6.231 DYNAMIC PROGRAMMING

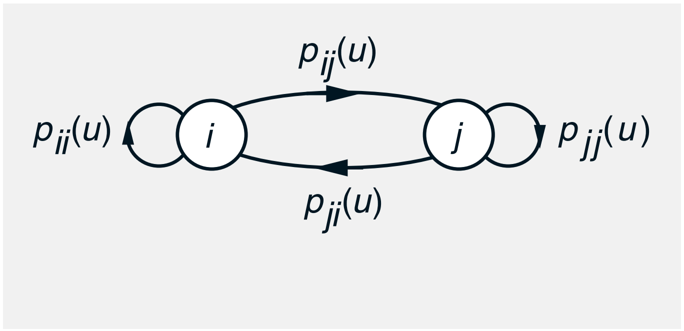
LECTURE 5

LECTURE OUTLINE

- Review of approximate PI
- Review of approximate policy evaluation based on projected Bellman equations
- Exploration enhancement in policy evaluation
- Oscillations in approximate PI
- Aggregation – An alternative to the projected equation/Galerkin approach
- Examples of aggregation
- Simulation-based aggregation

DISCOUNTED MDP

- System: Controlled Markov chain with states $i = 1, \ldots, n$ and finite set of controls $u \in U(i)$
- Transition probabilities: $p_{ij}(u)$



- Cost of a policy $\pi = \{\mu_0, \mu_1, \ldots\}$ starting at state i:

$$J_\pi(i) = \lim_{N \to \infty} E\left\{ \sum_{k=0}^{N} \alpha^k g\big(i_k, \mu_k(i_k), i_{k+1}\big) \mid i = i_0 \right\}$$

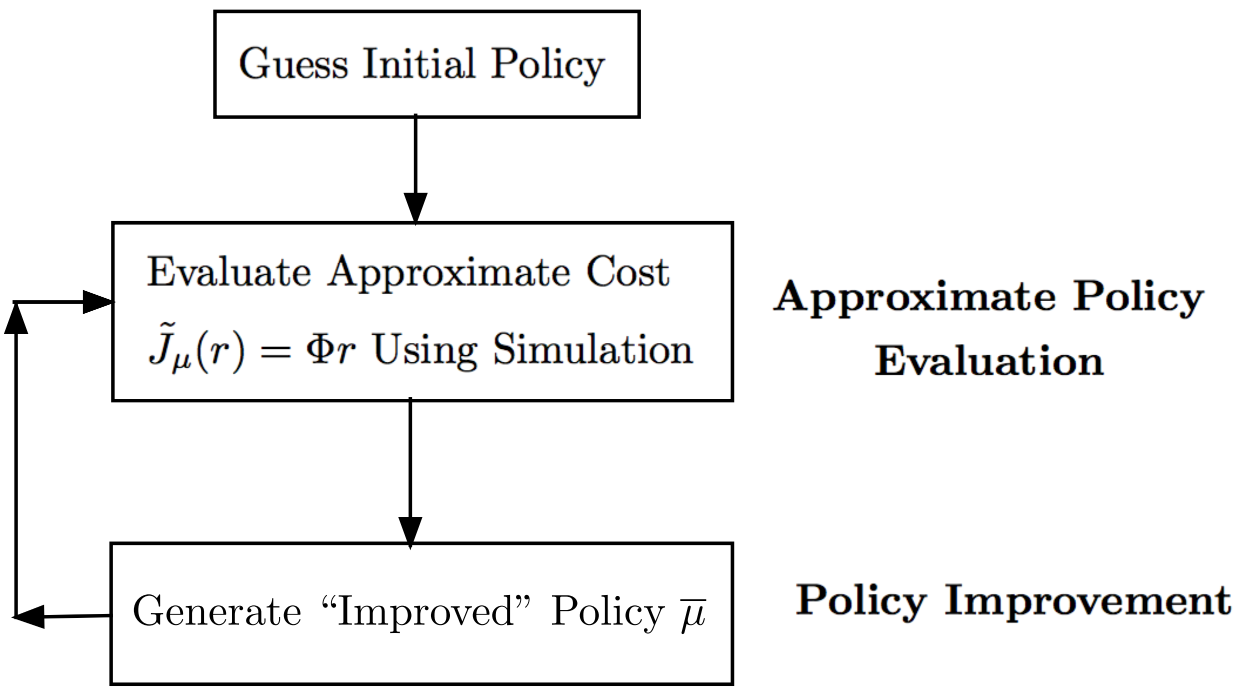
with $\alpha \in [0, 1)$

- Shorthand notation for DP mappings

$$(TJ)(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i, u, j) + \alpha J(j)\big), \quad i = 1, \ldots, n,$$

$$(T_\mu J)(i) = \sum_{j=1}^{n} p_{ij}\big(\mu(i)\big)\Big(g\big(i, \mu(i), j\big) + \alpha J(j)\Big), \quad i = 1, \ldots, n$$

