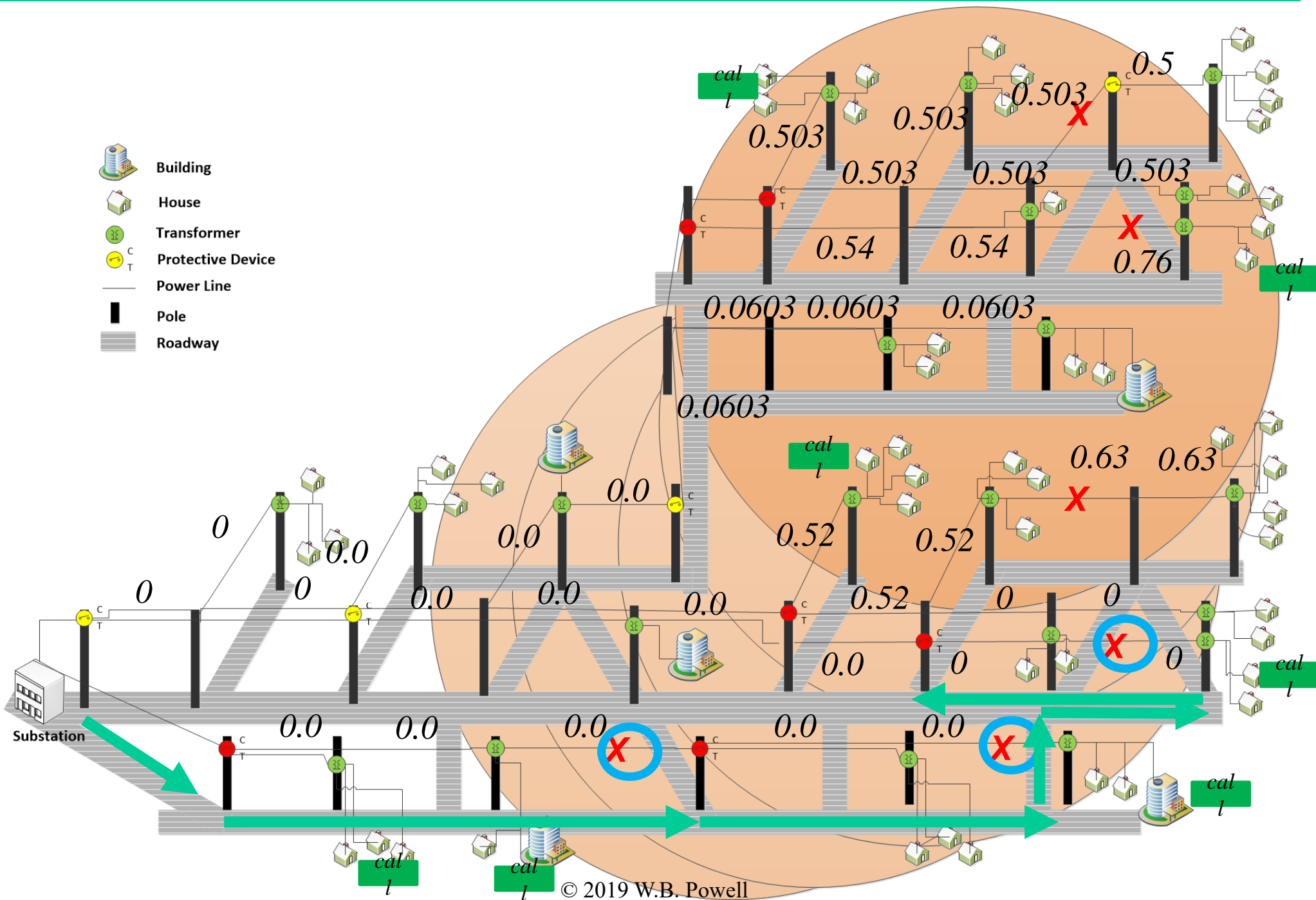
# Case study: MCTS for storm response



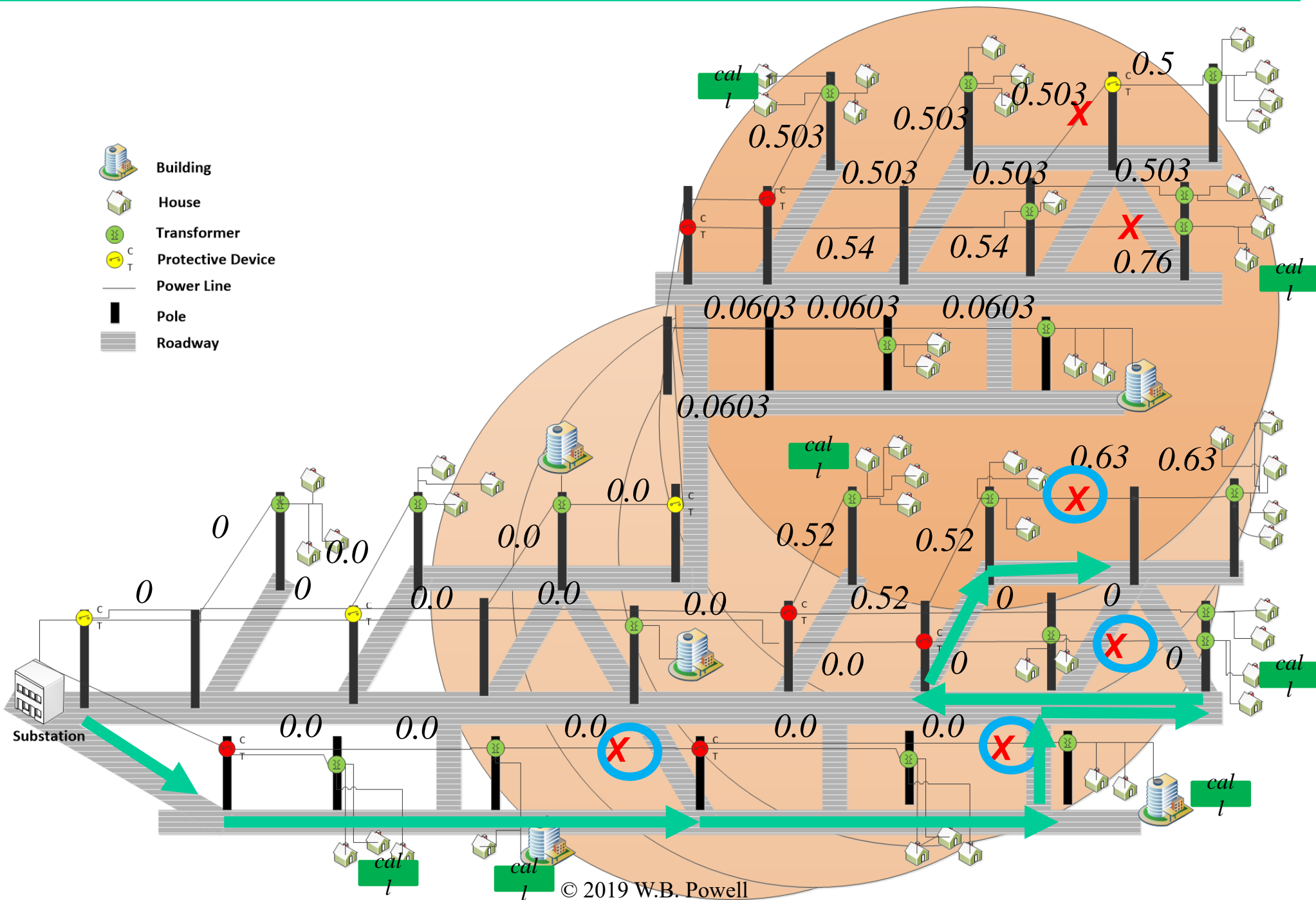
# Case study: MCTS for storm response



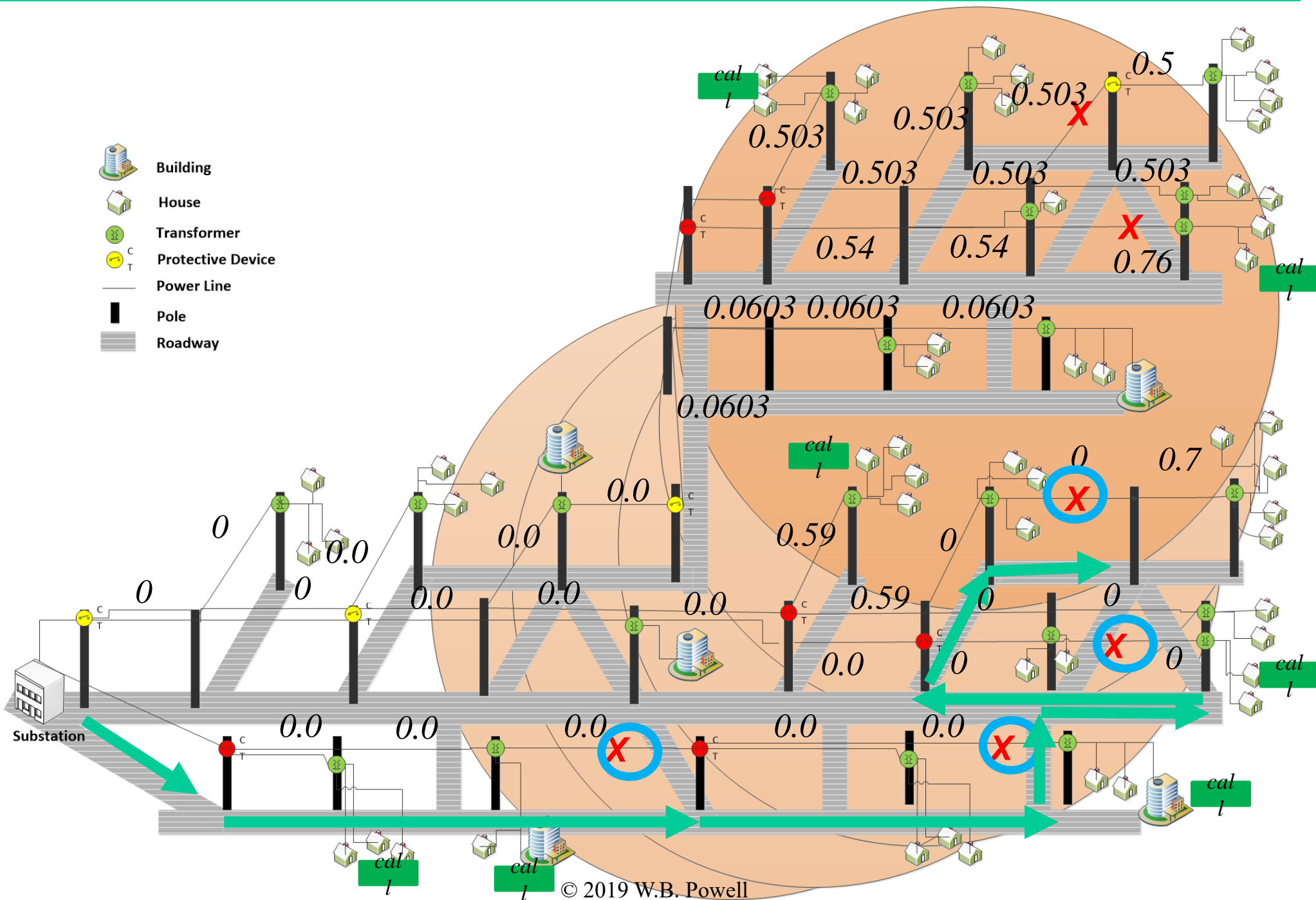
# Case study: MCTS for storm response



# Case study: MCTS for storm response

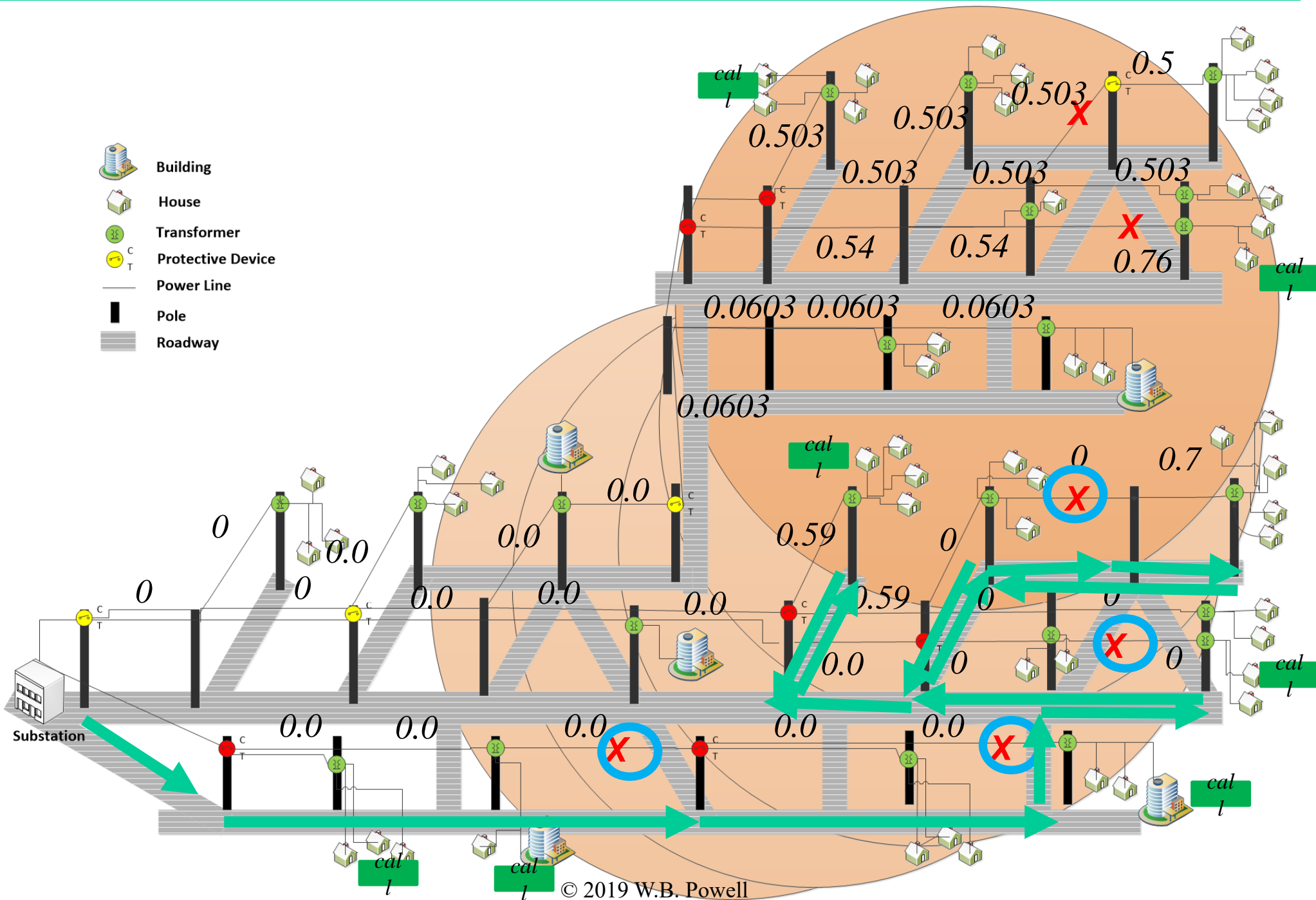


# Case study: MCTS for storm response

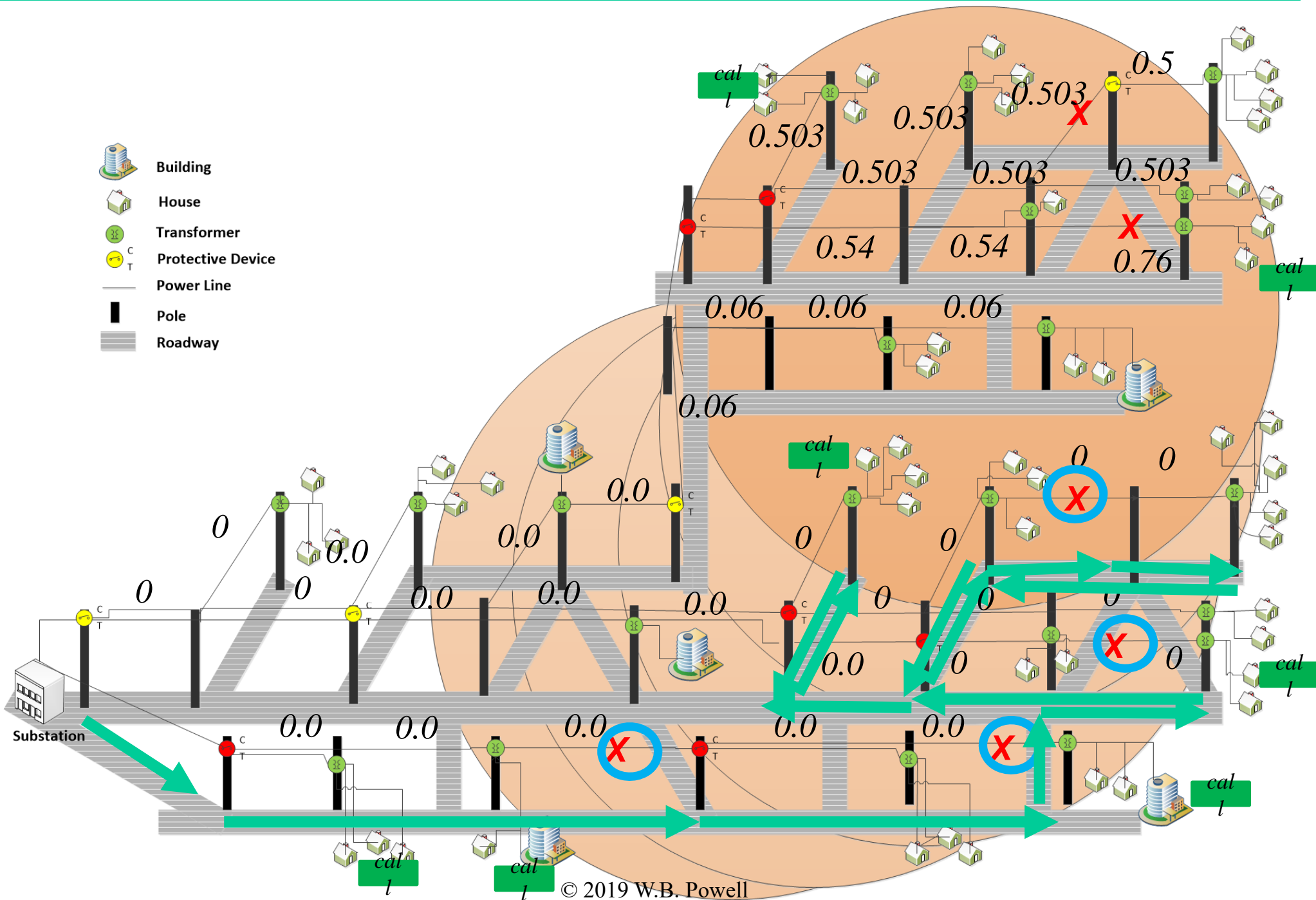




# Case study: MCTS for storm response

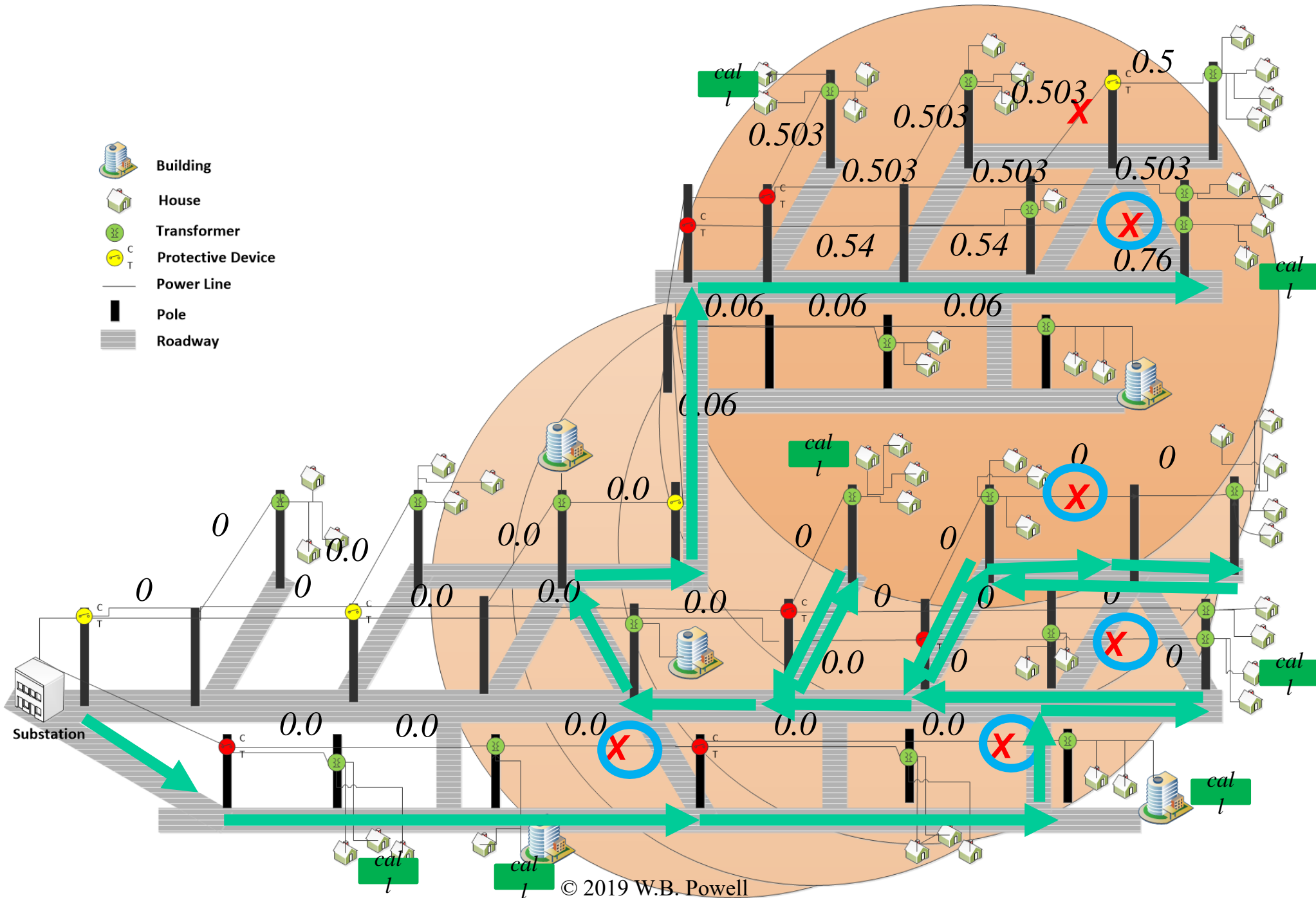


# Case study: MCTS for storm response

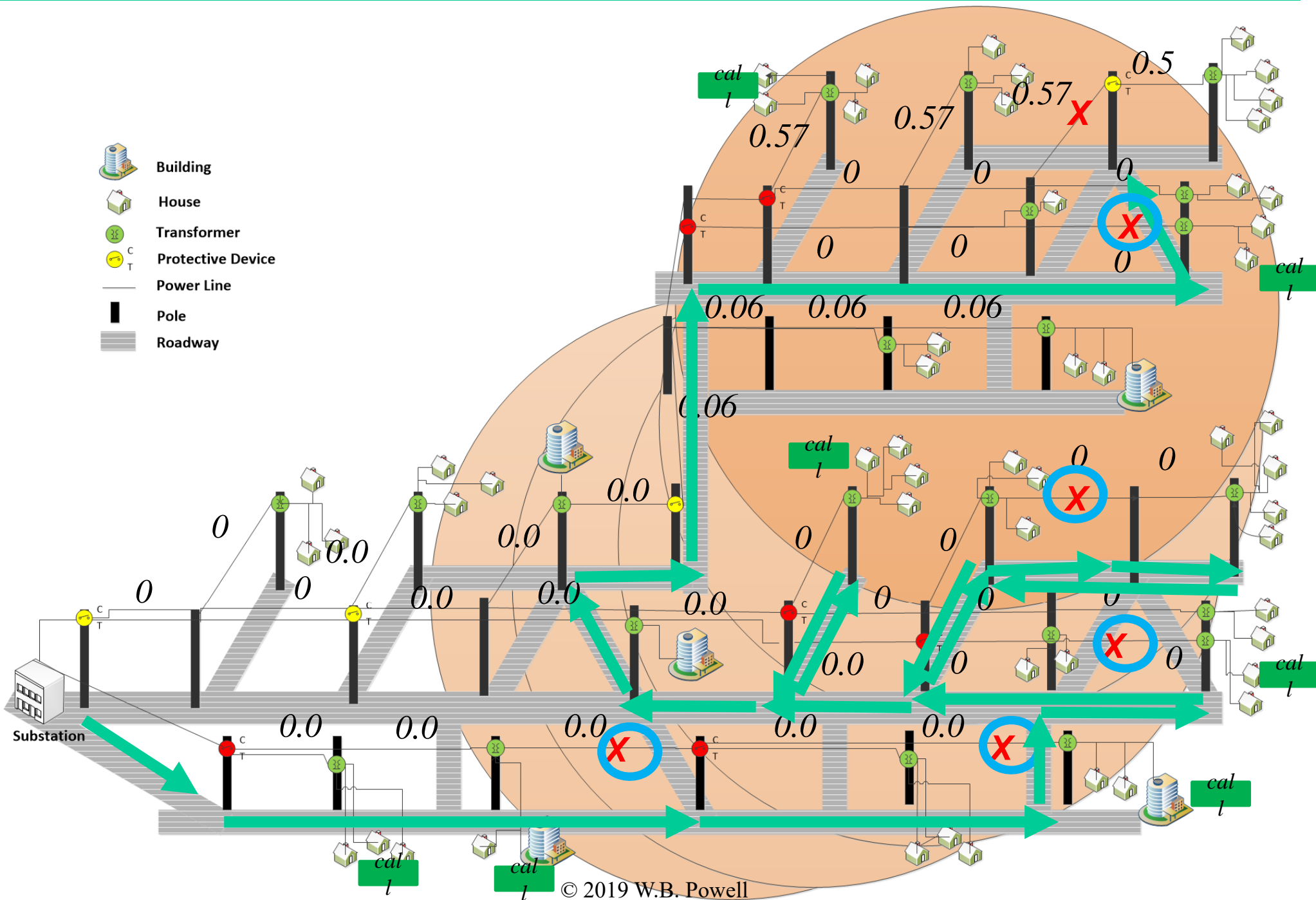




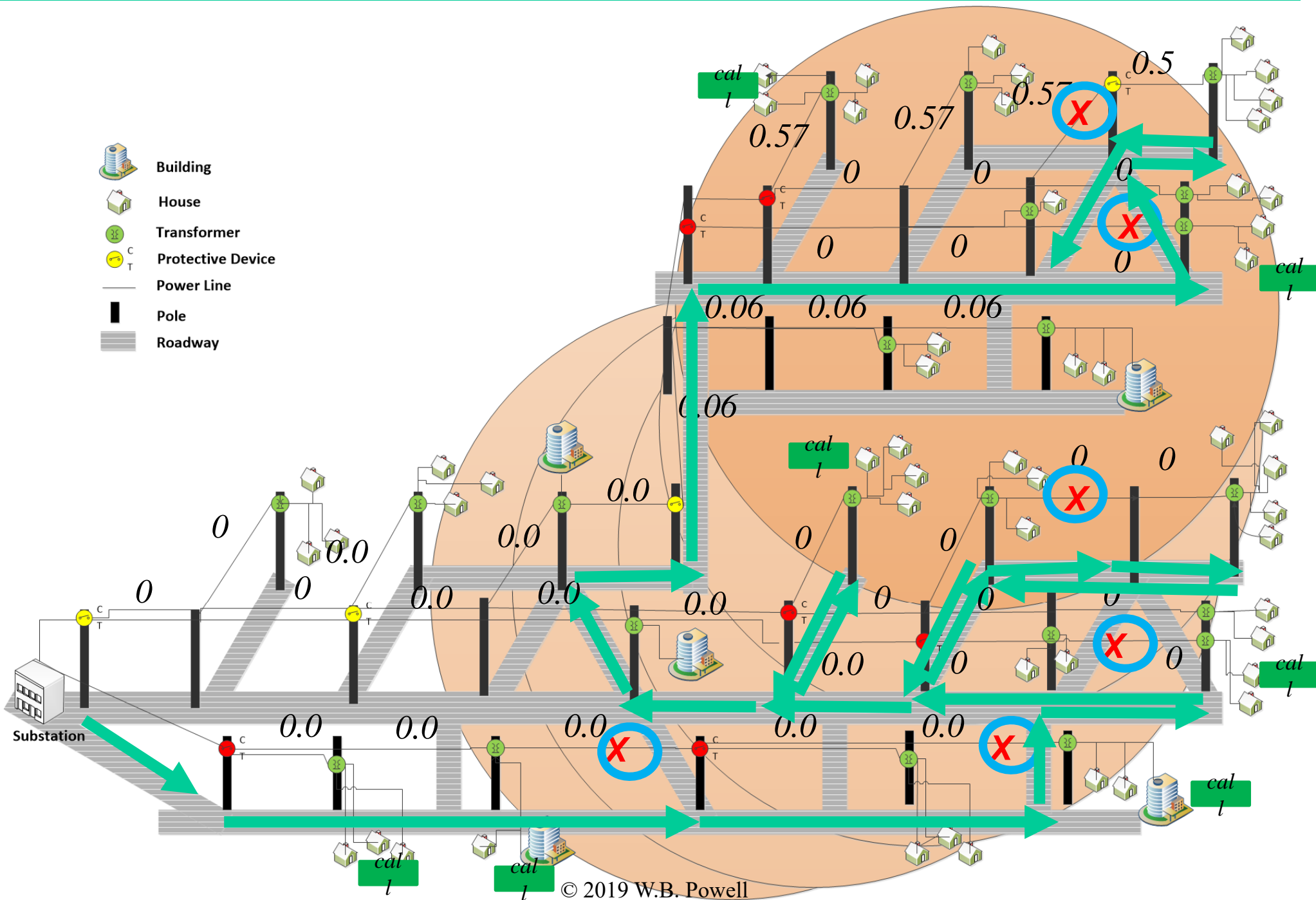
# Case study: MCTS for storm response



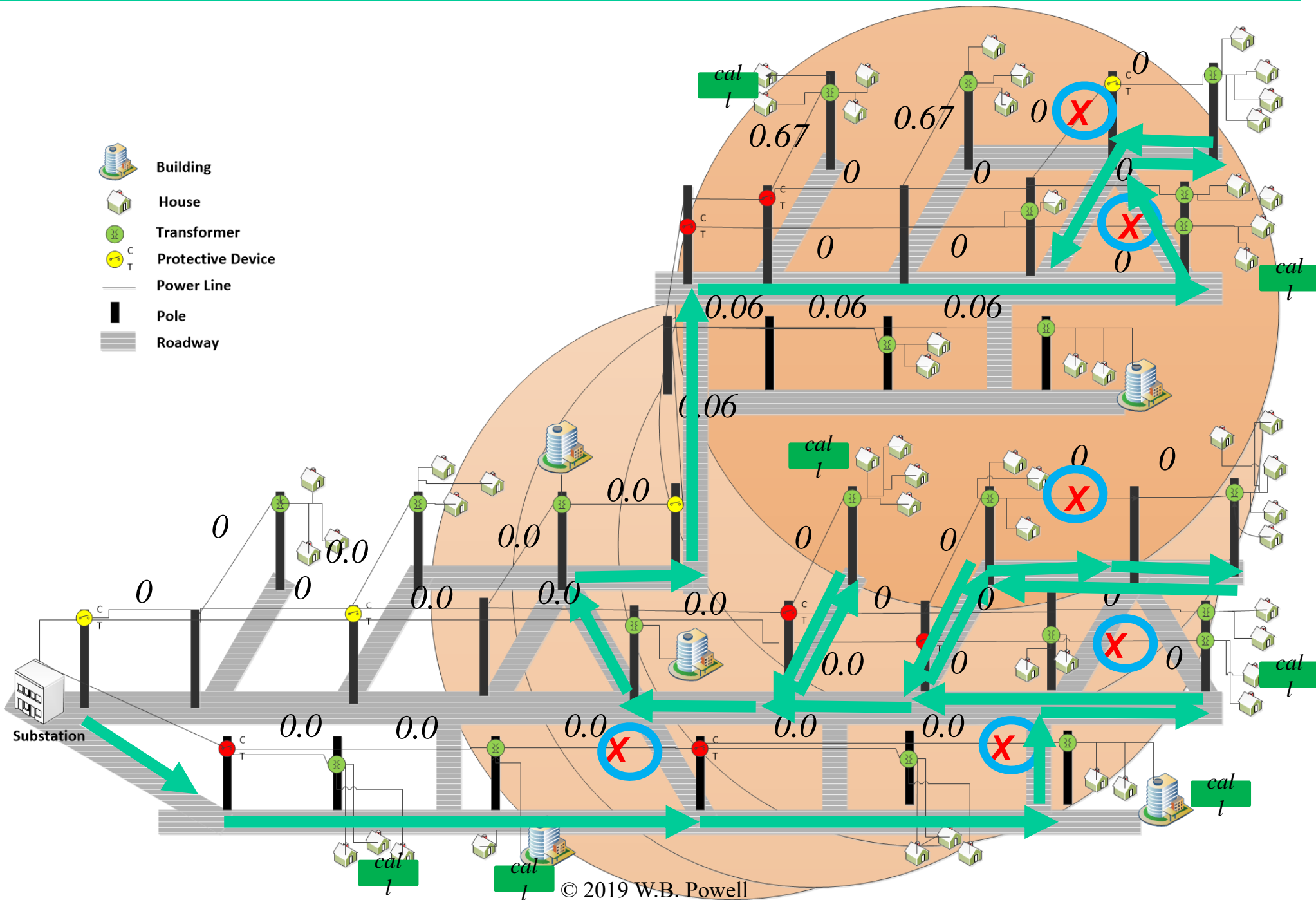
# Case study: MCTS for storm response



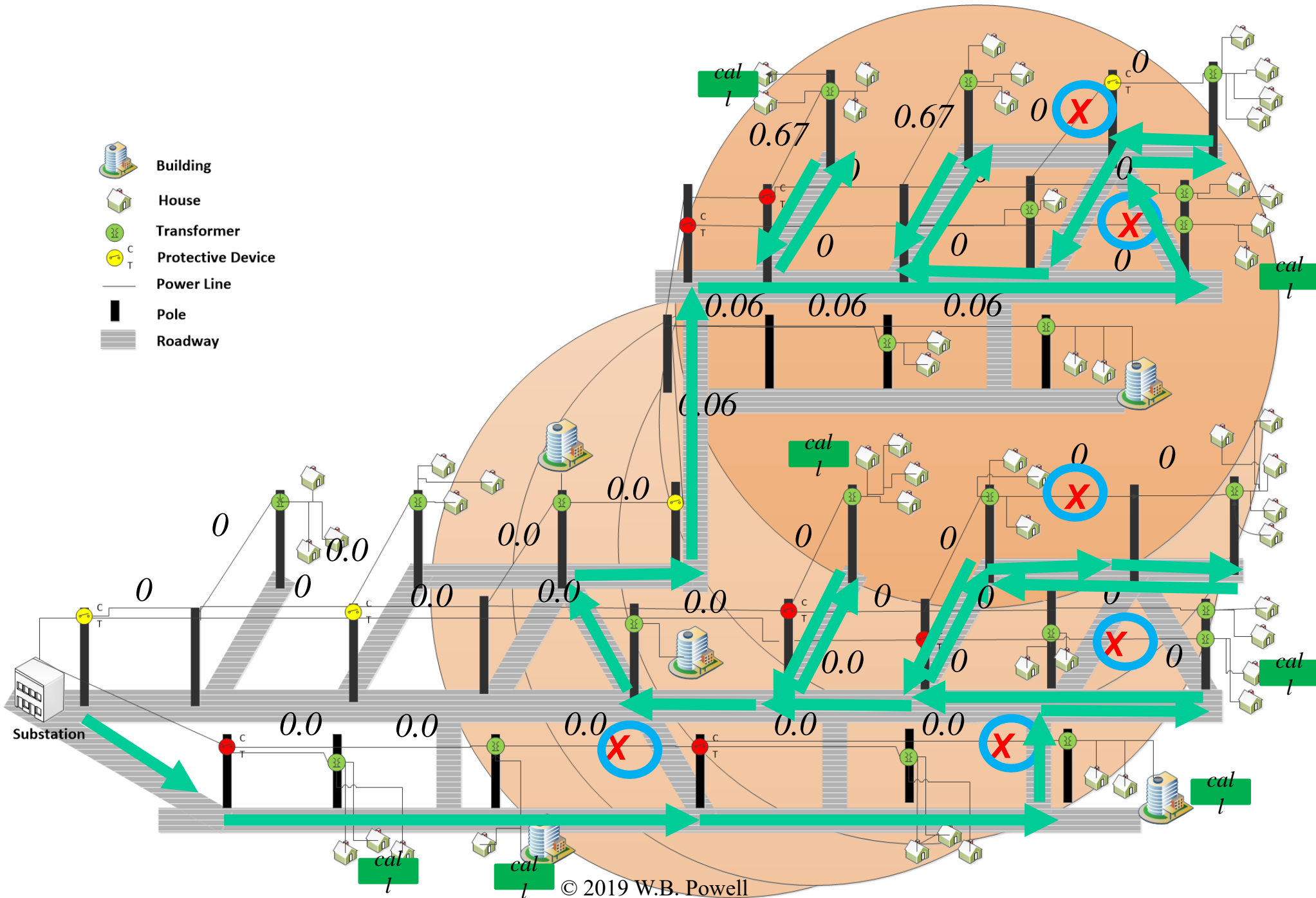
# Case study: MCTS for storm response



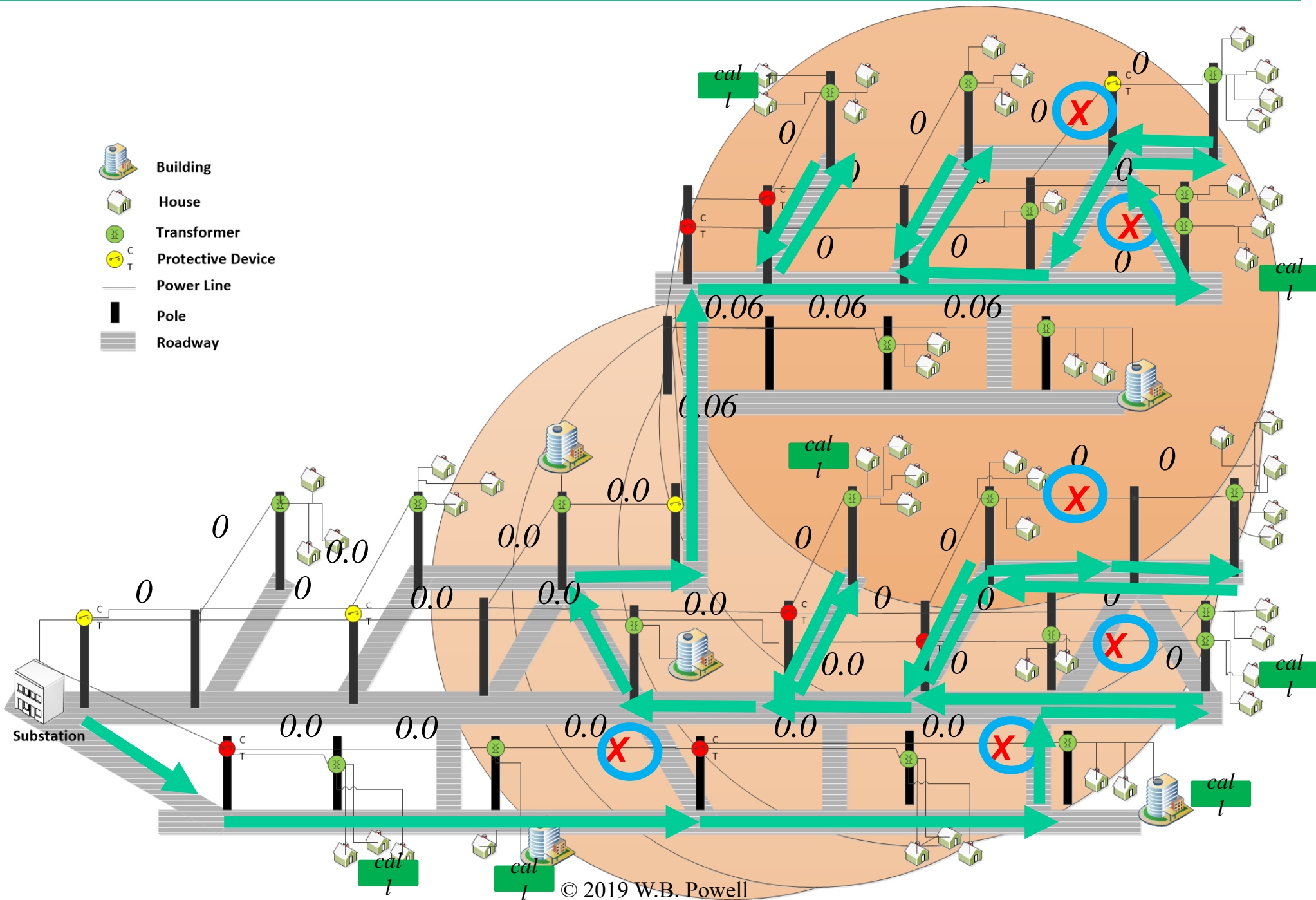
# Case study: MCTS for storm response



# Case study: MCTS for storm response



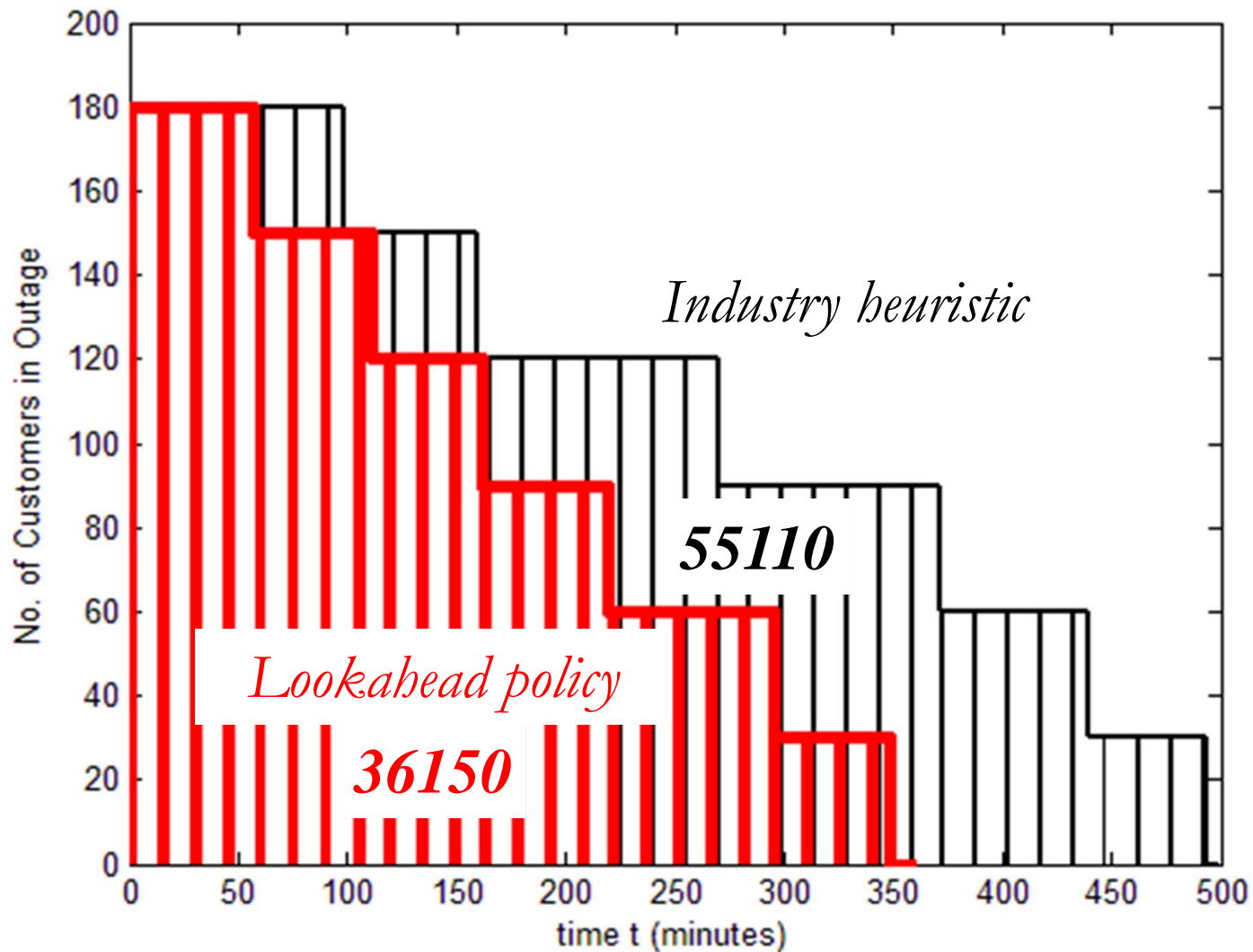
# Case study: MCTS for storm response





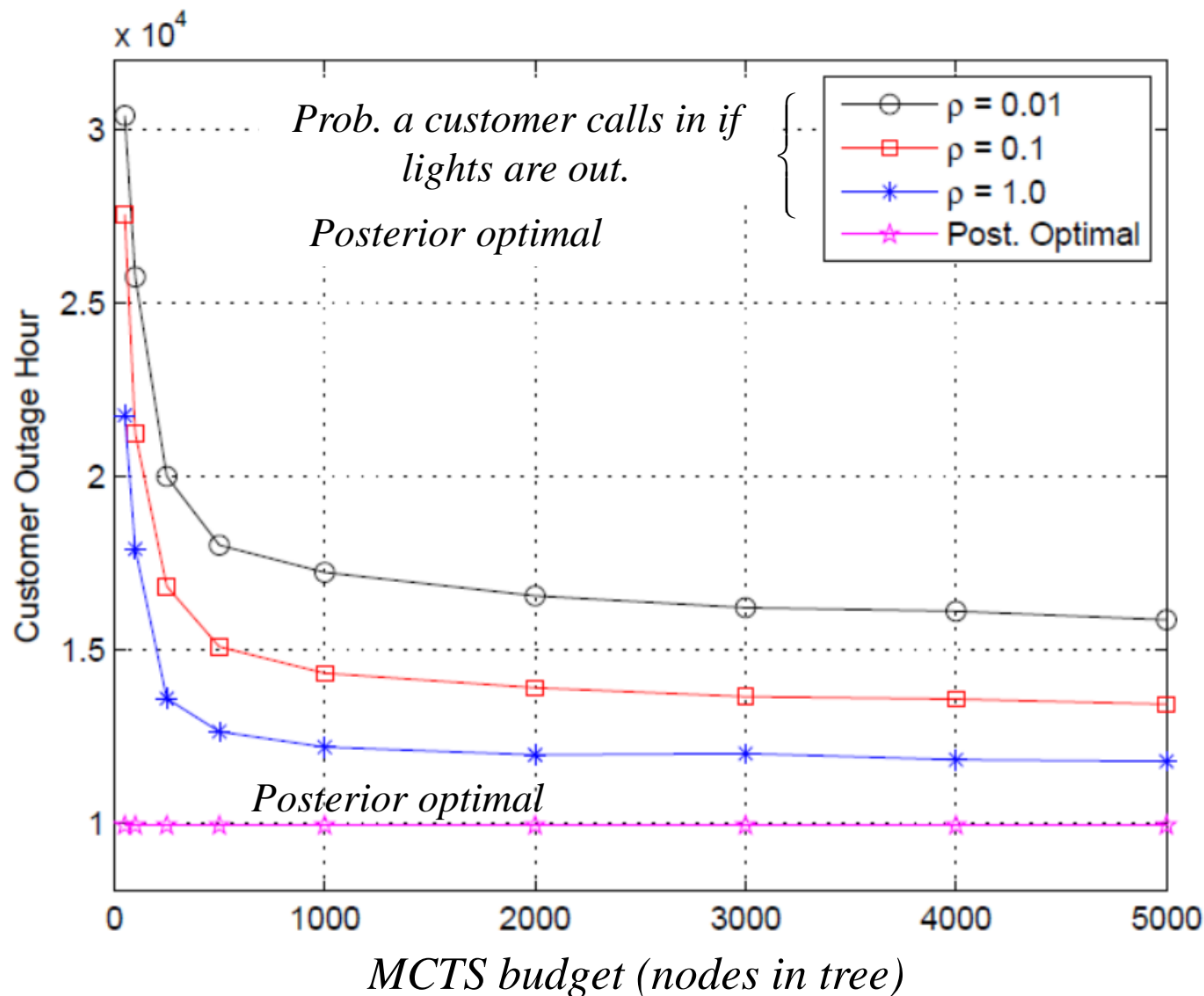
# Case study: MCTS for storm response

- Performance metrics
  - » Total outage minutes



# Case study: MCTS for storm response

## ● MCTS vs. posterior optimal benchmark





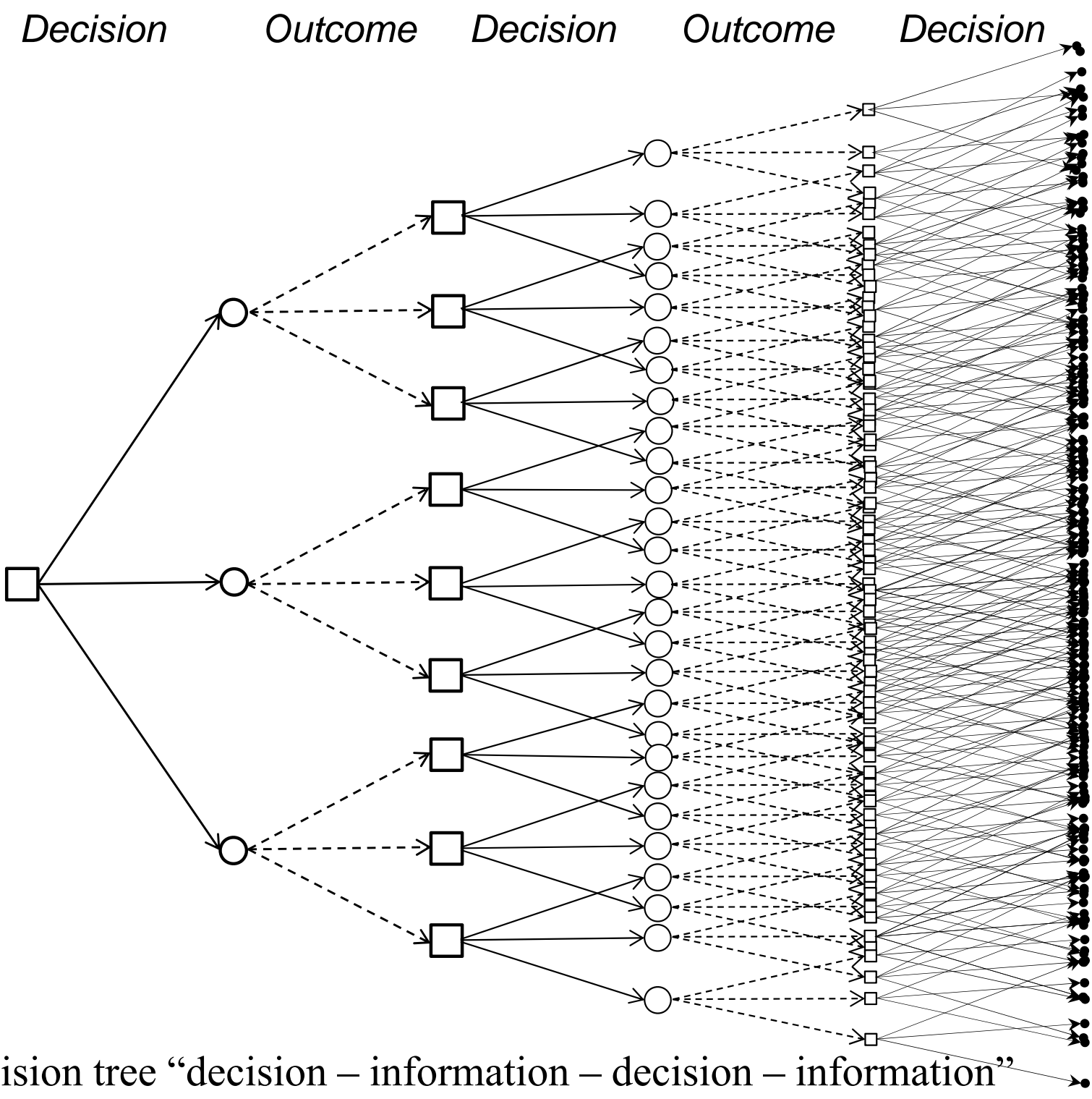
# Direct lookahead

Stochastic programming with vector-valued decisions

# Stochastic programming

---

- Setting:
  - » Amazon needs to allocate inventory to its fulfillment centers.
  - » It then observes the demand for product.
  - » It then has to decide from which fulfillment center to use to fill an order.
- We can approach this as a:
  - » Two-stage decision problem
    - Decision, information, decision, stop.
  - » A “multistage” decision problem.
    - Decision, information, decision, information, decision, ...
    - We saw earlier how these decision trees really explode.



A decision tree “decision – information – decision – information”

# Stochastic programming

---

- Stochastic programming methodology:
  - » Future information is represented in a form known as a *scenario tree*, which is a sampled version of what *might* happen.
  - » Here we just model

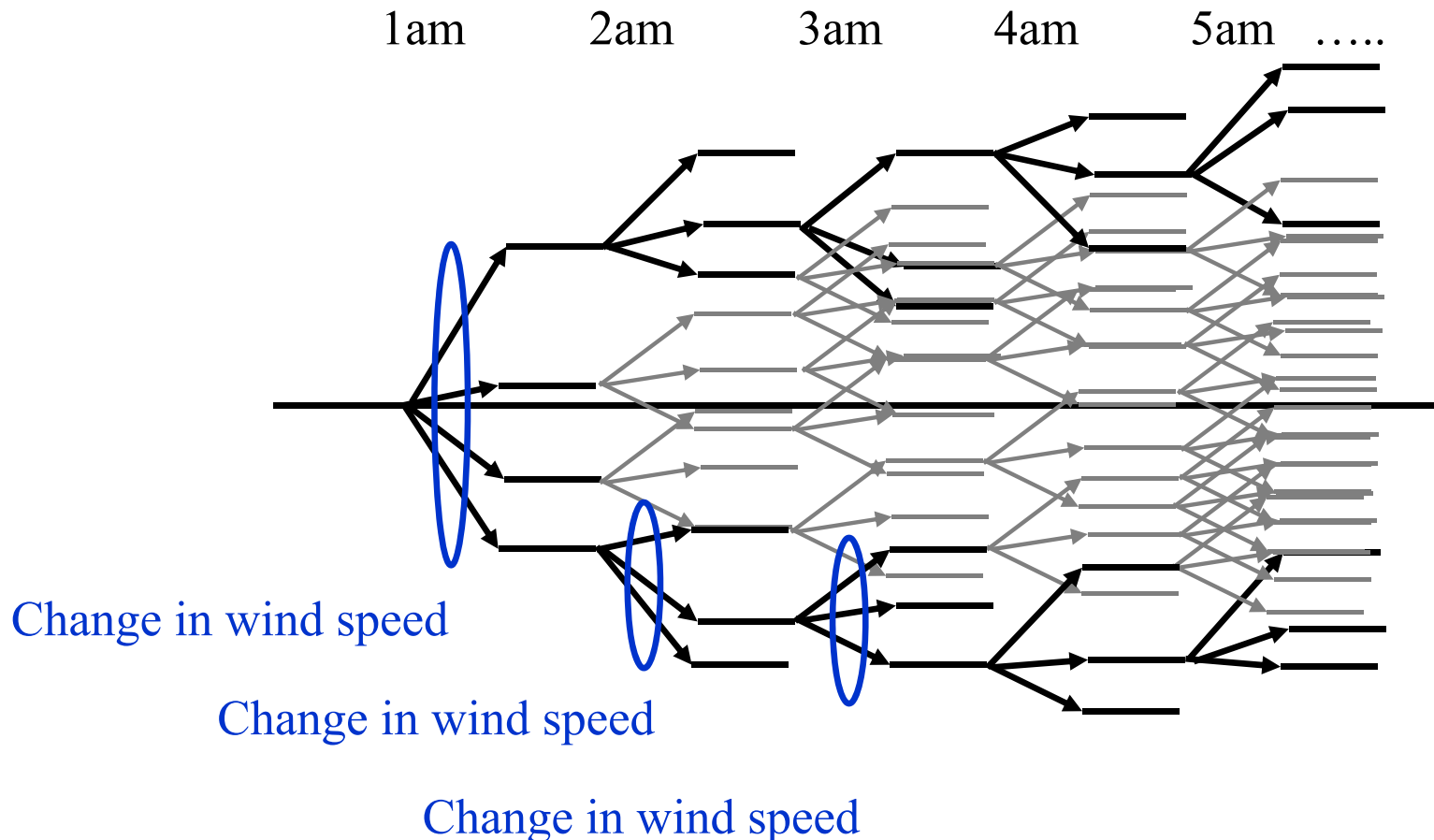
Information, information, information, ...

