6.231 DYNAMIC PROGRAMMING

LECTURE 16

LECTURE OUTLINE

- Review of computational theory of discounted problems
- Value iteration (VI), policy iteration (PI)
- Optimistic PI
- Computational methods for generalized discounted DP
- Asynchronous algorithms

DISCOUNTED PROBLEMS

- Stationary system with arbitrary state space

$$x_{k+1} = f(x_k, u_k, w_k), \qquad k = 0, 1, \ldots$$

- Bounded g. Cost of a policy $\pi = \{\mu_0, \mu_1, \ldots\}$

$$J_\pi(x_0) = \lim_{N \to \infty} \mathop{E}_{\substack{w_k \\ k=0,1,\ldots}} \left\{ \sum_{k=0}^{N-1} \alpha^k g\big(x_k, \mu_k(x_k), w_k\big) \right\}$$

- Shorthand notation for DP mappings (n-state Markov chain case)

$$(TJ)(x) = \min_{u \in U(x)} E\big\{g(x, u, w) + \alpha J\big(f(x, u, w)\big)\big\}, \ \forall \, x$$

TJ is the optimal cost function for the one-stage problem with stage cost g and terminal cost αJ.

- For any stationary policy μ

$$(T_\mu J)(x) = E\big\{g(x, \mu(x), w) + \alpha J\big(f(x, \mu(x), w)\big)\big\}, \ \forall \, x$$

Note: T_μ is linear [in short $T_\mu J = P_\mu(g_\mu + \alpha J)$].

"SHORTHAND" THEORY – A SUMMARY

- Cost function expressions (with $J_0 \equiv 0$)

$$J_\pi = \lim_{k\to\infty} T_{\mu_0} T_{\mu_1} \cdots T_{\mu_k} J_0, \qquad J_\mu = \lim_{k\to\infty} T_\mu^k J_0$$

- Bellman's equation: $J^* = TJ^*$, $J_\mu = T_\mu J_\mu$
- Optimality condition:

$$\mu\text{: optimal} \quad <==> \quad T_\mu J^* = TJ^*$$

- Contraction: $\|TJ_1 - TJ_2\| \leq \alpha \|J_1 - J_2\|$
- Value iteration: For any (bounded) J

$$J^* = \lim_{k\to\infty} T^k J$$

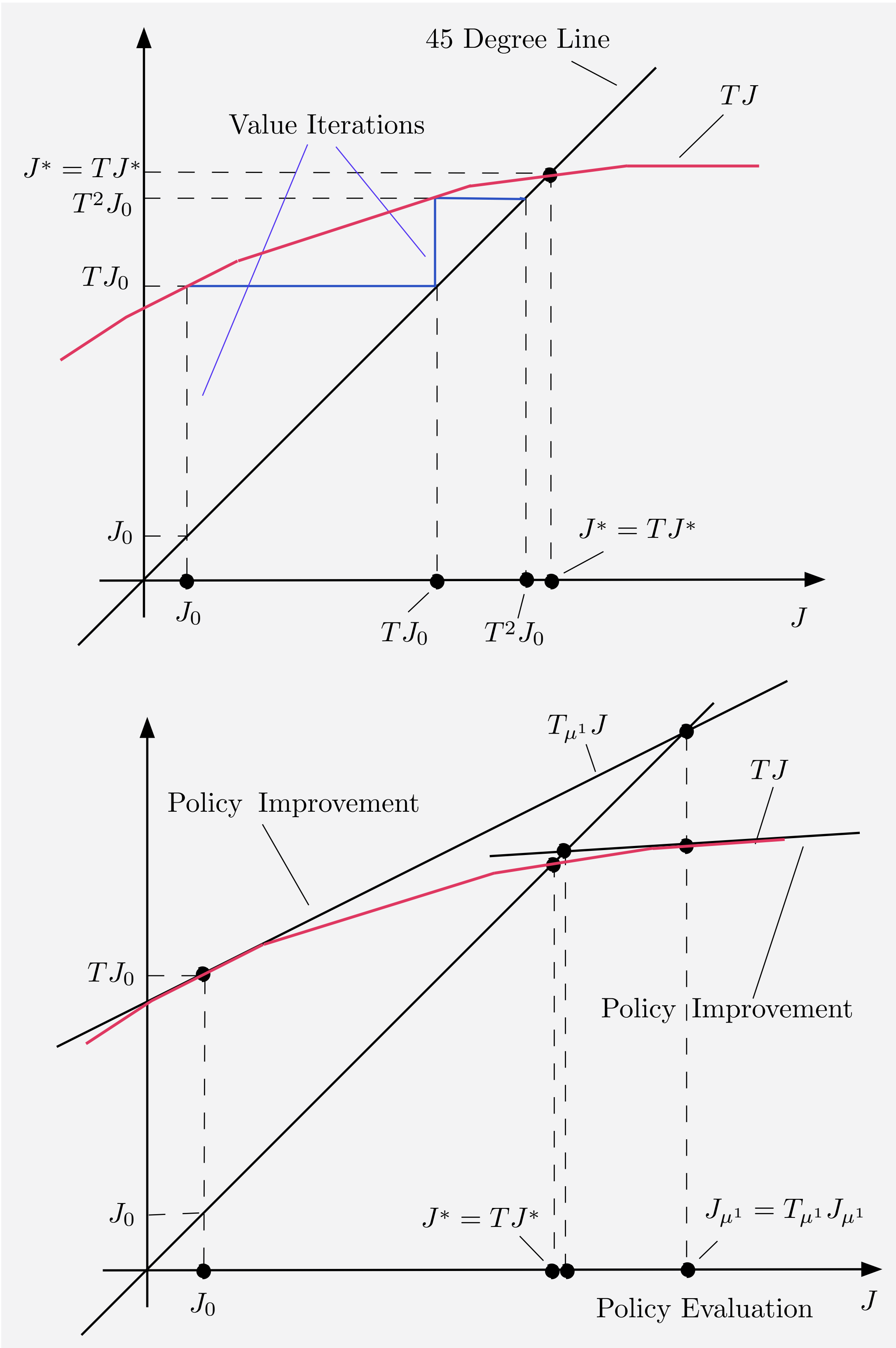
- Policy iteration: Given μ^k,
  - Policy evaluation: Find J_{μ^k} by solving

