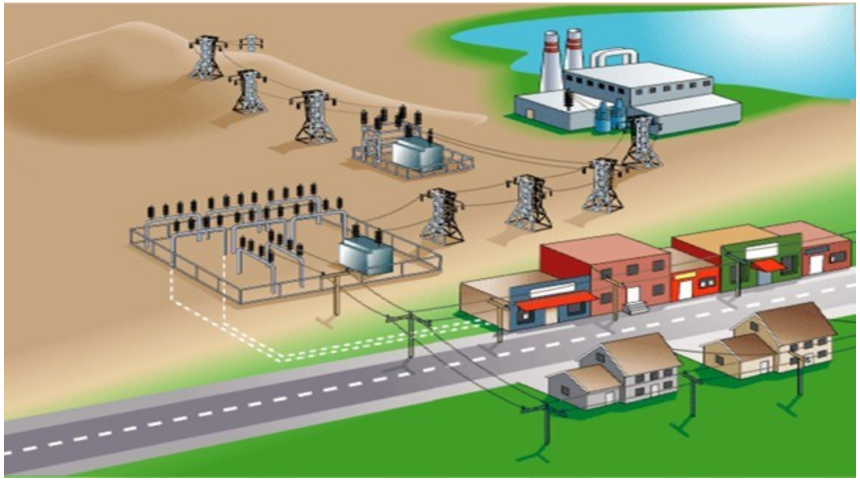
Dynamic Pricing and Energy Consumption Scheduling With Reinforcement Learning

Elham Foruzan

Department of Computer science and engineering
University of Nebraska-Lincoln
Lincoln, Nebraska
Elham.foruzan@huskers.unl.edu

University of Nebraska Lincoln

Overview Day One



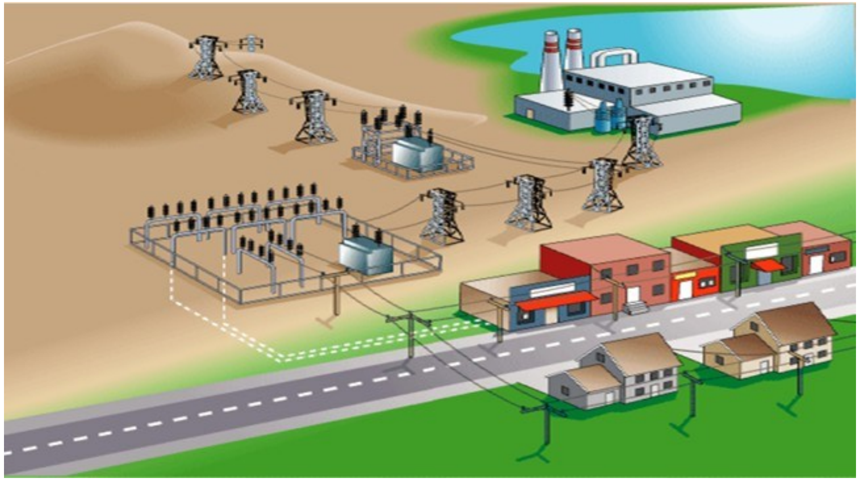
- Introduction & background
- Goal & objectives
- System Model
- Reinforcement Learning at Service Provider
- Energy Consumption-Based Approximated
- Virtual Experience for Accelerating Learning

Overview Day Two

- Reinforcement Learning at Customers

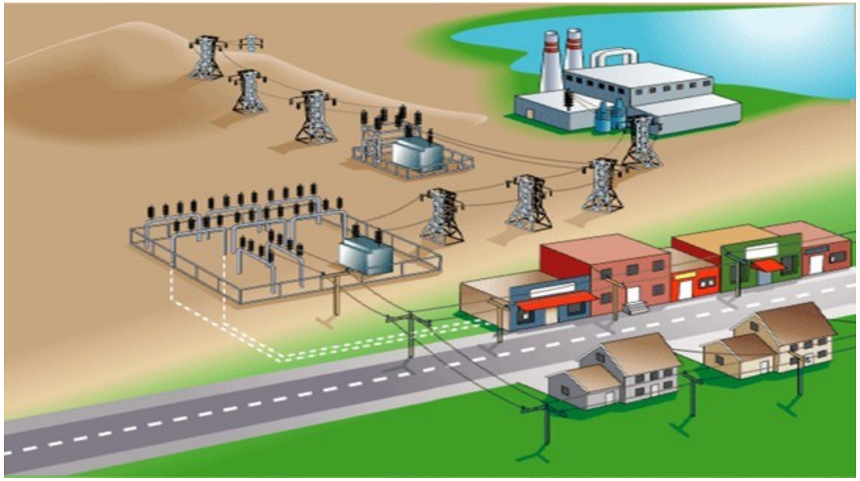
- Post Decision State Learning

- Numerical Results



Overview

- Introduction & background
- Goal & objectives
- System Model
- Reinforcement Learning at Service Provider
- Energy Consumption-Based Approximated
- Virtual Experience for Accelerating Learning



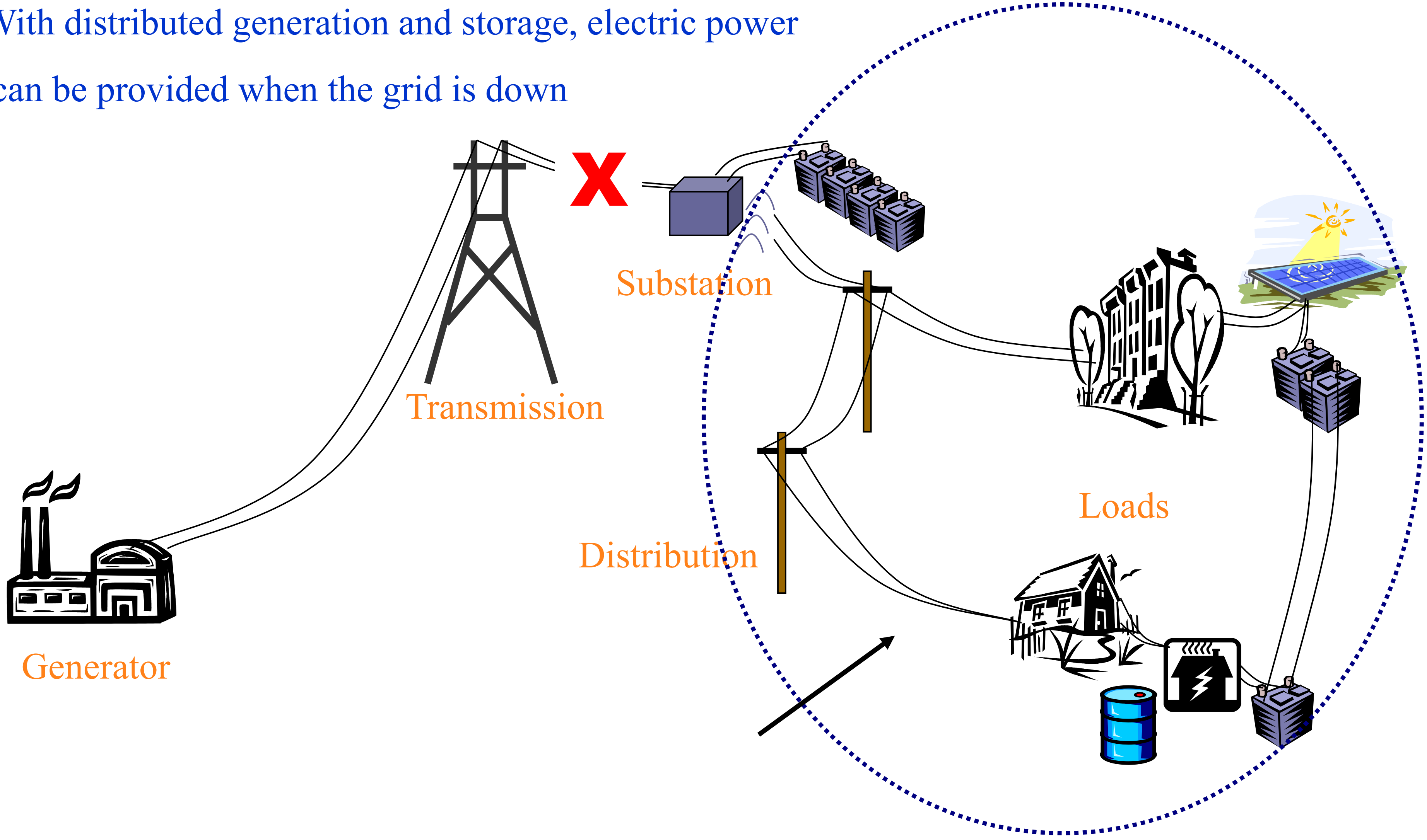
State of our planet

Source: http://theearthproject.com/


Sourcehttp://www.openculture.com/
Source: http://blog.livedoor.jp
Source: http://blog.livedoor.jp

Motivation: why microgrid?

