Resource Allocation for Sequential Decision Making under Uncertainty: Studies in Vehicular Traffic Control, Service Systems, Sensor Networks and Mechanism Design

Prashanth. L.A. Advisor: Prof. Shalabh Bhatnagar

Department of Computer Science and Automation
Indian Institute of Science Bangalore

March, 2013

Outline

The problem

- Question:"how to allocate resources amongst competing entities so as to maximize the rewards accumulated in the long run?"
- Resources: may be abstract (e.g. time) or concrete (e.g. manpower)
- The sequential decision making setting:
  - involves one or more agents interacting with an environment to procure rewards at every time instant, and
  - the goal is to find an optimal policy for choosing actions
- Uncertainties in the system
  - the stochastic noise and partial observability in a single-agent setting or private information of the agents in a multi-agent setting
- Real-world problems: high-dimensional state and action spaces and hence, the choice of knowledge representation is crucial

The studies conducted

Vehicular Traffic Control Here we optimize the 'green time' resource of the lanes in a road network so that traffic flow is maximized in the long term

Service Systems Here we optimize the 'workforce', while complying to queue stability as well as aggregate service level agreement (SLA) constraints

Wireless Sensor Networks Here we allocate the 'sleep time' (resource) of the individual sensors in an object tracking application such that the energy consumption from the sensors is reduced, while keeping the tracking error to a minimum

Mechanism Design In a setting of multiple self-interested agents with limited capacities, we attempt to find an incentive compatible transfer scheme following a socially efficient allocation

Outline

The problem

Traffic Signal Control [1]

The problem we are looking at

- Maximizing traffic flow: adaptive control of traffic lights at intersections
- Control decisions based on:
  - coarse estimates of the queue lengths at intersecting roads
  - time elapsed since last light switch over to red

how do we solve it?

- Apply reinforcement learning (RL)
  - Works with real data i.e., system model not assumed
  - Simple, efficient and convergent!
- Use Green Light District (GLD) simulator for performance comparisons

[1] work as a project associate with DIT-ASTec

Reinforcement Learning (RL)

- Combines
  - Dynamic programming - optimization and control
  - Supervised learning - training a parametrized function approximator
- Operation:
  - Environment: evolves probabilistically over states
  - Policy: determines which action to be taken in each state
  - Reinforcement: the reward received after performing an action in a given state
  - Goal: maximize the expected cumulative reward
- Using trial-and-error process the RL agent learns the policy that achieves the goal

Outline

Traffic Signal Control Problem

The MDP specifics

- State: vector of queue lengths and elapsed times $s_n = (q_1, \cdots, q_N, t_1, \cdots, t_N)$
- Actions: $a_n =$ {feasible sign configurations in state s_n}
- Cost:

$$\begin{aligned} k(s_n, a_n) = & \quad r_1 * (\sum_{i \in I_p} r_2 * q_i(n) + \sum_{i \notin I_p} s_2 * q_i(n)) \\ + & \quad s_1 * (\sum_{i \in I_p} r_2 * t_i(n) + \sum_{i \notin I_p} s_2 * t_i(n)), \end{aligned} \tag{1}$$

  - where $r_i, s_i \geq 0$ and $r_i + s_i = 1, i = 1, 2$.
  - more weightage to main road traffic

Qlearning based TLC algorithm

Q-learning

An off-policy temporal difference based control algorithm

$$Q(s_{n+1}, a_{n+1}) = Q(s_n, a_n) + \alpha(n)\left(k(s_n, a_n) + \gamma \min_a Q(s_{n+1}, a) - Q(s_n, a_n)\right). \quad (2)$$

Why function approximation?

- need look-up table to store Q-value for every (s, a) in (2)
- Computationally expensive (Why?)
  - two-junction corridor: 10 signalled lanes, 20 vehicles on each lane
  - $|S \times A(S)| \sim 10^{14}$
- Situation aggravated when we consider larger road networks

Q-learning with Function Approximation [1]

- Approximate

$$Q(s,a) \approx \theta^T \sigma_{s,a}, \quad \text{where}$$

  - $\sigma_{s,a}$: d-dimensional feature vector, with $d << |S \times A(S)|$
  - θ is a tunable d-dimensional parameter

- Feature-based analog of Q-learning:

$$\theta_{n+1} = \theta_n + \alpha(n)\sigma_{s_n,a_n}(k(s_n,a_n) + \gamma \min_{v \in A(s_{n+1})} \theta_n^T \sigma_{s_{n+1},v} - \theta_n^T \sigma_{s_n,a_n})$$

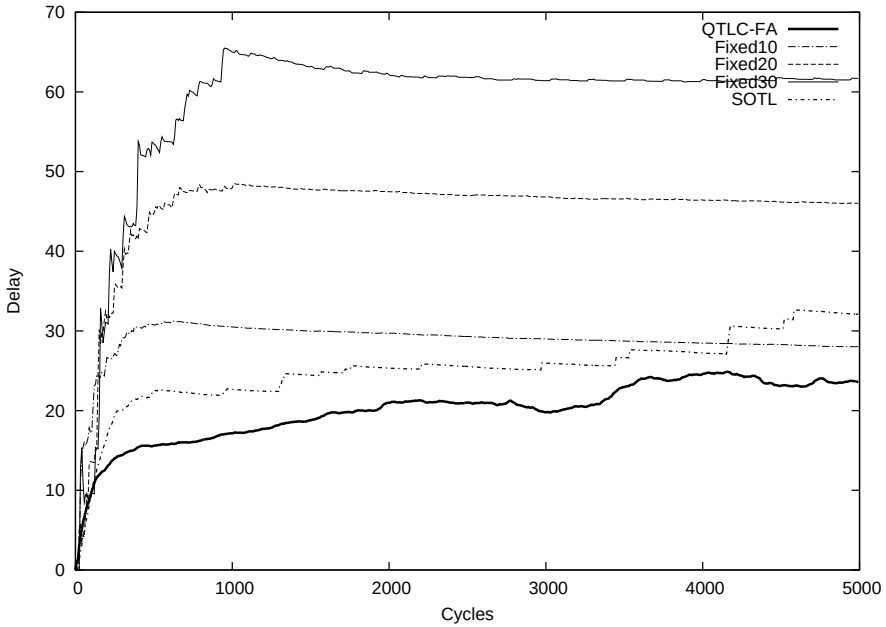
- σ_{s_n,a_n}: is graded and assigns a value for each lane based on its congestion level (low, medium or high)

Q-learning with Function Approximation [2]

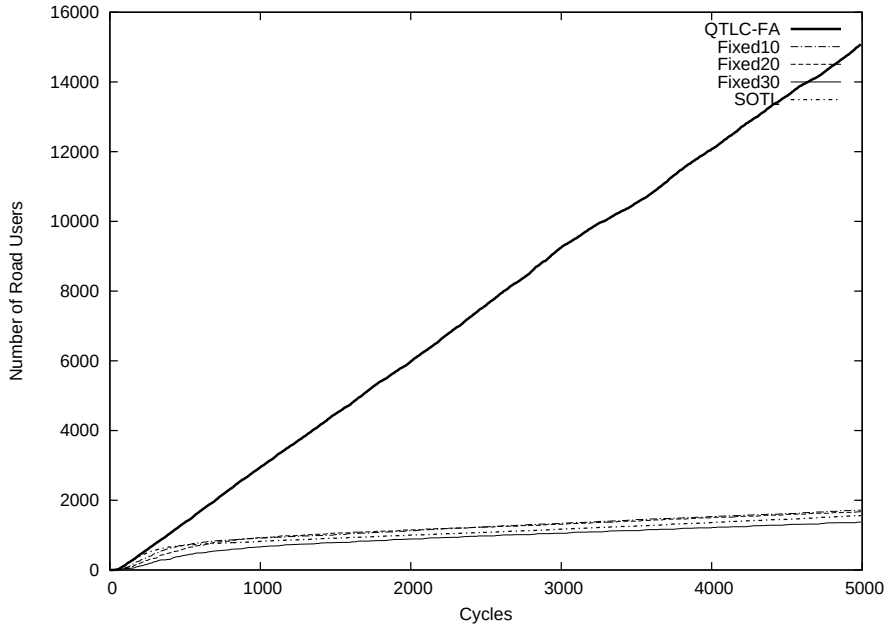
Feature Selection

State (s_n)	**Action** (a_n)	**Feature** (σ_{s_n,a_n})
$q_i(n) < L1$ and $t_i(n) < T1$	RED GREEN	0 1
$q_i(n) < L1$ and $t_i(n) \geq T1$	RED GREEN	0.2 0.8
$L1 \leq q_i(n) < L2$ and $t_i(n) < T1$	RED GREEN	0.4 0.6
$L1 \leq q_i(n) < L2$ and $t_i(n) \geq T1$	RED GREEN	0.6 0.4
$q_i(n) \geq L2$ and $t_i(n) < T1$	RED GREEN	0.8 0.2
$q_i(n) \geq L2$ and $t_i(n) \geq T1$	RED GREEN	1 0

Results on a 3x3-Grid Network



(a) Average junction waiting time



(b) Total Arrived Road Users

- Full state RL algorithms (cf. [B. Abdulhai et al. 2003][a]) are not feasible as $|S \times A(S)| \sim 10^{101}$, whereas $\dim(\sigma_{s_n,a_n}) \sim 200$
- **Self Organizing TLC** (SOTL) [b] switches a lane to green if elapsed time crosses a threshold, provided the # of vehicles crosses another threshold

[a] B. Abdulhai et al, "Reinforcement learning for true adaptive traffic signal control," *Journal of Transportation Engineering*, 2003.

