# 6.231 DYNAMIC PROGRAMMING

## LECTURE 5

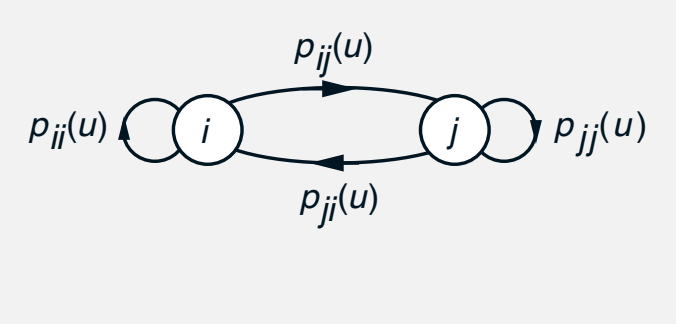
## LECTURE OUTLINE

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

# REVIEW

# 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 $i$:

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

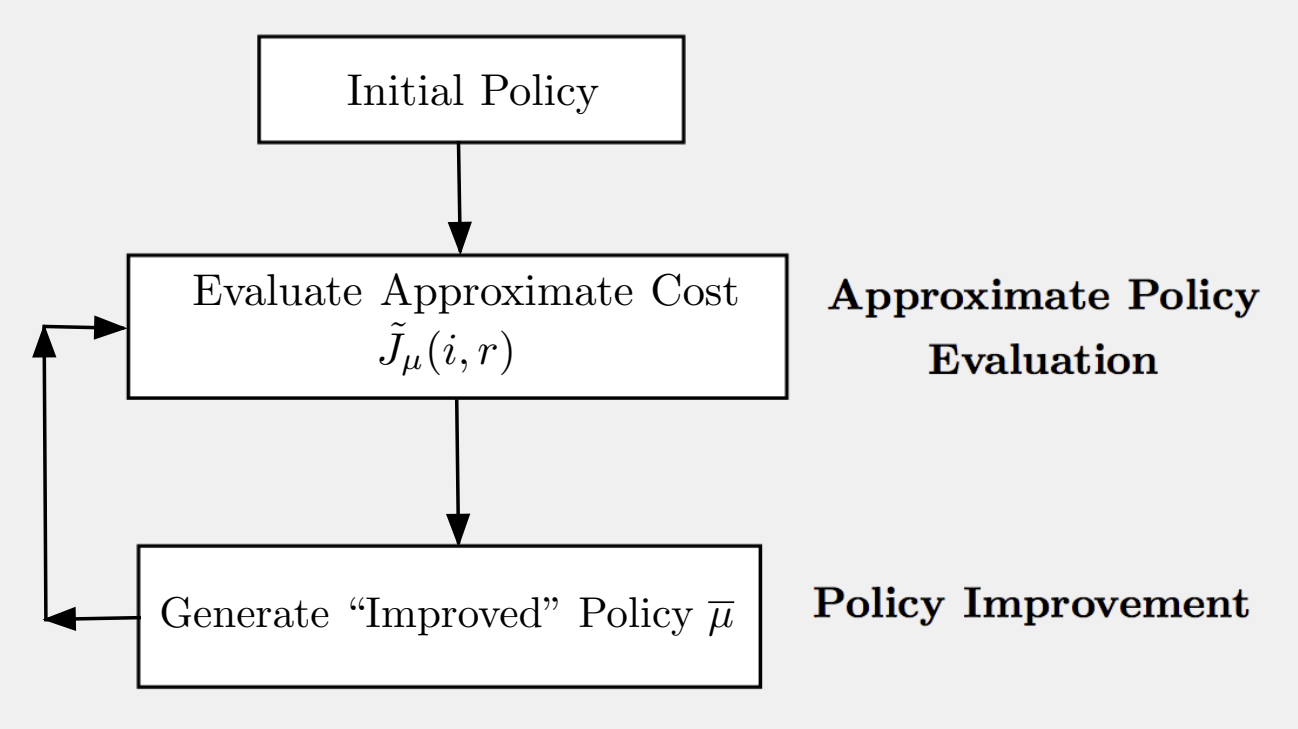
with $\alpha \in [0, 1)$

- 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




- Evaluation of typical policy $\mu$: Linear cost function approximation

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

where $\Phi$ is full rank $n \times s$ matrix with columns the basis functions, and $i$th 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

