Crossing State Stochastic Models and Backward Approximate Dynamic Programming in Energy Storage Optimization

Speaker: Joe Durante (jdurante@princeton.edu)*
Advisor: Warren Powell (powell@princeton.edu)**
Additional Contributors: Juliana Nascimento, Harvey Cheng*, Raj Patel****

***Electrical Engineering, Princeton University**
****Operations Research and Financial Engineering, Princeton University**

June 27, 2017

CASTLE Laboratory
Princeton University
http://www.castlelab.princeton.edu

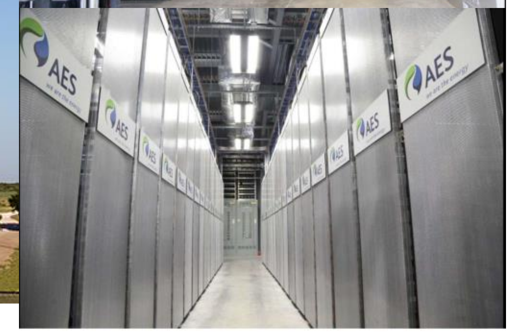
Outline

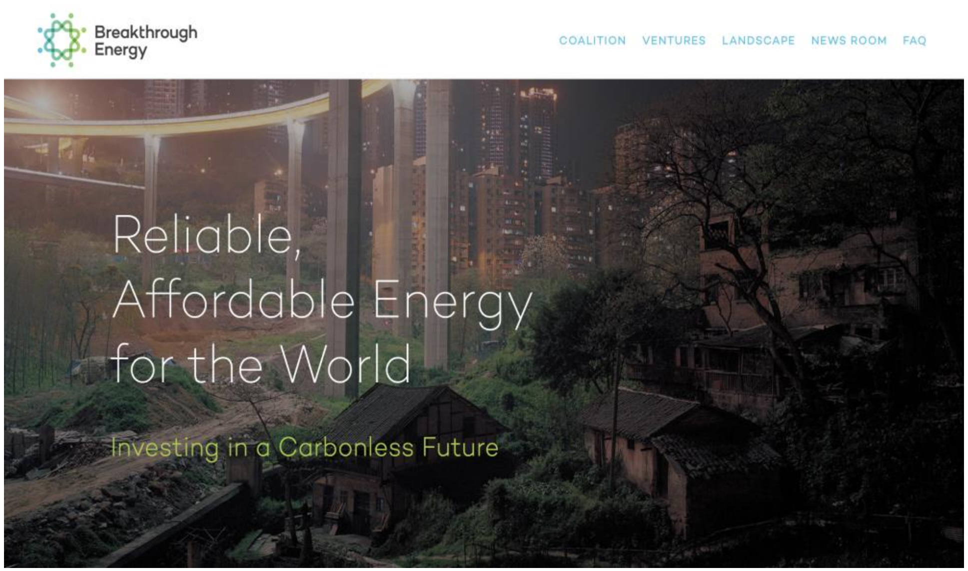
- Motivation
- Hidden Semi-Markov Crossing State Model
- Formulating the Energy Storage Problem as a Markov Decision Process
- Backward Approximate Dynamic Programming
- Numerical Results

Outline

- Motivation
- Hidden Semi-Markov Crossing State Model
- Formulating the Energy Storage Problem as a Markov Decision Process
- Backward Approximate Dynamic Programming
- Numerical Results

Motivation:

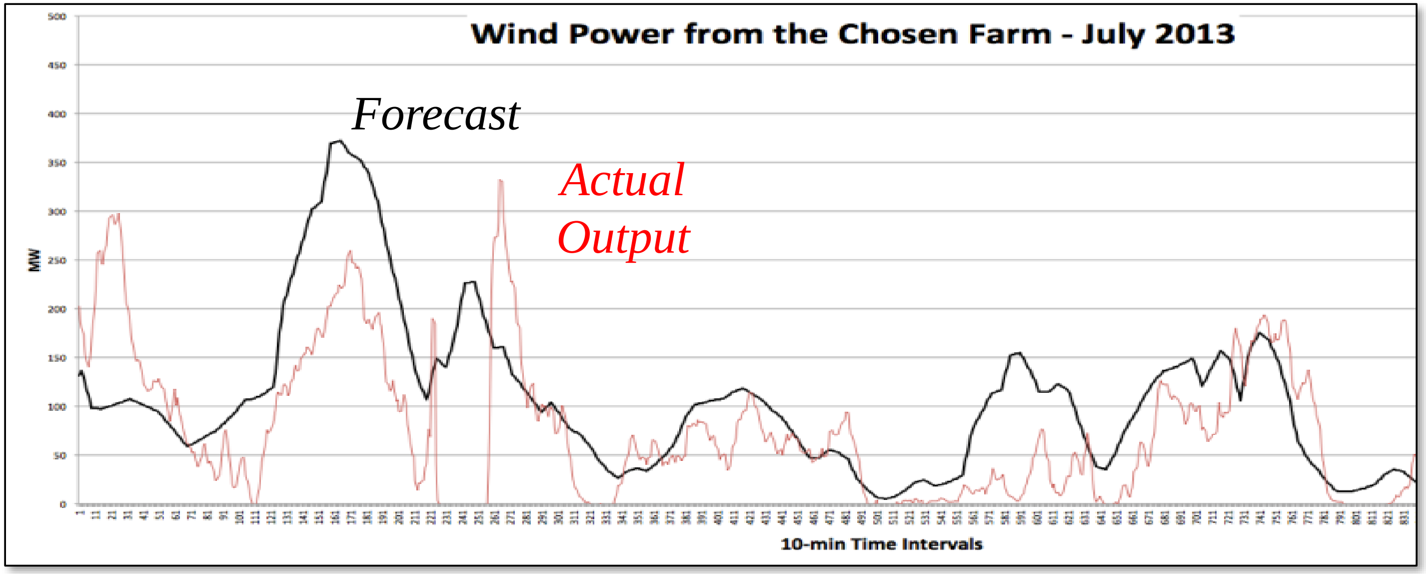




December 2016 - Bill Gates Launches $1 Billion Breakthrough Energy Investment Fund

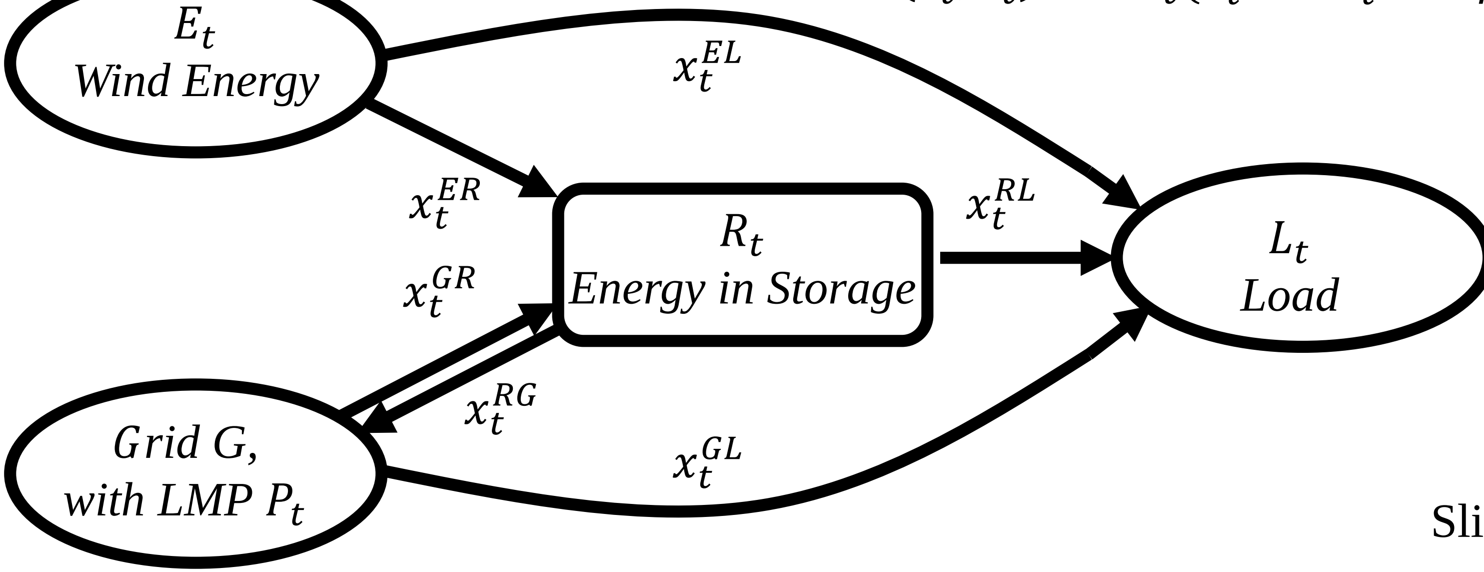
Motivation:

- Characteristics of renewable sources (wind, solar):
  - » Intermittency
  - » High volatility



Motivation: Wind Farm Paired with Storage Device

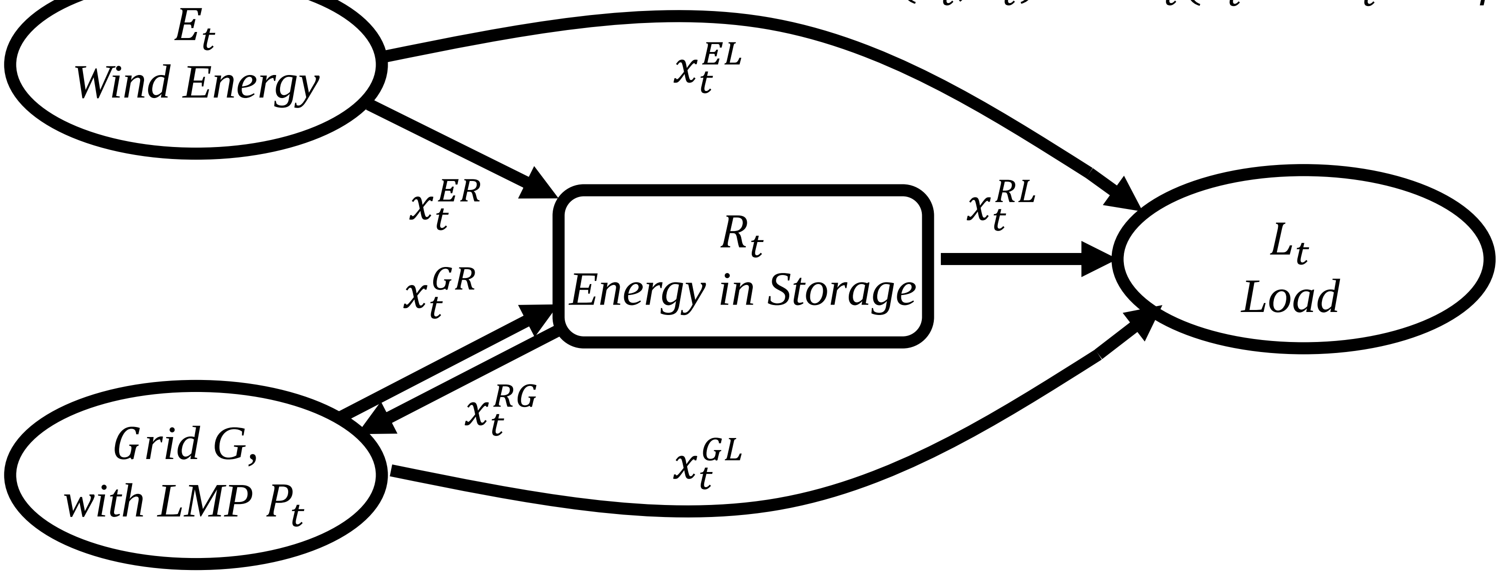
$$C(S_t, x_t) = -P_t(x_t^{GR} + x_t^{GL} - \eta x_t^{RG})$$



Motivation: Wind Farm Paired with Storage Device

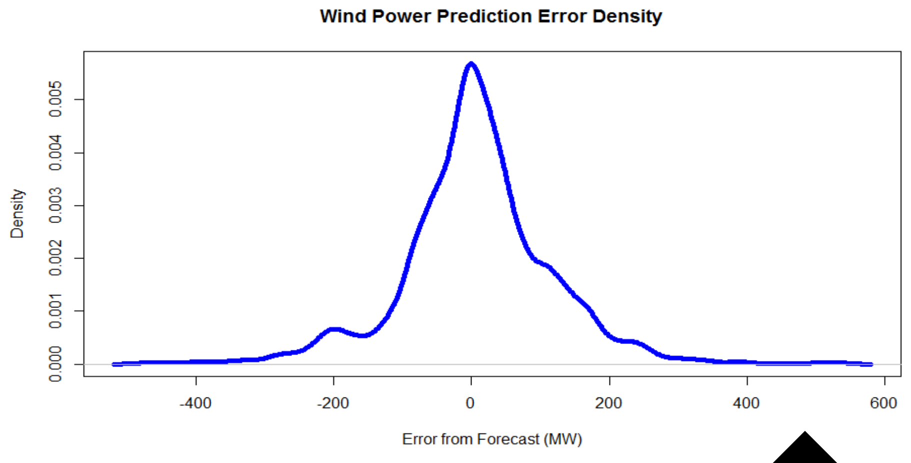
- Goal: Operate the system at minimum cost while satisfying the load at all times t
- First Step: Modeling the stochastic processes

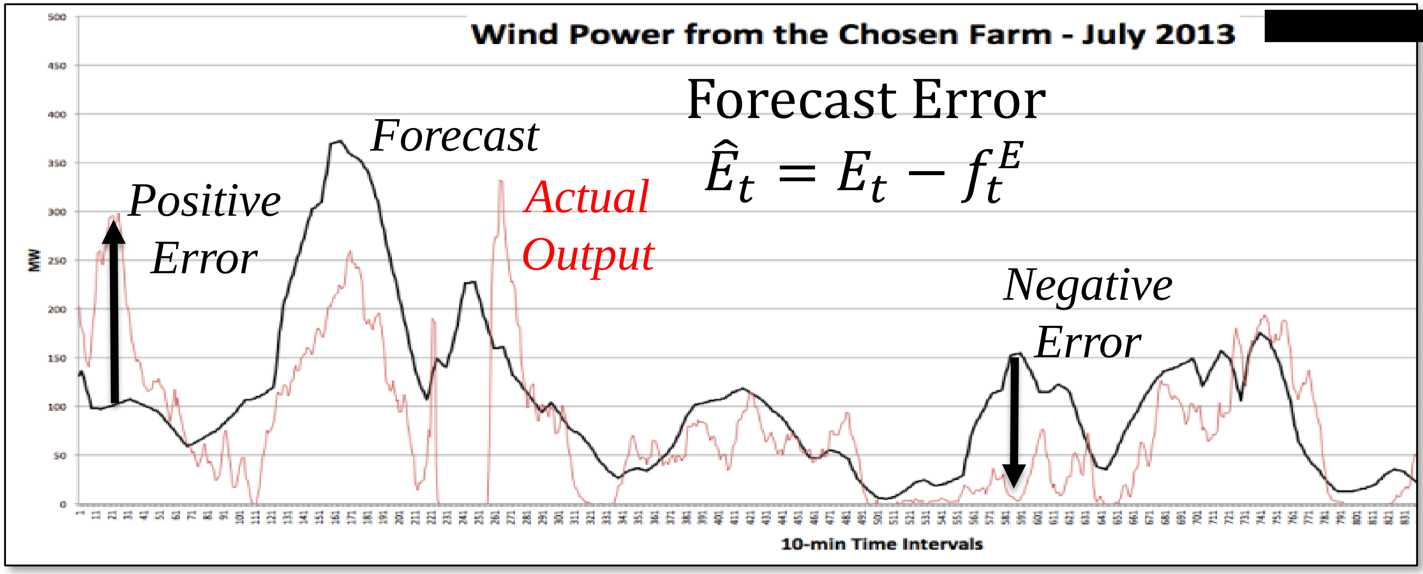
$$C(S_t, x_t) = -P_t(x_t^{GR} + x_t^{GL} - \eta x_t^{RG})$$



Error Distribution

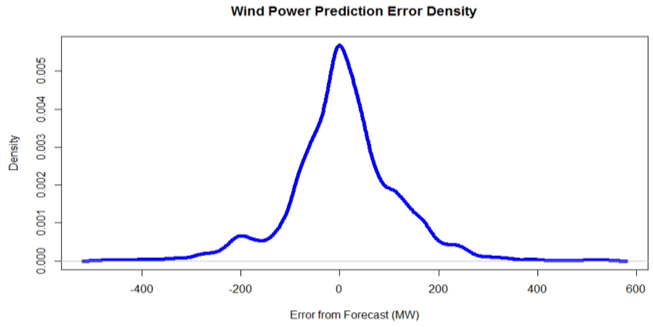
Empirical error distribution: $F^{\hat{E}}$



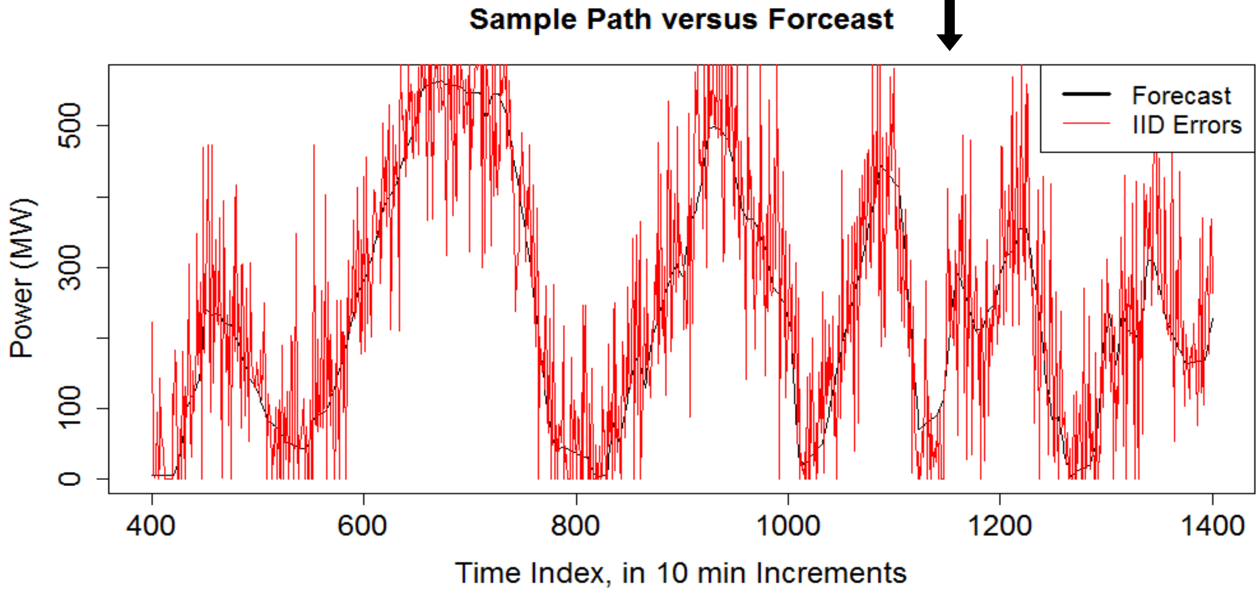


Possible Error Model: IID Errors

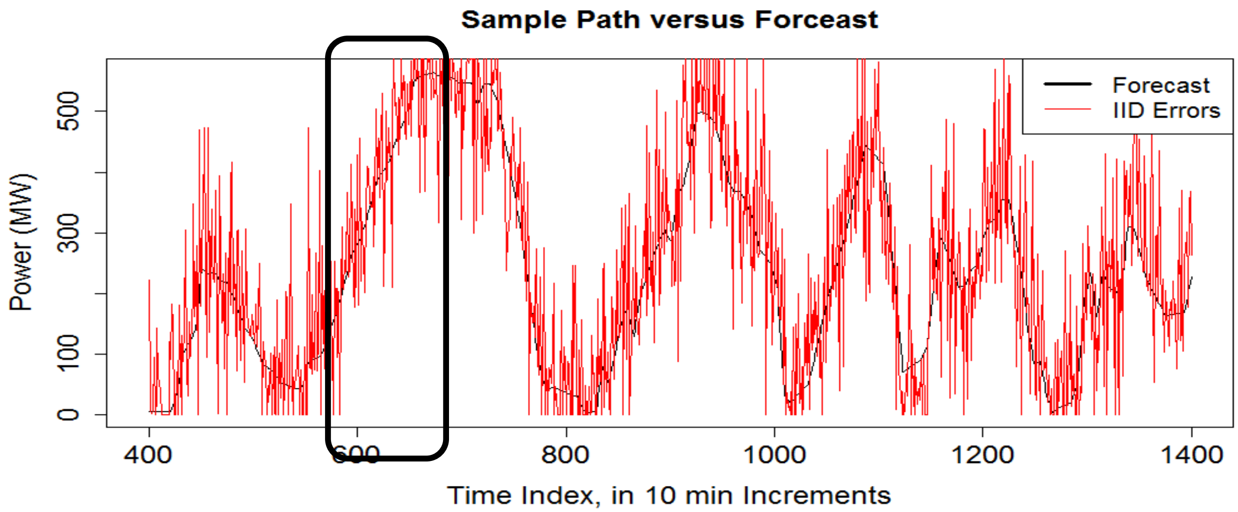
- $\hat{E}_t$ distributed according to empirical error distribution $F^{\hat{E}}$



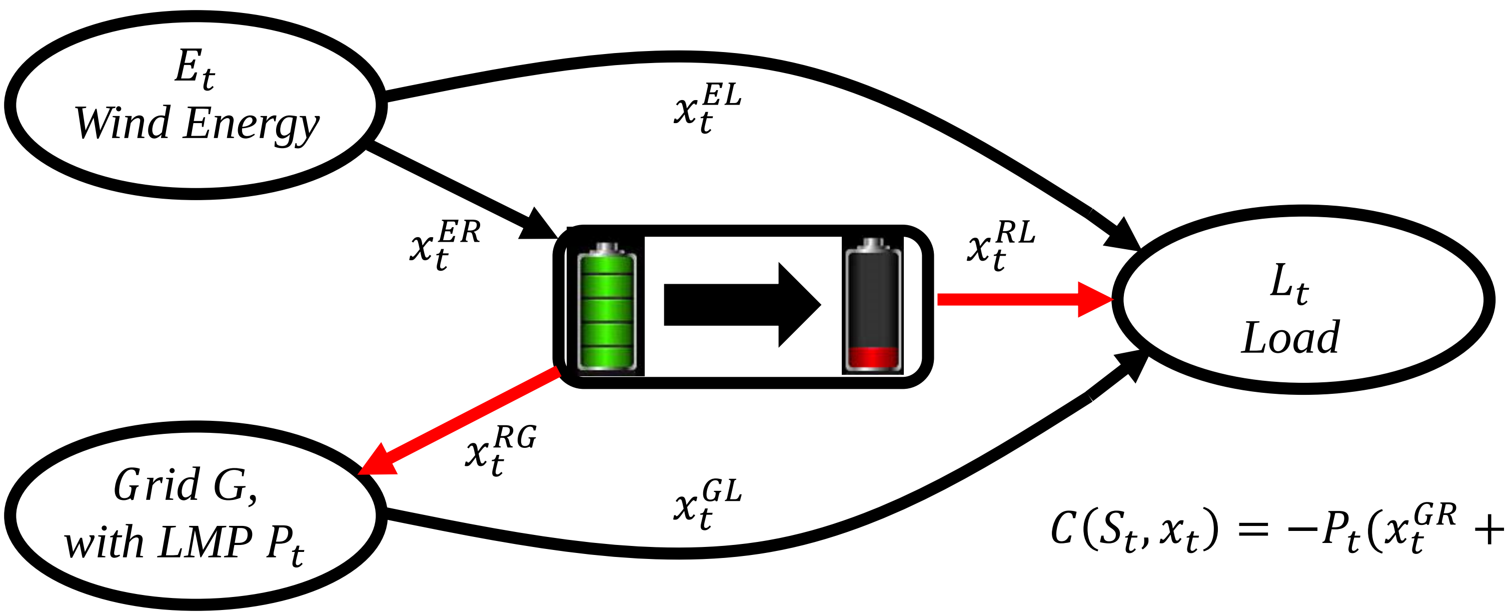
- Intertemporal independence



Policy based on IID Error Model

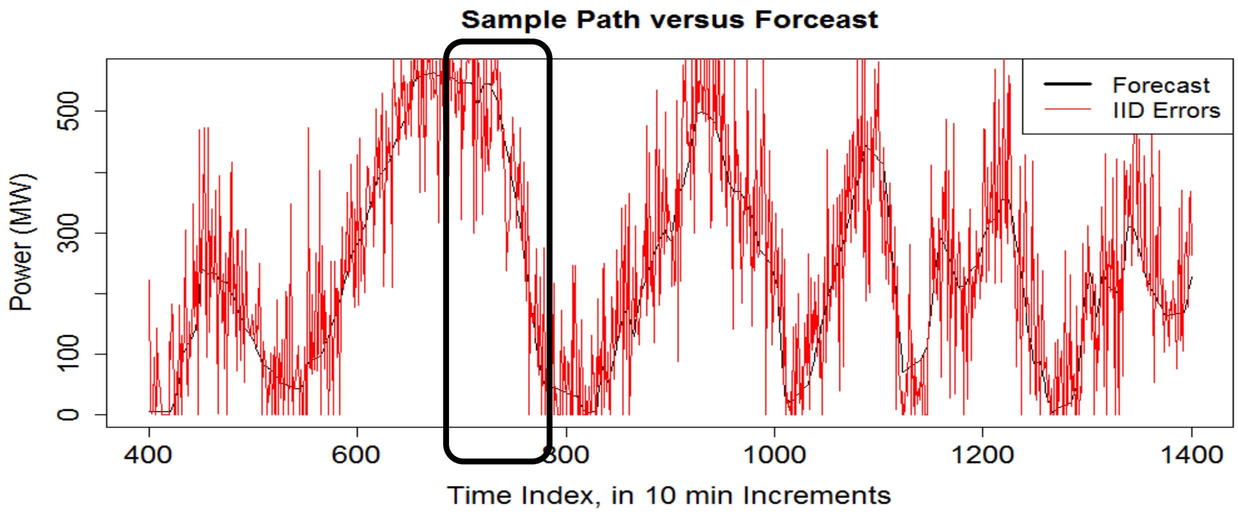


Time period A: Near full discharge of battery to maximize profit during this period

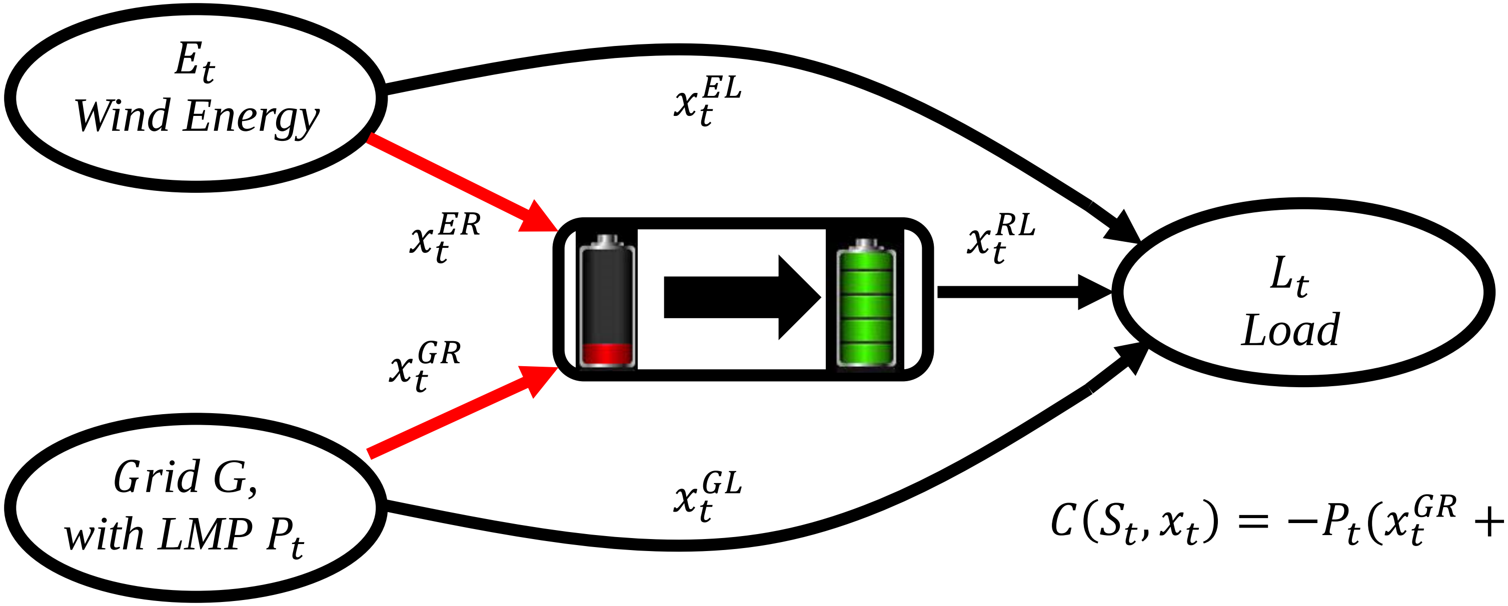


$$C(S_t, x_t) = -P_t(x_t^{GR} + x_t^{GL} - \eta x_t^{RG})$$

Policy based on IID Error Model

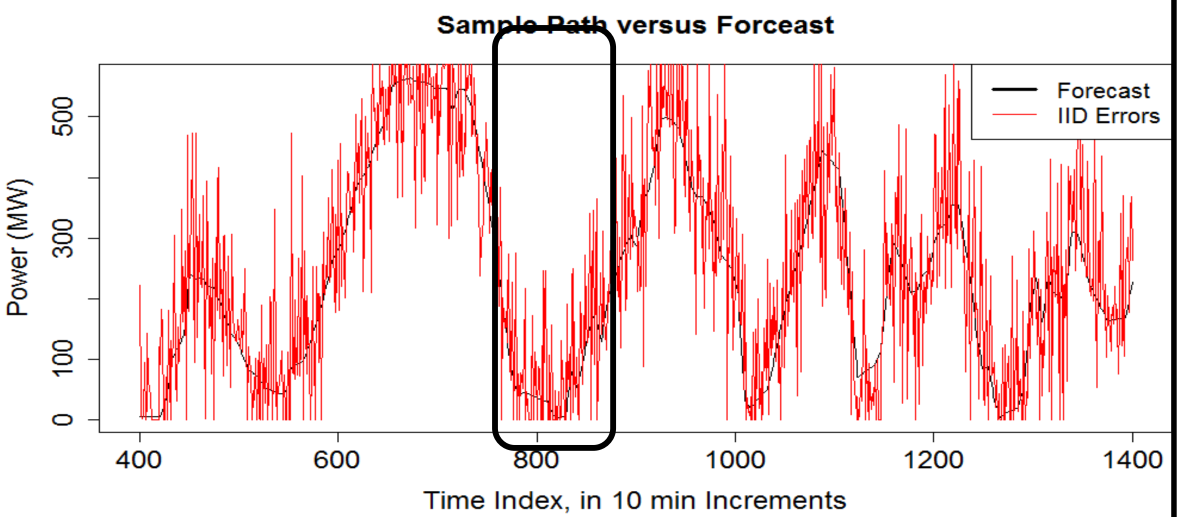


Time period B:
Plan for period of low wind by recharging battery

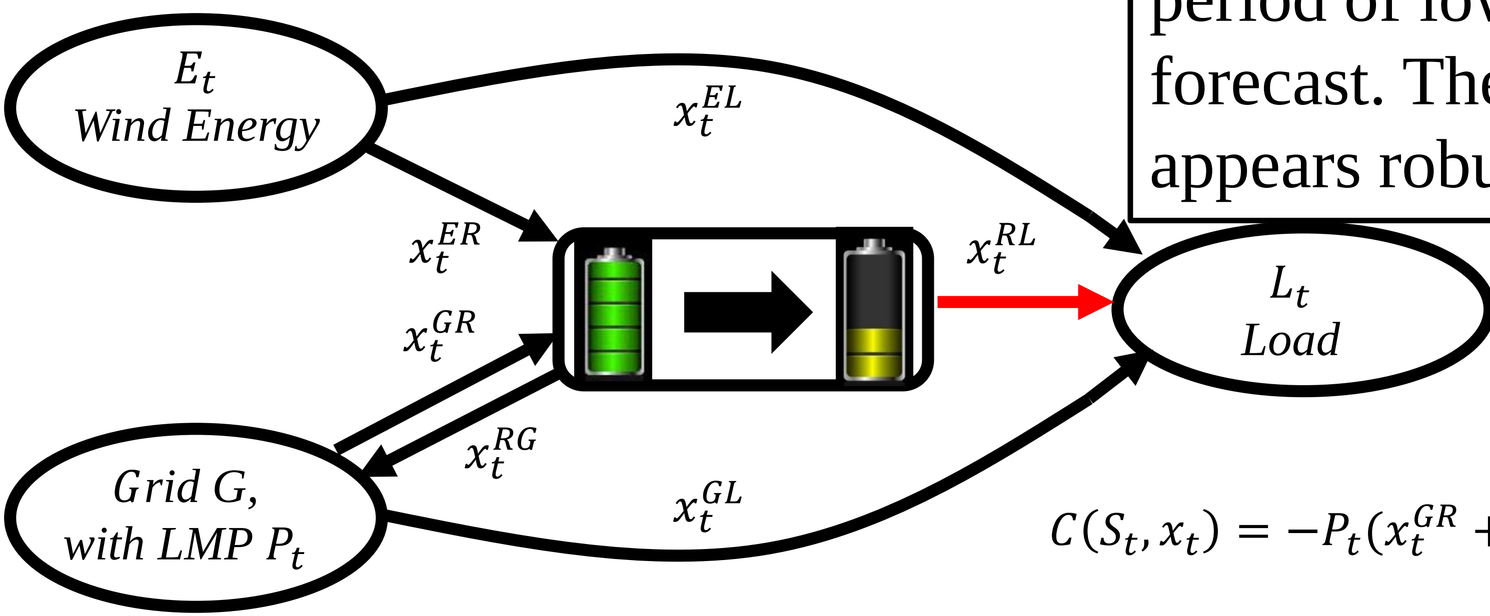


$$C(S_t, x_t) = -P_t(x_t^{GR} + x_t^{GL} - \eta x_t^{RG})$$

Policy based on IID Error Model

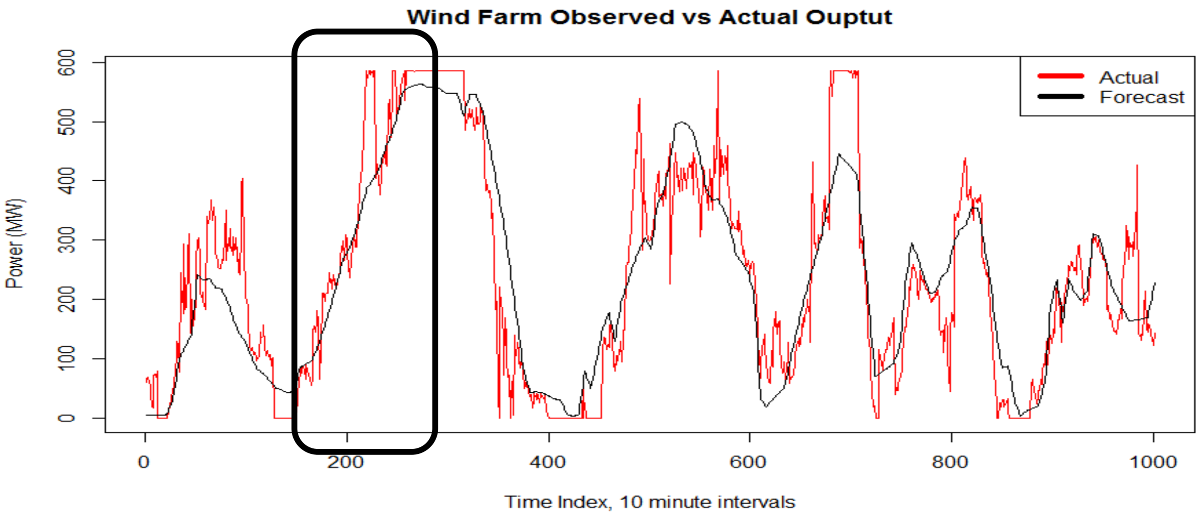


Time Period C: Assuming this model, in all scenarios, there is enough in storage from recharging in period B to account for the period of low wind forecast. The policy appears robust enough.