$$J_{\mu^k} = T_{\mu^k} J_{\mu^k}$$

  - Policy improvement: Find μ^{k+1} such that

$$T_{\mu^{k+1}} J_{\mu^k} = TJ_{\mu^k}$$

INTERPRETATION OF VI AND PI



VI AND PI METHODS FOR Q-LEARNING

- We can write Bellman's equation as

$$J^*(i) = \min_{u \in U(i)} Q^*(i,u) \qquad i = 1, \ldots, n,$$

where Q^* is the vector of optimal Q-factors

$$Q^*(i,u) = \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J^*(j)\big)$$

- VI and PI for Q-factors are mathematically equivalent to VI and PI for costs.
- They require equal amount of computation ... they just need more storage.
- For example, we can write the VI method as

$$J_{k+1}(i) = \min_{u \in U(i)} Q_{k+1}(i,u), \qquad i = 1, \ldots, n,$$

where Q_{k+1} is generated for all i and $u \in U(i)$ by

$$Q_{k+1}(i,u) = \sum_{j=1}^{n} p_{ij}(u)\left(g(i,u,j) + \alpha \min_{v \in U(j)} Q_k(j,v)\right)$$

APPROXIMATE PI

- Suppose that the policy evaluation is approximate, according to,

$$\max_x |J_k(x) - J_{\mu^k}(x)| \leq \delta, \qquad k = 0, 1, \ldots$$

and policy improvement is approximate, according to,

$$\max_x |(T_{\mu^{k+1}} J_k)(x) - (T J_k)(x)| \leq \epsilon, \qquad k = 0, 1, \ldots$$

where δ and ϵ are some positive scalars.

- Error Bound: The sequence $\{\mu^k\}$ generated by approximate policy iteration satisfies

$$\limsup_{k \to \infty} \max_{x \in S} \big(J_{\mu^k}(x) - J^*(x) \big) \leq \frac{\epsilon + 2\alpha\delta}{(1 - \alpha)^2}$$