With distributed generation and storage, electric power can be provided when the grid is down

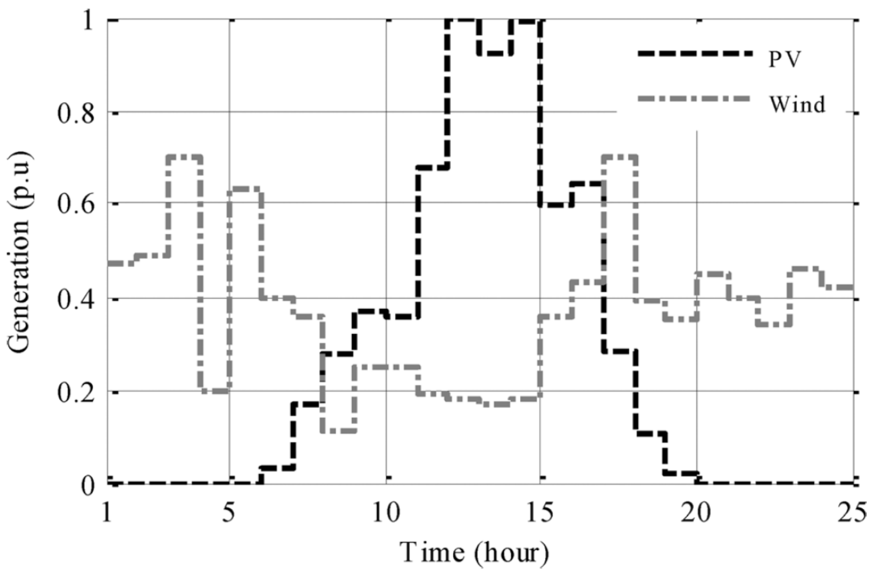


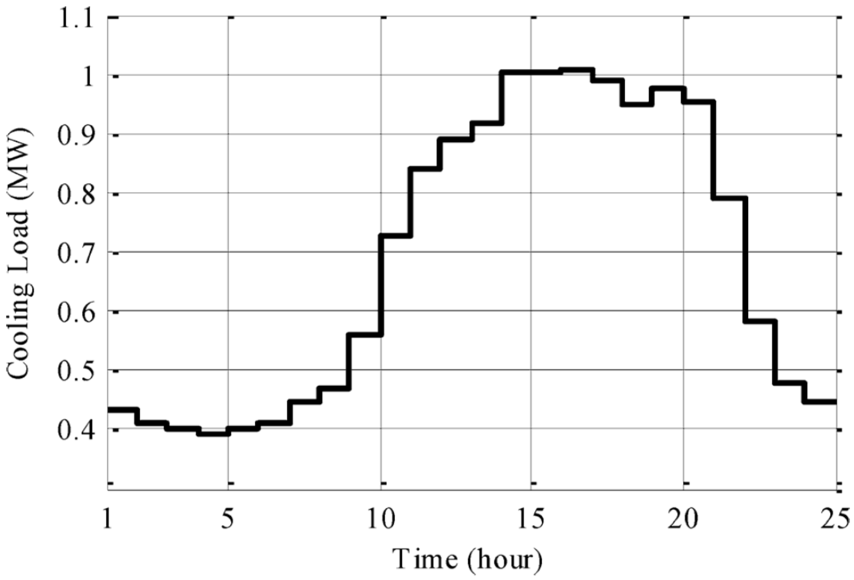
Source: http://www.engineering.com

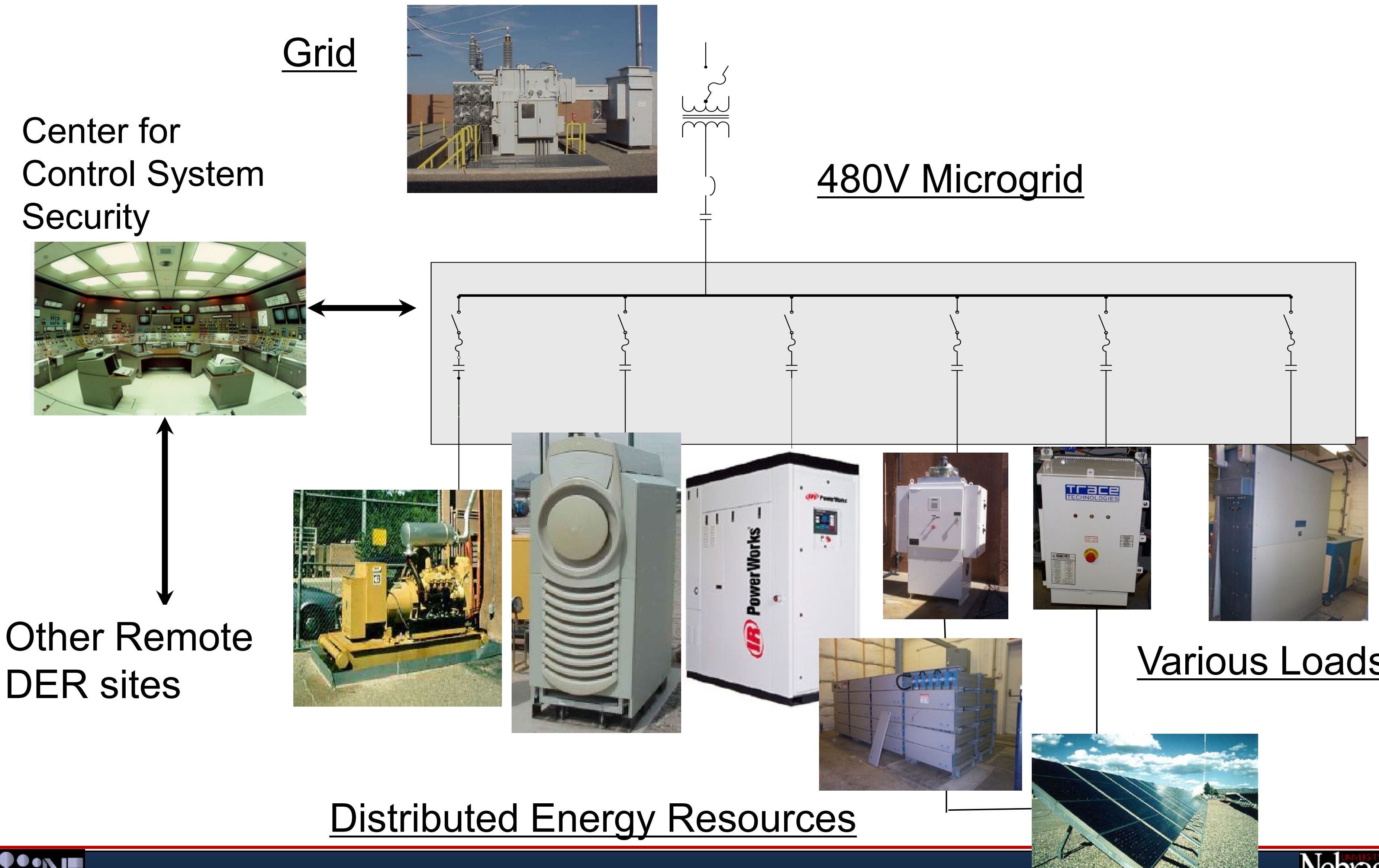
Microgrid

https://www.youtube.com/watch?v=EhkdYqNU-ac

PV and Load Daily Profile





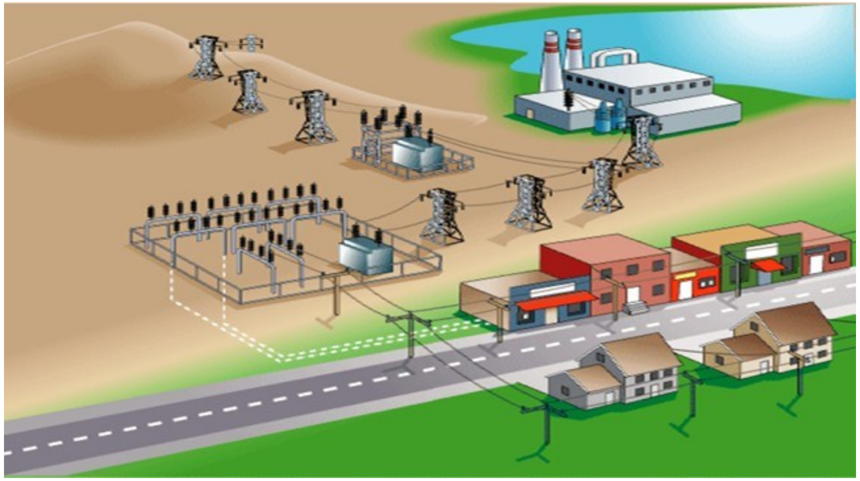

Grid
Center for Control System Security
480V Microgrid
PowerWorks
Trace Technologies
Other Remote DER sites
Various Loads
Distributed Energy Resources

Demand Response

https://www.sce.com/

Overview

- Introduction & background
- **Goal & objectives**
- System Model
- Reinforcement Learning at Service Provider
- Energy Consumption-Based Approximated
- Virtual Experience for Accelerating Learning



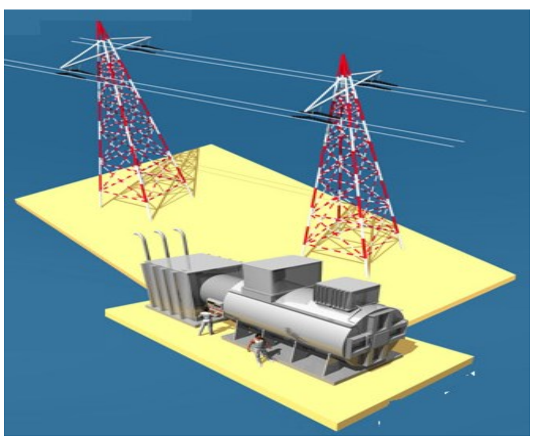
Abstract

Paper goal: Dynamic pricing and energy scheduling in microgrid.

Customer: consuming electricity
Utility: electricity Generators
Service Provider: Buy electricity from Utilities and sell to customer.

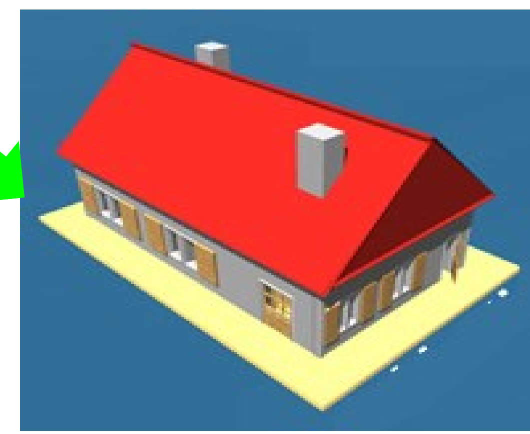
Method: Reinforcement learning implementation that allow costumers and service providers (SP) to strategically learn without prior information.

Service Provider (SP)



Power generation unit

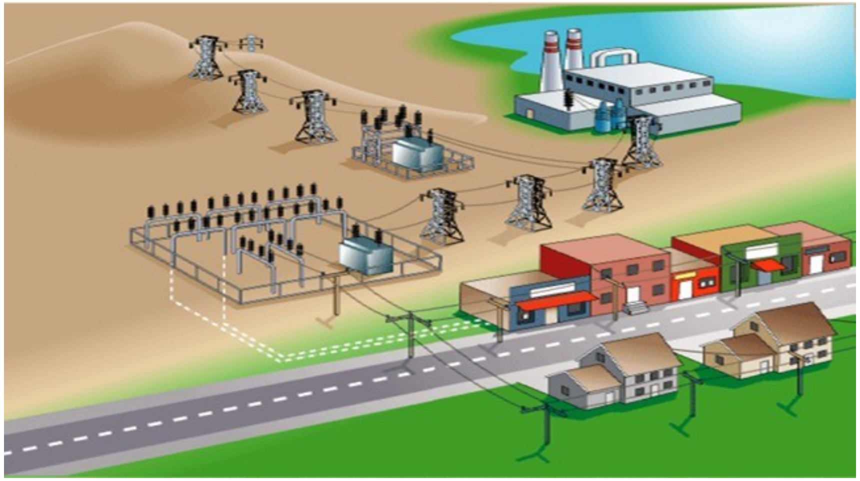
Source: energy.gov



Residential

Overview

- Introduction & background
- Goal & objectives
- **System Model**
- Reinforcement Learning at Service Provider
- Energy Consumption-Based Approximated
- Virtual Experience for Accelerating Learning



System Time Slot

- To consider the **variable load consumption** of customer and **retail price of electricity during a day**, the set of period **H={1,2,3,..,H-1}** was introduced.