APPROXIMATE PI


Guess Initial Policy
Evaluate Approximate Cost $\tilde{J}_\mu(r) = \Phi r$ Using Simulation
Approximate Policy Evaluation
Generate "Improved" Policy $\overline{\mu}$
Policy Improvement

- Evaluation of typical policy μ: Linear cost function approximation

$$\tilde{J}_\mu(r) = \Phi r$$

where Φ is full rank $n \times s$ matrix with columns the basis functions, and ith row denoted $\phi(i)'$.

- Policy "improvement" to generate $\overline{\mu}$:

$$\overline{\mu}(i) = \arg \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha \phi(j)' r\big)$$

EVALUATION BY PROJECTED EQUATIONS

- We discussed approximate policy evaluation by solving the projected equation

$$\Phi r = \Pi T_\mu(\Phi r)$$

Π: projection with a weighted Euclidean norm

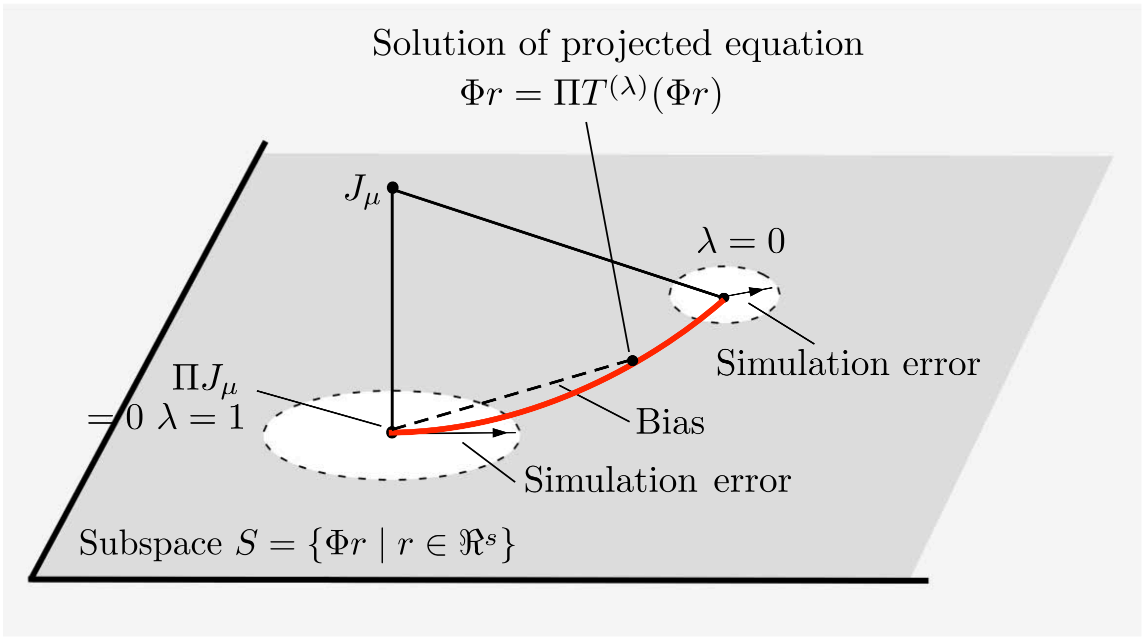
- Implementation by simulation (single long trajectory using current policy - important to make ΠT_μ a contraction). LSTD, LSPE methods.