- Approximate policy evaluation by solving

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

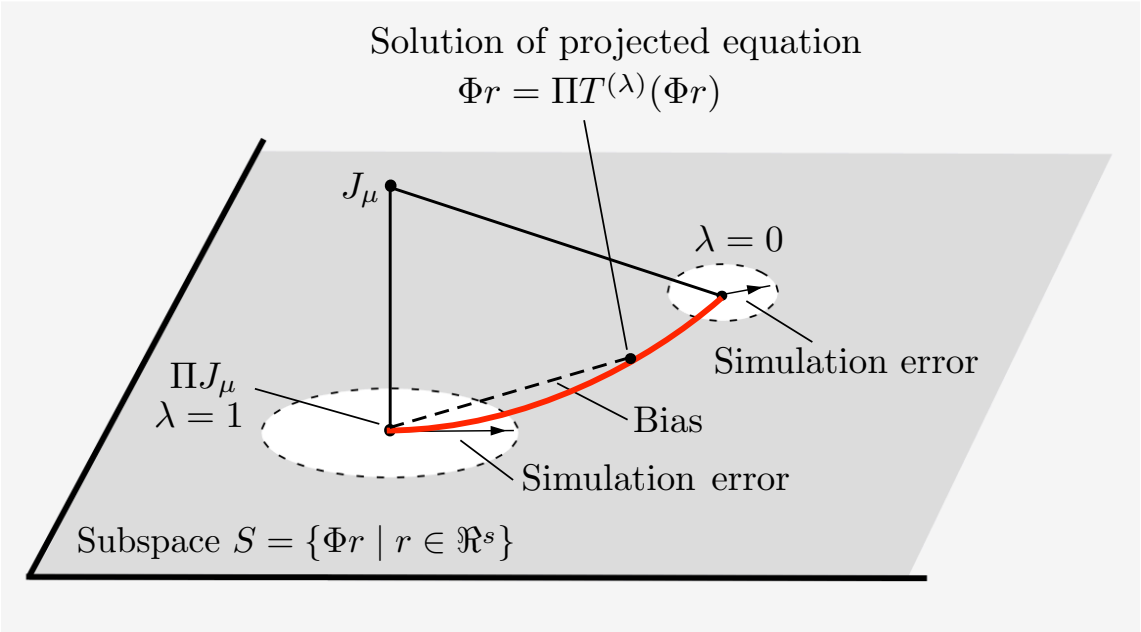
$\Pi$: weighted Euclidean projection; special nature of the steady-state distribution weighting.

- Implementation by simulation (single long trajectory using current policy - important to make $\Pi T_\mu$ a contraction). LSTD, LSPE methods.

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

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

  - As $\lambda \uparrow 1$, $\Pi T^{(\lambda)}$ becomes a contraction for any projection norm
  - Bias-variance tradeoff



# ISSUES OF POLICY IMPROVEMENT

# EXPLORATION

- **1st major issue:** exploration. To evaluate $\mu$, we need to generate cost samples using $\mu$

- This biases the simulation by underrepresenting states that are unlikely to occur under $\mu$.

- As a result, the cost-to-go estimates of these underrepresented states may be highly inaccurate, and seriously impact 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).

- To deal with this we must change the sampling mechanism and modify the simulation formulas.

- Solve

$$\Phi r = \overline{\Pi} T_\mu(\Phi r)$$

where $\overline{\Pi}$ is projection with respect to an exploration-enhanced norm [uses a weight distribution $\zeta = (\zeta_1, \ldots, \zeta_n)$].