- Each time-slot t maps to period h from set H using equation :

  h^t=mod(t,H)

- At each time slot, SP change the retail price.

Service Provider design

Utility

Source: ww.hydrogencarsnow.com

Energy Cost Function: c^t (.)

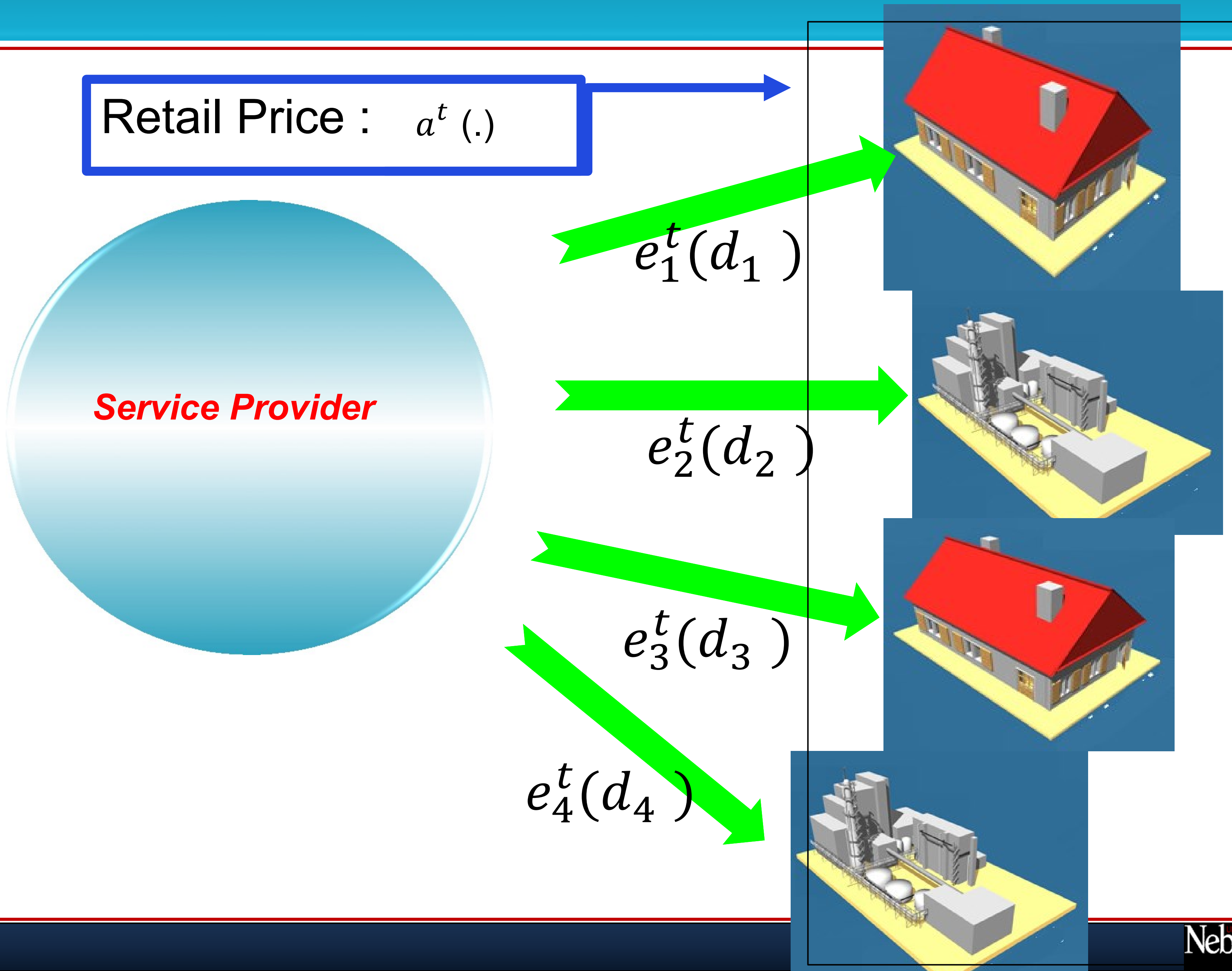
Service Provider

Energy consumption of customers : $\sum_{i \in \mathcal{I}} e_i^t$

The SP buys the electricity from the utility with the price of c^t (.) chosen from finite set C.

The price c^t is a function of time t and loads consumption $\sum_{i \in I} e_i^t$

SP and Customers



- In the microgrid system, the system provider determine the retail price function of the system (a^t can be a second order equation of customer consumption.

- The set of SP action, or retail price options are limited to a set A with n member $A=\{a_1,a_2,\ldots,a_n\}$.

- SP charge any customer $a^t(e_i^t)$, where e_i^t denote customer energy consumption at time t.

Model of Customer Response

Load model	
Condition	

1) Satisfy Load: e_i^t **;** Dissati d_i^t - e_i^t

2) Dissatisfy utility: u($d_i^t - e_i^t$)

3) $\phi_i^t(d_i^t, e_i^t) = u_i(d_i^t - e_i^t) + a^t(e_i^t)$

4) $d_i^{t+1} = \lambda_i(d_i^t - e_i^t) + \mu_i e_i^t + D_i^{t+1}$

Electricity Cost of Service Provider

- The SP buys the electricity from the utility with the price of c^t , chosen from finite set C.
- Transition probability from c^t at time t to c^{t+1}

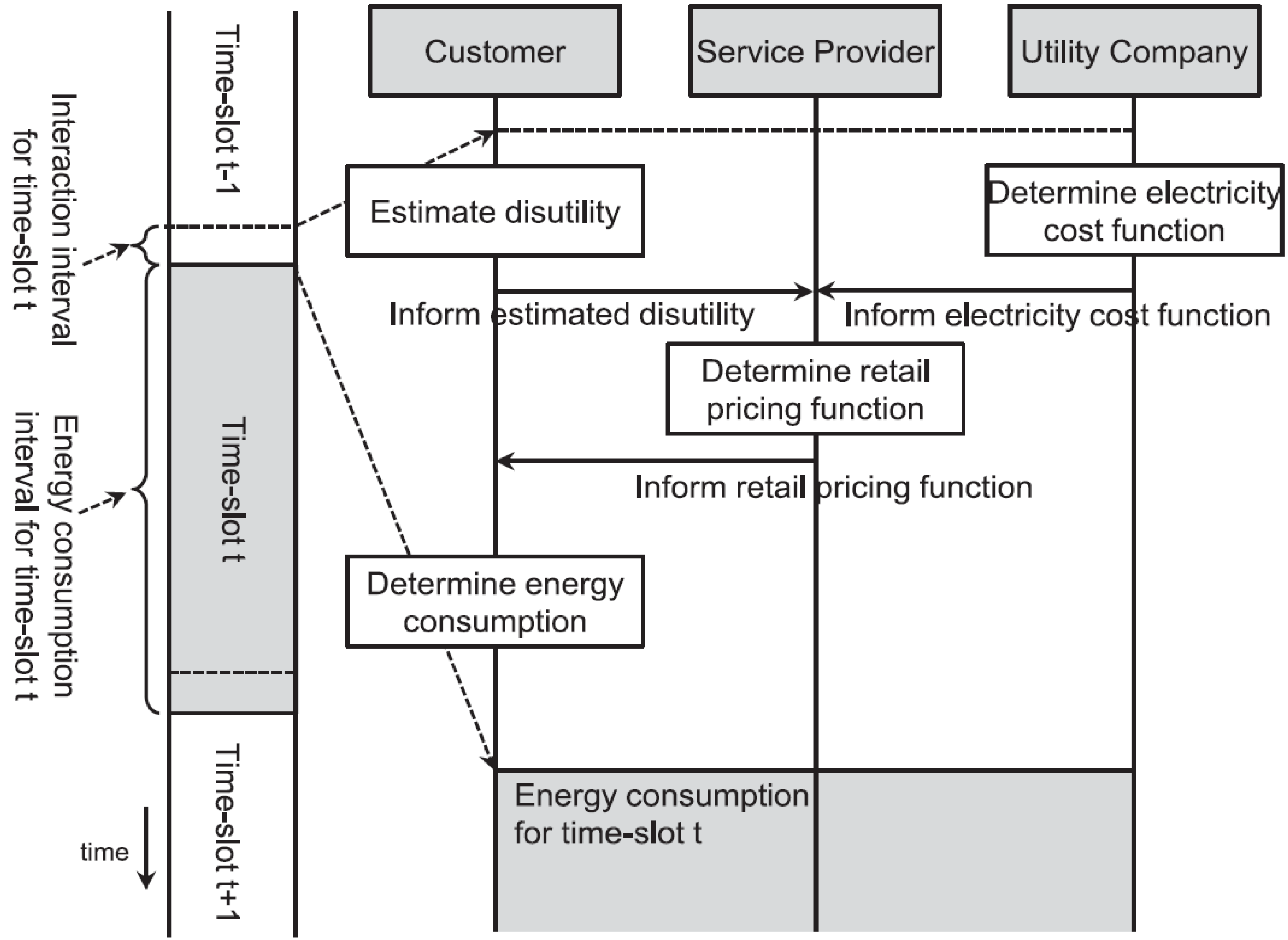
$$p_c(c^{t+1}|c^t, h^t)$$

- We denote the SP cost as:

$$\psi^t(\bar{d}^t, c^t, a^t) = c^t\left(\sum_{i\in\mathcal{I}} e_i^t\right) - \sum_{i\in\mathcal{I}} a^t(e_i^t)$$

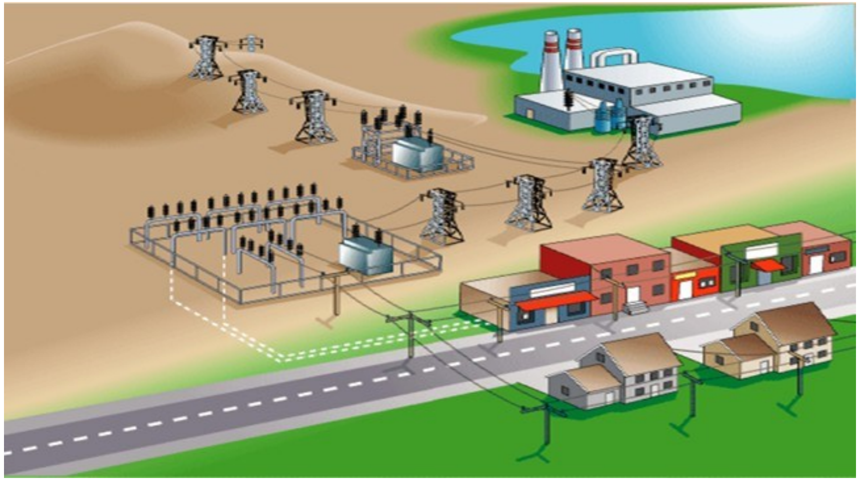
- where the first term denotes the electricity cost of the service provider and the second term denotes the service provider's revenue from selling energy to the customers.