- Multistep option: SolveΦ $r = \Pi T_\mu^{(\lambda)}(\Phi r)$ with

$$T_\mu^{(\lambda)} = (1-\lambda)\sum_{\ell=0}^{\infty} \lambda^\ell T_\mu^{\ell+1}$$

  – As $\lambda \uparrow 1$,Π $T^{(\lambda)}$ becomes a contraction for any projection norm

  – Bias-variance tradeoff


Solution of projected equation
$\Phi r = \Pi T^{(\lambda)}(\Phi r)$
J_μ
$\lambda = 0$
Simulation error
ΠJ_μ
$= 0\ \lambda = 1$
Bias
Simulation error
Subspace $S = \{\Phi r \mid r \in \Re^s\}$

POLICY ITERATION ISSUES: EXPLORATION

- **1st major issue:** exploration. To evaluate μ, we need to generate cost samples using μ
- This biases the simulation by underrepresenting states that are unlikely to occur under μ.
- As a result, the cost-to-go estimates of these underrepresented states may be highly inaccurate.
- This seriously impacts the improved policy $\overline{\mu}$.
- This is known as inadequate exploration - a particularly acute difficulty when the randomness embodied in the transition probabilities is “relatively small” (e.g., a deterministic system).
- Common remedy is the off-policy approach: Replace P of current policy with a “mixture”

$$\overline{P} = (I - B)P + BQ$$

where B is diagonal with diagonal components in $[0, 1]$ and Q is another transition matrix.

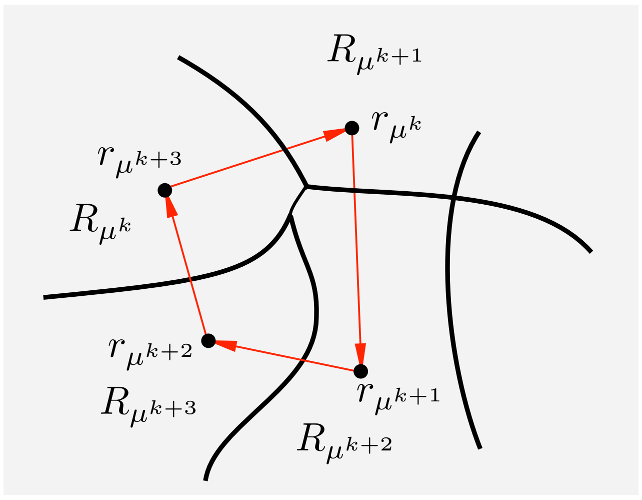
- LSTD and LSPE formulas must be modified ... otherwise the policy $\overline{P}$ (not P) is evaluated. Related methods and ideas: importance sampling, geometric and free-form sampling (see the text).

POLICY ITERATION ISSUES: OSCILLATIONS

- **2nd major issue:** oscillation of policies
- Analysis using the greedy partition: R_μ is the set of parameter vectors r for which μ is greedy with respect to $\tilde{J}(\cdot, r) = \Phi r$

$$R_\mu = \big\{ r \mid T_\mu(\Phi r) = T(\Phi r) \big\}$$

- There is a finite number of possible vectors r_μ, one generated from another in a deterministic way



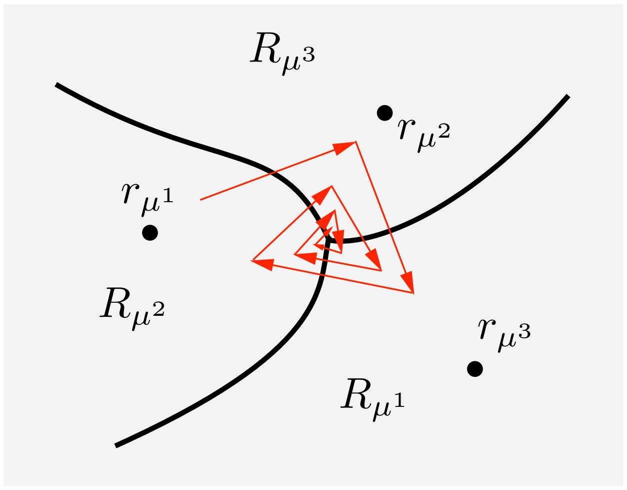
- The algorithm ends up repeating some cycle of policies $\mu^k, \mu^{k+1}, \ldots, \mu^{k+m}$ with

$$r_{\mu^k} \in R_{\mu^{k+1}},\ r_{\mu^{k+1}} \in R_{\mu^{k+2}}, \ldots, r_{\mu^{k+m}} \in R_{\mu^k};$$

- Many different cycles are possible

MORE ON OSCILLATIONS/CHATTERING

- In the case of optimistic policy iteration a different picture holds



- Oscillations are less violent, but the "limit" point is meaningless!