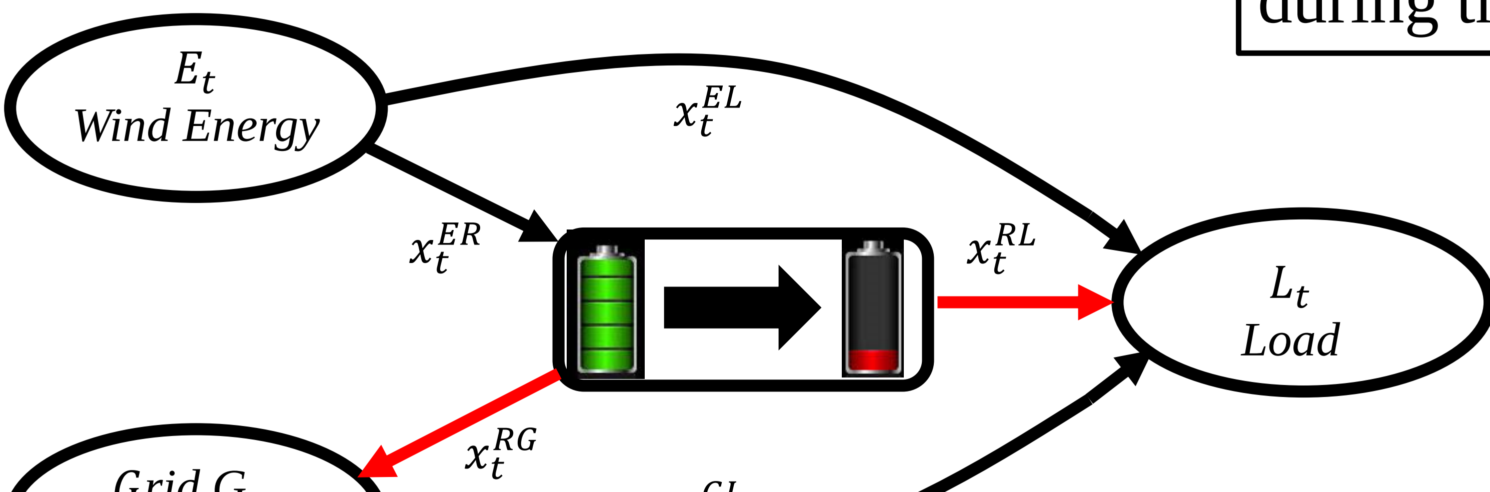


$$C(S_t, x_t) = -P_t(x_t^{GR} + x_t^{GL} - \eta x_t^{RG})$$

What can happen in practice:

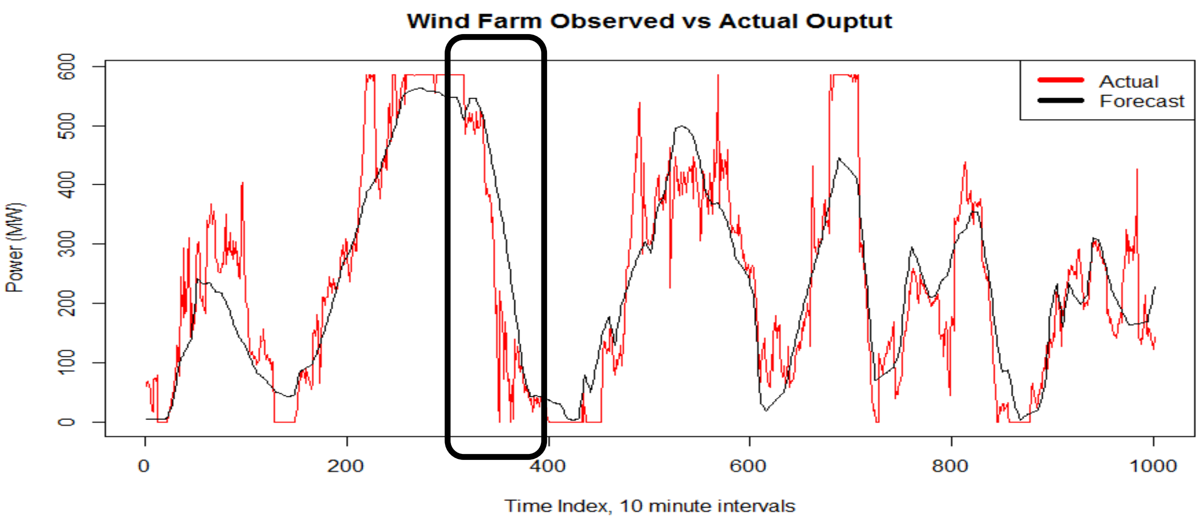


Time period A: According to policy, the battery is nearly fully discharged to maximize profit during this period

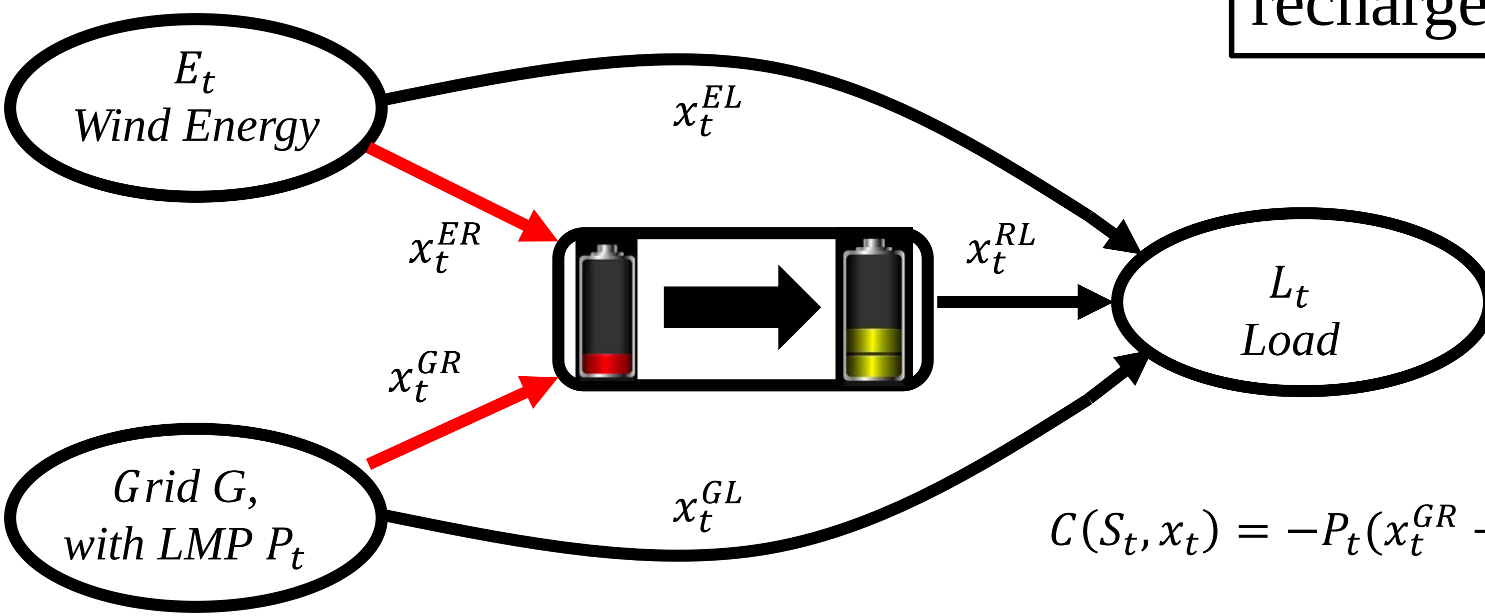


$$C(S_t, x_t) = -P_t(x_t^{GR} + x_t^{GL} - \eta x_t^{RG})$$

What can happen in practice:

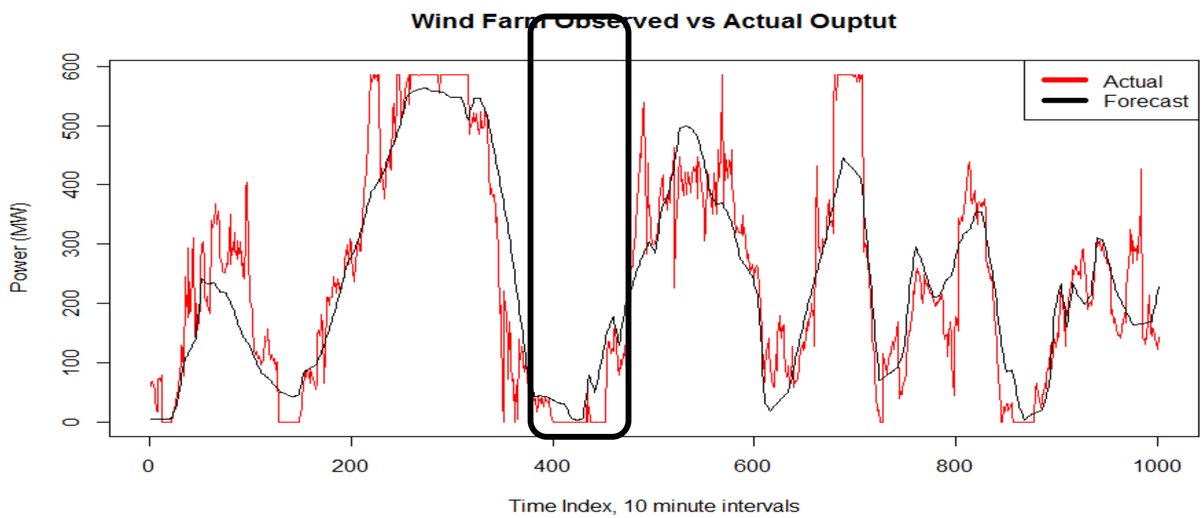


Time period B: Less wind than expected for an extended period of time, unable to fully recharge

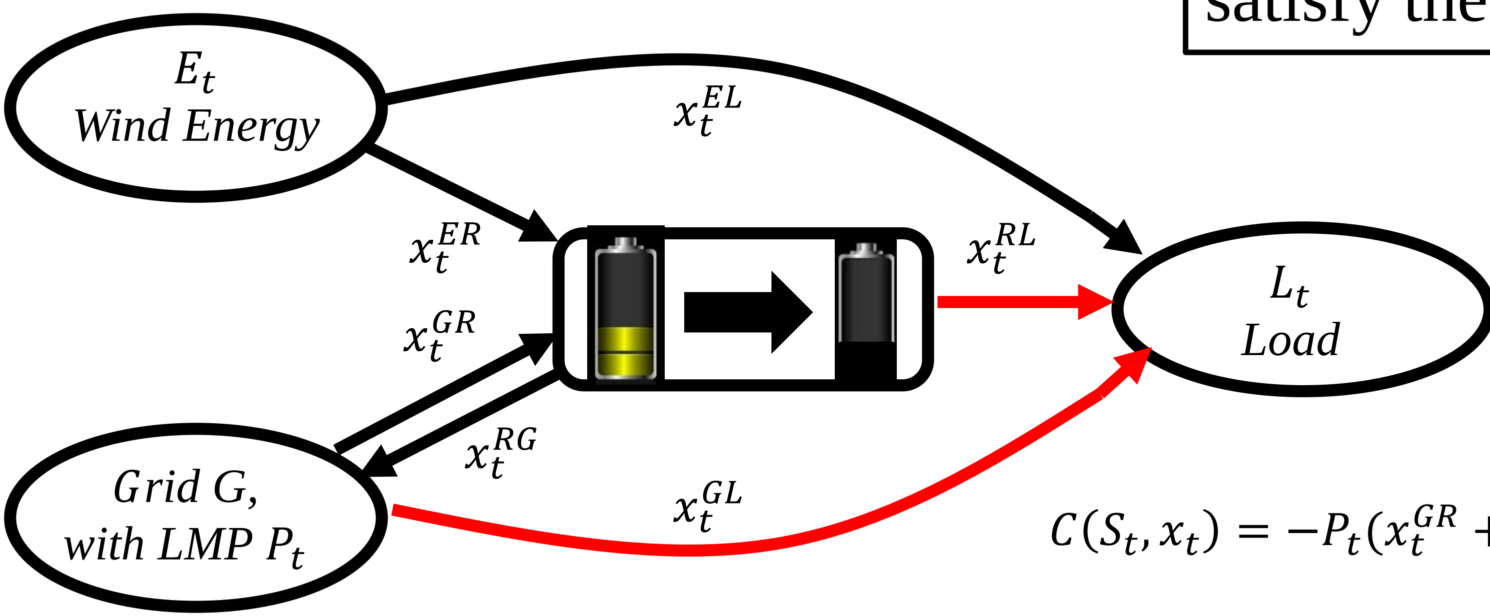


$$C(S_t, x_t) = -P_t(x_t^{GR} + x_t^{GL} - \eta x_t^{RG})$$

What can happen in practice:

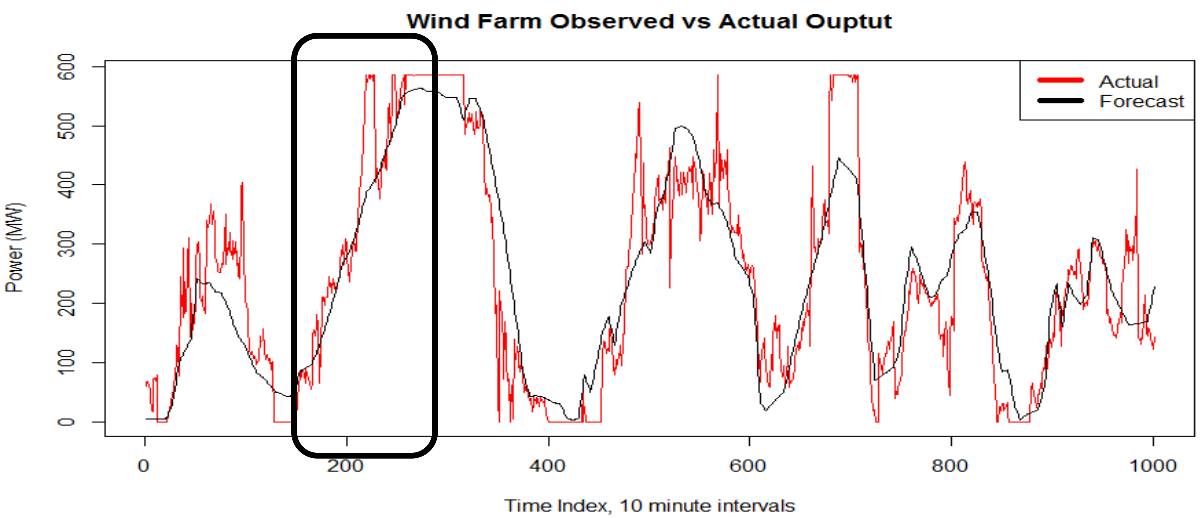


Time Period C: Battery drains completely, not much wind. Must buy from grid, no matter how high the LMP, to satisfy the load.

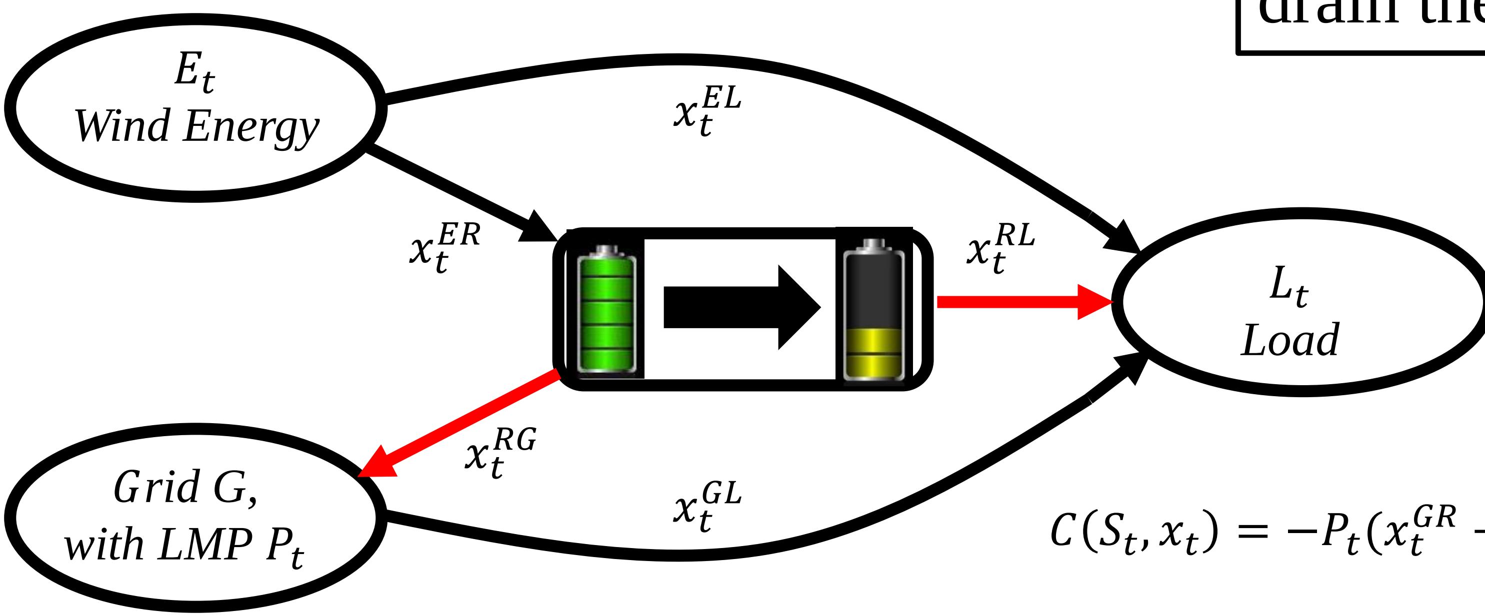


$$C(S_t, x_t) = -P_t(x_t^{GR} + x_t^{GL} - \eta x_t^{RG})$$

A More Robust Policy:

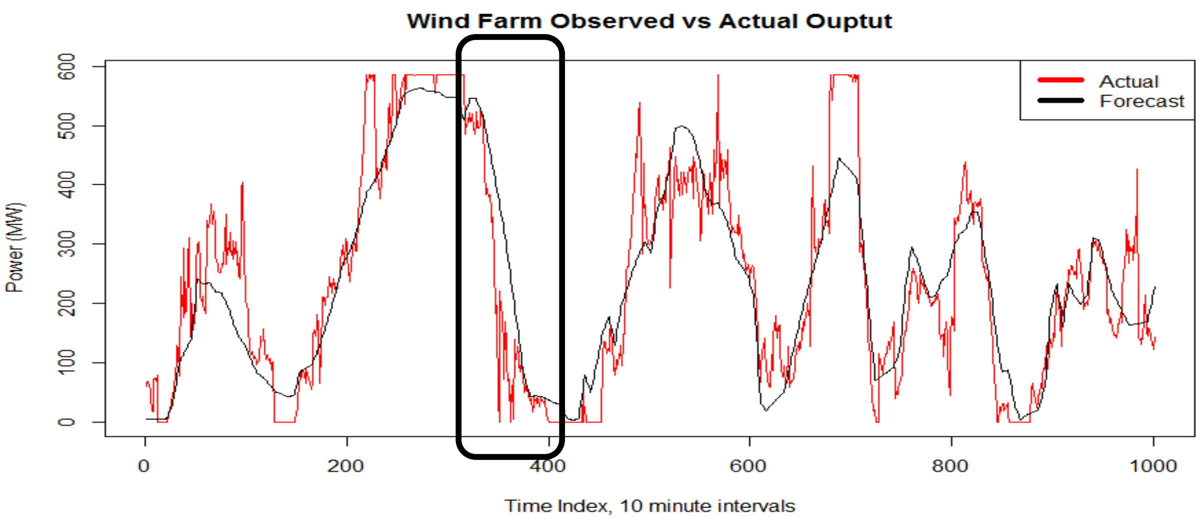


Time period A: Maximize profits by selling during high LMP periods, but do not completely drain the battery

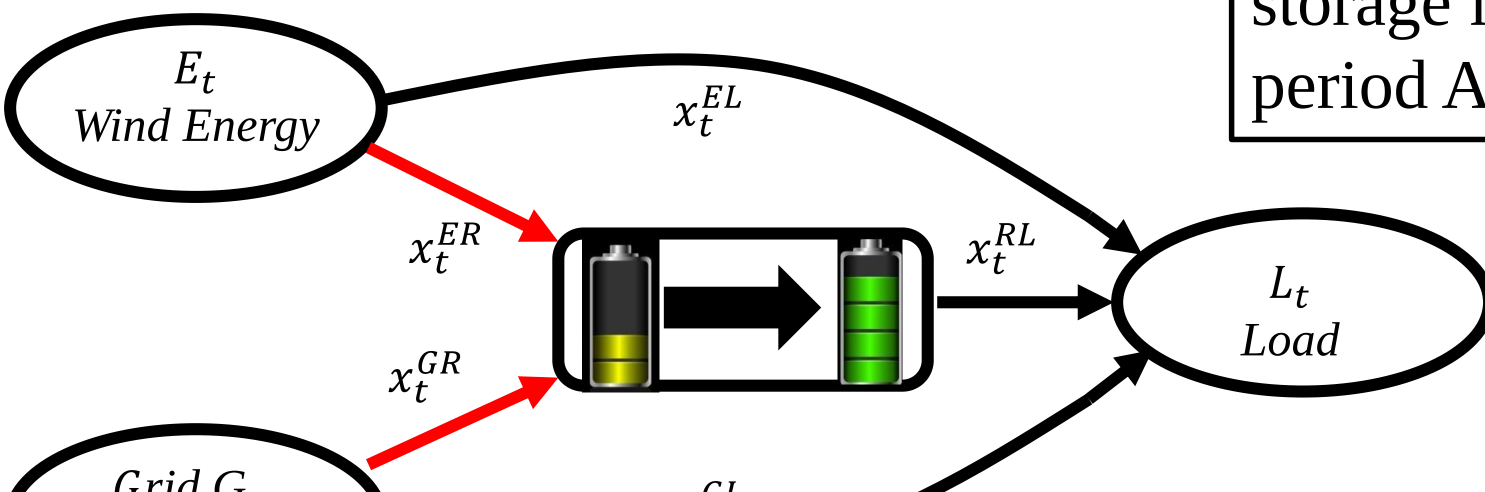


$$C(S_t, x_t) = -P_t(x_t^{GR} + x_t^{GL} - \eta x_t^{RG})$$

A More Robust Policy:

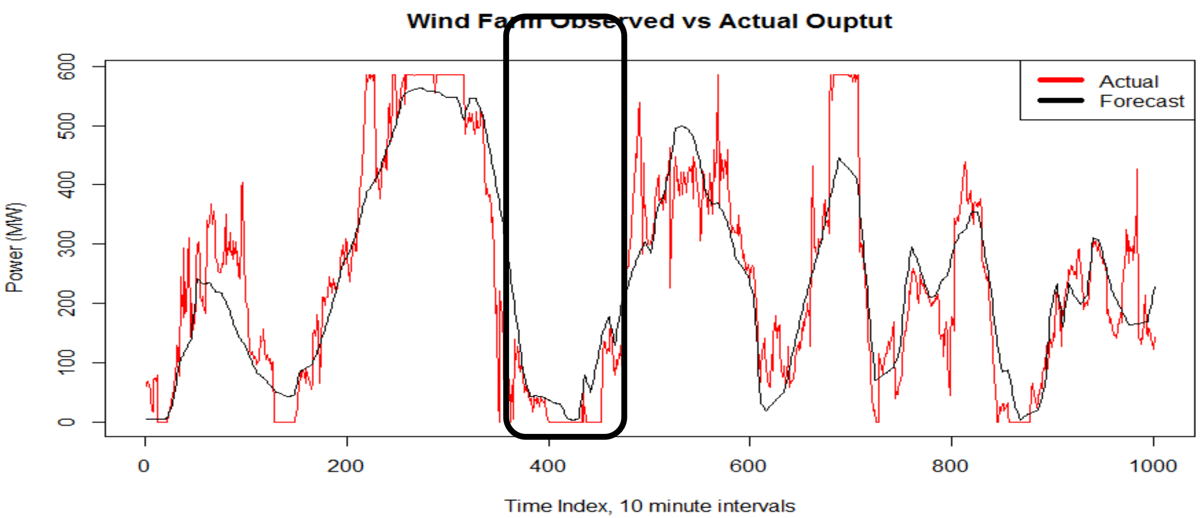


Time period B:
If there is less wind than expected now, we will still have backup energy in storage from time period A

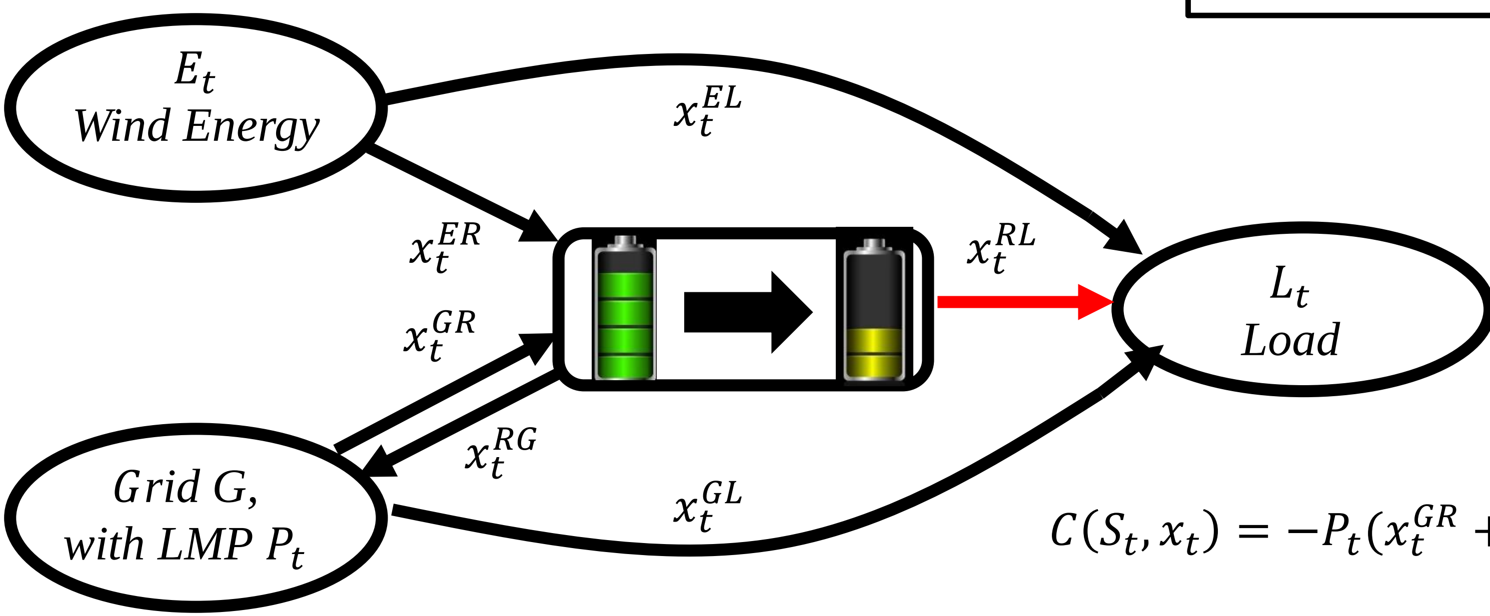


$$C(S_t, x_t) = -P_t(x_t^{GR} + x_t^{GL} - \eta x_t^{RG})$$

A More Robust Policy:



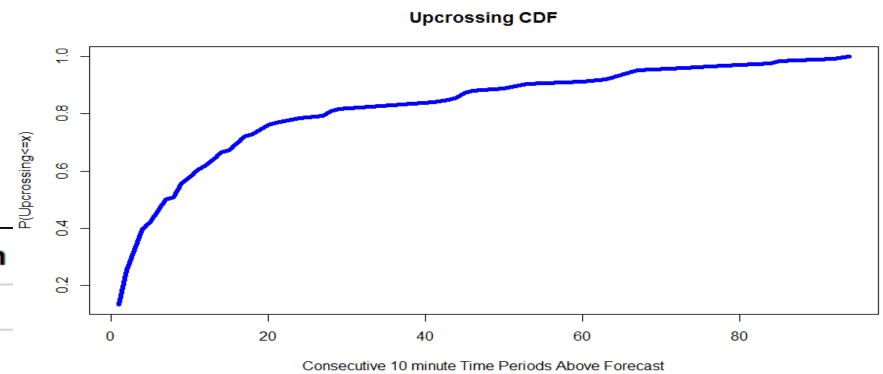
Time Period C:
Still enough energy in storage to handle period with little to no wind. The grid is an option, not a must.

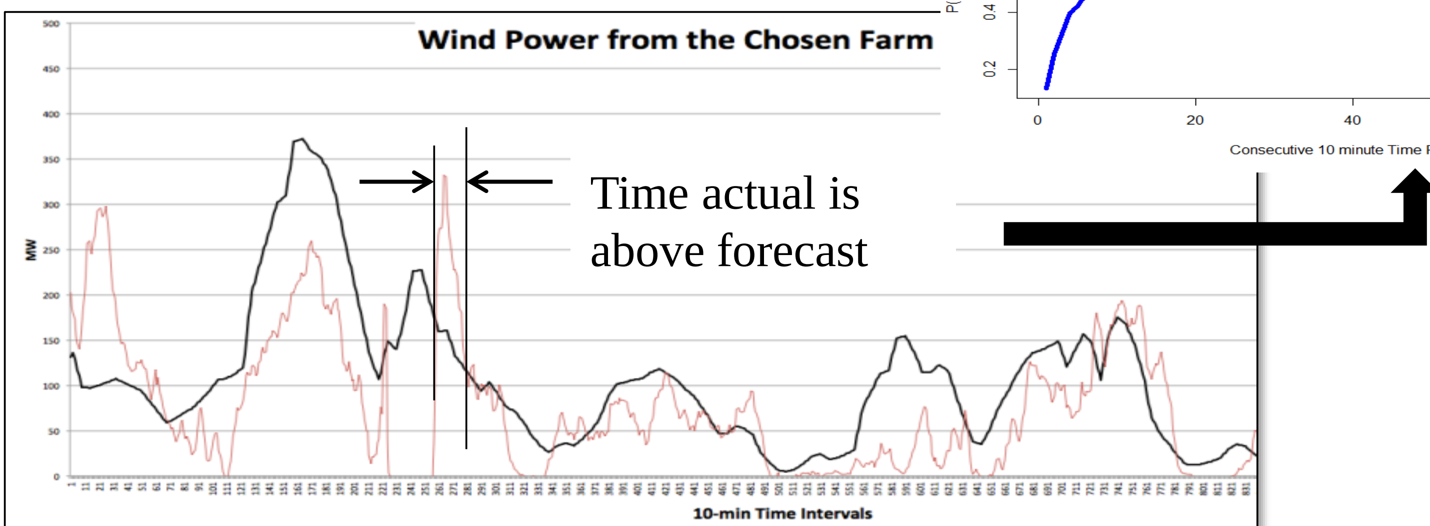


$$C(S_t, x_t) = -P_t(x_t^{GR} + x_t^{GL} - \eta x_t^{RG})$$

What went wrong?

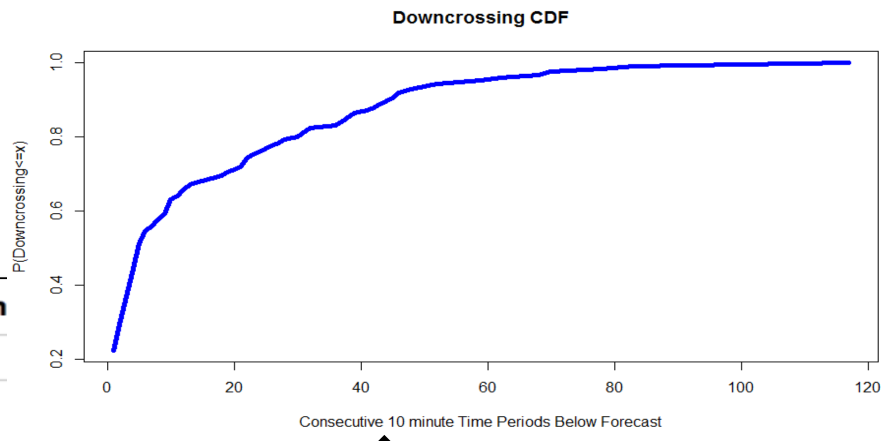
- The distribution of crossing times – contiguous blocks of time for which wind is above or below its forecast – were poorly replicated by the IID model.
- Up-crossings:

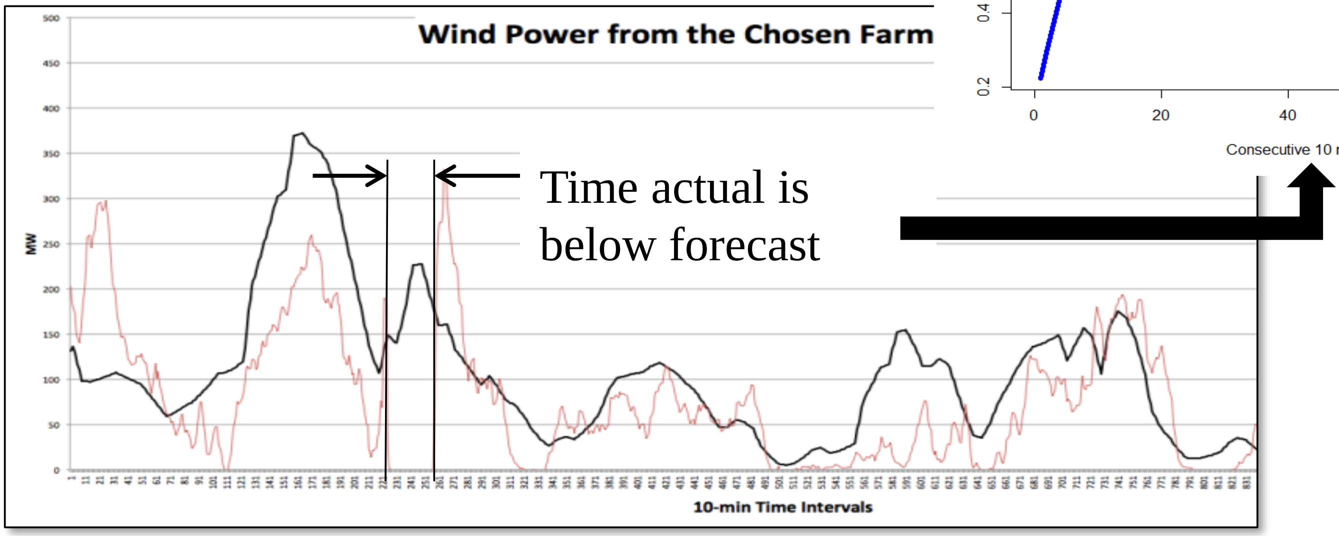




What went wrong?