Timeline of interaction among the microgrid



Overview

- Introduction & background
- Goal & objectives
- System Model
- **Reinforcement Learning at Service Provider**
- Energy Consumption-Based Approximated
- Virtual Experience for Accelerating Learning



Problem Formulation

- In this section, they first formulate a dynamic pricing problem in the framework of MDP.

- Then, by using reinforcement learning, They develop an efficient and fast dynamic pricing algorithm which does not require the information about the system dynamics and uncertainties.

MDP Formulation

- First consider customer as deterministic and myopic. Then for now, customer decision is to choose least possible cost:

$$e_i^t = \underset{0 \le e \le \min(e_i^{max}, d_i^t)}{\operatorname{argmin}} \phi_i^t(d_i^t, e)$$

- MDP problem was defined with
  - 1- Set of decision makers actions
  - 2- Set of system states
  - 3- System states transition
  - 4- System cost Function

❖ SP is a decision maker.

I. The SP actions is choosing a retail price from set A.

II. The microgrid states if function of customers demand vector, time and electricity price

$$s^t = (\bar{d}^t, h^t, c^t) \qquad \mathcal{S} = \prod_{i \in \mathcal{I}} \mathcal{D}_i \times \mathcal{H} \times \mathcal{C}$$

III. The transition from state $s^t = (\bar{d}^t, h^t, c^t)$ next state $s^{t+1} = (\bar{d}^{t+1}, h^{t+1}, c^{t+1})$

$$p_s\left(s^{t+1} \middle| s^t, a^t\right) = p_c\left(c^{t+1} \middle| c^t, h^t\right) \times \prod_{i \in \mathcal{I}} p_{d_i}\left(d_i^{t+1} \middle| d_i^t, h^t, a^t\right)$$

Continue

- System cost is defined as weighted sum of Sp and Customer cost:

$$r^t(s^t, a^t) = (1-\rho)\psi^t(\bar{d}^t, c^t, a^t) + \rho \sum_{i \in \mathcal{I}} \phi_i^t(d_i^t, e_i^t)$$

- In which choosing $\rho \in [0, 1]$ves priority on SP or customer cost.

Policy

- The objective is to find the stationary policy that:

1) maps states to action

$$\mathcal{S} \rightarrow \hat{\mathcal{A}}, \text{ i.e., } a^t = \pi(s^t)$$

2) minimize expected discount value

$$\mathbf{P}: \min_{\pi: \mathcal{S} \rightarrow \mathcal{A}} E\left[\sum_{t=0}^{\infty} (\gamma)^t r^t\left(s^t, \pi\left(s^t\right)\right)\right]$$

- The optimal stationary policy $\pi*$ can be well defined by using the optimal *action-value function* $Q* : S \times A \rightarrow$ R which satisfies the following Bellman optimality equation:

$$Q^*(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p\left(s' \middle| s,a\right) V^*\left(s'\right)$$

- In which $V^*(s')$ is optimal state-value function.

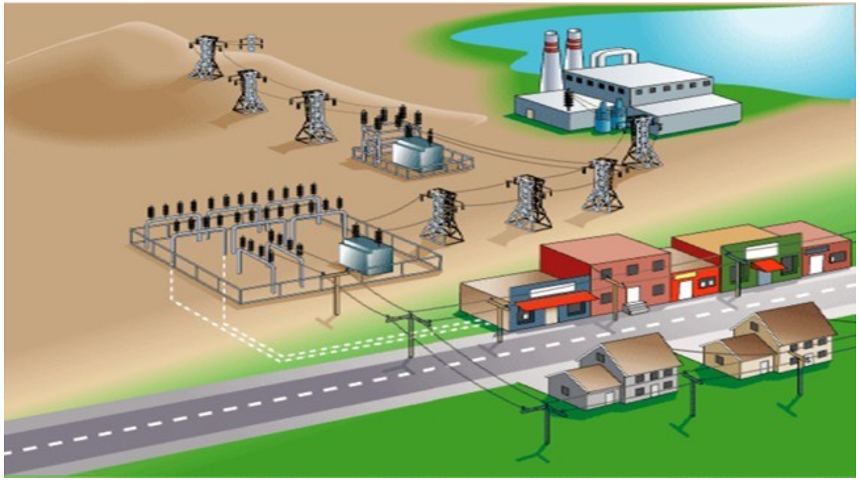
$$V^*\left(s'\right) = \min_{a \in \mathcal{A}} Q^*\left(s', a\right), \forall s \in \mathcal{S}$$

- Since *Q(s, a)* is the expected discounted system cost with action *a* in state *s*, we can obtain the optimal stationary policy as:

$$\pi^*(s) = \underset{a \in \mathcal{A}}{\operatorname{argmin}}\, Q^*(s,a)$$

Overview

- Introduction & background
- Goal & objectives
- System Model
- Reinforcement Learning at Service Provider
- **Energy Consumption-Based Approximated**
- Virtual Experience for Accelerating Learning



Energy Consumption-Based approximation State

- Two drawback Of their model:
  - 1- large number of states
  - 2- can't access customer states due to privacy

- To solve problems, they came up with new states:

$$x^t = \left(d^t_{app}, h^t, c^t\right) \in \mathcal{X}$$

$$\mathcal{X} = \mathcal{E} \times \mathcal{A} \times \mathcal{H} \times \mathcal{C}$$

Where $$d^t_{app} = \left(q_{\mathcal{E}}\left(\sum_{i \in \mathcal{I}} e_i^{t-1}\right), a^{t-1}\right)$$

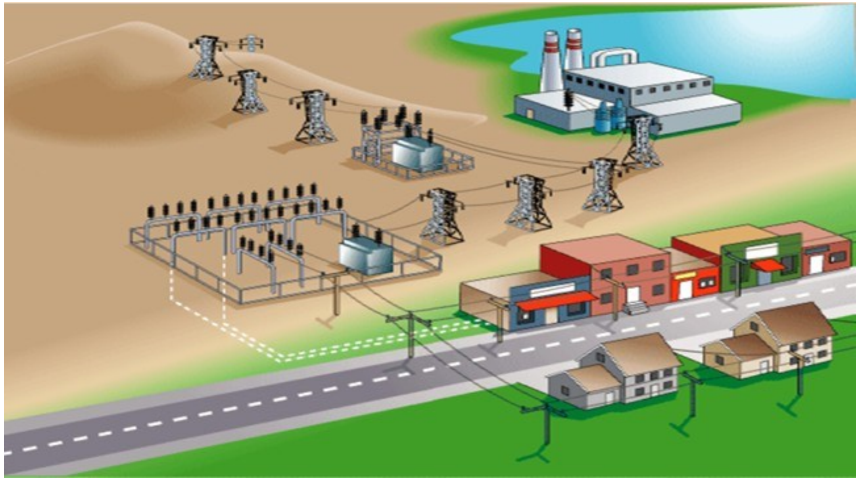
Energy Consumption-Based Approximation State

- Since D_i^t is set of independent variable, by the law of large number the $\sum_i {D_i^t}/{I}$ gose to expected value .

- In the practical microgrid system with a large number of customers, a $(\sum_{i \in \mathcal{I}} e_i^{t-1}, a^{t-1})$ provides enough information for the service provider to infer the D^t .

Overview

- Introduction & background
- Goal & objectives
- System Model
- Reinforcement Learning at Service Provider
- Energy Consumption-Based Approximated
- **Virtual Experience for Accelerating Learning**



Virtual Experience Definition

- Definition: Experience tuple is define as

$$\sigma^{t+1} = (x^t, a^t, r^t, x^{t+1})$$

- Update multiple state-action pair at each time.
  - set of equivalent tuple:

$$\sigma^{t+1} = (x^t, a^t, r^t, x^{t+1}) \quad \tilde{\sigma}^{t+1} = (\tilde{x}^t, \tilde{a}^t, \tilde{r}^t, \tilde{x}^{t+1})$$

  - If we have these two conditions:

$$p_X(\tilde{x}^{t+1} | \tilde{x}^t, \tilde{a}^t) = p_X(x^{t+1} | x^t, a^t), \ \tilde{a}^t = a^t$$

- Set of equivalent tuple which are statistically equivalent

$$\theta(\sigma^{t+1})$$

Virtual Experience in The System

- Assumption:
  - ✓ SP has a transition probability of $p_c(c^{t+1}|c^t, h^t)$

- Set of equivalent experience tuple:

$$\theta\left(\sigma^{t+1}\right) = \left\{ \tilde{\sigma}^{t+1} \middle| \begin{array}{l} \tilde{d}^t_{app} = d^t_{app}, \tilde{h}^t = h^t, \\ \tilde{a}^t = a^t, \tilde{r}^t = r(\tilde{c}^t), \\ p_c\left(\tilde{c}^{t+1} \middle| \tilde{c}^t, \tilde{h}^t\right) = p_c(c^{t+1}|c^t, h^t) \end{array} \right\}$$