[b] S. Cools et al, "Self-organizing traffic lights: A realistic simulation," *Advances in Applied Self-organizing Systems*,2008

Outline

Threshold tuning using stochastic optimization

- Thresholds are
  - $L1$ and $L2$ on the waiting queue lengths
- TLC algorithm uses broad congestion estimates instead of exact queue lengths
  - congestion is low, medium or high if the queue length falls below $L1$ or between $L1$ and $L2$ or above $L2$
- How to tune Li's? Use **stochastic optimization**
- Combine the tuning algorithm with
  - A full state Q-learning algorithm with state aggregation
  - A function approximation Q-learning TLC with a novel feature selection scheme
  - A priority based scheduling scheme

The Framework

- $\{X_n, n \geq 1\}$ Markov process parameterized with θ $(\in \mathcal{R}^3)$
  - θ takes values in a compact set
    $C \stackrel{\triangle}{=} [L1_{\min}, L1_{\max}] \times [L2_{\min}, L2_{\max}] \times [T1_{\min}, T1_{\max}]$
- $h : \mathcal{R}^d \to \mathcal{R}^+$ be a given bounded and continuous cost function.
- Goal: find a θ that minimizes:

$$J(\theta) = \lim_{l \to \infty} \frac{1}{l} \sum_{j=0}^{l-1} h(X_j). \tag{3}$$

- Thus, one needs to evaluate $\nabla J(\theta) \equiv (\nabla_1 J(\theta), \ldots, \nabla_N J(\theta))^T$.
- Gradient estimate:

$$\nabla J(\theta) \approx \frac{J(\theta + \delta \Delta_n)}{\delta} \Delta_n^{-1}, \tag{4}$$

  - $\delta > 0$ is a fixed small real number and $\Delta_n = (\Delta_n(1), \ldots, \Delta_n(N))^T$ is the perturbation vector constructed using Hadamard matrices

Threshold Tuning Algorithm

Consider $\{\hat{s}_l\}$ governed by $\{\hat{\theta}_l\}$, where $\hat{\theta}_l = \theta_n + \delta\triangle(n)$ for $n = \left[\frac{l}{L}\right]$, $L \geq 1$ fixed

Update rule

$$\begin{aligned} L1(n+1) &= \pi_1\left(L1(n) - a(n)\left(\frac{\tilde{Z}(nL)}{\delta\triangle_1(n)}\right)\right), \\ L2(n+1) &= \pi_2\left(L2(n) - a(n)\left(\frac{\tilde{Z}(nL)}{\delta\triangle_2(n)}\right)\right), \\ T1(n+1) &= \pi_3\left(T1(n) - a(n)\left(\frac{\tilde{Z}(nL)}{\delta\triangle_3(n)}\right)\right), \end{aligned} \tag{5}$$

where for $m = 0, 1, \ldots, L-1$,

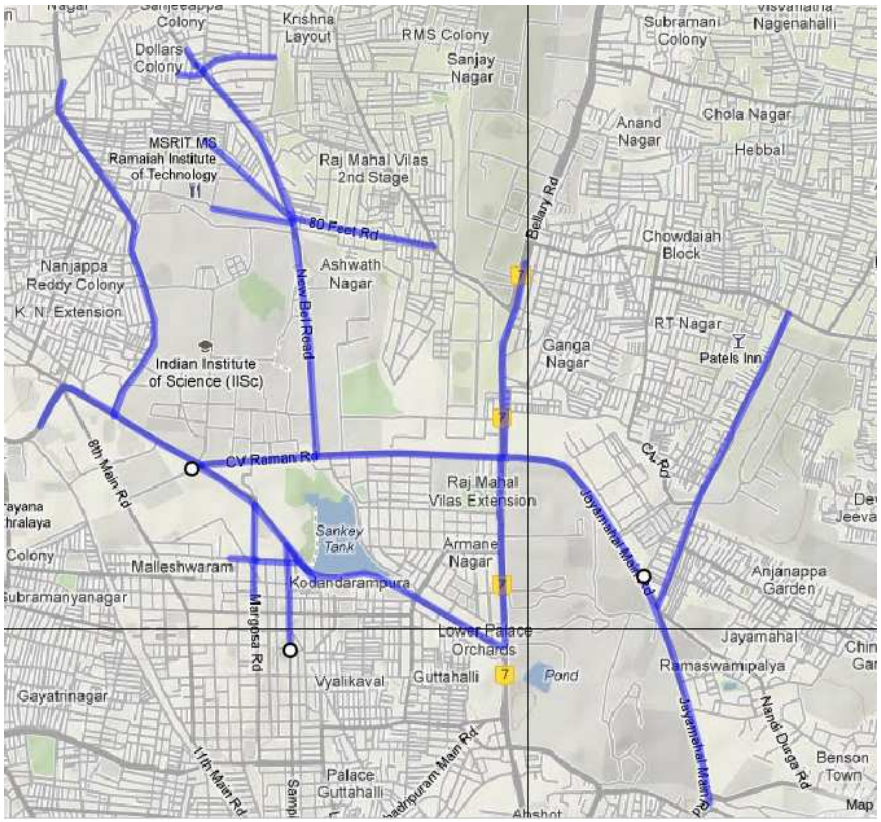
$$\tilde{Z}(nL+m+1) = \tilde{Z}(nL+m) + b(n)(k(\hat{s}_{nL+m}, \hat{a}_{nL+m}) - \tilde{Z}(nL+m)). \tag{6}$$

Priority based TLC (PTLC)

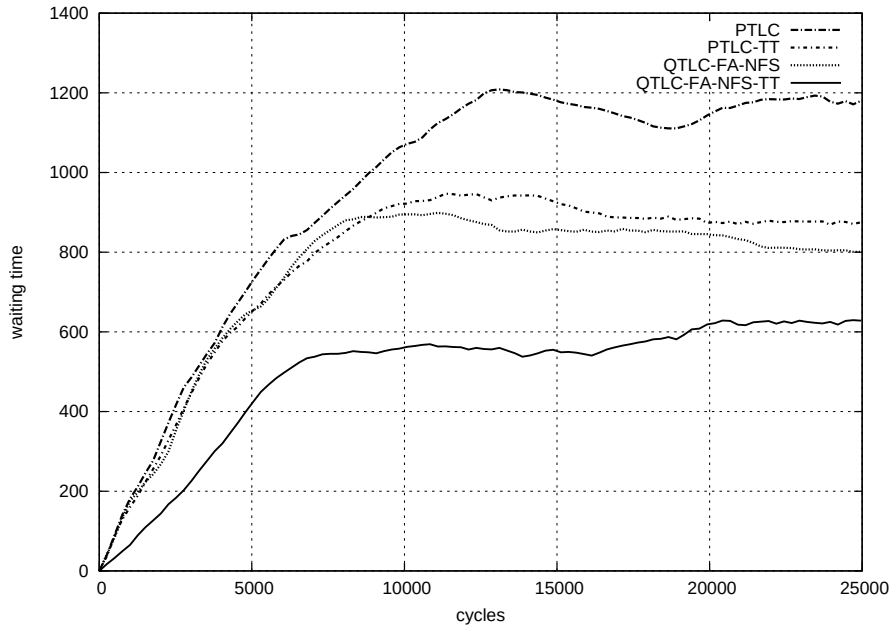
Condition	Priority value
$q_i < L1$ and $t_i < T1$	1
$q_i < L1$ and $t_i \geq T1$	2
$q_i \geq L1$ and $q_i < L2$ and $t_i < T1$	3
$q_i \geq L1$ and $q_i < L2$ and $t_i \geq T1$	4
$q_i \geq L2$ and $t_i < T1$	5
$q_i \geq L2$ and $t_i \geq T1$	6

PTLC selects the sign configuration with the maximum sum of lane priority values

Results on the IISc network



(c) IISc Network



(d) PTLC, QTLC-FA-NFS with and without threshold tuning

Outline

TD(0) WITH FUNCTION APPROXIMATION

- Approximate

$$V^{\mu} \approx \Phi\theta = \begin{bmatrix} \sum_{j=1}^{d} \phi_j(1)\theta_j \\ \sum_{j=1}^{d} \phi_j(2)\theta_j \\ \vdots \\ \sum_{j=1}^{d} \phi_j(|S|)\theta_j \end{bmatrix}.$$

  where
  - ϕ_i: d-dimensional feature vector corresponding to i, with $d << |S|$
  - θ is a tunable d-dimensional parameter
- The TD(0) update rule:

$$\begin{aligned} \theta_{n+1} &= \theta_n + a(n)\delta_n\phi(X_n), \text{where} \\ \delta_n &= (c(X_n, \mu(X_n)) + \gamma\phi(X_{n+1})^T\theta_n - \phi(X_n)^T\theta_n), \ n \geq 0 \end{aligned}$$

Feature adaptation in TD(0)

Let Φ^r denote the feature matrix during the rth step of the algorithm

Algorithm

Step 1 From TD(0) obtain θ^r_M (for some large M)

