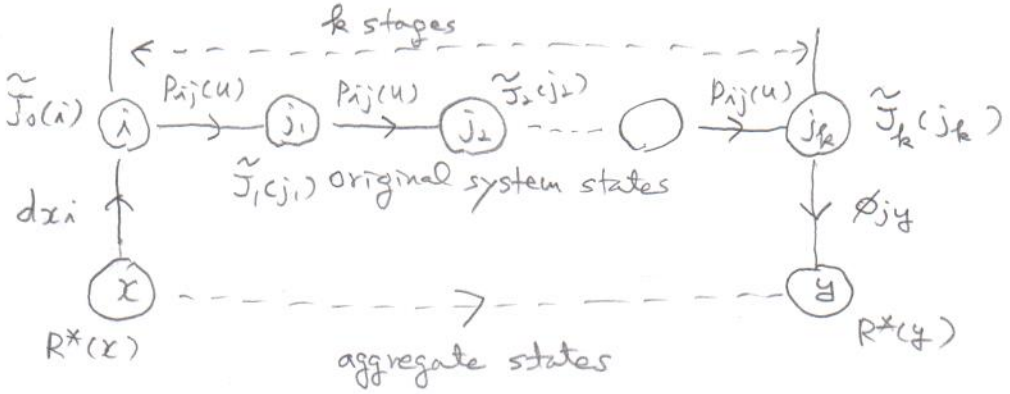
Multistep Aggregation



- Bellman equation for R^*

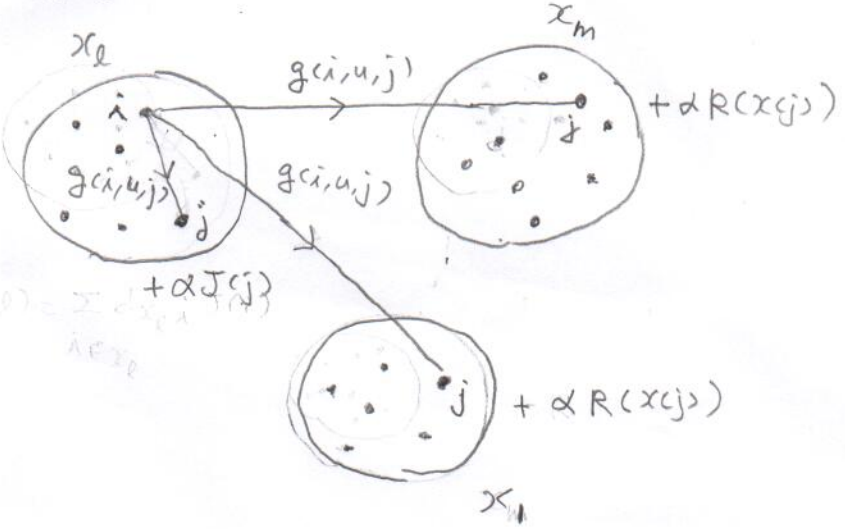
$$R^* = DT^k \Phi R^* \qquad (56)$$

- Improved error bound

$$\| J_{\tilde{m}} - J^* \|_\infty \leq \frac{2\alpha^k \delta}{1-\alpha^k}$$

Asynchronous Distributed Aggregation

- Partition original system states into a set of aggregate states $A = \{x_1, \dots x_m$
  i.e., hard aggregation



- Bellman equation for each $i \in x_\ell$

$$\triangleq H_\ell(i, u, J, R)$$

$$J(i) = \min_{u \in U(i)} \left(\sum_{j=1}^{n} P_{ij}(u)\, g(i,u,j) + \alpha \sum_{j \in x_\ell} P_{ij}(u)\, J(j) + \sum_{j \notin x_\ell} P_{ij}(u)\, R(x(j)) \right)$$

where $R(\ell) = \sum_{i \in x_\ell} d_{x_\ell i}\, J(i)$ $\qquad (57)$

- Synchronous distributed VI method

Each processor $\ell = 1, \ldots, m$ maintains/updates a local cost $J(i)$ for $i \in X_\ell$ and maintains/updates/communicates an aggregate cost

$$R(\ell) = \sum_{i \in X_\ell} d_{x_\ell i} J(i)$$

according to the following update rule:

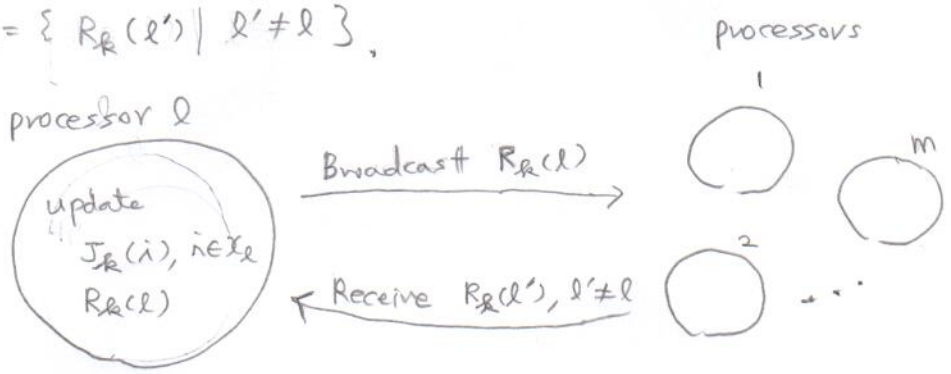
$$J_{k+1}(i) = \min_{u \in U(i)} H_\ell(i, u, J_k, R_k) \qquad (58)$$

$$R_k(\ell) = \sum_{i \in X_\ell} d_{x_\ell i} J_k(i)$$

where $H_\ell(i, u, J_k, R_k) = \sum_{j=1}^{n} P_{ij}(u) g(i, u, j) + \alpha \sum_{j \in X_\ell} P_{ij}(u) J_k(j) + \alpha \sum_{j \notin X_\ell} P_{ij}(u) R_k(x(j))$

$J_k = \{ J_k(i) \mid i \in X_\ell \}$

$R_k = \{ R_k(\ell') \mid \ell' \neq \ell \}$.



- Asynchronous distributed VI method

$$J_{k+1}(i) = \begin{cases} \min_{u \in U(i)} H_i(i, u, J_k, R_{\tau_{1,k}}(1), \ldots, R_{\tau_{m,k}}(m)), & \forall i \in I_{\ell,k} \\ J_k(i) & \text{otherwise} \end{cases} \qquad (59)$$

where $0 \leq \tau_{\ell,k} \leq k$ for $\ell = 1, \ldots, m$,

$$R_\tau(\ell) = \sum_{i \in X_\ell} d_{x_\ell i} J_\tau(i), \quad \forall \ell = 1, \ldots, m$$

and

$I_{\ell,k}$: the set of states $i \in X_\ell$ which are updated at time k

$\tau_{\ell,k}$: the time when the corresponding aggregate cost, lastly reported to processor ℓ by time k, was computed at the corresponding processor

$k - \tau_{\ell,k}$: delays between the current time k and the time $\tau_{\ell,k}$ when the corresponding aggregate costs were computed at other processors.

- Convergence of the asynchronous distributed VI method

a) Total asynchronism assumption

Every $i \in X_\ell$ belongs to $I_{\ell,k}$ for infinitely many k (so each cost component is updated infinitely often) and $\lim_{k\to\infty} \tau_{\ell,k} = \infty$ for all $\ell = 1, \ldots, m$ (so that processors eventually communicate more recently computed aggregate costs to other processors.

b) Asynchronous convergence theorem

The maximum-norm contraction property of the mapping underlying the synchronous iteration (58) establishes the total asynchronous convergen

Proof) Refer to Chapter 2 of the text book "Dynamic programming and Optimal Control : Approximate Dynamic Programming", Vol. 2, 4th Edition, Athena Scientific 2012.

Refer to "Parallel and Distributed Computation : Numerical Methods", Athena Scientific 2015.

Q-Learning

- Used when there is no explicit model of the system, i.e., a model-free method
- Instead of approximating the cost function of a particular policy, it updates Q-factors with an optimal policy, thereby avoiding the multiple policy evaluation steps of the PI method.

Q-Learning : A stochastic VI Algorithm

- The optimal Q-factors are defined for all pairs (i,u) with $u \in U(i)$, by

$$Q^*(i,u) = \sum_{j=1}^{n} P_{ij}(u)\,(g(i,u,j) + \alpha J(j)). \quad \text{---------- (60)}$$

- Note that

optimal Q-factor = expected 1st stage cost when action u is applied + expected cost-to-go associated with an optimal policy when action u is applied.