Algorithm 1 Q-Learning Algorithm With Virtual Experience

Initialize Q arbitrarily, $t = 0$
for each time-slot t
  Choose a^t according to policy $\pi(x^t)$
  Take action a^t, observe system cost $r(x^t, a^t)$ and next state x^{t+1}
  Obtain experience tuple $\sigma^{t+1} = (x^t, a^t, r^t, x^{t+1})$
  Generate set of virtual experience tuples $\theta(\sigma^{t+1})$
  for each virtual experience tuple $\tilde{\sigma}^{t+1} \in \theta(\sigma^{t+1})$
    $v = r(x^t, a^t) + \gamma \max_{a' \in \mathcal{A}} Q(x^{t+1}, a')$
    $Q(x^t, a^t) \leftarrow (1 - \epsilon)Q(x^t, a^t) + \epsilon v$
  end
end

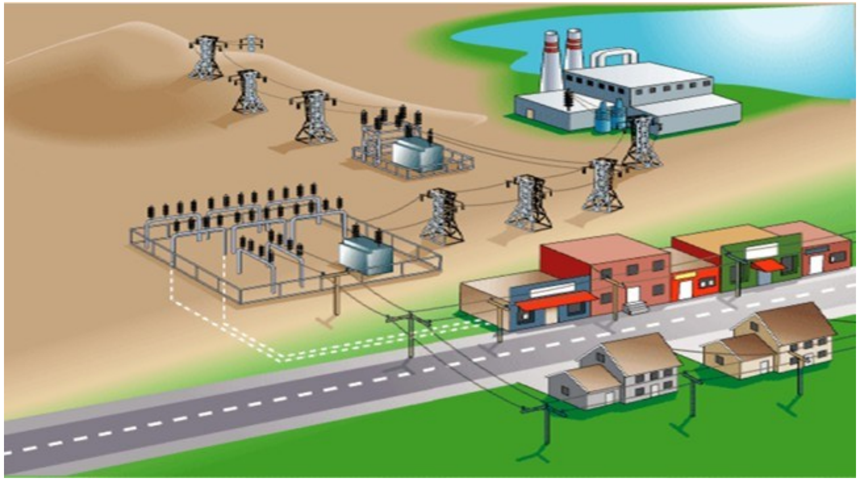
	Computational complexity	Memory complexity
Q-learning with original state	$O(\lvert\mathcal{A}\rvert)$	$O(\prod_{i\in\mathcal{I}} \lvert\mathcal{D}_i\rvert\lvert\mathcal{H}\rvert\lvert\mathcal{C}\rvert\lvert\mathcal{A}\rvert)$
Q-learning with EAS	$O(\lvert\mathcal{A}\rvert)$	$O(\lvert\mathcal{E}\rvert\lvert\mathcal{H}\rvert\lvert\mathcal{C}\rvert\lvert\mathcal{A}\rvert^2)$
Q-learning with EAS and virtual experience	$O(\lvert\theta(\hat{\sigma})\rvert\lvert\mathcal{A}\rvert)$	$O(\lvert\mathcal{E}\rvert\lvert\mathcal{H}\rvert\lvert\mathcal{C}\rvert\lvert\mathcal{A}\rvert^2)$

Recap

- Introduction to Microgrid
- SP and Load Model
- MDP formulation for SP to minimize system cost
- Presenting two methods for reducing space of Q-learning

Overview Day Two

- Reinforcement Learning at Customers
- Post Decision State Learning
- Numerical Results
- Conclusion



Customers Problem Formulation

- Q-learning for customer

1- Set of decision makers actions $e_i^t = a_i^t(d_i^t)$

- A set of finite energy consumption function $\mathcal{A}_i = \{a_{i,1}, a_{i,2}, \cdots, a_{i,A_i}\}$

2- Set of system states $s_i^t = \left(d_i^t, h^t, a^t\right) \in \mathcal{S}_i$

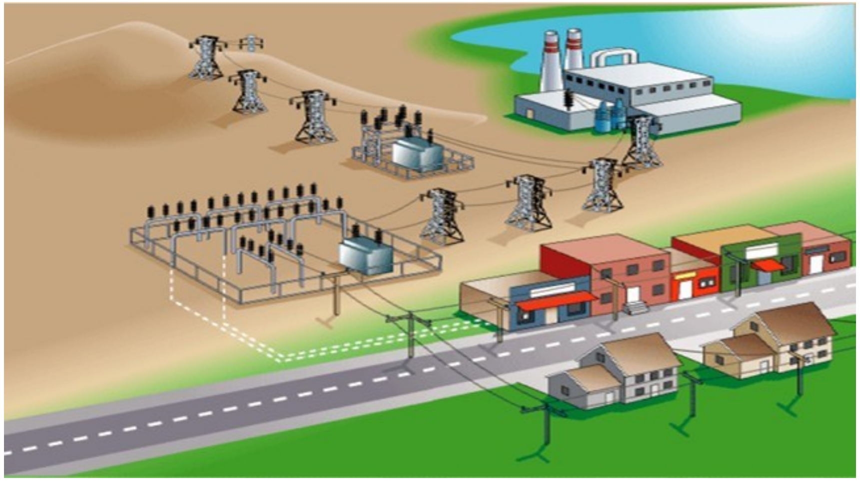
- A set of customer I's states $\mathcal{S}_i = \mathcal{D}_i \times \mathcal{H} \times \mathcal{A}$

3- System cost Function

$$\mathbf{P}_i : \min_{\pi_i : \mathcal{S}_i \to \mathcal{A}_i} E\left[\sum_{t=0}^{\infty} (\gamma)^t \phi_i^t\left(d_i^t, e_i^t\right)\right], \quad \forall i \in \mathcal{I}.$$

Overview Day Two

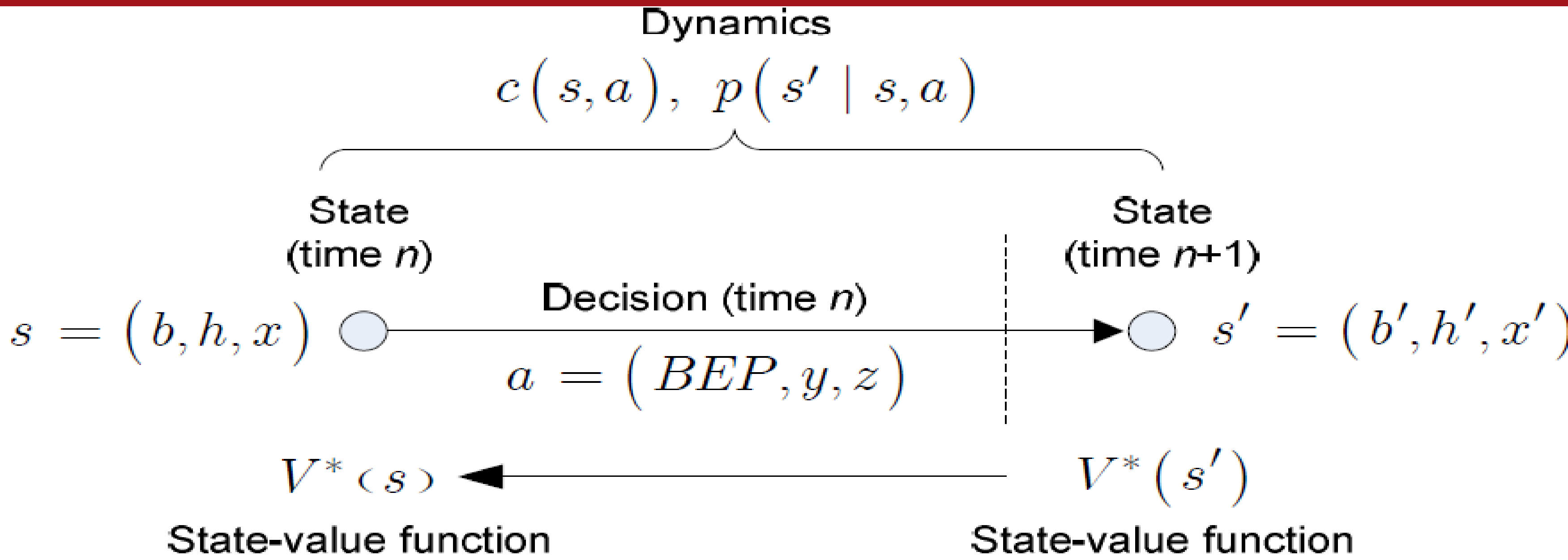
- Reinforcement Learning at Customers
- **Post Decision State Learning**
- Numerical Results
- Conclusion

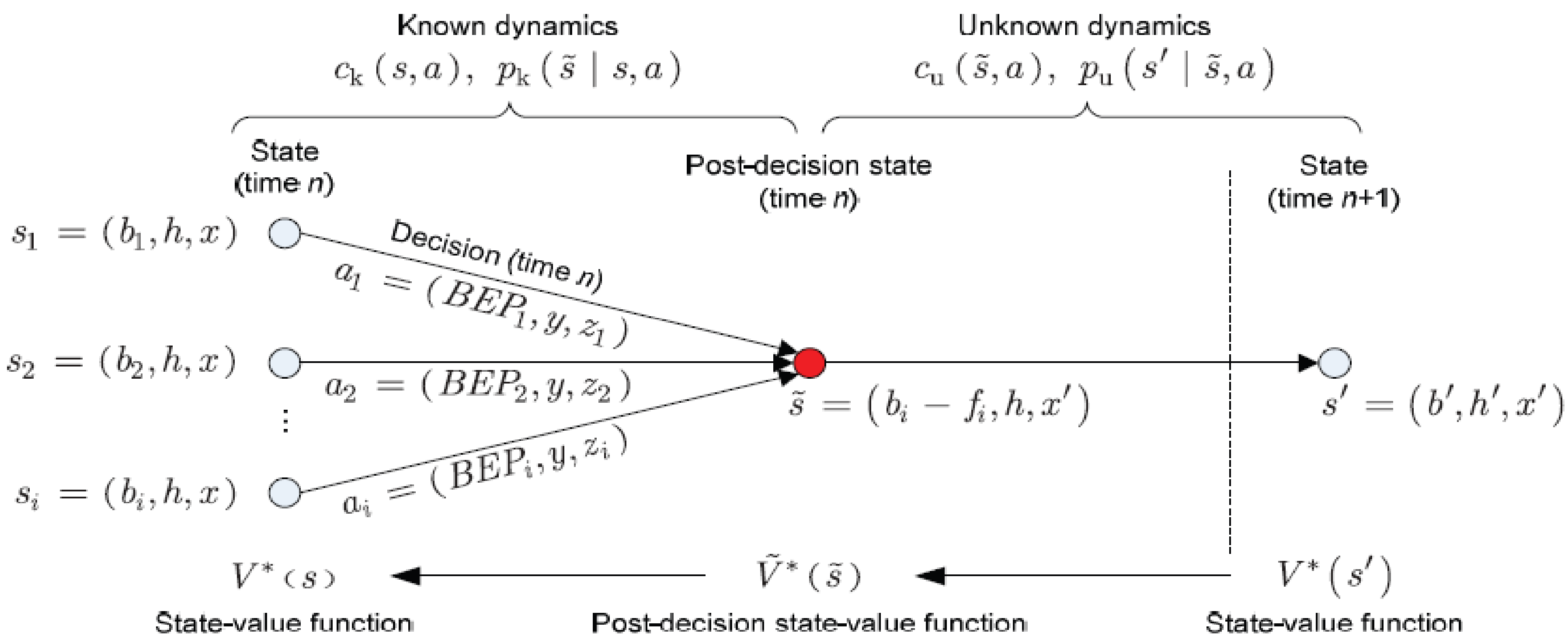


Post Decision State Learning Definition

- By introducing the PDS, we can factor the transition probability function into known and unknown components,
- Where the known component accounts for the transition from the current state to the PDS, $s \rightarrow \tilde{s}$
- And the unknown component accounts for the transition from the PDS to the next state $\tilde{s} \rightarrow s'$.