- » The decisions are handled separately. But even with this approach, the information process still explodes very quickly:

# Stochastic programming

## ● Stochastic lookahead

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



## ● Multistage stochastic program

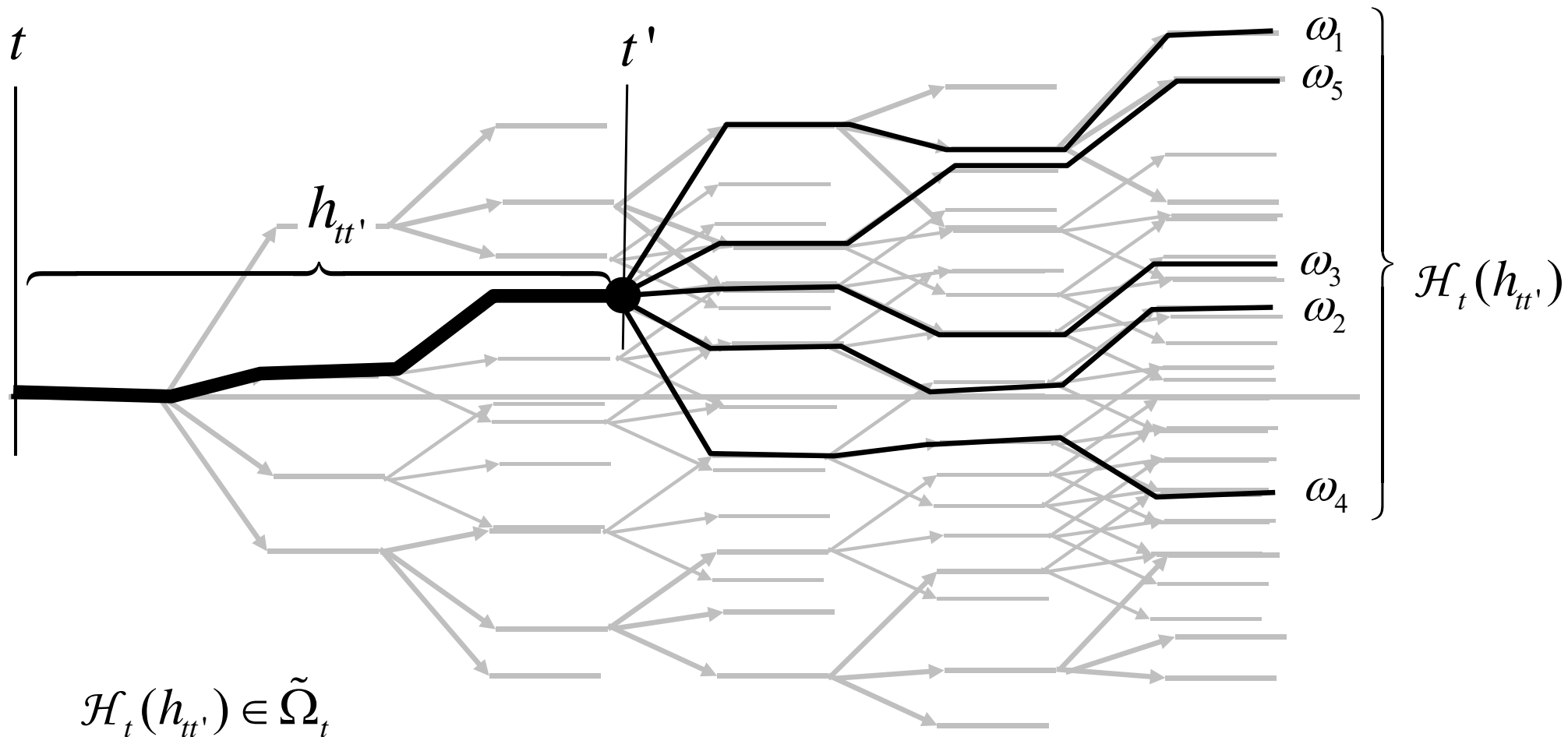
- » We are going to model the nesting of decisions  $x_t$  and new information  $W_{t+1}$ :

$$\min_{x \in X_0} \left( c_0 x_0 + \mathbb{E}_{W_1} \left\{ \min_{x_1 \in X_1} \left( c_1 x_1 + \mathbb{E}_{W_2} \left\{ \min_{x_2 \in X_2} \left( c_2 x_2 + \mathbb{E}_{W_3} \left\{ \min_{x_3 \in X_3} (c_3 x_3 + \dots) \mid \mathcal{S}_2 \right\} \dots \right) \mid \mathcal{S}_1 \right\} \right) \mid \mathcal{S}_0 \right\} \right)$$

- » Remember that  $x_t$  is a vector.
- » Don't worry, no-one can solve this directly, so the challenge is designing something that we can solve.
- » There is one special case that can be solved with an asymptotically optimal algorithm, *approximate dynamic programming* with Benders cuts, called *stochastic dual dynamic programming* (SDDP).

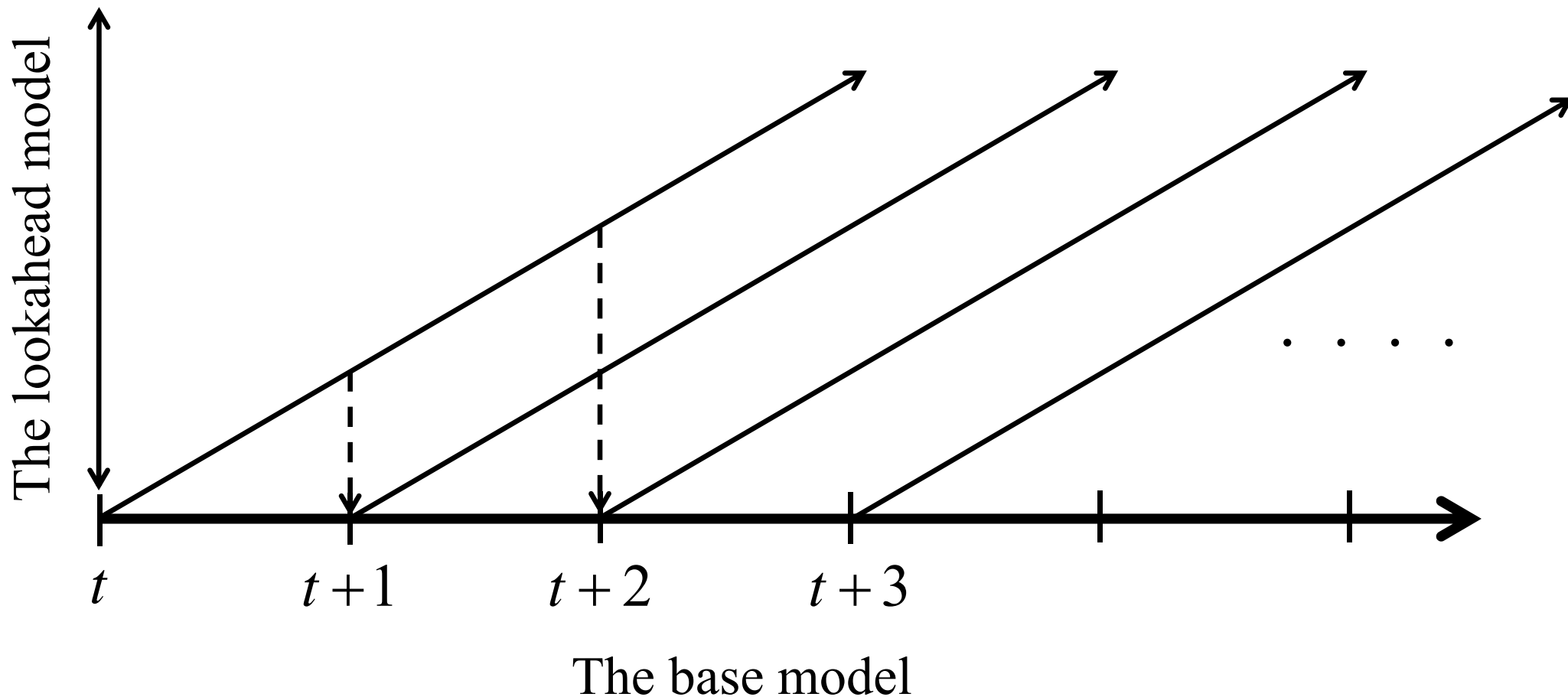
# Stochastic programming

- Information evolves using sequential branching:



# Stochastic programming

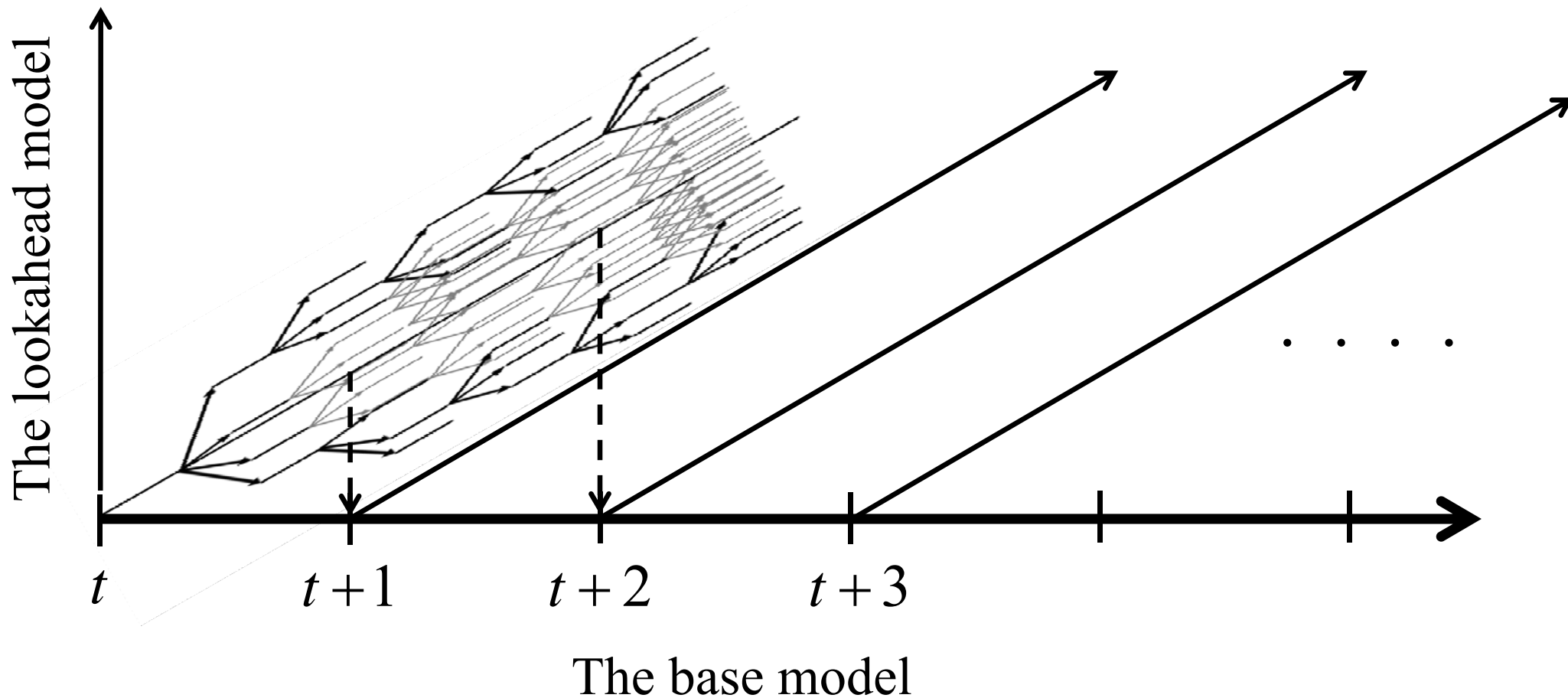
- We can then simulate this *lookahead policy* over time:





# Stochastic programming

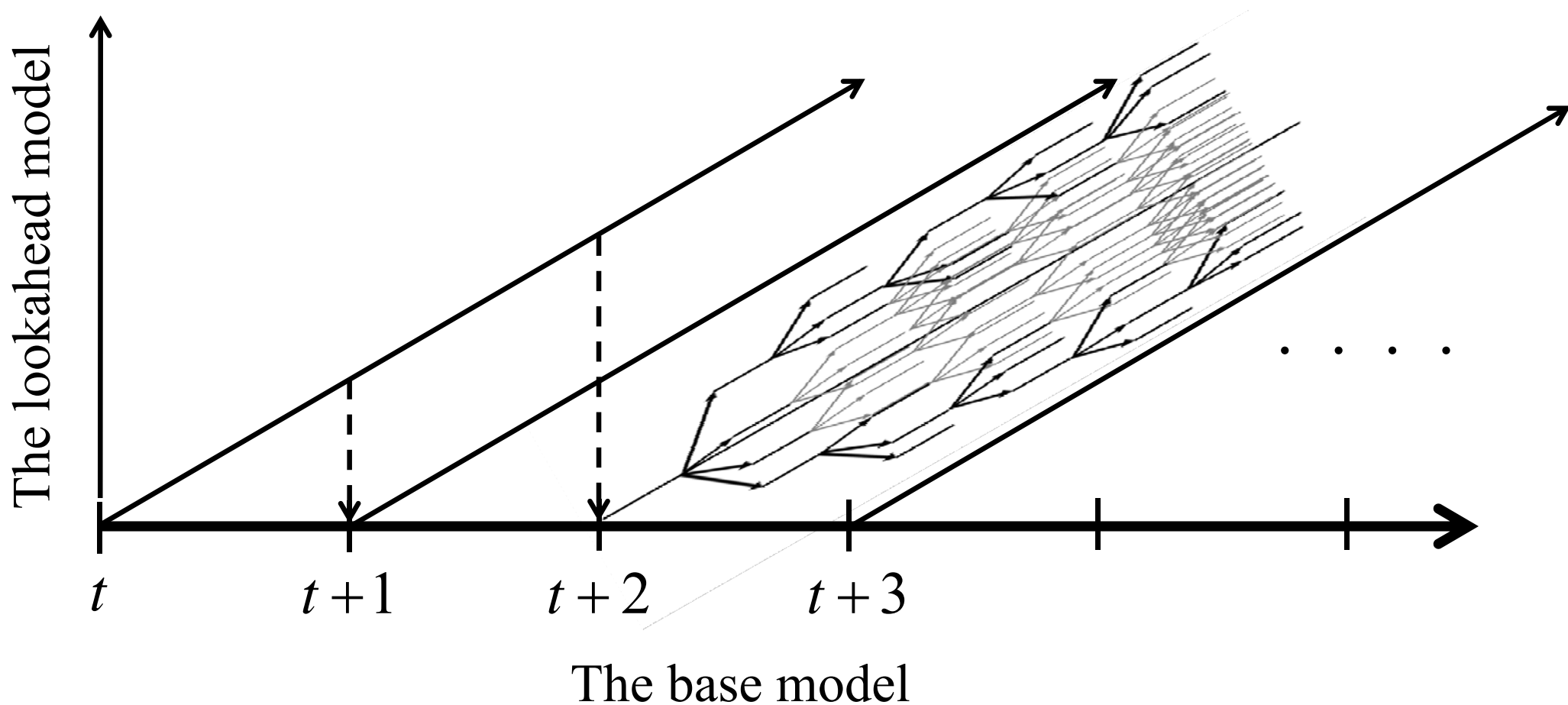
- We can then simulate this *lookahead policy* over time:





# Stochastic programming

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



# Stochastic programming

---

- Notes:

- » In most applications of stochastic programming, decision  $x_t$  is a vector (and may even have to be integer).
- » Solving even one instance of a multistage stochastic lookahead model is rarely tractable.

Direct lookahead

Two-stage stochastic programming

# Direct lookahead

- Classical approximation strategy: Two-stage stochastic programming

- » Collapse all the states (decision-information) for periods  $t + 1, t + 2, \dots, t + H$  into a single stage:

- Assume we see  $W_{t+2}, W_{t+3}, \dots, W_{t+H}$ . Let  $\omega$  be a sample realization of  $W_{t+2}, W_{t+3}, \dots, W_{t+H}$ , and create a sample  $\hat{\Omega}_t$ .