- $\zeta$ is more "balanced" than $\xi$ the steady-state distribution of the Markov chain of $\mu$.

- This also addresses any lack of ergodicity of $\mu$.

# EXPLORATION MECHANISMS

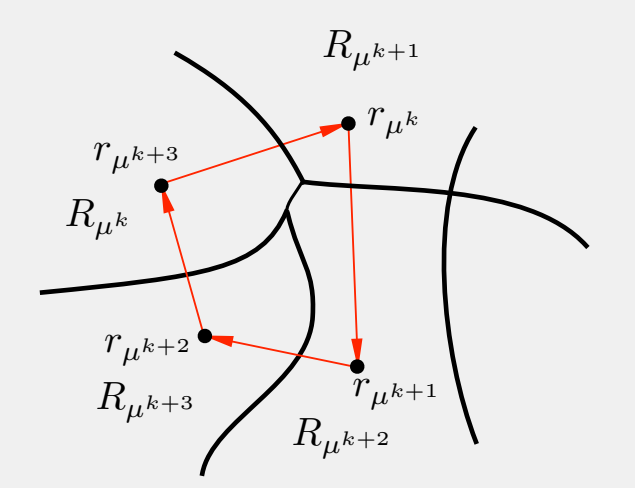
- One possibility: Use multiple short simulation trajectories instead of single long trajectory starting from a rich mixture of states. This is known as geometric sampling, or free-form sampling.
  - By properly choosing the starting states, we enhance exploration
  - The simulation formulas for LSTD($\lambda$) and LSPE($\lambda$) have to be modified to yield the solution of $\Phi r = \overline{\Pi} T(\Phi r)$ (see the DP text)
- Another possibility: Use a modified policy to generate a single long trajectory. This is called an off-policy approach.
  - Modify the transition probabilities of $\mu$ to enhance exploration
  - Again the simulation formulas for LSTD($\lambda$) and LSPE($\lambda$) have to be modified to yield the solution of $\Phi r = \overline{\Pi} T(\Phi r)$ (use of importance sampling; see the DP text)
- With larger values of $\lambda > 0$ the contraction property of $\overline{\Pi} T_\mu^{(\lambda)}$ is maintained.
- LSTD may be used without $\overline{\Pi} T_\mu^{(\lambda)}$ being a contraction ...

# POLICY ITERATION ISSUES: OSCILLATIONS

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

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

- There is a finite number of possible vectors $r_\mu$, one generated from another in a deterministic way



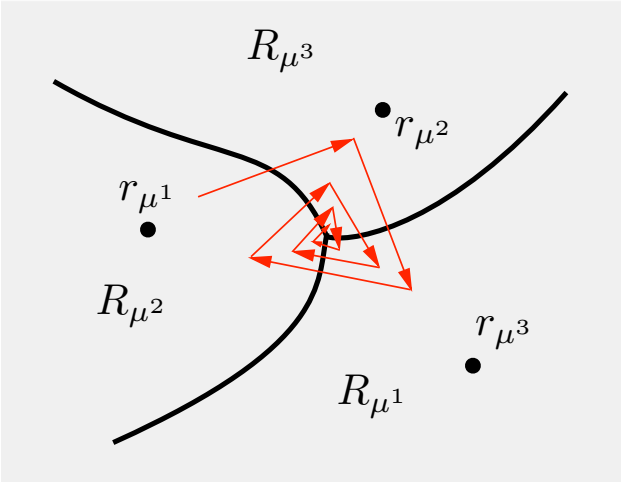

- The algorithm ends up repeating some cycle of policies $\mu^k, \mu^{k+1}, \ldots, \mu^{k+m}$ with

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

- 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 an evaluation of the form

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

with $W$: monotone and $WT_\mu$: contraction, the policies converge (to a possibly nonoptimal limit).

- The operator $W$ in the aggregation approach has the monotonicity and contraction properties.