$$p(s' \mid s, a) = \sum_{\tilde{s}} p_{\mathrm{u}}(s' \mid \tilde{s}, a) \, p_{\mathrm{k}}(\tilde{s} \mid s, a)$$


Dynamics
$c(s,a),\ p(s' \mid s,a)$
State
(time n)
State
(time n+1)
Decision (time n)
$s = (b,h,x)$
$a = (BEP,y,z)$
$s' = (b',h',x')$
$V^*(s)$
$V^*(s')$
State-value function
State-value function


Known dynamics
$c_k(s,a),\ p_k(\tilde{s} \mid s,a)$
Unknown dynamics
$c_u(\tilde{s},a),\ p_u(s' \mid \tilde{s},a)$
State
(time n)
Post-decision state
(time n)
State
(time n+1)
$s_1 = (b_1,h,x)$
$s_2 = (b_2,h,x)$
$s_i = (b_i,h,x)$
Decision (time n)
$a_1 = (BEP_1,y,z_1)$
$a_2 = (BEP_2,y,z_2)$
$a_i = (BEP_i,y,z_i)$
$\tilde{s} = (b_i - f_i,h,x')$
$s' = (b',h',x')$
$V^*(s)$
$\tilde{V}^*(\tilde{s})$
$V^*(s')$
State-value function
Post-decision state-value function
State-value function

PDS and Conventional Q Relationship

- The optimal PDS value function and conventional Q learning relation

$$\tilde{V}^*(\tilde{s}) = c_u(\tilde{s}) + \gamma \sum_{s'} p_u(s' \mid \tilde{s}) V^*(s'),$$

$$V^*(s) = \min_{a \in \mathcal{A}} \left\{ c_k(s,a) + \sum_{\tilde{s}} p_k(\tilde{s} \mid s,a) \tilde{V}^*(\tilde{s}) \right\}.$$

Given the optimal PDS value function, the optimal policy can be computed as

$$\pi^*_{PDS}(s) = \min_{a \in \mathcal{A}} \left\{ c_k(s,a) + \sum_{\tilde{s}} p_k(\tilde{s} \mid s,a) \tilde{V}^*(\tilde{s}) \right\}$$

PDS for Learning

- **Proposition 1:** π^*_{PDS} *and* π^* *are equivalent.*

- Therefore, it can use the PDS value function to learn the optimal policy.

- While Q-learning uses a sample average of the action-value function to approximate Q* , PDS learning uses a sample average of the PDS value function to approximate $\tilde{V}^*$.

$$V^*(s) = \min_{a \in \mathcal{A}} \left\{ c_{\mathrm{k}}(s,a) + \sum_{\tilde{s}} p_{\mathrm{k}}(\tilde{s} \mid s,a) \tilde{V}^*(\tilde{s}) \right\}$$

Post Decision State Learning

Table 4.Post-decision state-based learning algorithm.

1.	**Initialize:** At time $n=0$, initialize the PDS value function $\tilde{V}^0$ as described in Section VI.E.
2.	**Take the greedy action:** At time n, take the greedy action $$a^n = \arg\min_{a\in\mathcal{A}}\left\{c_{\mathrm{k}}(s^n,a)+\sum_{\tilde{s}} p_{\mathrm{k}}(\tilde{s}\mid s^n,a)\tilde{V}^n(\tilde{s})\right\}.$$
3.	**Observe experience:** Observe the PDS experience tuple $\tilde{\sigma}^n=\left(s^n,a^n,\tilde{s}^n,c_{\mathrm{u}}^n,s^{n+1}\right)$.
4.	**Evaluate the state-value function:** Compute the value of state s^{n+1}: $$V^n\left(s^{n+1}\right)=\min_{a\in\mathcal{A}}\left\{c_{\mathrm{k}}(s^{n+1},a)+\sum_{\tilde{s}} p_{\mathrm{k}}\left(\tilde{s}\mid s^{n+1},a\right)\tilde{V}^n(\tilde{s})\right\}.$$
5.	**Update the PDS value function:** At time n, update the PDS value function using the information from steps 3 and 4: $$\tilde{V}^{n+1}(\tilde{s}^n)\leftarrow(1-\alpha^n)\tilde{V}^n(\tilde{s}^n)+\alpha^n\left[c_{\mathrm{u}}^n+\gamma V^n\left(s^{n+1}\right)\right]$$
6.	**Lagrange multiplier update:** Update the Lagrange multiplier μ using (34).
7.	**Repeat:** Update the time index, i.e. $n\leftarrow n+1$. Go to step 2.

Post Decision State Learning

- States definition
  - State at time-slot t: $s_i^t = (d_i^t, h^t, a^t)$
  - PDS at time-slot t: $\bar{s}_i^t = (d_i^{t+1}, h^{t+1}, a^t)$
  - State at time-slot $t+1$: $s_i^{t+1} = (d_i^{t+1}, h^{t+1}, a^{t+1})$

- Costumer i have information about its consumptions and its cost

$$p_k(\bar{s}_i^t | s_i^t, a_i^t) \;=\; p_d(d_i^{t+1} | d_i^t, a_i^t)$$

$$\phi\left(s_i^t | a_i^t\right) = \phi_i^t\left(d_i^t, e_i^t\right) = u_i\left(d_i^t - e_i^t\right) + a^t\left(e_i^t\right)$$

- State transition probability

$$p_{s_i}\left(s_i^{t+1} \middle| s_i^t, a_i^t\right) = \sum_{\bar{s}_i \in \mathcal{S}_i} p_k\left(\bar{s}_i \middle| s_i^t, a_i^t\right) p_u\left(s_i^{t+1} \middle| \bar{s}_i\right)$$

- State value function of customer I's state and PDS

$$\bar{V}^*(\bar{s}_i) = \gamma \sum_{s_i' \in \mathcal{S}_i} p_u\left(s_i' \middle| \bar{s}_i, a_i\right) V^*\left(s_i'\right)$$

$$V^*(s_i) = \min_{a_i \in \mathcal{A}_i} \left[\phi(s_i, a_i) + \sum_{\bar{s}_i \in \mathcal{S}_i} p_k\left(\bar{s}_i \middle| s_i, a_i\right) \bar{V}^*(\bar{s}_i) \right]$$

- PDS optimal policy

$$\bar{\pi}_i^*(s_i) = \min_{a_i \in \mathcal{A}_i} \left[\phi(s_i, a_i) + \sum_{\bar{s}_i \in \mathcal{S}_i} p_k(\bar{s}_i | s_i, a_i) \bar{V}^*(\bar{s}_i) \right].$$