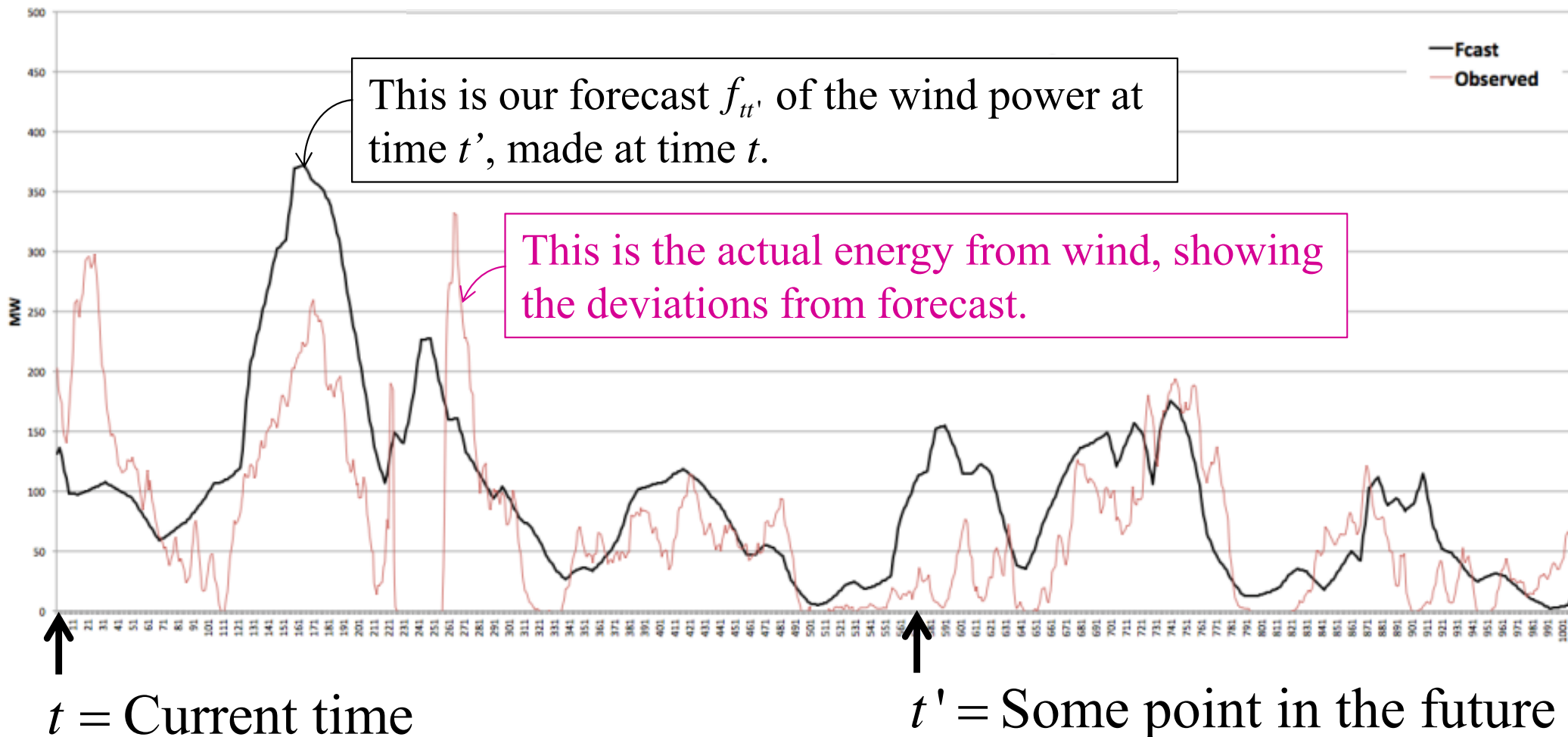
- Then make decisions  $\tilde{x}_{t,t+1}(\omega), \tilde{x}_{t,t+2}(\omega), \dots, \tilde{x}_{t,t+H}(\omega)$

- » This produces the two-stage stochastic optimization problem:

$$\min_{x_t, (\tilde{x}_{t,t+1}(\omega), \dots, \tilde{x}_{t,t+H}(\omega)), \omega \in \Omega_t} c_t x_t + \sum_{\omega \in \Omega_t} \sum_{t'=t+1}^{t+H} \tilde{c}_{tt'}(\omega) \tilde{x}_{tt'}(\omega)$$

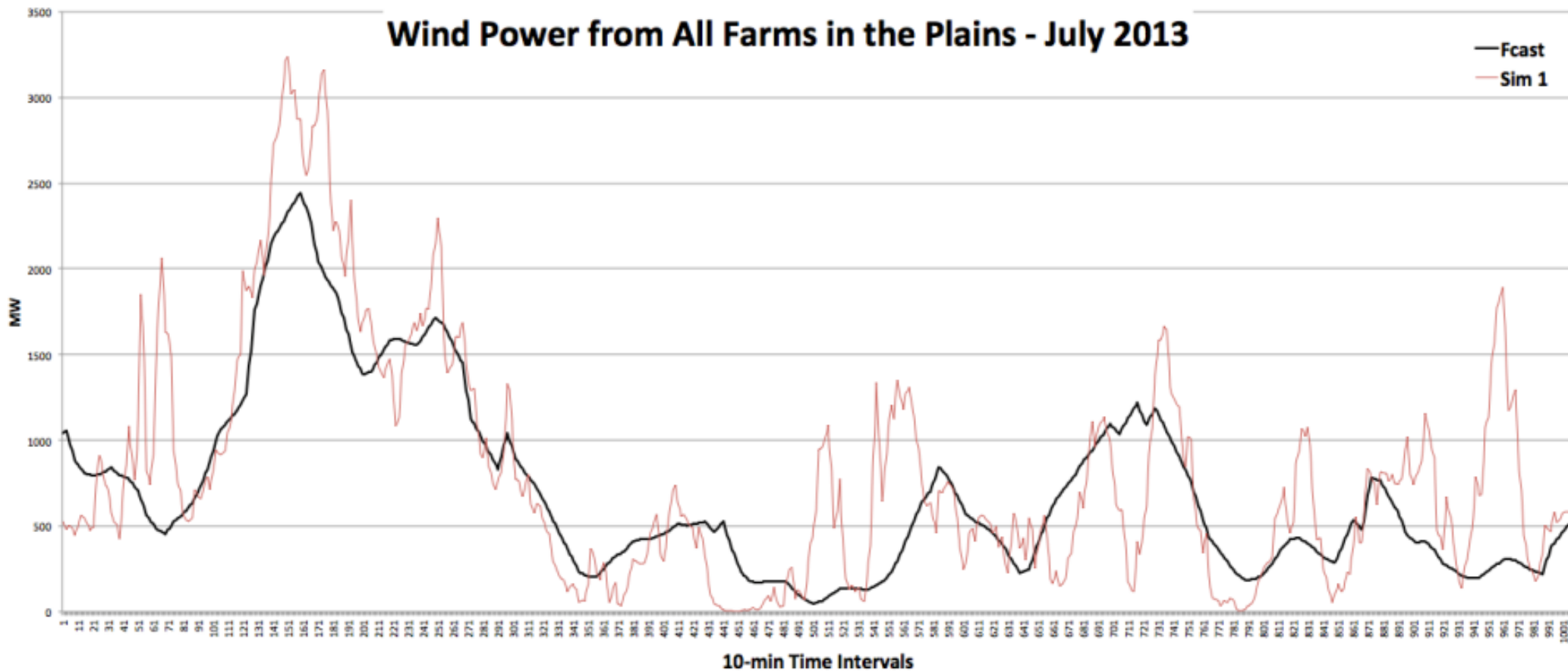
# Stochastic programming

- Actual vs. forecasted energy from wind



# Stochastic programming

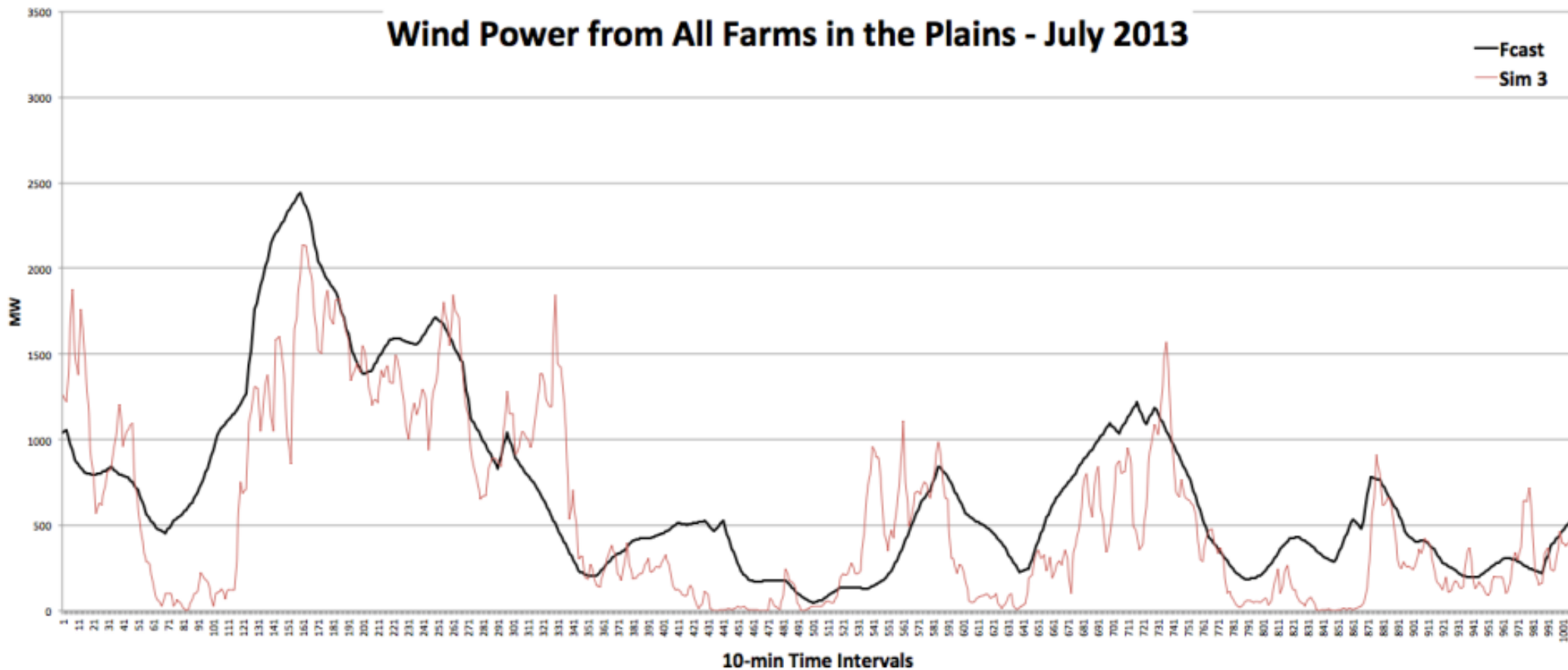
- Creating wind scenarios (Scenario #1)





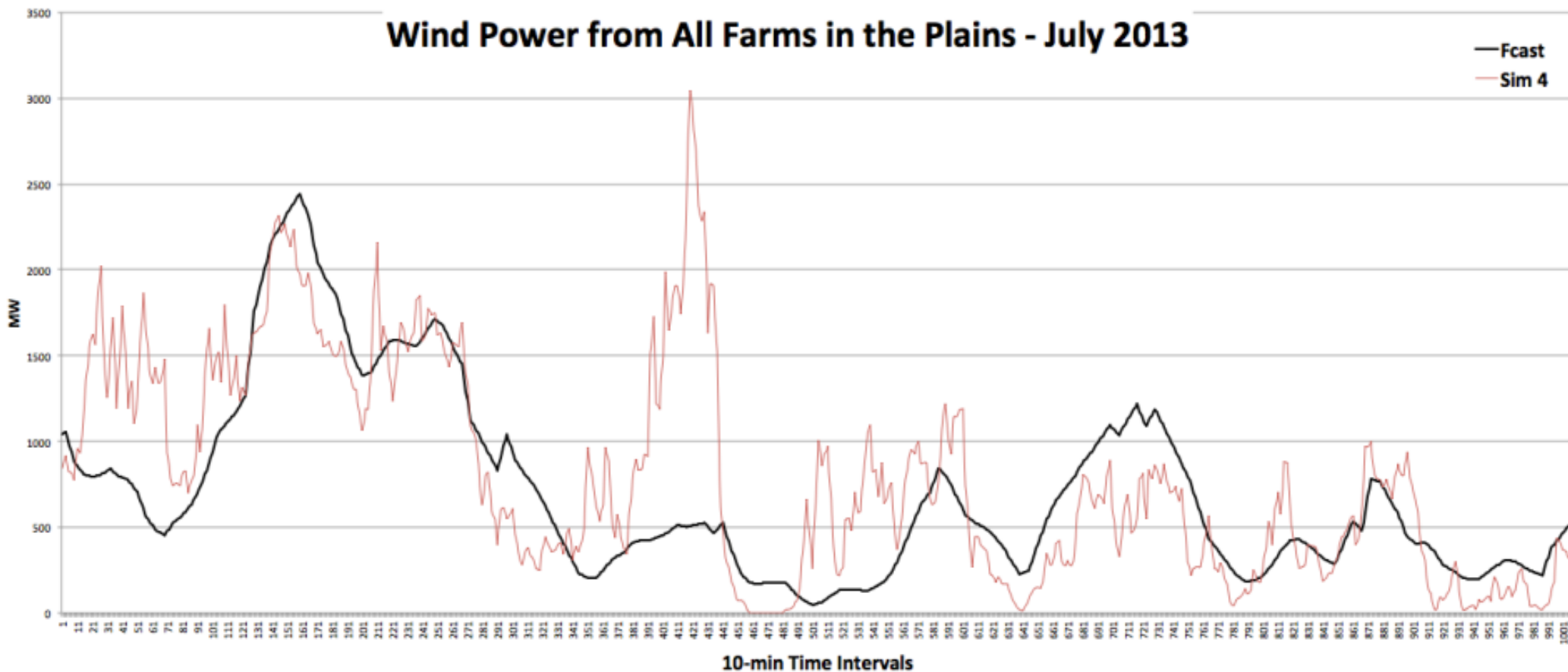
# Stochastic programming

- Creating wind scenarios (Scenario #3)



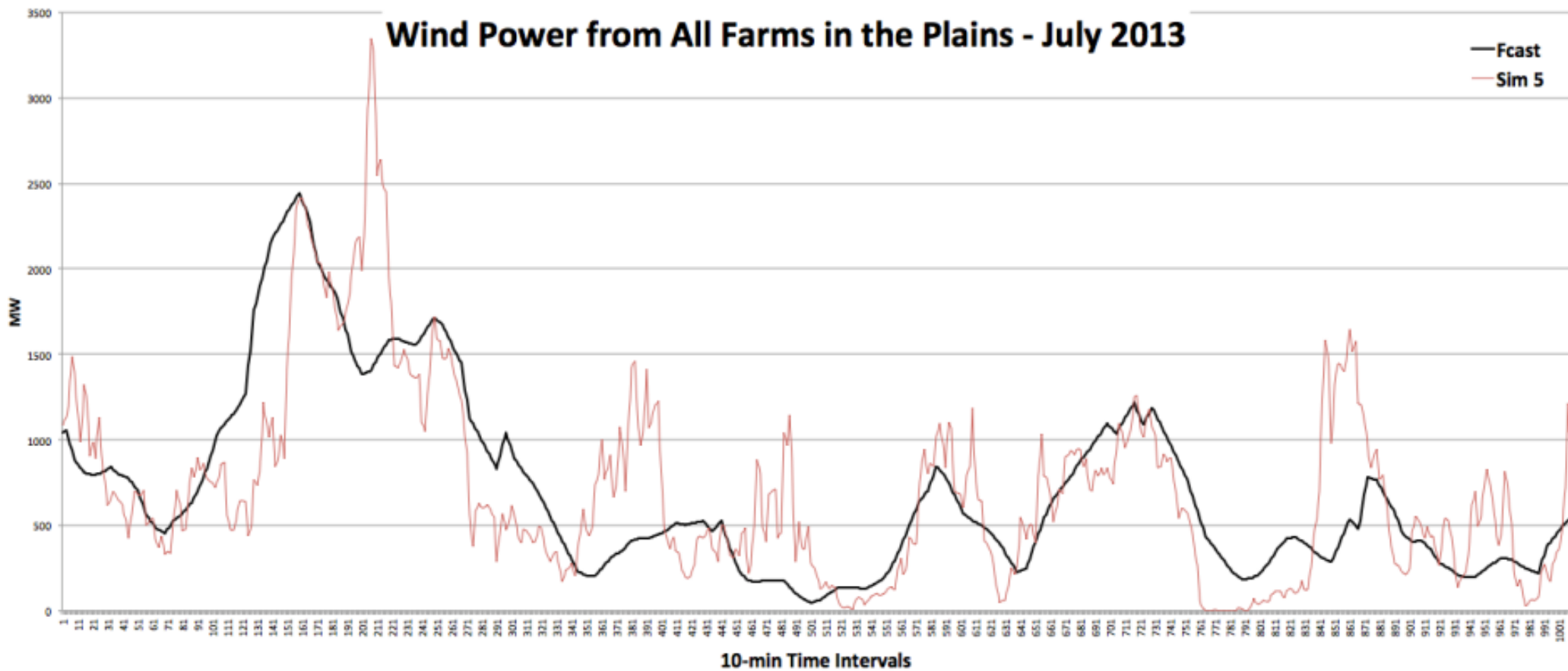
# Stochastic programming

- Creating wind scenarios (Scenario #4)



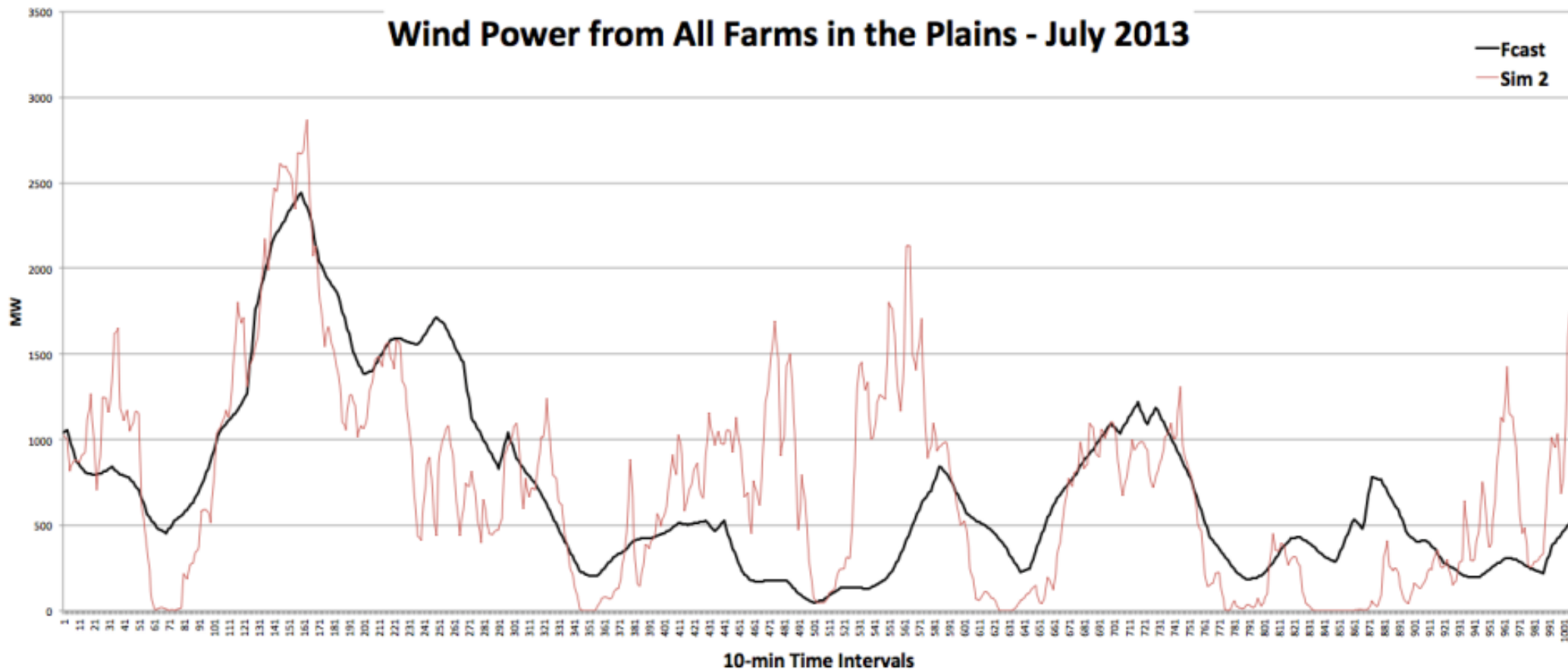
# Stochastic programming

- Creating wind scenarios (Scenario #5)



# Stochastic programming

- Creating wind scenarios (Scenario #2)



# Stochastic programming

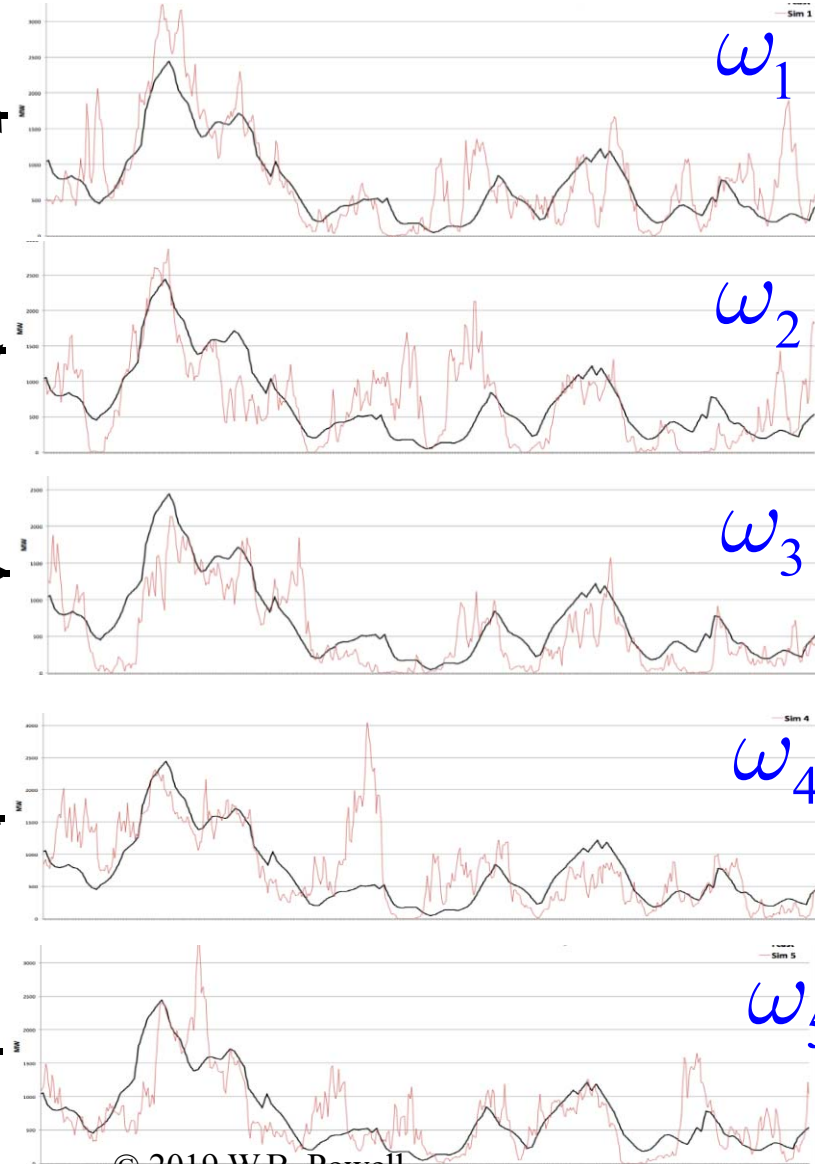
$$\min_{x_t, \tilde{x}_{t,t+1}(\omega)} c_t x_t + \sum_{\omega \in \Omega_t} \sum_{t'=t+1}^{t+H} \tilde{c}_{tt'}(\omega) \tilde{x}_{tt'}(\omega) \quad x_{t+1}(\omega), x_{t+2}(\omega), \dots, x_{t+H}(\omega)$$

1) Schedule steam

$x_t$

2) See wind:

3) Schedule turbines



$\Omega_t$

# Direct lookahead

- Classical approximation strategy: Two-stage stochastic programming

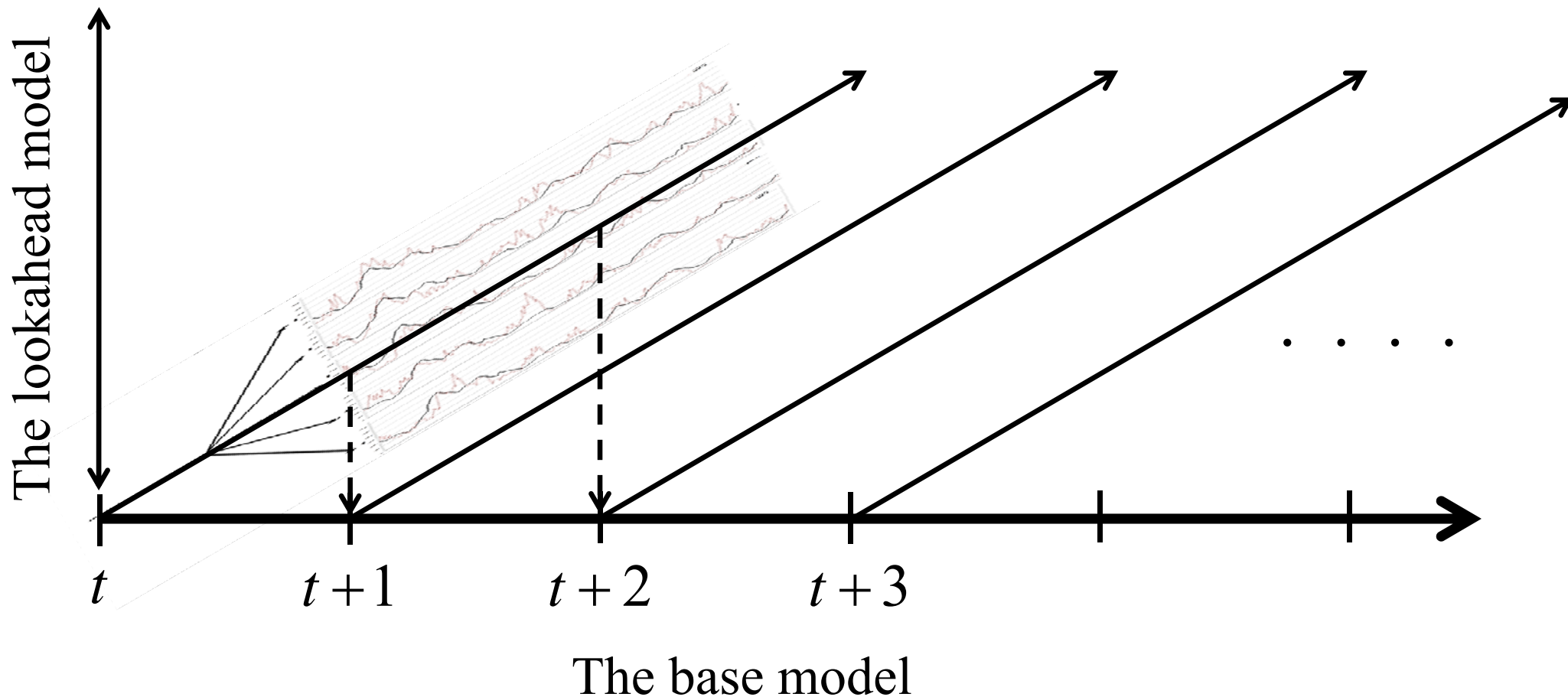
» Take a look at the two-stage stochastic program

$$\min_{x_t, (\tilde{x}_{t,t+1}(\omega), \dots, \tilde{x}_{t,t+H}(\omega)), \omega \in \Omega_t)} c_t x_t + \sum_{\omega \in \Omega_t} \sum_{t'=t+1}^{t+H} \tilde{c}_{tt'}(\omega) \tilde{x}_{tt'}(\omega)$$

- » We are choosing a single (vector)  $x_t$  for the decision to be made now, along with a family of vectors  $\tilde{x}_{t,t+1}(\omega)$ , representing decisions to be made in the future.
- » Note that the decision  $\tilde{x}_{tt'}(\omega)$  given “scenario”  $\omega$  is allowed to see the entire future. This is dismissed as a reasonable approximation, as long as  $x_t$  is not allowed to see the future.

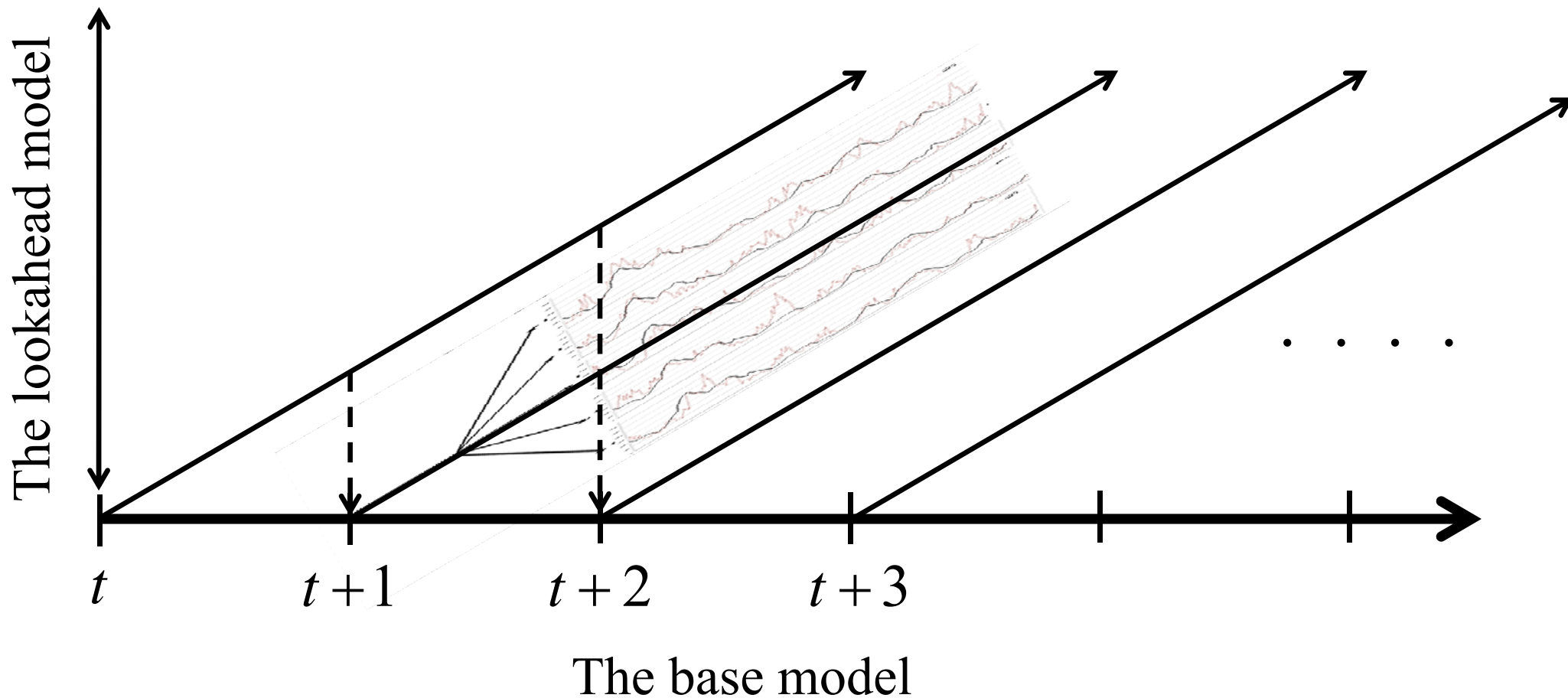
# Stochastic programming

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



# Stochastic programming

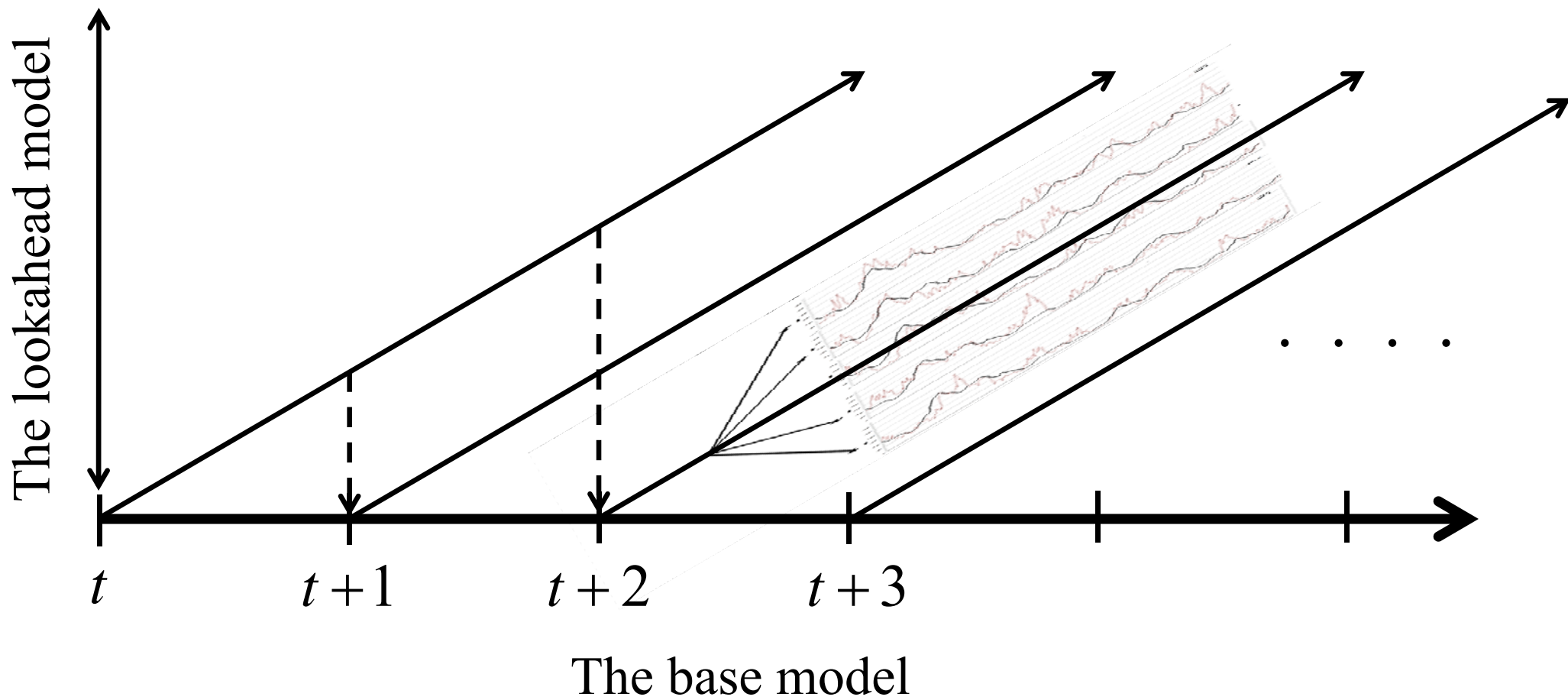
- We can then simulate this *lookahead policy* over time:





# Stochastic programming

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



# Stochastic programming

## ● Notes:

- » In practice, it is quite rare to simulate a lookahead policy based on scenario trees.
- » Solving even one instance of a stochastic lookahead problem can be extremely difficult:

$$\min_{x_t, \{\tilde{x}_{t,t+1}(\omega), \dots, \tilde{x}_{t,t+H}(\omega), \omega \in \Omega_t\}} c_t x_t + \sum_{\omega \in \Omega_t} \sum_{t'=t+1}^{t+H} \tilde{c}_{tt'}(\omega) \tilde{x}_{tt'}(\omega)$$

- » A number of authors (in stochastic programming) confuse the lookahead model with the “problem” being solved (we call this the base model). Even when we can solve the two-stage stochastic program, it is not an optimal policy because of all the approximations.

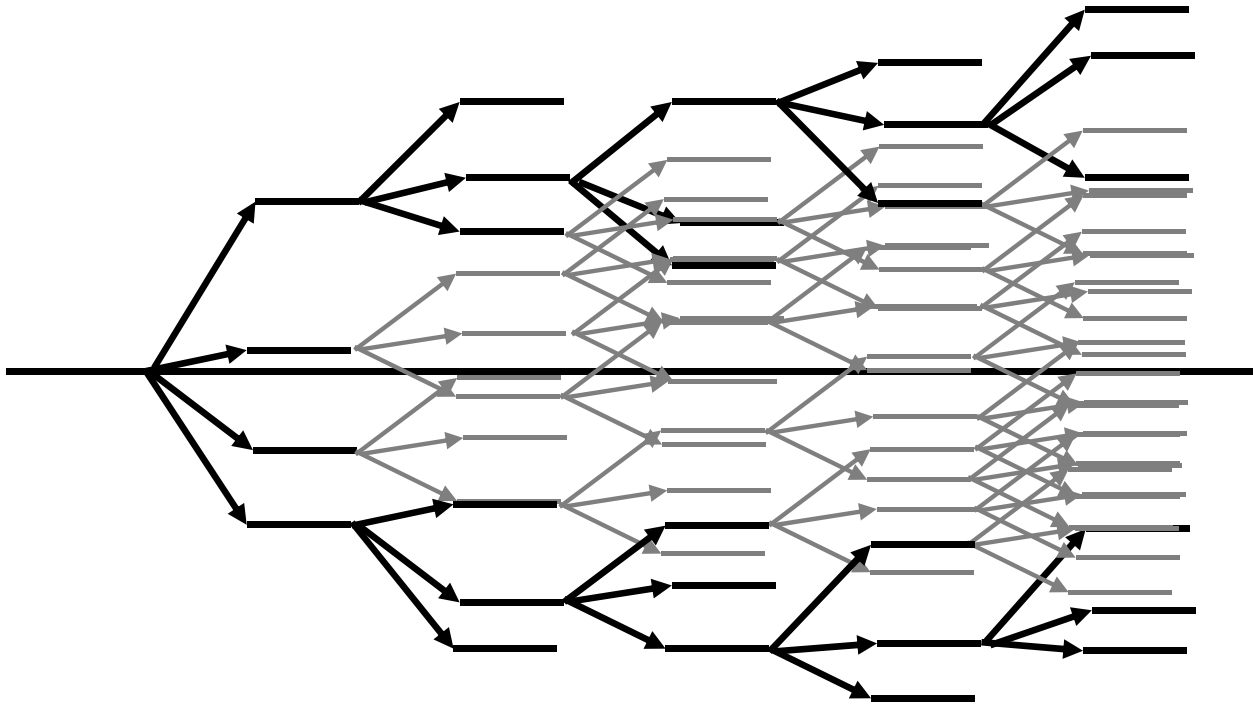
Direct lookahead

DP lookahead

# Stochastic lookahead

## ● Probabilistic lookahead

- » We can also solve the lookahead model using convex value function approximations (“dynamic programming”)

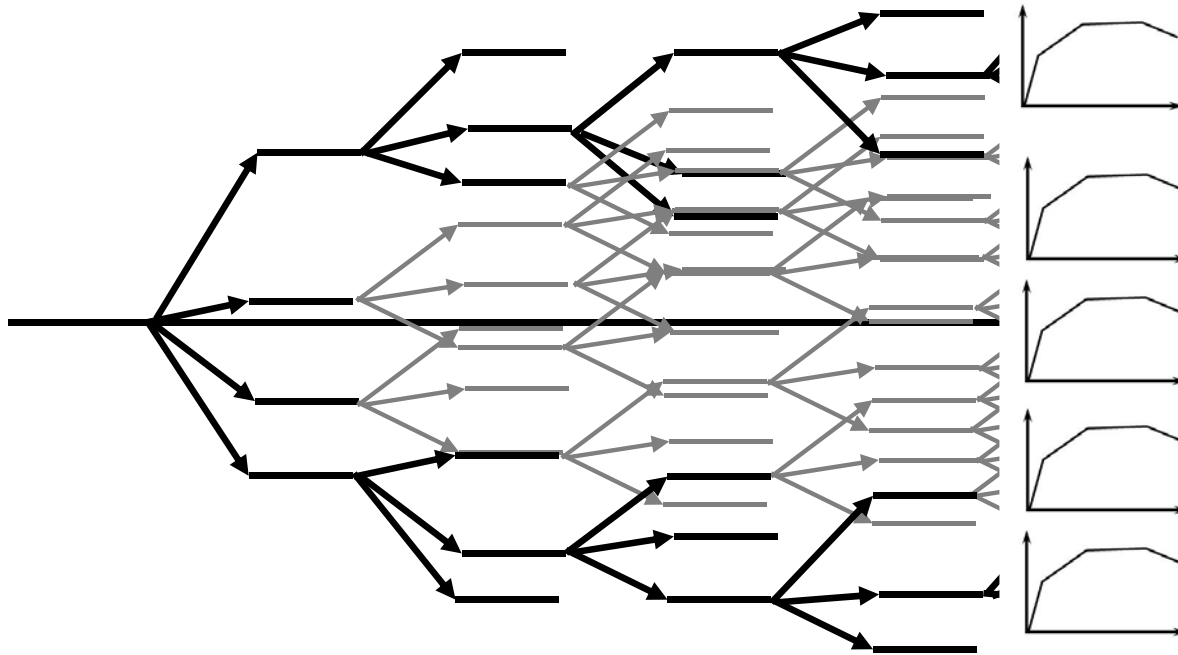


- » We can try to solve this as a single “deterministic” optimization problem. This is a *direct lookahead policy*.