Step 2 Pick the worst and second worst indices from θ^r_M, say k and l, i.e.,

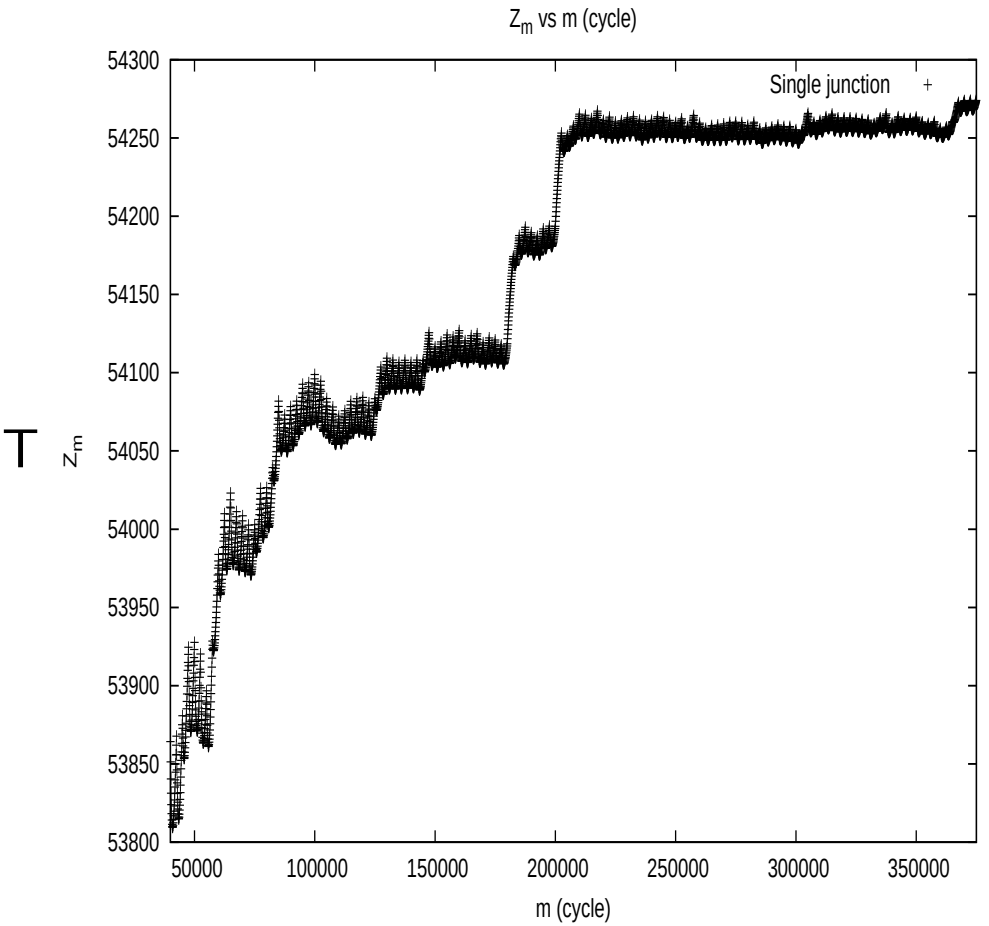
$$\theta^r_{M,k} \leq \theta^r_{M,l} \leq \theta^r_{M,j} \ \forall j \in \{1,\ldots,d, j \neq k, j \neq l\}.$$

Obtain a new feature matrix Φ^{r+1} as follows:

- Replace kth column of Φ as $\sum_{i=1}^{d} \phi^r_i \theta^r_i$ and
- replace lth column randomly (from a $U[0,1]$ distribution)

Step 3 Repeat Steps 1 and 2 until $r < R$. Output θ^R_M as the final parameter

Results – Single junction



# Cycle	Z_m	Z_m (ith episode) - Z_m (1st episode)
2499	51042.23	
74999	54003	2960.76
149999	54116.59	3074.36
224999	54260.28	3218.05
299999	54255.38	3213.15
374999	54274.72	3232.49

The difference of $||V_n||$ with the corresponding value at the end first episode, is seen to increase as features get adapted with episodes

Here $Z_m = (1-a) * Z_m + a * ||V_m||$; where

- $||V_n||$ is the Euclidean norm of $V_n = (V_n(i), i \in S)$ i.e., $||V_n|| = (\sum_{i \in S} V_n(i)^2)^{1/2}$ and
- $a = 0.001$

The road ahead

Outline

Motivation

Labor Cost Optimization [2]

The problem we are looking at

Find the optimal number of workers for each shift and of each skill level

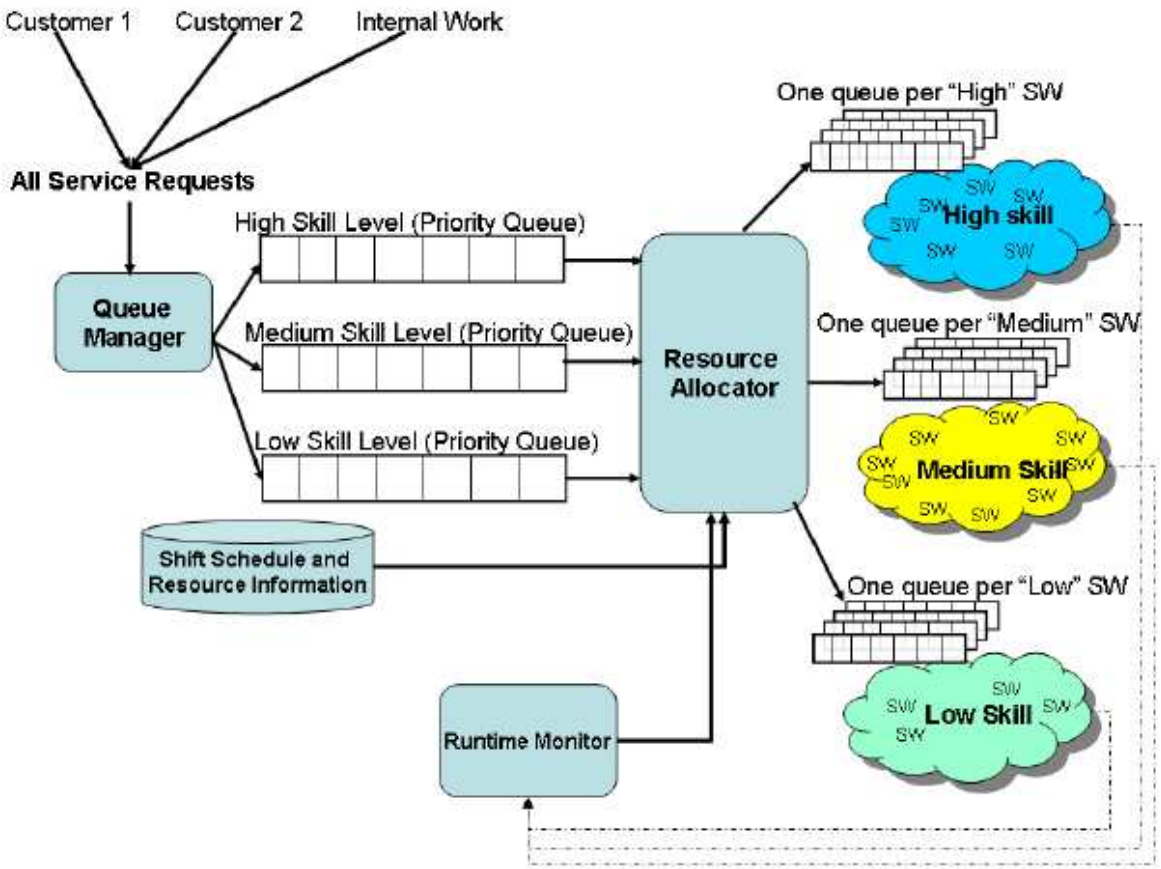
- that minimizes the long run average labor cost
- subject to service level agreement (SLA) constraints and queue stability

how do we solve it?

