Algorithms for Product Pricing and Energy Allocation in Energy Harvesting Sensor Networks

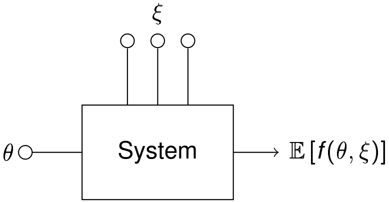
MSc(Engg.) Thesis Defense

Sindhu P R

Advisor: Prof. Shalabh Bhatnagar

Department of Computer Science and Automation
Indian Institute of Science, Bangalore

Stochastic System



- The output $f(\theta,\xi)$ depends on the control (or input) θ and a noise element ξ

Objective:

- Find input θ, which minimizes (or maximizes) $\mathbb{E}[f(\theta,\xi)]$

Overview

Overview

Product Demand and Price

- New product introduced into market faces demand uncertainty
  - Size of target population not known
  - Response to promotional campaigns cannot be ascertained
- Product price, quality and technolgy influence demand
- Dynamically varying price gives better profit compared to fixed pricing

Product Pricing Problem

- Product sold over the time interval $[0, T]$
- $X(t)$: Cumulative sales (or demand) upto time t
- $u(X(t))$: Manufacturing cost per unit of the product
- $P_r(t)$: Price of product at time t

Objective

Find $P_r(t)$ which maximizes expected cumulative profit over $[0, T]$

$$J^* = \max_{P_r(t)} \mathbb{E}\left[\int_0^T (P_r(t) - u(X(t)))\, X(t) dt \;\middle|\; X(0) = X_0\right]$$

Bass Diffusion Model

- Used to quantify how demand grows with time
- Driving forces for product diffusion - *mass-media* communication and *word-of-mouth* communication
- Change in cumulative sales $X(t)$ of a product is stated as

$$\frac{dX(t)}{dt} = p(M - X(t)) + \frac{qX(t)}{M}(M - X(t))$$

  - M: Market potential
  - p, q: Coefficients of innovation and imitation

Growth of Cumulative Demand

Change in $X(t)$ is modelled as a Stochastic Differential Equation (SDE)-

$$dX(t) = (M - X(t))\left(p + \frac{qX(t)}{M}\right)(1 - \gamma P_r(t))dt + \sigma(X(t))dW(t)$$

- γ determines the extent of influence of $P_r(t)$ on the product diffusion
- $\sigma(.)$ - diffusion term, $\{W(t),\ t \geq 0\}$ - standard Brownian motion.

Growth of Cumulative Demand

Change in $X(t)$ is modelled as a Stochastic Differential Equation (SDE)-

$$dX(t) = (M - X(t))\left(p + \frac{qX(t)}{M}\right)(1 - \gamma P_r(t))dt + \sigma(X(t))dW(t)$$

- γ determines the extent of influence of $P_r(t)$ on the product diffusion
- $\sigma(.)$ - diffusion term, $\{W(t),\ t \geq 0\}$ - standard Brownian motion.

Approach

- Discretize SDE and solve in discrete time setting
- Formulate the problem in the setting of simulation optimization
- Tune the (discrete) price trajectory to obtain optimum performance

SDE Discretization

- $T = Nh$ for some $N > 1$, $0 < h < 1$

SDE Discretization

- $T = Nh$ for some $N > 1$, $0 < h < 1$
- Cumulative demand at decision time j is X_j and $X_j \equiv X(jh)$

SDE Discretization

- $T = Nh$ for some $N > 1$, $0 < h < 1$
- Cumulative demand at decision time j is X_j and $X_j \equiv X(jh)$
- Price set at j is $P_{r(j)} \equiv P_r(jh)$

SDE Discretization

- $T = Nh$ for some $N > 1$, $0 < h < 1$
- Cumulative demand at decision time j is X_j and $X_j \equiv X(jh)$
- Price set at j is $P_{r(j)} \equiv P_r(jh)$
- Drift term, $b\left(X_j, P_{r(j)}\right) = (M - X_j)(p + \frac{q}{M}X_j)\left(1 - \gamma P_{r(j)}\right)$

SDE Discretization

- $T = Nh$ for some $N > 1$, $0 < h < 1$
- Cumulative demand at decision time j is X_j and $X_j \equiv X(jh)$
- Price set at j is $P_{r(j)} \equiv P_r(jh)$
- Drift term, $b\left(X_j, P_{r(j)}\right) = (M - X_j)(p + \frac{q}{M}X_j)\left(1 - \gamma P_{r(j)}\right)$
- $g\left(X_j, P_{r(j)}\right) = \left(u(X_j) - P_{r(j)}\right)$