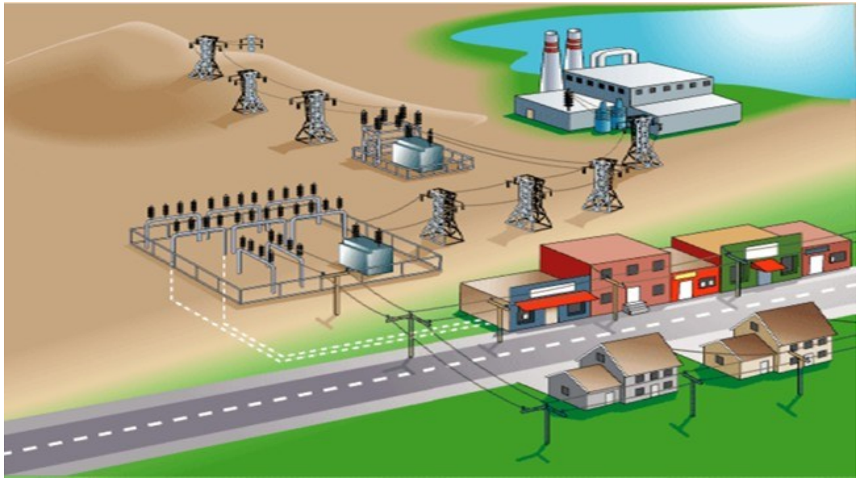
PDS Learning Algorithm

Algorithm 2 PDS Learning Algorithm

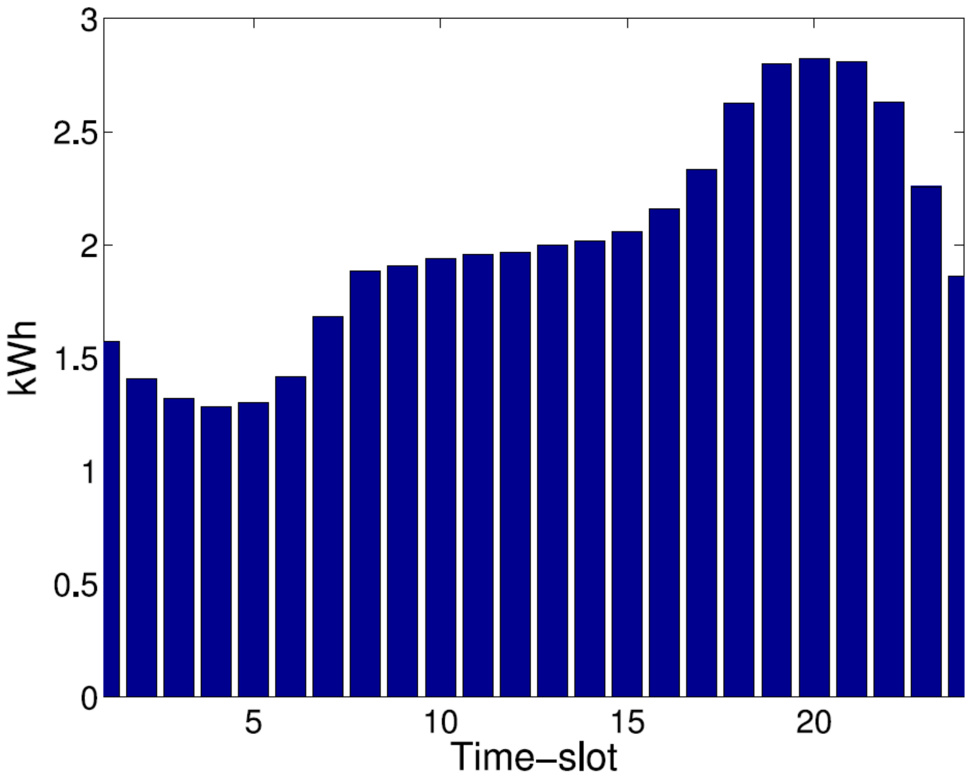
Initialize $\bar{V}$ arbitrarily, $t = 0$
for each time-slot t
    Choose a^t according to policy $\bar{\pi}(s_i^t)$
    Take action a^t, observe cost $\phi_i^t(d_i^t, e_i^t)$, PDS $\bar{s}_i^t$, and next state s_i^{t+1}
    $V(s_i^{t+1}) = \min_{a_i \in \mathcal{A}_i} \left[\phi(s_i^{t+1}, a_i) + \sum_{\bar{s}_i \in \mathcal{S}_i} p_k(\bar{s}_i | s_i^{t+1}, a_i)\bar{V}(\bar{s}_i)\right]$
    $\bar{V}(\bar{s}_i^t) \leftarrow (1-\epsilon)\bar{V}(\bar{s}_i^t) + \epsilon\gamma V(s_i^{t+1})$
end

Overview Day Two

- Reinforcement Learning at Customers
- Post Decision State Learning
- **Numerical Results**
- Conclusion



Numerical Results



Load profile

- H=24
- Total of 20 customers

Parameters

Backlog rate $\lambda_i = \lambda$ $\{0, 0.1, 0.2, 0.3, 0.4, 0.5\}$

$$u_i(d_i^t - e_i^t) = \kappa_i \times (d_i^t - e_i^t)^2$$

$\kappa = 0.1$

$$c^t\left(\sum_{i \in \mathcal{I}} e_i^t\right) = \alpha^t \times \sum_{i \in \mathcal{I}} e_i^t + \beta_{h^t}^t \times \left(\sum_{i \in \mathcal{I}} e_i^t\right)^2$$

$\alpha^t = 0.02$

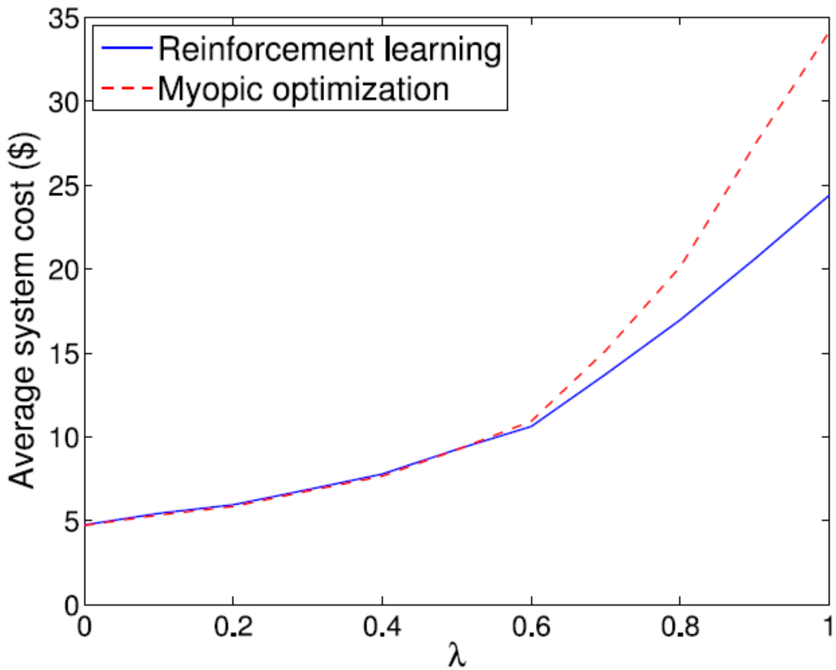
$$a^t(e_i^t) = \chi^t e_i^t$$

$\{0.2, 0.4, \cdots, 1.0\}$

Performance Comparison With Myopic Optimization

- Set cost coefficient $\rho = 0.5$
- Set Q-learning discount factor $\gamma=0$

1. The average system costs increase as λ increases in both pricing algorithms.
2. The performance gap between two algorithms increases as λ increases

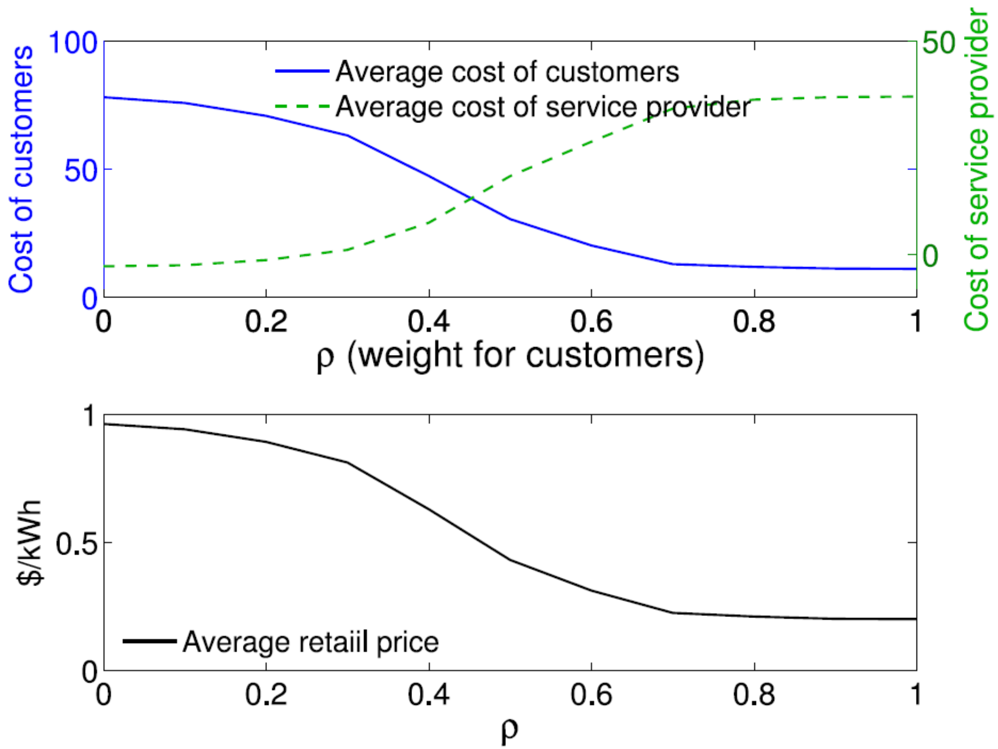


Performance comparison of our reinforcement learning algorithm and the myopic optimization algorithm varying λ

Impact of Weighting Factor ρ

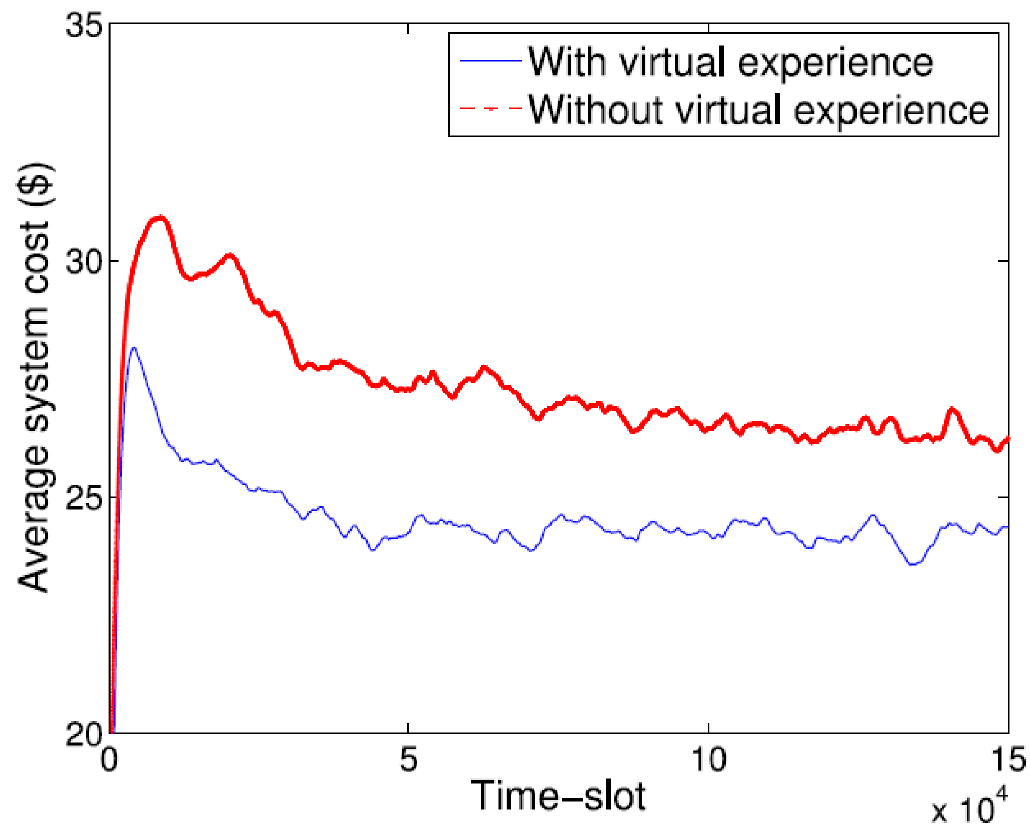
- Set $\lambda = 1$

2) As ρ increases, the cost of Customers decreases, and the cost of the service provider increases
3) As ρ increases, the service provider reduces the average retail price.



Impact of the weighting factor ρ on the performances of customers and service provider.