- Develop stochastic optimization methods that
  - work with simulation (noisy) estimates of a cost function
  - converge to the optimum of a long run performance objective,
  - satisfy SLA and queue stability constraints

[2] work as an intern at IBM Research, India

Operational model of the SS



Aim: Find the optimal number of workers for each shift and of each skill level

- that minimizes the long run average labour cost
- subject to SLA constraints and queue stability

TABLE: Workers $W_{i,j}$

	Skill levels		
Shift	High	Med	Low
S1	1	3	7
S2	0	5	2
S3	3	1	2

TABLE: Utilizations $u_{i,j}$

	Skill levels		
Shift	High	Med	Low
S1	67%	34%	26%
S2	45%	55%	39%
S3	23%	77%	62%

TABLE: SLA targets $\gamma_{i,j}$

	Customers	
Priority	Bossy Corp	Cool Inc
P_1	95%4h	89%5h
P_2	95%8h	98%12h
P_3	100%24h	95%48h
P_4	100%18h	95%144h

TABLE: SLA attainments $\gamma'_{i,j}$

	Customers	
Priority	Bossy Corp	Cool Inc
P_1	98%4h	95%5h
P_2	98%8h	99%12h
P_3	89%24h	90%48h
P_4	92%18h	95%144h

Outline

Constrained hidden Markov cost process with a discrete worker parameter

State:

$$X_n = (\underbrace{\mathcal{N}_1(n), \ldots, \mathcal{N}_{|B|}(n)}_{\text{complexity queue lengths}}, \underbrace{u_{1,1}(n), \ldots\ldots\ldots, u_{|A|,|B|}(n)}_{\text{worker utilizations}}, \underbrace{\gamma'_{1,1}(n), \ldots\ldots\ldots, \gamma'_{|C|,|P|}(n)}_{\text{SLAs attained}}, q(n)),$$

$$Y_n = \underbrace{(\mathcal{R}_{1,1,1}(n), \ldots, \mathcal{R}_{1,1,W_{\max}}(n), \ldots, \mathcal{R}_{|A|,|B|,W_{\max}}(n)))}_{\text{residual service times}}.$$

Single-stage cost:

$$c(X_n) = r \times \left(1 - \sum_{i=1}^{|A|}\sum_{j=1}^{|B|} \alpha_{i,j} \times u_{i,j}(n)\right) + s \times \left(\sum_{i=1}^{|C|}\sum_{j=1}^{|P|} \left|\gamma'_{i,j}(n) - \gamma_{i,j}\right|\right)$$

Idea: minimize *under-utilization* of workers and *over/under-achievement* of SLAs

Constraints:

$$g_{i,j}(X_n) = \gamma_{i,j} - \gamma'_{i,j}(n) \leq 0, \forall i,j \qquad \textbf{(SLA attainments)}$$

$$h(X_n) = 1 - q(n) \leq 0, \qquad \textbf{(Queue Stability)}$$

Constrained Optimization Problem

Parameter $\theta = (\underbrace{W_{1,1}, \ldots\ldots, W_{|A|,|B|}}_{number\ of\ workers})^T$

Average Cost $J(\theta) \stackrel{\triangle}{=} \lim_{n\to\infty} \frac{1}{n} \sum_{m=0}^{n-1} E[c(X_m)]$

subject to

SLA constraints $G_{i,j}(\theta) \stackrel{\triangle}{=} \lim_{n\to\infty} \frac{1}{n} \sum_{m=0}^{n-1} E[g_{i,j}(X_m)] \leq 0,$

Queue Stability $H(\theta) \stackrel{\triangle}{=} \lim_{n\to\infty} \frac{1}{n} \sum_{m=0}^{n-1} E[h(X_m)] \leq 0$

θ^* cannot be found by traditional methods - not a closed form formula!

Outline

Lagrange Theory and a Three-Stage Solution

$$\max_{\lambda} \min_{\theta} L(\theta, \lambda) \stackrel{\triangle}{=} J(\theta) + \sum_{i=1}^{|C|} \sum_{j=1}^{|P|} \lambda_{i,j} G_{i,j}(\theta) + \lambda_f H(\theta)$$

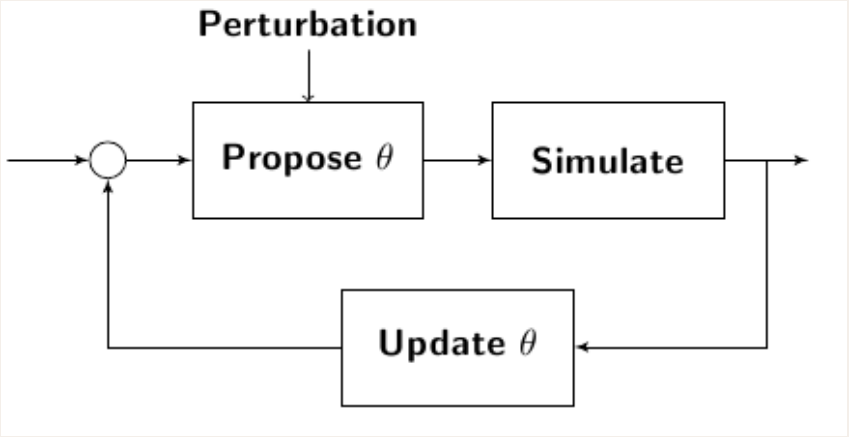
Three-Stage Solution:

INNER-MOST STAGE simulate the SS for several time steps

NEXT OUTER STAGE compute a gradient estimate using simulation results and then update θ along descent direction

OUTER-MOST STAGE update the Lagrange multipliers λ using the constraints in the ascent direction

SASOC Algorithms



Multi-timescale stochastic approximation SASOC runs all three loops simultaneously with varying step-sizes

SPSA for estimating $\nabla L(\theta, \lambda)$ using simulation results

Lagrange theory SASOC does gradient descent on the primal using SPSA and dual-ascent on the Lagrange multipliers

Generalized projection All SASOC algorithms involve a certain generalized smooth projection operator that helps imitate a continuous parameter system

SASOC-G Algorithm

Update rule

$$W_i(n+1) = \bar{\Gamma}_i \left[W_i(n) + b(n) \left(\frac{\bar{L}(nK) - \bar{L}'(nK)}{\delta \Delta_i(n)} \right) \right], \forall i = 1, 2, \ldots, N$$

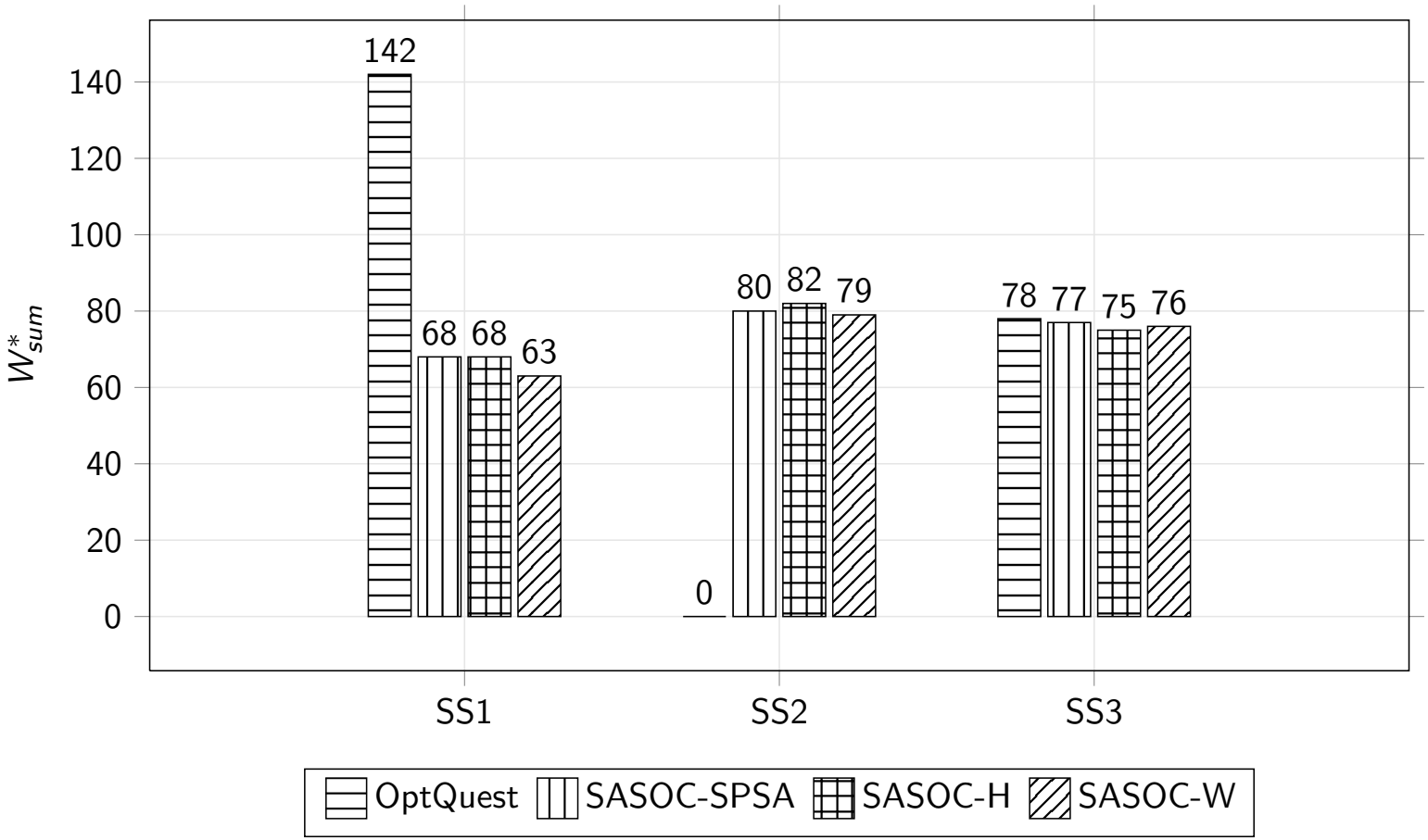
where for $m = 0, 1, \ldots, K-1$,

$$\begin{aligned}
\bar{L}(nK+m+1) &= \bar{L}(nK+m) + d(n)(l(X_{nK+m}, \lambda(nK)) - \bar{L}(nK+m)), \\
\bar{L}'(nK+m+1) &= \bar{L}'(nK+m) + d(n)(l(\hat{X}_{nK+m}, \lambda(nK)) - \bar{L}'(nK+m)), \\
\lambda_{i,j}(n+1) &= (\lambda_{i,j}(n) + a(n) g_{i,j}(X_n))^+, \forall i = 1, 2, \ldots, |C|, j = 1, 2, \ldots, |P|, \\
\lambda_f(n+1) &= (\lambda_f(n) + a(n) h(X_n))^+ .
\end{aligned}$$

In the above, $l(X, \lambda) = c(X) + \sum_{i=1}^{|C|} \sum_{j=1}^{|P|} \lambda_{i,j} g_{i,j}(X) + \lambda_f h(X)$.