SDE Discretization

- $T = Nh$ for some $N > 1$, $0 < h < 1$
- Cumulative demand at decision time j is X_j and $X_j \equiv X(jh)$
- Price set at j is $P_{r(j)} \equiv P_r(jh)$
- Drift term, $b\left(X_j, P_{r(j)}\right) = (M - X_j)(p + \frac{q}{M}X_j)\left(1 - \gamma P_{r(j)}\right)$
- $g\left(X_j, P_{r(j)}\right) = \left(u(X_j) - P_{r(j)}\right)$
- $X_{-1} = 0$ and X_0 is assumed to be known

SDE Discretization

- $T = Nh$ for some $N > 1$, $0 < h < 1$
- Cumulative demand at decision time j is X_j and $X_j \equiv X(jh)$
- Price set at j is $P_{r(j)} \equiv P_r(jh)$
- Drift term, $b\left(X_j, P_{r(j)}\right) = (M - X_j)(p + \frac{q}{M}X_j)\left(1 - \gamma P_{r(j)}\right)$
- $g\left(X_j, P_{r(j)}\right) = \left(u(X_j) - P_{r(j)}\right)$
- $X_{-1} = 0$ and X_0 is assumed to be known
- $\sigma(X_j) \equiv \sigma(jh)$

SDE Discretization

- $T = Nh$ for some $N > 1$, $0 < h < 1$
- Cumulative demand at decision time j is X_j and $X_j \equiv X(jh)$
- Price set at j is $P_{r(j)} \equiv P_r(jh)$
- Drift term, $b\left(X_j, P_{r(j)}\right) = (M - X_j)(p + \frac{q}{M}X_j)\left(1 - \gamma P_{r(j)}\right)$
- $g\left(X_j, P_{r(j)}\right) = \left(u(X_j) - P_{r(j)}\right)$
- $X_{-1} = 0$ and X_0 is assumed to be known
- $\sigma(X_j) \equiv \sigma(jh)$
- $\sigma'(\cdot)$ is the derivative of $\sigma(\cdot)$

SDE Discretization (contd.)

Discretized SDE

$$X_{j+1} = X_j + b(X_j, P_{r(j)})h + \sigma(X_j)\sqrt{h}Z_{j+1} + \frac{1}{2}\sigma'(X_j)\sigma(X_j)h(Z_{j+1}^2 - 1)$$

- For $0 \leq j \leq N-1$, Z_{j+1} are independent $N(0,1)$ distributed samples

SDE Discretization (contd.)

Discretized SDE

$$X_{j+1} = X_j + b(X_j, P_{r(j)})h + \sigma(X_j)\sqrt{h}Z_{j+1} + \frac{1}{2}\sigma'(X_j)\sigma(X_j)h(Z_{j+1}^2 - 1)$$

- For $0 \leq j \leq N-1$, Z_{j+1} are independent $N(0,1)$ distributed samples

Discretized Objective Function

$$J\left(P_{r(0)}, P_{r(1)}, \cdots, P_{r(N-1)}\right) = \mathbb{E}\left[\sum_{j=0}^{N-1} g(X_j, P_{r(j)})(X_j - X_{j-1}) \;\middle|\; X_0\right]$$

Optimal Price Vector

Objective: Find $\mathbf{P}_\mathbf{r}^* = (P^*_{r(0)}, P^*_{r(1)}, \cdots, P^*_{r(N-1)})^\top$ where,

$$\{P^*_{r(0)}, P^*_{r(1)}, \cdots, P^*_{r(N-1)}\} = \underset{\{P_{r(0)},\cdots,P_{r(N-1)}\}}{\arg\min} J_{X_0}(P_{r(0)}, P_{r(1)}, \cdots, P_{r(N-1)})$$

- Price set at stage k influences the demand in that stage as well as in future stages
- The initial price influences the single-stage costs of all stages

Gradient Search Using Simulations

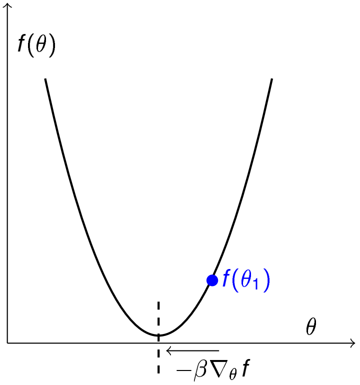
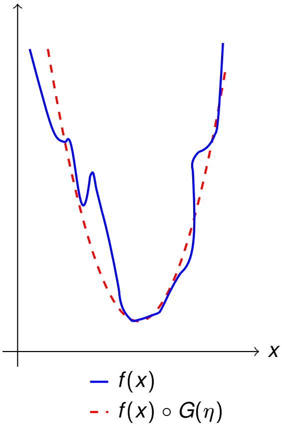


Figure: Gradient Search

- Need to minimize $f(\theta)$
- Find $\nabla_\theta f$
- Update θ in the negative direction of gradient
- In pricing problem, $f(\theta) = J(P_{r(0)}, P_{r(1)}, \cdots, P_{r(N-1)})$