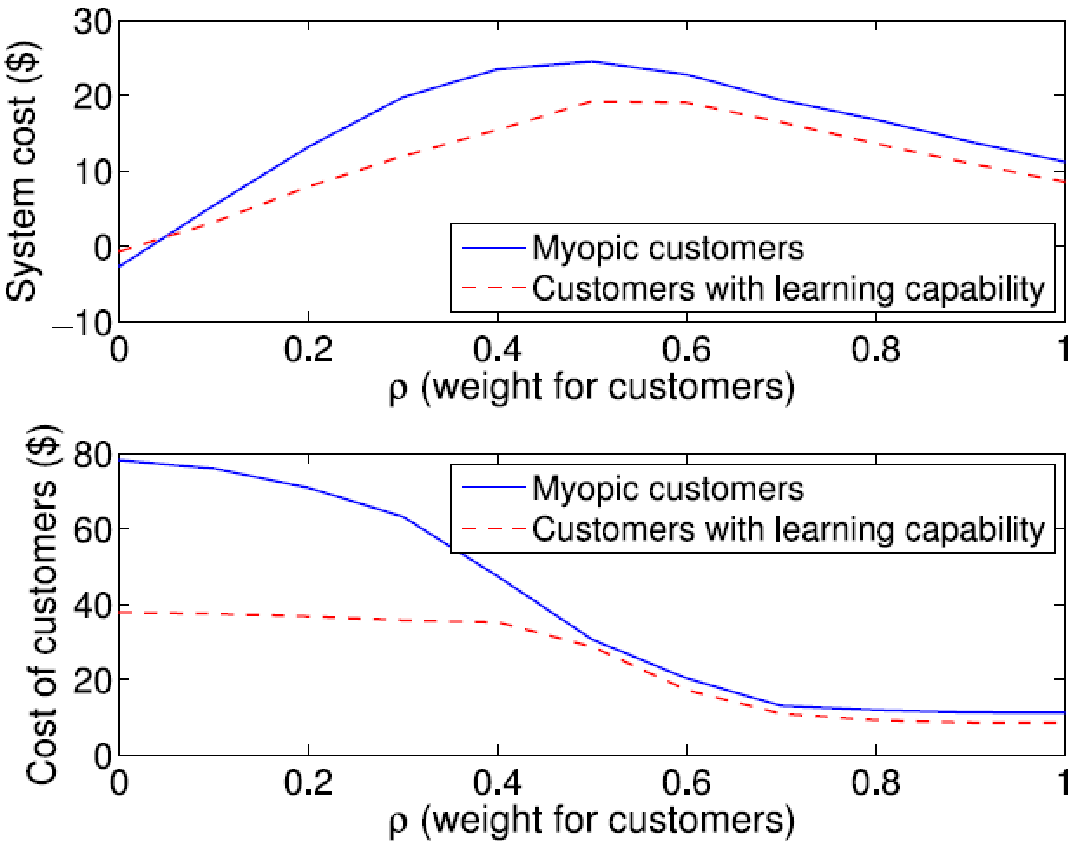
Virtual Experience Update



- Set $\lambda = 1$ and $\rho = 0.5$
- They Claimed :

We can observe that our algorithm with virtual experience provides a significantly improved learning speed compared to that of the conventional Q-learning algorithm.!!!

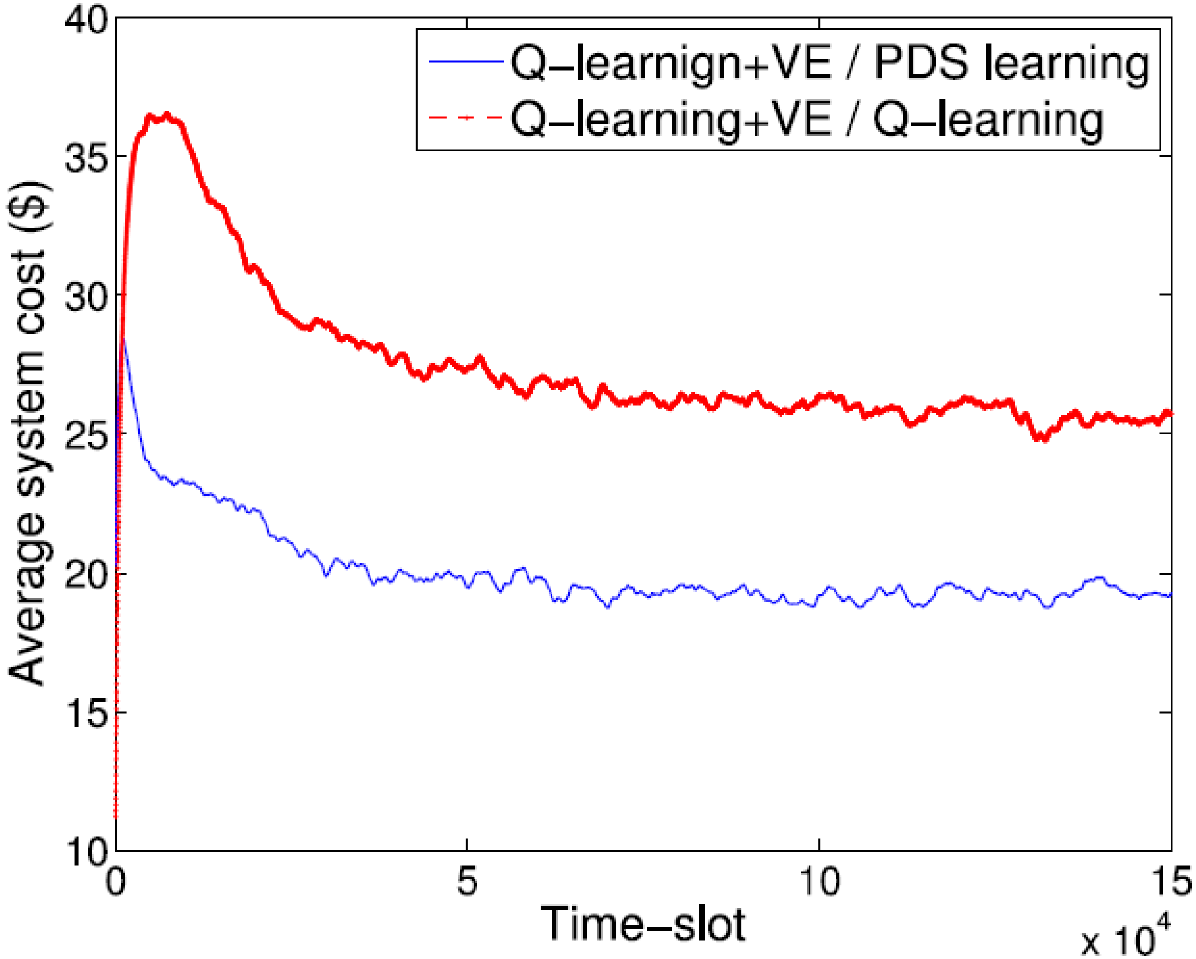
Customers With Learning Capability



- Set $\lambda = 1$
- lower average system cost
- lower customers' average
- Acceptable performance for $\rho = 0$

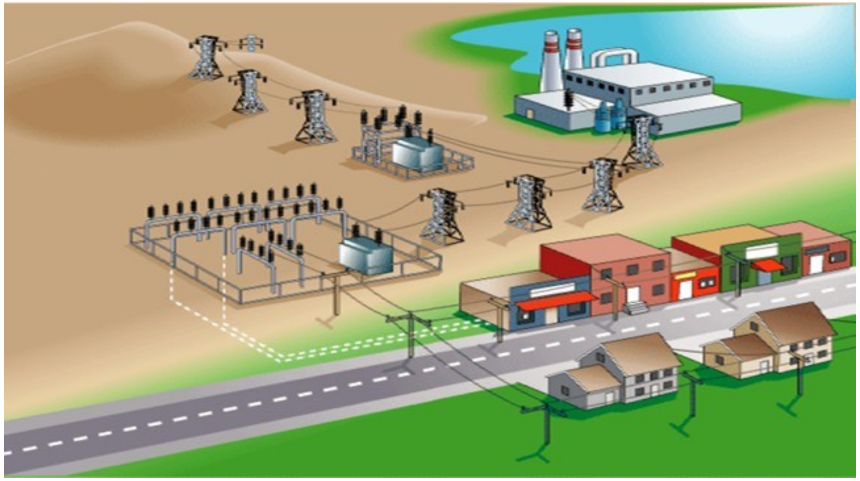
PDS Learning VS. Conventional Q Learning

- set $\lambda = 1$ and $\rho = 0.5$



Overview Day Two

- Reinforcement Learning at Customers
- Post Decision State Learning
- Numerical Results
- **Conclusion**



Conclusion

- This paper formulate an MDP problem, where the service provider observes the system states transmission and decide the retail electricity price to minimize the total expected cost of customer disutility.
- Each customer can decide its energy consumption based on the observed retail price aiming at minimizing its expected cost.
- The Q learning algorithm can be used to solve Bellman optimality equation when we don’t have a prior knowledge about system transition.
- The type of customers and their disutility function can change optimization results. Industrial loads may have a high dissatisfying utility.

Conclusion

- System with high λ has high system cost; High value for λ indicate that customers are shifting their extra loads to the next hour. Since they shift their loads every time, they have almost the same profile after demand response.
- The effect of virtual experience depends on the number of different cost function in the set C.
- Q-learning with the big λ show big system cost. However, when loads have the learning ability, the big λ will has less impact on the system cost.
- Three presented methods Including Energy Consumption Based Approximate State, Virtual Experience, and Post-Decision Learning had an effective response on accelerate the Q-learning algorithm.

- Customer learning capability, significantly reduced system and customers' cost.

Future Work

- Studying the strategic behaviors of the rational agents and its impact on the system performance.

- Considering the impact of various type of energy in dynamic pricing.

References

1. Kim, Byung-Gook, et al. "Dynamic pricing and energy consumption scheduling with reinforcement learning." *IEEE Transactions on Smart Grid,* (2016)
2. Mastronarde, Nicholas, and Mihaela van der Schaar. "Joint physical-layer and system-level power management for delay-sensitive wireless communications." *IEEE Transactions on Mobile Computing* 12.4 (2013): 694-709.
3. N. Mastronarde and M. van der Schaar, “Fast Reinforcement Learning for Energy-Efficient Wireless Communications,” technical report, http://arxiv.org/abs/1009.5773, 2012.

Thank you very much