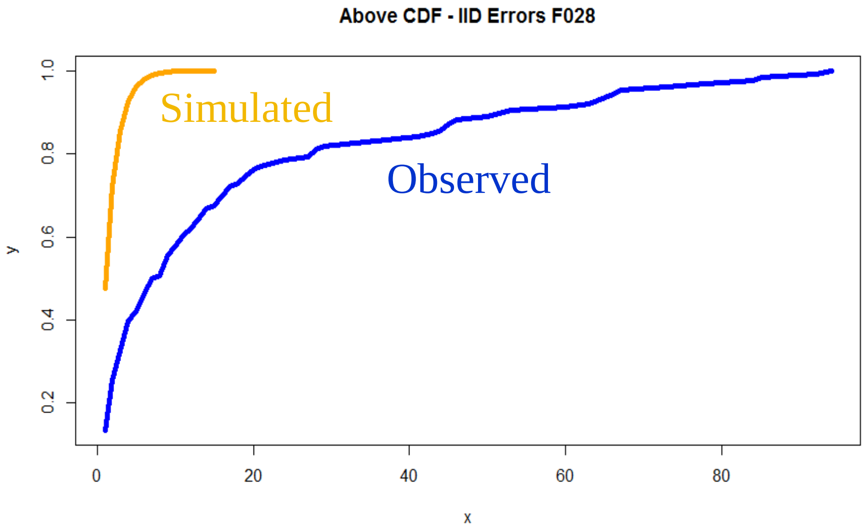
- The distribution of crossing times – contiguous blocks of time for which wind is above or below its forecast – were poorly replicated by the IID model.
- Down-crossings:

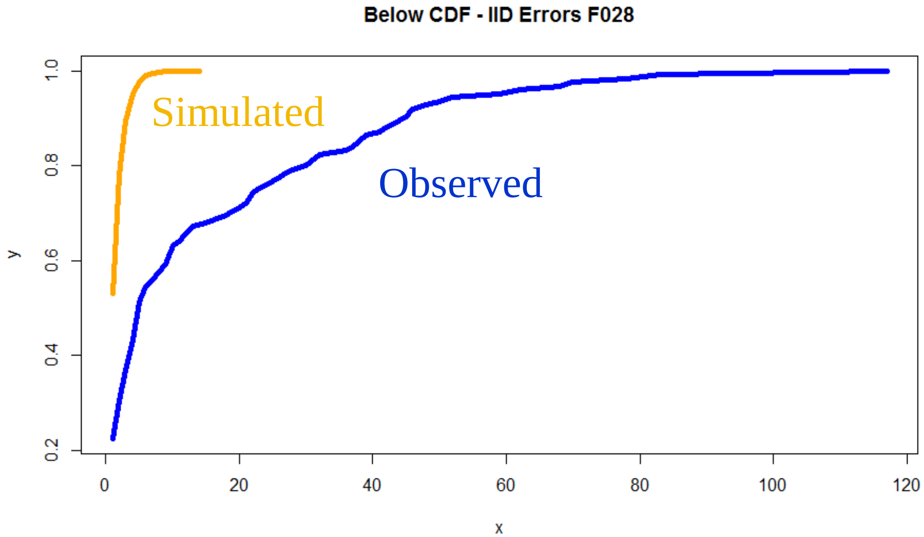




What went wrong?

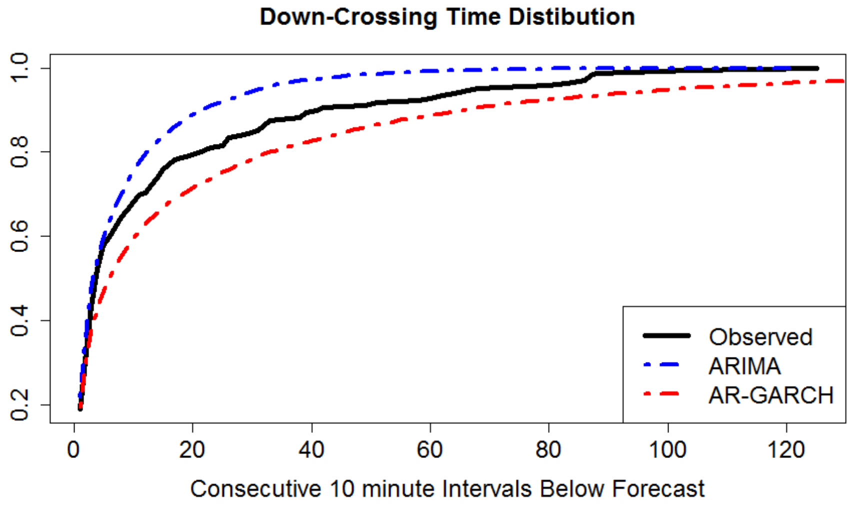
- The distribution of crossing times – contiguous blocks of time for which wind is above or below its forecast – were poorly replicated by the IID model.

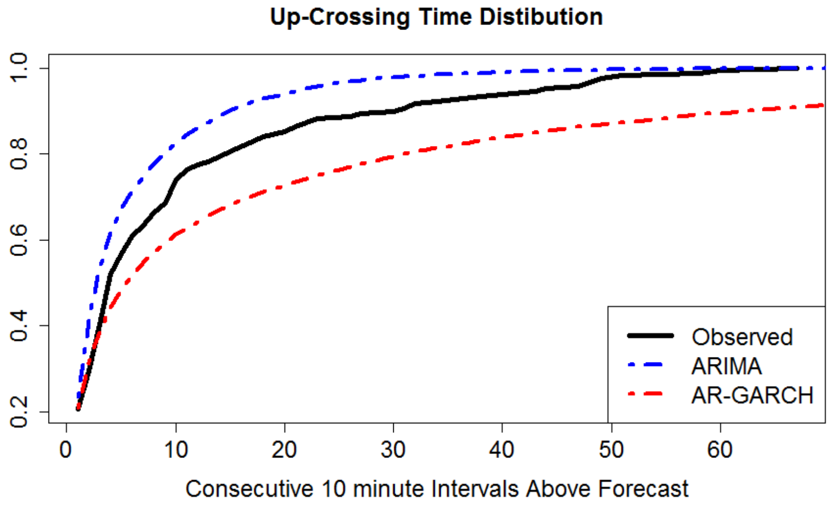




What went wrong?

- Other common error models that do not replicate crossing time distributions well:
  - » ARIMA
  - » ARIMA-GARCH





Outline

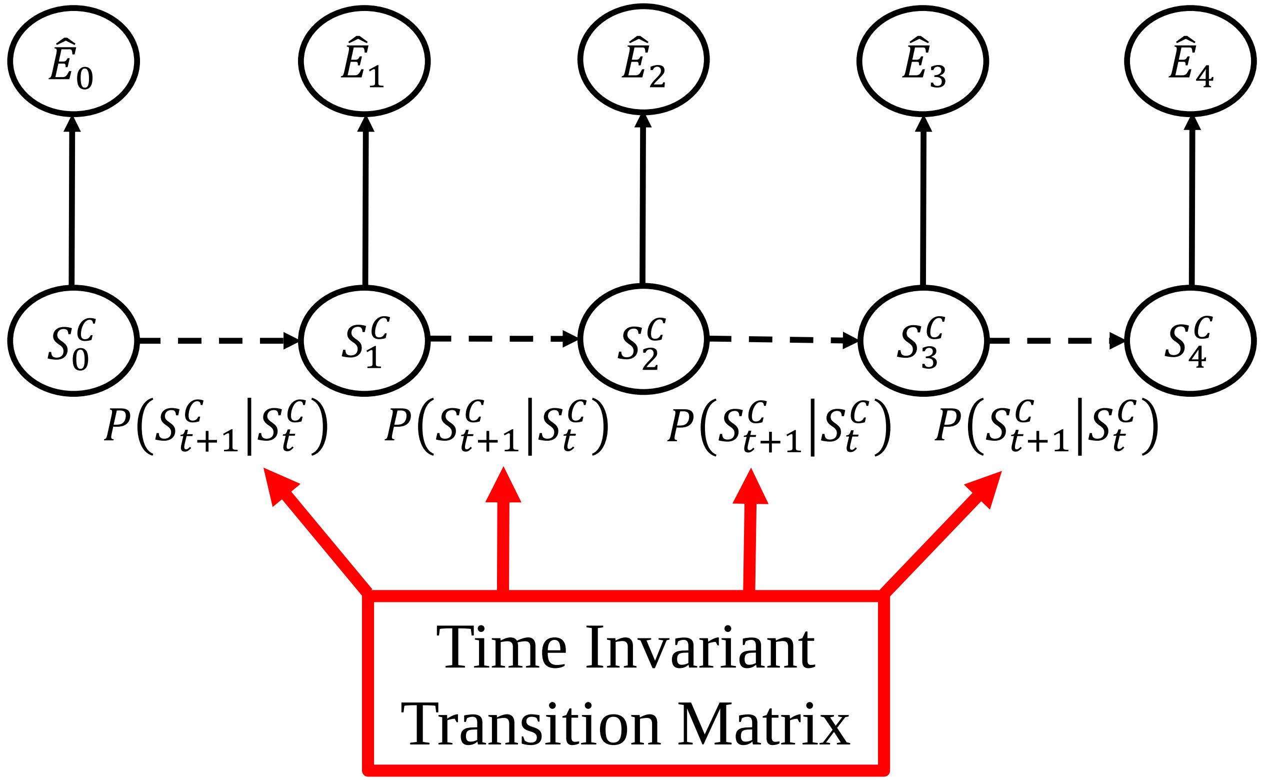
- Motivation
- Hidden Semi-Markov Crossing State Model
- Formulating the Energy Storage Problem as a Markov Decision Process
- Backward Approximate Dynamic Programming
- Numerical Results

Solution: Crossing State Models

- Incorporates a hidden state variable, the ***crossing state***, to control the crossing times of the process, forming a hidden semi-Markov model (HSMM).
- This state variable determines whether the process is above or below its forecast and for how long.

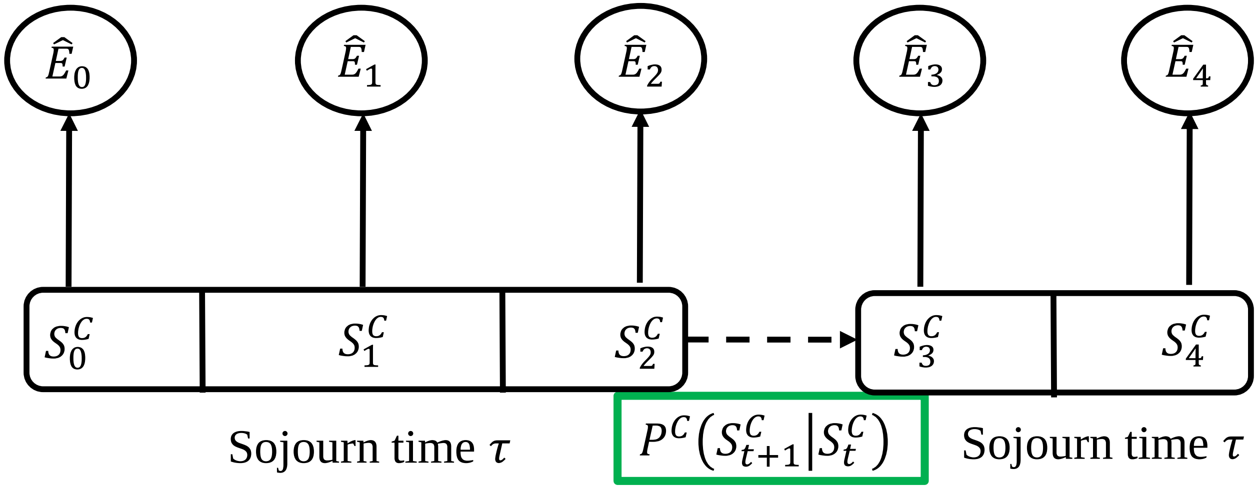
Univariate Crossing State Model

- Hidden Markov Model vs Hidden Semi-Markov Model:
  - » HMM:



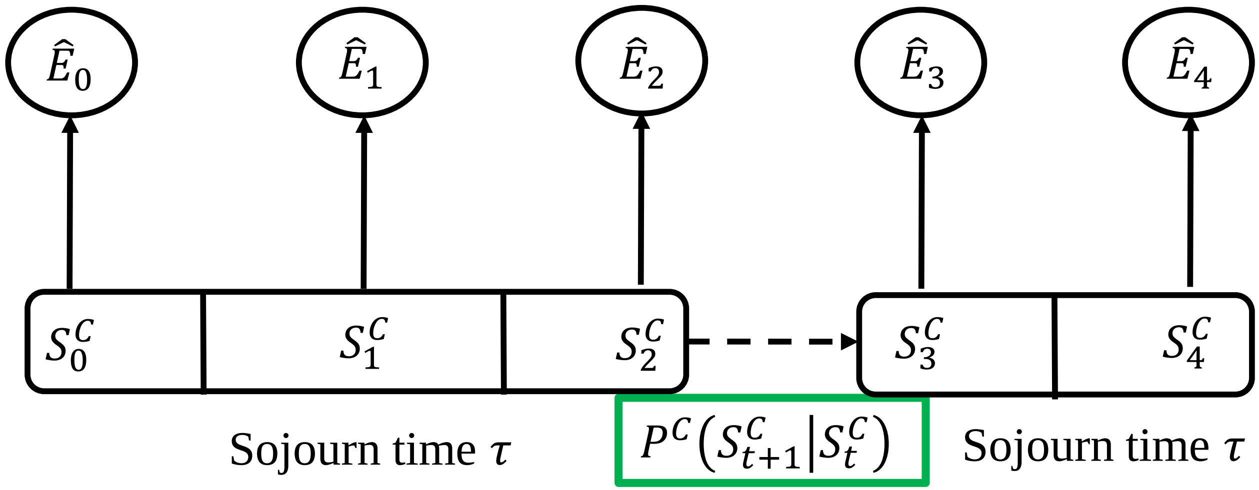
Univariate Crossing State Model

- Hidden Markov Model vs Hidden Semi-Markov Model:
  - » HSMM:



Univariate Crossing State Model

- Hidden Markov Model vs Hidden Semi-Markov Model:
  - » HSMM:

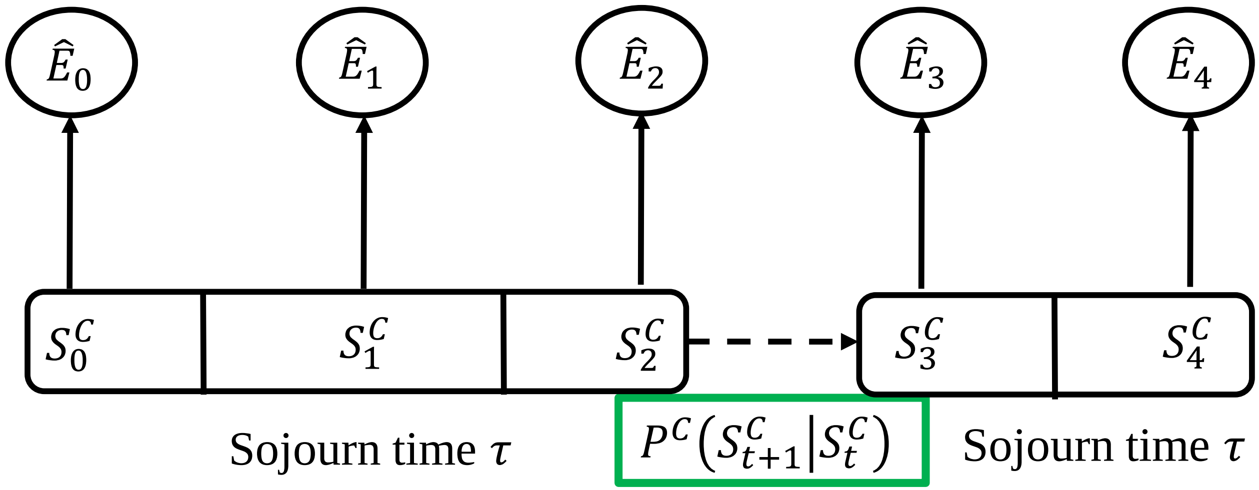


State duration-dependent transition matrix

$$P\left(S_{t+1}^C = s' \middle| S_t^C = s, \tau_t\right) = \begin{cases} \left(1 - F_s^\tau(\tau_t)\right) & if\ s' = s \\ F_s^\tau(\tau_t) P^C(s'|s) & if\ s' \neq s \end{cases}$$

Univariate Crossing State Model

- Hidden Markov Model vs Hidden Semi-Markov Model:
  - » HSMM:

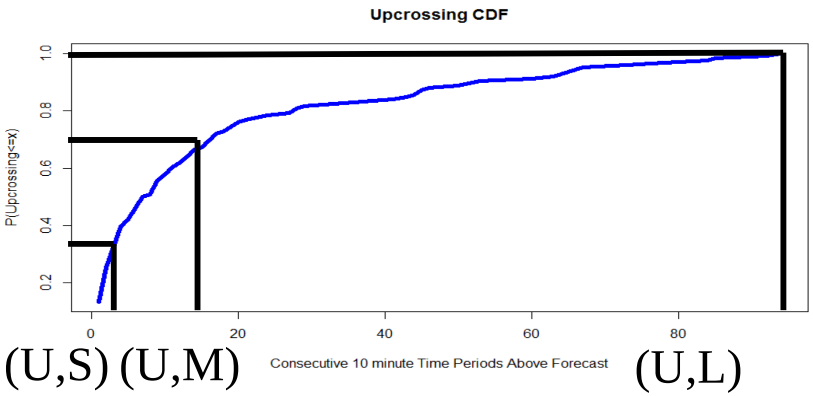


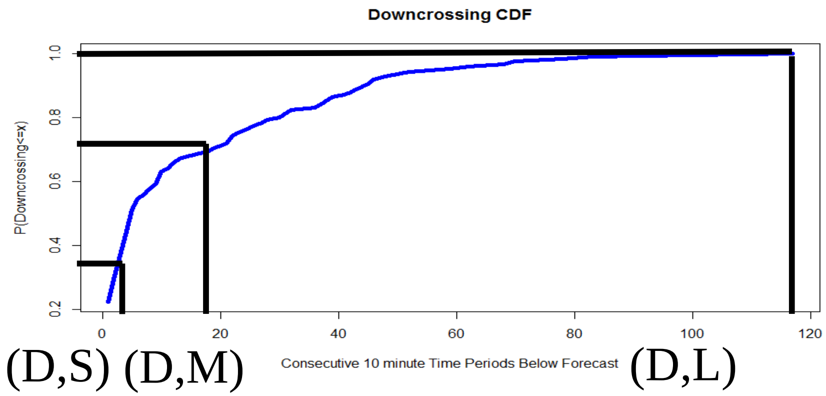
State duration-dependent transition matrix

$$P\left(S_{t+1}^C = s' \middle| S_t^C = s, \tau_t\right) = \begin{cases} \left(1 - F_s^\tau(\tau_t)\right) & if\ s' = s \\ F_s^\tau(\tau_t) P^C(s'|s) & if\ s' \neq s \end{cases}$$

Univariate Crossing State Model

- $S_t^C \equiv$ (U/D, S/M/L): the crossing state
  - » Observable: U/D: Up- or Down- Crossing
  - » Hidden: S/M/L: Short, Medium, or Long Crossing Time

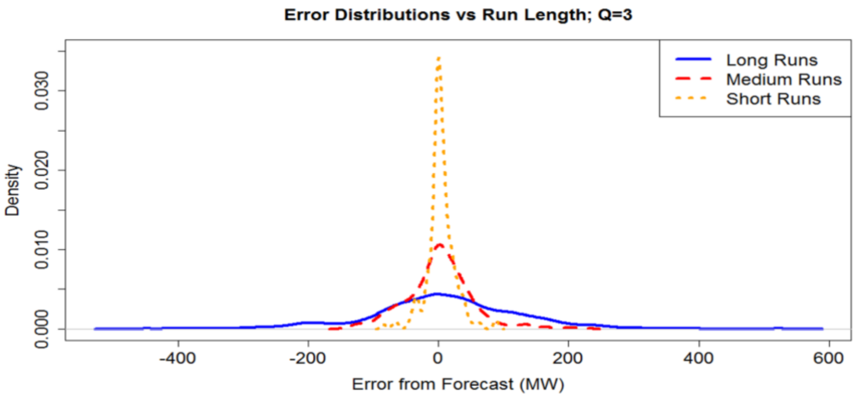




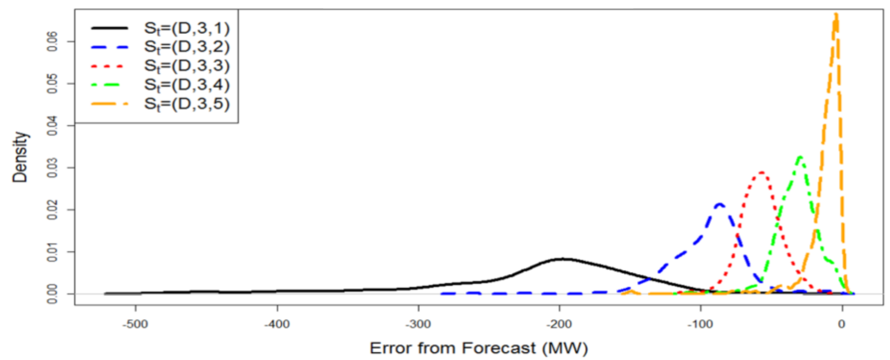
- For each crossing state s, there exists a distribution of crossing times F_s^{τ}. These also serve as the sojourn time distributions for the crossing states.

Univariate Crossing State Model

- Error generation is conditioned on the crossing state as well
- $\hat{E}_t^g$ = Aggregated Forecast Error
  - » Aggregated into *b* bins for each crossing state
  - » Partitions are based on error quantiles given the crossing state



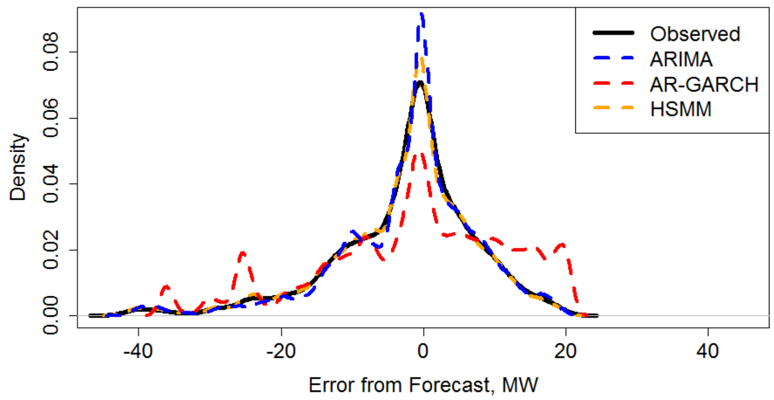
$P(\hat{E}_{t+1}|\hat{E}_t^g, S_t^C)$

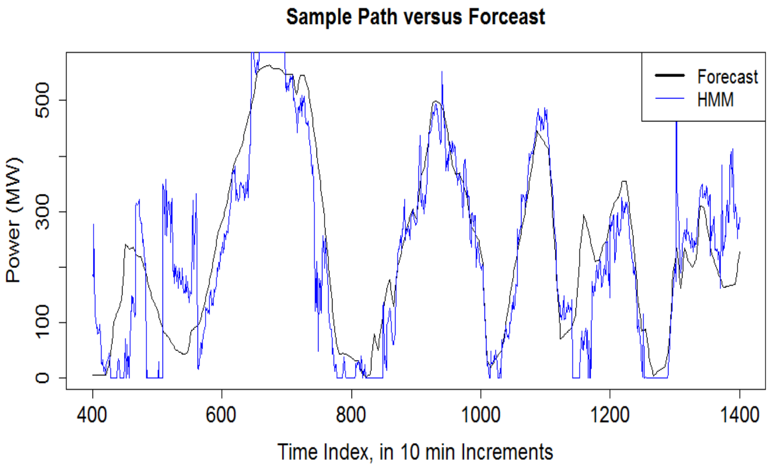


- $P(\hat{E}_{t+1}|\hat{E}_t^g, S_t^C)$ density of next error given current error bin and crossing state

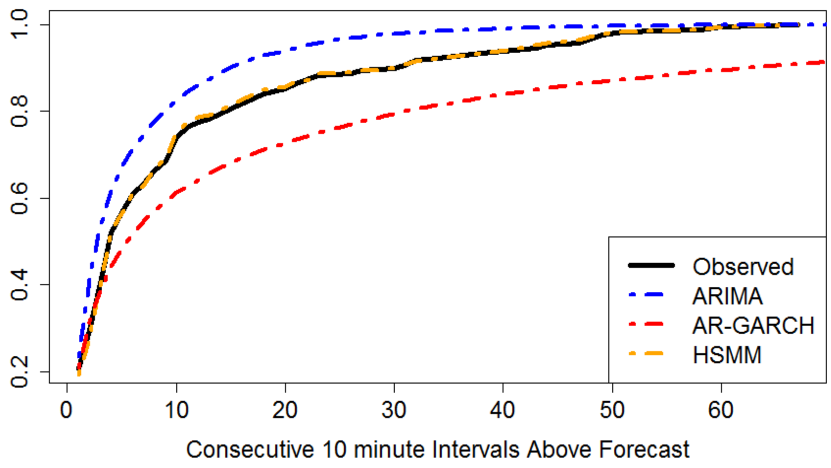
Resulting Distributions:

Error Density

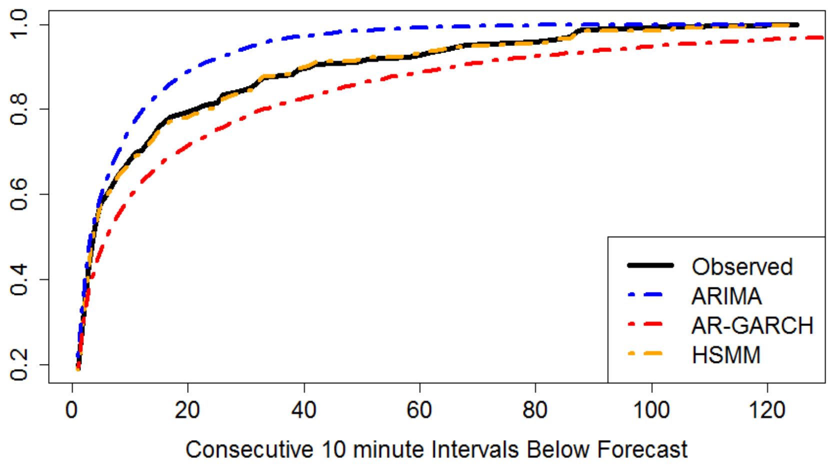




Up-crossing CDF



Down-crossing CDF



Outline

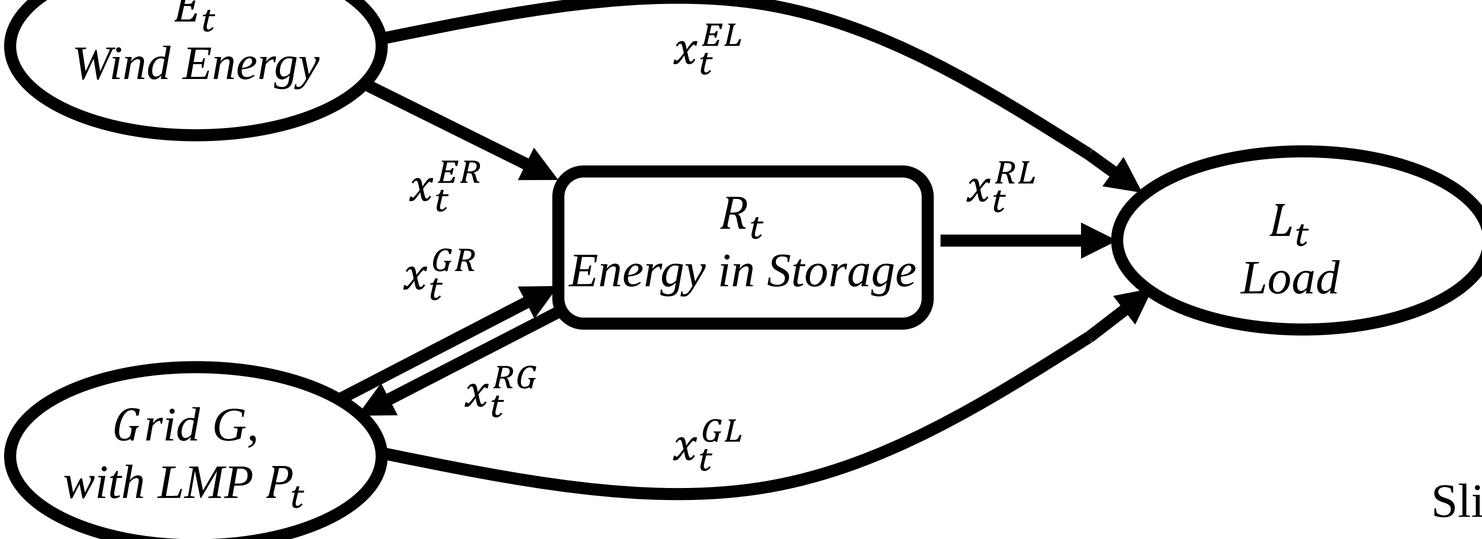
- Motivation
- Hidden Semi-Markov Crossing State Model
- Formulating the Energy Storage Problem as a Markov Decision Process
- Backward Approximate Dynamic Programming
- Numerical Results

Goals:

- Create a very realistic energy storage problem by using crossing state models for the stochastics involved
- Develop near optimal control policies using backward approximate dynamic programming (ADP)

Back to the Energy Storage Problem

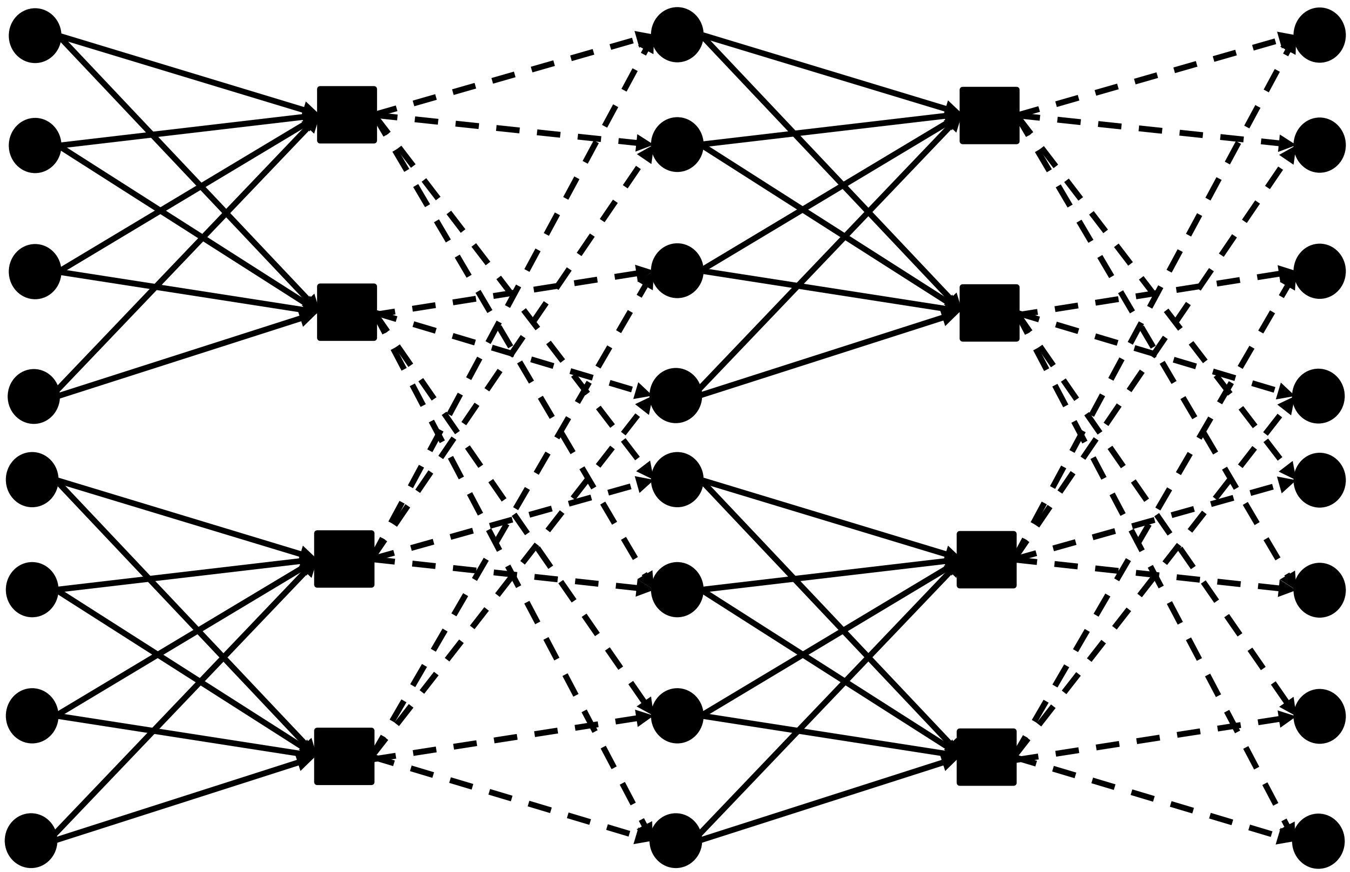
$$C(S_t, x_t) = -P_t(x_t^{GR} + x_t^{GL} - \eta x_t^{RG})$$