- In contrast, the Bellman's equation for J_μ is defined for all i by

$$J_\mu(i) = \sum_{j=1}^{n} P_{ij}(\mu(i))\,(g(i,\mu(i),j) + \alpha J_\mu(j)) \qquad J_\mu = T_\mu J_\mu$$

i.e., expected 1st stage cost associated with a policy μ + expected cost-to-go associated with a policy μ

- The Bellman's equation for J is defined for all i by

$$J(i) = \min_{u \in U(i)} \sum_{j=1}^{n} P_{ij}(u)\,(g(i,u,j) + \alpha J(j)) \qquad J = TJ \quad \text{---------- (61)}$$

i.e., minimum$_u$ [expected 1st stage cost when action u is applied + expected cost-to-go associated with an optimal policy when action u is applied]

- By combining (60) and (61), we obtain the Bellman's equation for $Q^*(i,u)$ a

$$Q^*(i,u) = \sum_{j=1}^{n} p_{ij}(u)\left(g(i,u,j) + \alpha \min_{v \in U(j)} Q^*(j,v)\right) \quad \text{——— (62)}$$

and the relationship between $J(i)$ and $Q^*(i,u)$ as

$$J^*(i) = \min_{u \in U(i)} Q^*(i,u) \quad \text{——— (63)}$$

- The proof of existence and uniqueness of the fixed point of the Bellman equation (62) is the same as the proof of that of the Bellman equation for J.

- VI method for Q^*

$$Q_{k+1} = F Q_k$$

where

$$(FQ)(i,u) = \sum_{j=1}^{n} p_{ij}(u)\left(g(i,u,j) + \alpha \min_{v \in U(j)} Q(j,u)\right), \quad \forall (i,u)$$

——— (

- One can show that F is a maximum-norm contraction with modulus α so that the VI algorithm converges to Q^* from every starting point Q_0.

- Simulation-based VI method for Q^*

Generate an infinitely long sequence of state-control pairs $\{(i_k, u_k)\}$

Given the pair (i_k, u_k), generate next state j_k according to the probabilities $p_{i_k j}(u_k)$.

For each (i_k, u_k, j_k) generated, update Q-factor of (i_k, u_k) as follows:

$$Q_{k+1}(i,u) = (1-\gamma_k) Q_k(i,u) + \gamma_k (F_k Q_k)(i,u), \quad \forall (i,u)$$

Where

$$(F_k Q_k)(i,u) = \begin{cases} g(i_k, u_k, j_k) + \alpha \min_{v \in U(j_k)} Q_k(j_k, v) & \text{if } (i,u) = (i_k, u_k) \\ Q_k(i,u) & \text{if } (i,u) \neq (i_k, u_k) \end{cases}$$

— (65)

- Note that $(F_k Q_k)(i_k, u_k)$ in (65) is a single sample approximation of the expected value defining $(FQ_k)(i_k, u_k)$ in (64).

- The stepsize γ_k should satisfy

$$\gamma_k > 0, \ \forall k, \quad \sum_{k=0}^{\infty} \gamma_k = \infty, \quad \sum_{k=0}^{\infty} \gamma_k^2 < \infty$$

which are typical of stochastic approximation methods, as for example when $\gamma_k = c_1/(k+c_2)$ where c_1 and c_2 are some positive constants.

- The rigorous convergence proof is beyond our scope; see Tsitsiklis, J. N., "Asynchronous stochastic Approximation and Q-Learning", Vol. 16, pp. 185-202, which embeds Q-learning with a broad class of asynchronous stochastic approximation algorithms.

- The simulation-based VI method in (65), which belongs to a class of asynchronous stochastic approximation algorithms, is the original and most widely known Q-learning algorithm. This algorithm was originally reported in C. J. C. H. Watkins, "Learning from Delayed Rewards," Ph.D. Thesis, Cambridge University.

- In practice, Q-learning has some drawbacks, the most important of which is that the number of Q-factors/state-control pairs (i,u) may be excessive.

  $\Rightarrow$ linear approximation architecture for Q and/or state aggregation