# AGGREGATION

# PROBLEM APPROXIMATION - AGGREGATION

- Another major idea in ADP is to approximate the cost-to-go function of the given problem with the cost-to-go function of a simpler problem.

- 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)

- If $\hat{R}(y)$ is the optimal cost of aggregate state $y$, we use the approximation

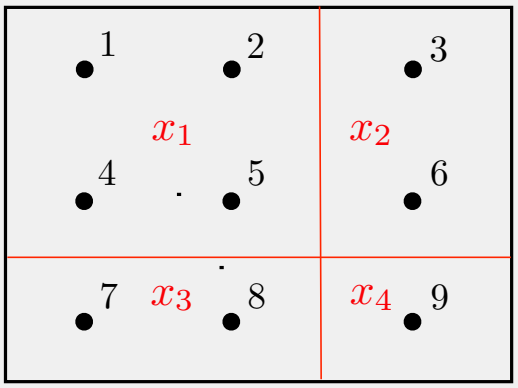
$$J^*(j) \approx \sum_y \phi_{jy} \hat{R}(y), \qquad \forall\ j$$

where $\phi_{jy}$ are the aggregation probabilities, encoding the "degree of membership of $j$ in the aggregate state $y$"

- This is a linear architecture: $\phi_{jy}$ are the features of state $j$

# HARD AGGREGATION EXAMPLE

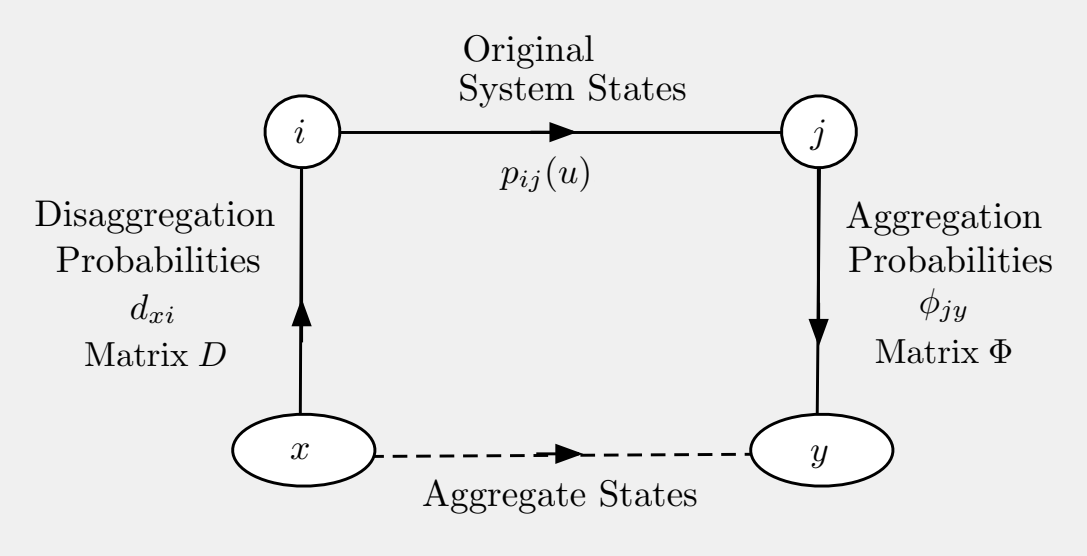
- Group the original system states into subsets, and view each subset as an aggregate state
- Aggregation probs.: $\phi_{jy} = 1$ if $j$ belongs to aggregate state $y$ (piecewise constant approx).




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

- What should be the transition probs. out of $x$? Select $i \in x$ and use the transition probs. of $i$.
- Suppose I am at aggregate state $x$, what does this tell me about which of the states $i \in x$ I am?
- The simplest possibility is to assume that all states in $x$ are equally likely.
- A generalization is to use the disaggregation probabilities $d_{xi}$: Roughly, the "degree to which $i$ is representative of $x$."