Formulating the Energy Storage Problem as a Markov Decision Process

Feasible Decision

Transition Probability

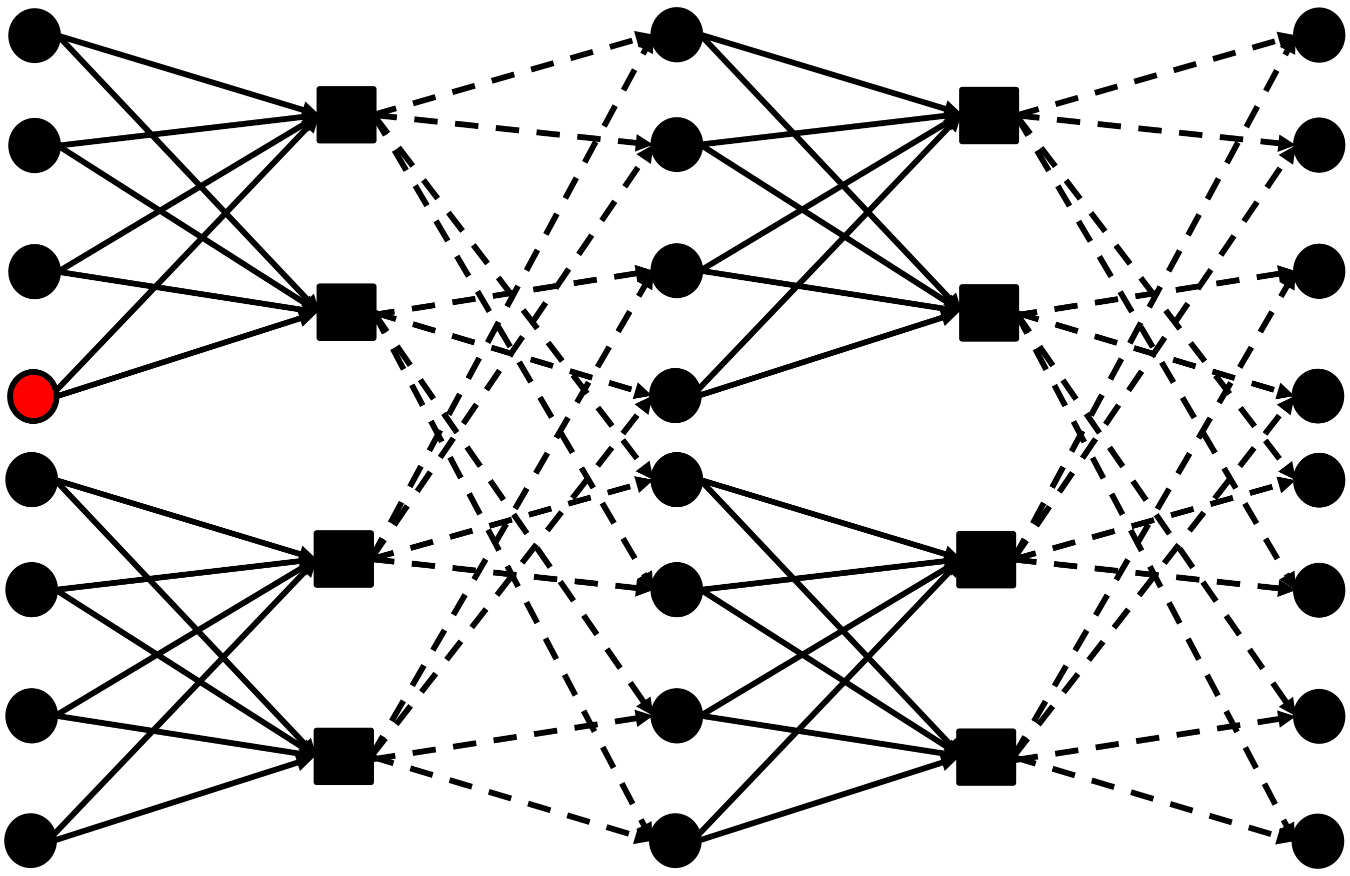


Formulating the Energy Storage Problem as a Markov Decision Process

Feasible Decision

Transition Probability

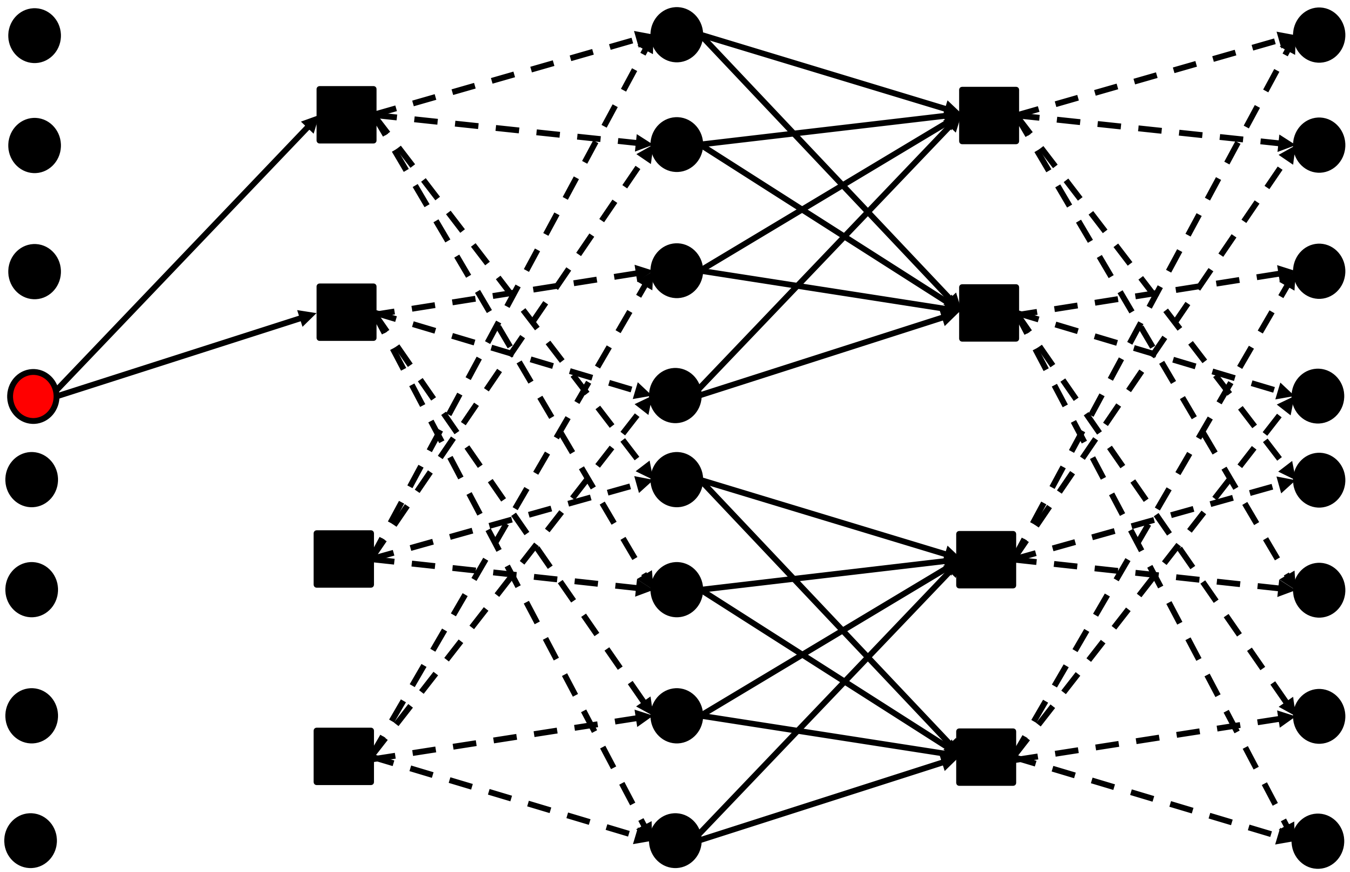
Pre-decision state S_t



Formulating the Energy Storage Problem as a Markov Decision Process

Feasible Decision
Transition Probability

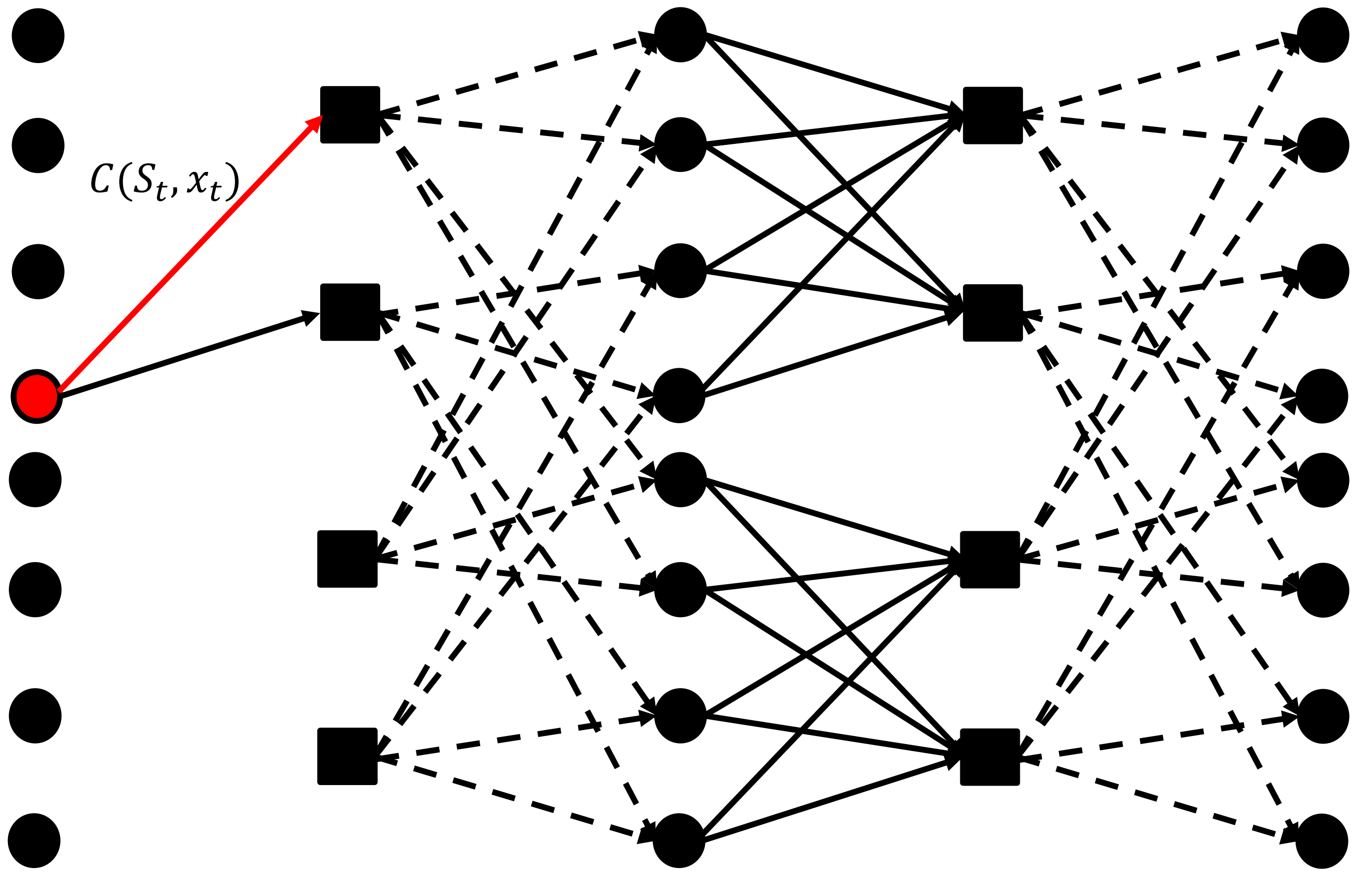
Pre-decision state S_t



Formulating the Energy Storage Problem as a Markov Decision Process

Feasible Decision
Transition Probability

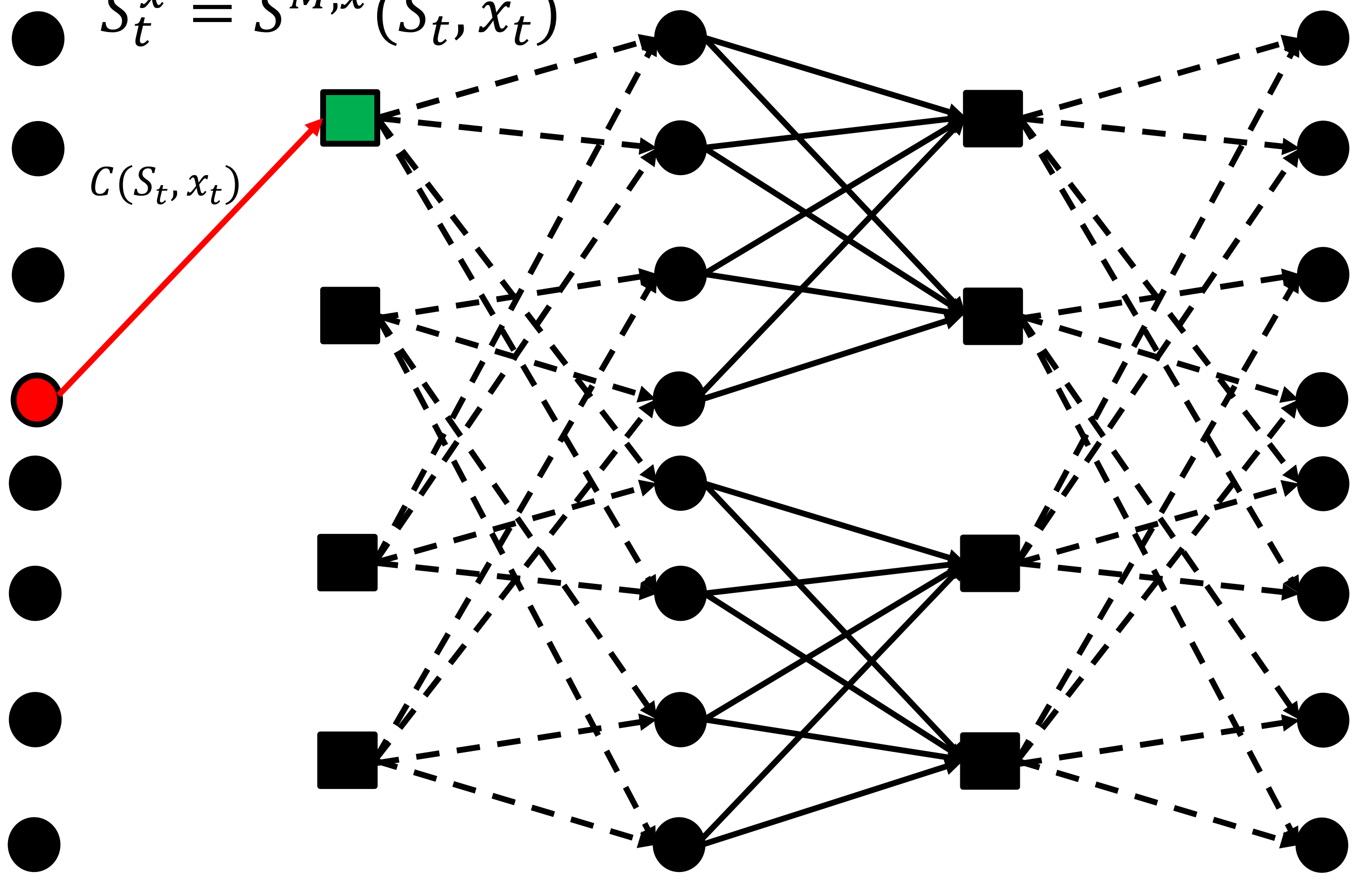
Pre-decision state S_t



Formulating the Energy Storage Problem as a Markov Decision Process

Feasible Decision
Transition Probability

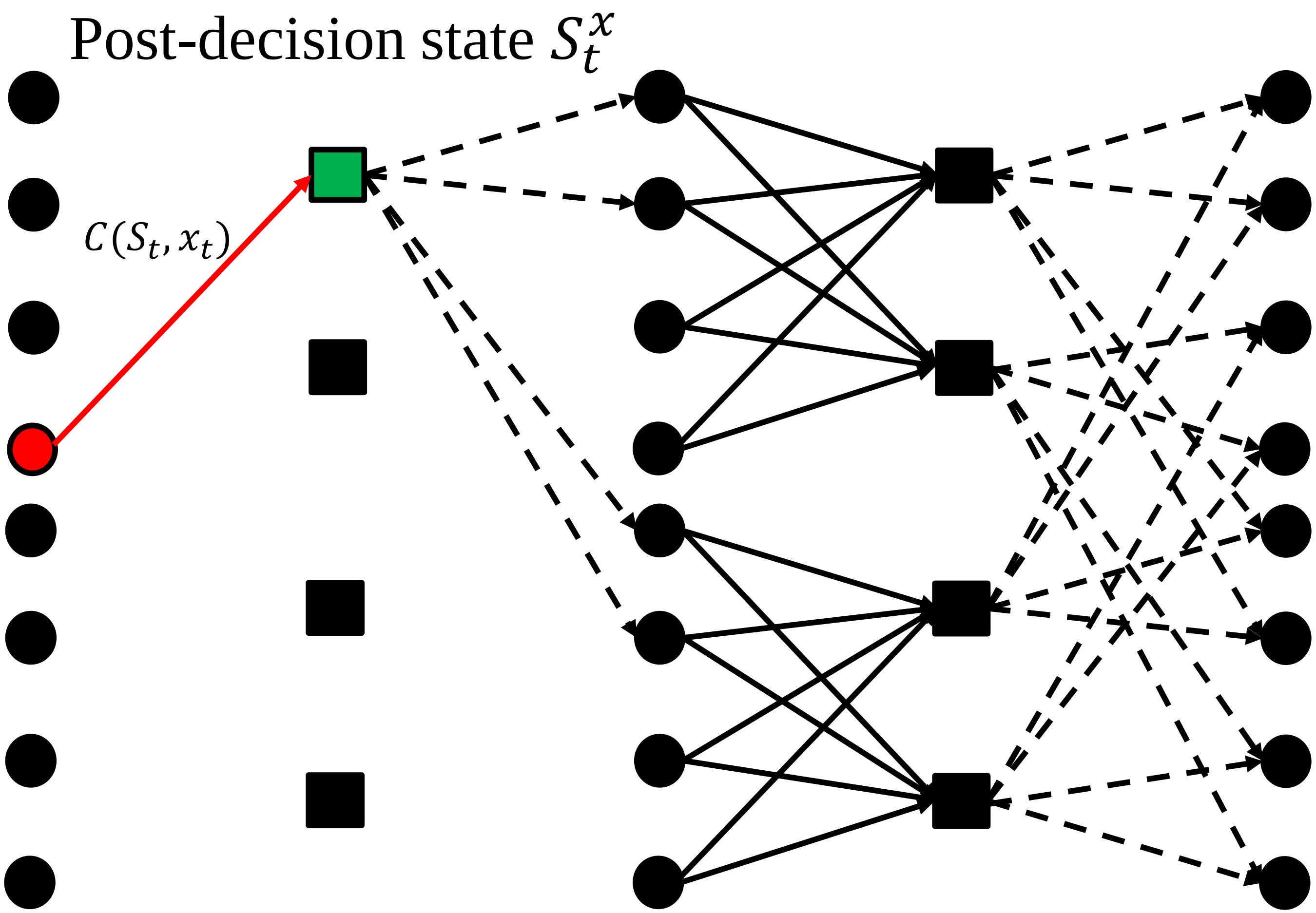
Post-decision state:
$S_t^x = S^{M,x}(S_t, x_t)$



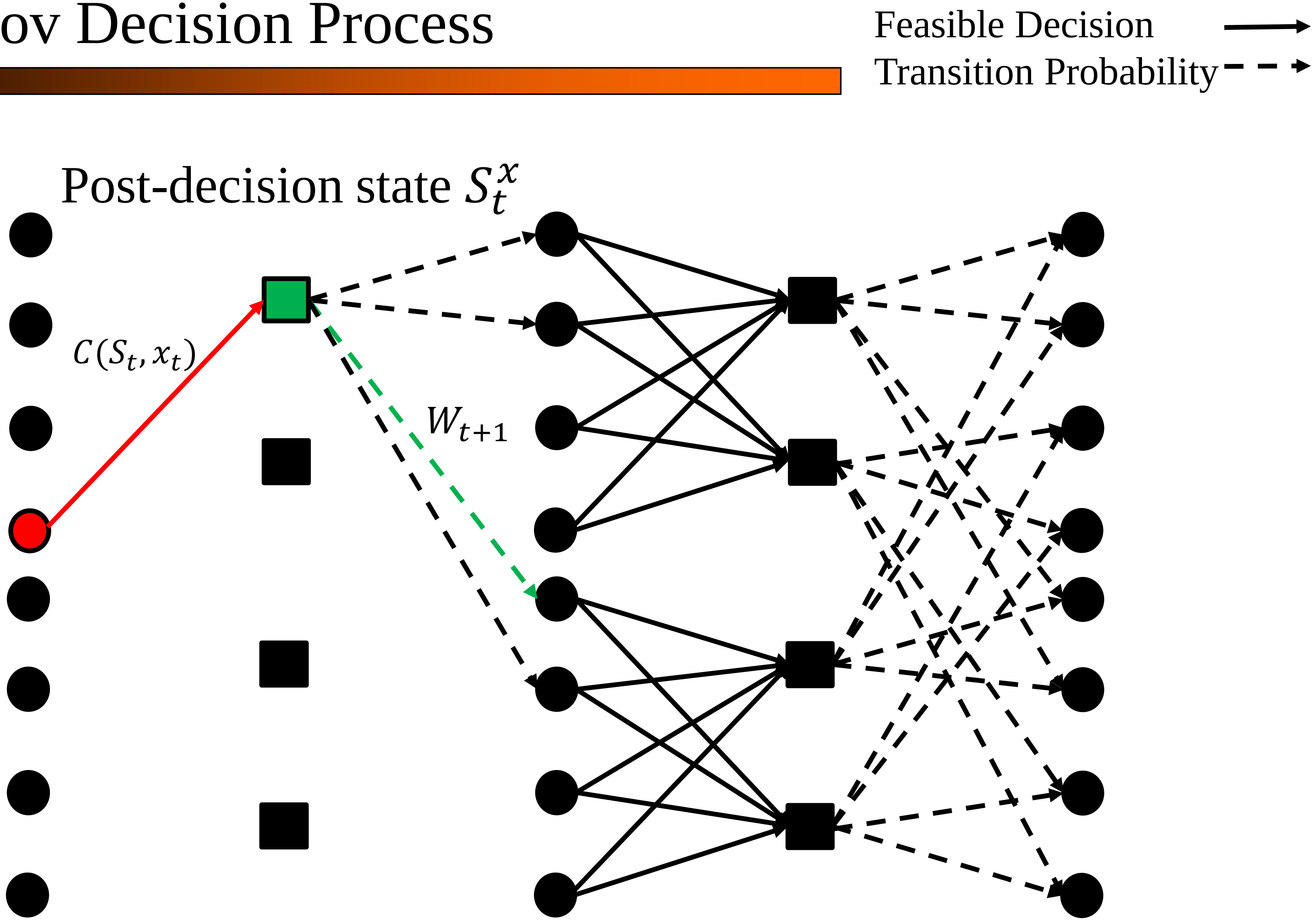
Formulating the Energy Storage Problem as a Markov Decision Process

Feasible Decision

Transition Probability



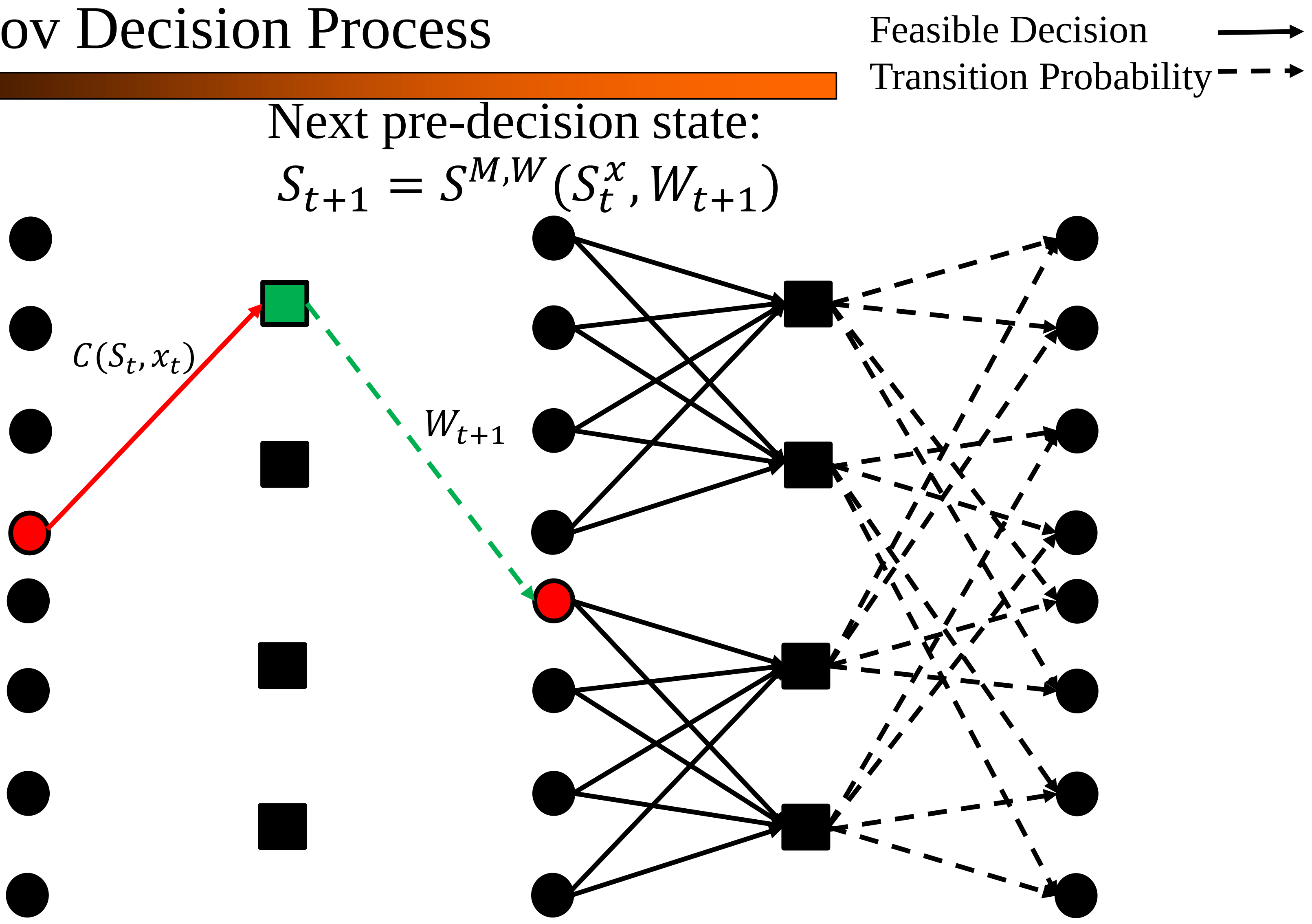
Formulating the Energy Storage Problem as a Markov Decision Process



Formulating the Energy Storage Problem as a Markov Decision Process

Next pre-decision state:

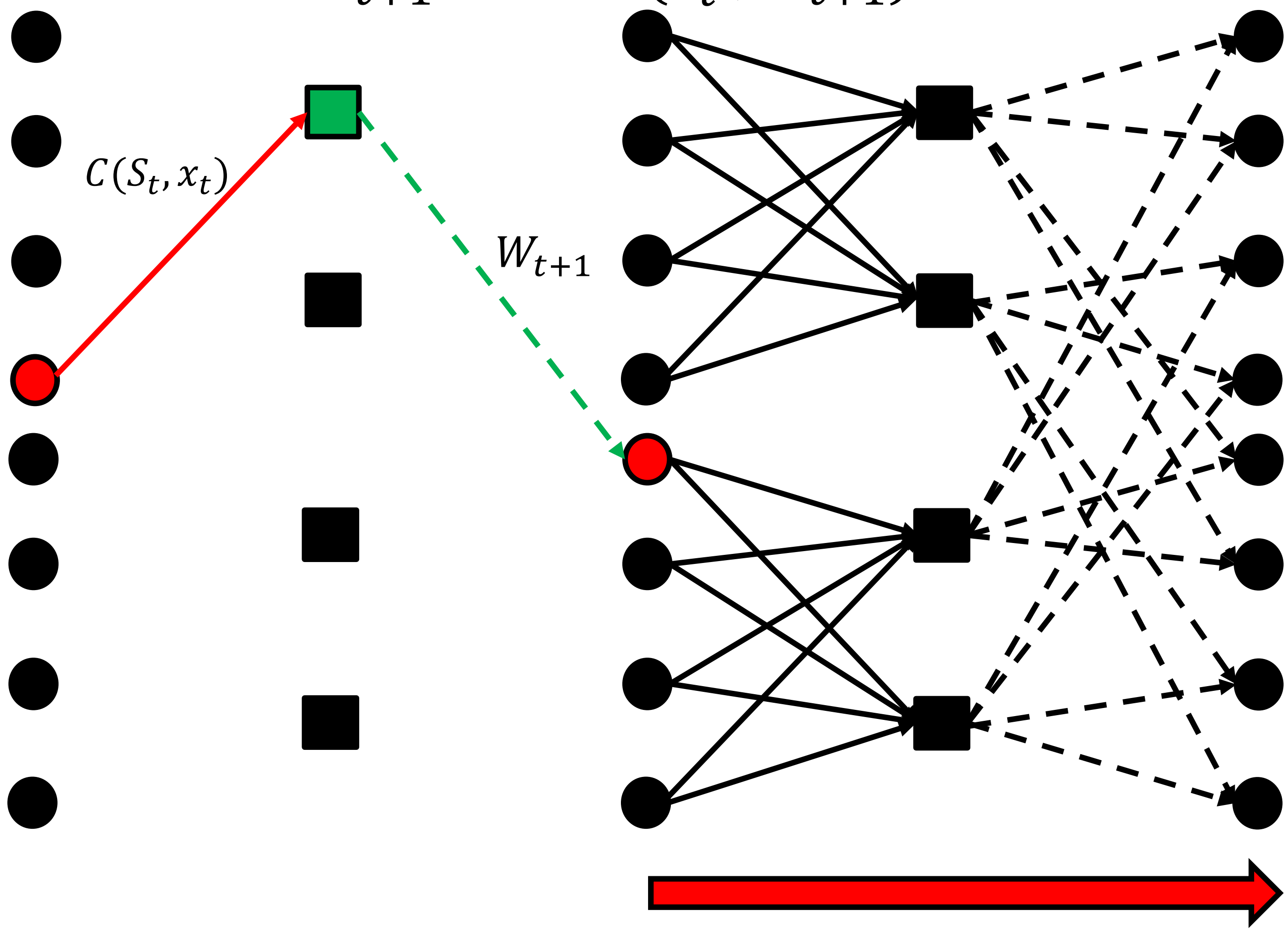
$$S_{t+1} = S^{M,W}(S_t^x, W_{t+1})$$



Formulating the Energy Storage Problem as a Markov Decision Process

Feasible Decision
Transition Probability

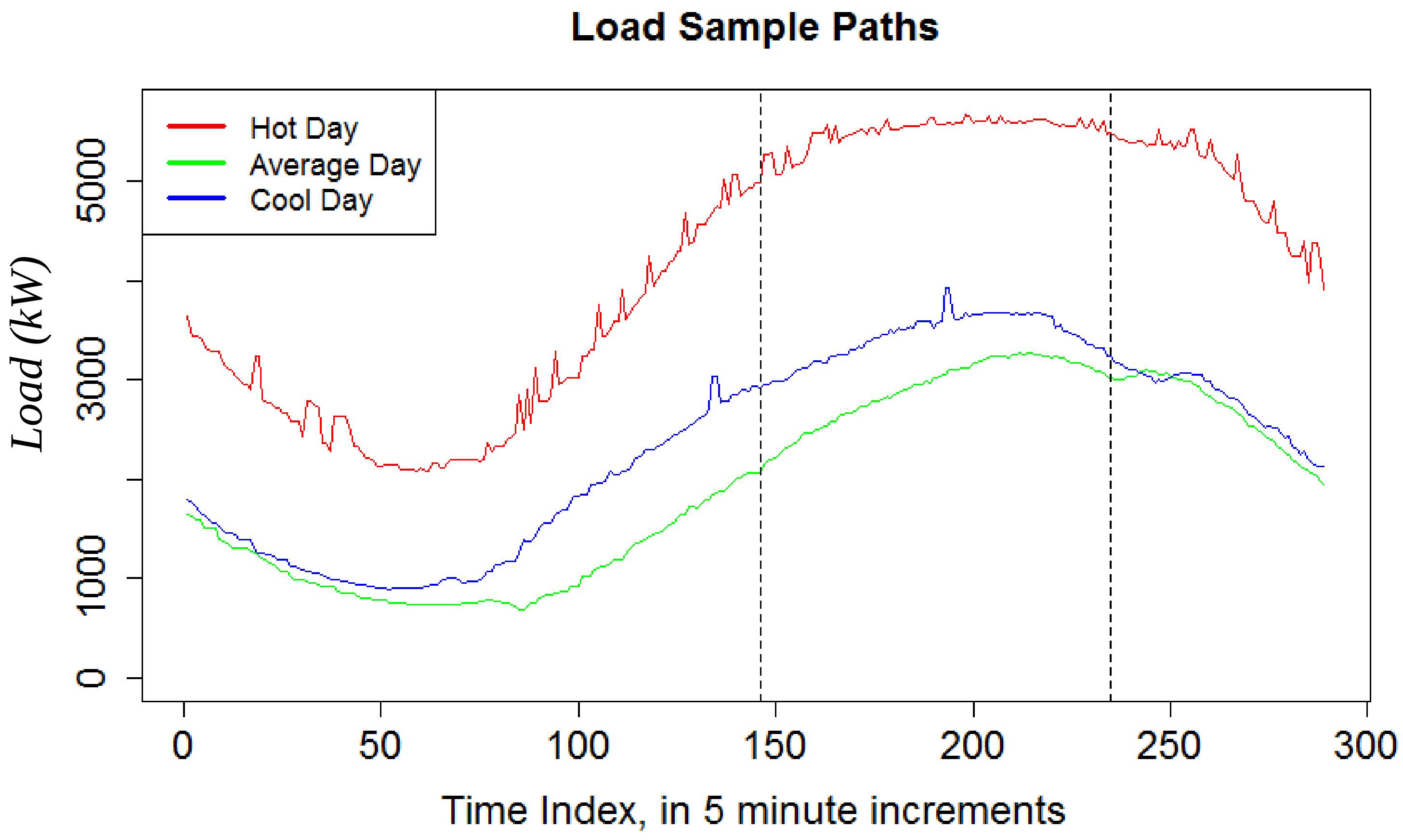
Next pre-decision state:
$S_{t+1} = S^{M,W}(S_t^x, W_{t+1})$



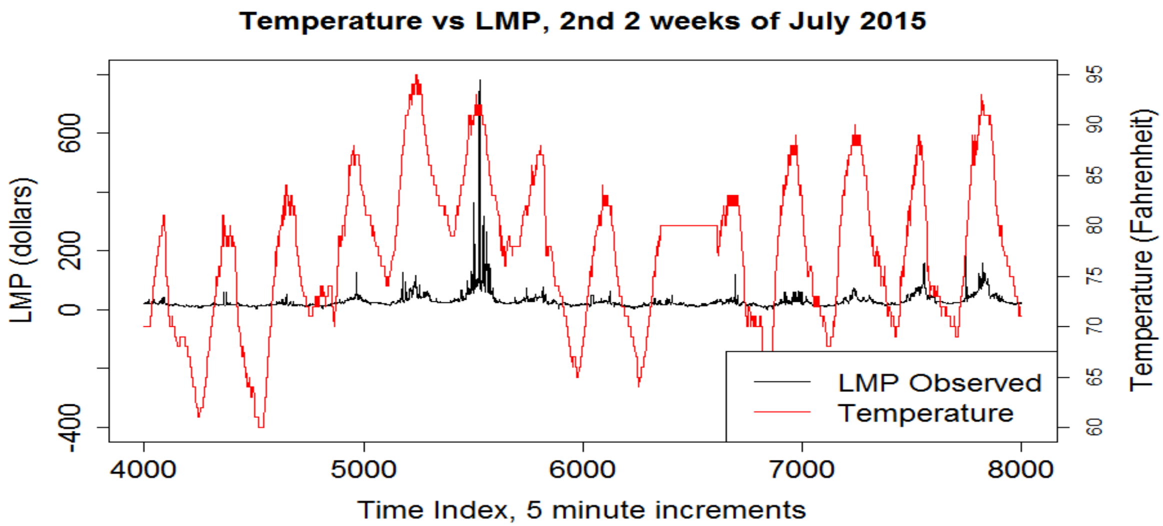
Modeling the Exogenous Processes

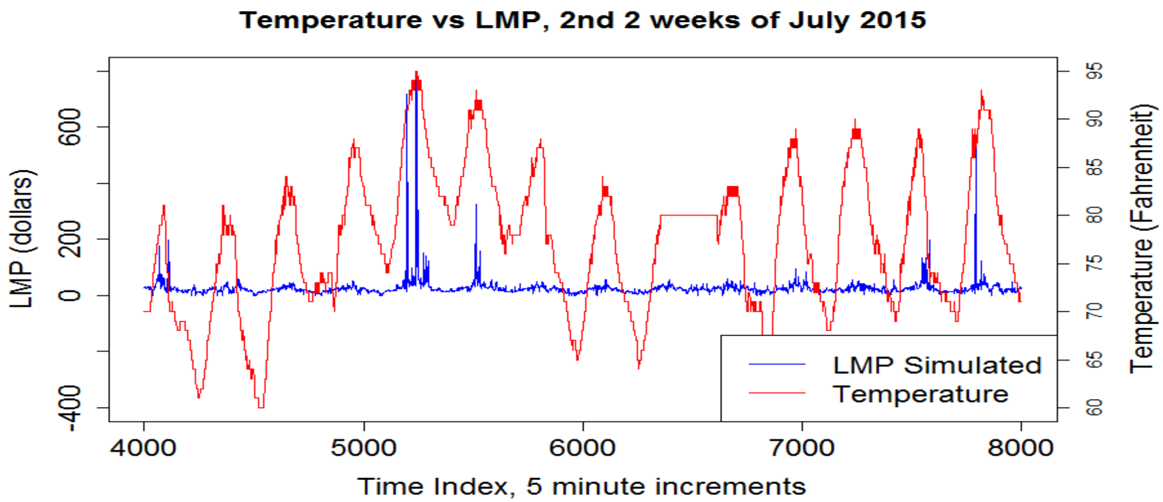
- Models determine our distribution of W_{t+1}
- Load L_t, Wind Energy E_t, Electricity Price P_t
  - » $L_t = f_t^L + \hat{L}_t$
  - » $E_t = f_t^E + \hat{E}_t$
  - » $P_t = f_t^P + \hat{P}_t$
- Of these, load exhibits far less deviation from forecast
- To reduce dimensionality of the problem, model load as a deterministic function
  - » $L_t = f_t^L \ \forall t \in \{0,1,\ldots,T\}$
- $\hat{E}_t, \hat{P}_t$ modeled with crossing state HSMM