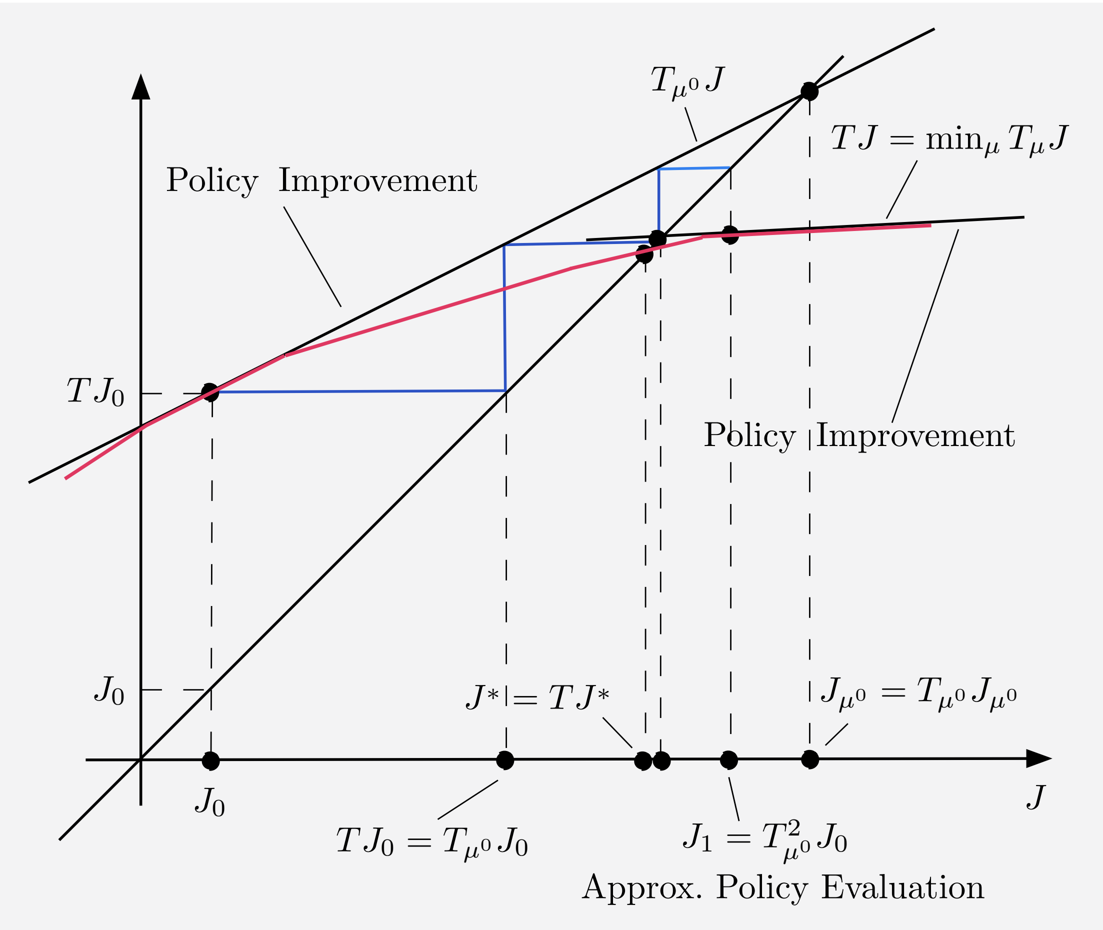
- Typical practical behavior: The method makes steady progress up to a point and then the iterates J_{μ^k} oscillate within a neighborhood of J^*.

OPTIMISTIC PI

- This is PI, where policy evaluation is carried out by a finite number of VI
- Shorthand definition: For some integers m_k

$$T_{\mu^k} J_k = T J_k, \qquad J_{k+1} = T_{\mu^k}^{m_k} J_k, \qquad k = 0, 1, \ldots$$

  - If $m_k \equiv 1$ it becomes VI
  - If $m_k = \infty$ it becomes PI
  - For intermediate values of m_k, it is generally more efficient than either VI or PI



EXTENSIONS TO GENERALIZED DISC. DP

- All the preceding VI and PI methods extend to generalized/abstract discounted DP.

- Summary: For a mapping $H : X \times U \times R(X) \mapsto \Re$, consider

$$(TJ)(x) = \min_{u \in U(x)} H(x, u, J), \qquad \forall\, x \in X.$$

$$(T_\mu J)(x) = H\big(x, \mu(x), J\big), \qquad \forall\, x \in X.$$

- We want to find J^* such that

$$J^*(x) = \min_{u \in U(x)} H(x, u, J^*), \qquad \forall\, x \in X$$

and a μ^* such that $T_{\mu^*} J^* = T J^*$.