# AGGREGATION/DISAGGREGATION PROBS




- Define the aggregate system transition probabilities via two (somewhat arbitrary) choices.

- For each original system state $j$ and aggregate state $y$, the aggregation probability $\phi_{jy}$
  - 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 $x$."

- Aggregation scheme is defined by the two matrices $D$ and $\Phi$. The rows of $D$ and $\Phi$ must be probability distributions.

# AGGREGATE SYSTEM DESCRIPTION

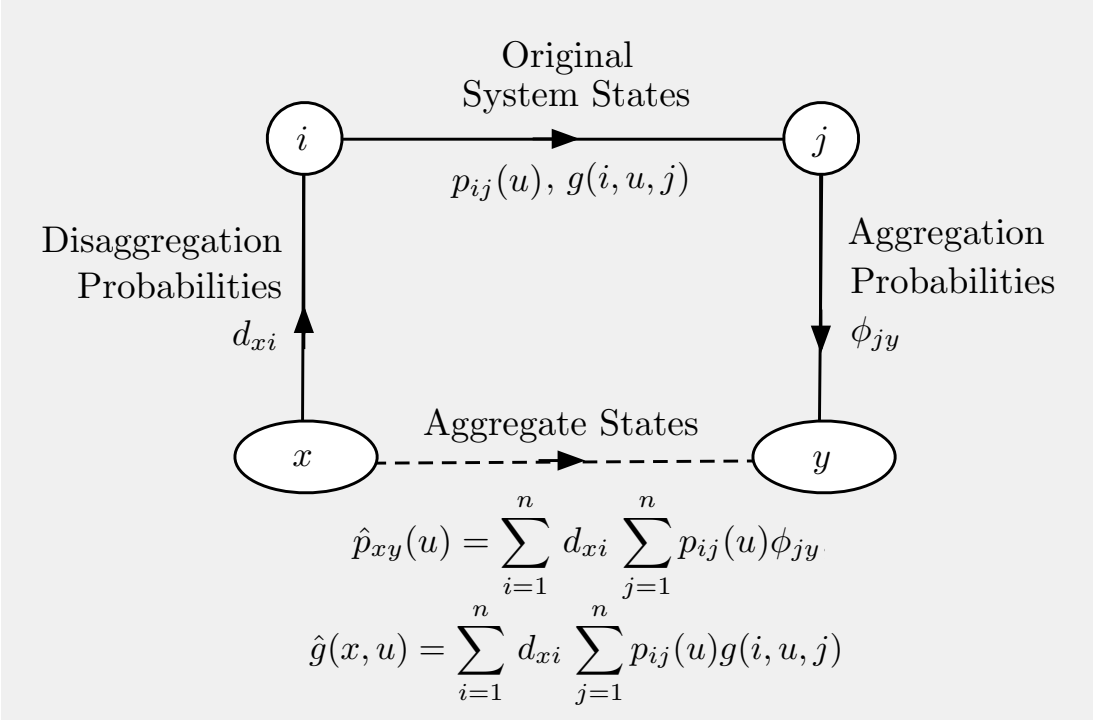
- The transition probability from aggregate state $x$ to aggregate state $y$ under control $u$