- Fundamentally, oscillations are due to the lack of monotonicity of the projection operator, i.e., $J \leq J'$ does not imply$\Pi\, J \leq \Pi J'$.

- If approximate PI uses policy evaluation

$$\Phi r = (WT_\mu)(\Phi r)$$

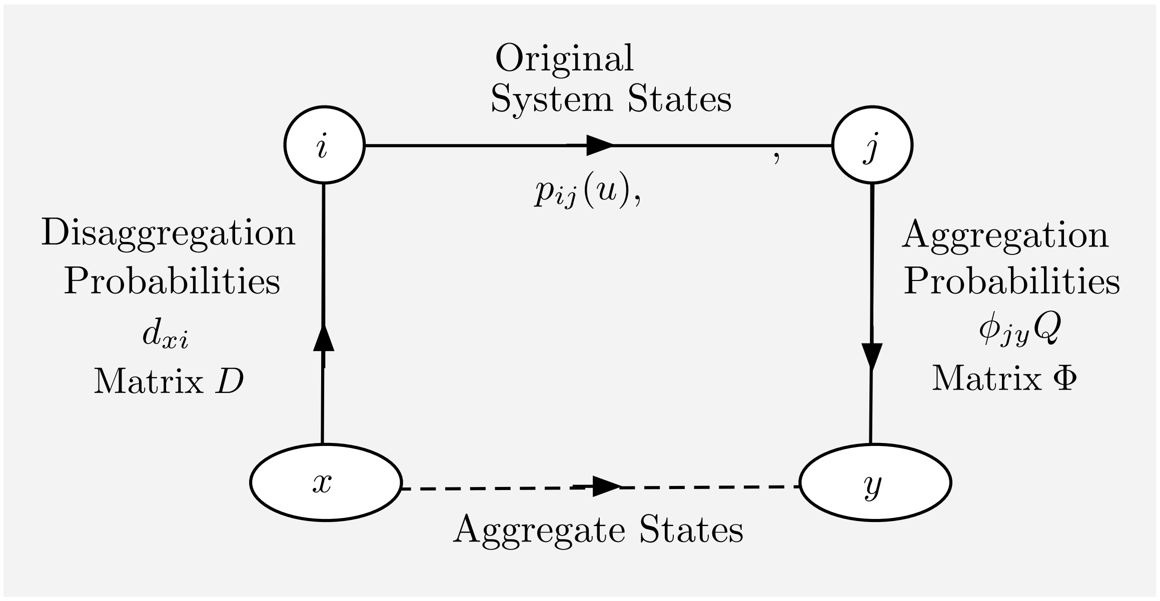
with W a monotone operator, the generated policies converge (to a possibly nonoptimal limit).

- The operator W used in the aggregation approach has this monotonicity property.

PROBLEM APPROXIMATION - AGGREGATION

- Another major idea in ADP is to approximate the cost-to-go function of the problem with the cost-to-go function of a simpler problem.
- The simplification is often ad-hoc/problem-dependent.
- Aggregation is a systematic approach for problem approximation. Main elements:
  - Introduce a few "aggregate" states, viewed as the states of an "aggregate" system
  - Define transition probabilities and costs of the aggregate system, by relating original system states with aggregate states
  - Solve (exactly or approximately) the "aggregate" problem by any kind of VI or PI method (including simulation-based methods)
  - Use the optimal cost of the aggregate problem to approximate the optimal cost of the original problem
- Hard aggregation example: Aggregate states are subsets of original system states, treated as if they all have the same cost.

AGGREGATION/DISAGGREGATION PROBS



- The aggregate system transition probabilities are defined via two (somewhat arbitrary) choices
- For each original system state j and aggregate state y, the aggregation probability ϕ_{jy}
  - Roughly, the "degree of membership of j in the aggregate state y."
  - In hard aggregation, $\phi_{jy} = 1$ if state j belongs to aggregate state/subset y.
- For each aggregate state x and original system state i, the disaggregation probability d_{xi}
  - Roughly, the "degree to which i is representative of x."
  - In hard aggregation, equal d_{xi}

AGGREGATE SYSTEM DESCRIPTION

- The transition probability from aggregate state x to aggregate state y under control u

$$\hat{p}_{xy}(u) = \sum_{i=1}^{n} d_{xi} \sum_{j=1}^{n} p_{ij}(u)\phi_{jy}, \quad \text{or } \hat{P}(u) = DP(u)\Phi$$

where the rows of D and Φ are the disaggregation and aggregation probs.