Deterministic Load Profiles



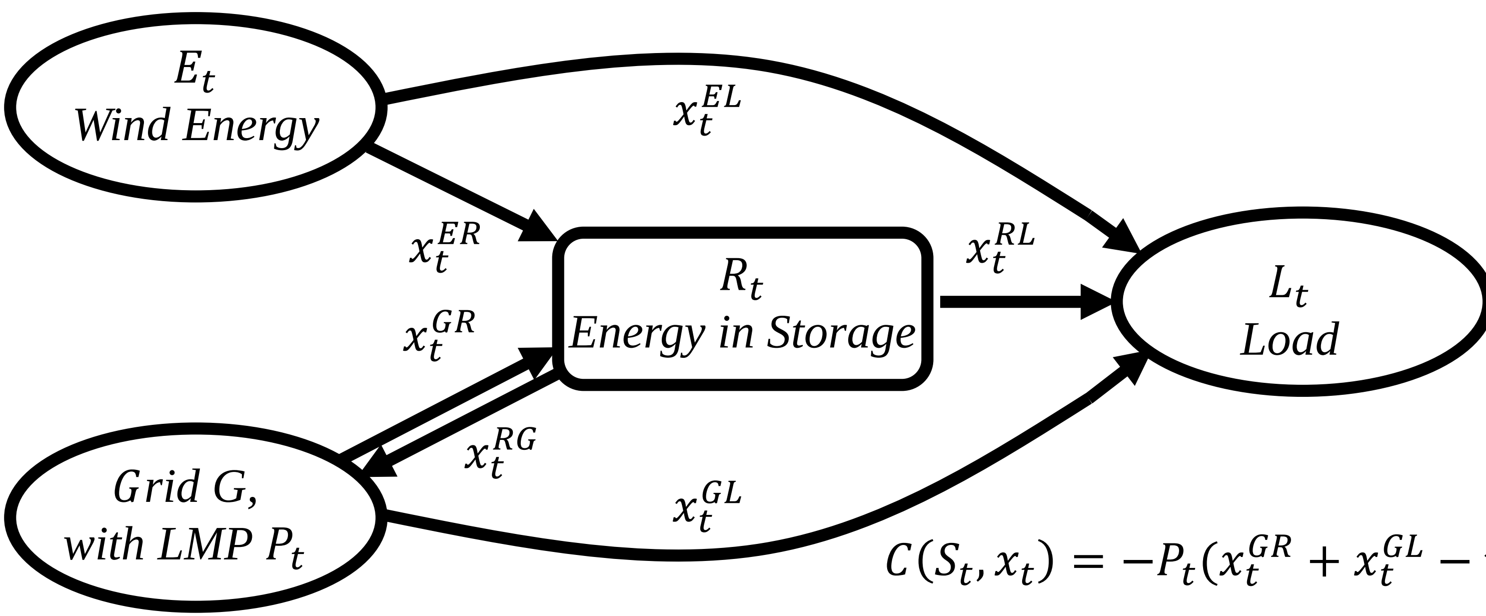
Electricity Prices





Formulating the Standard Energy Storage Problem as a Markov Decision Process

- State Variable:
  - » $S_t = \left(P_t, E_t, R_t, K_t^E, K_t^P\right)$, the pre-decision state
    - P_t, E_t necessary to determine constraints and costs at time t
  - » $S_t^x = \left(R_t^x, K_t^E, K_t^P\right)$, the post-decision state
    - K_t^E, K_t^P knowledge states about state of HSMM's
    - Belief about the distribution of E_{t+1}, P_{t+1}

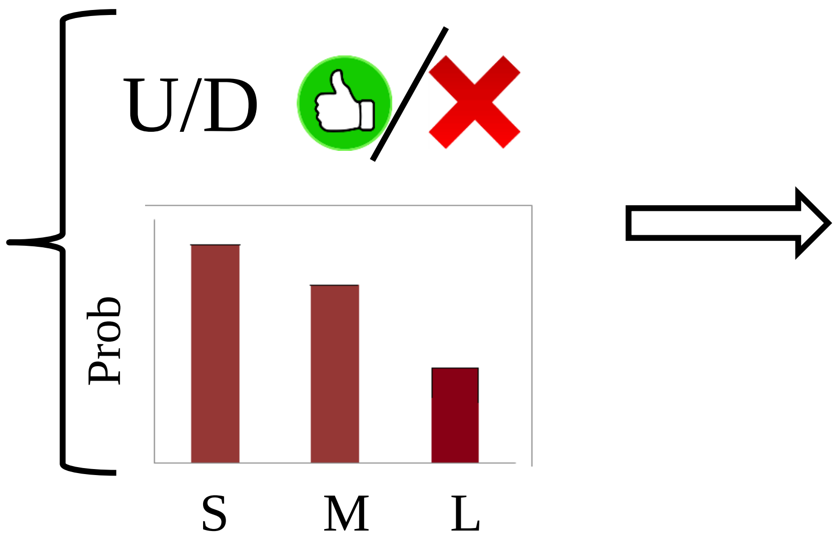


$C(S_t, x_t) = -P_t(x_t^{GR} + x_t^{GL} - \eta x_t^{RG})$

Formulating the Standard Energy Storage Problem as a Markov Decision Process

- State Variable:
  - » $S_t = \left(P_t, E_t, R_t, K_t^E, K_t^P\right)$, the pre-decision state
    - P_t, E_t necessary to determine constraints and costs at time t
  - » $S_t^x = \left(R_t^x, K_t^E, K_t^P\right)$, the post-decision state
    - K_t^E, K_t^P knowledge states about state of HSMM's
    - Belief about the distribution of E_{t+1}, P_{t+1}

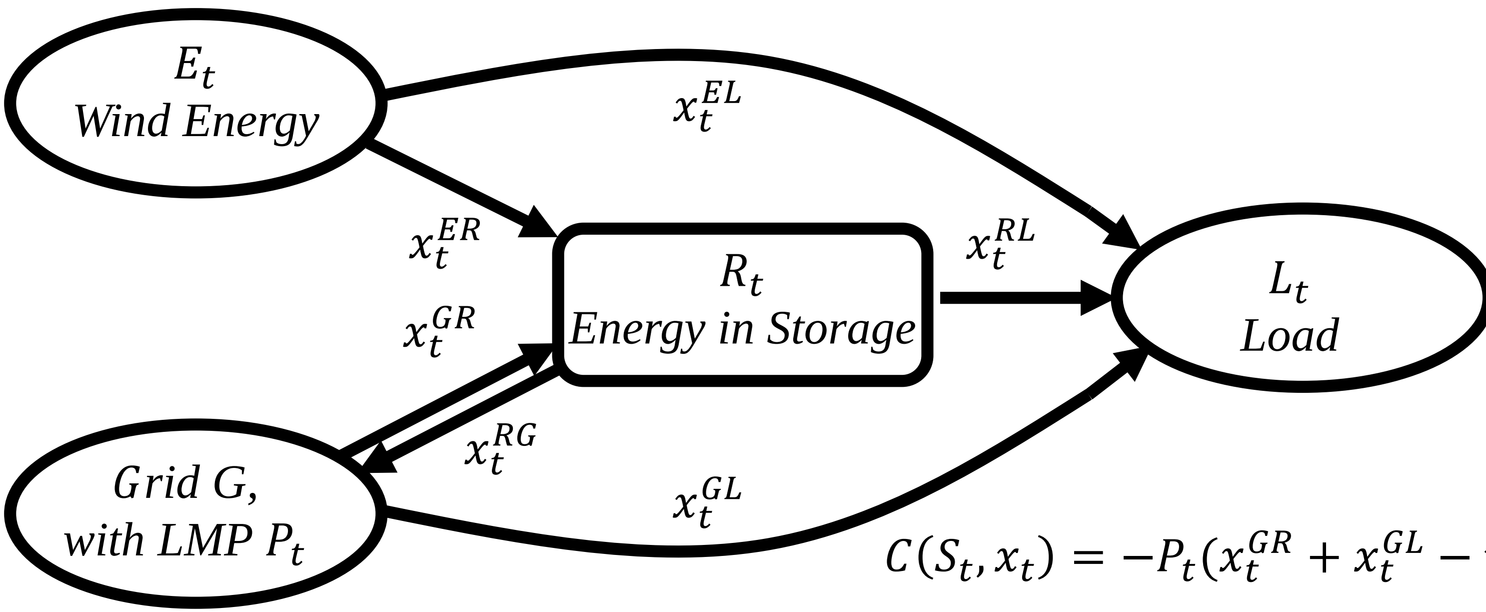
- K_t^E:



$P\left(\hat{E}_{t+1} \middle| K_t^E\right)$

Formulating the Standard Energy Storage Problem as a Markov Decision Process

- Transition Functions:
  - » $S_t^x = S^{M,x}(S_t, x_t)$ given by:
    - $R_t^x = R_t + \eta\left(x_t^{GR} + x_t^{ER}\right) - x_t^{RL} - x_t^{RG}$
    - E_t, P_t dropped from $S_t \rightarrow S_t^x$

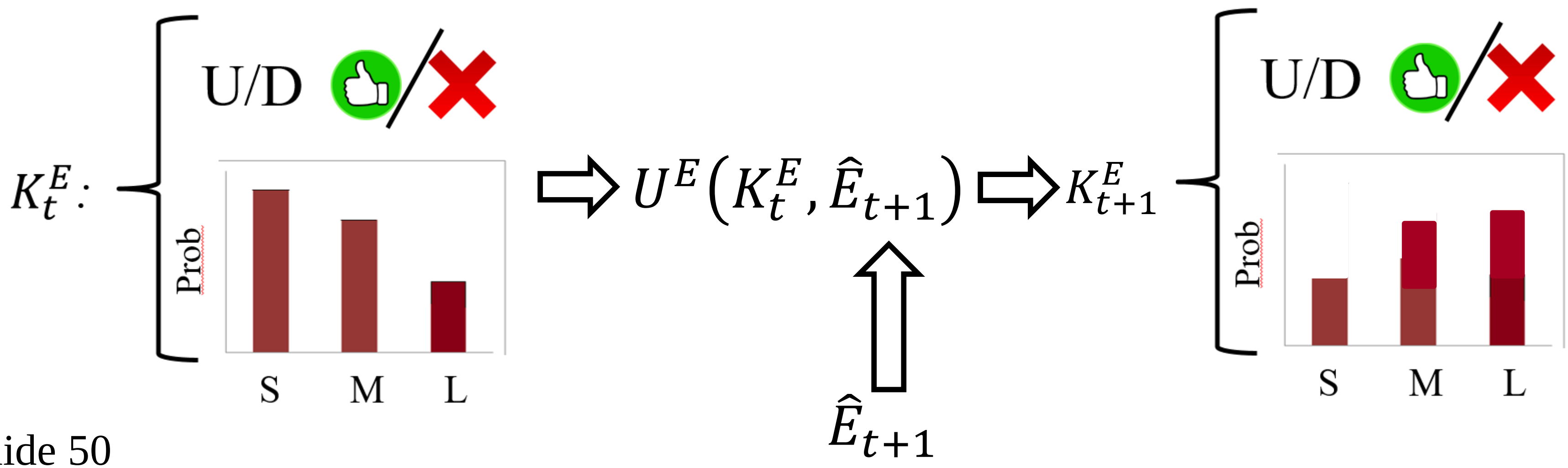


$C(S_t, x_t) = -P_t(x_t^{GR} + x_t^{GL} - \eta x_t^{RG})$

Formulating the Standard Energy Storage Problem as a Markov Decision Process

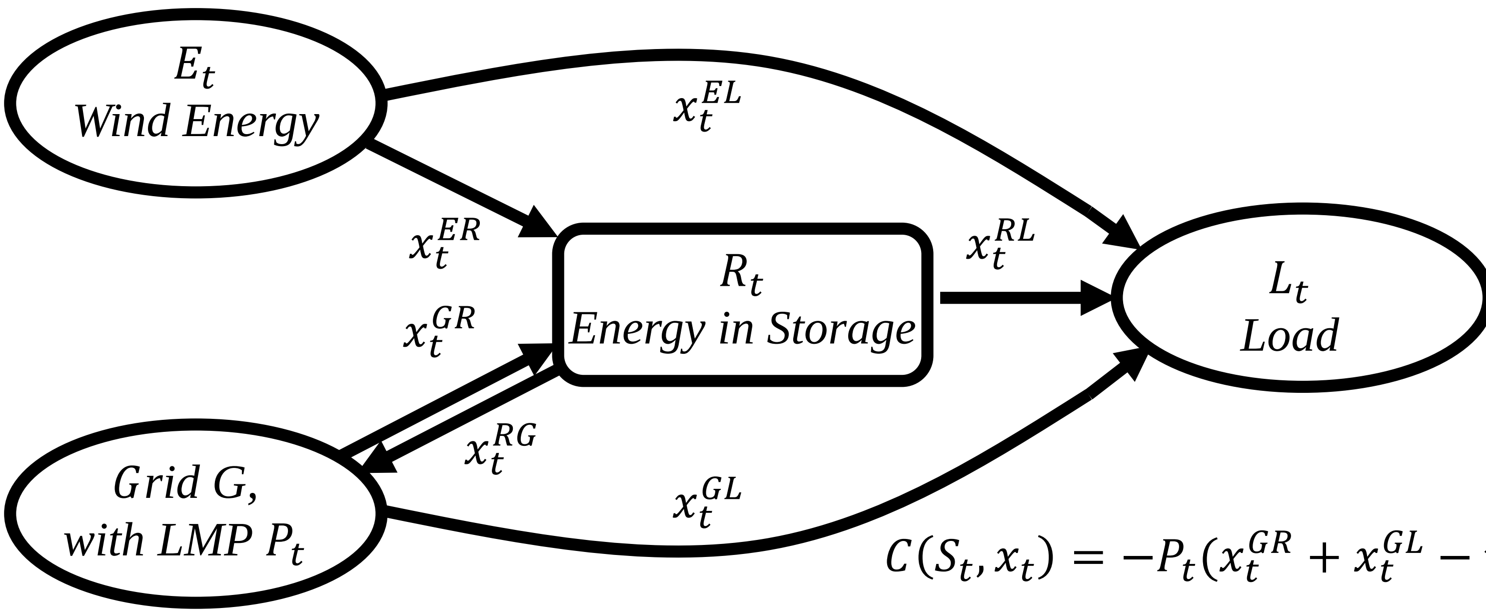
- Transition Functions:
  - » $S_{t+1} = S^{M,W}(S_t^x, W_{t+1})$:
    - $R_{t+1} = R_t^x$
    - $E_{t+1} = f_{t+1}^E + \hat{E}_{t+1}$
    - $P_{t+1} = f_{t+1}^P + \hat{P}_{t+1}$
    - $K_{t+1}^E = U^E\left(K_t^E, \hat{E}_{t+1}\right)$
    - $K_{t+1}^P = U^P\left(K_t^P, \hat{P}_{t+1}\right)$

Bayesian updating formulas given current knowledge state and observed errors



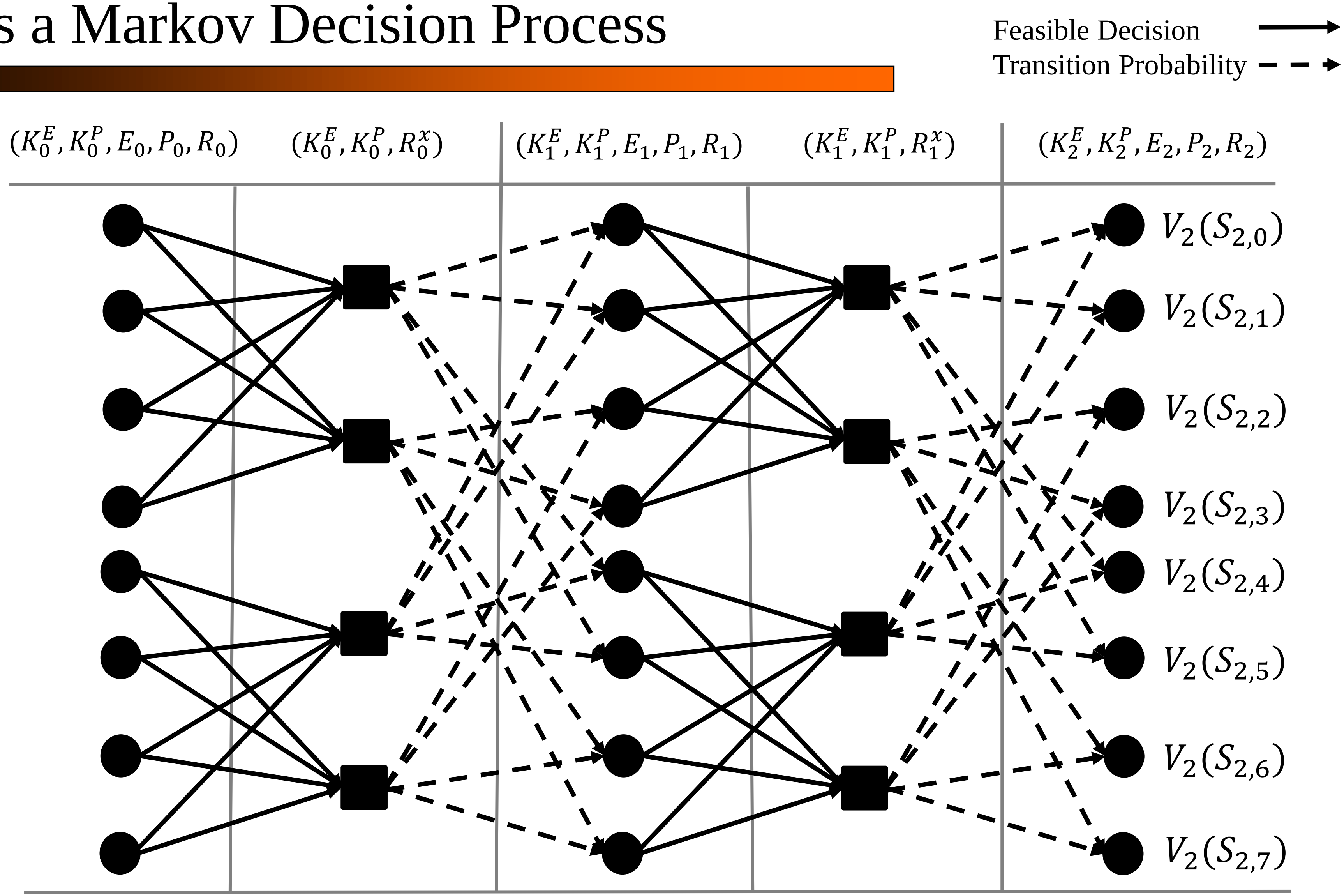
Formulating the Standard Energy Storage Problem as a Markov Decision Process

- Objective Function:
  - » $\max_{\pi \in \Pi} \mathbb{E}^{\pi}\left[\sum_{t=0}^{T} C\left(S_t, X^{\pi}(S_t)\right) | S_0\right]$ where
  - » $C(S_t, x_t) = -P_t(x_t^{GR} + x_t^{GL} - \eta x_t^{RG})$ and
  - » $S_t^x = S^{M,x}\left(S_t, X_t^{\pi}(S_t)\right)$
  - » $S_{t+1} = S^{M,W}(S_t^x, W_{t+1})$



$C(S_t, x_t) = -P_t(x_t^{GR} + x_t^{GL} - \eta x_t^{RG})$

Formulating the Standard Energy Storage Problem as a Markov Decision Process



$S_0 \rightarrow S^{M,x}(S_0, x_0\ ) \rightarrow S_0^x \rightarrow S^{M,W}(S_0^x, W_1) \rightarrow S_1 \rightarrow S^{M,x}(S_1, x_1\ ) \rightarrow S_1^x \rightarrow S^{M,W}(S_1^x, W_2) \rightarrow S_2$

Outline

- Motivation
- Hidden Semi-Markov Crossing State Model
- Formulating the Energy Storage Problem as a Markov Decision Process
- Backward Approximate Dynamic Programming
- Numerical Results

Standard Backward Dynamic Programming

- $\max_{\pi \in \Pi} \mathbb{E}^{\pi}\left[\sum_{t=0}^{T} C\left(S_t, X^{\pi}(S_t)\right) | S_0\right]$
- Value functions, or contribution-to-go functions, $V_t^*(S_t)$, are given by Bellman's equation for finite horizon problems:

$$V_t^*(S_t) = \max_{x_t \in \mathcal{X}_t} \left(C(S_t, x_t) + \mathbb{E}[V_{t+1}^*(S_{t+1}) | S_t, x_t]\right)$$

- Once value functions are found, the optimal policy:

$$X_t^*(S_t) = \arg\max_{x_t \in \mathcal{X}_t} \left(C(S_t, x_t) + \mathbb{E}[V_{t+1}^*(S_{t+1}) | S_t, x_t]\right)$$

maximizes the one-step contribution plus the expected value of the downstream state.

Standard Backward Dynamic Programming

- Alternatively, we can find the value of each post-decision state:

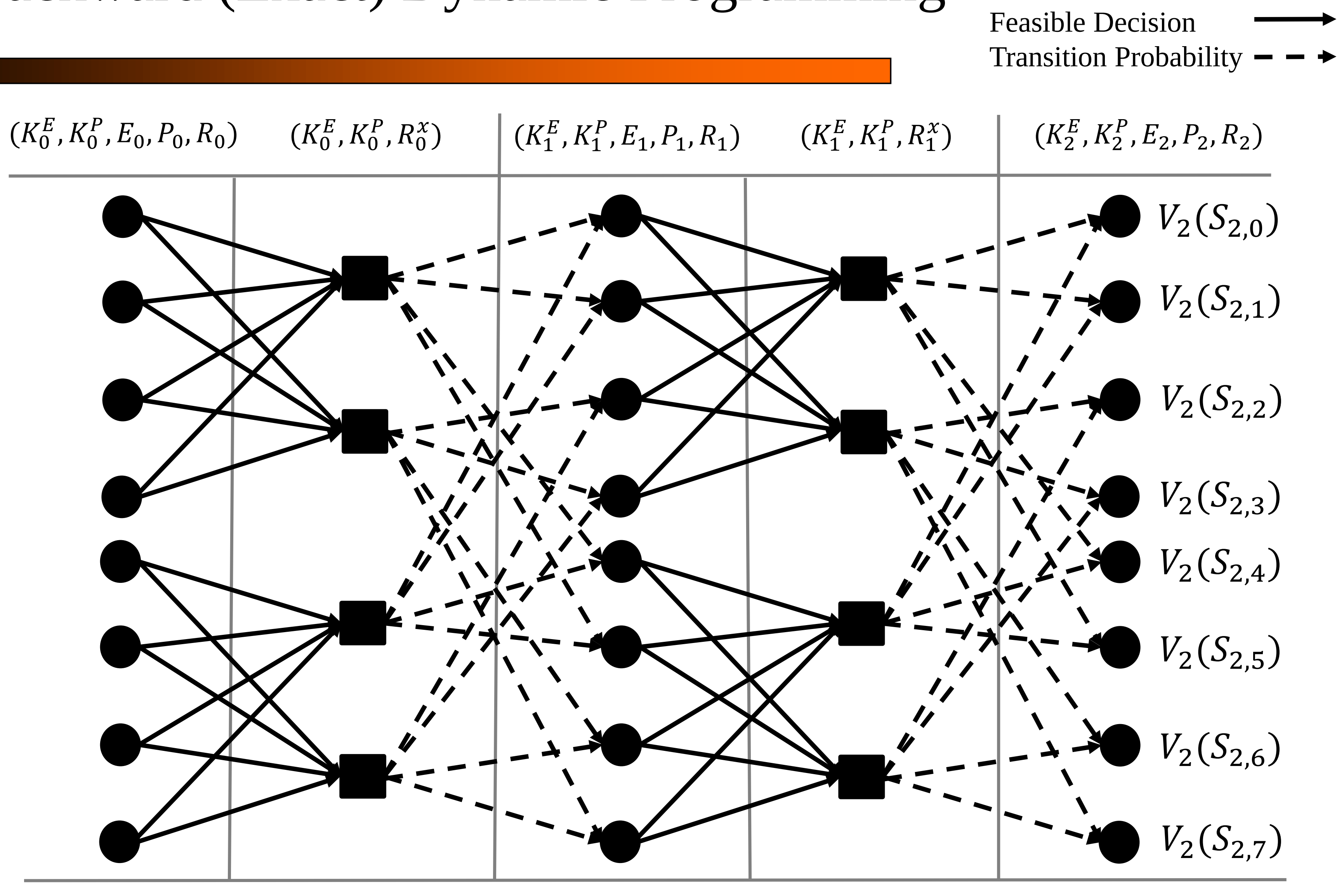
$$V_t^{x,*}(S_t^x) = \mathbb{E}[V_{t+1}^*(S_{t+1})|S_t^x\ ]$$

- And, stepping backwards, for each pre-decision state:

$$V_t^*(S_t) = \underset{x_t \in \mathcal{X}_t}{max}\left(C(S_t, x_t) + V_t^{x,*}(S_t^x)\right)$$

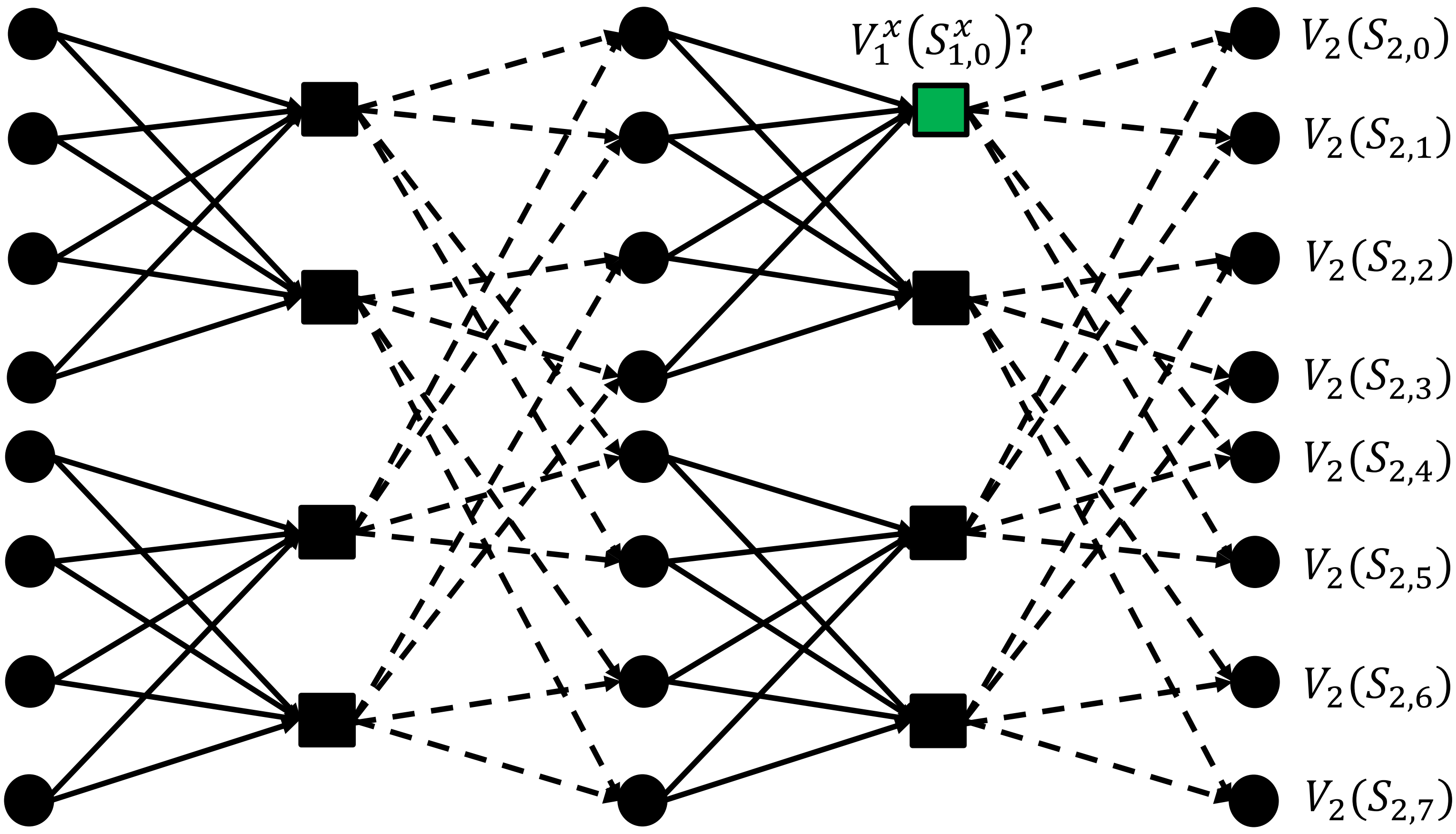
- Once post-decision state value functions are found, the optimal policy: $X_t^*(S_t) = \underset{x_t \in \mathcal{X}_t}{\arg\max}\left(C(S_t, x_t) + V_t^{x,*}(S_t^x)\right)$
- This removes the expectation from the policy.

Backward (Exact) Dynamic Programming



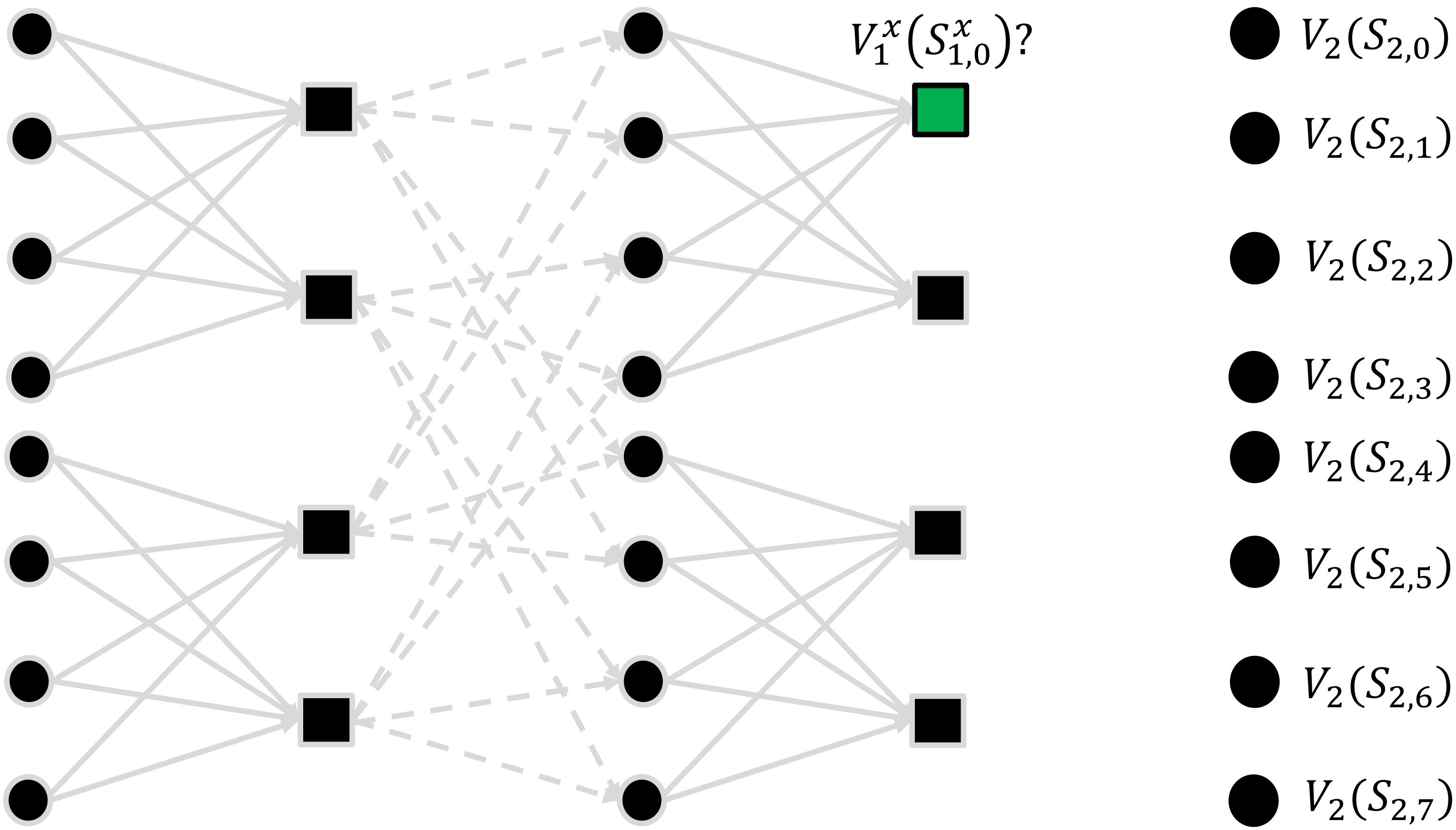
$S_0 \to S^{M,x}(S_0, x_0) \to S_0^x \to S^{M,W}(S_0^x, W_1) \to S_1 \to S^{M,x}(S_1, x_1) \to S_1^x \to S^{M,W}(S_1^x, W_2) \to S_2$

Backward (Exact) Dynamic Programming



$$S_0 \rightarrow S^{M,x}(S_0, x_0) \rightarrow S_0^x \rightarrow S^{M,W}(S_0^x, W_1) \rightarrow S_1 \rightarrow S^{M,x}(S_1, x_1) \rightarrow S_1^x \rightarrow S^{M,W}(S_1^x, W_2) \rightarrow S_2$$

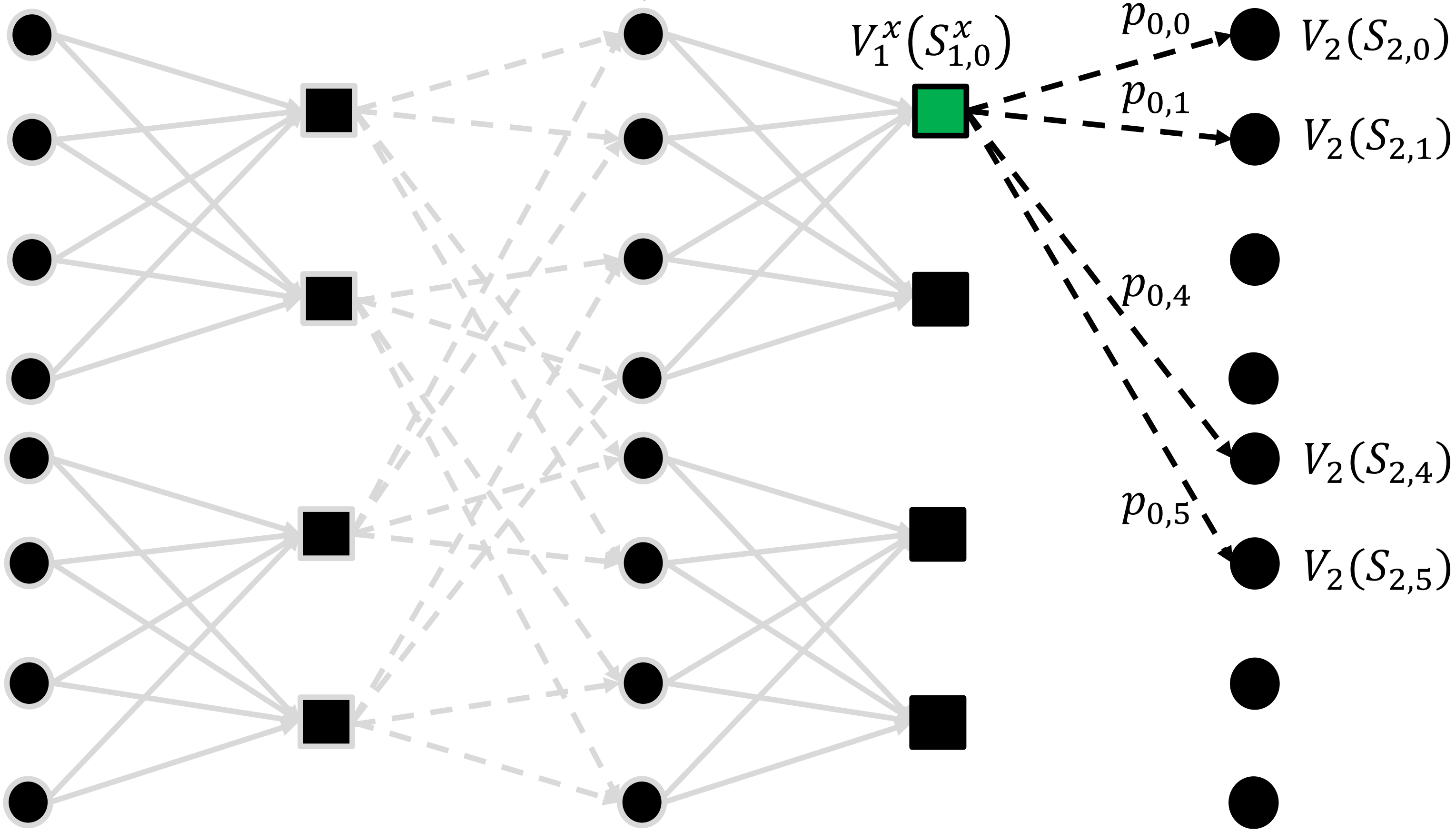
Backward (Exact) Dynamic Programming



$S_0 \rightarrow S^{M,x}(S_0, x_0) \rightarrow S_0^x \rightarrow S^{M,W}(S_0^x, W_1) \rightarrow S_1 \rightarrow S^{M,x}(S_1, x_1) \rightarrow S_1^x \rightarrow S^{M,W}(S_1^x, W_2) \rightarrow S_2$