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

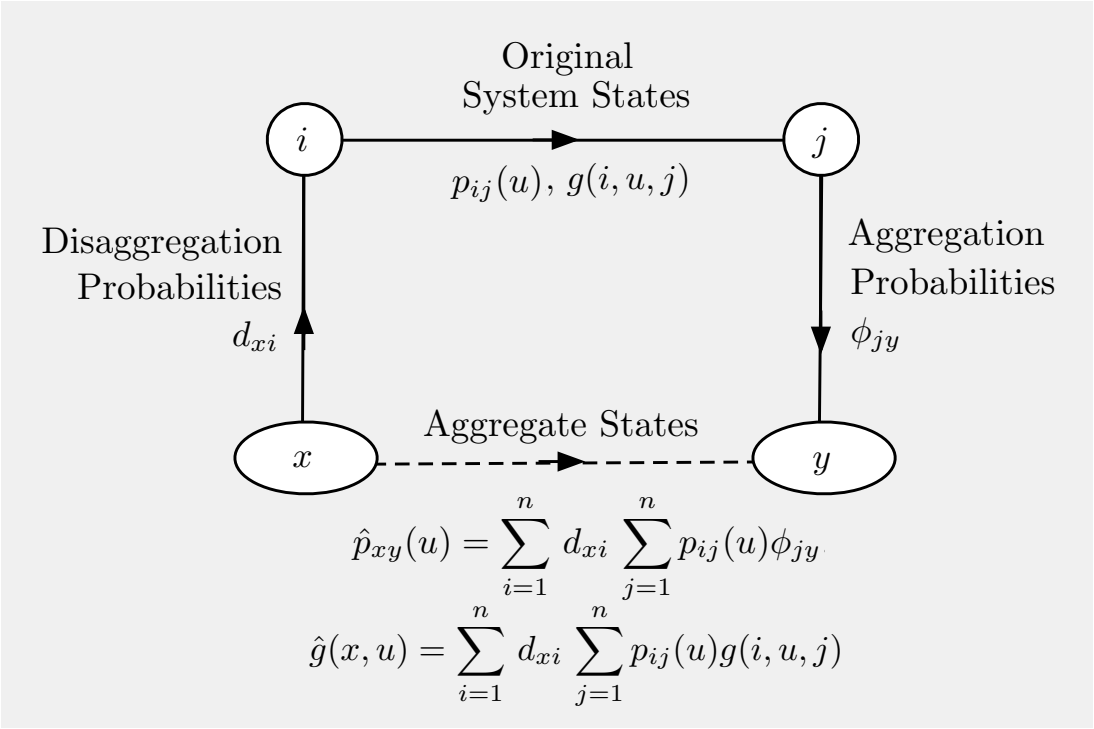
where the rows of $D$ and $\Phi$ are the disaggregation and aggregation probs.

- The expected transition cost is

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



# AGGREGATE BELLMAN'S EQUATION




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

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

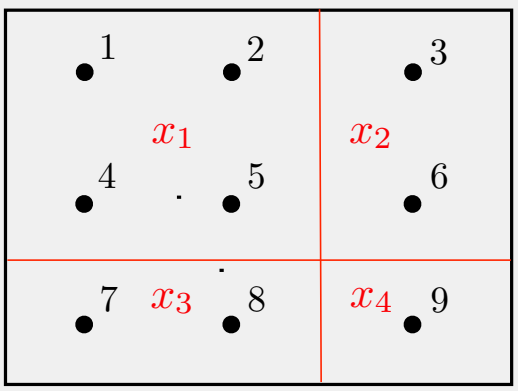
Bellman's equation for the aggregate problem.

• 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 $y$.

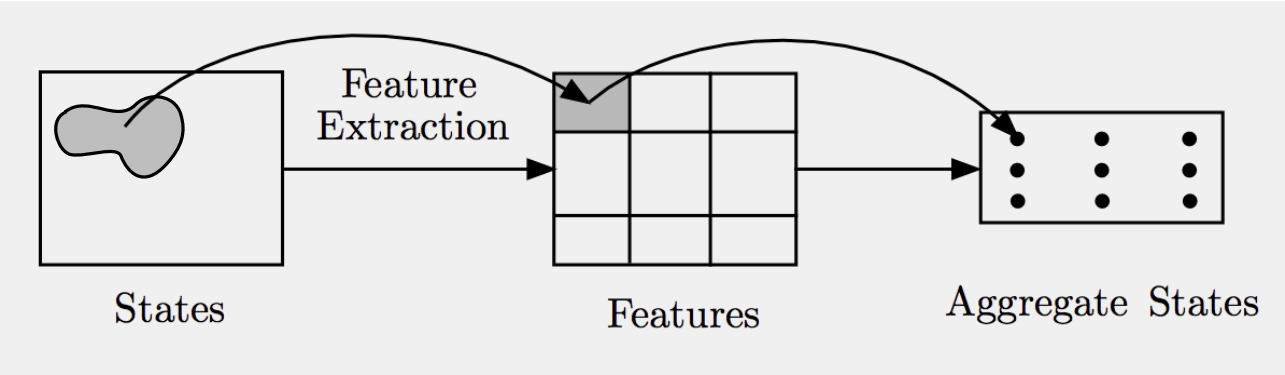



$$\Phi = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- 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