- The expected transition cost is

$$\hat{g}(x,u) = \sum_{i=1}^{n} d_{xi} \sum_{j=1}^{n} p_{ij}(u)g(i,u,j), \quad \text{or } \hat{g} = DPg$$

- The optimal cost function of the aggregate problem, denoted $\hat{R}$, is

$$\hat{R}(x) = \min_{u\in U}\left[\hat{g}(x,u) + \alpha\sum_{y}\hat{p}_{xy}(u)\hat{R}(y)\right], \qquad \forall\, x$$

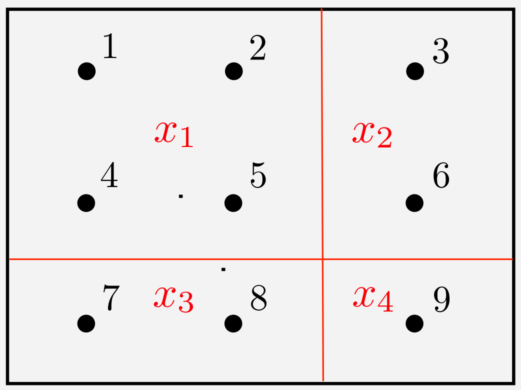
Bellman's equation for the aggregate problem.

- The optimal cost function J^* of the original problem is approximated by $\tilde{J}$ given by

$$\tilde{J}(j) = \sum_{y}\phi_{jy}\hat{R}(y), \qquad \forall\, j$$

EXAMPLE I: HARD AGGREGATION

- Group the original system states into subsets, and view each subset as an aggregate state
- Aggregation probs.: $\phi_{jy} = 1$ if j belongs to aggregate state y.

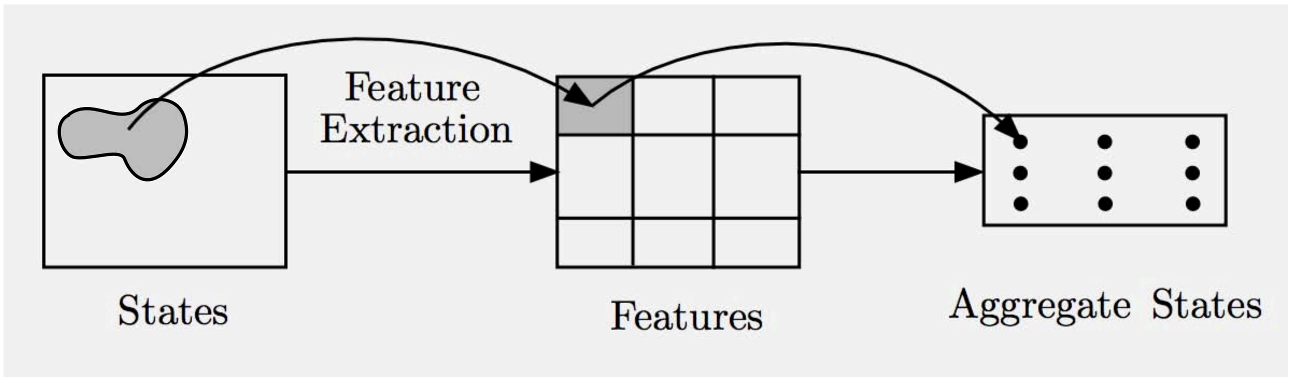


$$\Phi = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Disaggregation probs.: There are many possibilities, e.g., all states i within aggregate state x have equal prob. d_{xi}.
- If optimal cost vector J^* is piecewise constant over the aggregate states/subsets, hard aggregation is exact. Suggests grouping states with "roughly equal" cost into aggregates.
- A variant: Soft aggregation (provides "soft boundaries" between aggregate states).