- SASOC-H and SASOC-W are second-order Newton methods
- SASOC-H involves an explicit inversion of the Hessian at each update step, whereas SASOC-W leverages the Woodbury's identity to directly tune the inverse of the Hessian

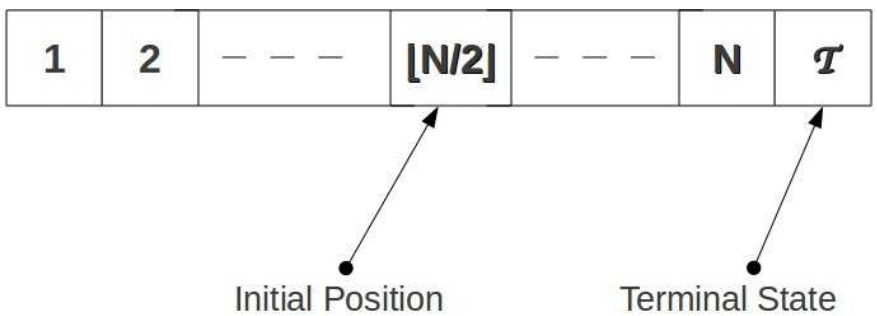
Results for EDF dispatching policy



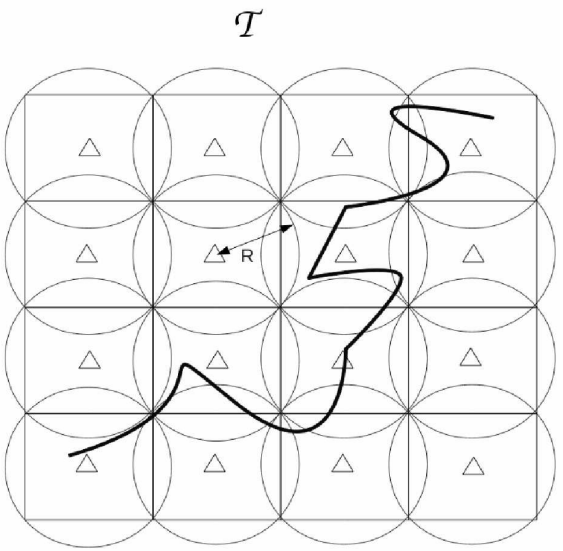
- **SASOC** is compared against **OptQuest** – a state-of-the-art optimization package – on five real-life SS via AnyLogic Simulation Toolkit
- **SASOC** is an order of magnitude faster than **OptQuest** and finds better solutions in many cases, both from number of workers as well as their utilization viewpoints

Outline

The Setting [1]



(e) 1-d network setup



(f) 2-d network setup

The Setting [2]

- Sensors can be either awake or sleep
- sleep time $\in \{0, \ldots, \Lambda\}$
- Object movement evolves as a Markov chain, with transition probability matrix $\mathbf{P} = [P_{ij}]_{(N+1)\times(N+1)}$
- $\mathcal{T}$: exterior of the network

The Setting [2]

- Sensors can be either awake or sleep
- sleep time $\in \{0, \ldots, \Lambda\}$
- Object movement evolves as a Markov chain, with transition probability matrix $\mathbf{P} = [P_{ij}]_{(N+1)\times(N+1)}$
- $\mathcal{T}$: exterior of the network

What are we trying to optimize ?

- Make sensors sleep to save energy
- Keep minimum sensors awake to have good tracking accuracy
- Find "good trade-off" between the above two conflicting objectives

Sleep–wake control POMDP [1]

State, Action and Observation

- State: $\mathbf{s}_k = (l_k, \mathbf{r}_k)$
  - l_k refers to the location of the object at instant k and can take values $1, \ldots, n, T$
  - $\mathbf{r}_k = (r_k(1), ..., r_k(N))$ where $r_k(i)$ denotes the remaining sleep time of the i^{th} sensor
- the remaining sleep time vector $\mathbf{r}_k$ evolves as follows

$$r_{k+1}(i) = (r_k(i) - 1)\mathcal{I}_{\{r_k(i)>0\}} + a_k(i)\mathcal{I}_{\{r_k(i)=0\}}, \tag{7}$$

- The action $\mathbf{a}_k$ at instant k is the vector of chosen sleep times of the sensors

Sleep–wake control POMDP [2]

Why POMDP?

- It is not possible to track the object (l_k) at each time instant as the sensors at the object's location may be in sleep state
- Let $\mathbf{p}_k = (p_k(1), ..., p_k(N), p_k(\mathcal{T}))$ be the distribution of the object's location being one of $1, 2, ..., N, \mathcal{T}$
  - p_k is a sufficient statistic in this POMDP setting
  - p_k evolves according to

$$\mathbf{p}_{k+1} = \mathbf{p}_k \mathbf{P} \mathcal{I}_{\{r_{k+1}(l_{k+1})>0\}} + \mathbf{e}_{l_{k+1}} \mathcal{I}_{\{r_{k+1}(l_{k+1})=0\}} + \mathbf{e}_{\mathcal{T}} \mathcal{I}_{\{l_{k+1}=\mathcal{T}\}}. \tag{8}$$

- Single-stage cost:

$$g(\mathbf{s}_k, \mathbf{a}_k) = \mathcal{I}_{\{l_k \neq \mathcal{T}\}} \left(\sum_{\{i: r_k(i)=0\}} c + \mathcal{I}_{\{r_k(l_k)>0\}} \mathcal{K} \right) \tag{9}$$

Outline

RL algorithms – discounted setting

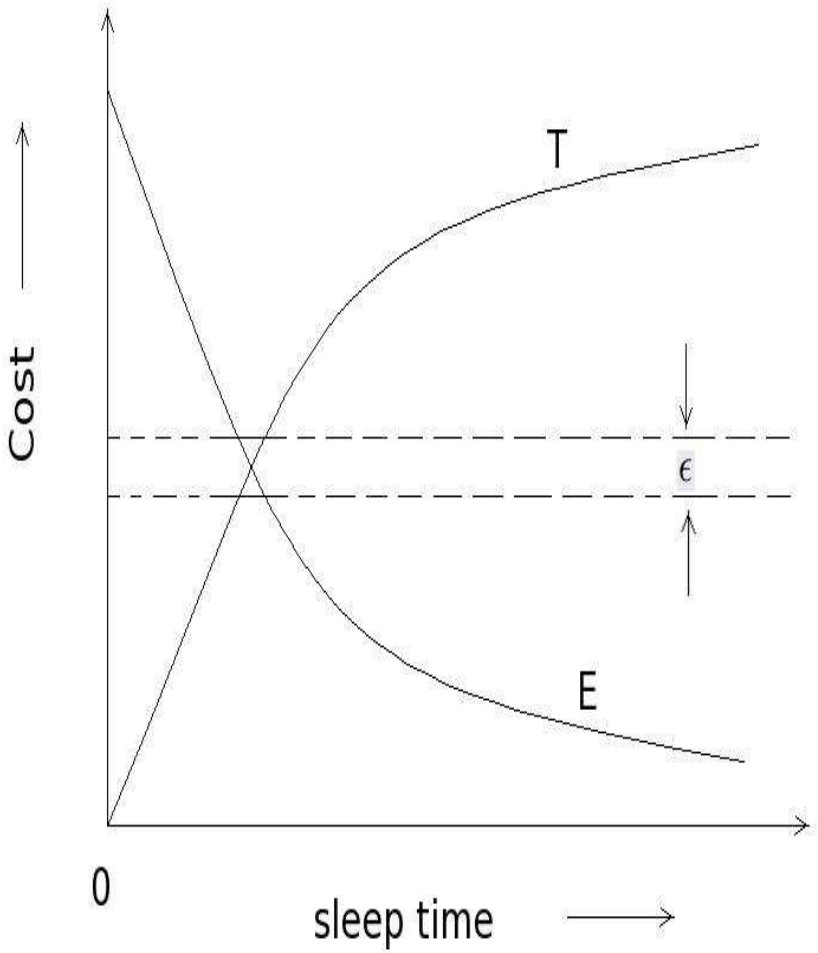
Q-Learning with function approximation - $\mathcal{QSA}$

$$\theta_{k+1} = \theta_k + \alpha(k)\sigma_{\mathbf{s}_k,\mathbf{a}_k}\left(r(\mathbf{s}_k,\mathbf{a}_k) + \gamma \max_{\mathbf{b}\in A(\mathbf{s}_{k+1})} \theta_k^T \sigma_{\mathbf{s}_{k+1},\mathbf{b}} - \theta_k^T \sigma_{\mathbf{s}_k,\mathbf{a}_k}\right)$$

Why function approximation?