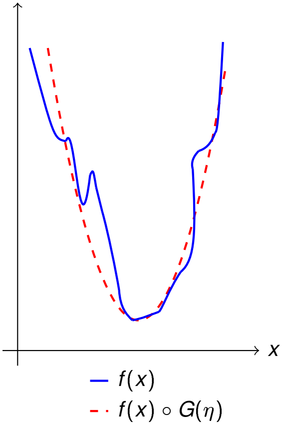
Smoothed Functional Technique

- Smooth $\nabla J(P_{r(0)}, P_{r(1)}, \cdots, P_{r(N-1)})$ by convolution with $N-$dimensional Gaussian density function, $G(\boldsymbol{\eta})$, $\boldsymbol{\eta} \in \mathbb{R}^N$
- $\beta > 0$ is a small constant



Smoothed Functional Technique

- Smooth $\nabla J(P_{r(0)}, P_{r(1)}, \cdots, P_{r(N-1)})$ by convolution with $N-$dimensional Gaussian density function, $G(\boldsymbol{\eta})$, $\boldsymbol{\eta} \in \mathbb{R}^N$
- $\beta > 0$ is a small constant



The gradient estimate is

$$D_\beta J_{x_0}(\mathbf{P_r}) = \frac{1}{\beta}\,\mathbb{E}\left[\frac{\boldsymbol{\eta}}{2}(J_{x_0}(\mathbf{P_r} + \beta\boldsymbol{\eta}) - J_{x_0}(\mathbf{P_r} - \beta\boldsymbol{\eta})) \mid \mathbf{P_r}\right]$$

Gradient Estimate

- In our simulation-based algorithm, expectation is replaced by samples obtained from simulation.
- For large L, small β,

$$\nabla J_{X_0}(\mathbf{P_r}) = \frac{1}{\beta}\frac{1}{L}\left[\sum_{n=1}^{L}\frac{\boldsymbol{\eta}(n)}{2}\left(J_{X_0}(\mathbf{P_r}+\beta\boldsymbol{\eta}(n)) - J_{X_0}(\mathbf{P_r}-\beta\boldsymbol{\eta}(n))\right)\right]$$

- $\| D_\beta J_{X_0}(\mathbf{P_r}) - \nabla J_{X_0}(\mathbf{P_r}) \| \to 0$ as $\beta \to 0$ and $L \to \infty$

Two-timescale Smoothed Functional (SF) Algorithm

1. Generate $\boldsymbol{\eta}(n)$
2. Get $\mathbf{P_r^+} = \mathbf{P_r}(n) + \beta\boldsymbol{\eta}(n)$ and $\mathbf{P_r^-} = \mathbf{P_r}(n) - \beta\boldsymbol{\eta}(n)$
3. Compute $\mathbf{X}^+$ and $\mathbf{X}^-$
4. Compute $\delta_j^+ = g(X_i^+(n), P_{r(j)}^+(n))(X_i^+(n) - X_{i-1}^+(n))$,
   $\delta_j^- = g(X_i^-(n), P_{r(j)}^-(n))(X_i^-(n) - X_{i-1}^-(n))$, $0 \le j \le N-1$
5. Compute $\delta_j = \frac{\eta_j(n)}{2\beta}\sum_{i=j}^{N}\left(\delta_j^+ - \delta_j^-\right)$, $0 \le j \le N-1$, $\boldsymbol{\delta} = (\delta_0, \ldots, \delta_{N-1})^\top$
6. Gradient estimate: $\mathbf{Y}(n+1) = (1 - c(n))\mathbf{Y}(n) + c(n)\boldsymbol{\delta}$
7. Update price: $\mathbf{P_r}(n+1) = \mathbf{P_r}(n) - \alpha(n)\mathbf{Y}(n+1)$

Performance of SF Algorithm

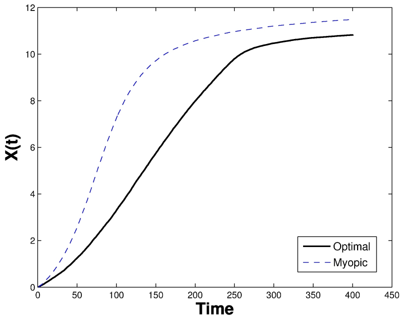


Figure: Evolution of cumulative demand

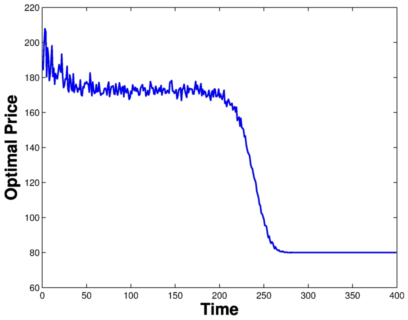


Figure: Optimal Price Trajectory

Simulation Parameters:

- M = 10, $\sigma = 0.1$, $u(X_j) = 100 - 0.2X_j$, $p = 0.02$, $q = 0.5$ and $\gamma = 0.005$
- $L = 50,000$, $N = 400$, $h = 0.25$, $T = 100$
- Myopic price is 140
- Objective function values: myopic $= -472.6 \pm 0.9$, optimal = -670 ± 6

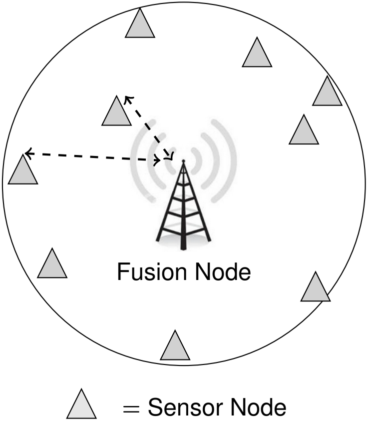
Summary

- We proposed a simulation-based algorithm for pricing a product over the interval $[0, T]$, when demand is uncertain
- The algorithm utilised smoothed functional gradient estimator to find the optimal pricing policy

Future Directions:

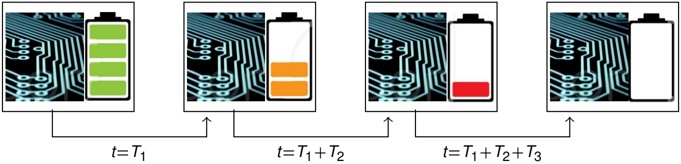
- Simulation based optimization methods for products in the presence of competing products

Sensor Network



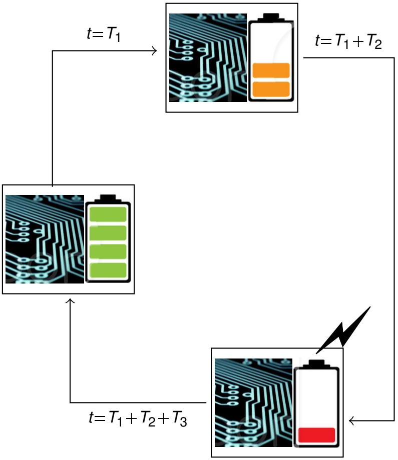
- Group of autonomous sensing nodes
- Sensed data is transmitted to a fusion node
- Fusion node obtains data from multiple nodes and processes it

Conventional Sensor Node



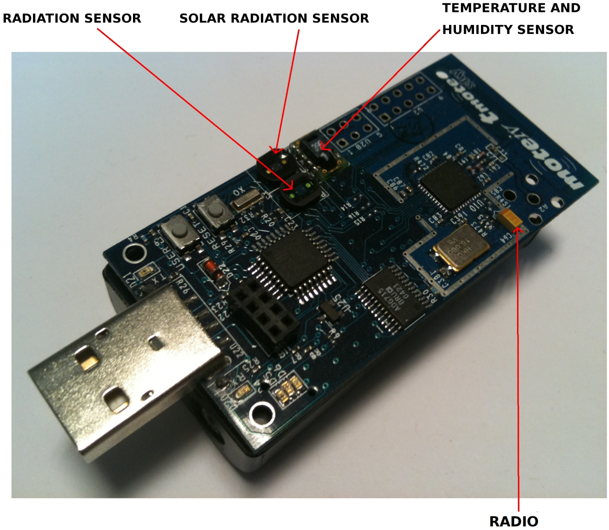
- Node is powered by a non-rechargeable battery
- Energy is used for sensing and transmission
- Node stops operating once battery is exhausted

Energy Harvesting Sensor Node



- Node replenishes energy, by harvesting energy from the environment
- Energy harvested is location and time dependent
- Techniques required to prevent energy starvation in nodes

Sensor Mote


RADIATION SENSOR
SOLAR RADIATION SENSOR
TEMPERATURE AND
HUMIDITY SENSOR
RADIO

Energy Sharing Problem

- System comprises of n sensors and a common energy harvesting (EH) source
- Each sensor has a finite data buffer in which sensed data is stored
- The EH source has a finite buffer to store the harvested energy
- Nodes share the energy harvested by the EH source

Energy Sharing Problem

- System comprises of n sensors and a common energy harvesting (EH) source
- Each sensor has a finite data buffer in which sensed data is stored
- The EH source has a finite buffer to store the harvested energy
- Nodes share the energy harvested by the EH source

Aim

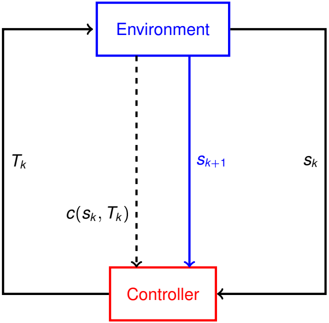
To develop energy sharing algorithms which minimize the average delay in transmission of data from the nodes