# Lookahead policies

- Probabilistic lookahead

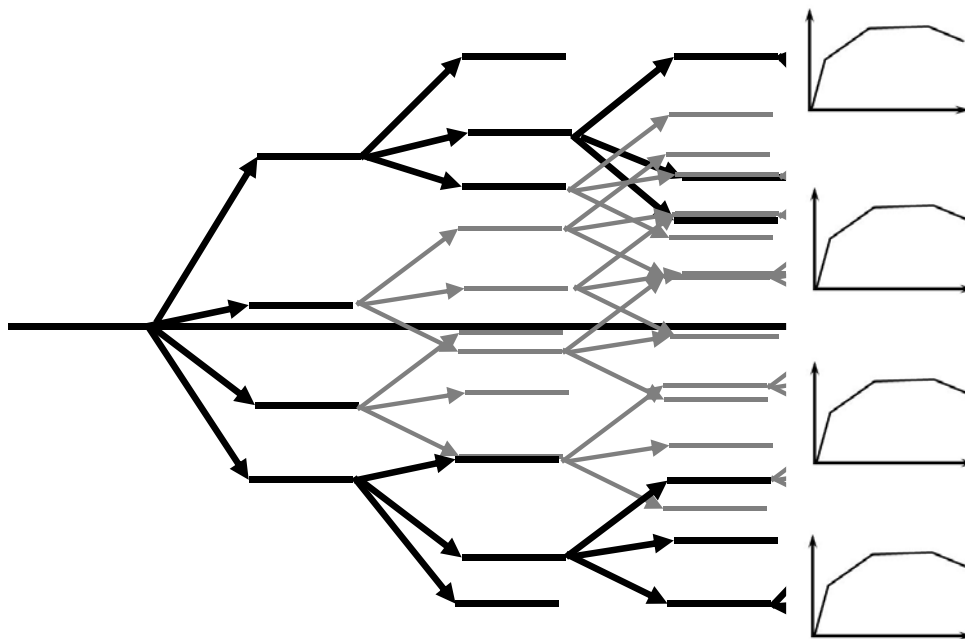
- » We can also solve the lookahead model using convex value function approximations (“dynamic programming”)



# Lookahead policies

- Probabilistic lookahead

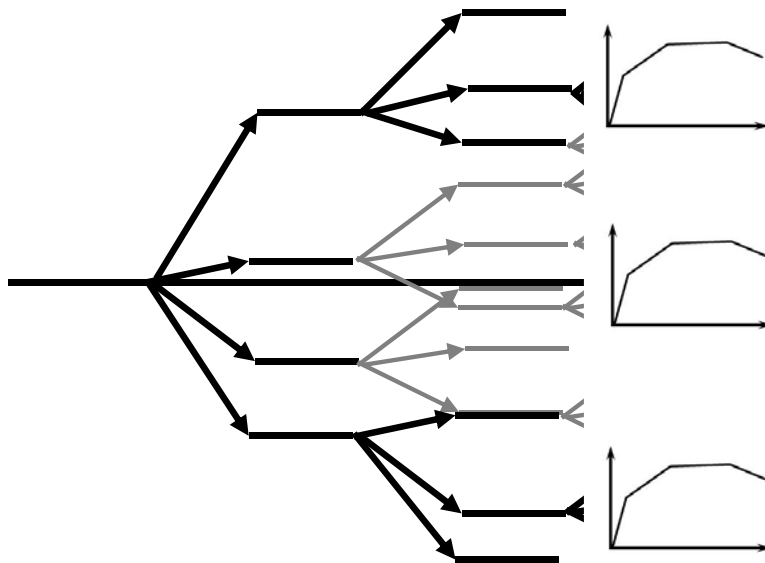
- » We can also solve the lookahead model using convex value function approximations (“dynamic programming”)



# Lookahead policies

- Probabilistic lookahead

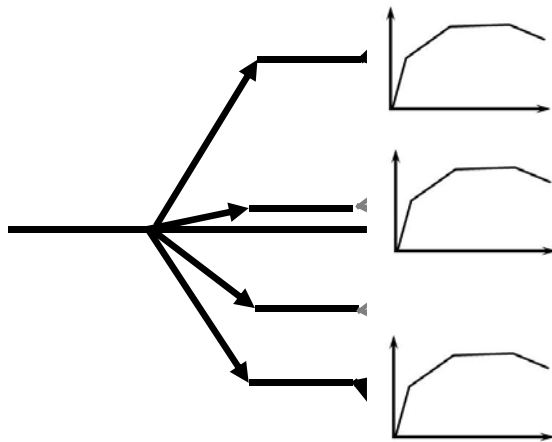
- » We can also solve the lookahead model using convex value function approximations (“dynamic programming”)



# Lookahead policies

- Probabilistic lookahead

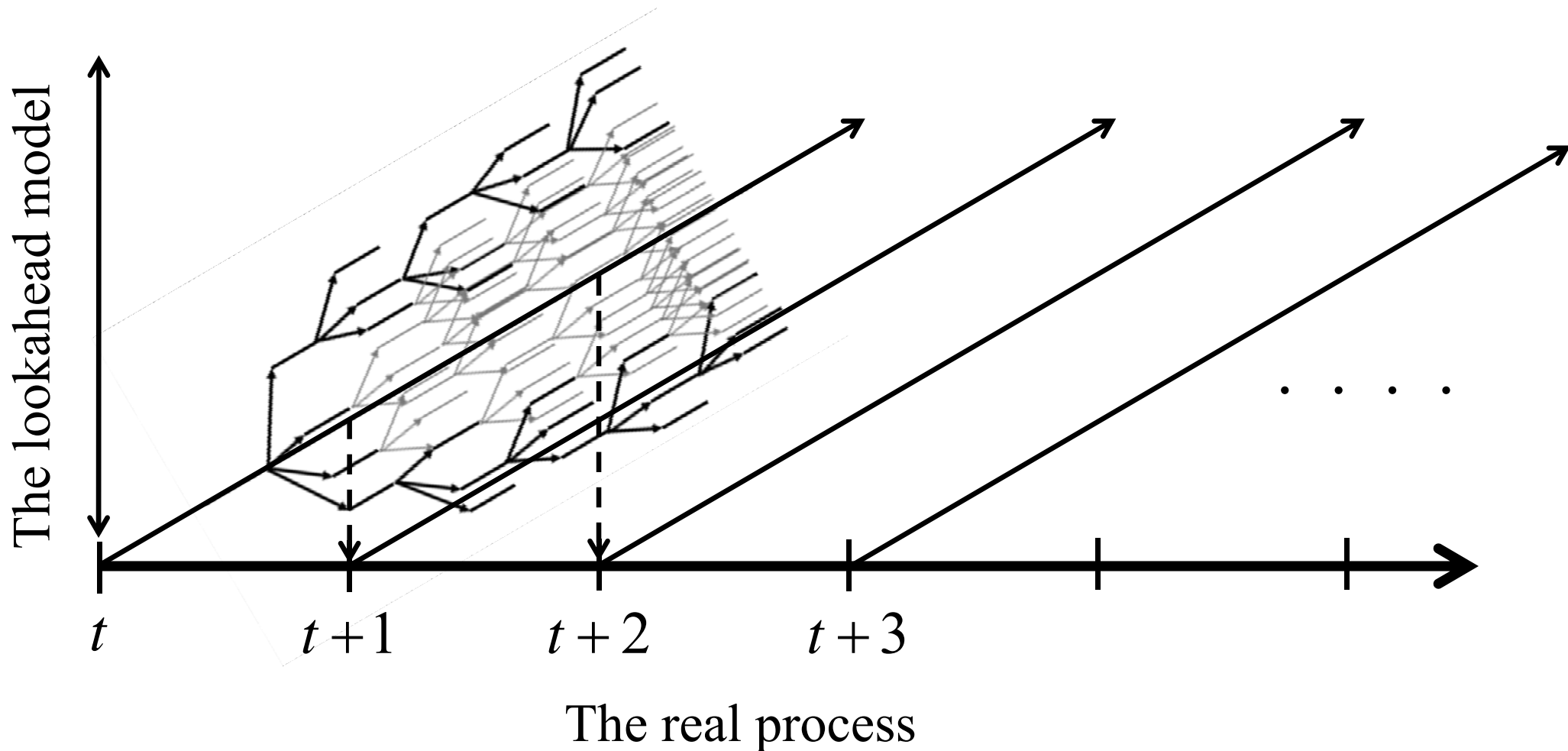
- » We can also solve the lookahead model using convex value function approximations (“dynamic programming”)





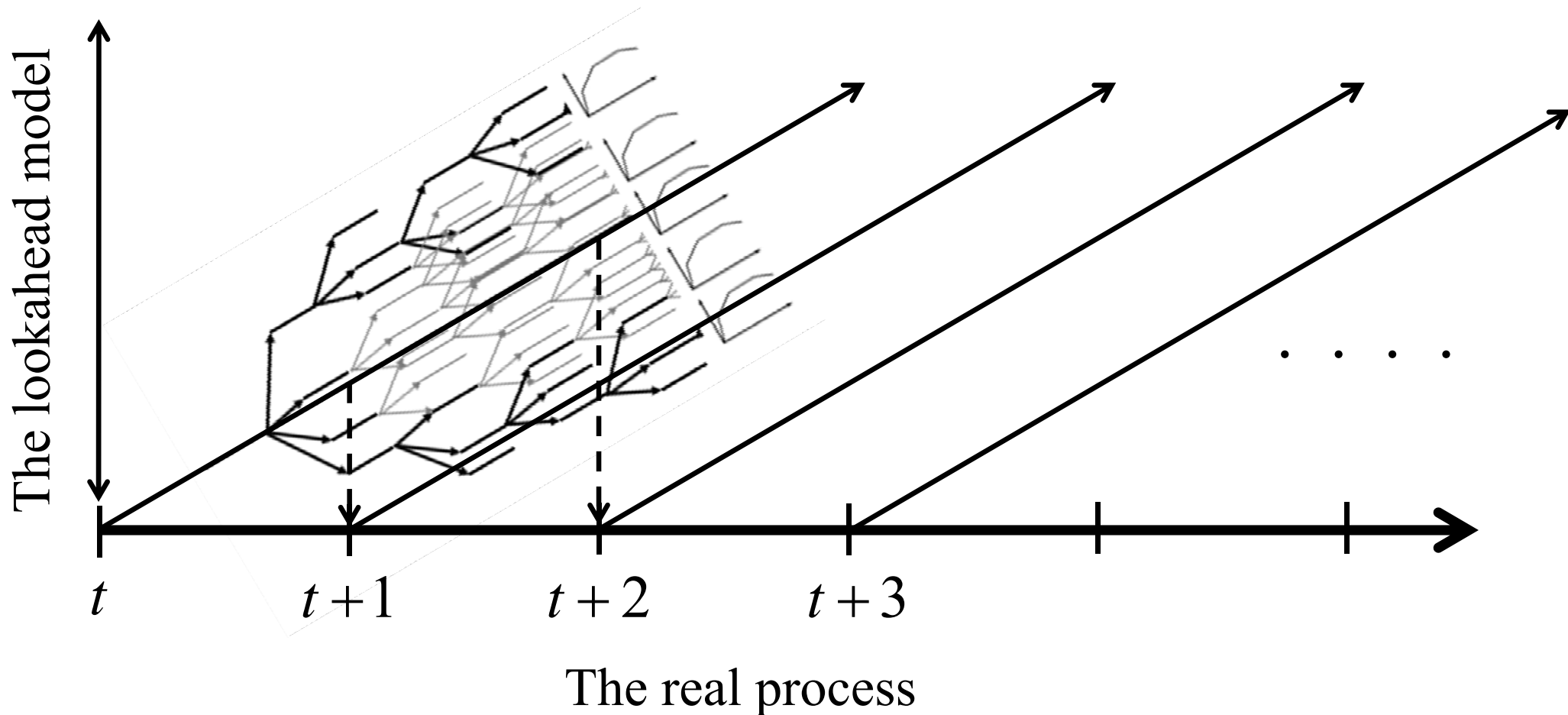
# Lookahead policies

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



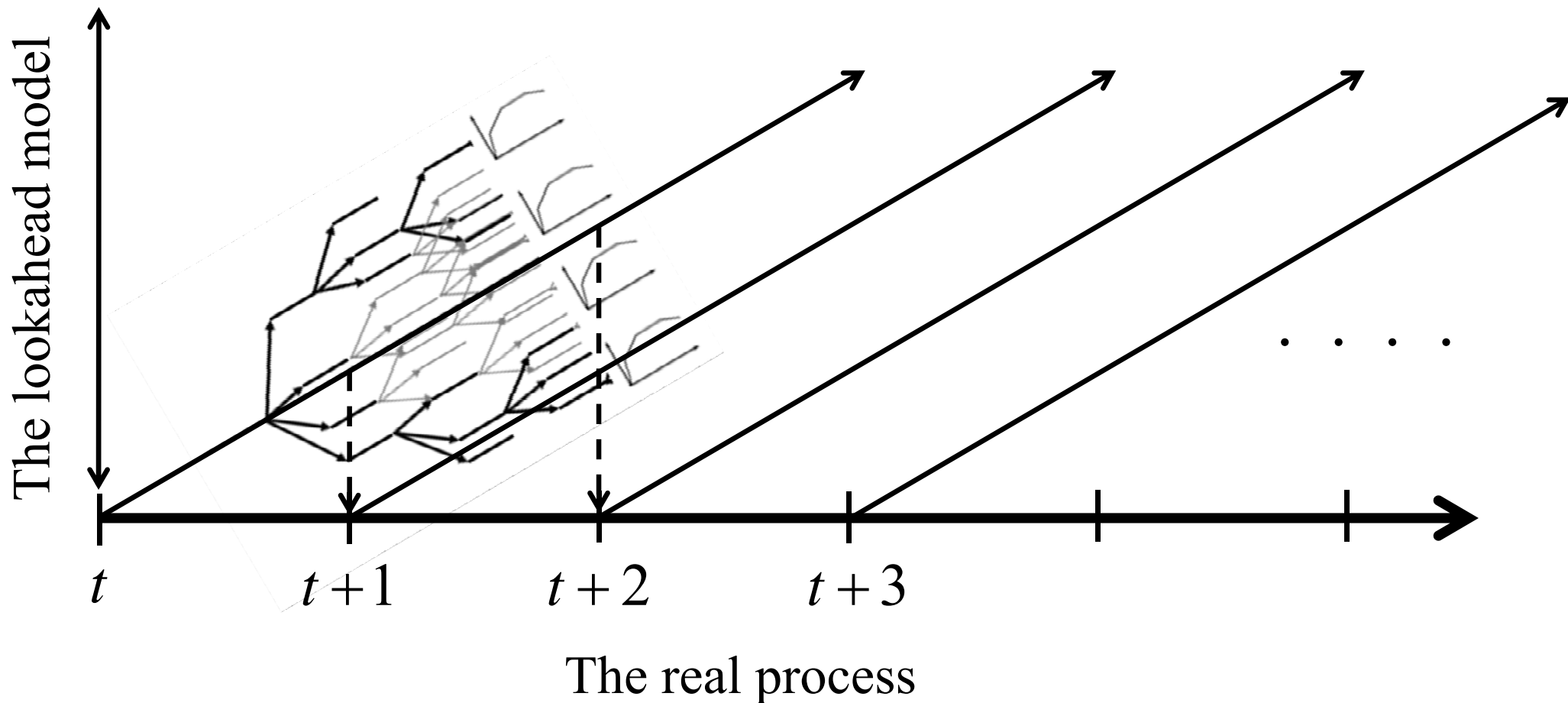
# Lookahead policies

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



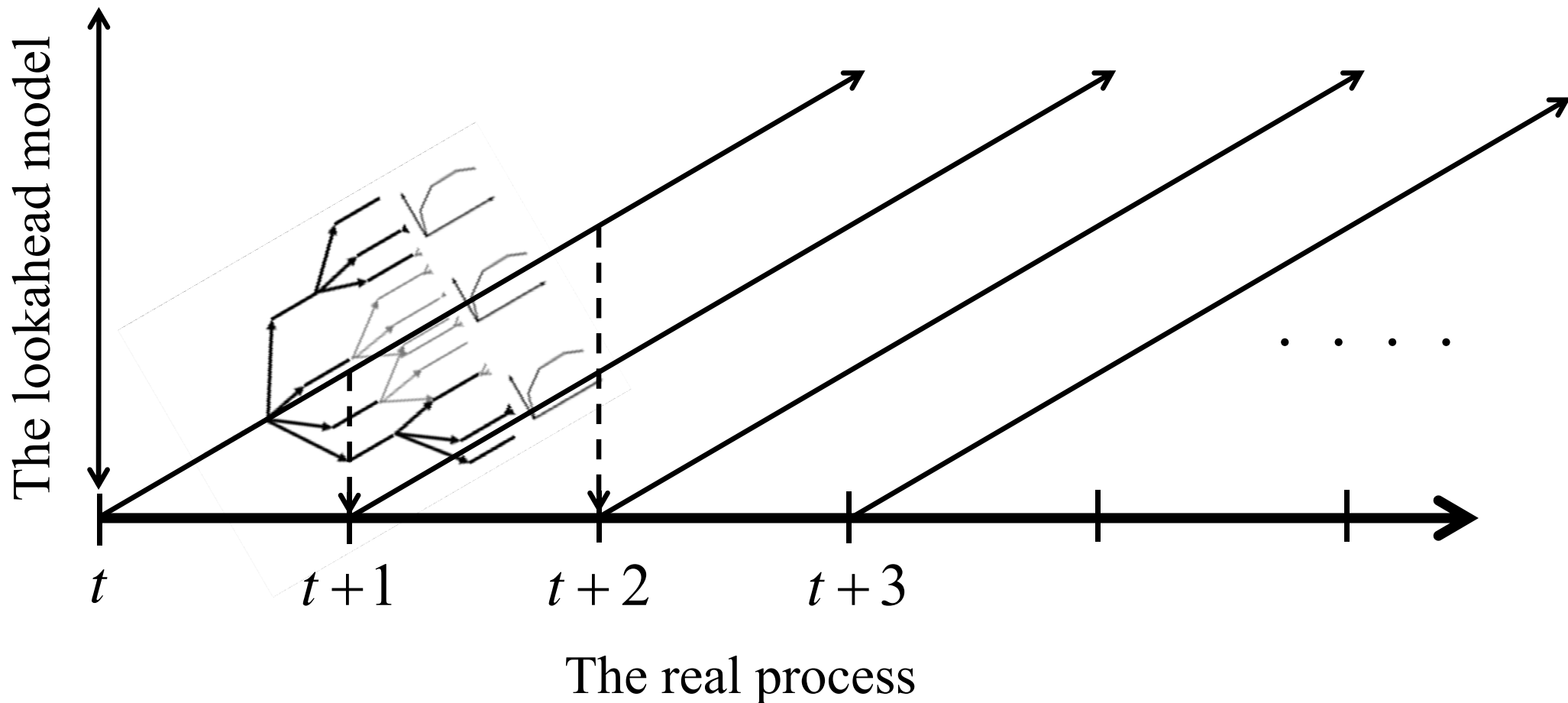
# Lookahead policies

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



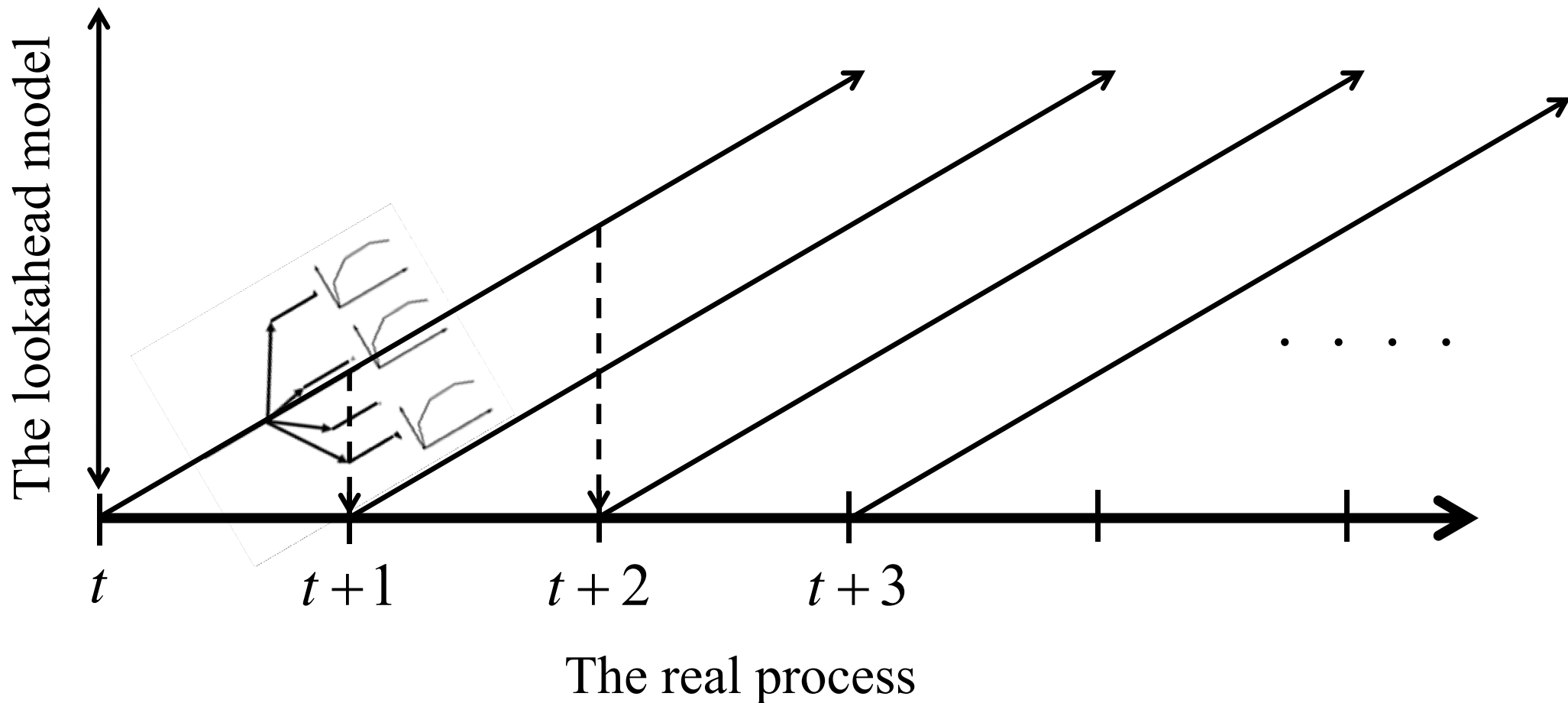
# Lookahead policies

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



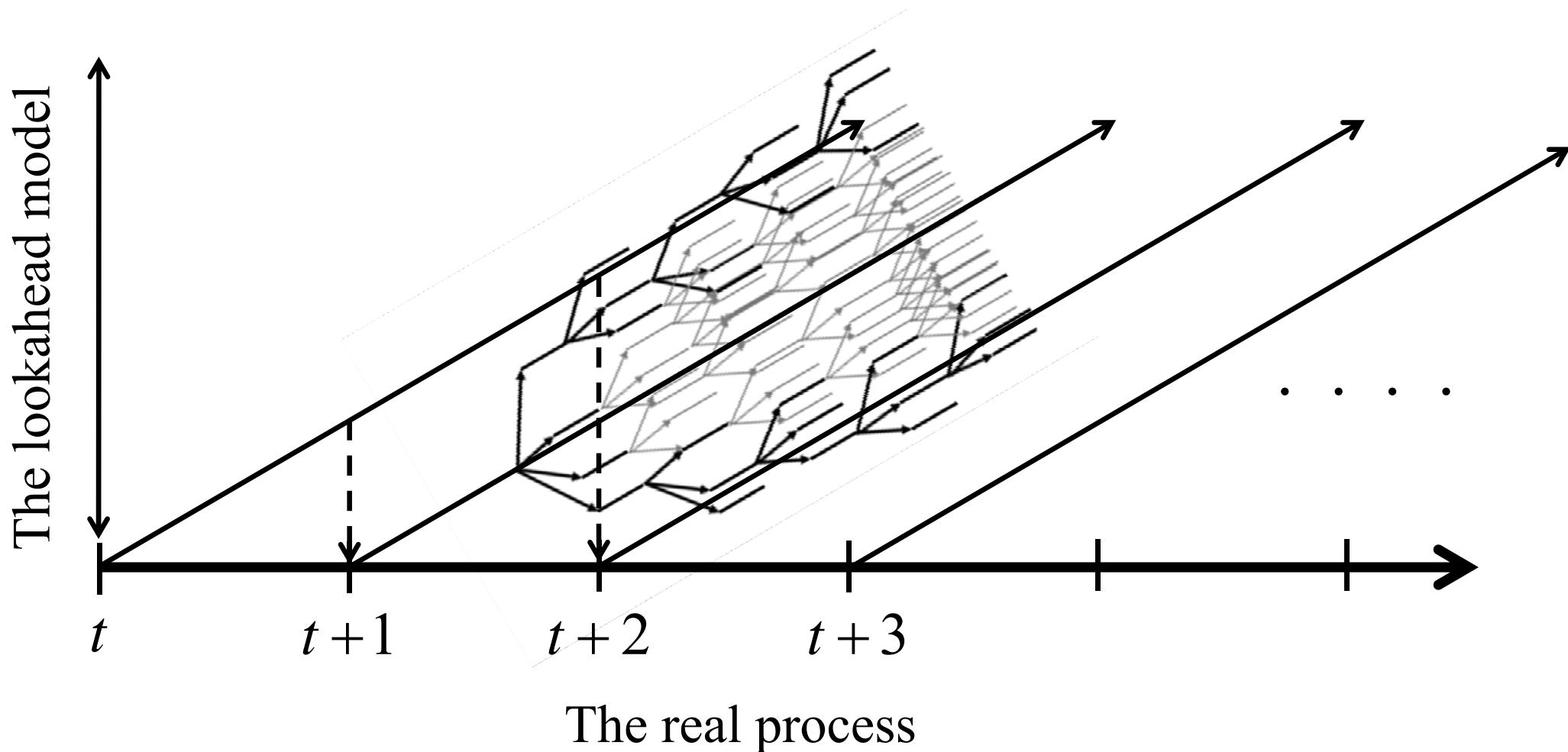
# Lookahead policies

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



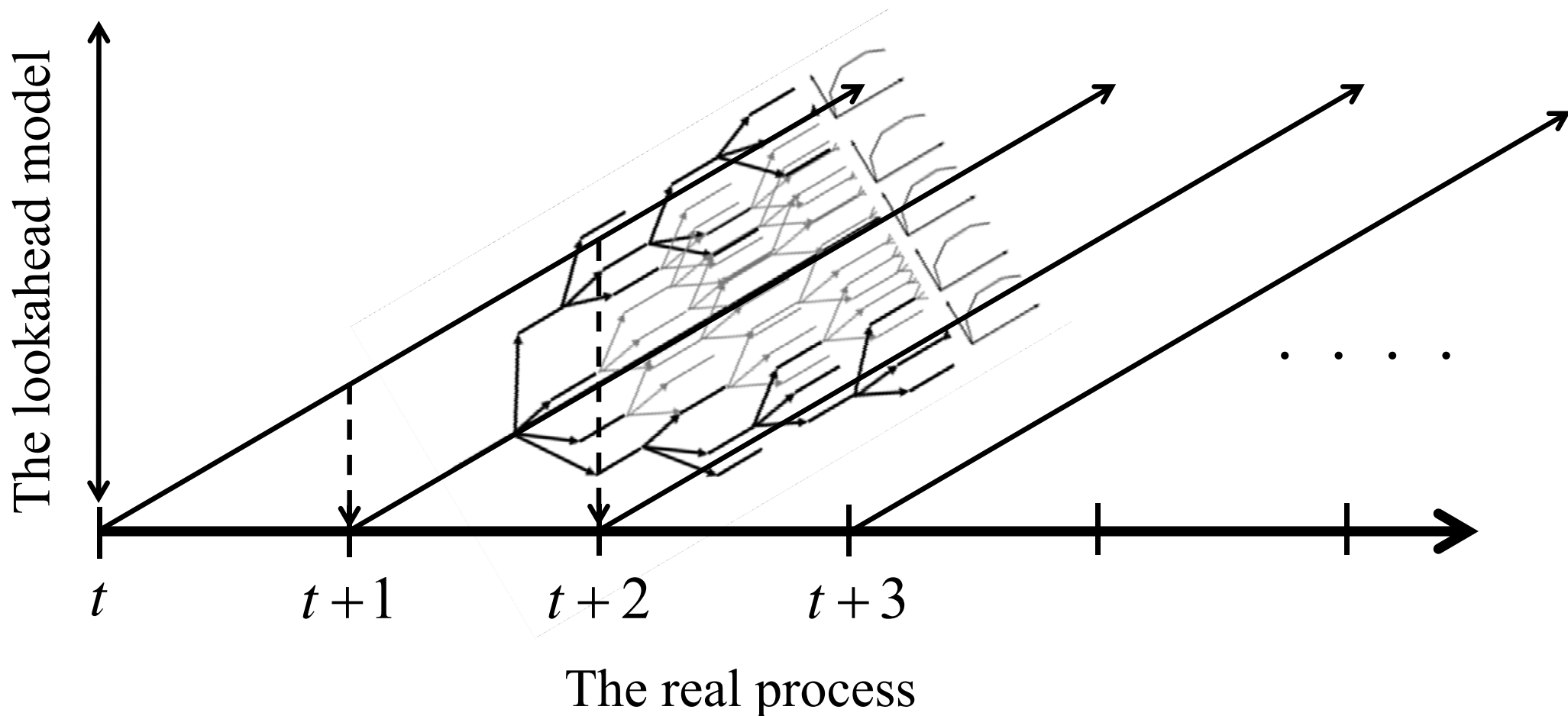
# Lookahead policies

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



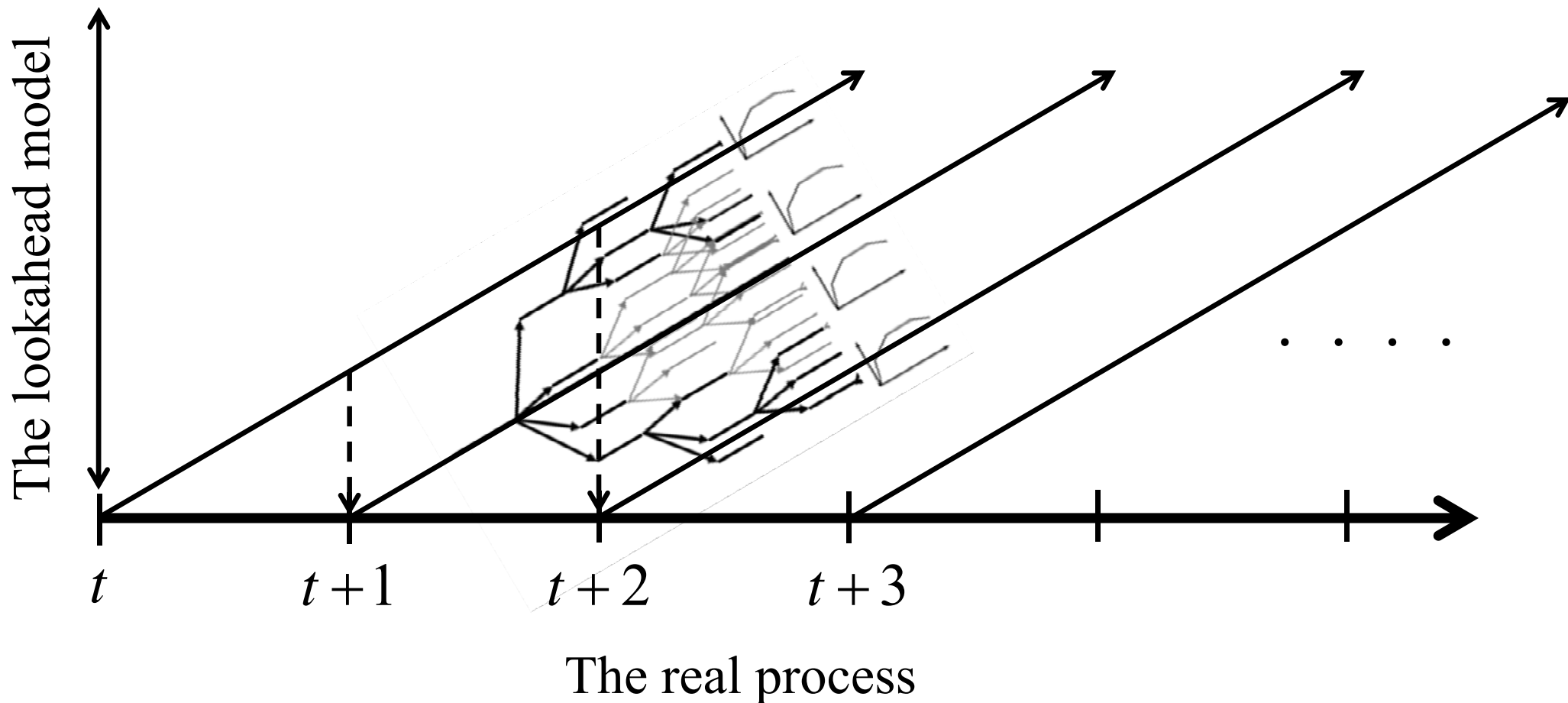
# Lookahead policies

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



# Lookahead policies

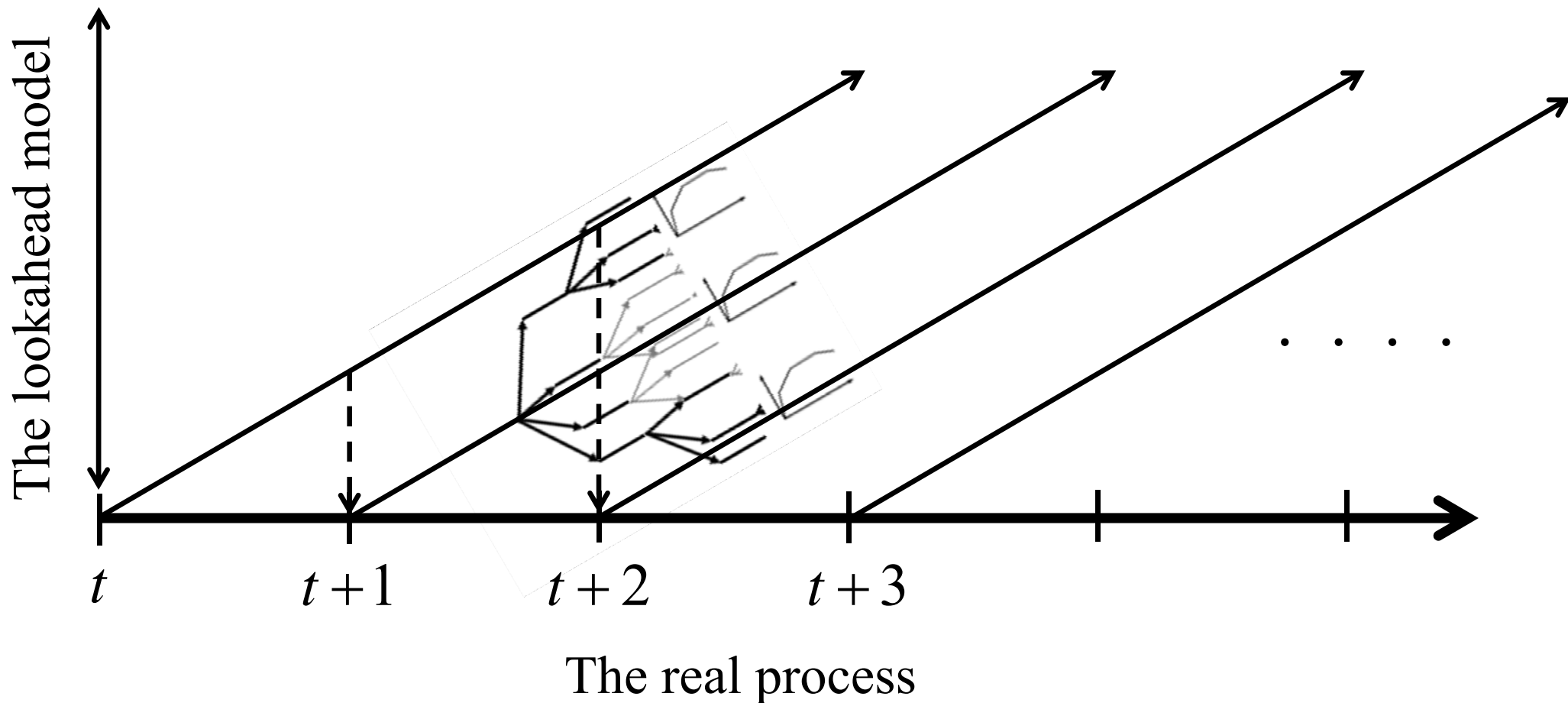
- We can then simulate this *lookahead policy* over time:





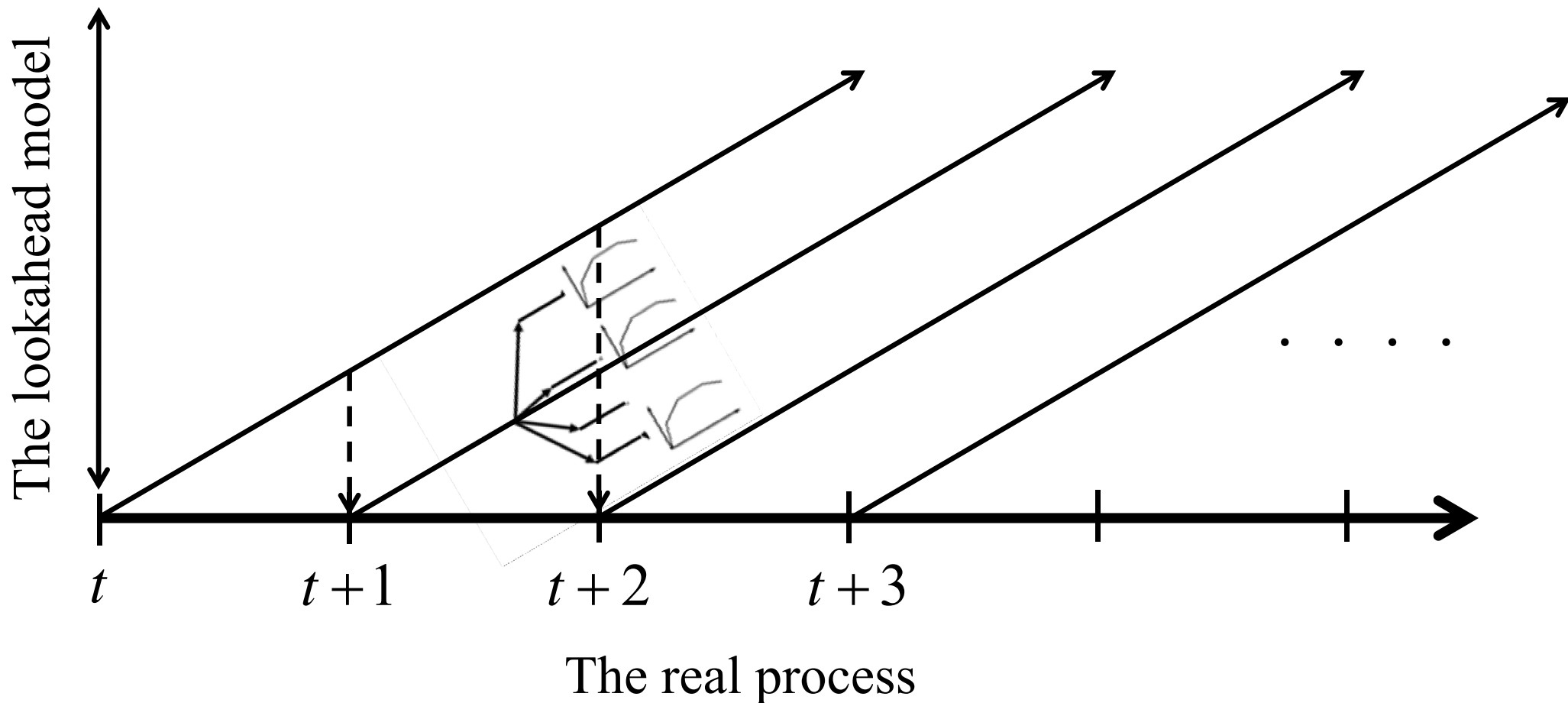
# Lookahead policies

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



# Lookahead policies

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



# Direct lookahead

## Multistage stochastic programming

- a) This material is really hard.
- b) You are never going to use it.

# Two-stage stochastic programming

## 20.5.1 The basic two-stage stochastic program

In section 4.3.2, we introduced what is known as the *two stage stochastic program* where we make an initial decision  $x_0$  (such as where to locate warehouses), after which we see information  $W_1 = W_t(\omega)$  (which might be the demands for product), and then we make a second set of decisions  $x_1(\omega)$  which depend on this information (the decisions  $x_1$  are known as the *recourse variables*).

As we did in section 4.3.2, the two-stage stochastic programming problem is written

$$\max_{x_0} (c_0 x_0 + \mathbb{E}Q_1(x_0, W_1)), \quad (20.24)$$

subject to the constraints,

$$A_0 x_0 = b_0, \quad (20.25)$$

$$x_0 \geq 0, \quad (20.26)$$

The initial decisions  $x_0$  (which determines the inventories in the warehouses) then impacts the decisions that can be made after the information (the demand) becomes known, producing the second stage problem

$$Q_1(x_0, \omega) = \max_{x_1(\omega)} c_1(\omega) x_1(\omega), \quad (20.27)$$

subject to, for all  $\omega \in \Omega$ ,

$$A_1 x_1(\omega) \leq B_1 x_0, \quad (20.28)$$

$$B_1 x_1(\omega) \leq D_1(\omega), \quad (20.29)$$

$$x_1(\omega) \geq 0. \quad (20.30)$$

We note that while  $Q_1(x_0) = \mathbb{E}Q_1(x_0, W_1)$  is a value function, the  $Q(\cdot)$  notation is standard in this literature.

# Two-stage stochastic programming

## ● Notes

- » This strategy replaces the second-stage expectation with a sampled approximation
- » Discuss two ways of representing first stage decision:
  - One decision – Means that the second stage problems are all linked.
  - One decision per scenario (outcome) – Implies that the first stage decision is allowed to see the future. We then introduce “nonanticipativity constraint”:

$$x_0(\omega) = x_0, \text{ for all } \omega \in \hat{\Omega}.$$

- » Distinguish between:
  - A two-stage stochastic programming *problem*
  - A two-stage stochastic program as an approximate lookahead for a fully sequential problem

# Modeling as a stochastic program

- An alternative strategy is to use the vocabulary of “stochastic programming.”