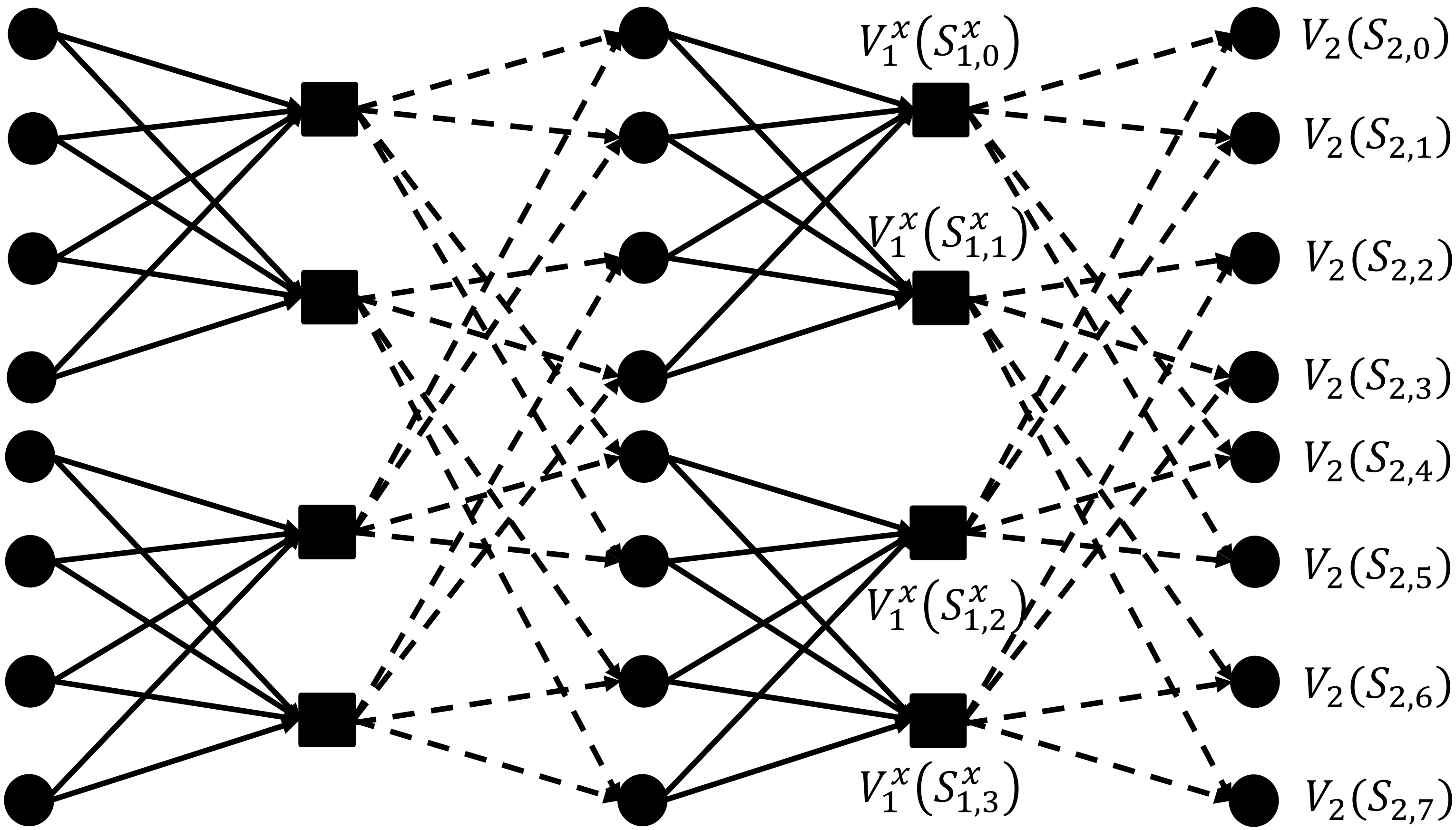
Backward (Exact) Dynamic Programming

$$V_1^x(S_{1,0}^x) = \sum_i p_{0,i} V_2(S_{2,i})$$



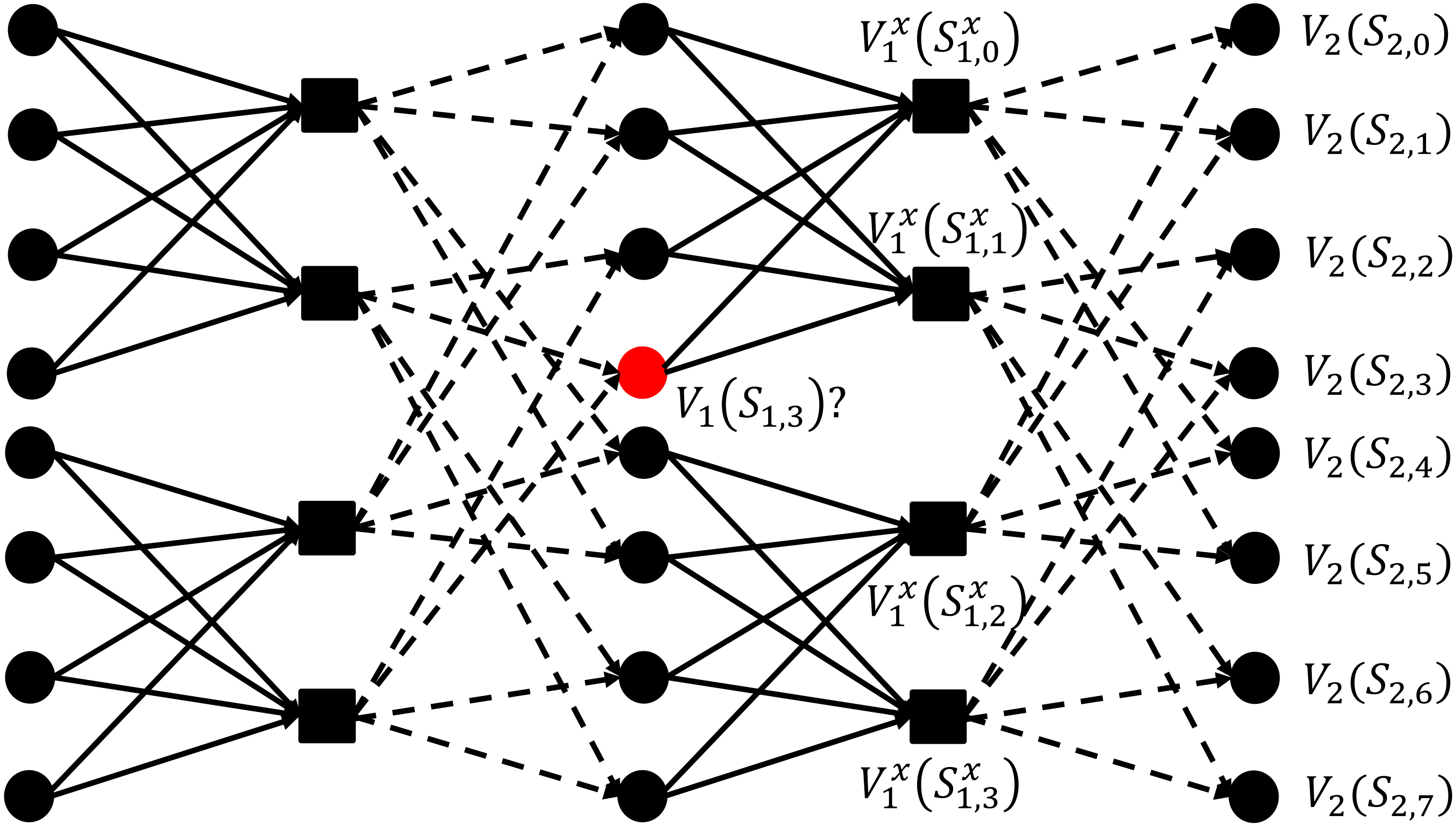
$$S_0 \to S^{M,x}(S_0, x_0) \to S_0^x \to S^{M,W}(S_0^x, W_1) \to S_1 \to S^{M,x}(S_1, x_1) \to S_1^x \to S^{M,W}(S_1^x, W_2) \to S_2$$

Backward (Exact) Dynamic Programming



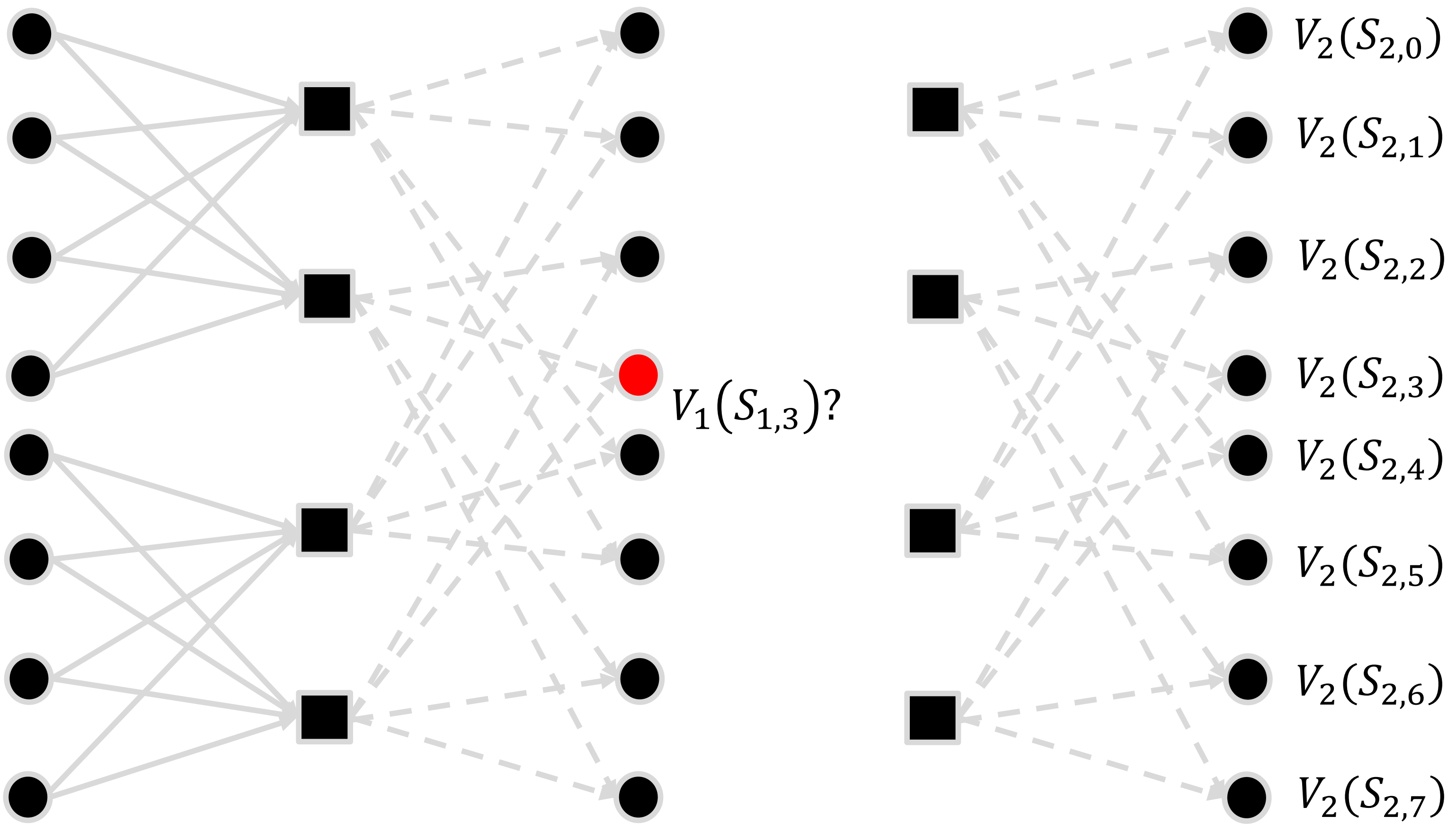
$S_0 \rightarrow S^{M,x}(S_0, x_0) \rightarrow S_0^x \rightarrow S^{M,W}(S_0^x, W_1) \rightarrow S_1 \rightarrow S^{M,x}(S_1, x_1) \rightarrow S_1^x \rightarrow S^{M,W}(S_1^x, W_2) \rightarrow S_2$

Backward (Exact) Dynamic Programming



$S_0 \rightarrow S^{M,x}(S_0, x_0) \rightarrow S_0^x \rightarrow S^{M,W}(S_0^x, W_1) \rightarrow S_1 \rightarrow S^{M,x}(S_1, x_1) \rightarrow S_1^x \rightarrow S^{M,W}(S_1^x, W_2) \rightarrow S_2$

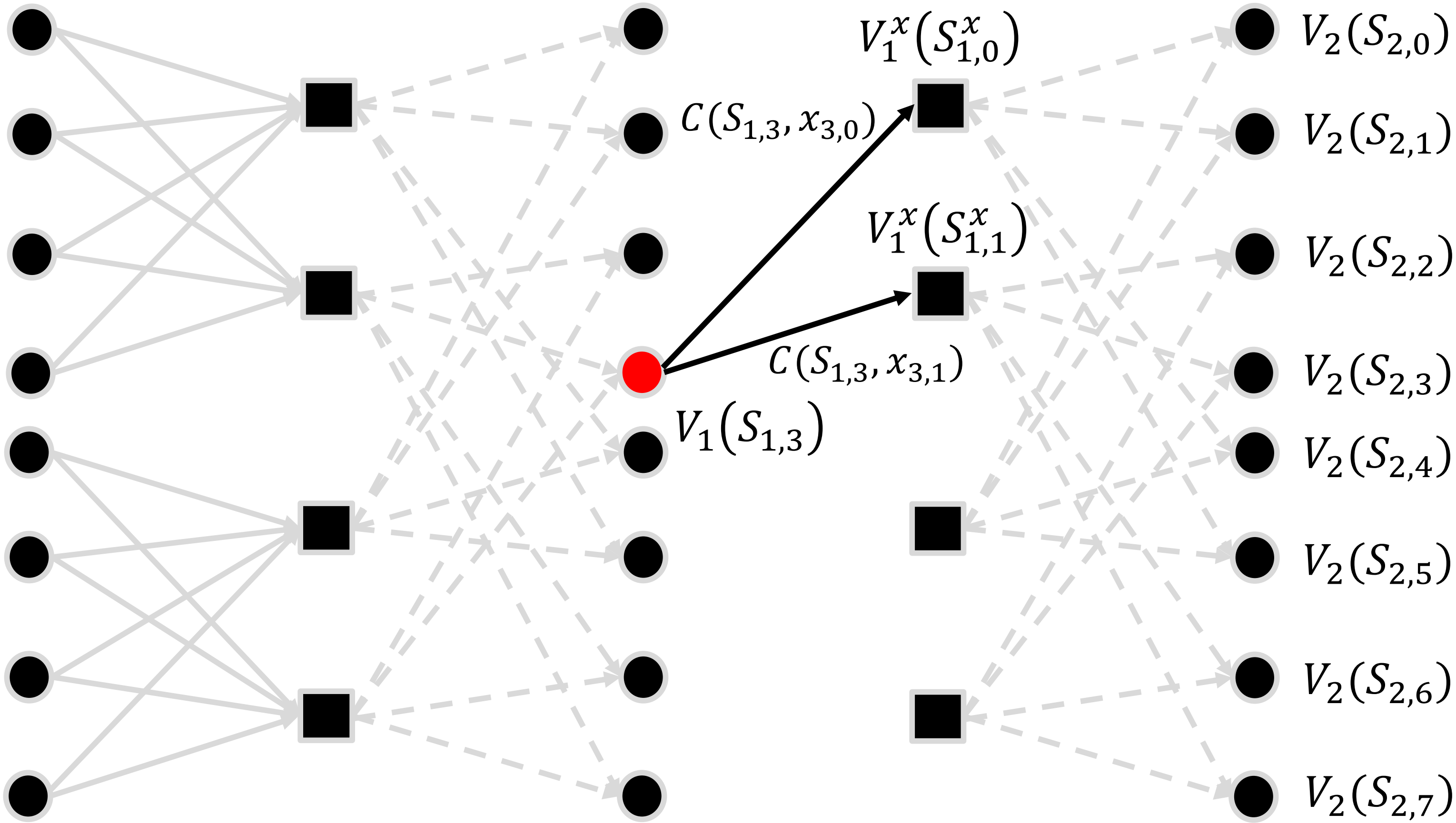
Backward (Exact) Dynamic Programming



$$S_0 \rightarrow S^{M,x}(S_0, x_0) \rightarrow S_0^x \rightarrow S^{M,W}(S_0^x, W_1) \rightarrow S_1 \rightarrow S^{M,x}(S_1, x_1) \rightarrow S_1^x \rightarrow S^{M,W}(S_1^x, W_2) \rightarrow S_2$$

Backward (Exact) Dynamic Programming

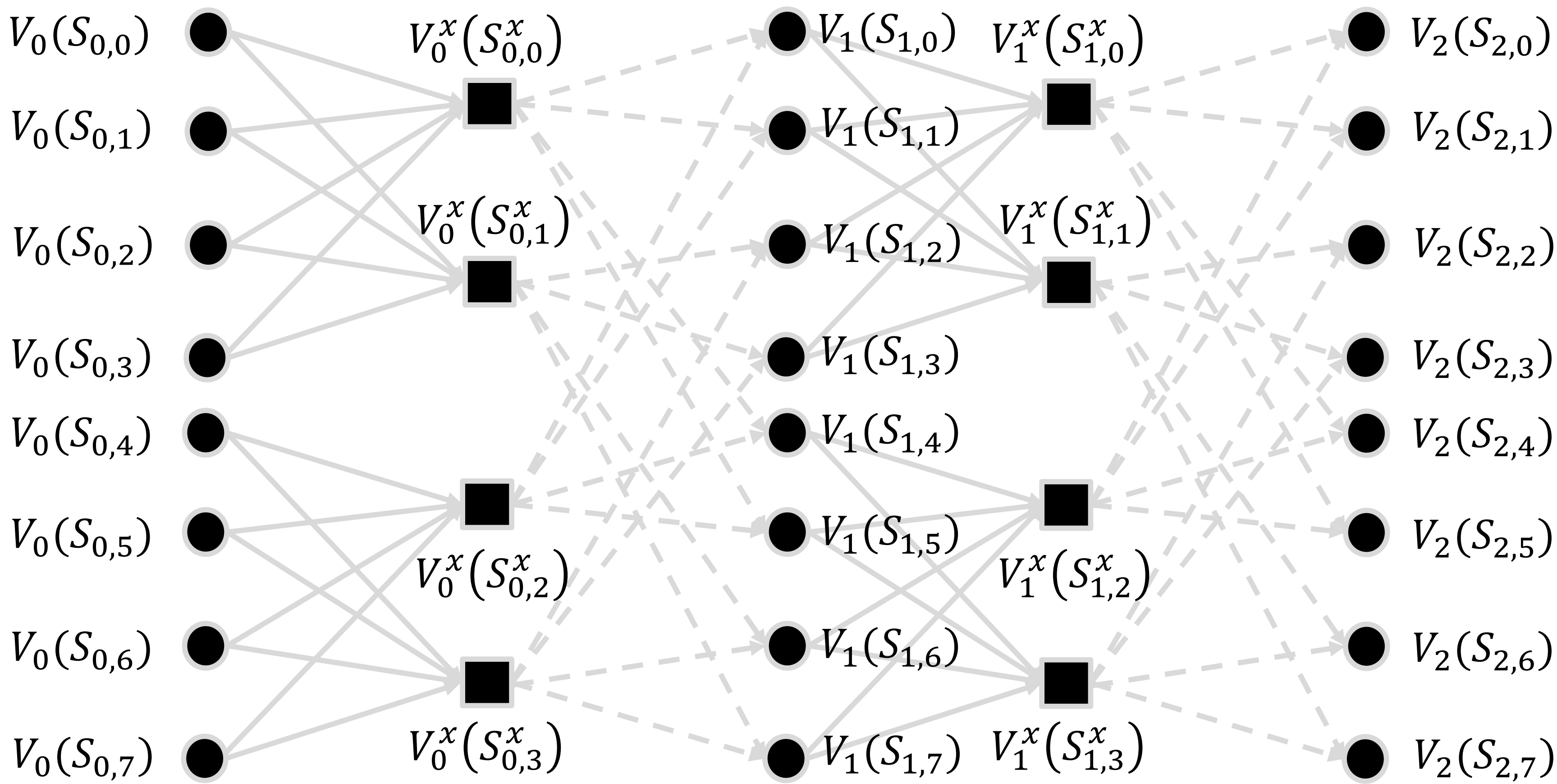
$$V_1(S_{1,3}) = \max_{x_{3,i}} \left(C(S_{1,3}, x_{3,i}) + V_1^x(S_{1,i}^x) \right)$$



$$S_0 \rightarrow S^{M,x}(S_0, x_0) \rightarrow S_0^x \rightarrow S^{M,W}(S_0^x, W_1) \rightarrow S_1 \rightarrow S^{M,x}(S_1, x_1) \rightarrow S_1^x \rightarrow S^{M,W}(S_1^x, W_2) \rightarrow S_2$$

Backward (Exact) Dynamic Programming

$$\text{Optimal Policy: } X_t^*(S_t) = \arg\max_{x_t \in \mathcal{X}_t} \left(C(S_t, x_t) + V_t^{x,*}(S_t^x) \right)$$



$$S_0 \to S^{M,x}(S_0, x_0) \to S_0^x \to S^{M,W}(S_0^x, W_1) \to S_1 \to S^{M,x}(S_1, x_1) \to S_1^x \to S^{M,W}(S_1^x, W_2) \to S_2$$

Approximate Dynamic Programming

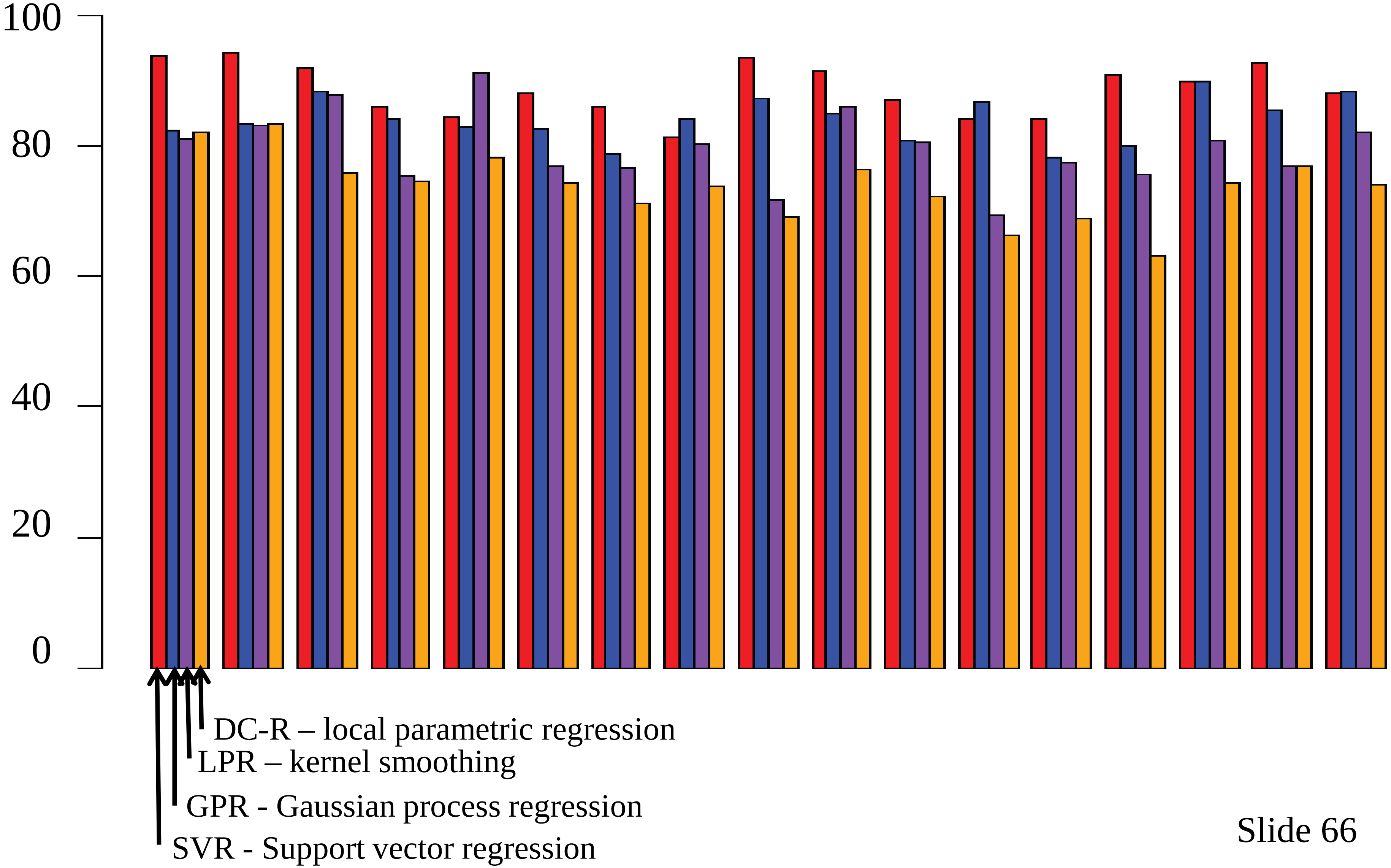
- In cases where performing a full backward pass cannot be done, we can instead rely on value function ***approximations*** (VFA's) and a VFA-based policy:

$$X_t^{\pi}(S_t) = \arg\max_{x_t \in \mathcal{X}_t} \left(C(S_t, x_t) + \mathbb{E}\left[\bar{V}_{t+1}(S_{t+1}) \middle| S_t, x_t \right] \right)$$

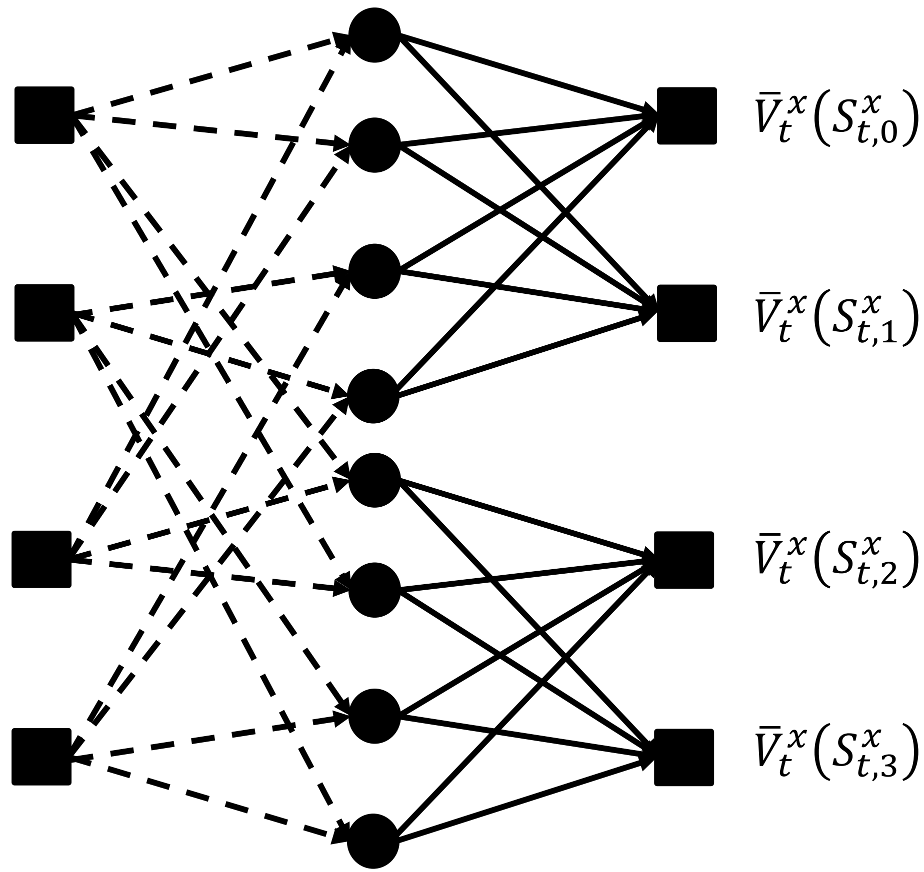
- Alternatively, we can fit value functions ***approximations*** to post-decision states instead, giving the policy:

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

Forward ADP in Energy Storage Applications



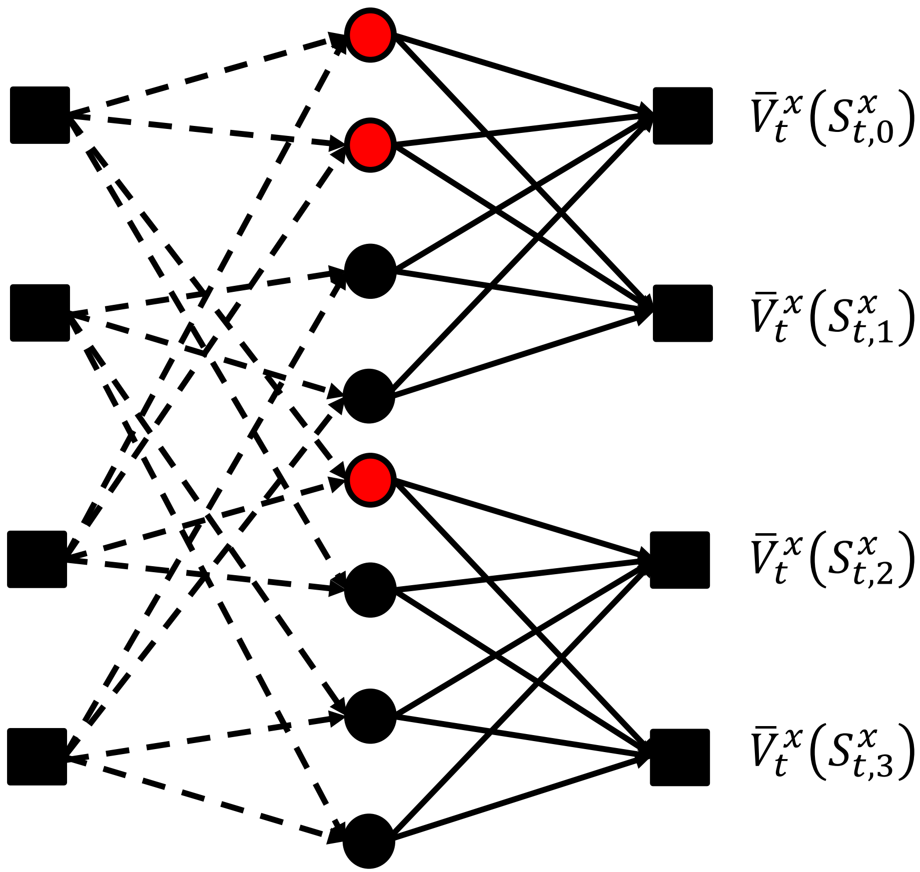
Backward ADP I: Lookup Table form for Post-Decision States



$$S_{t-1}^x \rightarrow S^{M,W}(S_{t-1}^x, W_t) \rightarrow S_t \rightarrow S^{M,x}(S_t, x_t) \rightarrow S_t^x$$

Backward ADP I: Lookup Table form for Post-Decision States

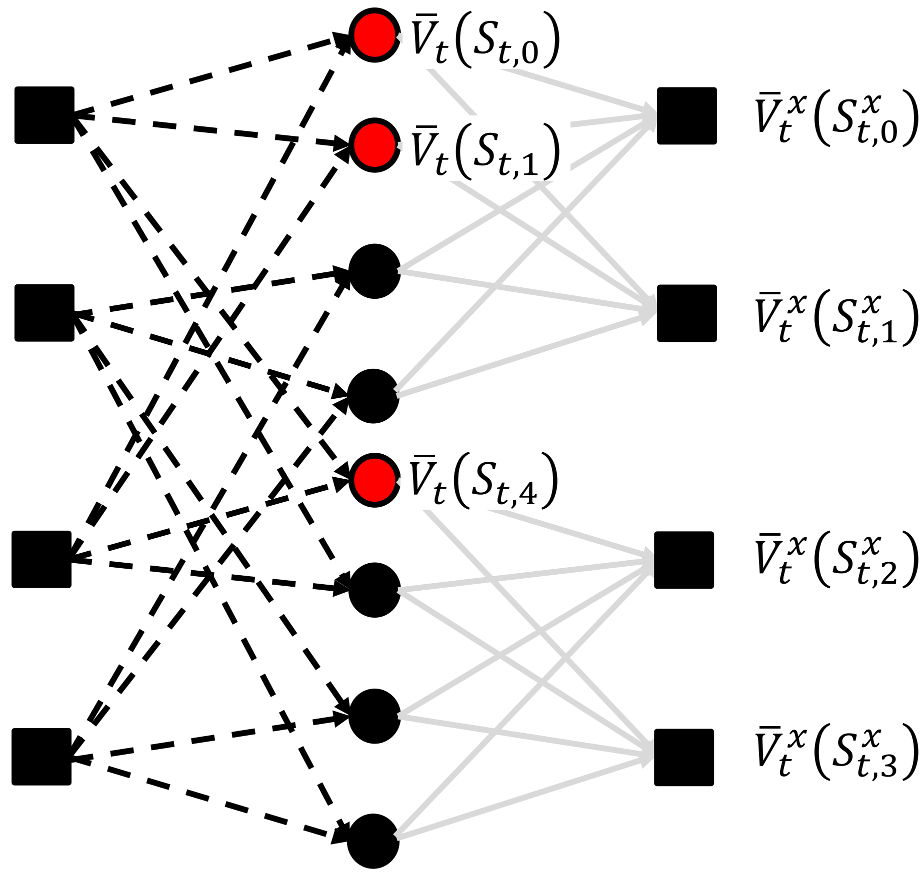
1. Sample pre-decision states and find their values by maximizing over feasible decisions.