Approach

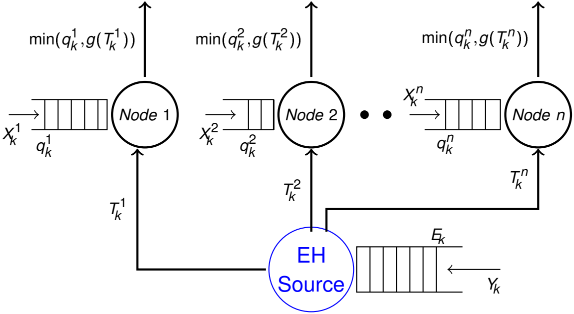
- Model the problem as an infinite-horizon average cost Markov decision process (MDP)
- Control decisions based on:
  - amount of data in each of the sensors
  - energy available in the EH source
- Develop Reinforcement Learning (RL) algorithms which
  - optimally allocate energy to every sensor
  - work without requiring a system model

Reinforcement Learning

- Environment: probabilistically evolves over states
- Reinforcement: the cost incurred after performing an action in a state
- Policy: determines which action to be taken at every state
- Goal: learn a policy which minimizes the long-run average cost
- The RL controller learns the policy which achieves this goal using trial-and-error process

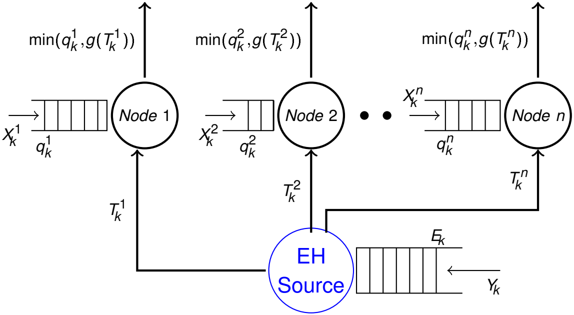


System Model



- Slotted, discrete-time model for the environment
- D_{MAX}: Data buffer size of each sensor
- E_{MAX}: Energy buffer size of EH source

System Model



- In slot k,
  - E_k: Energy buffer level, q_k^i, $1 \leq i \leq n$: Data buffer level of node i
  - EH source harvests Y_k units of energy
  - Source provides T_k^i units of energy to node i, $1 \leq i \leq n$
  - Node i generates X_k^i bits of data and transmits $g(T_k^i)$ bits of data, $1 \leq i \leq n$

System Model (contd.)

- Data buffer queue lengths evolve as

$$q^i_{k+1} = (q^i_k - g(T^i_k))^+ + X^i_k \quad 1 \le i \le n$$

  where $(q^i_k - g(T^i_k))^+ = \max\left((q^i_k - g(T^i_k)), 0\right)$

- Energy buffer queue length evolves as

$$E_{k+1} = \left(E_k - \sum_{i=1}^{n} T^i_k\right) + Y_k$$

Model Assumptions

1. Generated data at time $k+1$, $X_{k+1} \triangleq (X_{k+1}^1, \ldots, X_{k+1}^n)$ evolves as a jointly Markov process:

$$X_{k+1} = f^1(X_k, W_k), \qquad k \geq 0$$

   - f^1 is some arbitrary vector valued function with n components
   - $\{W_k, k \geq 1\}$ is a noise sequence with probability distribution $P(W_k \mid X_k)$ depending on X_k.

2. Energy arrival process evolves as:

$$Y_{k+1} = f^2(Y_k, V_k), \qquad k \geq 0$$

   - f^2 is some scalar vector valued function
   - $\{V_k, k \geq 1\}$ is a noise sequence with probability distribution $P(V_k \mid Y_k)$ depending on Y_k.

MDP Formulation

- State: $s_k = \left(q_k^1, q_k^2, \ldots, q_k^n, E_k, X_{k-1}^1, \ldots, X_{k-1}^n, Y_{k-1}\right)$, $s_k \in S$
- Action: $T(s_k) = \left(T^1(s_k), T^2(s_k), \ldots, T^n(s_k)\right) \in A$, specifies the number of energy bits to be given to each node at time k
- Stationary Policy $\pi = \{T, T, \ldots\}$
- Single Stage Cost: $c(s_k, T(s_k)) = \sum_{i=1}^{n} \left(q_k^i - g(T^i(s_k))\right)^+$.
- Long-run average cost per step λ^π: $\lim_{m \to \infty} \mathbb{E}\left[\frac{1}{m} \sum_{k=0}^{m-1} c(s_k, T(s_k))\right]$