- Discounted, Discounted Semi-Markov, Minimax

$$H(x, u, J) = E\Big\{g(x, u, w) + \alpha J\big(f(x, u, w)\big)\Big\}$$

$$H(x, u, J) = G(x, u) + \sum_{y=1}^{n} m_{xy}(u) J(y)$$

$$H(x, u, J) = \max_{w \in W(x,u)} \Big[g(x, u, w) + \alpha J\big(f(x, u, w)\big)\Big]$$

ASSUMPTIONS AND RESULTS

- Monotonicity assumption: If $J, J' \in R(X)$ and $J \leq J'$, then

$$H(x,u,J) \leq H(x,u,J'), \qquad \forall\, x \in X,\ u \in U(x)$$

- Contraction assumption:
  - For every $J \in B(X)$, the functions $T_\mu J$ and TJ belong to $B(X)$.
  - For some $\alpha \in (0,1)$ and all $J, J' \in B(X)$, H satisfies

$$\big|H(x,u,J) - H(x,u,J')\big| \leq \alpha \max_{y \in X} \big|J(y) - J'(y)\big|$$

for all $x \in X$ and $u \in U(x)$.

- Standard algorithmic results extend:
  - Generalized VI converges to J^*, the unique fixed point of T
  - Generalized PI and optimistic PI generate $\{\mu^k\}$ such that

$$\lim_{k\to\infty} \|J_{\mu^k} - J^*\| = 0, \qquad \lim_{k\to\infty} \|J_k - J^*\| = 0$$

- Analytical Approach: Start with a problem, match it with an H, invoke the general results.

ASYNCHRONOUS ALGORITHMS

- Motivation for asynchronous algorithms
  - Faster convergence
  - Parallel and distributed computation
  - Simulation-based implementations

- General framework: Partition X into disjoint nonempty subsets $X_1, \ldots, X_m$, and use separate processor ℓ updating $J(x)$ for $x \in X_\ell$.

- Let J be partitioned as $J = (J_1, \ldots, J_m)$, where J_ℓ is the restriction of J on the set X_ℓ.

- Synchronous algorithm: Processor ℓ updates J for the states $x \in X_\ell$ at all times t,

$$J_\ell^{t+1}(x) = T(J_1^t, \ldots, J_m^t)(x), \quad x \in X_\ell, \; \ell = 1, \ldots, m$$

- Asynchronous algorithm: Processor ℓ updates J for the states $x \in X_\ell$ only at a subset of times $\mathcal{R}_\ell$,

$$J_\ell^{t+1}(x) = \begin{cases} T\big(J_1^{\tau_{\ell 1}(t)}, \ldots, J_m^{\tau_{\ell m}(t)}\big)(x) & \text{if } t \in \mathcal{R}_\ell, \\ J_\ell^t(x) & \text{if } t \notin \mathcal{R}_\ell \end{cases}$$

where $t - \tau_{\ell j}(t)$ are communication “delays”

ONE-STATE-AT-A-TIME ITERATIONS

- Important special case: Assume n "states", a separate processor for each state, and no delays

- Generate a sequence of states $\{x^0, x^1, \ldots\}$, generated in some way, possibly by simulation (each state is generated infinitely often)

- Asynchronous VI: Change any one component of J^t at time t, the one that corresponds to x^t:

$$J^{t+1}(\ell) = \begin{cases} T\big(J^t(1), \ldots, J^t(n)\big)(\ell) & \text{if } \ell = x^t, \\ J^t(\ell) & \text{if } \ell \neq x^t, \end{cases}$$

- The special case where

$$\{x^0, x^1, \ldots\} = \{1, \ldots, n, 1, \ldots, n, 1, \ldots\}$$

is the Gauss-Seidel method

- More generally, the components used at time t are delayed by $t - \tau_{\ell j}(t)$

- Flexible in terms of timing and "location" of the iterations

- We can show that $J^t \to J^*$ under assumptions typically satisfied in DP

ASYNCHRONOUS CONV. THEOREM I

- Assume that for all $\ell, j = 1, \ldots, m$, the set of times $\mathcal{R}_\ell$ is infinite and $\lim_{t\to\infty} \tau_{\ell j}(t) = \infty$

- Proposition: Let T have a unique fixed point J^*, and assume that there is a sequence of nonempty subsets $\{S(k)\} \subset R(X)$ with $S(k+1) \subset S(k)$ for all k, and with the following properties:

  (1) Synchronous Convergence Condition: Every sequence $\{J^k\}$ with $J^k \in S(k)$ for each k, converges pointwise to J^*. Moreover, we have

  $$TJ \in S(k+1), \qquad \forall\, J \in S(k),\ k = 0, 1, \ldots .$$

  (2) Box Condition: For all k, $S(k)$ is a Cartesian product of the form

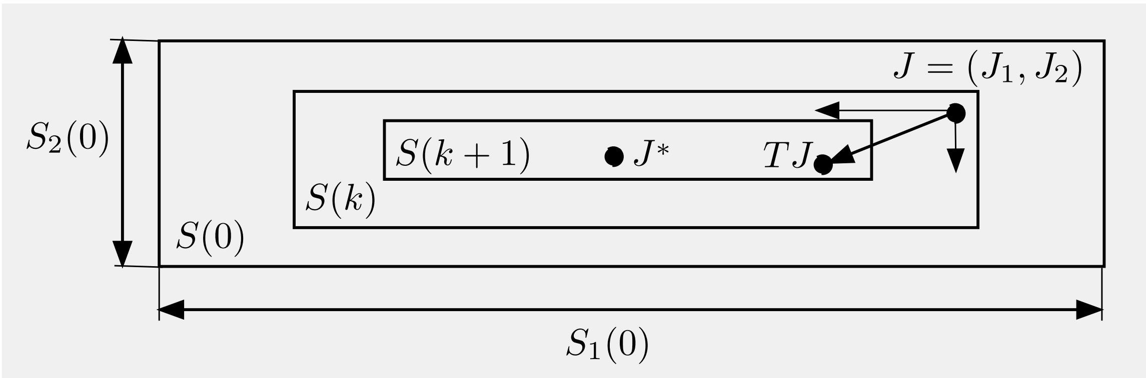
  $$S(k) = S_1(k) \times \cdots \times S_m(k),$$

  where $S_\ell(k)$ is a set of real-valued functions on X_ℓ, $\ell = 1, \ldots, m$.

Then for every $J \in S(0)$, the sequence $\{J^t\}$ generated by the asynchronous algorithm converges pointwise to J^*.

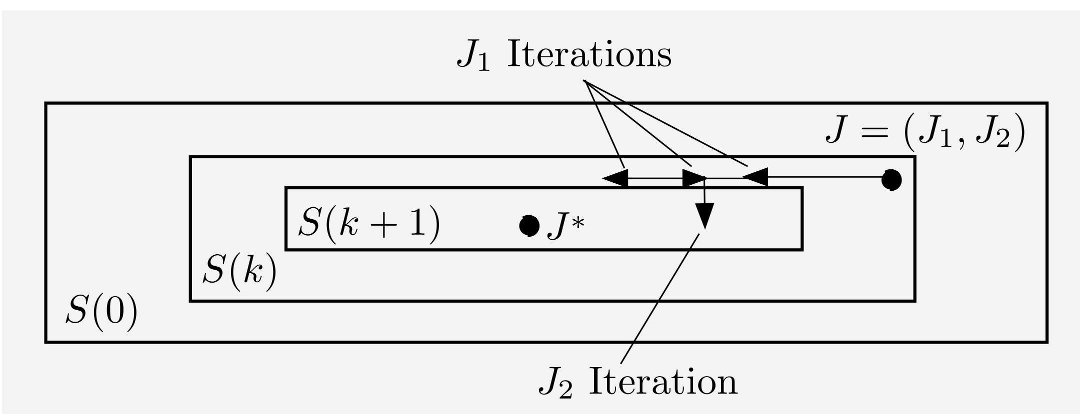
ASYNCHRONOUS CONV. THEOREM II

- Interpretation of assumptions:



A synchronous iteration from any J in $S(k)$ moves into $S(k+1)$ (component-by-component)

- Convergence mechanism:



Key: "Independent" component-wise improvement. An asynchronous component iteration from any J in $S(k)$ moves into the corresponding component portion of $S(k+1)$ permanently!

PRINCIPAL DP APPLICATIONS

- The assumptions of the asynchronous convergence theorem are satisfied in two principal cases:
  - When T is a (weighted) sup-norm contraction.
  - When T is monotone and the Bellman equation $J = TJ$ has a unique solution.
- The theorem can be applied also to convergence of asynchronous optimistic PI for:
  - Discounted problems (Section 2.6.2 of the text).
  - SSP problems (Section 3.5 of the text).
- There are variants of the theorem that can be applied in the presence of special structure.
- Asynchronous convergence ideas also underlie stochastic VI algorithms like Q-learning.