$S_{t-1}^x \rightarrow S^{M,W}(S_{t-1}^x, W_t) \rightarrow S_t \rightarrow S^{M,x}(S_t, x_t) \rightarrow S_t^x$

Backward ADP I: Lookup Table form for Post-Decision States

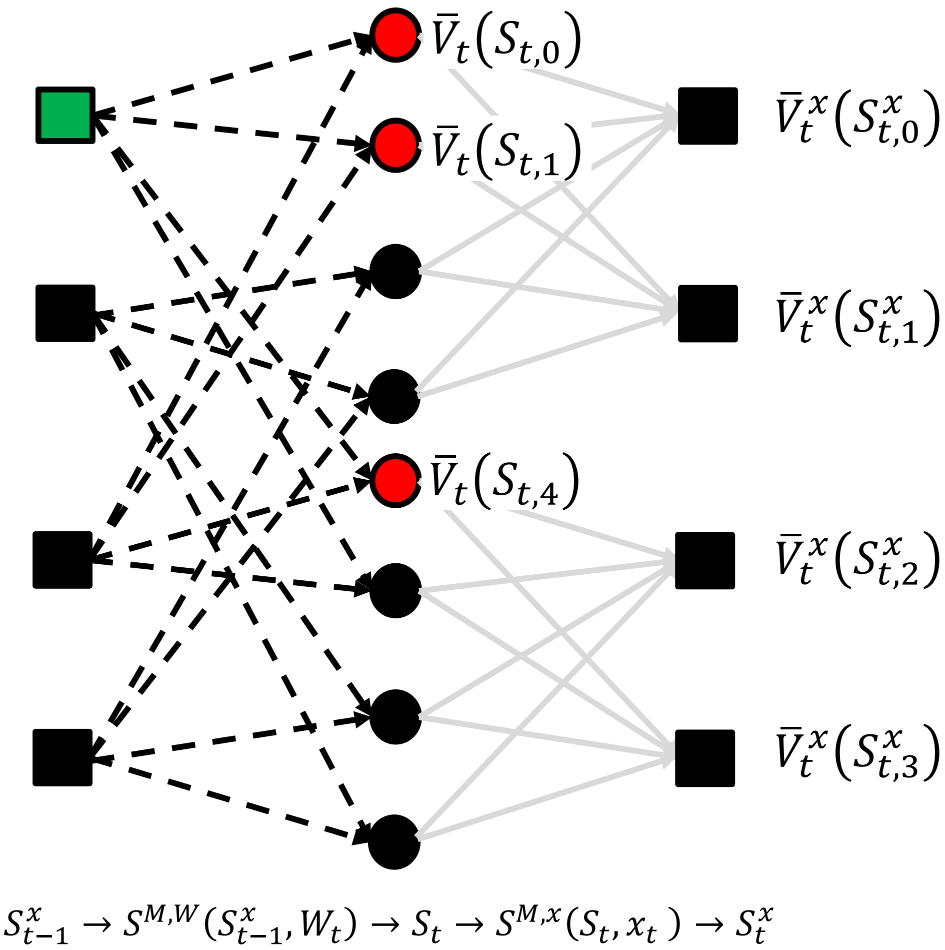
1. Sample pre-decision states and find their values by maximizing over feasible decisions.



$S_{t-1}^x \rightarrow S^{M,W}(S_{t-1}^x, W_t) \rightarrow S_t \rightarrow S^{M,x}(S_t, x_t) \rightarrow S_t^x$

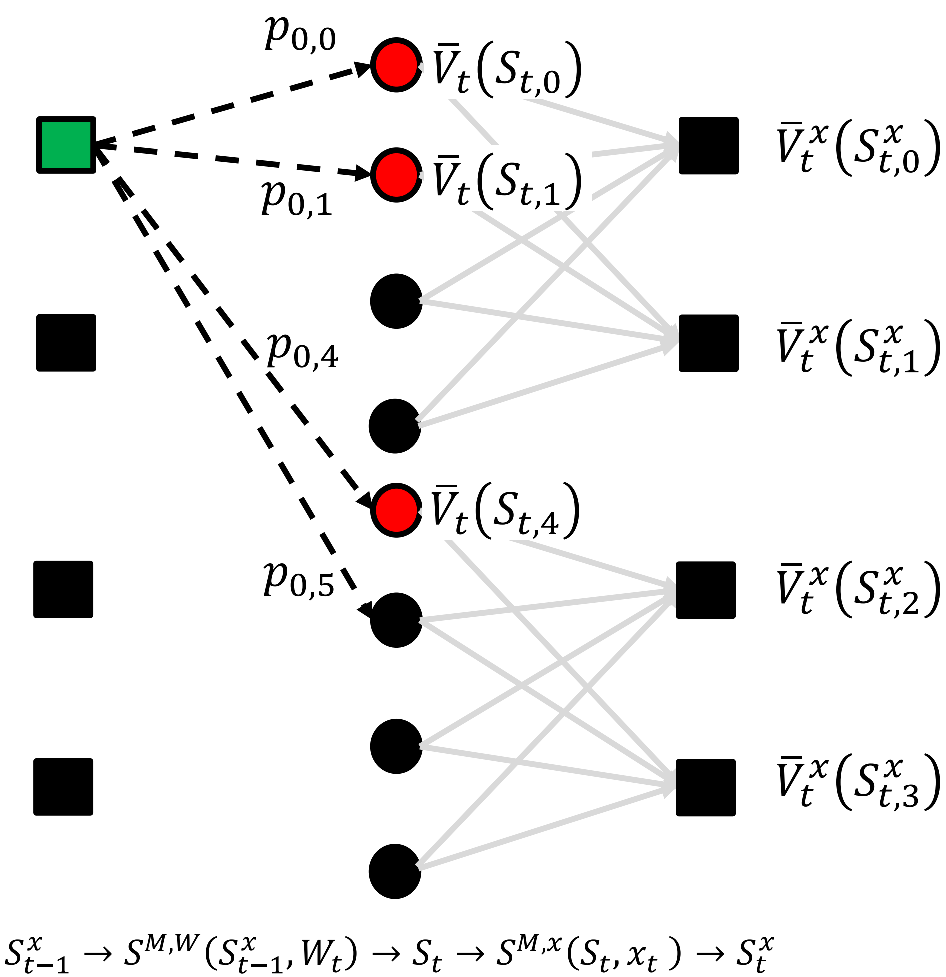
Backward ADP I: Lookup Table form for Post-Decision States

1. Sample pre-decision states and find their values by maximizing over feasible decisions.
2. Compute an approximate $\bar{V}_{t-1}^{x}\left(S_{t-1,s}^{x}\right) = \mathbb{E}[\bar{V}_t(S_t)|S_t^x]$ based on sampled states and re-normalized transition probabilities: $\tilde{p}_{s',s} = \frac{p_{s',s}}{p_s^{norm}}$



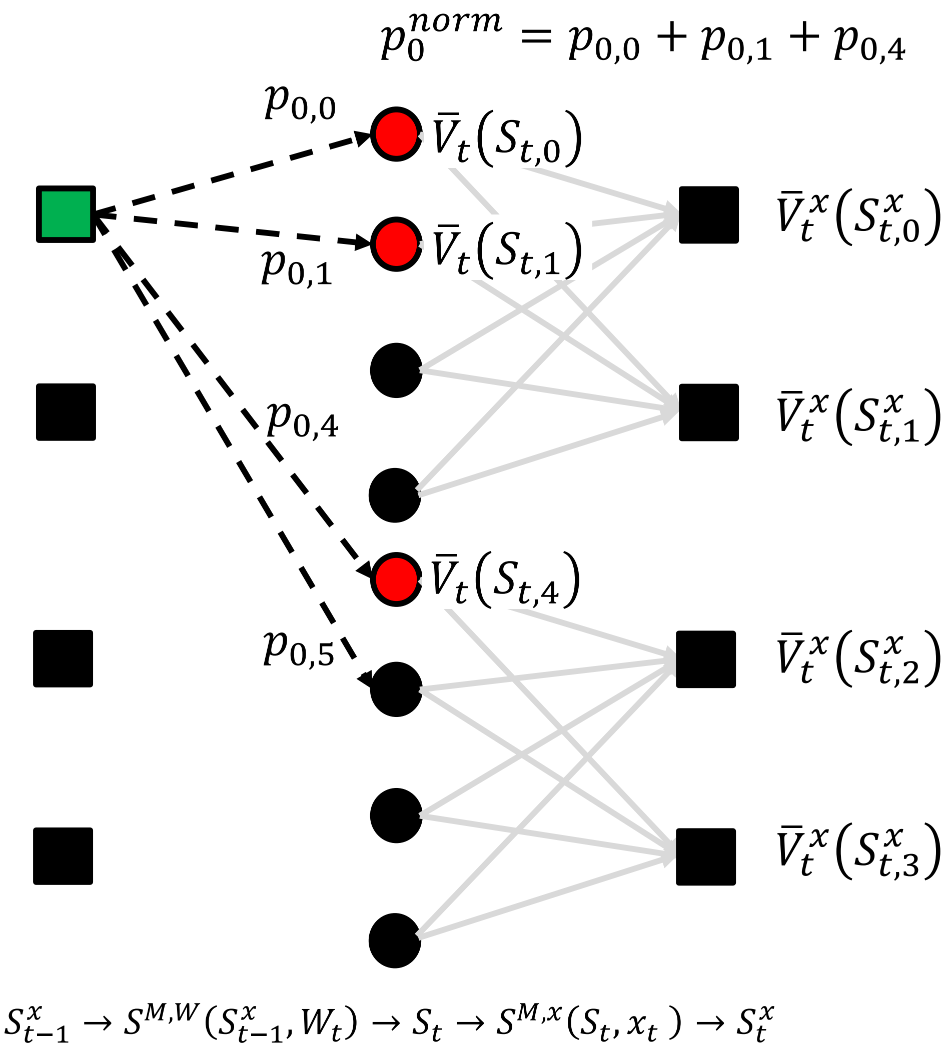
Backward ADP I: Lookup Table form for Post-Decision States

1. Sample pre-decision states and find their values by maximizing over feasible decisions.
2. Compute an approximate $\bar{V}^x_{t-1}\left(S^x_{t-1,s}\right) = \mathbb{E}[\bar{V}_t(S_t)|S^x_t]$ based on sampled states and re-normalized transition probabilities: $\tilde{p}_{s',s} = \frac{p_{s',s}}{p^{norm}_s}$



Backward ADP I: Lookup Table form for Post-Decision States

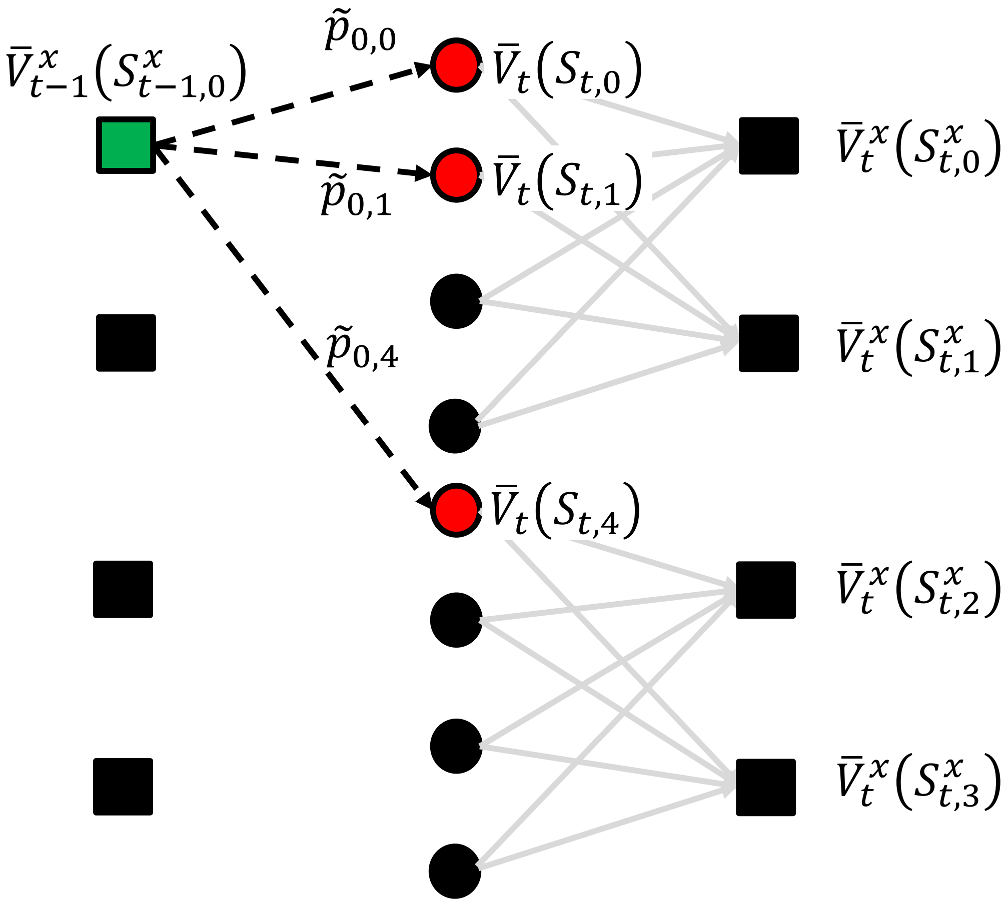
1. Sample pre-decision states and find their values by maximizing over feasible decisions.
2. Compute an approximate $\bar{V}_{t-1}^{x}(S_{t-1,s}^{x}) = \mathbb{E}[\bar{V}_t(S_t)|S_t^x]$ based on sampled states and re-normalized transition probabilities: $\tilde{p}_{s',s} = \frac{p_{s',s}}{p_s^{norm}}$



Backward ADP I: Lookup Table form for Post-Decision States

1. Sample pre-decision states and find their values by maximizing over feasible decisions.
2. Compute an approximate $\bar{V}^x_{t-1}\left(S^x_{t-1,s}\right) = \mathbb{E}[\bar{V}_t(S_t)|S^x_t]$ based on sampled states and re-normalized transition probabilities: $\tilde{p}_{s',s} = \frac{p_{s',s}}{p^{norm}_s}$
3. Store values of post-decision states in a look-up table

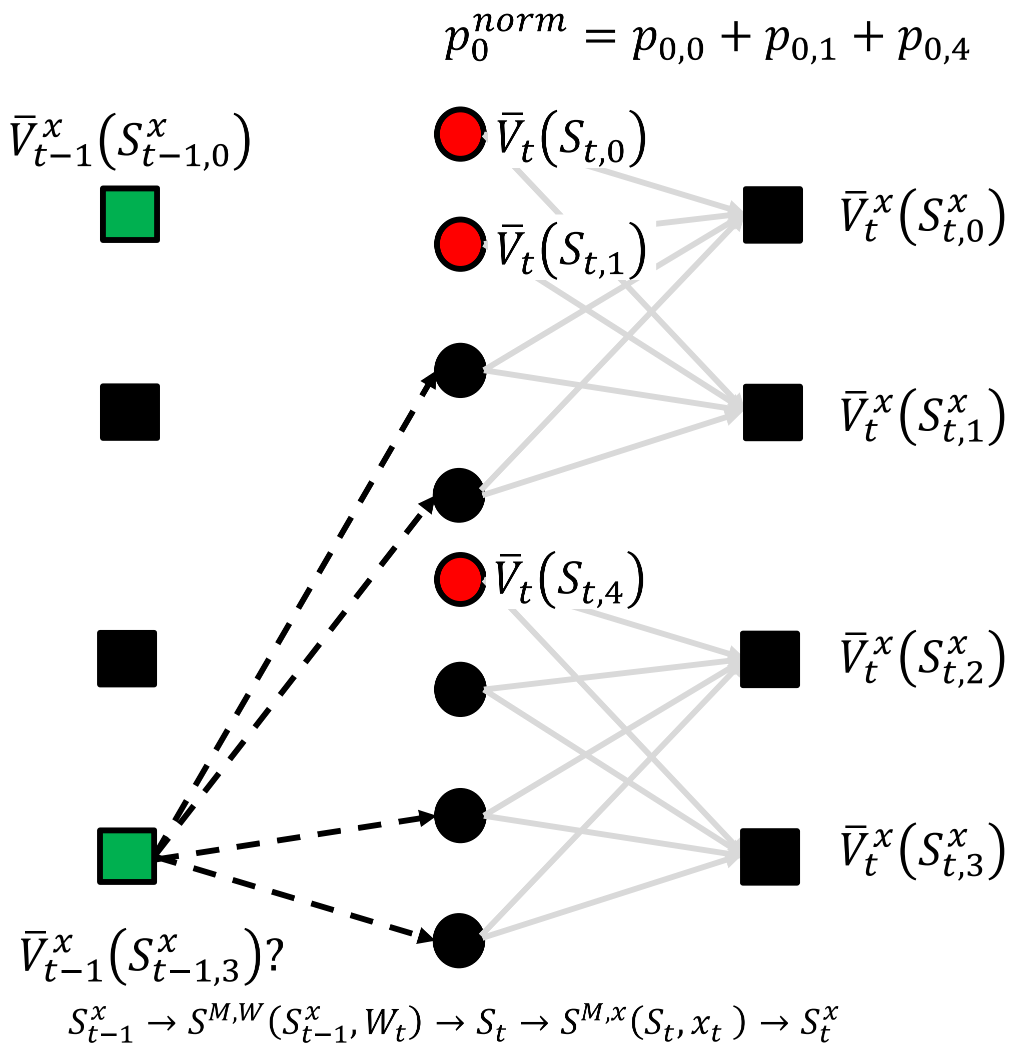
$p^{norm}_0 = p_{0,0} + p_{0,1} + p_{0,4}$



$S^x_{t-1} \to S^{M,W}(S^x_{t-1}, W_t) \to S_t \to S^{M,x}(S_t, x_t) \to S^x_t$

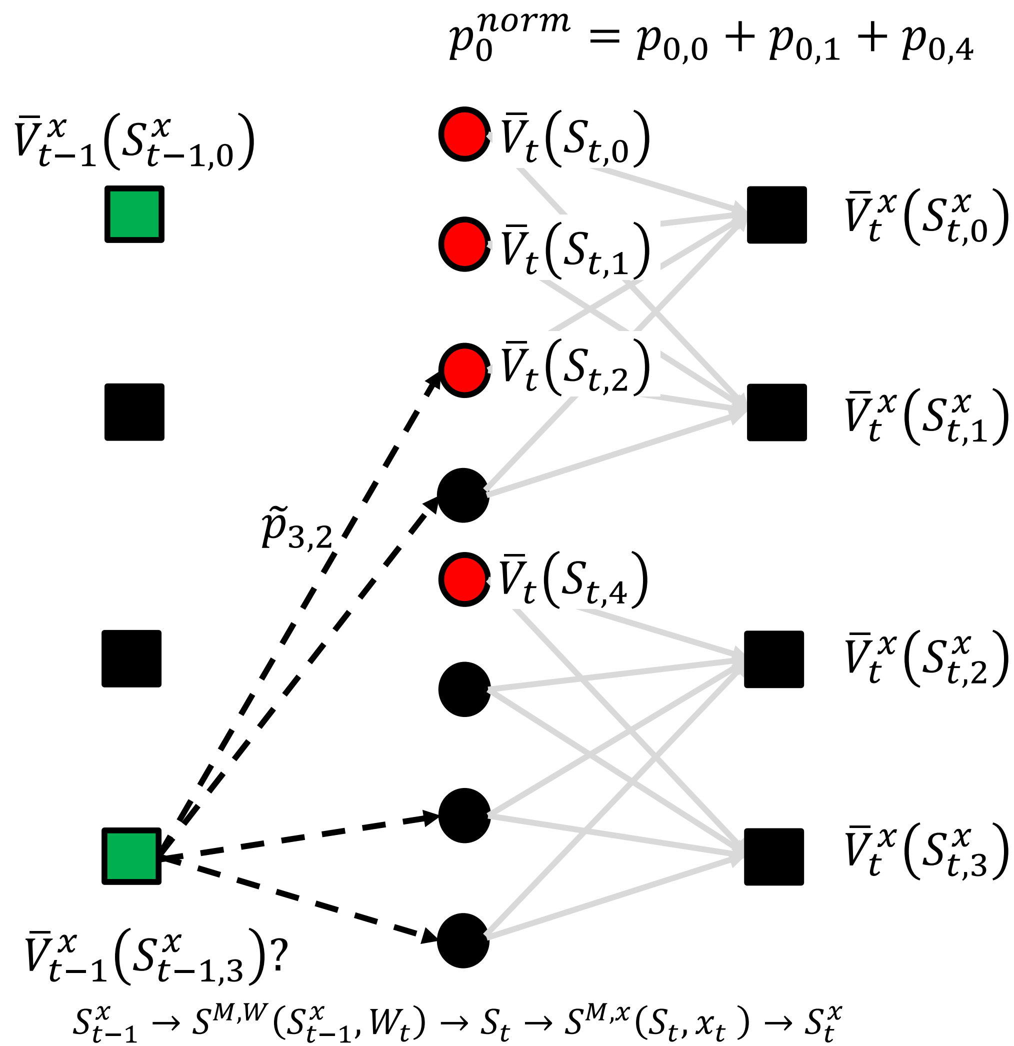
Backward ADP I: Lookup Table form for Post-Decision States

1. Sample pre-decision states and find their values by maximizing over feasible decisions.
2. Compute an approximate $\bar{V}^x_{t-1}\left(S^x_{t-1,s}\right) = \mathbb{E}[\bar{V}_t(S_t)|S^x_t]$ based on sampled states and re-normalized transition probabilities: $\tilde{p}_{s',s} = \frac{p_{s',s}}{p^{norm}_s}$
3. Store values of post-decision states in a look-up table

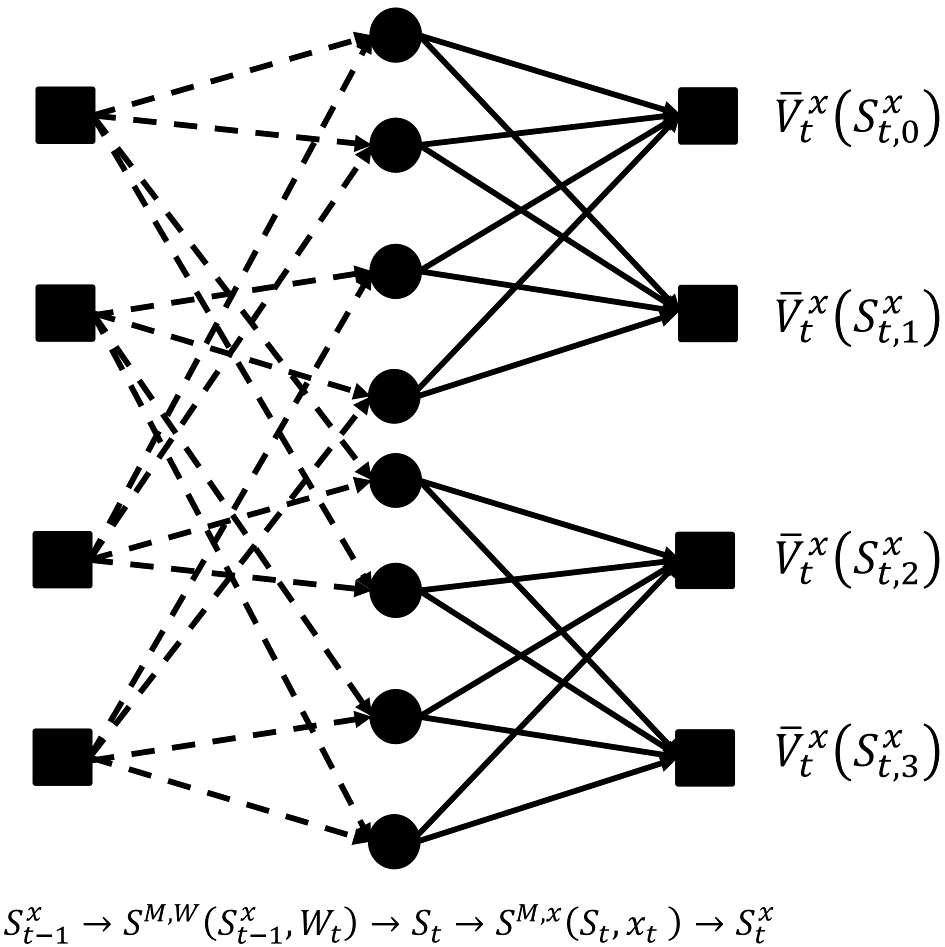


Backward ADP I: Lookup Table form for Post-Decision States

1. Sample pre-decision states and find their values by maximizing over feasible decisions.
2. Compute an approximate $\bar{V}^x_{t-1}\left(S^x_{t-1,s}\right) = \mathbb{E}[\bar{V}_t(S_t)|S^x_t]$ based on sampled states and re-normalized transition probabilities: $\tilde{p}_{s',s} = \frac{p_{s',s}}{p_s^{norm}}$
3. Store values of post-decision states in a look-up table

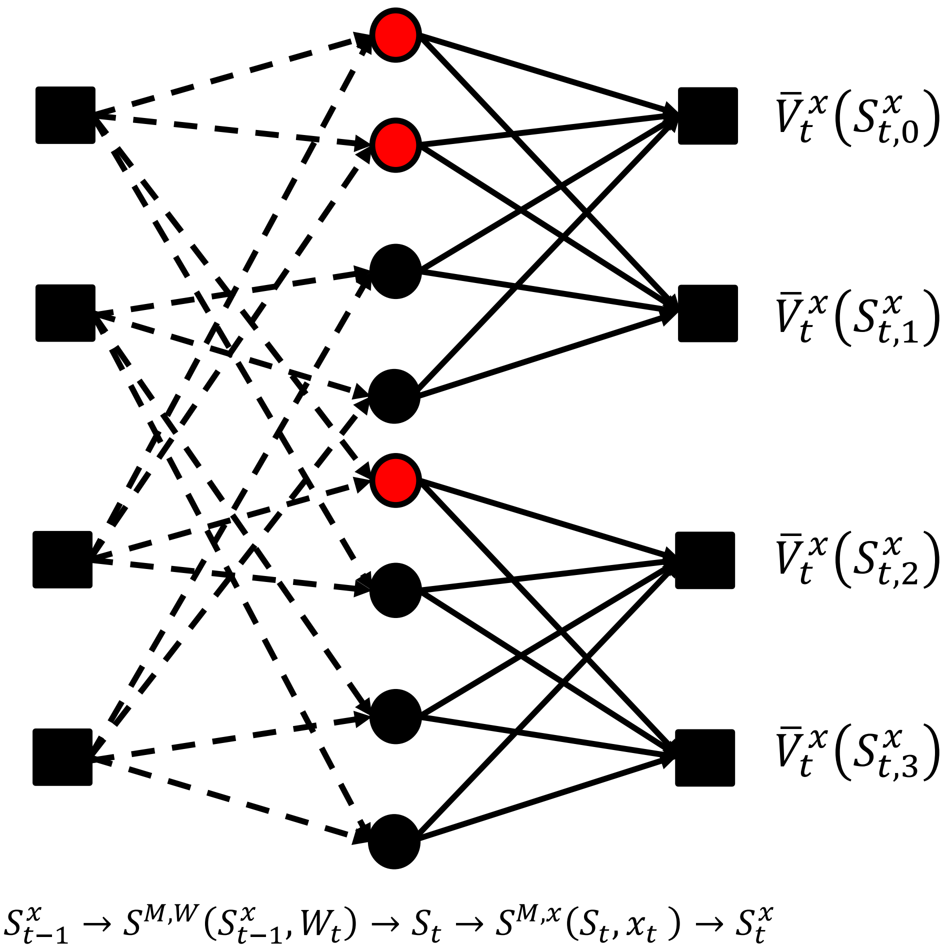


Backward ADP II: Linear VFA's $\bar{V}_t(S_t|\theta_t) = \theta_{t,0} + \sum_{f \in \mathcal{F}} \theta_{t,f}\phi_f(S_t)$



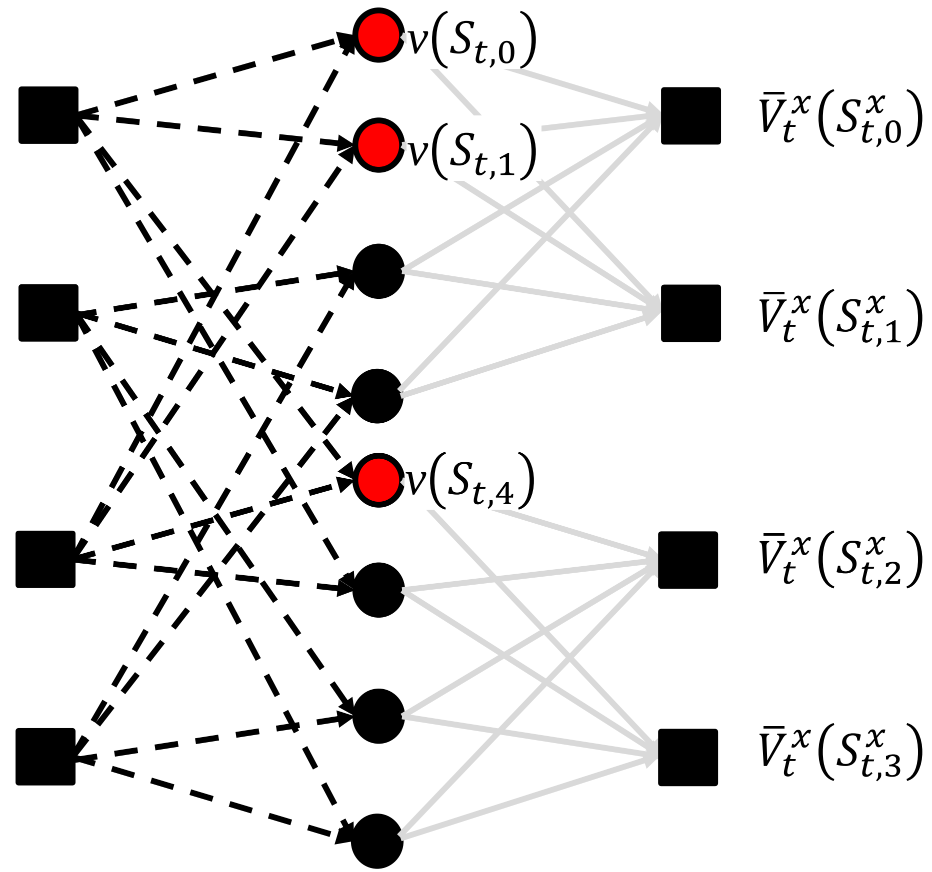
Backward ADP II: Linear VFA's $\bar{V}_t(S_t|\theta_t) = \theta_{t,0} + \sum_{f \in \mathcal{F}} \theta_{t,f}\phi_f(S_t)$

1. Sample pre-decision states and find their values by maximizing over feasible decisions.



Backward ADP II: Linear VFA's $\bar{V}_t(S_t|\theta_t) = \theta_{t,0} + \sum_{f\in\mathcal{F}} \theta_{t,f}\phi_f(S_t)$

1. Sample pre-decision states and find their values by maximizing over feasible decisions.



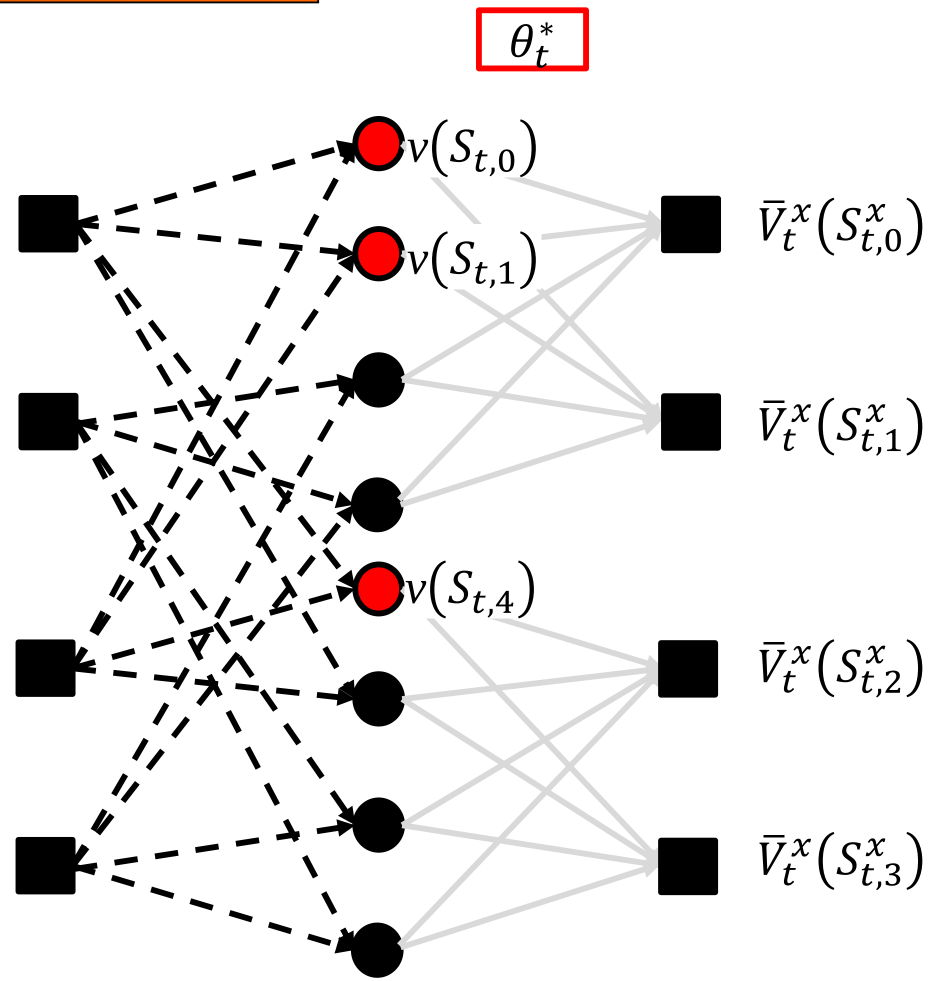
$S_{t-1}^x \rightarrow S^{M,W}(S_{t-1}^x, W_t) \rightarrow S_t \rightarrow S^{M,x}(S_t, x_t) \rightarrow S_t^x$

Backward ADP II: Linear VFA's $\bar{V}_t(S_t|\theta_t) = \theta_{t,0} + \sum_{f\in\mathcal{F}} \theta_{t,f}\phi_f(S_t)$

1. Sample pre-decision states and find their values by maximizing over feasible decisions.

2. Find best fit parameter vector θ_t^* by least squares regression based on set of sampled states and values

3. Store θ_t^* in memory



$S_{t-1}^x \rightarrow S^{M,W}(S_{t-1}^x, W_t) \rightarrow S_t \rightarrow S^{M,x}(S_t, x_t) \rightarrow S_t^x$

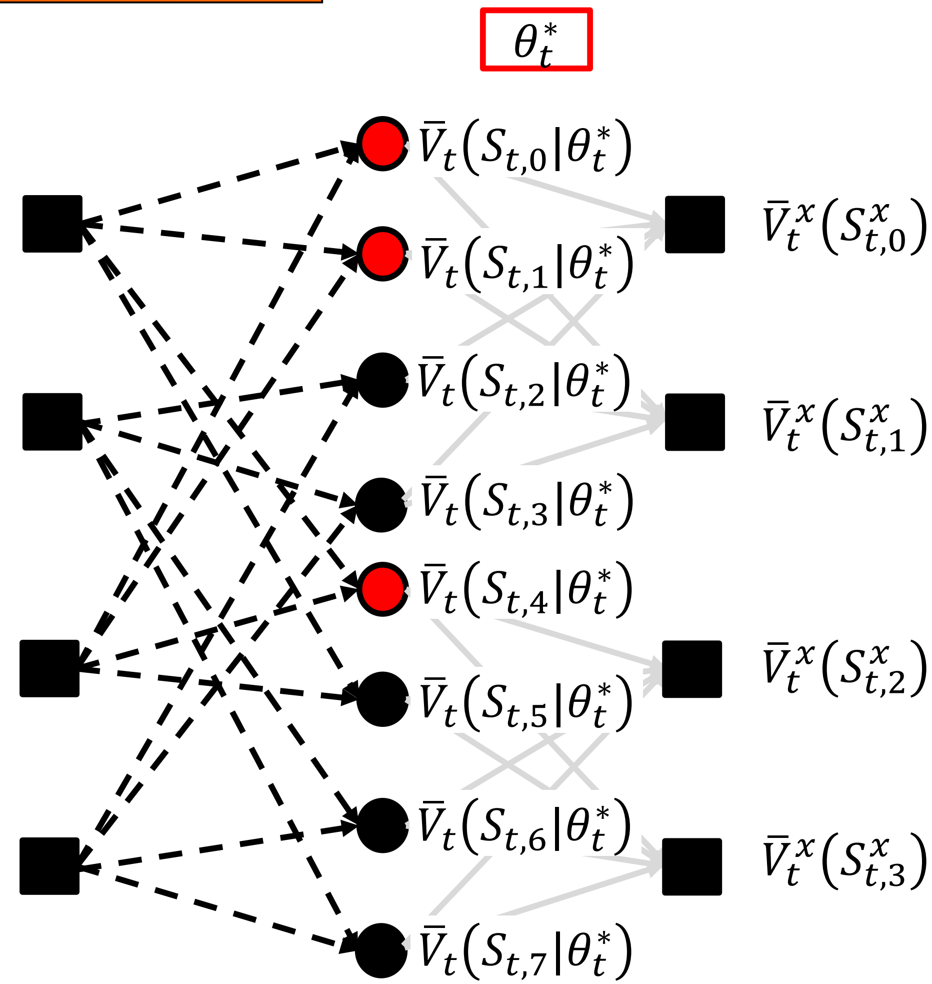
Backward ADP II: Linear VFA's $\bar{V}_t(S_t|\theta_t) = \theta_{t,0} + \sum_{f \in \mathcal{F}} \theta_{t,f}\phi_f(S_t)$

1. Sample pre-decision states and find their values by maximizing over feasible decisions.

2. Find best fit parameter vector θ_t^* by least squares regression based on set of sampled states and values

3. Store θ_t^* in memory

4. Using θ_t^*, find $\bar{V}_t(S_t|\theta_t^*)$ for all time t pre-decision states.



$S_{t-1}^x \rightarrow S^{M,W}(S_{t-1}^x, W_t) \rightarrow S_t \rightarrow S^{M,x}(S_t, x_t) \rightarrow S_t^x$

Backward ADP II: Linear VFA's $\bar{V}_t(S_t|\theta_t) = \theta_{t,0} + \sum_{f\in\mathcal{F}} \theta_{t,f}\phi_f(S_t)$

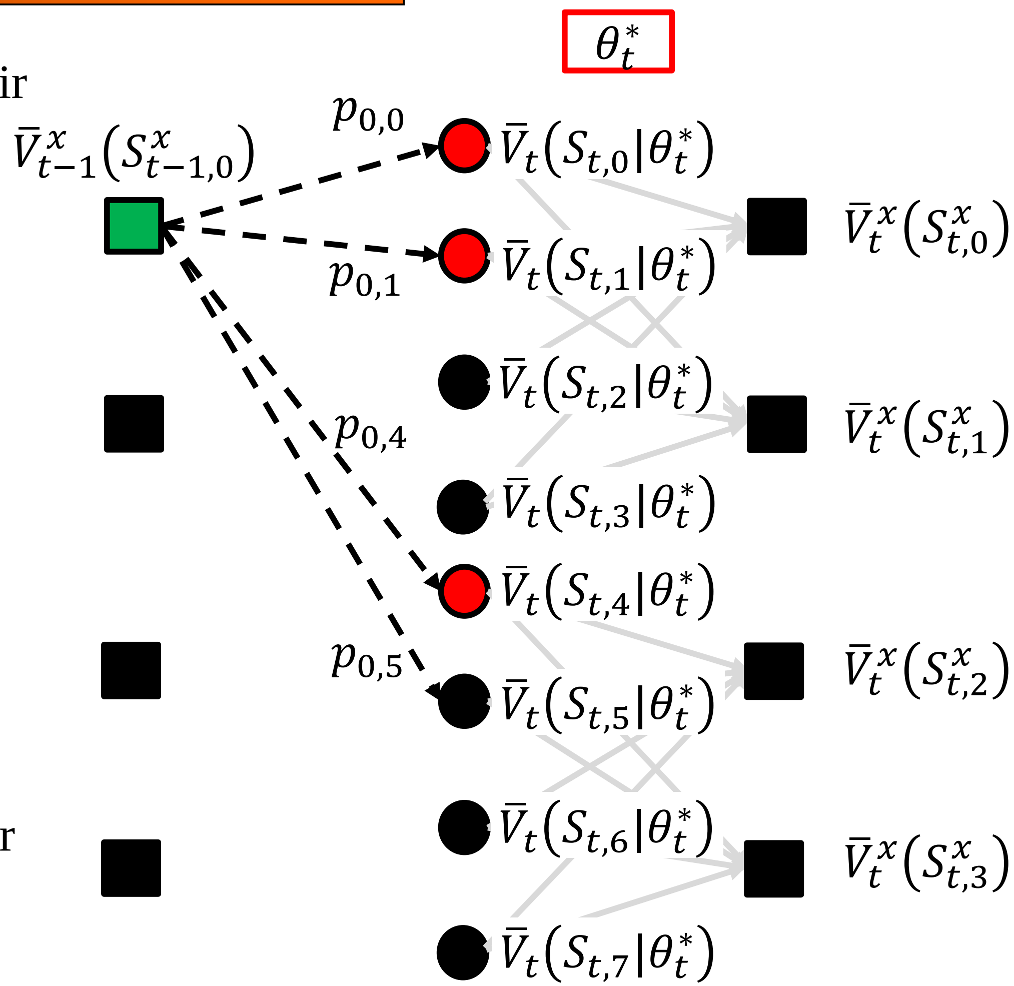
1. Sample pre-decision states and find their values by maximizing over feasible decisions.