MDP Formulation

- State: $s_k = \left(q_k^1, q_k^2, \ldots, q_k^n, E_k, X_{k-1}^1, \ldots, X_{k-1}^n, Y_{k-1}\right)$, $s_k \in S$
- Action: $T(s_k) = \left(T^1(s_k), T^2(s_k), \ldots, T^n(s_k)\right) \in A$, specifies the number of energy bits to be given to each node at time k
- Stationary Policy $\pi = \{T, T, \ldots\}$
- Single Stage Cost: $c(s_k, T(s_k)) = \sum_{i=1}^{n} \left(q_k^i - g(T^i(s_k))\right)^+$.
- Long-run average cost per step λ^π: $\lim_{m \to \infty} \mathbb{E}\left[\frac{1}{m} \sum_{k=0}^{m-1} c(s_k, T(s_k))\right]$

Objective

Find a stationary optimal policy $\pi^* = (T^*, T^*, \ldots)$ which minimizes the sum of remaining data queue lengths of all buffers. This policy also minimizes the mean delay of data transmission.

Q-learning

- Q-value of a state-action pair (s, a) is $Q(s, a)$
- Initially, $Q(s, a) = 0 \ \forall s \in S, a \in A$
- i_r: Reference state
- Simulate the MDP for a large number of iterations
- At iteration j, for the state-action pair visited during simulation, Q value is updated:

$$Q_{j+1}(s, a) = (1 - \alpha(j))Q_j(s, a) + \alpha(j) \times \left(c(s, a) + \min_{b \in A(s')} Q_j(s', b) - \min_{u \in A(i_r)} Q_j(i_r, u) \right)$$

- Step-size sequence $\alpha(j) > 0, \ \forall j \geq 0$ satisfies the following conditions:

$$\sum_j \alpha(j) = \infty \text{ and } \sum_j \alpha^2(j) < \infty.$$

Q-Learning: Exploration Mechanisms

- ϵ-greedy Method: At state s, in iteration j select random action with probability ϵ and greedy action with probability $1 - \epsilon$
- UCB Method:
  - $N_s(j)$: number of times state s is visited until time j
  - $N_{s,a}(j)$ be the number of times action a is picked in state s upto time j
  - Q-value of state-action pair (s, a) at time j is $Q_j(s, a)$
  - Action for the current state s is chosen using the following rule

$$a' = \arg\max_{a \in A(s)} \left(-Q_j(s, a) + \beta \sqrt{\frac{\ln N_s(j)}{N_{s,a}(j)}} \right)$$

Q-learning: Optimal Policy

- Q-learning algorithm converges to optimal value function Q^*
- The optimal action a^* for state s is obtained by

$$a^* = \arg\min_{a' \in A(s)} Q^*(s, a')$$

- Optimal Policy: $T^*(s) = a^*, \forall s \in S$

Q-learning: Optimal Policy

- Q-learning algorithm converges to optimal value function Q^*
- The optimal action a^* for state s is obtained by

$$a^* = \underset{a' \in A(s)}{\arg\min}\, Q^*(s, a')$$

- Optimal Policy: $T^*(s) = a^*, \forall s \in S$

Issues

- Need lookup table to store and update Q-value for every (s, a) tuple
- Computationally expensive: With $n = 2$, $|S \times A| \approx 30^6$
- Condition exacerbated when there are more number of sensors, $n = 4$, $|S \times A| \approx 30^{10}$

Approximation Approaches

1. State and action space aggregation
   - Concept is to cluster states and actions based on monotonicity property of value function
   - Define a Q-value for aggregate state-action pair
   - Results in cardinality reduction
2. Policy Approximation
   - Need to search in the space of all policies
   - Aim is to find a near optimal stationary randomized policy

Property of Value Function

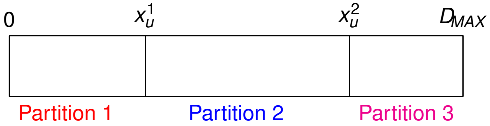
- Consider two nodes and a source
- $H^*(q_1, q_2, E)$ be the differential value of state (q_1, q_2, E)
- $q < q^L \leq D_{MAX}$ and $E_{MAX} \geq E^L > E$. Then H^* is monotonic

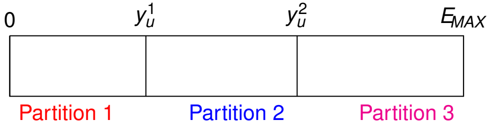
$$H^*(q_1, q_2, E) \leq H^*(q_1^L, q_2, E)$$

$$H^*(q_1, q_2, E) \geq H^*(q_1, q_2, E^L).$$

- Value function of the clustered state will be the average of the value function of the individual states
- Value function of aggregated state will be close to the value function of the unaggregated state if the difference between values of states in a cluster is small