EXAMPLE II: FEATURE-BASED AGGREGATION

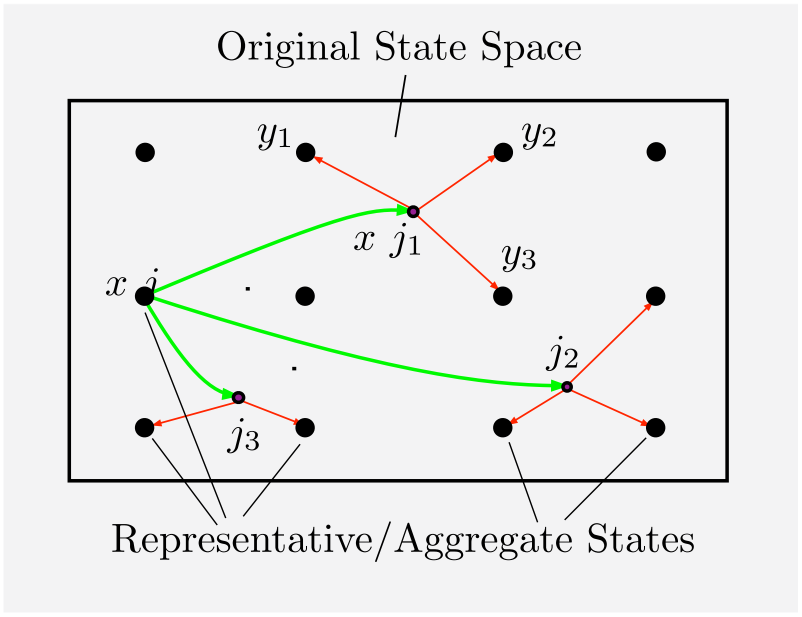
- Important question: How do we group states together?
- If we know good features, it makes sense to group together states that have “similar features”



- A general approach for passing from a feature-based state representation to an aggregation-based architecture
- Essentially discretize the features and generate a corresponding piecewise constant approximation to the optimal cost function
- Aggregation-based architecture is more powerful (nonlinear in the features)
- ... but may require many more aggregate states to reach the same level of performance as the corresponding linear feature-based architecture

EXAMPLE III: REP. STATES/COARSE GRID

- Choose a collection of "representative" original system states, and associate each one of them with an aggregate state



- Disaggregation probabilities are $d_{xi} = 1$ if i is equal to representative state x.

- Aggregation probabilities associate original system states with convex combinations of representative states

$$j \sim \sum_{y \in \mathcal{A}} \phi_{jy} y$$

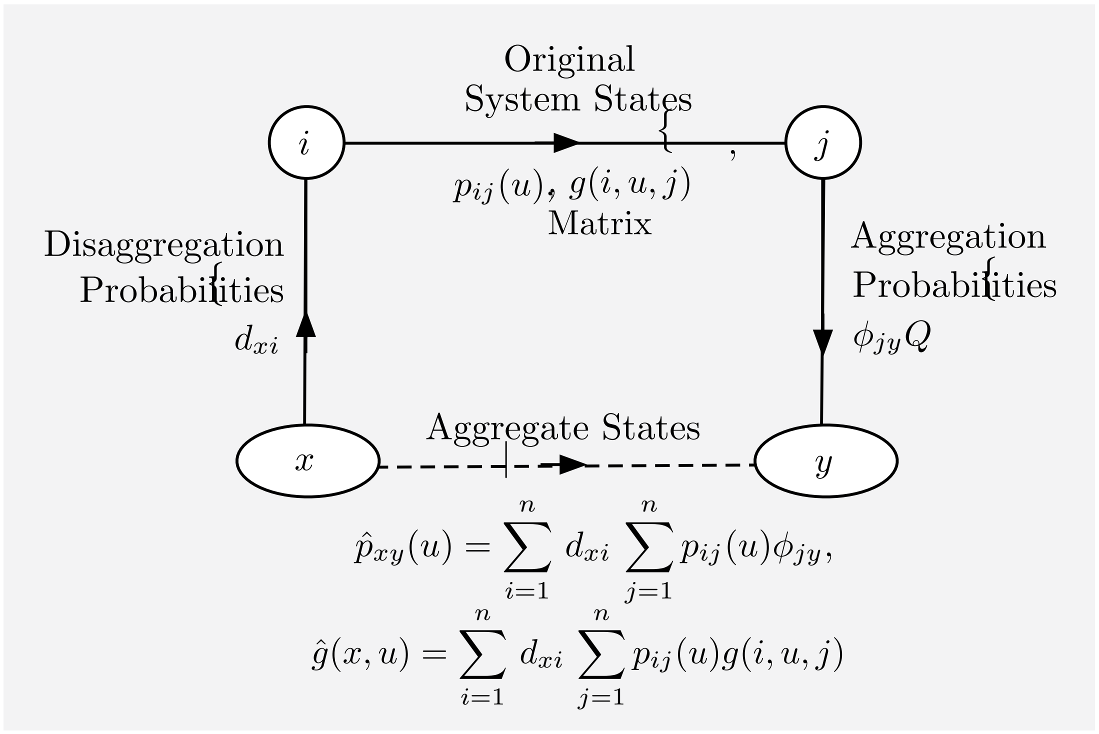
- Well-suited for Euclidean space discretization