- Q-learning with full state representations: need look-up table to store Q-value for every $(\mathbf{s},\mathbf{a})$
- Computationally expensive:
  121 sensors and $\Lambda = 3$, $|S \times A(S)| \sim 100^{122} \times 4^{121} \times 4^{121}$
- Solution: Function approximation with feature-based representations

Feature Selection Scheme



$$\sigma_{s_k,a_k} = (\sigma_{s_k,a_k}(1), \ldots, \sigma_{s_k,a_k}(N))^T,$$

where $\sigma_i(k), i \leq N$ is the feature value corresponding to sensor i

$$\text{Let } \rho_k = c(\Lambda - a_k(i)) - \sum_{j=1}^{a_k(i)} [\mathbf{pP}^j]_i$$

Then,

$$\sigma_{\mathbf{s_k},\mathbf{a_k}}(i) = \begin{cases} V \times sgn(\theta_k(i)) & \text{if } 0 \leq |\rho_k| \leq \epsilon, \\ -V \times sgn(\theta_k(i)) & \text{otherwise} \end{cases}$$

RL algorithms – discounted setting

Two-timescale Online Convergent Q-learning

- Q-learning with function approximation - not proven to converge [a]
- $\mathcal{TQSA}$ adapted from [S. Bhatnagar et al. 2012] [b], updates according to

$$\theta_{n+1} = \Gamma_1\left(\theta_n + b(n)\sigma_{\mathbf{s}_n,\mathbf{a}_n}\left(r(\mathbf{s}_n,\mathbf{a}_n) + \gamma\theta_n^T\sigma_{\mathbf{s}_{n+1},\mathbf{a}_{n+1}} - \theta_n^T\sigma_{\mathbf{s}_n,\mathbf{a}_n}\right)\right),$$

$$\mathbf{w}_{n+1} = \Gamma_2\left(\mathbf{w}_n + a(n)\frac{\theta_n^T\sigma_{\mathbf{s}_n,\mathbf{a}_n}}{\delta}\Delta_n^{-1}\right)$$

- π is a Boltzmann-like policy parameterized by θ
- Γ_1, Γ_2 are projection operators that keep the iterates θ, w bounded
- Step-sizes $a(n), b(n)$ are such that θ is updated on slower timescale and w on the faster one

[a] L. Baird. Residual Algorithms: Reinforcement Learning with Function Approximation, ICML, 1995.

[b] S. Bhatnagar and K. Lakshmanan. An online convergent Q-learning algorithm with linear function approximation. JMLR (Under Review), 2012

Outline

RL algorithms – average setting

Q-learning with full state representation

$$Q_{n+1}(\mathbf{i},\mathbf{a}) = Q_n(\mathbf{i},\mathbf{a}) + \alpha(n)(r(\mathbf{s_n},\mathbf{a_n}) + \max_{\mathbf{r}\in A(\mathbf{j})} Q_n(\mathbf{j},\mathbf{r}) - \max_{\mathbf{b}\in A(\mathbf{s})} Q_n(\mathbf{s},\mathbf{b})), \quad \mathbf{i}\in S, \mathbf{a}\in A($$

$QSA-A$ Update Rule

$$\theta_{n+1} = \theta_n + \alpha(n)\sigma_{\mathbf{s}_n,\mathbf{a}_n}\left(r(\mathbf{s_n},\mathbf{a_n}) + \max_{\mathbf{v}\in A(\mathbf{s}_{n+1})} \theta_n^T \sigma_{\mathbf{s}_{n+1},\mathbf{v}} - \max_{\mathbf{r}\in A(\mathbf{s})} \theta_n^T \sigma_{\mathbf{s},\mathbf{r}}\right)$$

This is similar to the QTLC-FA-AC TLC algorithm outlined before[a]

[a] L.A. Prashanth and S. Bhatnagar. Reinforcement learning with average cost for adaptive control of traffic lights at intersections. In Proceedings of IEEE ITSC, 2011.

RL algorithms – average setting

$\mathcal{TQSA}$-$\mathcal{A}$

- Extension of $\mathcal{TQSA}$ to the average cost setting is not straightforward (Why?)
- $\mathcal{TQSA}$-$\mathcal{A}$ is a two-timescale stochastic approximation algorithm using deterministic perturbation sequences based on certain Hadamard matrices [S. Bhatnagar et al. 2003][a]
- Unlike $\mathcal{QSA}$-$\mathcal{A}$, $\mathcal{TQSA}$-$\mathcal{A}$ has theoretical convergence guarantees

[a] S. Bhatnagar, M.C. Fu, S.I. Marcus and I. Wang. Two-timescale simultaneous perturbation stochastic approximation using deterministic perturbation sequences.ACM Transactions on Modeling and Computer Simulation (TOMACS), 13(2):180 - 209, 2003.

RL algorithms – average setting

$\mathcal{TQSA}$-$\mathcal{A}$ update rule

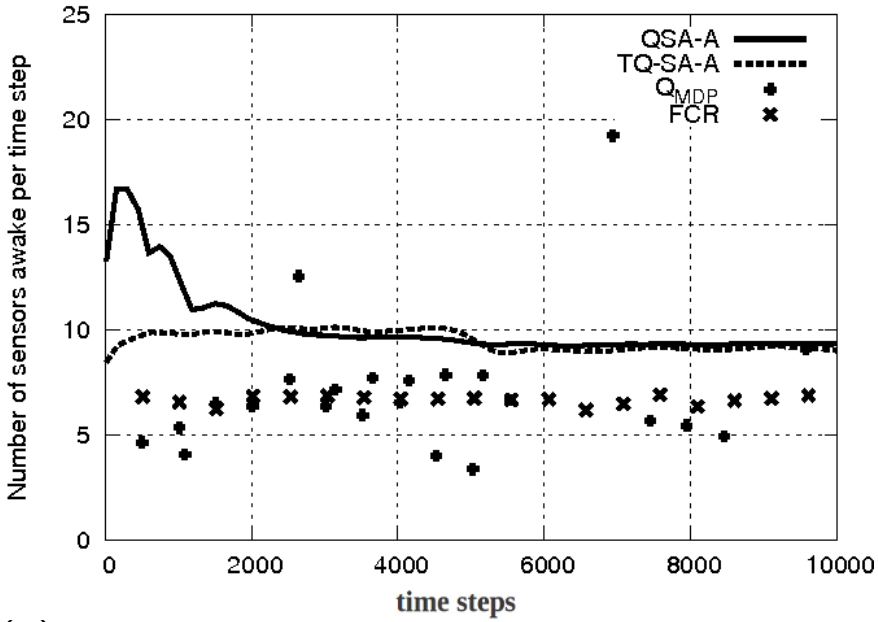
$$\theta_{n+1} = \Gamma_1\left(\theta_n + b(n)\sigma_{\mathbf{s}_n,\mathbf{a}_n}\left(r(\mathbf{s}_n,\mathbf{a}_n) - \hat{J}_{n+1} + \theta_n^T\sigma_{\mathbf{s}_{n+1},\mathbf{a}_{n+1}} - \theta_n^T\sigma_{\mathbf{s}_n,\mathbf{a}_n}\right)\right), \quad (13)$$

$$\hat{J}_{n+1} = \hat{J}_n + c(n)\left(r(\mathbf{s}_n,\mathbf{a}_n) - \hat{J}_n\right), \quad (14)$$

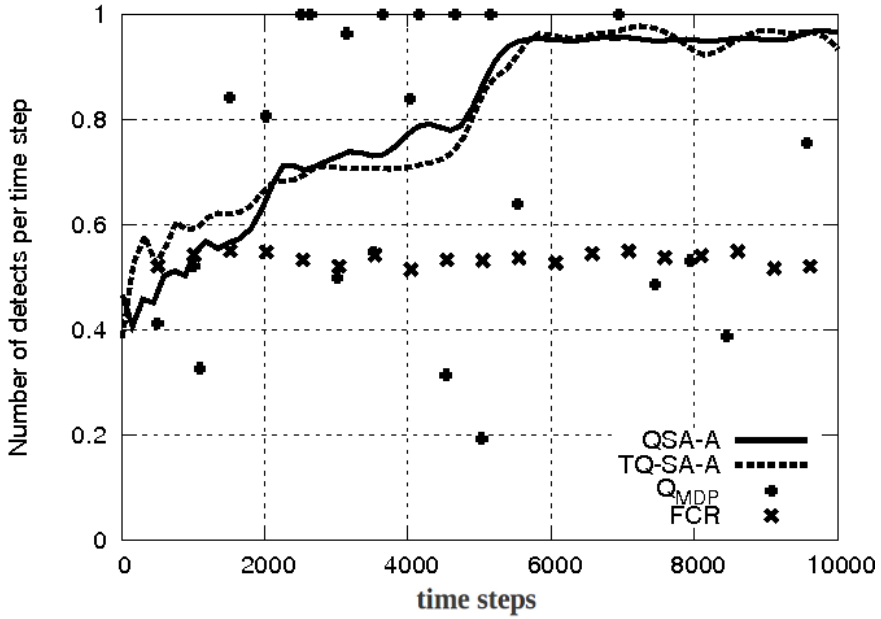
$$\mathbf{w}_{n+1} = \Gamma_2\left(\mathbf{w}_n + a(n)\frac{\theta_n^T\sigma_{\mathbf{s}_n,\mathbf{a}_n}}{\delta}\Delta_n^{-1}\right) \quad (15)$$

- On the slower timescale, the Q-value parameter is updated in a on-policy Q-learning manner
- on the faster timescale, the policy parameter is updated along a gradient descent direction using an SPSA-like estimate
- the average cost is estimated using (15) and this is used in (13)

Results on a 1-d network – average setting



(g) Number of sensors awake per time step



(h) Number of detects per time step

- While the number of sensors awake for *FCR* algorithm is lesser than that for $\mathcal{QSA}$-$\mathcal{A}$ and $\mathcal{TQSA}$-$\mathcal{A}$ algorithms, the tracking accuracy however is significantly lower in comparison
- While Q_{MDP} [3] keeps a lower number of sensors awake, it also results in lower tracking accuracy

[3] J.A. Fuemmeler and V.V. Veeravalli. Smart sleeping policies for energy efficient tracking in sensor networks. IEEE Transactions on Signal Processing, 56(5): 2091 – 2101, 2008.