State and Action Space Aggregation





- Data buffer partitions: sets $d_1, d_2, \ldots, d_s$, where $d_i = (x_L^i, x_U^i)$
- Energy buffer partitions: sets $e_1, e_2, \ldots, e_r$, where $e_j = (y_L^j, y_U^j)$

$$0 = x_L^1 < x_U^1 < x_L^2 < x_U^2 < \ldots < x_L^s < x_U^s = D_{MAX} \text{ and}$$
$$x_U^i + 1 = x_L^{i+1}, \quad 1 \leq i \leq s-1.$$

$$0 = y_L^1 < y_U^1 < y_L^2 < y_U^2 < \ldots < y_L^r < y_U^r = E_{MAX} \text{ and}$$
$$y_U^i + 1 = y_L^{i+1} \quad 1 \leq i \leq r-1.$$

Cardinality Reduction

- Aggregate state: $s' = \{l^1, \ldots, l^{n+1}, l^{n+2}, \ldots, l^{2n+1}, l^{2n+2}\}$
  - l^i is the data buffer level in the i^{th} node, $1 \leq i \leq n$
  - l^{n+1} is the energy buffer level
  - $l^i \in \{1, \ldots, s\}$, $1 \leq i \leq n$, $n + 2 \leq i \leq 2n + 1$
  - $l^{n+1}, l^{2n+2} \in \{1, \ldots, r\}$
- Aggregate action for state s' is $t' = (t^1, \ldots, t^n)$ where $t^i \in \{1, \ldots, l^{n+1}\}$, $1 \leq i \leq n$
- $s \ll D_{MAX}$, $r \ll E_{MAX}$
- $|S' \times A'| \approx 4^{10}$ with four nodes, $E_{MAX} = D_{MAX} = 30$ and four partitions of data and energy buffers

Q-learning with State Aggregation

- $S^{'}$: Aggregate state space
- $A^{'}$: Aggregate action space
- Q-value $Q(s', t')$: defined for every aggregate state-action tuple
- Initially, $Q(s', t') = 0$ and updated using the following rule

$$Q_{j+1}(s', a') = (1 - \alpha(j))Q_j(s', a') + \alpha(j)\times \left(c(s', a') + \min_{b \in A'(v')} Q_j(v', b) - \min_{u \in A'(r')} Q_j(r', u) \right)$$

Policy Approximation

- The set of policies is approximated by a parameter vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_M)$
- Class of parameterized random policies is $\{\pi^{\boldsymbol{\theta}}, \boldsymbol{\theta} \in \mathbb{R}^M\}$
- $\pi^{\boldsymbol{\theta}}(s', a')$: Probability of picking aggregate action a' in aggregate state s'
- Goal: Tune $\boldsymbol{\theta}$ such that $\pi^{\boldsymbol{\theta}}$ is near optimal
- Search the policy space in an organised manner

Parameterized Boltzmann Policies

- $\phi_{sa} \in \mathbb{R}^M$: Feature vector for aggregate state-action tuple (s, a)

$$\pi^{\theta}(s,a) = \frac{e^{\boldsymbol{\theta}^\top \phi_{sa}}}{\sum_{b \in A(s)} e^{\boldsymbol{\theta}^\top \phi_{sb}}} \qquad \forall s \in S^{'},\ \forall a \in A^{'}(s)$$

- $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_M)$, $\theta_i \sim N(\mu_i, \sigma_i)$, where $1 \le i \le M$

Approach

- Generate N parameter vectors $(\boldsymbol{\theta}^1, \ldots, \boldsymbol{\theta}^N)$
- Simulate MDP trajectory using each parameter vector $\boldsymbol{\theta}^j$
- Compute average cost λ_j of each policy $\pi^{\boldsymbol{\theta}^j}$
- Choose trajectory j, if $\lambda_j < \lambda_c$
- Update (μ_i, σ_i) using the parameter vectors corresponding to these trajectories

Cross-Entropy Method (contd.)

- A quantile value $\rho \in (0, 1)$ is selected
- The average cost values are sorted in descending order. Let $\lambda_1, \ldots, \lambda_N$ be the sorted order. Hence $\lambda_1 \geq \ldots \geq \lambda_N$.
- $\lambda_c = \lambda_{\lceil (1-\rho) \rceil N}$ average cost is picked as the threshold level
- Meta-parameters $\{(\mu_i^t, \sigma_i^t),\ 1 \leq i \leq M\}$ are updated after iteration t as follows:

$$\mu_i^{(t+1)} = \frac{\sum_{j=1}^{N} I_{\{\lambda_j \leq \lambda_c\}} \theta_i^j}{\sum_{j=1}^{N} I_{\{\lambda_j \leq \lambda_c\}}}$$

$$\sigma_i^{2(t+1)} = \frac{\sum_{j=1}^{N} I_{\{\lambda_j \leq \lambda_c\}} \left(\theta_i^j - \mu_i^{(t+1)}\right)^2}{\sum_{j=1}^{N} I_{\{\lambda_j \leq \lambda_c\}}}.$$