2. Find best fit parameter vector θ_t^* by least squares regression based on set of sampled states and values

3. Store θ_t^* in memory

4. Using θ_t^*, find $\bar{V}_t(S_t|\theta_t^*)$ for all time t pre-decision states.

5. $\bar{V}_{t-1}^x(S_{t-1,s}^x) = \sum_{s'} p_{s,s'} \bar{V}_t(S_{t,s'}|\theta_t^*)$ for each time $t-1$ post-decision state



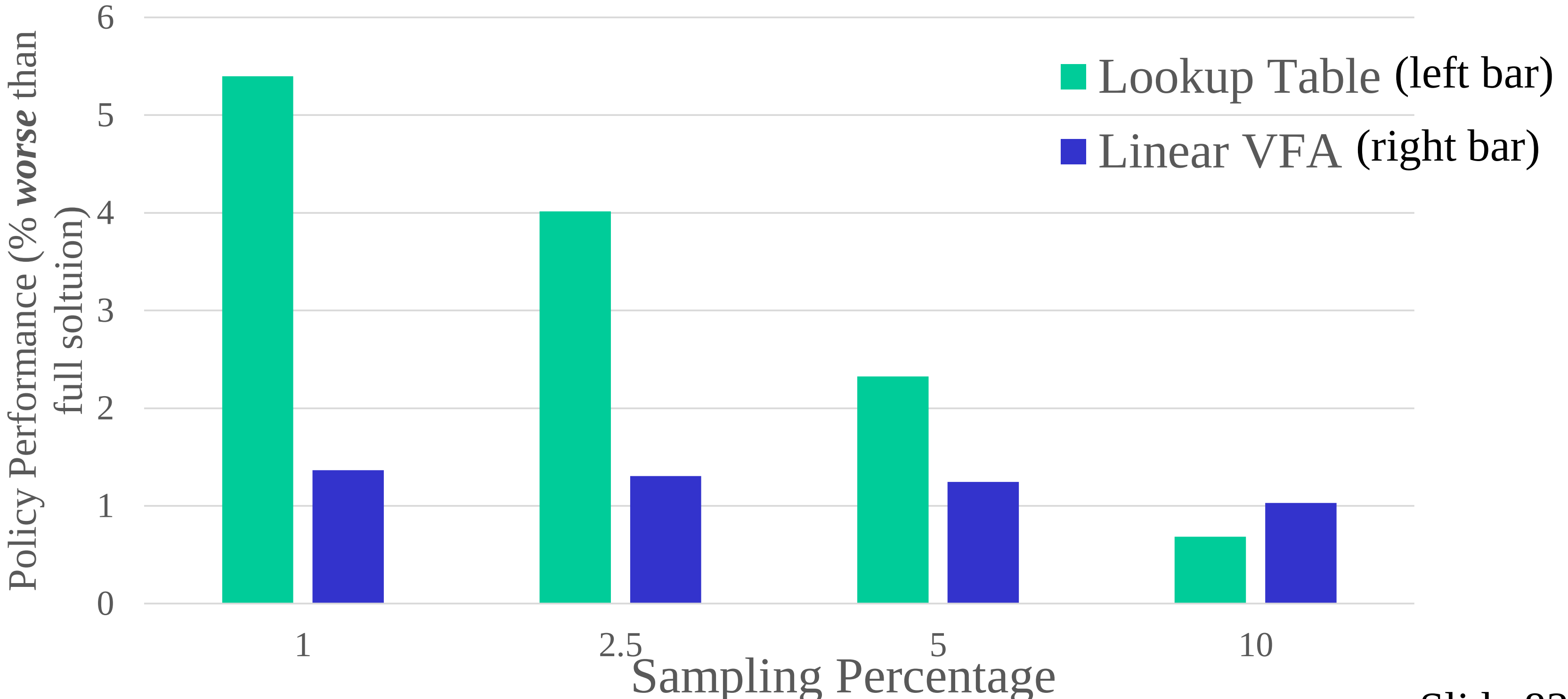
$S_{t-1}^x \rightarrow S^{M,W}(S_{t-1}^x, W_t) \rightarrow S_t \rightarrow S^{M,x}(S_t, x_t) \rightarrow S_t^x$

Outline

- Motivation
- Hidden Semi-Markov Crossing State Model
- Formulating the Energy Storage Problem as a Markov Decision Process
- Backward Approximate Dynamic Programming
- Numerical Results

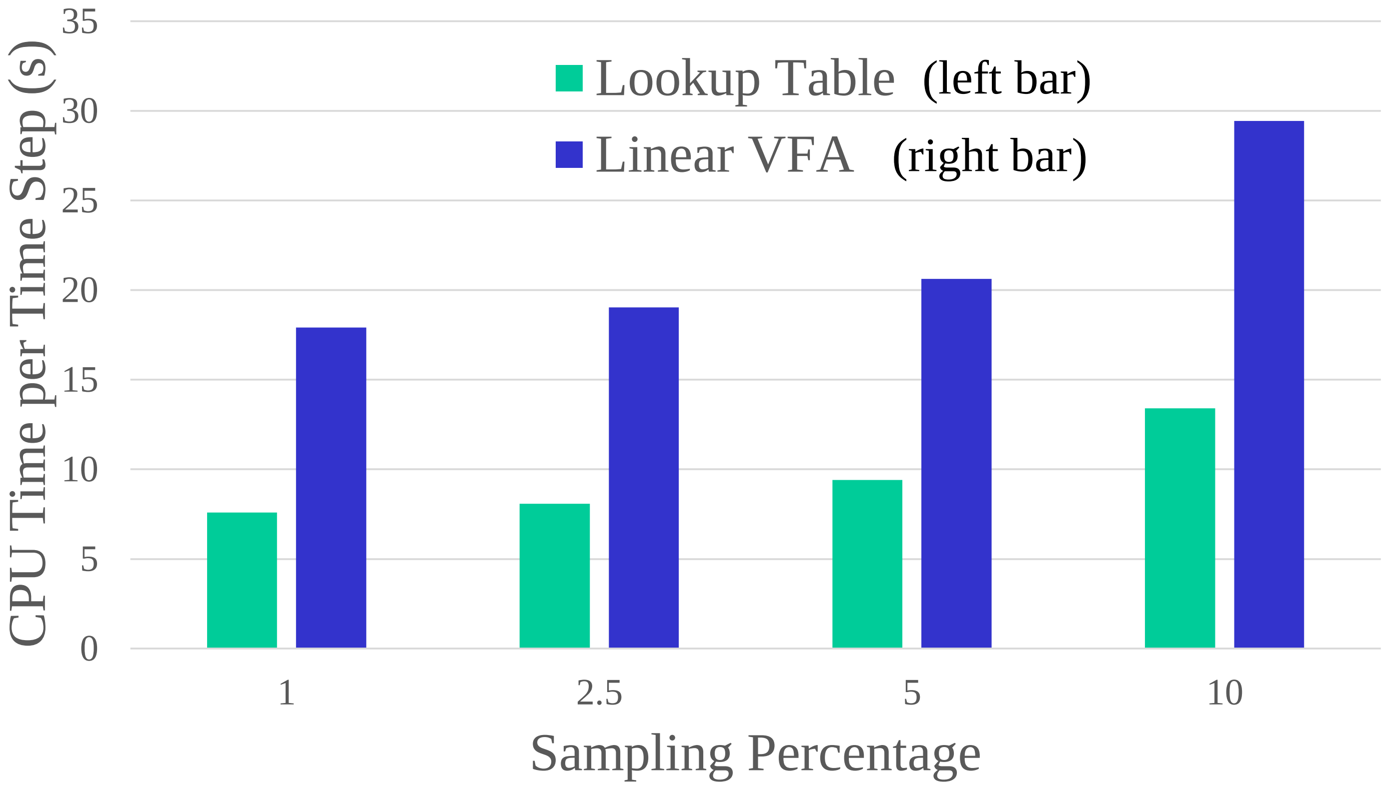
Numerical Results: Policy Performance

- Average performance over 100 realistic wind and price sample paths for one scenario, results consistent in others as well
- Results compared to performance of the full MDP solution
- Properly tuned Buy-Low, Sell-High Industry Heuristic: 18.62 % worse performance

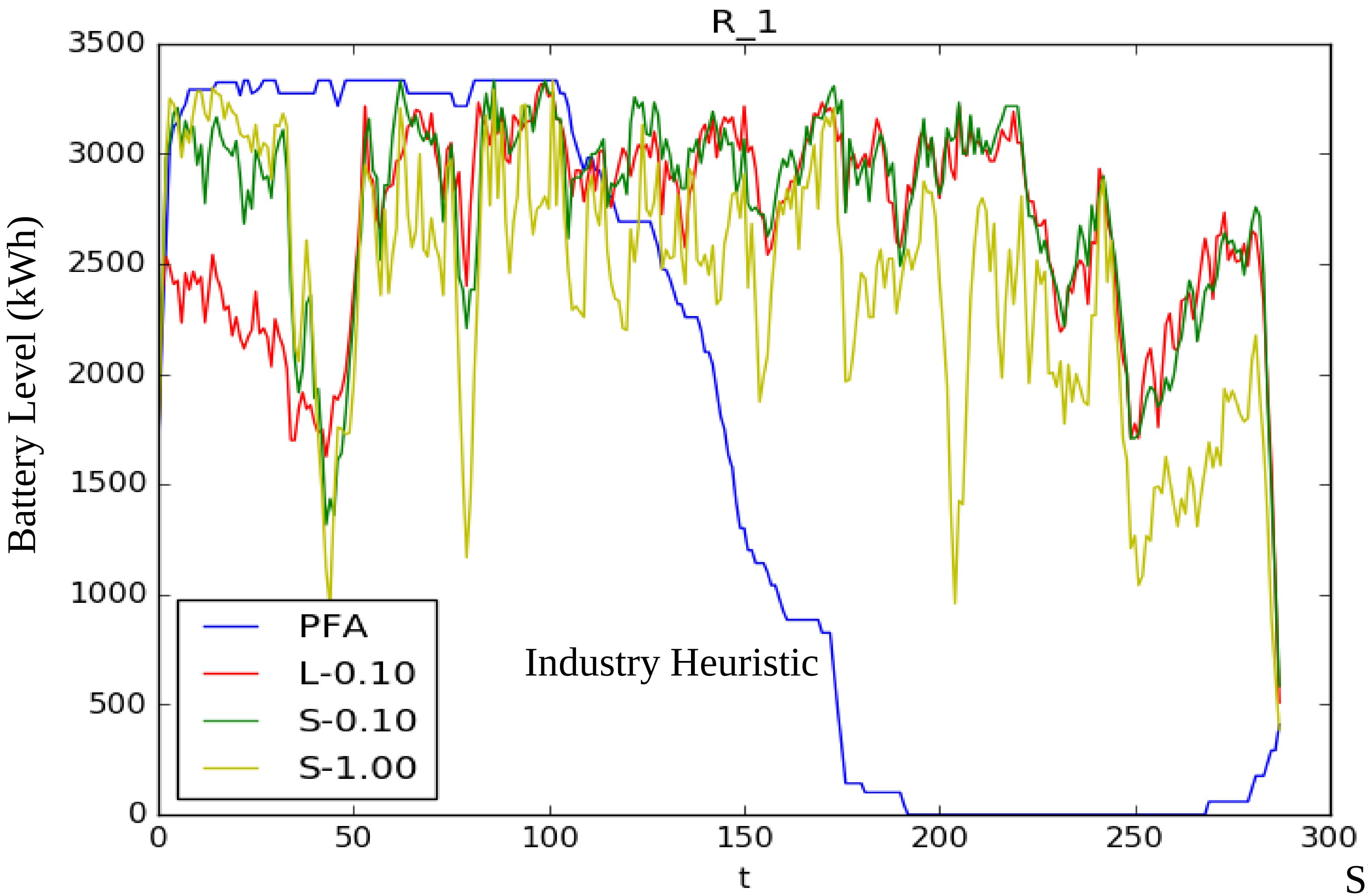


Numerical Results: CPU Time and Memory

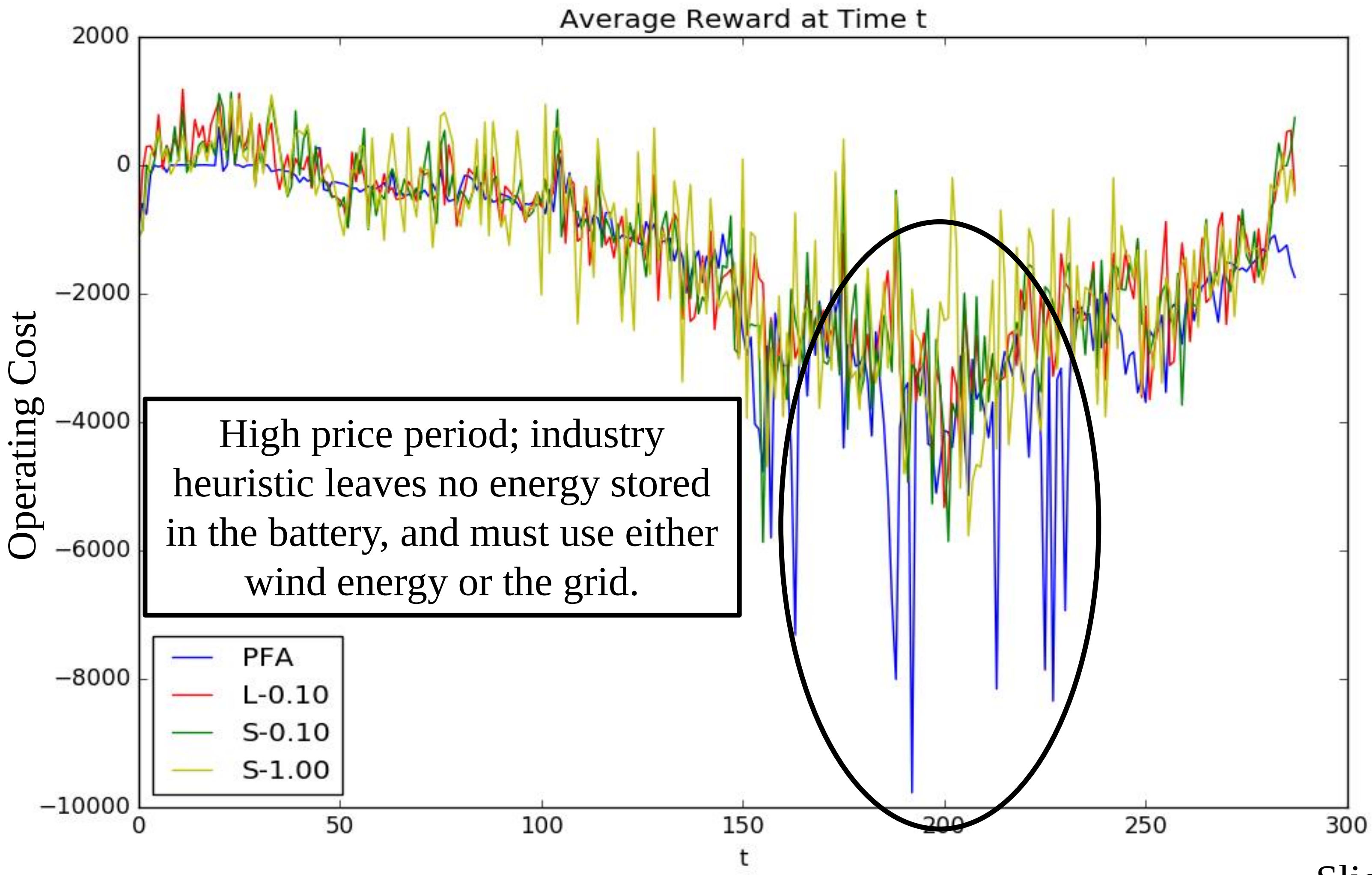
- Full solution: ~300 seconds per time step
- Memory to store value functions per time step:
  - » Lookup Table: 350 kB
  - » Linear VFA: 158 ***bytes***



Numerical Results: Policy Behavior



Numerical Results: Policy Behavior



Thank You!

- Any questions?

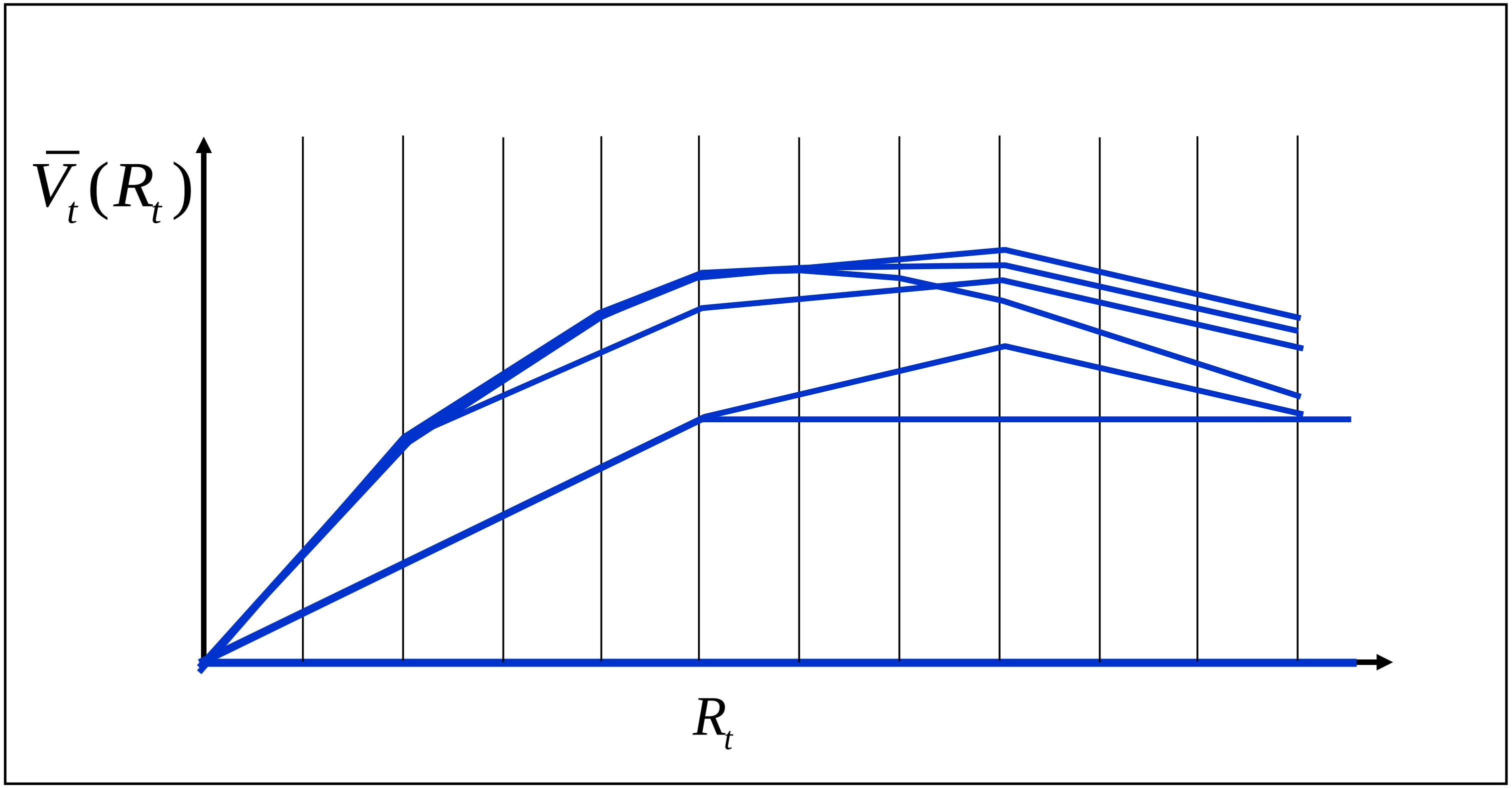
- Imagine 20 large storage devices spread around the PJM grid:

Grid Level Energy Storage

- Goal: Show we can reduce shortages by modeling offshore wind power output with a crossing state model.
- Stochastic Dual Decomposition Procedure (SDDP) used to fit value functions (form of forward ADP)
- Overcomes curse of dimensionality by exploiting concavity in the resource state

Exploiting convexity/concavity

- Derivatives are used to estimate a piecewise linear approximation



Traditional SDDP – Stagewise Independence (IID)

Forward Pass at Iteration k

Sample $\omega \in \Omega$.
for $t = 0, \dots, T$ **do**
  if $(k = 0)$ **then**

$$\text{Select } x_t^k \in \underset{x_t \in \mathcal{X}_t(R_{t-1}^{x,k}, I_t(\omega))}{\arg\min} \{C(S_t(\omega), x_t)\}$$

  else

$$\text{Select } x_t^k \in \underset{x_t \in \mathcal{X}_t(R_{t-1}^{x,k}, I_t(\omega))}{\arg\min} \left\{C(S_t(\omega), x_t) + \overline{V}_t^{k-1}(R_t^x)\right\}$$

  end if
  Set $R_t^{x,k} \leftarrow B_t^k x_t^k$; $S_{t+1}(\omega) \leftarrow (R_t^{x,k} - b_{t+1}(\omega), I_{t+1}(\omega))$
end for

Traditional SDDP – Stagewise Independence (IID)

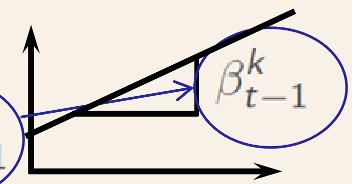
Backward Pass at Iteration k

for $t = T, \ldots, 1$ **do**

$$\text{Define } \underline{V}_t^k(R_{t-1}^x, \omega_t) := \min_{x_t \in \mathcal{X}_t(R_{t-1}^x, I_t(\omega_t))} \left\{ C(S_t(\omega_t), x_t) + \overline{V}_t^k(R_t^x) \right\}$$

  for all $\omega_t \in \Omega_t$ **do**

$$\text{Select } \underline{\beta}_t^k(\omega_t) \in \partial_{R_{t-1}^x} \underline{V}_t^k(R_{t-1}^{x,k}, \omega_t)$$

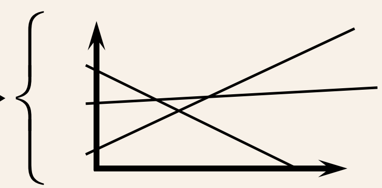


  end for

  $\alpha_{t-1}^k \leftarrow \sum_{\omega_t \in \Omega_t} \mathbb{P}(\omega_t) \underline{V}_t^k(R_t^{x,k}, \omega_t), \quad \beta_{t-1}^k \leftarrow \sum_{\omega_t \in \Omega_t} \mathbb{P}(\omega_t) \underline{\beta}_t^k(\omega_t)$

  $h_{t-1}^k(R_{t-1}^x) := \alpha_{t-1}^k + \langle \beta_{t-1}^k, R_{t-1}^x - R_{t-1}^{x,k} \rangle$

  $\overline{V}_{t-1}^k(R_{t-1}^x) := \max \left\{ \overline{V}_{t-1}^{k-1}(R_{t-1}^x),\ h_{t-1}^k(R_{t-1}^x) \right\}$



end for

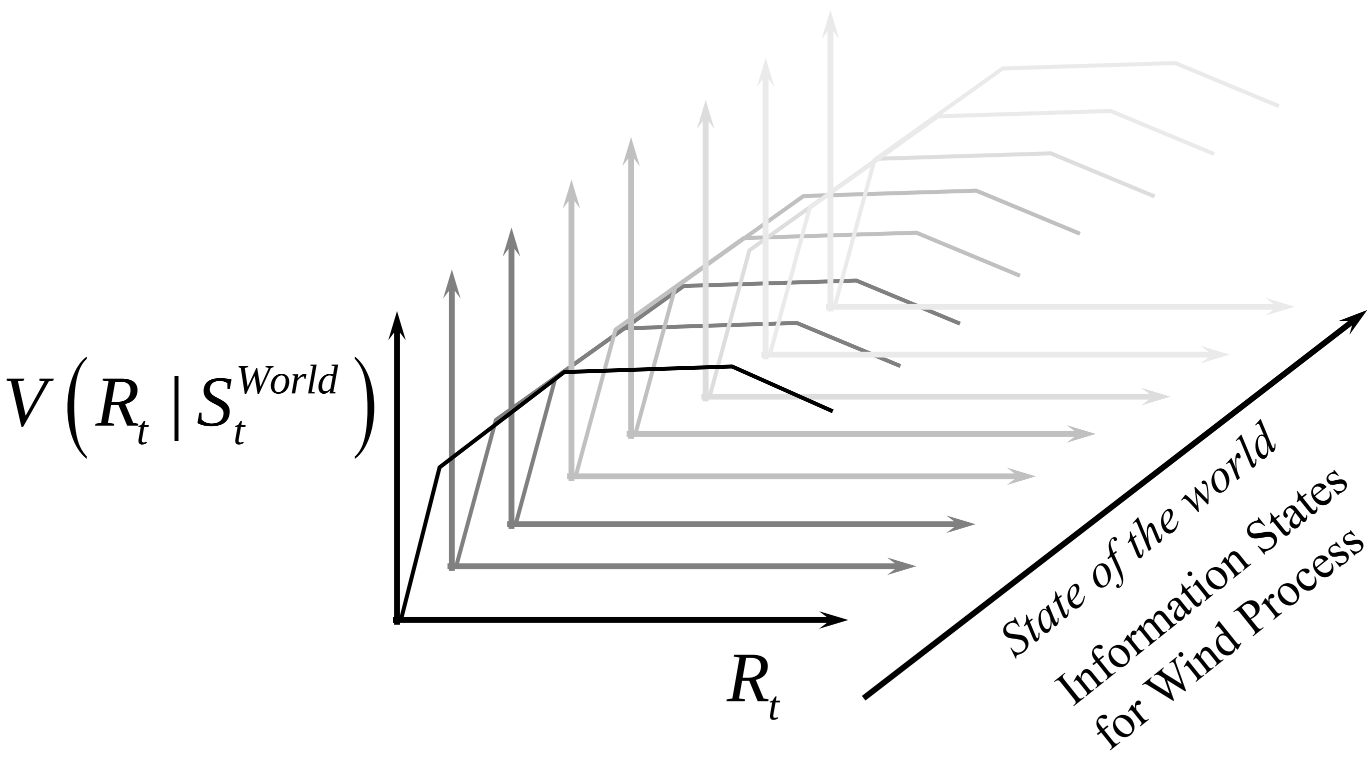
$\underline{V}_0^k \leftarrow \left\{ \min_{x_0 \in \mathcal{X}_0(S_0)} C(S_0, x_0) + \overline{V}_0^k(R_0^x) \right\}$

Grid Level Energy Storage

- Classic SDDP assumes intertemporal independence (IID Errors)
- New algorithm – SDDP with Markov Uncertainty
  - » Regularized Decomposition of High-Dimensional Multistage Stochastic Programs with Markov Uncertainty
  - » Used in combination with the HSMM

State-of-the-world variables

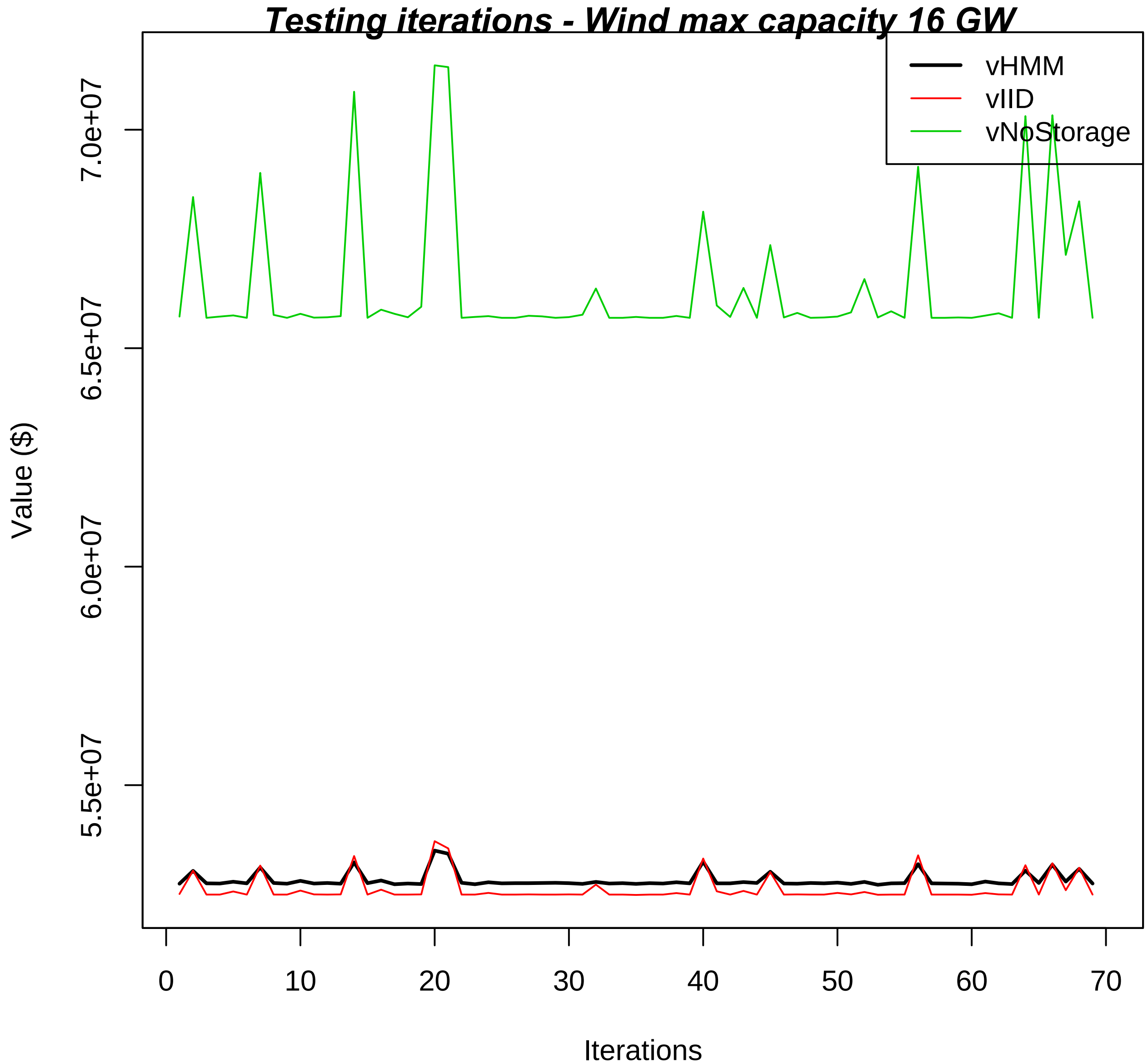
- Value functions fit to the information states of the wind process (U/D, S/M/L, $\hat{E}_t^g$ error bin)



Numerical Experiments

- **Training** (finding value functions) was performed assuming both **intertemporal independence (IID errors)** and the **hidden semi-Markov crossing state model (HSMM)**
- **Testing for both set of VFA's (IID and HSMM)** performed using sample paths from the hidden semi-Markov model as this produces more realistic wind behavior

Numerical Experiments



Numerical Experiments