Outline

The setting

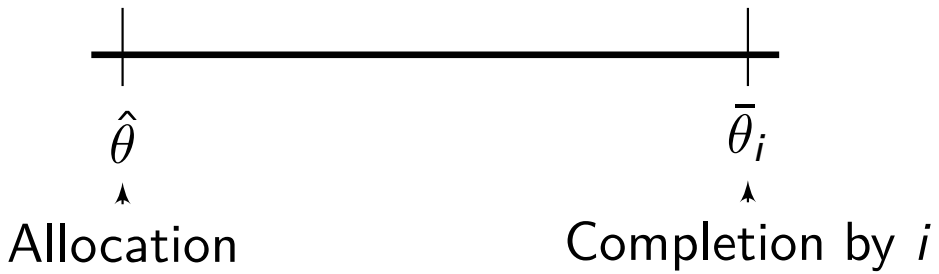
- Procurement scenario with agents $1, 2, \ldots, N$
- Agent i's type $\theta_i = (u_i, c_i)$, where u_i is the unit price and c_i the capacity
- Socially efficient allocation:

$$\begin{aligned} &\text{Find } \pi(\theta) = \underset{y \in \mathcal{Y}}{\operatorname{argmin}} \sum_{j=1}^{N} u_j y_j \\ &\text{s.t. } 0 \leq y_j \leq c_j, \quad j = 1, 2, \ldots, N, \\ &\text{and } \sum_{j=1}^{N} y_j = D. \end{aligned} \tag{16}$$

- Agent i's utility:

$$U_i = t_i - u_i \bar{c}_i + \pi_i((u_i, \hat{c}_i), \theta_{-i}) \tag{17}$$

The mechanism $\mathcal{MC}$



Notation	Description	Input type
$\pi(\hat{\theta})$	Efficient allocation with reported types	$\hat{\theta} = (\hat{\theta}_1, \hat{\theta}_2, \ldots, \hat{\theta}_N)$, where $\hat{\theta}_i = (\hat{u}_i, \hat{c}_i)$
$\pi(\bar{\theta}_i, \hat{\theta}_{-i})$	Efficient allocation with achieved type of agent i and reported types of other agents	$(\bar{\theta}_i, \hat{\theta}_{-i})$, where $\bar{\theta}_i = (\hat{u}_i, \bar{c}_i)$

Motivation [1]

Example 1

- Consider three agents with types $(u_1, c_1) = (1, 100)$, $(u_2, c_2) = (2, 50)$ and $(u_3, c_3) = (3, 130)$
- Agent 1 misreports his capacity to be 125, while the rest of the type is reported truly
- $\pi(\hat{\theta}) = (125, 25, 0)$ and achieved capacities are $(100, 25, 0)$
- A VCG-like payment:

$$t_i = \sum_{j \neq i} \widehat{u}_j \pi_{-i,j}(\hat{\theta}_{-i}) - \sum_{j \neq i} \widehat{u}_j \pi_j(\hat{\theta}). \tag{18}$$

  - Agent 1's payoff is $t_1 = (2 \times 50 + 3 \times 100) - (2 \times 25) = 350$
  - With true report, the same is $t_1 = (2 \times 50 + 3 \times 100) - (2 \times 50) = 300$
  - Agents have an incentive to misreport!

Motivation [2]

[Dash et al. 2007]

- a fixed δ-penalty based delayed transfer scheme:

$$t_i = \sum_{j \neq i} \widehat{u}_j \pi_{-i,j}(\hat{\theta}_{-i}) - \sum_{j \neq i} \widehat{u}_j \pi_j(\bar{\theta}_i, \hat{\theta}_{-i}) - \delta\beta_i. \tag{19}$$

  - β_i is a binary variable which is equal to 1 if $\bar{c}_i < \pi_i(\hat{\theta})$
- Agent 1's payoff (in Example 1) would be $t_1 = (2 \times 50 + 3 \times 100) - (2 \times 50) - \delta = 300 - \delta$ and under true capacity report, $t_1 = 300$ (as before)
- The corresponding utilities are [a] $325 - \delta$ and 300 respectively
- Thus, truthful capacity reports does not guarantee a higher utility for all values of δ !!

[a] the utility U_i of agent i in our setting is $U_i(\pi, t_i, \theta) = t_i - u_i \bar{c}_i + \pi_i((u_i, \hat{c}_i), \theta_{-i})$,

Static Mechanism $\mathcal{MC}$ [1]

Transfer Scheme

$$
\begin{aligned}
t_i &= x_i + p_i, \\
&\text{where} \\
x_i &= \sum_{j \neq i} \widehat{u}_j \pi_{-i,j}(\hat{\theta}_{-i}) - \sum_{j \neq i} \widehat{u}_j \pi_j(\bar{\theta}_i, \hat{\theta}_{-i}) \qquad (20) \\
p_i &= \sum_{j \neq i} \pi_j(\hat{\theta}) - \sum_{j \neq i} \pi_j(\bar{\theta}_i, \hat{\theta}_{-i}).
\end{aligned}
$$

- x_i is the marginal contribution of agent i (in the spirit of VCG)
- p_i is the loss in allocation to other agents due to agent i's misreport

Static Mechanism $\mathcal{MC}$ [2]

Payoffs in Example 1 by $\mathcal{MC}$

- $\pi_{-1}(\hat{\theta}_{-1}) = (50, 100)$ and $\pi(\bar{\theta}_1, \hat{\theta}_{-1}) = (100, 50, 0)$
- Marginal contribution $x_1 = (2 \times 50) - (2 \times 50) = 0$ to agent 1 and
- Penalty $p_1 = 25 - 50 = -25$
- Agent 1's utility under capacity misreport is $U_1 = (300 - 25) - 1 \times 100 + 125 = 250$. This is strictly lesser than the utility of 300 derived under true report

Theorem

The mechanism $\mathcal{MC}$ is strategyproof, i.e.,reporting true type is always a utility-maximizing strategy, regardless of what other agents do

Outline

Dynamic Mechanism $\mathcal{DMC}$

- Here we consider a dynamic setting where agent types evolve
- In each period, agents report types and the center takes a (socially-efficient) action
  - The agents here again have a preference to harm others via capacity misreports
- By a counterexample we show that the dynamic pivot mechanism [a] cannot be directly applied in our setting
- $\mathcal{DMC}$ enhances the dynamic pivot mechanism to add a delayed (variable) penalty scheme, which ensures truthtelling w.r.t. capacity type element

[a] D. Bergemann and J. Valimaki, "The dynamic pivot mechanism," Econometrica, vol. 78, no. 2, pp. 771âĂŞ789, 2010.

Motivation [1]

Example 2

- Demand $D^n = 150, n \geq 0$
- Three agents with types $(u_1^n, c_1^n) = (1, 100)$, $(u_2^n, c_2^n) = (2, 50)$ and $(u_3^n, c_3^n) = (3, 100), \forall n$,
- Fix n and suppose that $(\hat{u}_1^n, \hat{c}_1^n) = (1, 125)$, $(\hat{u}_2^n, \hat{c}_2^n) = (2, 50)$ and $(\hat{u}_3^n, \hat{c}_3^n) = (3, 100)$
- Also, assume that the agents report truthfully for all time instants $m > n$

Motivation [2]

Example 2 (contd)

- Let $V_i(\theta, y) = \mathsf{E}\left[\sum_{k=0}^{\infty} \gamma^k u_i^k y_i | \theta^0 = \theta, y\right]$. Then,

$$V_i(\theta^m, \pi) = \sum_{k=m}^{\infty} \gamma^{k-m} u_i^k \pi_i(\theta^k) = u_i \pi_i \sum_{k=m}^{\infty} \gamma^{k-m} = \frac{u_i \pi_i}{1-\gamma}$$

- We observe that for instant n, $\pi(\hat{\theta}^n) = (125, 25, 0)$ and $\pi_{-1}(\hat{\theta}^n_{-1}) = (50, 100)$. Hence, $V_{-1}(\hat{\theta}, \pi_{-1}) = \frac{(2 \times 50 + 3 \times 100)}{\frac{1}{4}} = 1600$

Motivation [3]

[Bergemann and Valimaki. 2010]

$$\tilde{x}_i^n(\hat{\theta}) = V_{-i}(\hat{\theta}, \pi_{-i}) - \left(v_{-i}(\hat{\theta}_{-i}, \pi(\hat{\theta})) + \gamma \, \mathbf{E}_{\theta'} \left[V_{-i}(\theta', \pi_{-i}) | \hat{\theta}, \pi(\hat{\theta}) \right] \right).$$