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

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:

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

# Modeling as a stochastic program

- An alternative strategy is to use the vocabulary of “stochastic programming.”

$$\min_{x_t \in X_t} c_t x_t + \mathbb{E}Q(x_t, \xi_{t+1})$$

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:

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

# 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  $t$ ,  $t' = t, t+1, \dots, t+H$

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

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

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

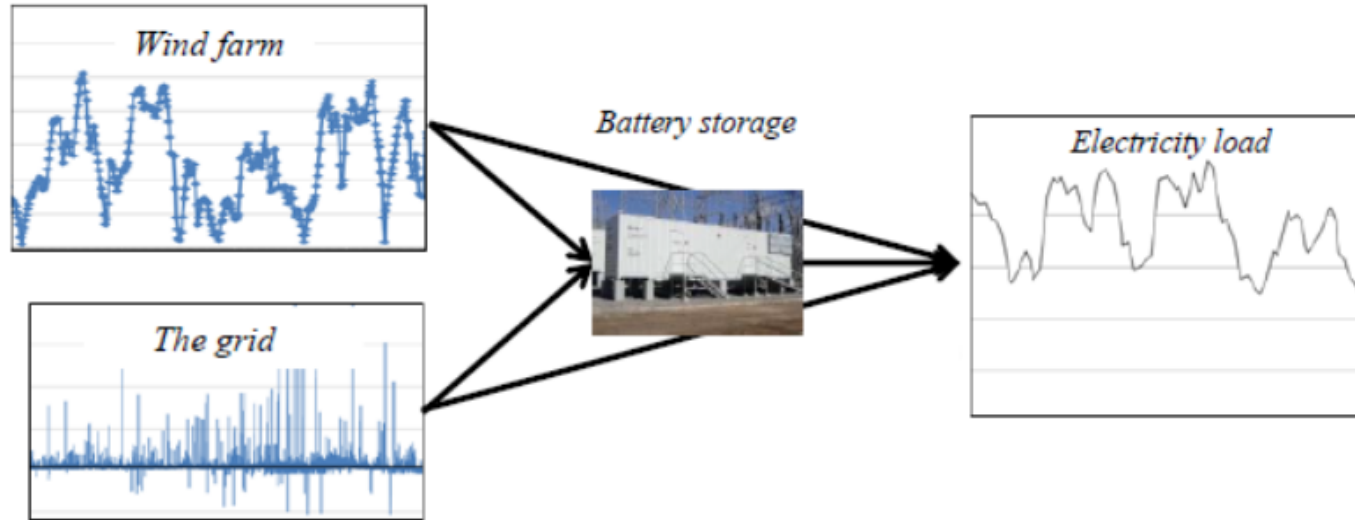
$\tilde{b}_{t,t'}$  = Forecast of right hand sides for time  $t'$  made at time  $t$

- » This notation helps avoid confusion between the base model and the lookahead model.



# Two-stage stochastic programming

## ● Energy systems application



$$\min_{x_t, (\tilde{x}_{tt'}(\tilde{\omega}_t)), t'=t+1, \dots, t+H, \tilde{\omega}_t \in \tilde{\Omega}_t} p_t (x_t^{gb} + p_t x_t^{gd}) + \sum_{\tilde{\omega}_t \in \tilde{\Omega}_t} P(\tilde{\omega}_t) \sum_{t'=t+1} (\tilde{p}_{tt'}(\tilde{\omega}_t) (\tilde{x}_{tt'}^{gb}(\tilde{\omega}_t) + \tilde{x}_{tt'}^{gd}(\tilde{\omega}_t)) - \theta^{pen} \tilde{x}_{tt'}^{slack})$$

subject to the constraints

$$\tilde{R}_{t'+1}(\tilde{\omega}_t) - (\tilde{x}_{tt'}^{gb}(\tilde{\omega}_t) + \tilde{x}_{tt'}^{sb}(\tilde{\omega}_t) - \tilde{x}_{tt'}^{bd}(\tilde{\omega}_t)) = \tilde{R}_{t'}(\tilde{\omega}_t), \quad (20.33)$$

$$\tilde{x}_{tt'}^{sd}(\tilde{\omega}_t) + \tilde{x}_{tt'}^{sb}(\tilde{\omega}_t) \leq \tilde{h}_{t'}(\tilde{\omega}_t), \quad (20.34)$$

$$\tilde{x}_{tt'}^{bd}(\tilde{\omega}_t) + \tilde{x}_{tt'}^{gd}(\tilde{\omega}_t) + \tilde{x}_{tt'}^{sd}(\tilde{\omega}_t) + \tilde{x}_{tt'}^{slack} = \tilde{D}_{t'}(\tilde{\omega}_t), \quad (20.35)$$

$$\tilde{x}_{tt'}^{gb}(\tilde{\omega}_t), \tilde{x}_{tt'}^{sb}(\tilde{\omega}_t), \tilde{x}_{tt'}^{bd}(\tilde{\omega}_t), \tilde{x}_{tt'}^{sd}(\tilde{\omega}_t), \tilde{x}_{tt'}^{slack} \geq 0. \quad (20.36)$$

# Multistage-stage stochastic programming

## 20.6.1 Modeling multistage stochastic programs

It is common to model a stochastic program as if it starts at time 0, and extends over a horizon  $t = (1, 2, \dots, T)$ , ignoring the fact that this is a lookahead model that starts at time  $t$ , and extends over a horizon  $t' = (t, t + 1, \dots, t + H)$ . This ignores the fact that the stochastic program, which is, by itself, a challenging stochastic optimization problem, is really just a lookahead policy for another stochastic optimization problem that we have been calling the base model. We are going to stay with our notation to model the fact that our lookahead model is being created at time  $t$  in the base model, and we use  $t'$  to indicate the time within the lookahead model. We also use tilde's (most of the time) to indicate variables in the lookahead model.

We begin by describing modeling assumptions used when formulating multistage (or even two-stage) stochastic programs. Most important is to separate the physical process controlled by  $\tilde{x}_{tt'}$  which determines the physical state  $\tilde{R}_{tt'}$ , and the information process  $\tilde{I}_{tt'}$  that evolves exogenously. Combined these make up the state

$$\tilde{S}_{tt'} = (\tilde{R}_{tt'}, \tilde{I}_{tt'}).$$

These are treated separately, which also means that decisions cannot have an impact on information, an assumption that we do not require in any of our other classes of policies (PFAs, CFAs, VFAs). It is important to recognize that just as  $\tilde{R}_{tt'}$  is the physical state at time  $t'$ ,  $\tilde{I}_{tt'}$  is only the information we need at time  $t'$  to model the lookahead model from time  $t'$  onward. Thus, while  $\tilde{I}_{tt'}$  might include the entire history  $h_{tt'}$ , in most applications  $\tilde{I}_{tt'}$  is likely to be much more compact than  $h_{tt'}$  (but it may still be quite high dimensional).

# Multistage-stage stochastic programming

## ● Physical process

To model the physical process, we let  $\tilde{R}_{tt'}$  be a vector of resources, where an element might be  $\tilde{R}_{t,t'k}$  where  $k$  is a location (the number of freight containers at port or rail yard  $k$ ), or a type of good  $k$ . We might also use  $\tilde{R}_{tt'r}$  where  $r = (r_1, \dots, r_M)$  is a vector of attributes, perhaps to describe a driver or complex equipment such as an aircraft. The problem that arises when  $r$  is a vector is that the dimensionality of  $\tilde{R}_{tt'}$  becomes extremely large. For this reason, we will assume that  $\tilde{R}_{tt'} = (\tilde{R}_{tt'k})_{k \in \mathcal{K}}$  where the size of the set  $\mathcal{K}$  is “not too large” (100’s, perhaps 1,000’s, maybe 10,000).

We then assume that our decision  $\tilde{x}_{tt'}$  at time  $t'$  in the lookahead model is subject to constraints of the form

$$\tilde{A}_{tt'} \tilde{x}_{tt'} = \tilde{R}_{tt'},$$

where these are typically flow conservation constraints. We then assume that this vector evolves over time according to

$$\tilde{R}_{t,t'+1} = \tilde{B}_{tt'} \tilde{x}_{tt'} + \delta \tilde{R}_{t,t'+1}, \quad (20.37)$$

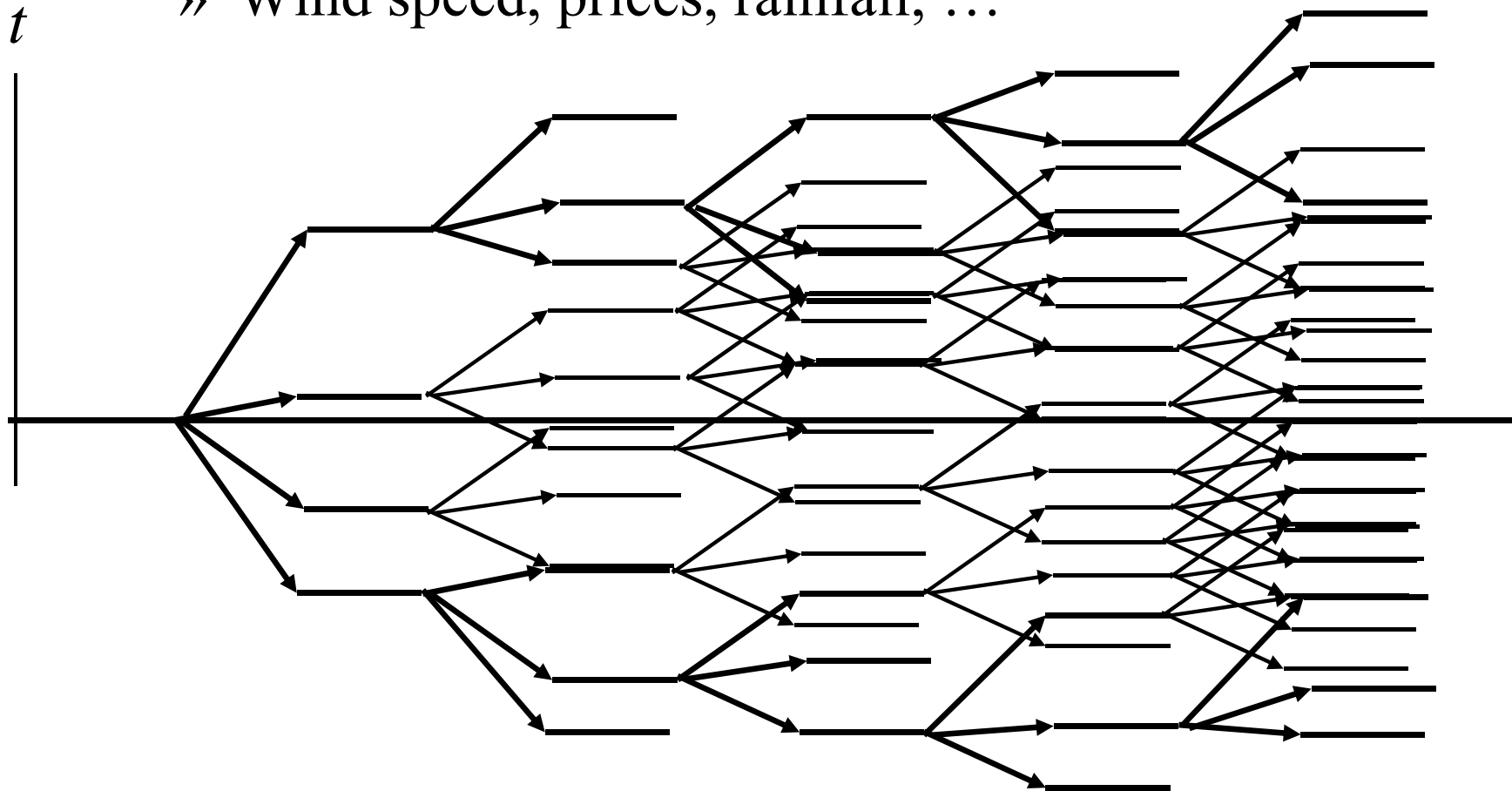
$$\tilde{A}_{t,t'+1} \tilde{x}_{t,t'+1} = \tilde{R}_{t,t'+1}, \quad (20.38)$$

where  $\delta \tilde{R}_{t,t'+1}$  represents exogenous changes (new arrivals, departures, theft of product, rainfall). Previously, we used  $\hat{R}_{tt'}$  to capture these exogenous changes, but in this section, we are modeling every variable indexed by  $t'$  as if it first becomes known at time  $t'$ .

# Multistage-stage stochastic programming

- The scenario tree models the evolution of information

» Wind speed, prices, rainfall, ...





# Multistage-stage stochastic programming

## ● Information process

Separate from the physical process is the information process. The data in our linear program at time  $t'$  in the lookahead model consists of costs  $\tilde{c}_{tt'}$ , the constraint matrices  $\tilde{A}_{tt'}$  and  $\tilde{B}_{tt'}$ , and the exogenous changes in supplies  $\delta\tilde{R}_{tt'}$  (that enters the problem in  $R_{t,t'+1}$  at time  $t' + 1$ ). We let  $\tilde{W}_{tt'} = (\tilde{A}_{tt'}, \tilde{B}_{tt'}, \tilde{c}_{tt'}, \delta\tilde{R}_{tt'})$  be our exogenous information process if everything is random (there are many applications where only  $\delta\tilde{R}_{tt'}$  is random, while the other variables are time-dependent but deterministic).

Let  $\tilde{h}_{tt'}$  be the history of our information process which we can write

$$\begin{aligned}\tilde{h}_{tt'} &= (\tilde{W}_{tt}, \tilde{W}_{t,t+1}, \dots, \tilde{W}_{tt'}), \\ &= ((\tilde{A}_{tt}, \tilde{B}_{tt}, \tilde{c}_{tt}, \delta\tilde{R}_{tt}), \dots, (\tilde{A}_{tt'}, \tilde{B}_{tt'}, \tilde{c}_{tt'}, \delta\tilde{R}_{tt'})).\end{aligned}$$

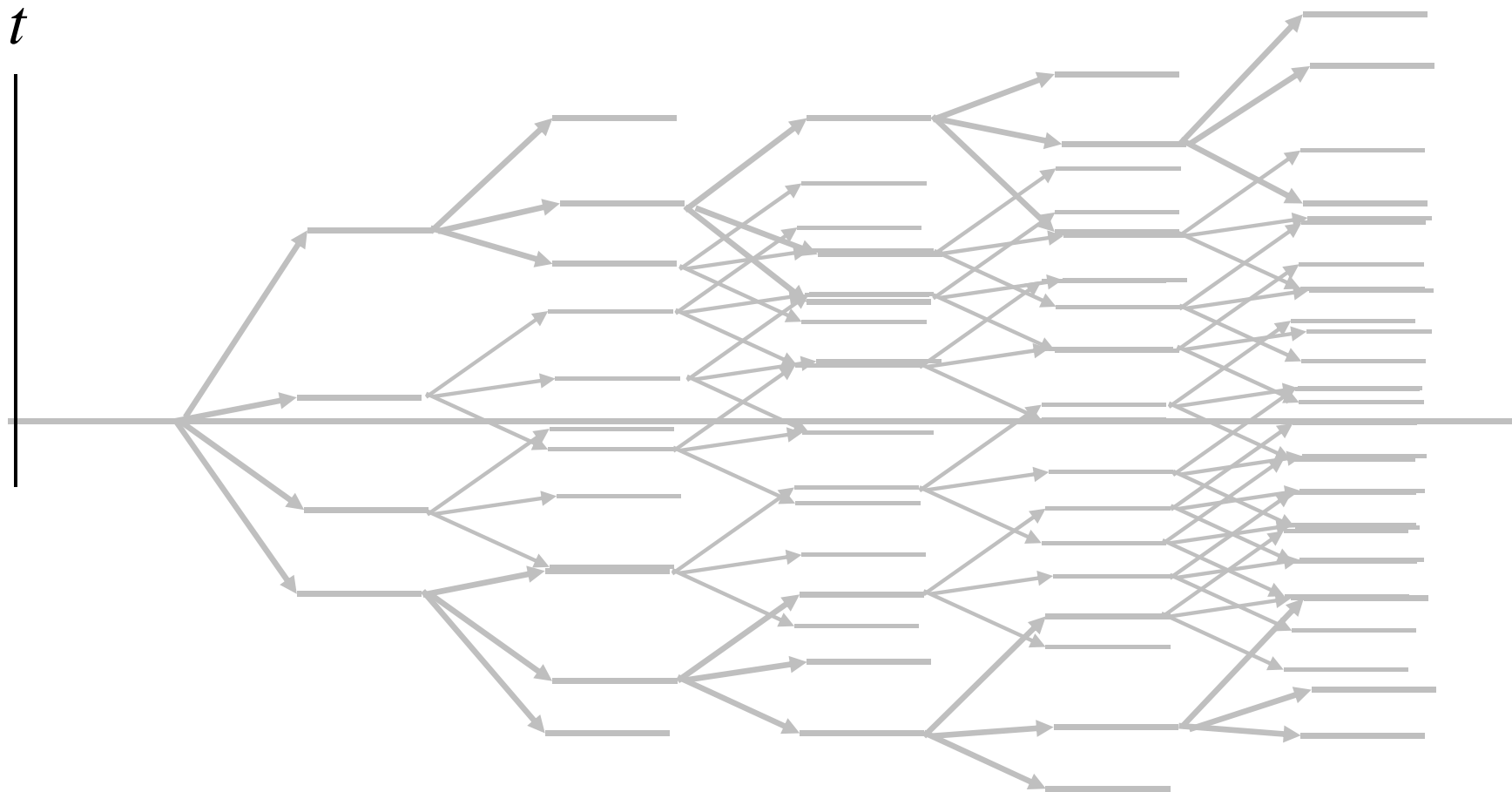
We next let  $\tilde{\Omega}_t$  be the set of all the sample paths  $\tilde{W}_{tt}, \dots, \tilde{W}_{t,t+H}$  over our horizon. This means that when we use an index  $\tilde{\omega}_t \in \tilde{\Omega}_t$ , it refers to the information over the entire planning horizon (in our lookahead model). Specifying  $\tilde{\omega}_t$  is like specifying the entire future (within the lookahead model).

It is useful to be able to label all the elements  $\tilde{\omega}_t$  that correspond to a particular history  $\tilde{h}_{tt'}$ . For this purpose, we define the set of outcomes that share a history, which we write as

$$\mathcal{H}_t(\tilde{h}_{tt'}) = \{\tilde{\omega}_t \in \tilde{\Omega}_t | (\tilde{W}_{tt}, \dots, \tilde{W}_{tt'}) = \tilde{h}_{tt'}\}.$$

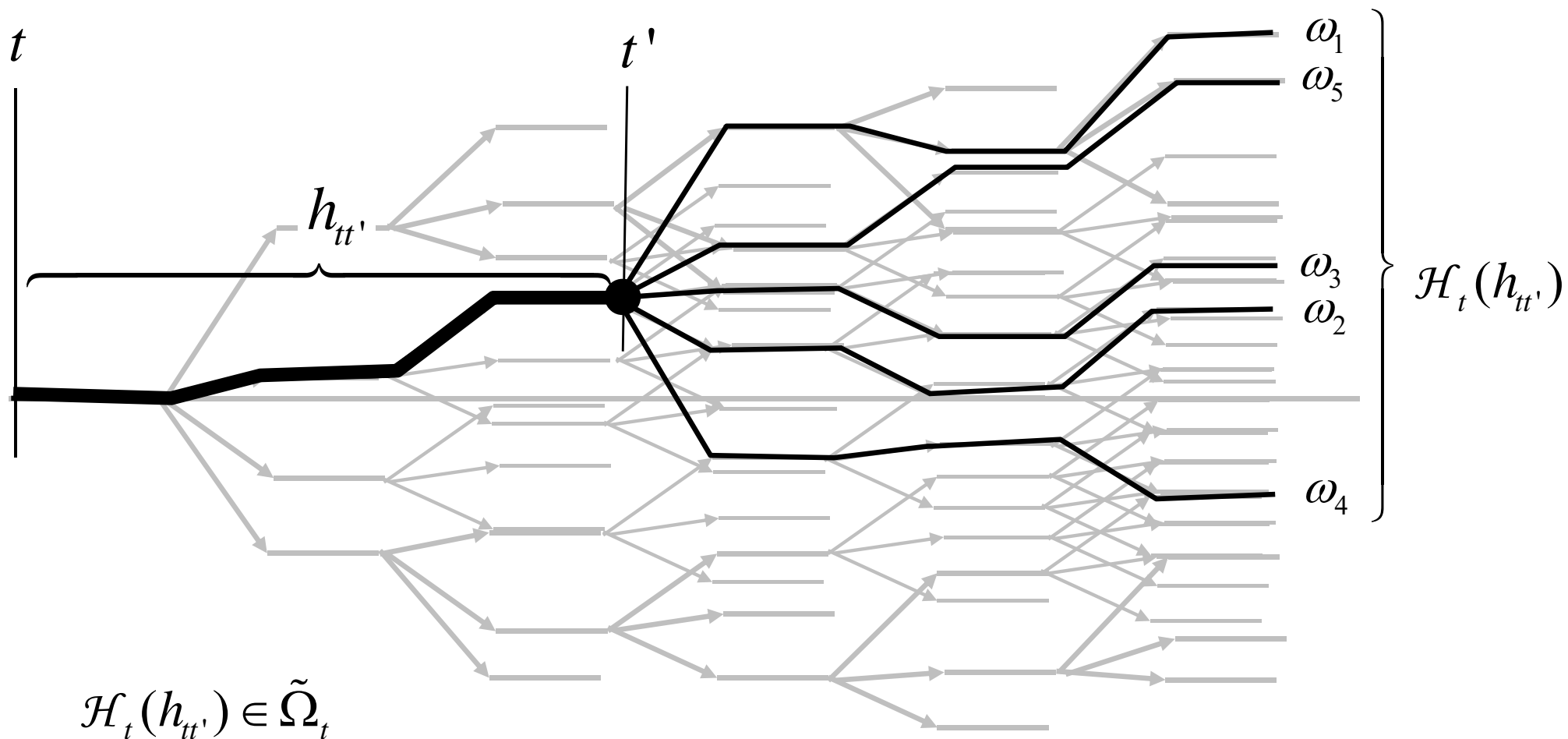
# Multistage-stage stochastic programming

- The information process:
  - » The sampled process  $\tilde{\Omega}_t$



# Multistage-stage stochastic programming

- The information process:



# Multistage-stage stochastic programming

## ● The state variable

- » Resource state, and information state, in the lookahead model

$$\begin{aligned}\tilde{S}_{tt'} &= (\tilde{R}_{tt'}, (\tilde{A}_{tt'}, \tilde{B}_{tt'}, \tilde{c}_{tt'}, \delta \tilde{R}_{tt'})) \\ &= (\tilde{R}_{tt'}, \tilde{I}_{tt'}), \\ \tilde{W}_{tt'} &= (\tilde{A}_{tt'}, \tilde{B}_{tt'}, \tilde{c}_{tt'}, \delta \tilde{R}_{tt'}).\end{aligned}$$

Conditioning on a set  $\mathcal{H}_t(h_{tt'})$  is the same as fixing a node in the scenario tree corresponding to the end of the history  $h_{tt'}$  in figure 20.13. Some authors even replace the history  $h_{tt'}$  with the index of the node corresponding to the end of the history at time  $t'$ , but requires that we condition on the entire history. While there are situations where this may be appropriate (applications in finance may use this), in most applications the information state does not require the entire history. In fact, in our problem the information state is just the exogenous information, which is to say

$$\begin{aligned}\tilde{I}_{tt'} &= \tilde{W}_{tt'} \\ &= (\tilde{A}_{tt'}, \tilde{B}_{tt'}, \tilde{c}_{tt'}, \delta \tilde{R}_{tt'}).\end{aligned}$$





# Multistage-stage stochastic programming

- Classical formulation:

A classical approach for writing a multistage stochastic optimization problems, solved at time  $t$  over a horizon  $t' = t, \dots, t + H$  is given by

$$\begin{aligned} \max_{\substack{\bar{A}_{t0}\bar{x}_{t0}=\bar{R}_{t0} \\ \bar{x}_{t0}\geq 0}} \tilde{c}_{t0}\tilde{x}_{t0} + \mathbb{E} \left[ \max_{\substack{\bar{B}_{t0}\bar{x}_{t0}+\bar{A}_{t1}\bar{x}_{t1}=\bar{R}_{t1} \\ \bar{x}_{t1}\geq 0}} \tilde{c}_{t1}\tilde{x}_{t1} + \mathbb{E} \left[ \dots + \right. \right. \\ \left. \left. \mathbb{E} \left[ \max_{\substack{\bar{B}_{t,T-1}\bar{x}_{t,T-1}+\bar{A}_{tT}\bar{x}_{tT}=\bar{R}_{tT} \\ \bar{x}_{tT}\geq 0}} \tilde{c}_{tT}, \tilde{x}_{tT} \mid \mathcal{H}_t(h_{tT}) \right] \dots \mid \mathcal{H}_t(h_{t2}) \right] \mid \mathcal{H}_t(h_{t1}) \right]. \end{aligned}$$

- State/policy formulation

Given this structure, we may condition all of our expectations on the information state  $\tilde{I}_{tt'}$  rather than the full state  $\tilde{S}_{tt'}$ , in which case equation (20.39) is equivalent to

$$\begin{aligned} \max_{\bar{x}_{tt} \in \mathcal{X}_t(S_t)} C(\tilde{S}_{tt}, \tilde{x}_{tt}) + \mathbb{E} \left[ \max_{\bar{x}_{t1} \in \mathcal{X}_1(\tilde{S}_{t1})} C(\tilde{S}_{t1}, \tilde{x}_{t1}) + \mathbb{E} \left[ \dots \right. \right. \\ \left. \left. + \mathbb{E} \left[ \max_{\bar{x}_{tT} \in \mathcal{X}_T(\tilde{S}_{tT})} C(\tilde{S}_{tT}, \tilde{x}_{tT}) \mid \tilde{I}_{tT} \right] \dots \mid \tilde{I}_{t2} \right] \mid \tilde{I}_{t1} \right], \quad (20.40) \end{aligned}$$

$$\tilde{X}_{tt'}^{SP}(\tilde{S}_{tt'}) = \arg \max_{\bar{x}_{tt'} \in \mathcal{X}_{t'}(\tilde{S}_{t'})} C(\tilde{S}_{tt'}, \tilde{x}_{tt'}) + \mathbb{E} \left[ \dots \mid \tilde{I}_{tt'} \right]$$

# Multistage-stage stochastic programming

- Stochastic programming with scenario trees

- » Associate each node with an information state  $\tilde{I}_{tt'}$

- » Expectation over sampled events out of an information node:

The outcomes  $\tilde{\omega}_{tt'} \in \tilde{\Omega}_{tt'}$  each take us to a downstream information node  $\tilde{I}_{t,t+1}$ , so there is a downstream information state for each element of  $\tilde{\Omega}_{tt'}(\tilde{I}_{tt'})$ . We assign a probability  $\tilde{p}_{tt'}(\tilde{\omega}_{tt'}|\tilde{I}_{tt'})$  for each of these outcomes, where we often have

$$\tilde{p}_{tt'}(\tilde{\omega}_{tt'}|\tilde{I}_{tt'}) = \frac{1}{|\tilde{\Omega}_{tt'}(\tilde{I}_{tt'})|}.$$

- » LP at node  $\tilde{I}_{tt'}$ , indexes all variables (costs, decisions, constraints) with  $\tilde{I}_{tt'}$

# Multistage-stage stochastic programming

- Stochastic programming with scenario trees

- » There is a linear program at each information node :

$$C(\tilde{S}_{t1}, \tilde{x}_{t1}) = \tilde{c}_{tt'}(\tilde{I}_{tt'})\tilde{x}_{tt'}(\tilde{I}_{tt'}),$$

we can write our stochastic linear program as

$$\max_{x_{tt'}(i), t'=t, \dots, t+H, i \in \mathcal{I}} \sum_{t'=t}^{t+H} \tilde{c}_{tt'}(i)\tilde{x}_{tt'}(i),$$

subject to (20.41), the constraints at time  $(t, t')$

$$\tilde{A}_{tt'}(\tilde{I}_{tt'})x_{tt'}(\tilde{I}_{tt'}) = \tilde{R}_{tt'}(\tilde{I}_{tt'}). \quad \heartsuit$$

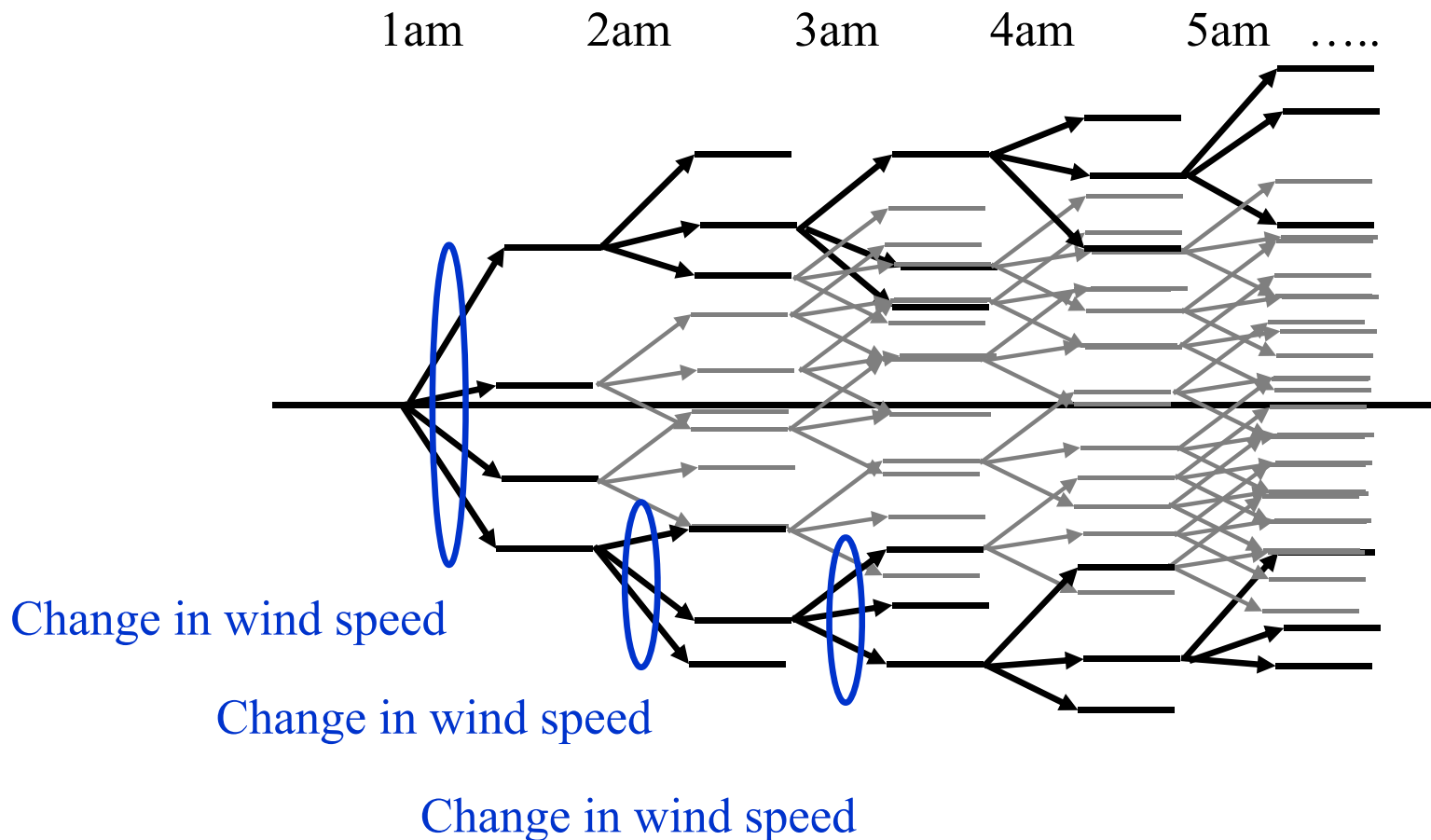
- » Resource variables are linked through a large set of linear equations:

$$\tilde{R}_{t, \nu+1}(\tilde{I}_{t, \nu+1}) = \tilde{B}_{tt'}(\tilde{I}_{tt'})\tilde{x}_{tt'}(\tilde{I}_{tt'}) + \delta\tilde{R}_{t, \nu+1}(\tilde{I}_{t, \nu+1}).$$

- » This creates one *very* large linear program. Not surprising that almost no-one actually formulates these.

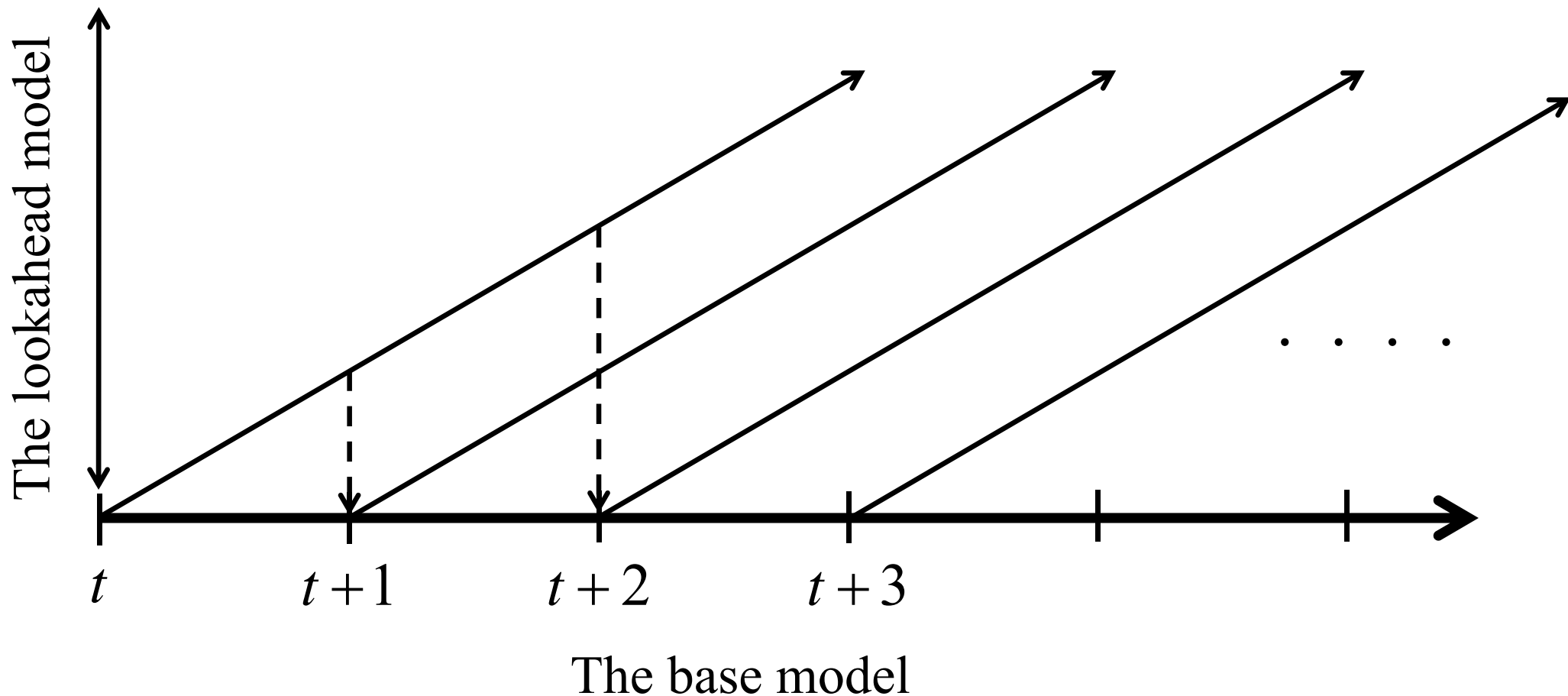
# Multistage lookahead models

- From multistage lookahead ...