- Extends nicely to continuous state space, including belief space of POMDP

EXAMPLE IV: REPRESENTATIVE FEATURES

- Here the aggregate states are nonempty subsets of original system states (but need not form a partition of the state space)
- Example: Choose a collection of distinct "representative" feature vectors, and associate each of them with an aggregate state consisting of original system states with similar features
- Restrictions:
  - The aggregate states/subsets are disjoint.
  - The disaggregation probabilities satisfy $d_{xi} > 0$ if and only if $i \in x$.
  - The aggregation probabilities satisfy $\phi_{jy} = 1$ for all $j \in y$.
- If every original system state i belongs to some aggregate state we obtain hard aggregation
- If every aggregate state consists of a single original system state, we obtain aggregation with representative states
- With the above restrictions $D\Phi = I$, so $(\Phi D)(\Phi D) = \Phi D$, and ΦD is an oblique projection (orthogonal projection in case of hard aggregation)

APPROXIMATE PI BY AGGREGATION



- Consider approximate policy iteration for the original problem, with policy evaluation done by aggregation.

- Evaluation of policy μ: $\tilde{J} = \Phi R$, where $R = DT_\mu(\Phi R)$ (R is the vector of costs of aggregate states for μ). Can be done by simulation.

- Looks like projected equationΦ $R = \Pi T_\mu(\Phi R)$ (but withΦ D in place of Π).

- Advantages: It has no problem with exploration or with oscillations.

- Disadvantage: The rows of D and Φ must be probability distributions.

DISTRIBUTED AGGREGATION I

- We consider decomposition/distributed solution of large-scale discounted DP problems by aggregation.
- Partition the original system states into subsets $S_1, \ldots, S_m$
- Each subset S_ℓ, $\ell = 1, \ldots, m$:
  - Maintains detailed/exact local costs

    $$J(i) \quad \text{for every original system state } i \in S_\ell$$

    using aggregate costs of other subsets
  - Maintains an aggregate cost $R(\ell) = \sum_{i \in S_\ell} d_{\ell i} J(i)$
  - Sends $R(\ell)$ to other aggregate states
- $J(i)$ and $R(\ell)$ are updated by VI according to

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

with R_k being the vector of $R(\ell)$ at time k, and

$$H_\ell(i, u, J, R) = \sum_{j=1}^{n} p_{ij}(u) g(i, u, j) + \alpha \sum_{j \in S_\ell} p_{ij}(u) J(j) + \alpha \sum_{j \in S_{\ell'},\, \ell' \neq \ell} p_{ij}(u) R(\ell')$$

DISTRIBUTED AGGREGATION II

- Can show that this iteration involves a sup-norm contraction mapping of modulus α, so it converges to the unique solution of the system of equations in (J, R)

$$J(i) = \min_{u \in U(i)} H_\ell(i, u, J, R), \quad R(\ell) = \sum_{i \in S_\ell} d_{\ell i} J(i),$$
$$\forall\, i \in S_\ell,\ \ell = 1, \ldots, m.$$

- This follows from the fact that $\{d_{\ell i} \mid i = 1, \ldots, n\}$ is a probability distribution.

- View these equations as a set of Bellman equations for an "aggregate" DP problem. The difference is that the mapping H involves $J(j)$ rather than $R\big(x(j)\big)$ for $j \in S_\ell$.

- In an asynchronous version of the method, the aggregate costs $R(\ell)$ may be outdated to account for communication "delays" between aggregate states.

- Convergence can be shown using the general theory of asynchronous distributed computation (see the text).