- the first term $V_{-i}(\hat{\theta}, \pi_{-i})$ is the total cost without agent i
- the second term is the total cost incurred by other agents with agent i

Payoffs in Example 2

- With overstated capacity, agent 1's payoff $\tilde{x}_1^n(\hat{\theta}) = 1600 - (2 \times 25 + \frac{3}{4} \times 1600) = 350$, and
- with true report, the same is $x_1^n(\theta) = 1600 - (2 \times 50 + \frac{3}{4} \times 1600) = 300$
- As in the static setting, an agent has an incentive to misreport with a dynamic-VCG like payment structure

Dynamic mechanism $\mathcal{DMC}$ [1]

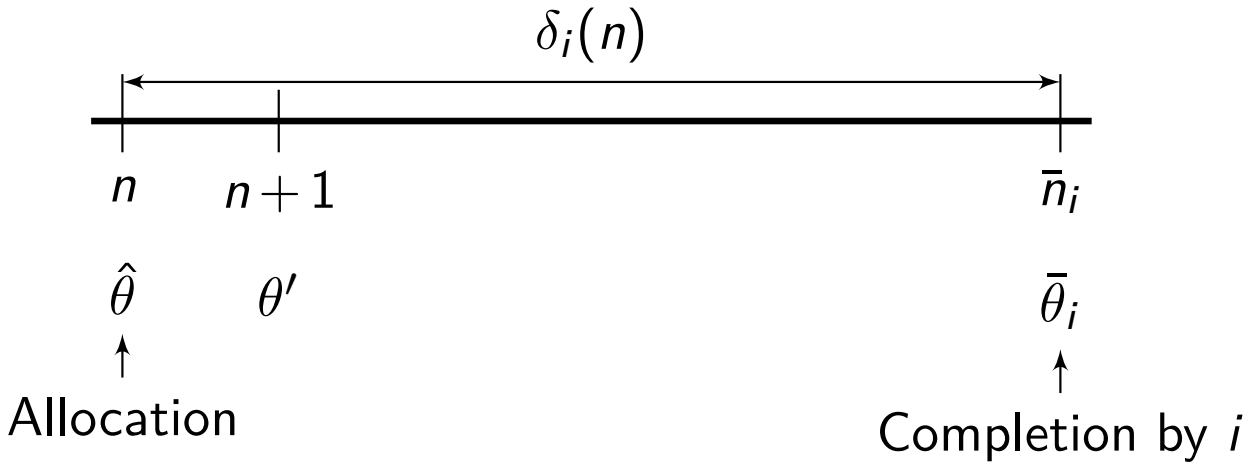


Figure: A portion of the time-line illustrating the process

Dynamic mechanism $\mathcal{DMC}$ [2]

Transfer Scheme

$$t_i(\bar{\theta}_i, \hat{\theta}) = \frac{1}{\gamma^{\delta_i(n)}} \left[x_i(\bar{\theta}_i, \hat{\theta}) + p_i(\bar{\theta}_i, \hat{\theta}) \right], \text{ where}$$

$$x_i(\bar{\theta}_i, \hat{\theta}) = V_{-i}(\hat{\theta}, \pi_{-i}) - \left(v_{-i}(\theta_{-i}, \pi(\hat{\theta})) + \gamma \, \mathbf{E}_{\theta'} \left[V_{-i}(\theta', \pi_{-i}) | (\bar{\theta}_i, \hat{\theta}_{-i}), \pi(\bar{\theta}_i, \hat{\theta}_{-i}) \right] \right),$$

$$p_i(\bar{\theta}_i, \hat{\theta}) = \pi_i(\bar{\theta}_i, \hat{\theta}_{-i}) - \pi_i(\hat{\theta})$$

- $x_i(\bar{\theta}_i, \hat{\theta})$, the marginal gain brought into the process by agent i's participation at instant n
- $p_i(\bar{\theta}_i, \hat{\theta})$, the penalty imposed on agent i to cover the damage caused to the process by misreport of capacity by him

Dynamic mechanism $\mathcal{DMC}$ [3]

Payoffs in Example 2

- Here $\bar{c}_1^n = 100$ and hence, $\bar{\pi}(\bar{\theta}_i^n, \hat{\theta}_{-i}^n) = (100, 50, 0)$
- Payoff to agent 1 under $\mathcal{DMC}$ is

$$\begin{aligned} x_1^n(\bar{\theta}_1^n, \hat{\theta}^n) &= 1600 - (100 + \frac{3}{4} \times 1600) = 300, \\ p_1^n(\bar{\theta}_1^n, \hat{\theta}^n) &= 25 - 50 = -25 < 0 \end{aligned}$$

- The utility derived by agent 1 with an overstated capacity of 125 is $300 - 25 - 1 \times 100 + 125 = 250$. This is strictly lesser than the the utility with true capacity report, i.e., 300

Theorem

$\mathcal{DMC}$ is ex-post incentive compatible, i.e., reporting true type is utility maximizing, whatever the types of other agents, assuming they're truthful

Publications I


Prashanth L. A. and S. Bhatnagar,
Threshold Tuning using Stochastic Optimization for Graded Signal Control,
IEEE Transactions on Vehicular Technology, 2012 (Accepted).


Prashanth L. A. and S. Bhatnagar,
Reinforcement learning with function approximation for traffic signal control
IEEE Transactions on Intelligent Transportation Systems, 2011.


Prashanth L.A., H.L.Prasad, N.Desai, S.Bhatnagar and G.Dasgupta,
Stochastic optimization for adaptive labor staffing in service systems,
Intl. Conf. on Service Oriented Computing, 2011.

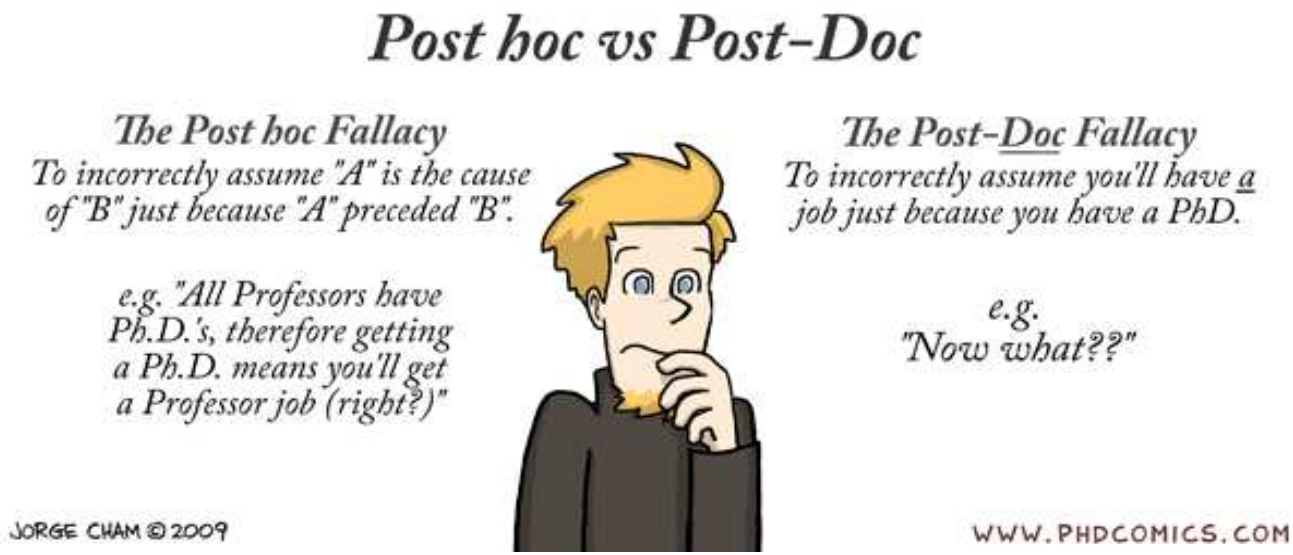

Prashanth L.A. and S.Bhatnagar,
Reinforcement Learning with Average Cost for Adaptive Control of Traffic Lights at Intersections,
IEEE Conference on Intelligent Transportation Systems, 2011.

Publications II


S. Bhatnagar, V. Borkar and **Prashanth.L.A.**,
Adaptive Feature Pursuit: Online Adaptation of Features in Reinforcement Learning,
Reinforcement Learning and Approximate Dynamic Programming for Feedback Control, by F. Lewis and D. Liu (eds.), IEEE Press Computational Intelligence Series.


S.Bhatnagar, H.L.Prasad and **Prashanth.L.A.**,
Stochastic Recursive Algorithms for Optimization: Simultaneous Perturbation Methods,
Lecture Notes in Control and Information Sciences Series, Springer (Accepted), 2012.

What next?


Post hoc vs Post-Doc
The Post hoc Fallacy
To incorrectly assume "A" is the cause of "B" just because "A" preceded "B".
e.g. "All Professors have Ph.D.'s, therefore getting a Ph.D. means you'll get a Professor job (right?)"
The Post-Doc Fallacy
To incorrectly assume you'll have a job just because you have a PhD.
e.g. "Now what??"
JORGE CHAM © 2009
WWW.PHDCOMICS.COM