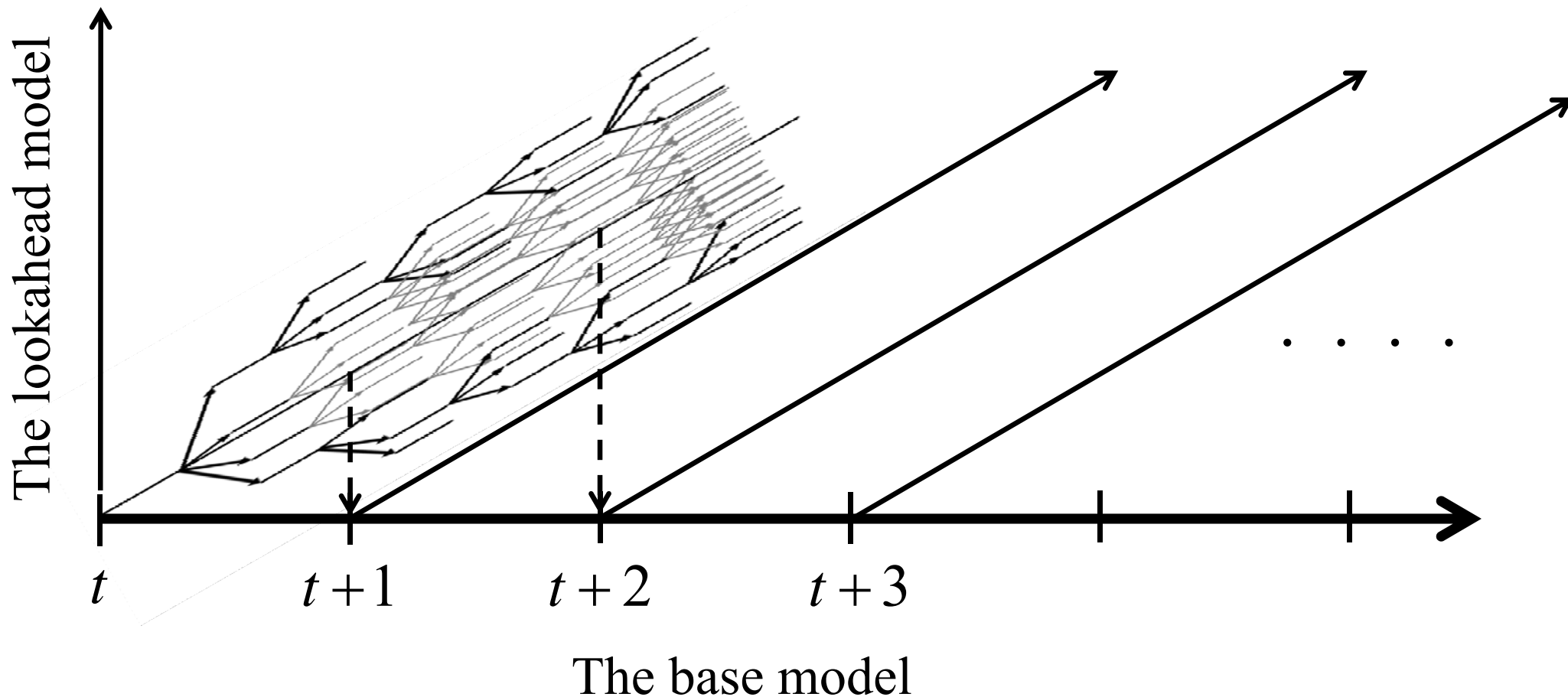
# Multistage lookahead models

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



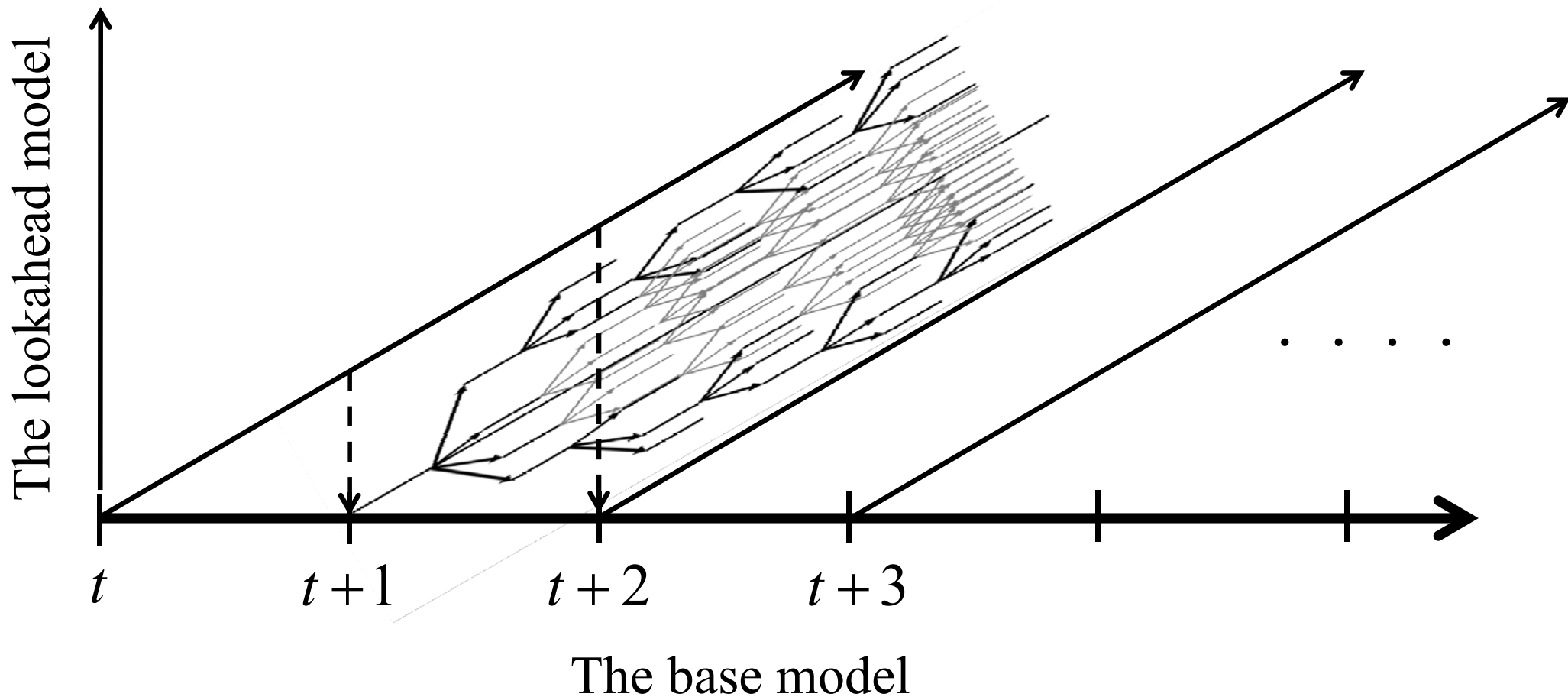
# Multistage lookahead models

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



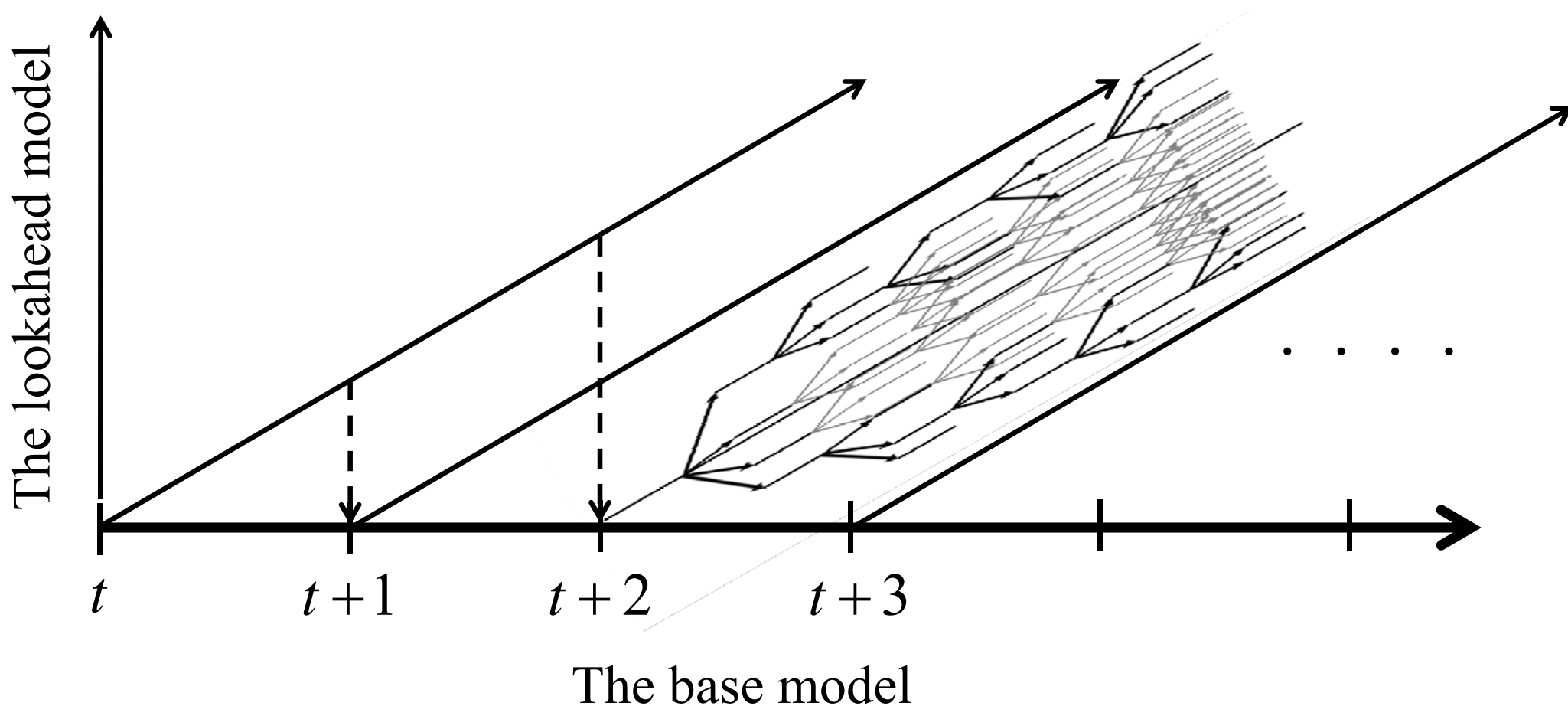
# Multistage lookahead models

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



# Multistage lookahead models

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





# Stochastic lookahead policies

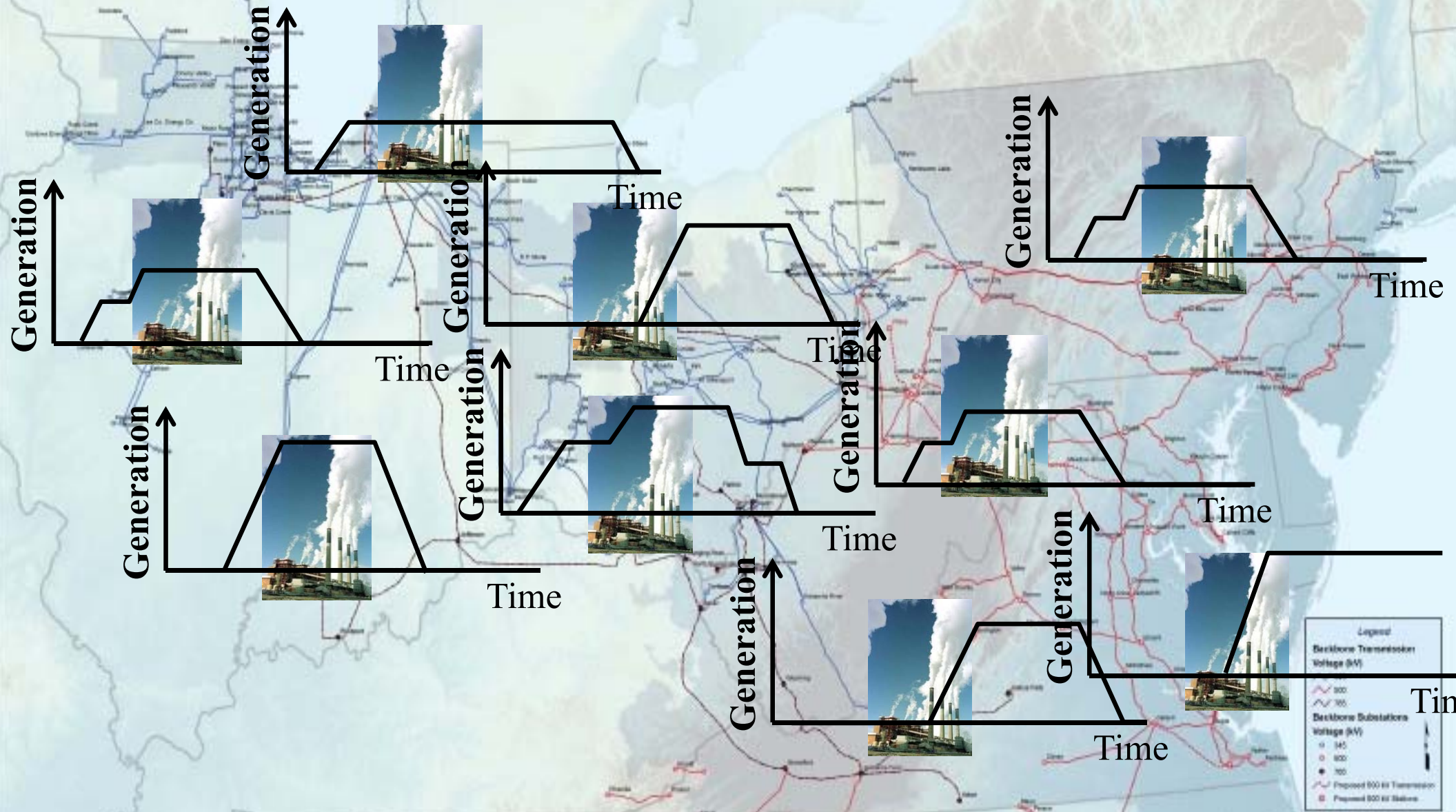
---

- Some common misconceptions about stochastic programming (for sequential problems):
  - » Solving a “stochastic program” is hard, but getting an optimal solution does not produce an optimal policy.
  - » Bounds on the quality of the solution to a stochastic program is not a bound on the quality of the policy.
  - » We only care about the quality of the policy, which can *only* be evaluated using a stochastic *base model*.

# Direct lookahead

Scenario trees with two-stage  
approximation

# The unit commitment problem (for PJM) Planning tomorrow's schedule



# Stochastic programming

---

## ● Strategies for generating scenarios

### » 1) Pull sample paths from history

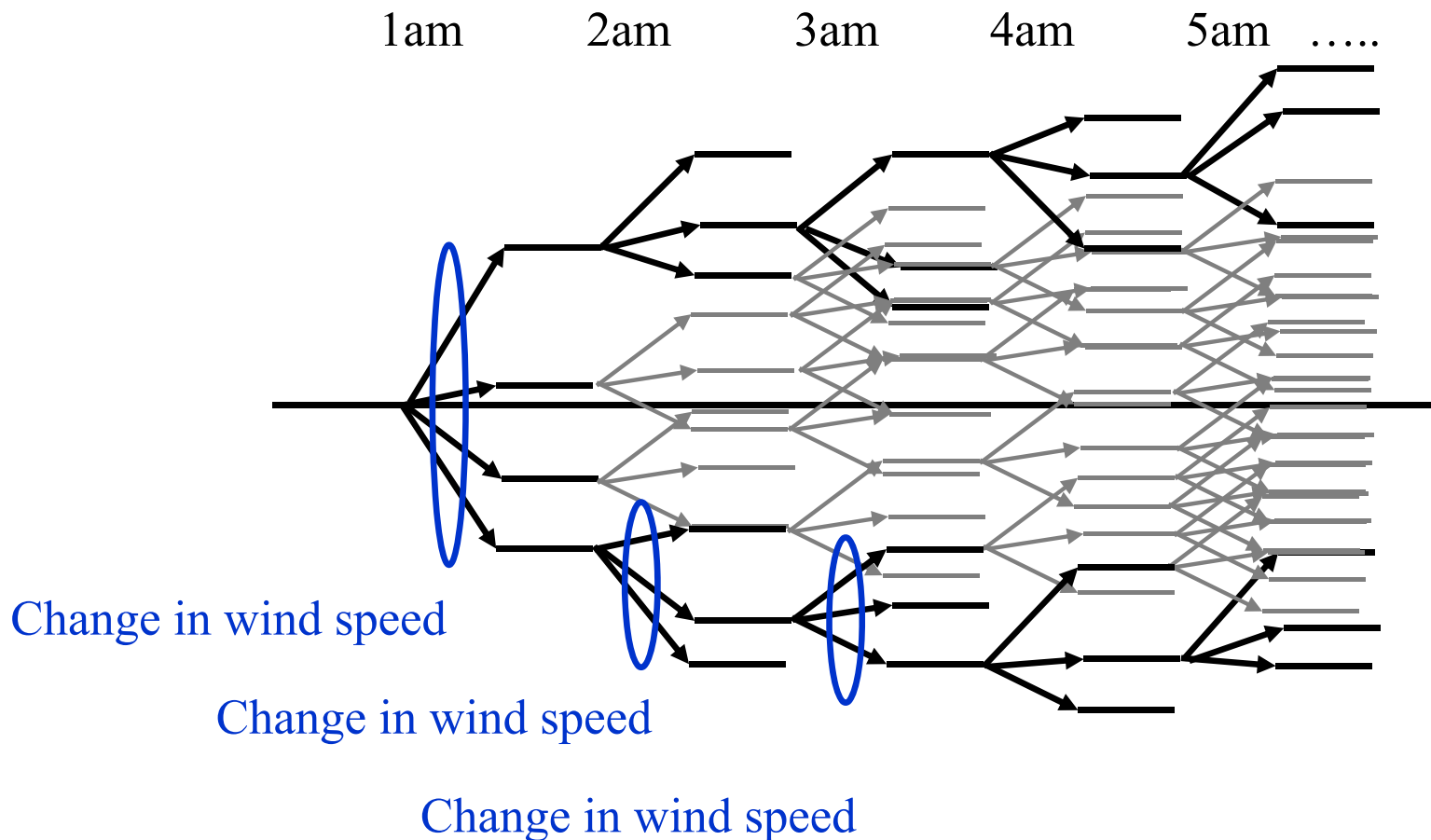
- Avoids modeling assumptions
- Will not look like the branching tree

### » 2) Build a mathematical model

- Draw on the tools from uncertainty modeling (e.g. chapter 10) to create a mathematical model of the process.
- Then, simulate the branching process of a scenario tree.

# Multistage lookahead models

- From multistage lookahead ...



# Lookahead policies

- Deterministic lookahead

$$X_t^{LA-D}(S_t) = \arg \min_{\tilde{x}_t, \tilde{x}_{t,t+1}, \dots, \tilde{x}_{t,t+T}} C(\tilde{S}_t, \tilde{x}_t) + \sum_{t'=t+1}^T \gamma^{t'-t} C(\tilde{S}_{t'}, \tilde{x}_{t'})$$

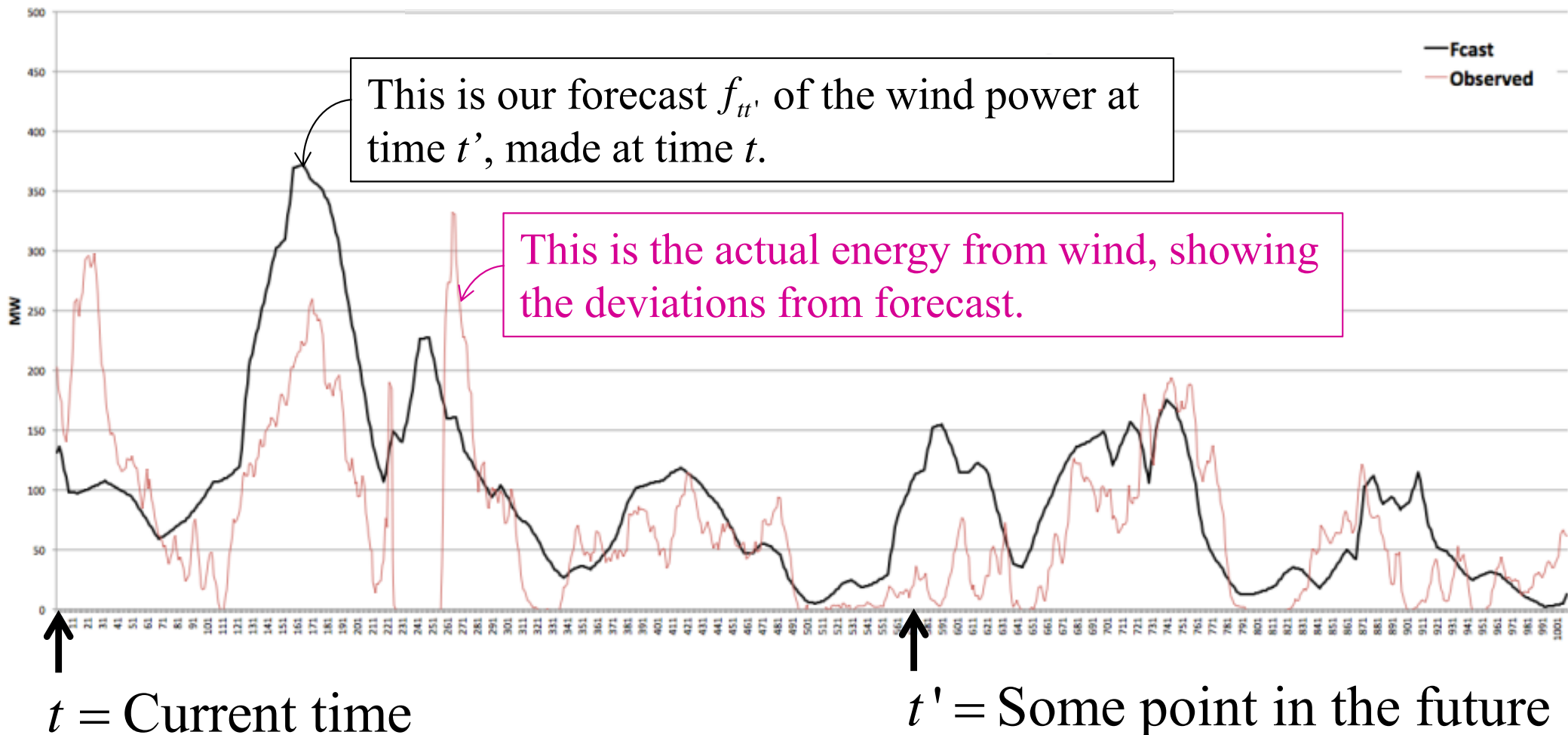
- Stochastic lookahead (with two-stage approximation)

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

Scenario trees

# Modeling stochastic wind

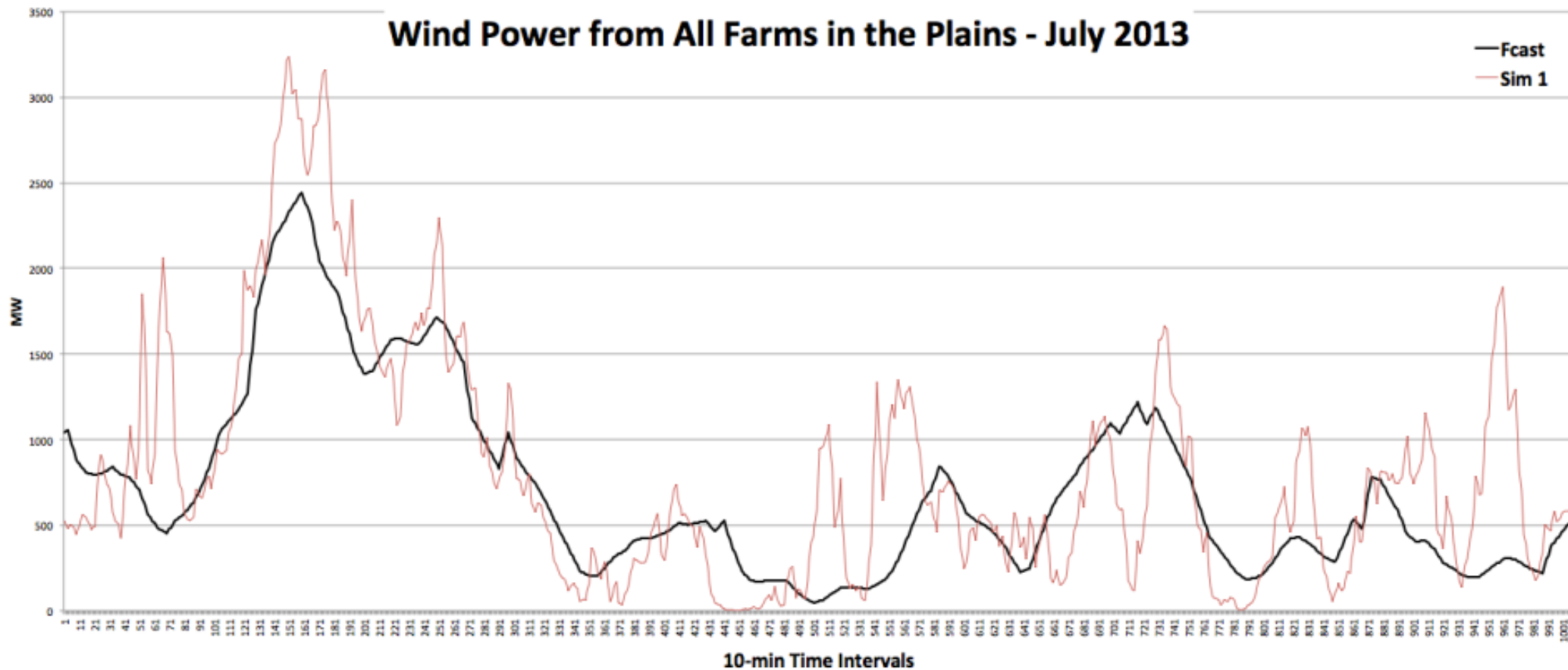
- Actual vs. forecasted energy from wind





# Modeling stochastic wind

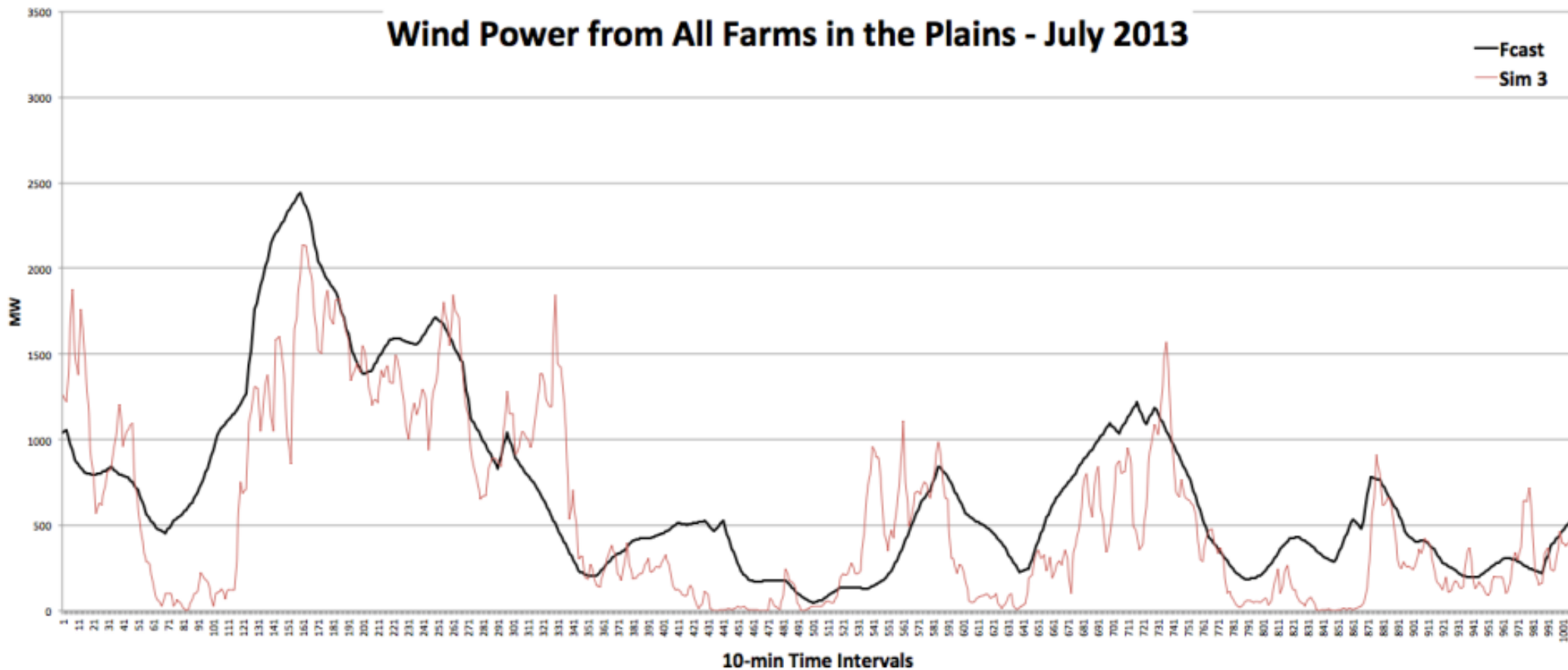
- Creating wind scenarios (Scenario #1)





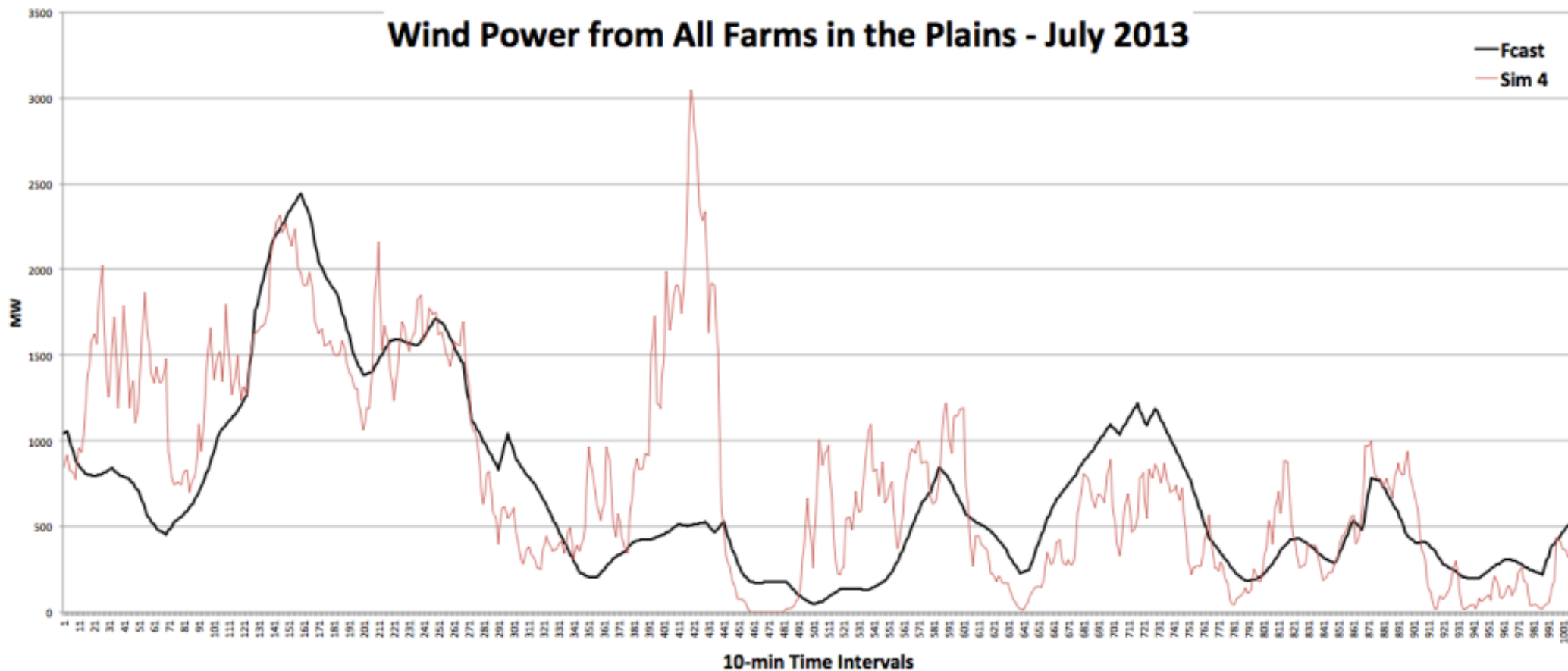
# Modeling stochastic wind

- Creating wind scenarios (Scenario #3)



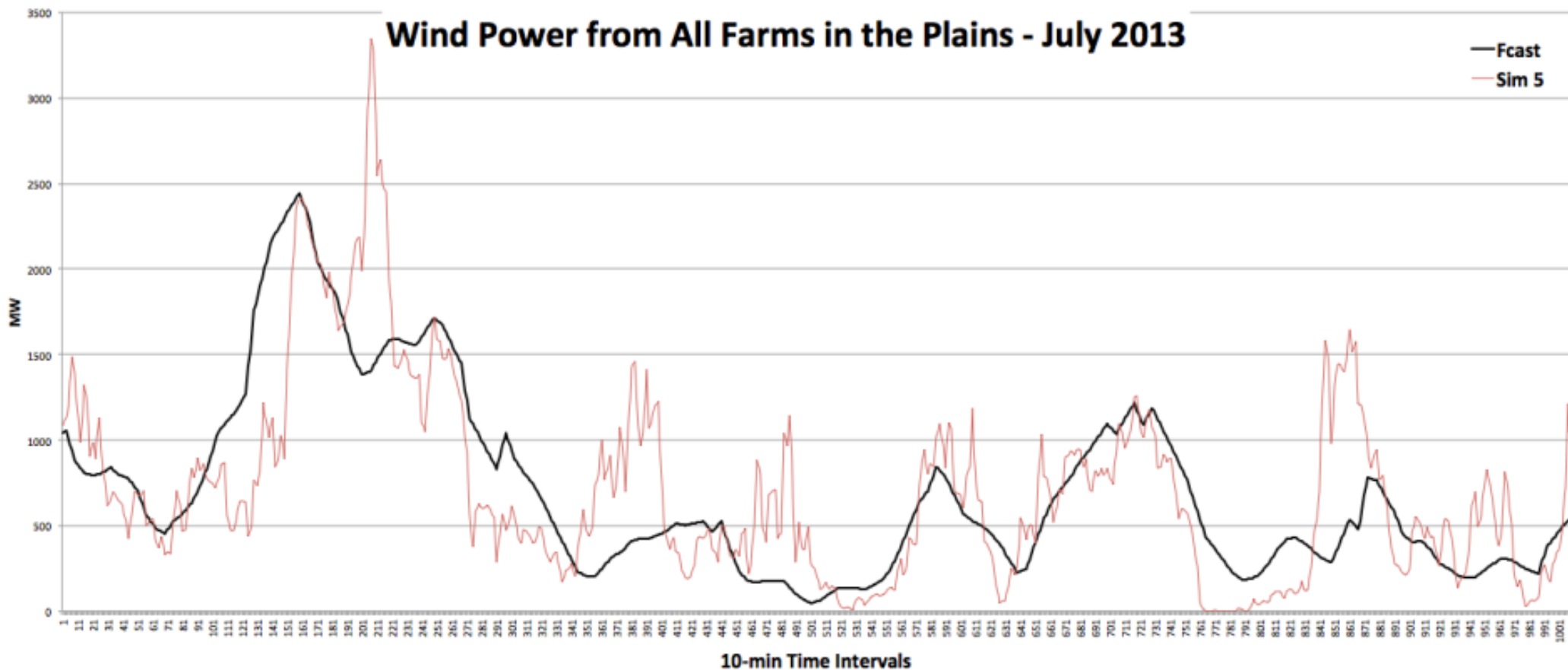
# Stochastic lookahead policies

- Creating wind scenarios (Scenario #4)



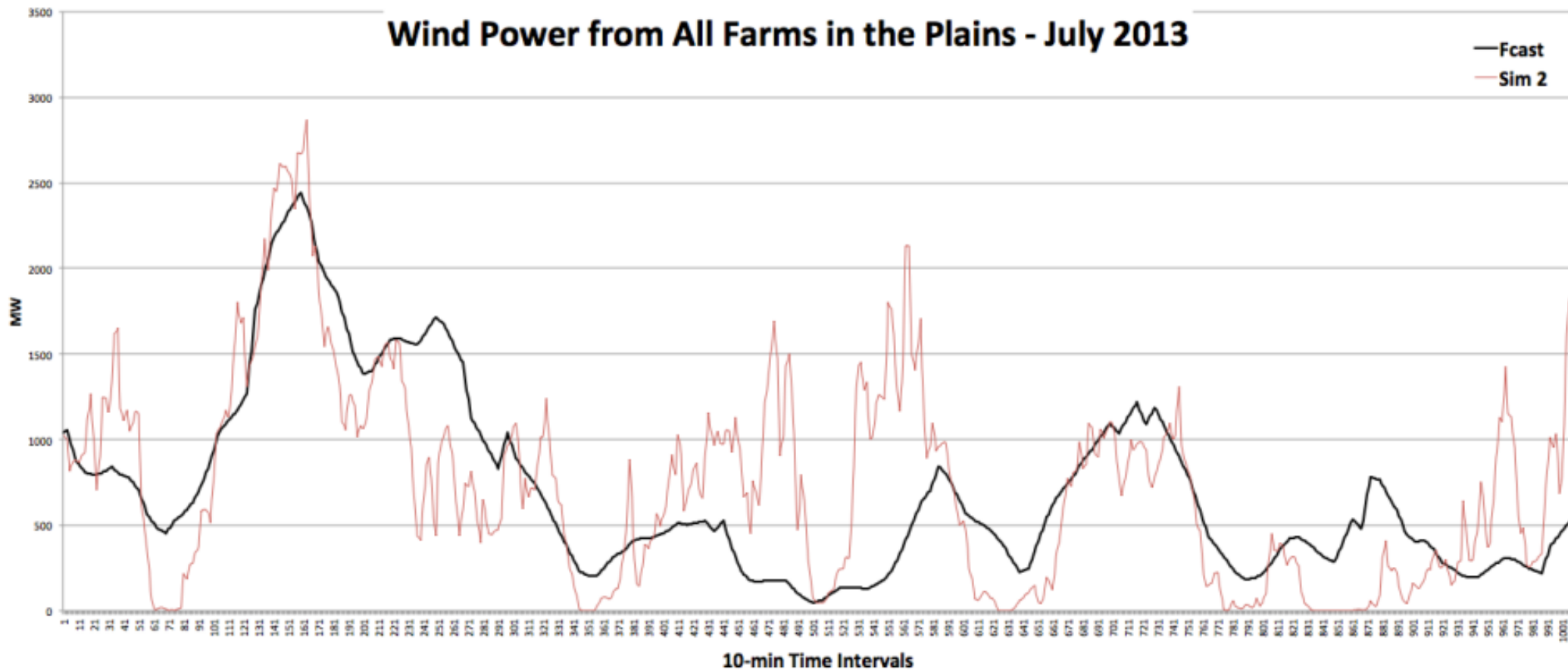
# Stochastic lookahead policies

- Creating wind scenarios (Scenario #5)



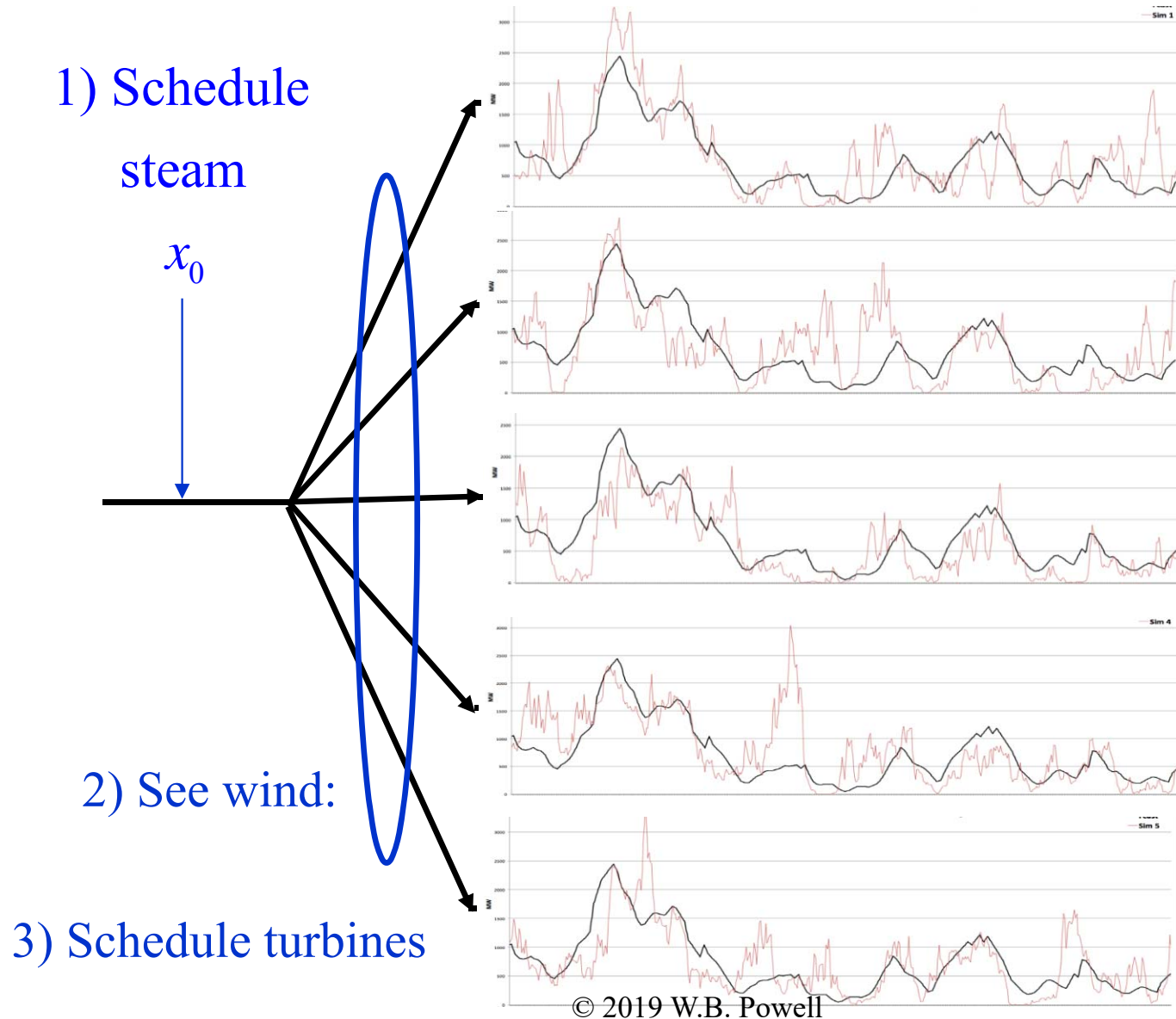
# Modeling stochastic wind

- Creating wind scenarios (Scenario #2)