Comparison Algorithms Used

1. Greedy method
   - Takes as input $(q_k^1, \ldots, q_k^n)$
   - Energy available in source is E_k
   - Number of energy units required to transmit q_k^i bits of data = $g^{-1}(q_k^i)$
   - Energy provided is $t_k = \min\left(E_k, \sum_{i=1}^{n} g^{-1}(q_k^i)\right)$
2. Combined nodes method
   - State: $w_k = \left(\sum_{i=1}^{n} q_k^i, E_k\right)$
   - Action: $v_k =$ total energy that needs to be distributed between the nodes
   - Uses Q-learning update rule to update $Q(w_k, v_k)$
   - Finds the total optimal energy to be supplied, but not the exact split

Results - Two nodes and EH source

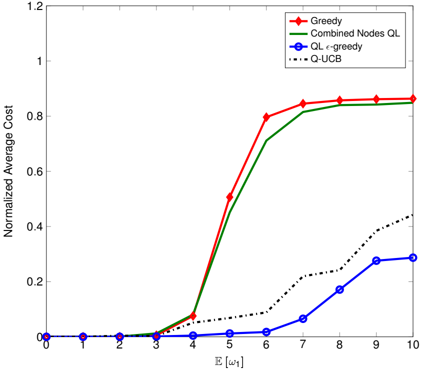


Figure: Performance comparison of policies

- $E_{MAX} = 20$, $D_{MAX} = 10$
- $X_k = AX_{k-1} + \boldsymbol{\omega}$, $\boldsymbol{\omega} = (\omega_1, \omega_2)^\top$
- $\mathbb{E}[\omega_2] = 1.0$
- $A = \begin{pmatrix} 0.2 & 0.3 \\ 0.3 & 0.2 \end{pmatrix}$
- $Y_k = bY_{k-1} + \chi$, $b = 0.5$, $\mathbb{E}[\chi] = 20$
- χ, ω are Poisson distributed
- Conversion function $g(T_k^i) = 2\ln(1 + T_k^i)$

Results - Four nodes and EH source

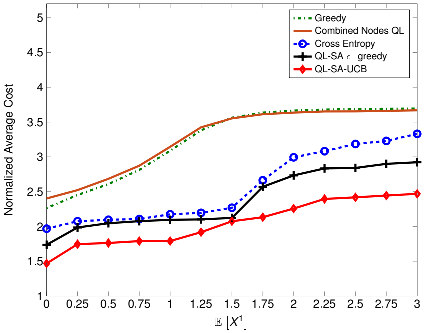


Figure: Performance comparison of policies

- $E_{MAX} = D_{MAX} = 30$
- X^1, X^2, X^3, X^4, Y are Poisson Distributed
- $\mathbb{E}\left[X^2\right] = \mathbb{E}\left[X^3\right] = \mathbb{E}\left[X^4\right] = 1.0$, $\mathbb{E}\left[Y\right] = 25$
- $\mathbb{E}\left[X^1\right]$ is varied
- Six partitions of energy and data buffers
- $\epsilon = 0.1$
- Conversion function $g(T_k^i) = \ln(1 + T_k^i)$

Summary

- We proposed algorithms to manage energy available through harvesting
- In order to deal with the curse of dimensionality, we also developed approximation algorithms

  Future Directions:

- Gradient-based approaches
- Features used in approximation methods can be tuned

Questions?

References I

- S. Bhatnagar, V. S. Borkar, and K. J. Prabuchandran, "Feature search in the grassmanian in online reinforcement learning," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, pp. 746–758, 2013.
- I. Menache, S. Mannor, and N. Shimkin, "Basis function adaptation in temporal difference reinforcement learning," *Annals of Operations Research*, vol. 134, no. 1, pp. 215–238, 2005.
- K. J. Prabuchandran, S. K. Meena, and S. Bhatnagar, "Q-learning based energy management policies for a single sensor node with finite buffer," *Wireless Communications Letters, IEEE*, vol. 2, no. 1, pp. 82–85, 2013.
- K. Raman and R. Chatterjee, "Optimal monopolist pricing under demand uncertainty in dynamic markets," *Management Science*, pp. 144–162, 1995.
- V. Sharma, U. Mukherji, V. Joseph, and S. Gupta, "Optimal energy management policies for energy harvesting sensor nodes," *IEEE Transactions on Wireless Communications*, vol. 9, no. 4, pp. 1326–1336, 2010.

References II

S. Chakravarty, S. Padakandla, and S. Bhatnagar, "A simulation-based algorithm for optimal pricing policy under demand uncertainty," *International Transactions in Operational Research*, 2013. [Online]. Available: http://dx.doi.org/10.1111/itor.12064

S. Padakandla, K. J. Prabuchandran, and S. Bhatnagar, "Energy sharing for multiple sensor nodes with finite buffers," *IEEE Transactions on Communications*, 2014 (Under Review).

Thank You