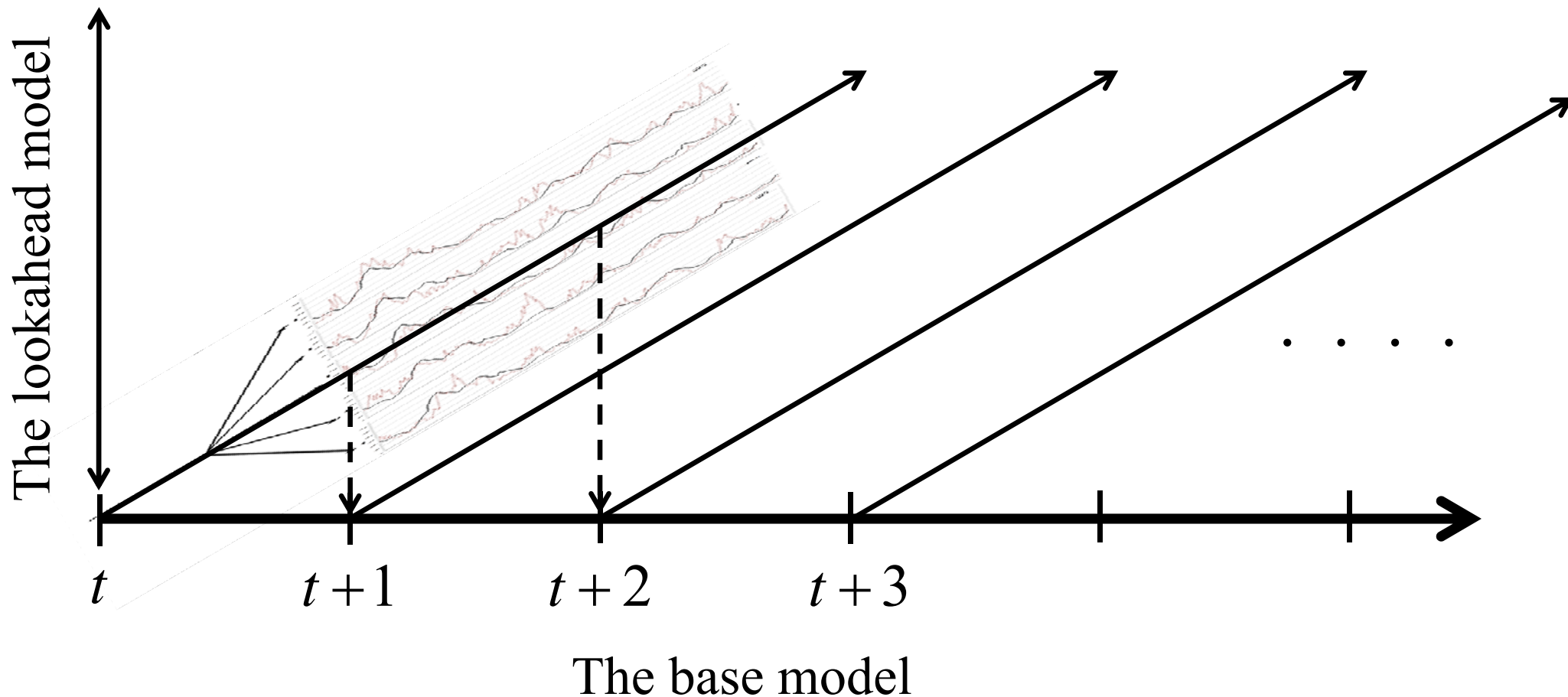
# Two-stage lookahead models

- ... to a two-stage approximation



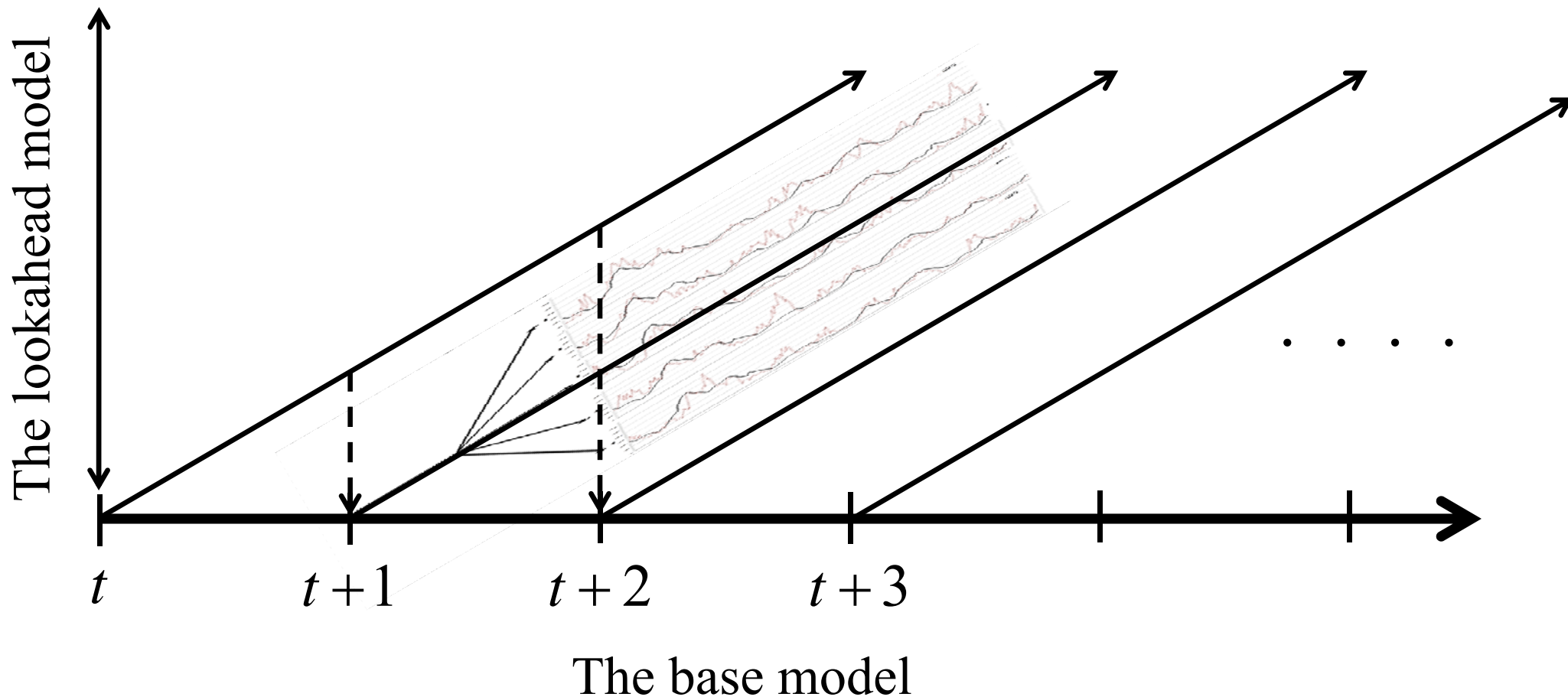
# Lookahead policies

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



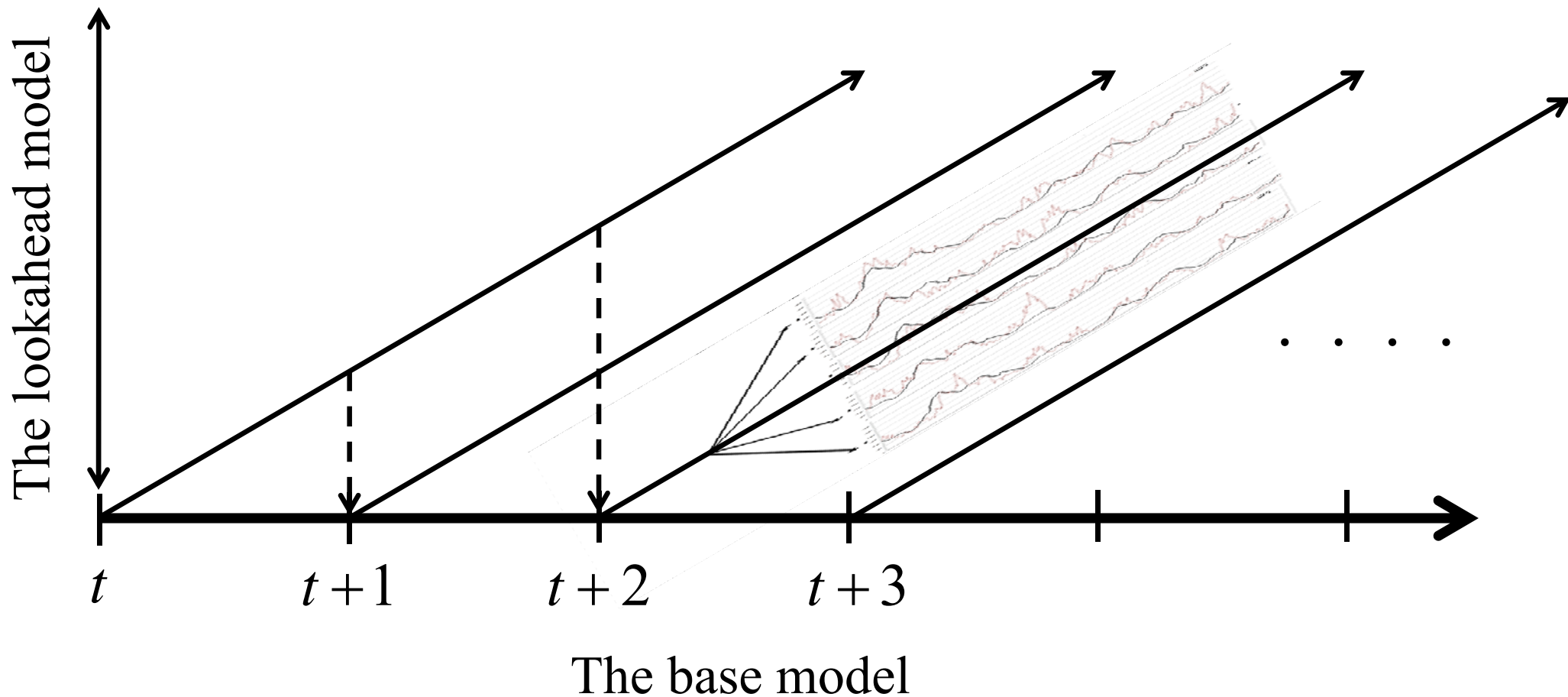
# Lookahead policies

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



# Lookahead policies

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





# Lookahead policies

---

## ● Notes:

- » Two-stage approximations of the unit commitment problem have been pursued by a number of national labs in the U.S.
  - Sandia, Argonne, Livermore, National Renewable Energy Laboratory
- » Deterministic unit commitment problems are hard; two-stage stochastic models, even with a small number of scenarios, are *much* harder.
- » The two-stage approximation introduces serious errors. Our biggest problem is not the uncertainty in wind forecasting tomorrow, it is forecasting wind an hour from now. This is ignored in a two-stage model.
- » The stochastic programming community does not understand the concept that the two-stage lookahead is a *policy*, to solve a base model (which at best they think of as a simulator). Relatively few teams build simulators to test two-stage policies.

# Direct lookahead

Case study: Stochastic unit commitment

# Stochastic unit commitment

● March 18, 2019

Use/benefits of stochastic unit commitment ➤ Inbox x



**Ben Hobbs** via [princetonu.onmicrosoft.com](mailto:princetonu.onmicrosoft.com)  
to Warren, Venkat, Elina ▾

Mon, Mar 18, 3:27 PM (2 days ago)



Hi Warren,

My colleague Venkat Krishnan of NREL is on a panel tomorrow at a meeting about solar forecasting, and he anticipates being asked "why don't ISOs do stochastic dispatch and commitment?"

I know you'll have an opinion about using explicitly stochastic methods (e.g., 2 or more stage with scenarios, an expected cost objective, and redispatch/commitment by scenario). Do you think it is something that some ISO will seriously consider in the near future? Are you testing it on the PJM or other large systems?

many thanks!



# Stochastic unit commitment

🟡 March 18, 2019



Warren Powell <wbpowell328@gmail.com>

to Scott, Ben, Venkat, Elina ▾

📧 Mon, Mar 18, 4:09 PM (2 days ago)



Ben,

I am cc'ing in Scott Baker from PJM. I just gave a talk on this issue a few weeks ago.

1) I do not think a stochastic lookahead with scenario trees will ever be useful. Even without CPU time issues (which are severe), the use of scenario trees to obtain a robust solution is deeply flawed. I have some powerpoint slides that report on a specific instance where we "fixed" a problem by using a scenario describing a drop in the wind that produced an outage. The fix was to increase steam in the day-ahead model over a specific 30 minute time period. Since we would never know *when* an outage like this would happen, we would need 48 scenarios to recreate the outage at each time of day. Further, for PJM, this would have to be done for *each* of about seven zones.

2) What the ISOs are doing is using something that I have named a "parametric cost function approximation." In plain English, the ISOs use a deterministic lookahead, with parameters inserted to produce the reserves needed in different regions of the network. This is *far* more powerful than scenario trees. It is amazing that industry is already doing something that is better than what the academic community (and research labs) have been pushing.

Attached is a paper (and powerpoint presentation) where I illustrate the parametric cost function approximation. You can see it most easily in the powerpoint slide, where we show the deterministic lookahead. I then put a red box around a constraint that uses a forecast, and the next slide I show how this constraint can be parameterized (in this simple example, we just multiple the forecast by a tunable parameter that depends on how many time periods you are forecasting into the future).

A parametric CFA is much simpler and more effective than scenario trees, which are highly simplified approximations of the stochastic lookahead (not just the use of a small sample - the whole two-stage assumption is a much bigger approximation than this community recognizes). The hard part of a parametric CFA is the tuning, and PJM knows this. They have an elaborate process called "perfect dispatch" that helps them, but I think that there is a lot of domain knowledge in the tuning.

# Stochastic unit commitment

---

- When solving the unit commitment problem, people often ask:
  - » Should we use a deterministic model?
    - ... which is what we are doing now.
  - » Or should we adopt a stochastic model?

***This is the wrong question!***

# Stochastic unit commitment

---

## ● Observations:

- » The real world is stochastic, always has been.
- » Grid operators already have to manage different types of uncertainty:
  - Will a generator fail?
  - Will a transmission line fail?
  - What will the temperature be?
  - How much will people increase air conditioning if the temperature or humidity increases unexpectedly?
- » Grid operators have been dealing with uncertainty all the time, using different strategies to account for uncertainty.

# Stochastic unit commitment

- There are two models we need to recognize:
  - » The “base model” – This might be:
    - A computer simulator, or
    - The real world.
  - » The “lookahead model”
    - This is the model we solve when we are peeking into the future.
  - » Notes:
    - The base model is *always* stochastic.
    - Lookahead models are usually designed to handle the uncertainty in the base model/real world. Strategies:
      - Stochastic lookahead:
        - » Scenario trees
        - » Stochastic dynamic programming (e.g. SDDP)
      - Parametric cost function approximation
        - » Parametrically modified deterministic lookahead

# Stochastic unit commitment

---

- Strategies for handling uncertainties:
  - » We can build a “stochastic” lookahead model.
  - » We can build a robust “parametric cost function approximation” which is a modified deterministic model, designed to handle uncertainty.
    - These modifications are designed based on an intuitive understanding of how decisions should change to handle uncertainty.
    - This is what industry does, but without a formal model, or formal methodology for designing and setting these parameters.



# Stochastic unit commitment

---

- Notes:

- » Multistage scenario trees produce *extremely* large problems.
- » Benders cuts are likely to introduce significant errors.

- Alternative:

- » Use two-stage approximation.

# Stochastic unit commitment

---

## ● Notes:

- » Standard practice for the “stochastic unit commitment problem” (in the U.S.) is to use this “two-stage” formulation:
  - Make steam generation decisions today
  - Model the entire sample path of what might happen tomorrow
  - Optimize gas generators tomorrow as if you can see the entire future.
  
- » Problem – The big uncertainty is not the day-ahead forecast, but the hour-ahead forecast.
  - If we get the day-ahead forecast wrong, we can use gas turbines to handle the error.
  - If we get the hour-ahead forecast wrong, all we can do is to ramp generators that are already on.

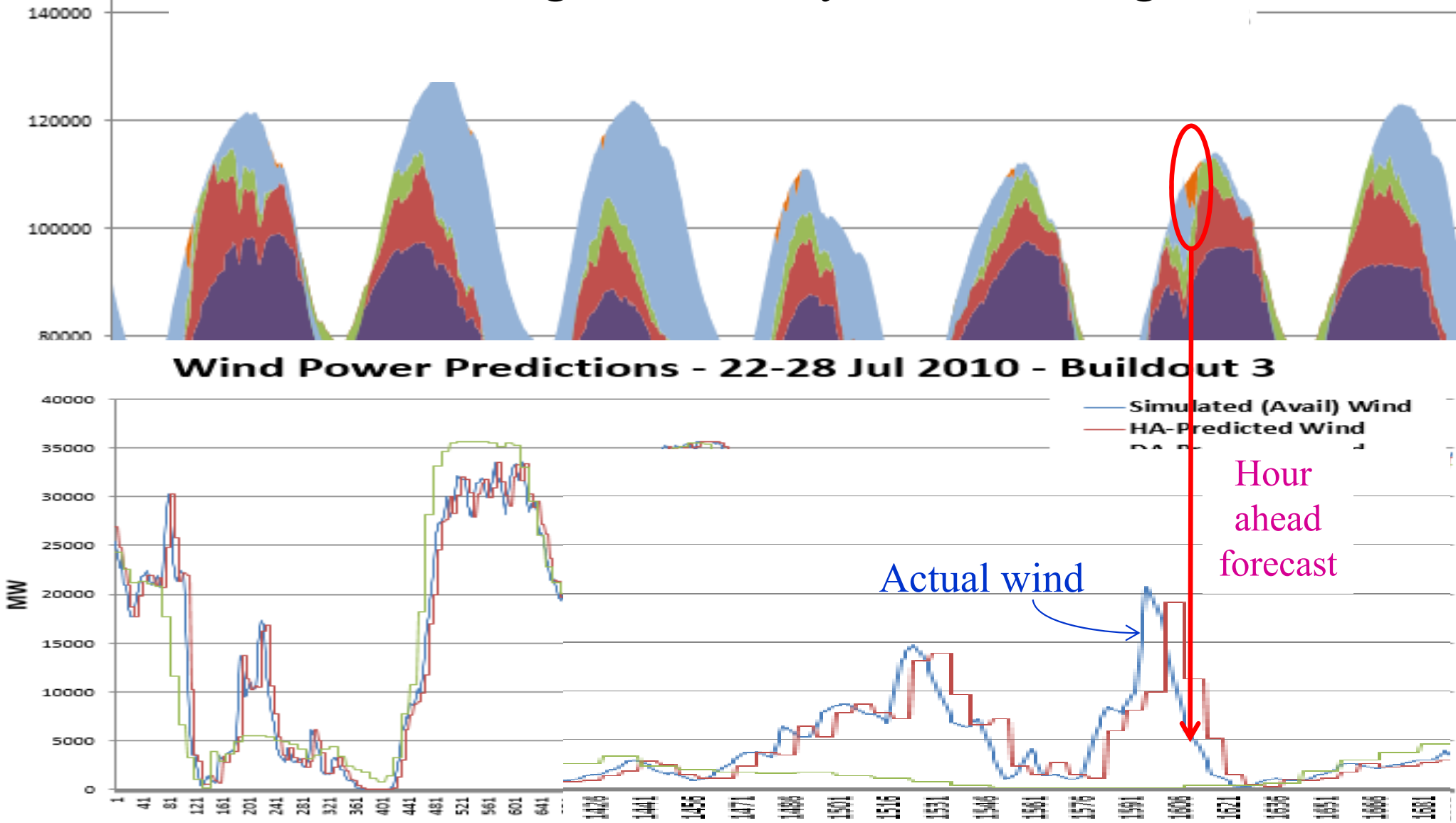
# Stochastic unit commitment

## ● Notes:

- » In the next slides, we are going to demonstrate the problems when the energy from wind dips quickly.
- » A problem is that short-term forecasting is hard. Meteorologists do not know the state of the atmosphere right now, so forecasting an hour from now is hard.
- » Industry standard practice for short-term forecasting is to use *persistence forecasting*, which means they assume that  $W_{t+1} = W_t$ .
- » If we know that the wind is going to dip, we can turn on extra gas turbines now (these are planned an hour in the future).

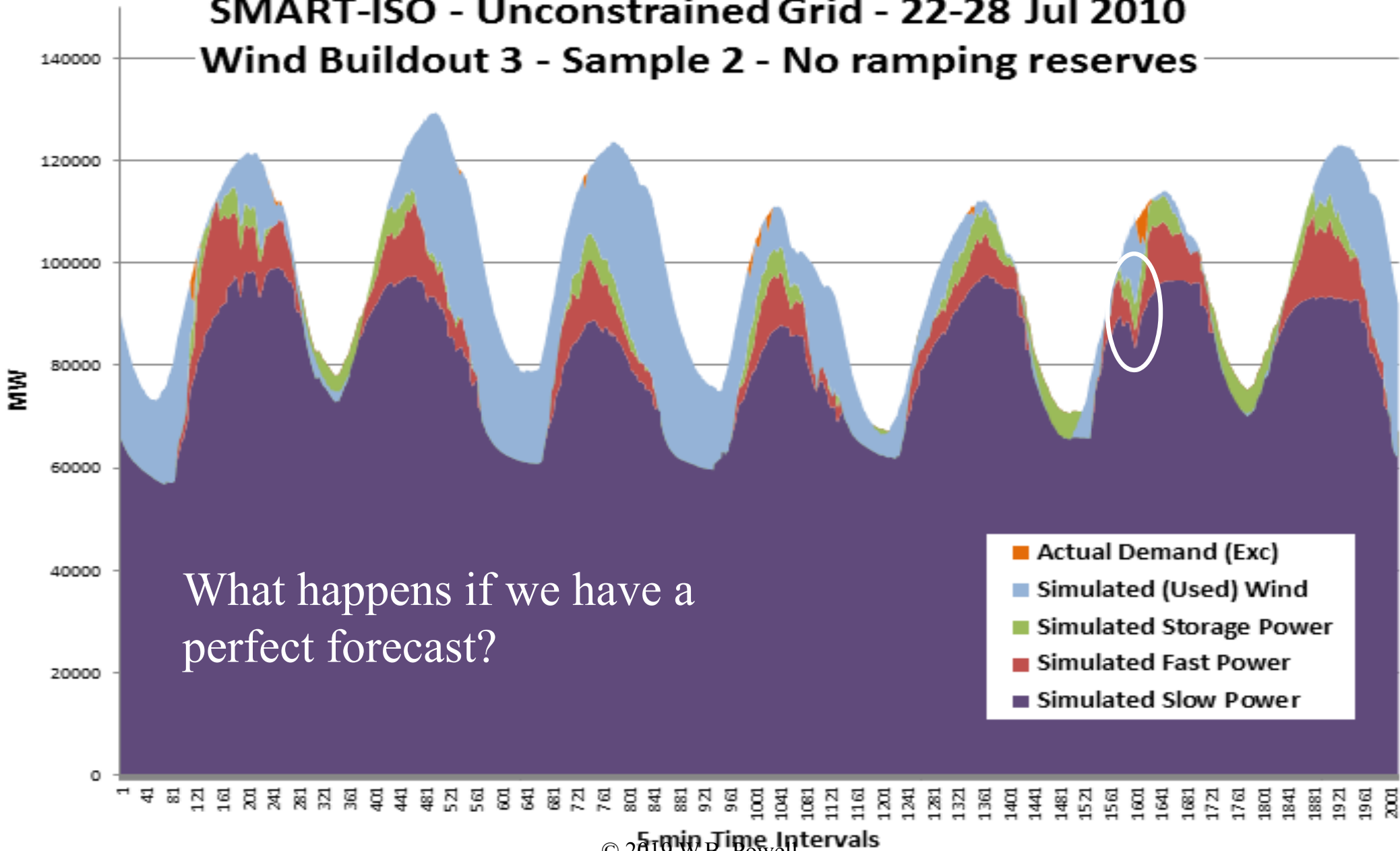
# Stochastic unit commitment

How forecasting uncertainty causes outages.



# Stochastic unit commitment

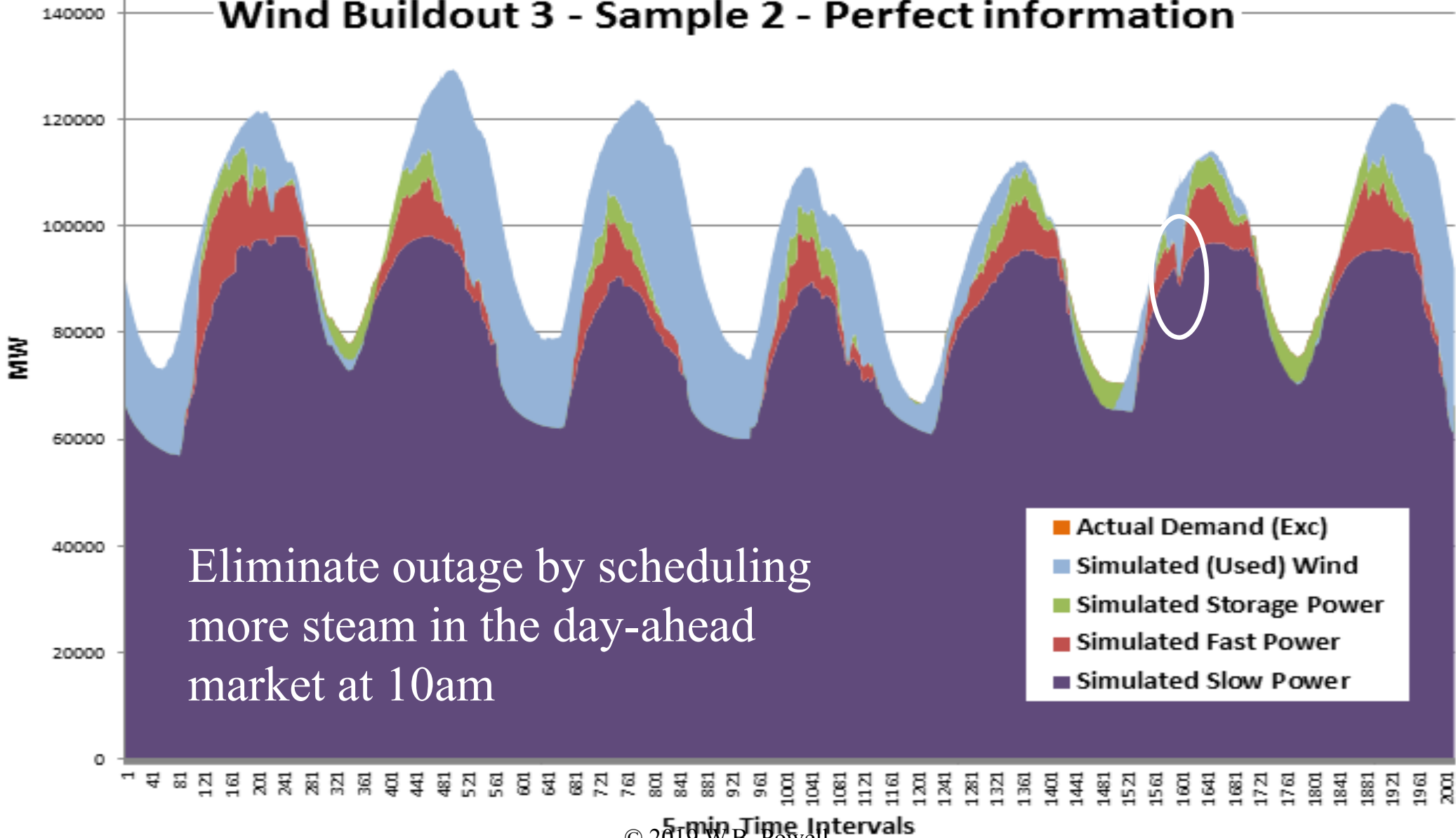
**SMART-ISO - Unconstrained Grid - 22-28 Jul 2010**  
**Wind Buildout 3 - Sample 2 - No ramping reserves**



What happens if we have a perfect forecast?

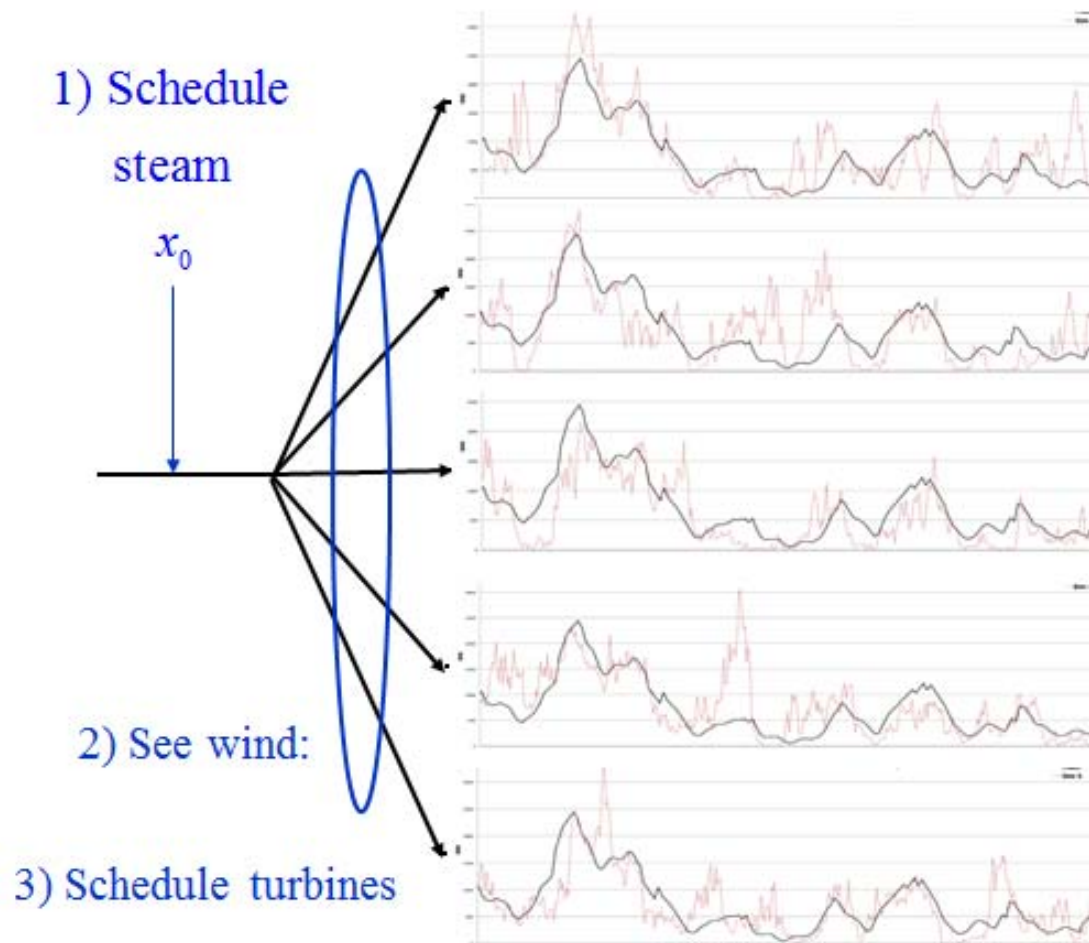
# Stochastic unit commitment

**SMART-ISO - Unconstrained Grid - 22-28 Jul 2010**  
**Wind Buildout 3 - Sample 2 - Perfect information**



# Stochastic unit commitment

- Using random samples, we will not reflect the reality that these sudden drops can happen any time.

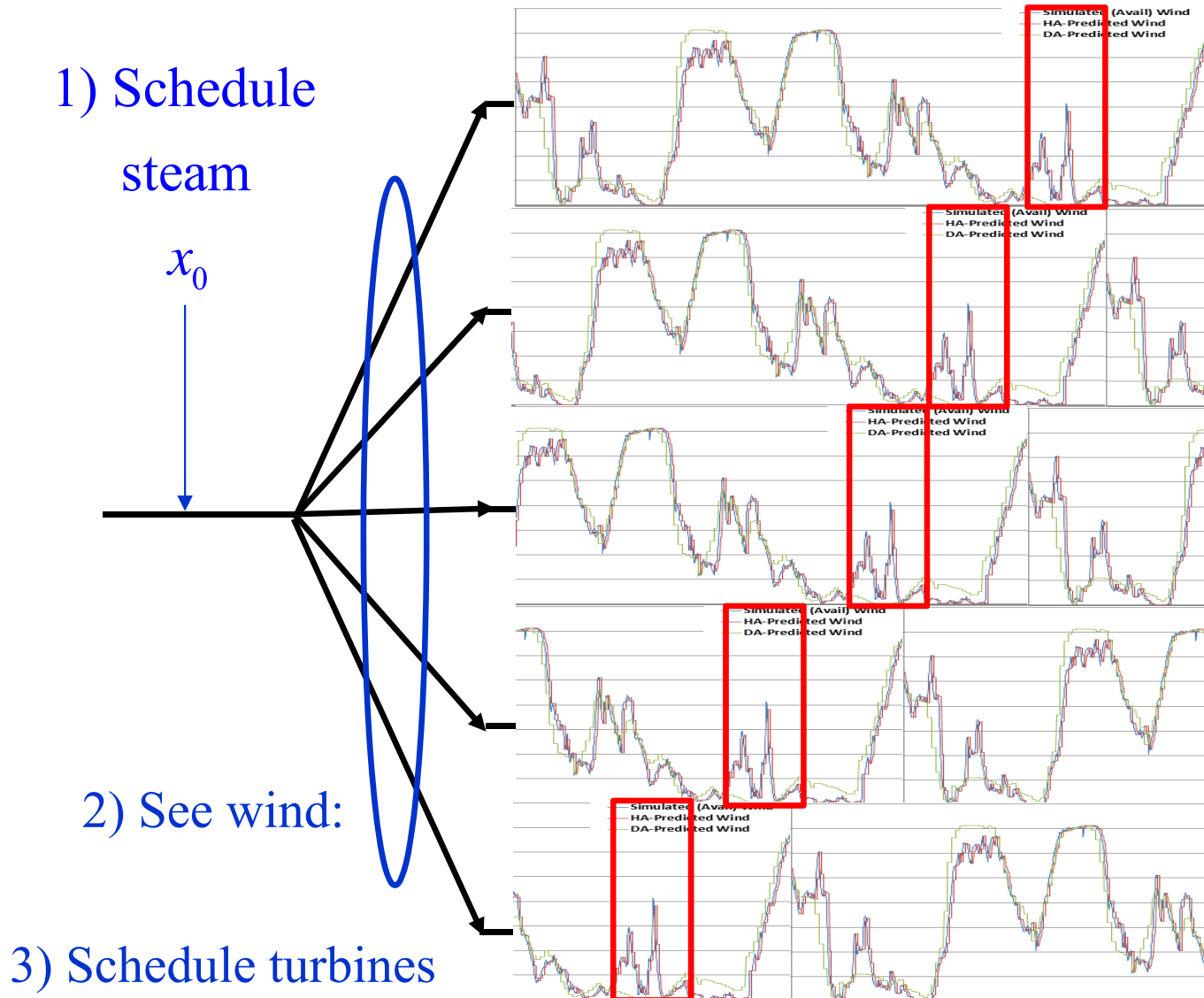


But if we just use a random sample, it is very likely that we will not simulate these drops for many times of the day.

This means that if these drops happen at these times of day, we are not protected.

# Stochastic unit commitment

- We can construct drops at every time of day...



We can construct a series of scenarios so that these extreme drops happen at every time of day.

This will make sure that we schedule reserve at every time of day.



# Stochastic unit commitment

---

## ● Notes:

- » We can construct a set of scenarios that reflects that sudden drops can happen at any time of day.
- » We will need a *lot* of scenarios to communicate that these drops where we need extra reserve can happen at any time....
- » ... or we could simply require that we schedule reserves for all times of the day!

# Stochastic unit commitment

- A deterministic lookahead model
  - » We use a point forecast of the future

$$\min_{\substack{(x_{tt'})_{t'=1,\dots,24} \\ (y_{tt'})_{t'=1,\dots,24}}} \sum_{t'=t}^{t+48} C(x_{tt'}, y_{tt'})$$

Steam generation

Gas turbines

- » These decisions need to be made with different horizons
  - Steam generation is made day-ahead
  - Gas turbines can be planned an hour ahead or less

# Stochastic unit commitment

- A deterministic lookahead policy

$$X^\pi(S_t) = \arg \min_{\substack{(x_{tt'})_{t'=1, \dots, 24} \\ (y_{tt'})_{t'=1, \dots, 24}}} \sum_{t'=t}^{t+48} C(x_{tt'}, y_{tt'})$$

- » No ISO would ever use a policy like this – it is too vulnerable to variability.

# Stochastic unit commitment

- A robust cost function approximation
  - » We add in up and down fast ramping reserves

$$X^\pi(S_t, \theta) = \arg \min_{\substack{(x_{tt'})_{t'=1, \dots, 24} \\ (y_{tt'})_{t'=1, \dots, 24}}} \sum_{t'=t}^{t+48} C(x_{tt'}, y_{tt'})$$

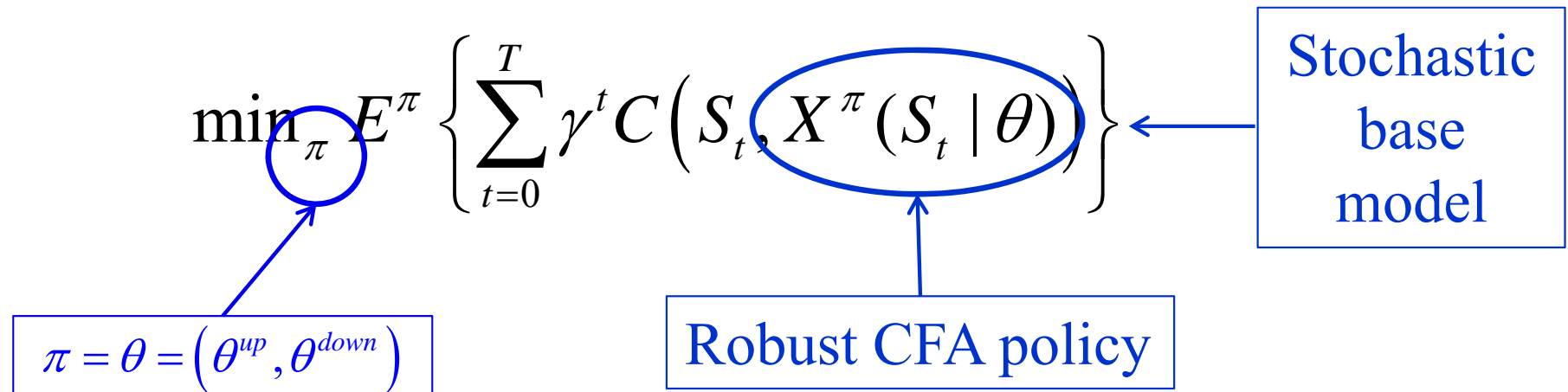
$$x_{t,t'}^{\max} - x_{t,t'} \geq L_{tt'} + \theta^{up} \quad \text{Up-ramping reserve}$$

$$x_{t,t'} - x_{t,t'}^{\max} \geq L_{tt'} + \theta^{down} \quad \text{Down-ramping reserve}$$

- » This is a (parametric) cost function approximation, parameterized by the ramping parameters  $\theta$ .
- » Now we can be sure that we will have ramping reserves at *every* time of day.

# Stochastic unit commitment

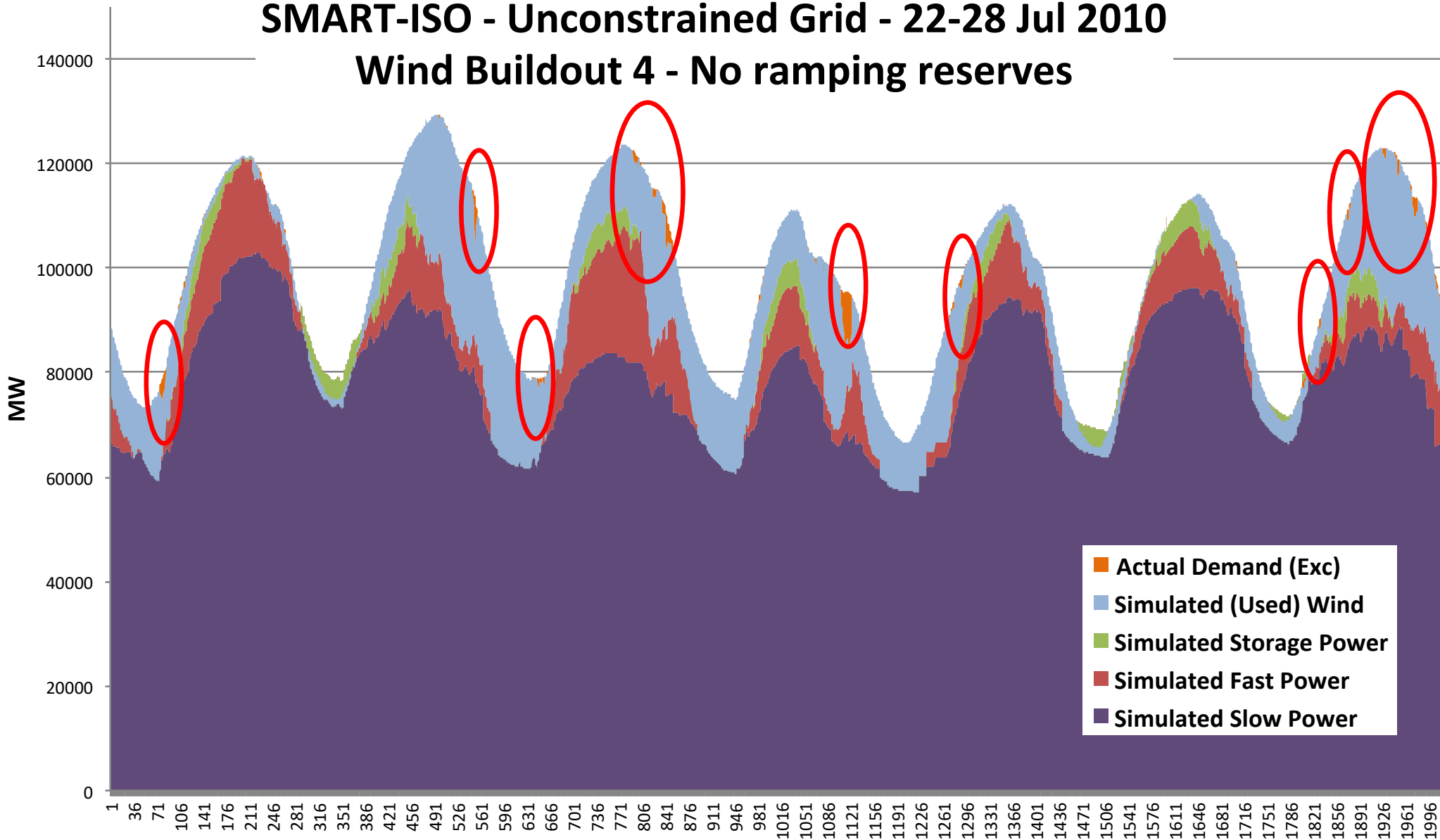
- We then have to tune the parameters of this policy in our *stochastic base model*.



- » The challenge now is to adaptively estimate the ramping constraints  $\theta = (\theta^{up}, \theta^{down})$ .
- » This is policy search – it can be done in a simulator, or online using real-world observations.

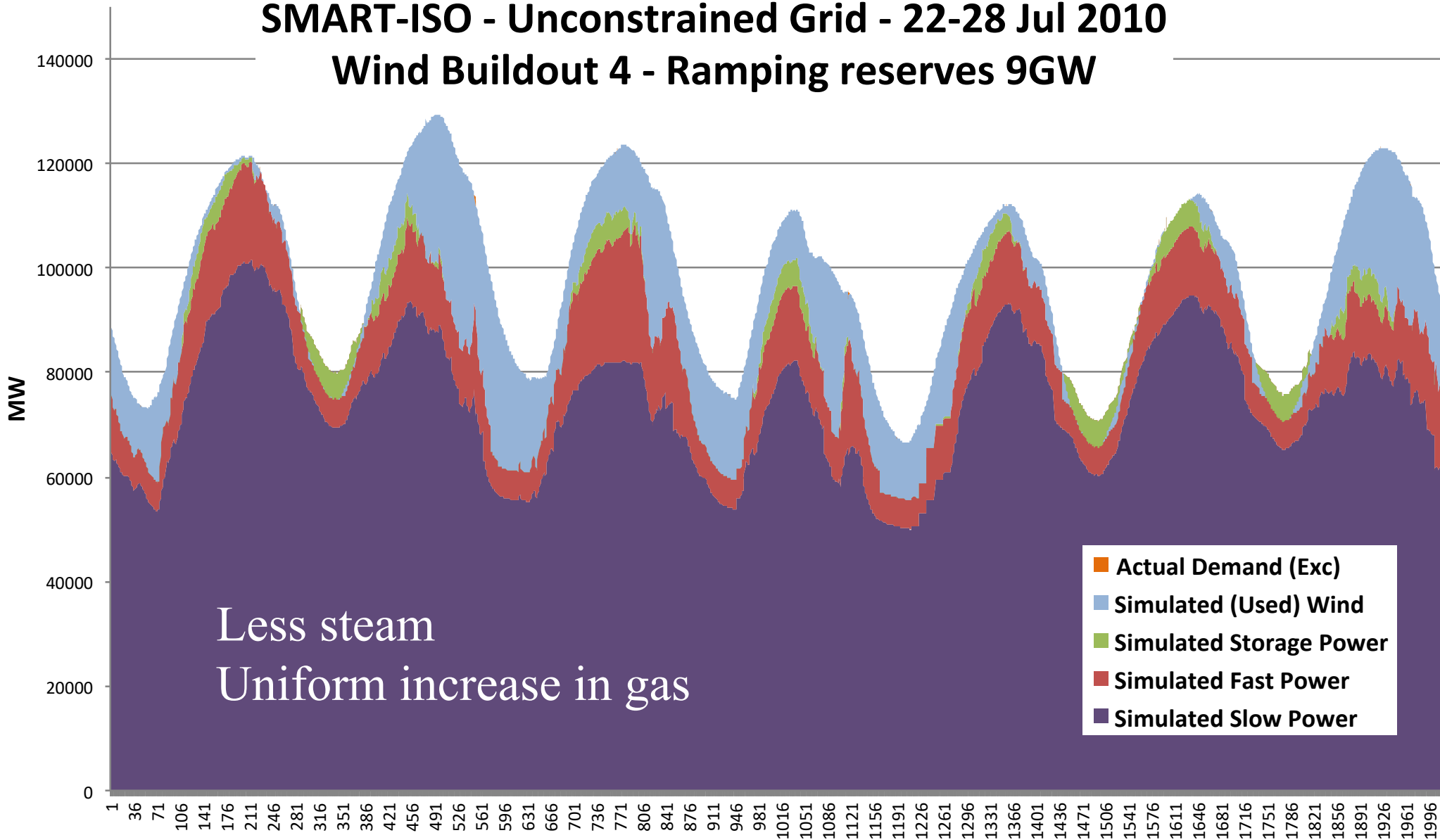
# Stochastic unit commitment

**SMART-ISO - Unconstrained Grid - 22-28 Jul 2010**  
**Wind Buildout 4 - No ramping reserves**



# Stochastic unit commitment

**SMART-ISO - Unconstrained Grid - 22-28 Jul 2010**  
**Wind Buildout 4 - Ramping reserves 9GW**



# Stochastic unit commitment

---

## ● Notes:

- » This is an example of a *parametric cost function approximation*.
- » We have modified our deterministic lookahead so we produce the behaviors that we need to make our solution robust.
- » We *know* that to handle uncertainty we need to schedule reserve...
- » ... so rather than solving a complex stochastic lookahead model, we modify our deterministic lookahead model to force it to schedule reserve.
- » Now we just have to figure out how much reserve to schedule!

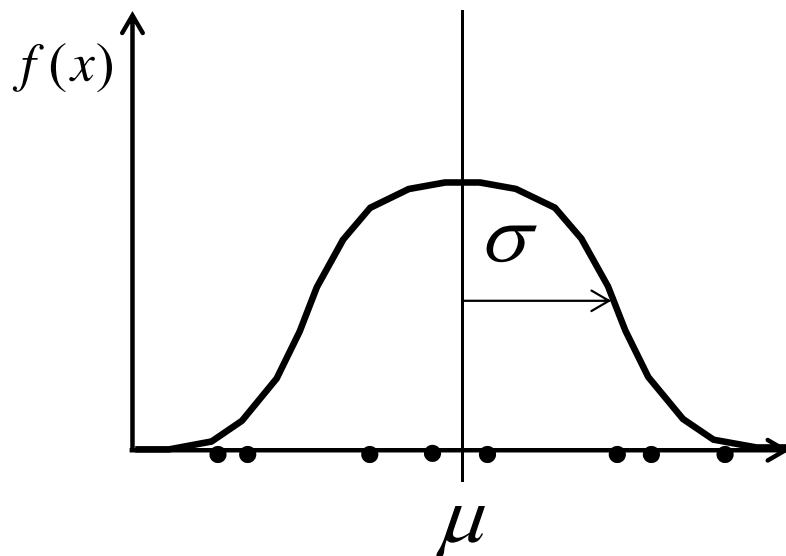


# Stochastic unit commitment

- Stochastic lookahead model

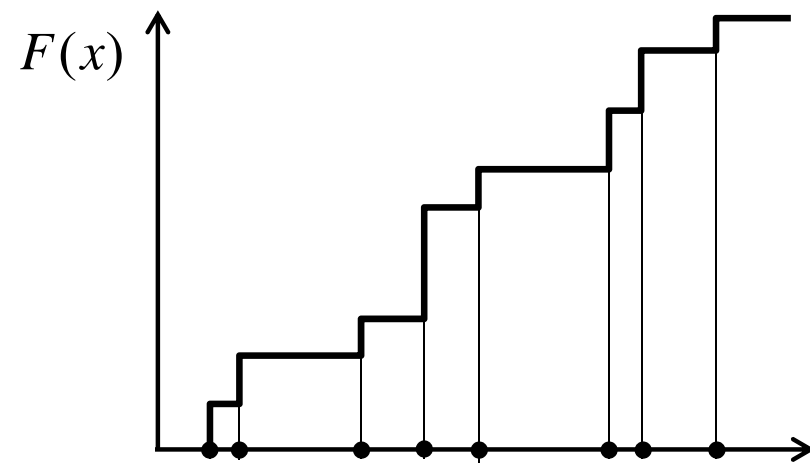
- » Uses approximation of the *information process* in the lookahead model

Parametric distribution (pdf)



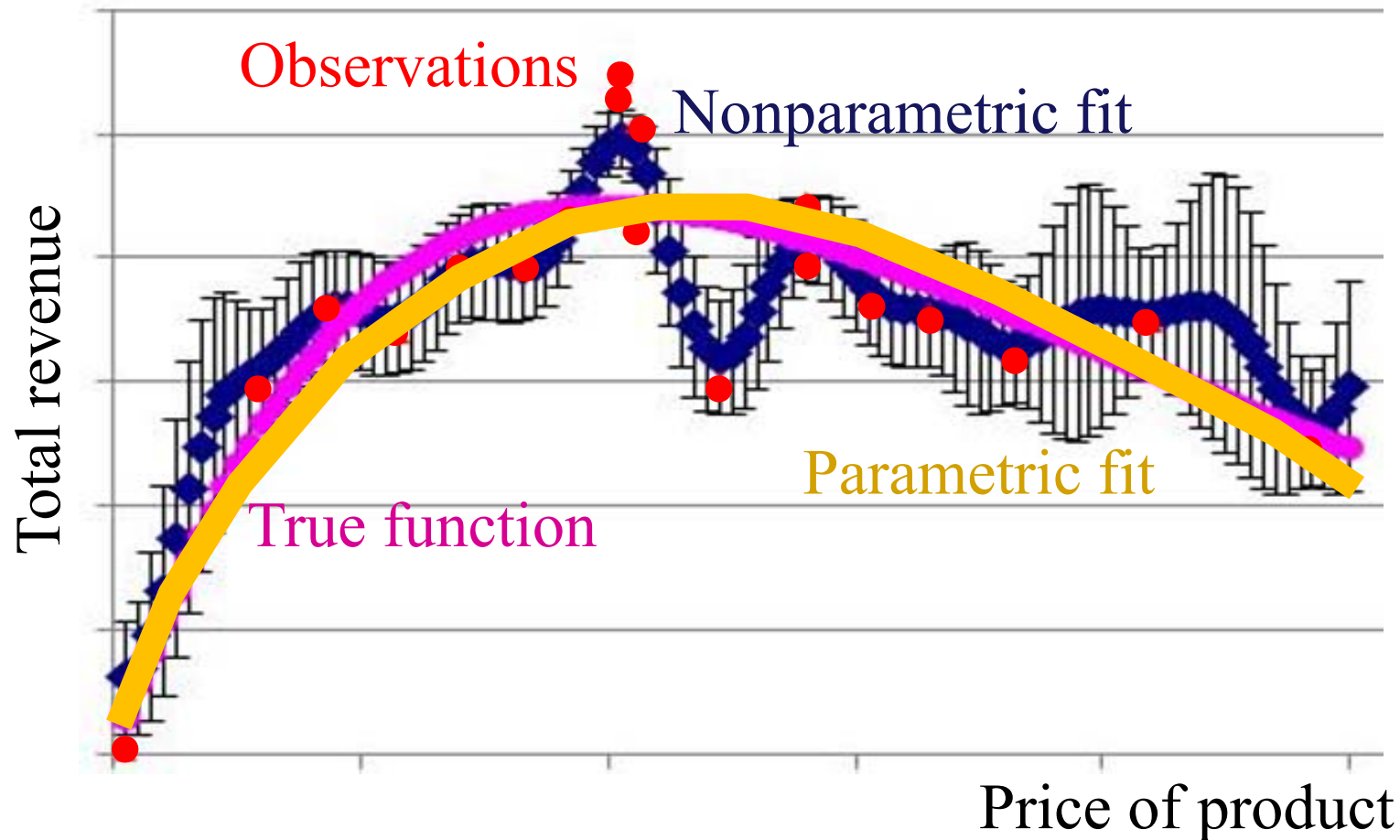
$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\left(\frac{(x-\mu)^2}{\sigma^2}\right)}$$

Nonparametric distribution (cdf)



# Stochastic unit commitment

- Parametric vs. nonparametric



- » Robust CFAs are *parametric*
- » Scenario trees are *nonparametric*

# Stochastic unit commitment

---

## ● Notes:

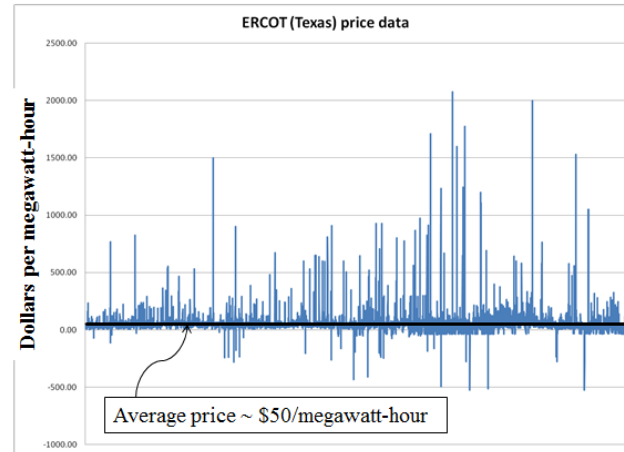
- » Our parametric cost function approximation is just like any parametric model.
- » It is able to capture structure, as long as we know the structure exists.
- » For example, we could see that we needed extra reserve whenever there was a sudden drop in the wind.
- » In a sampled model, we only scheduled reserve when drops happened, but in a large enough sample, this could happen at any time.
- » With a parametric model, we simply force reserves at all points in time. But this only works because we can identify the structure of a robust policy.

# Choosing the best policy

Some sample problems

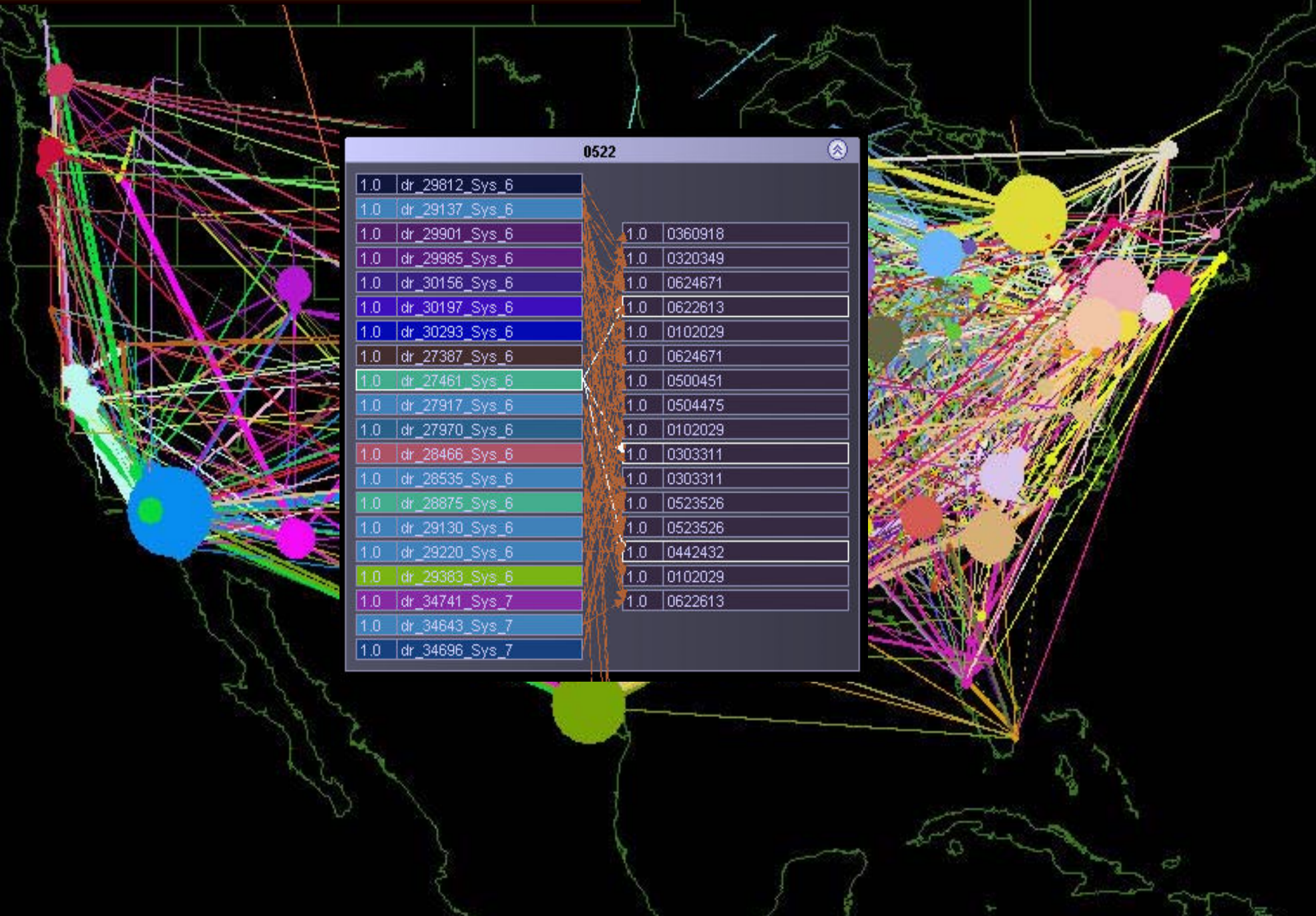
# Optimizing energy storage

- Take advantage of price variations



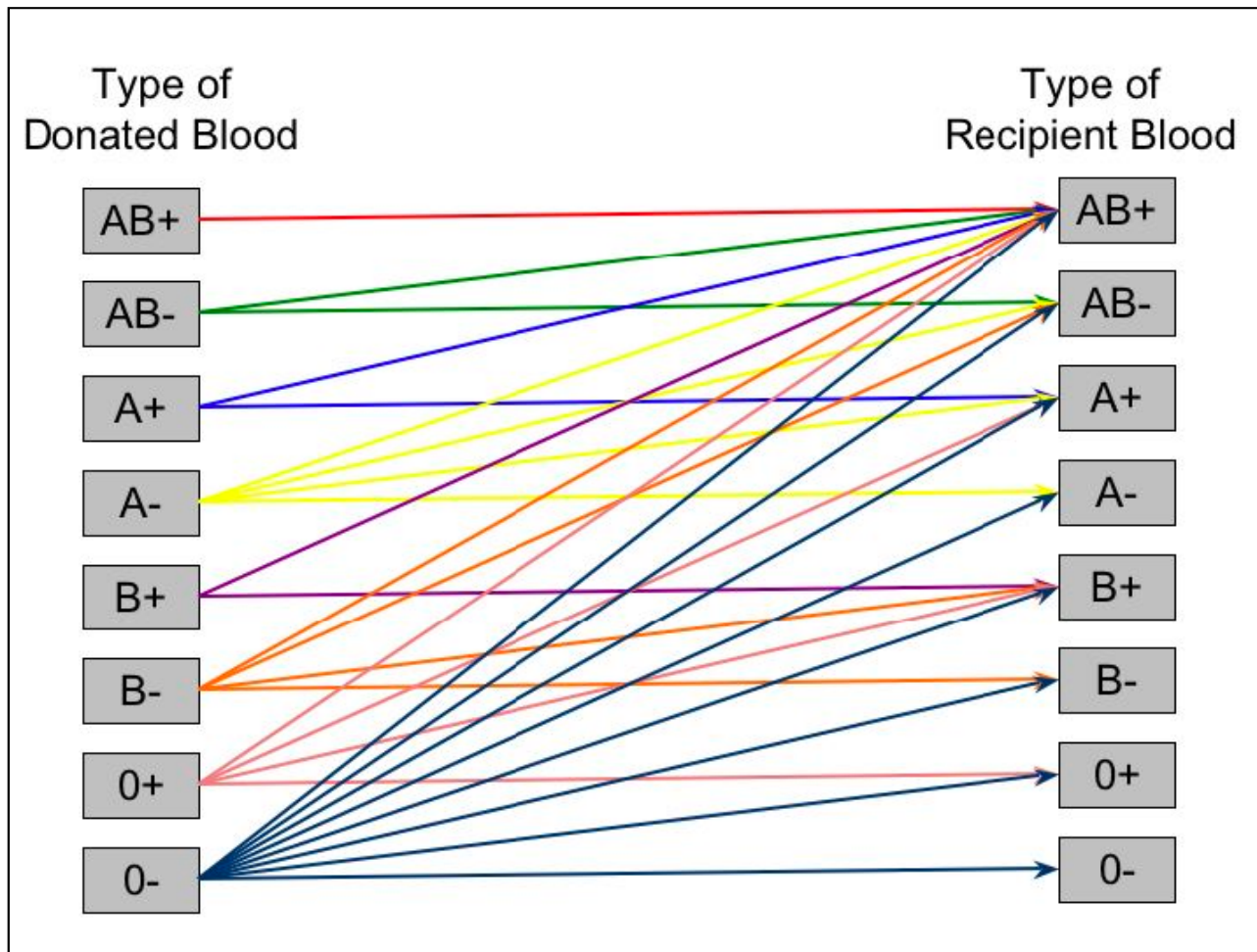


# Schneider National

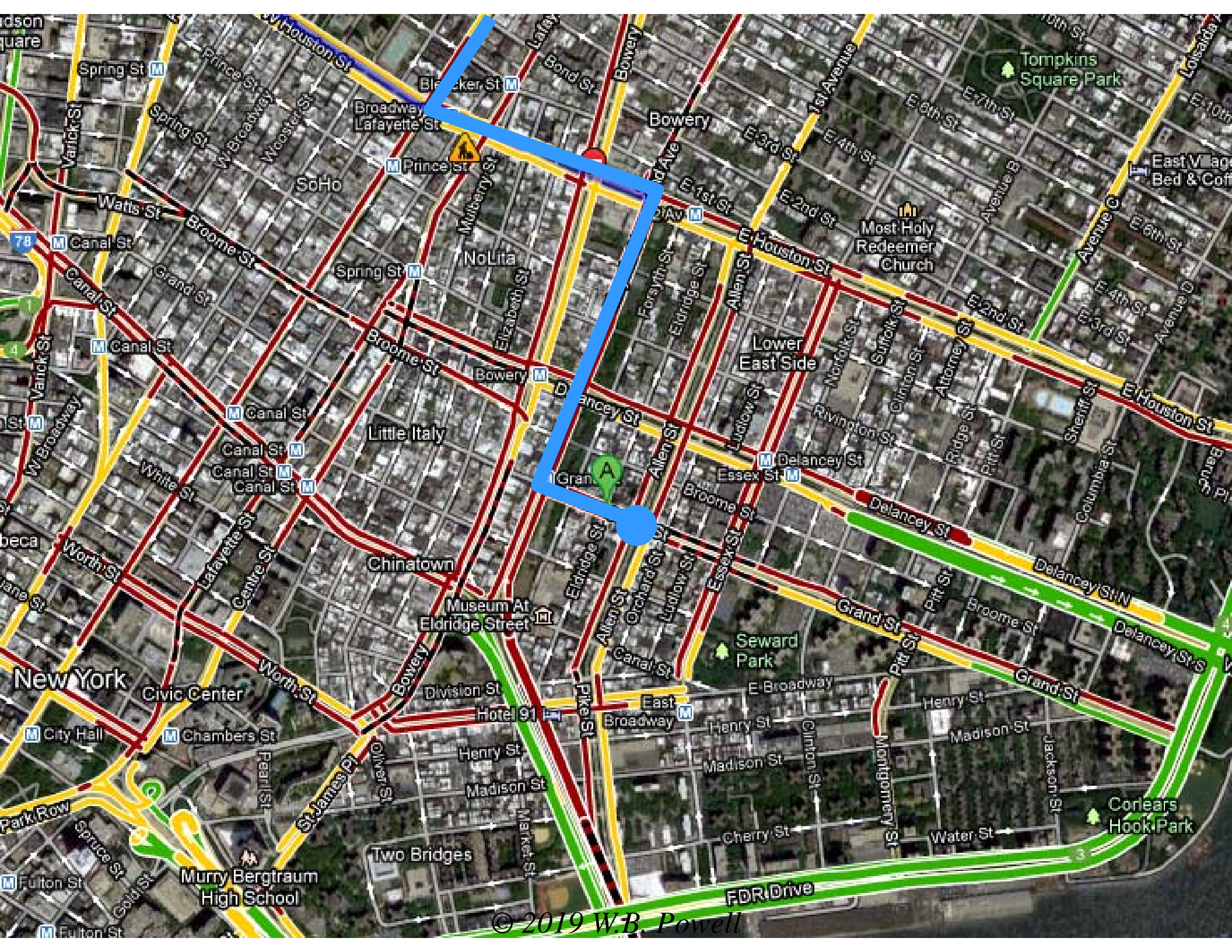


# Blood management

- Managing blood inventories

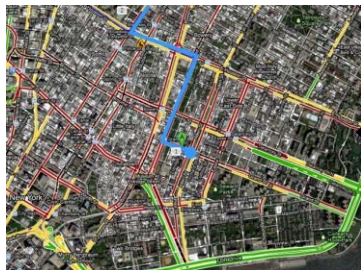








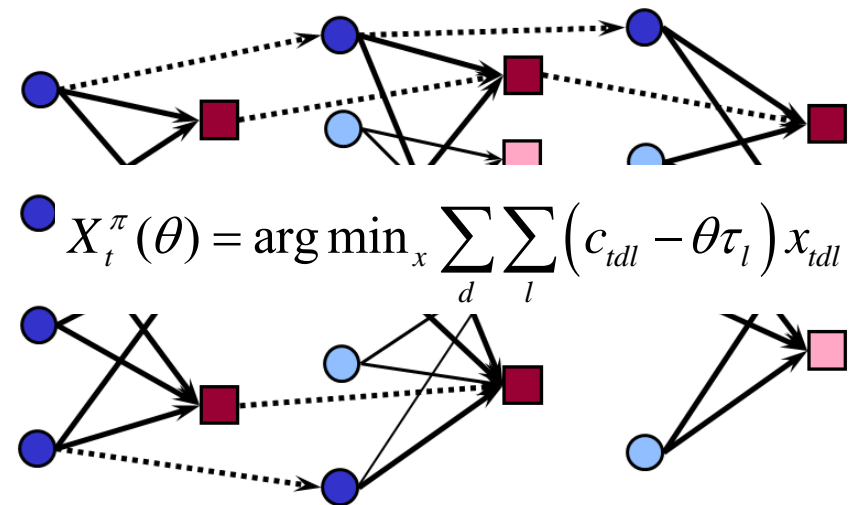
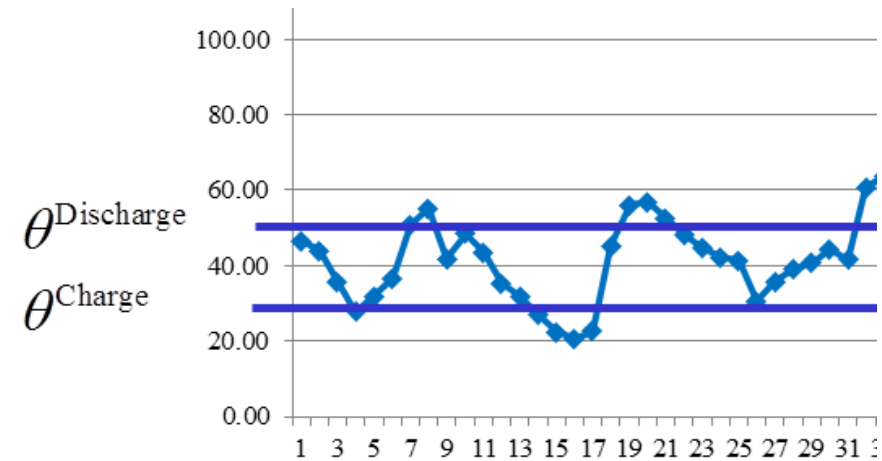
# Choosing a policy



- Robust cost function approximation
- Lookahead policy
- Policy function approximation
- Policy based on value function approximation

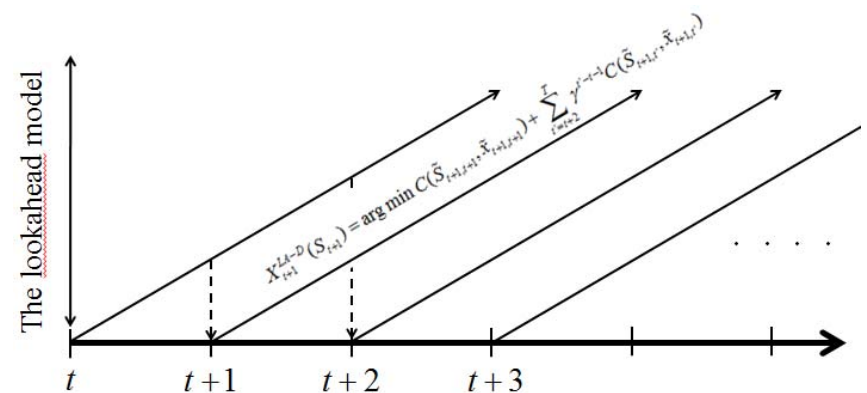
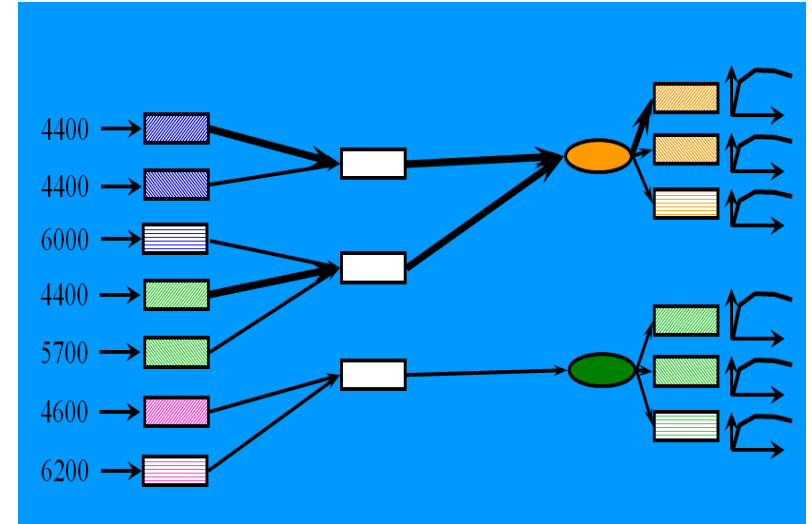
# Choosing a policy

- Which policy to use?
  - » PFAs are best for low-dimensional problems where the structure of the policy is apparent from the problem.
  - » CFAs work for high-dimensional problems, where we can get desired behavior by manipulating the cost function.



# Choosing a policy

- Which policy to use?
  - » VFAs work best when the lookahead model is easy to approximate
  - » Lookahead models should be used only when all else fails (which is often)



# Modeling sequential decision problems

## ● First build your model

- » Objective function

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

- » Policy

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

- » Constraints at time t

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

- » Transition function

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

- » Exogenous information

$$(W_1, W_2, \dots, W_T)$$

## ● Then design your policies:

- » PFA? Exploit obvious problem structure.

- » CFA? Can you tune a deterministic approximation to make it work better?

- » VFA? Can you approximate the value of being in a downstream state?

- » Lookahead? Do you have a forecast? What is the nature of the uncertainty?

- » Hybrid?



John R. Birge  
François Louveaux

# Introduction to Stochastic Programming

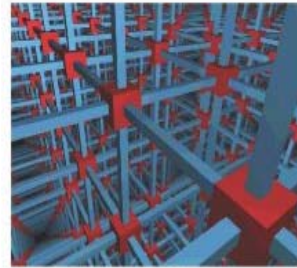
Second Edition

Michael C. Fu Editor

# Handbook of Simulation Optimization

Princeton Series in Applied Mathematics

# Robust Optimization



# Introduction to Decision Analysis

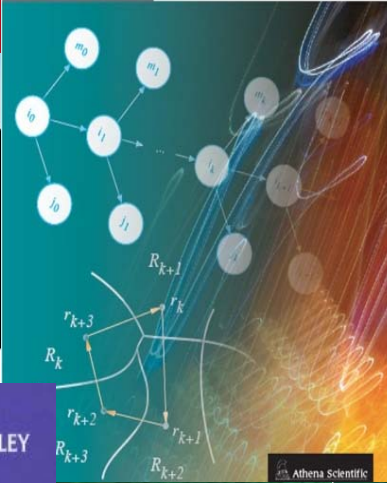
A Practitioner's Guide to Improving Decision Quality

VOLUME 2 • 4TH EDITION

# Dynamic Programming and Optimal Control

APPROXIMATE DYNAMIC PROGRAMMING

Dimitri P. Bertsekas



SECOND EDITION

# Approximate Dynamic Programming

Solving the Curses of Dimensionality

Warren B. Powell

Wiley Series in Probability and Statistics

# Optimal Learning

Springer

# MULTI-ARMED BANDIT ALLOCATION INDICES

SECOND EDITION

John Gittins, Kevin Glazebrook, and Richard V. D. Ryan

SECOND EDITION

# Model Predictive Control

# OPTIMAL CONTROL

Frank L. Lewis

# INTRODUCTION TO STOCHASTIC SEARCH AND OPTIMIZATION

Estimation, Simulation, and Control

JAMES C. SPALL

# Active Learning

Burr Settles

WILEY

Journal of Mathematical Modelling and Applied Probability

43

Jiongmin Yong  
Xun Yu Zhou

# Stochastic Controls

Hamiltonian Systems and HJB Equations

# Reinforcement Learning

An Introduction  
second edition

Richard S. Sutton and Andrew G. Barto

# Markov Decision Processes

Discrete Stochastic  
Dynamic Programming

MARTIN L. PUTERMAN

# Online Computation and Competitive Analysis

Yossi Azar, Avrim Blum, and Ran El-Yaniv

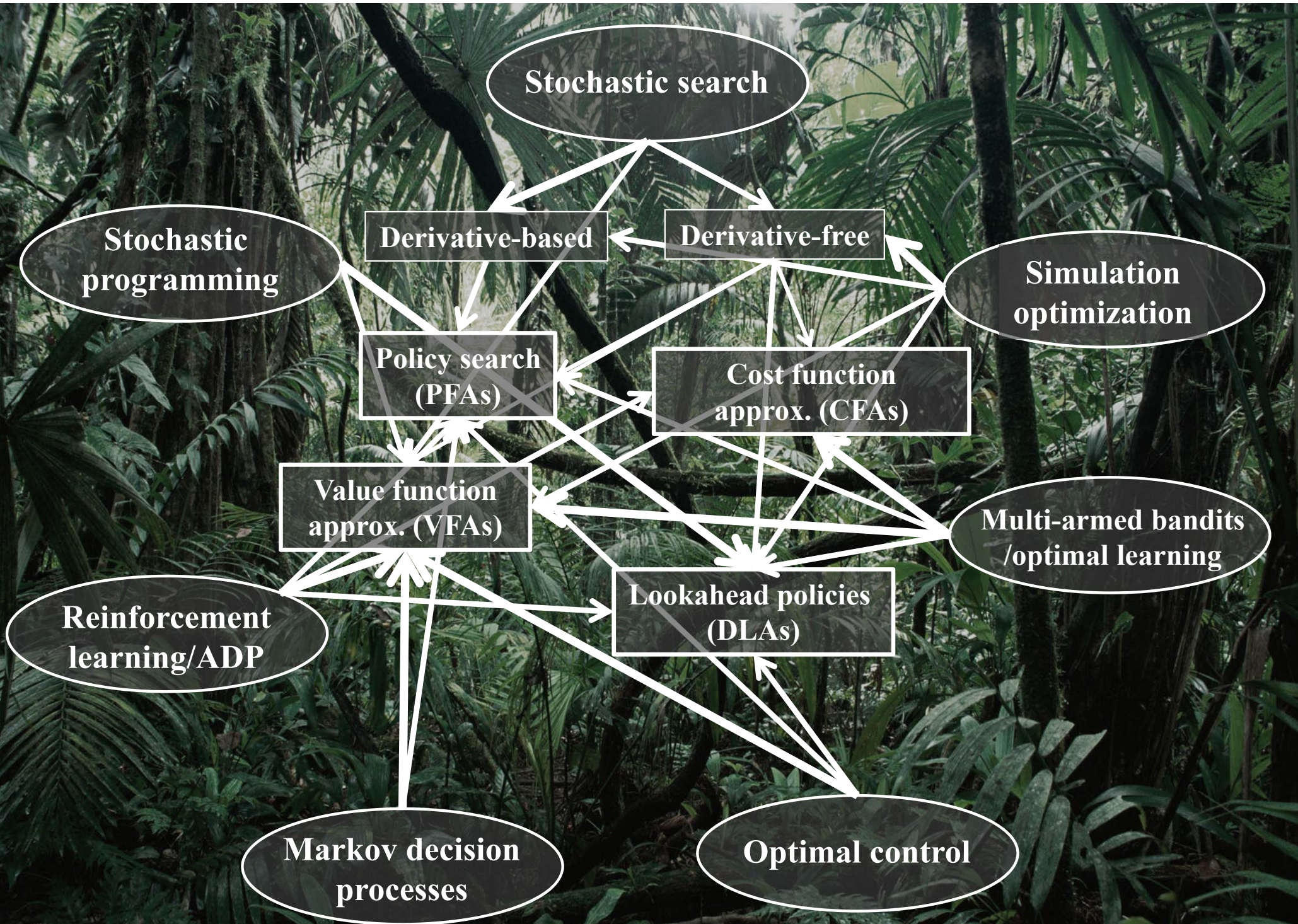
# STOCHASTIC SIMULATION OPTIMIZATION

An Optimal Computing Budget Allocation

Chun-Hung Chen • Loo Hay Lee









Theory

Computation

Modeling

Applications