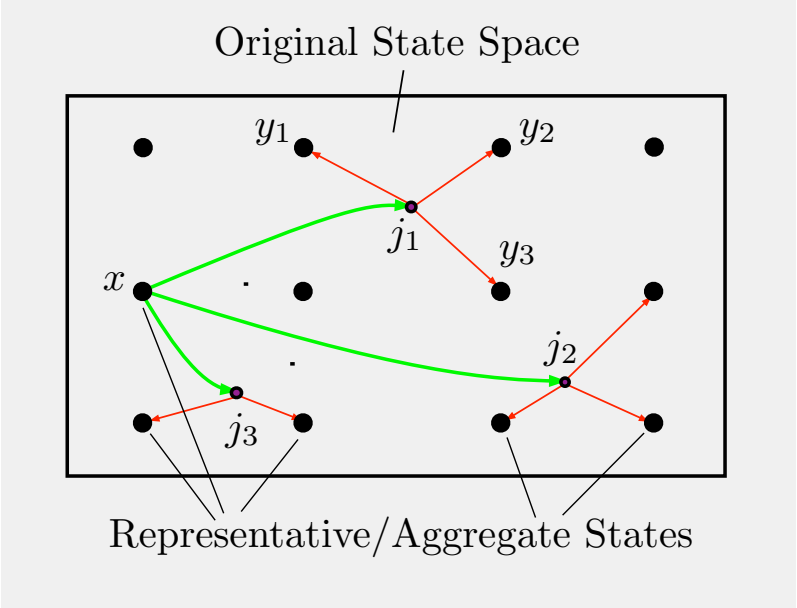
- Important question: How do we group states together?
- If we know good features, it makes sense to group together states that have "similar features"




- 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

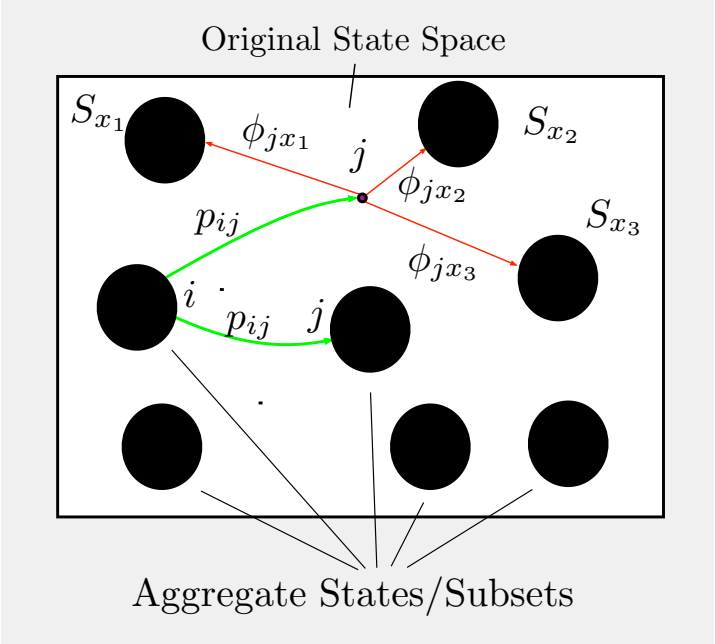
$$j \sim \sum_{y \in \mathcal{A}} \phi_{jy} y$$

- 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. Common case: Each $S_x$ is a group of states with "similar features"

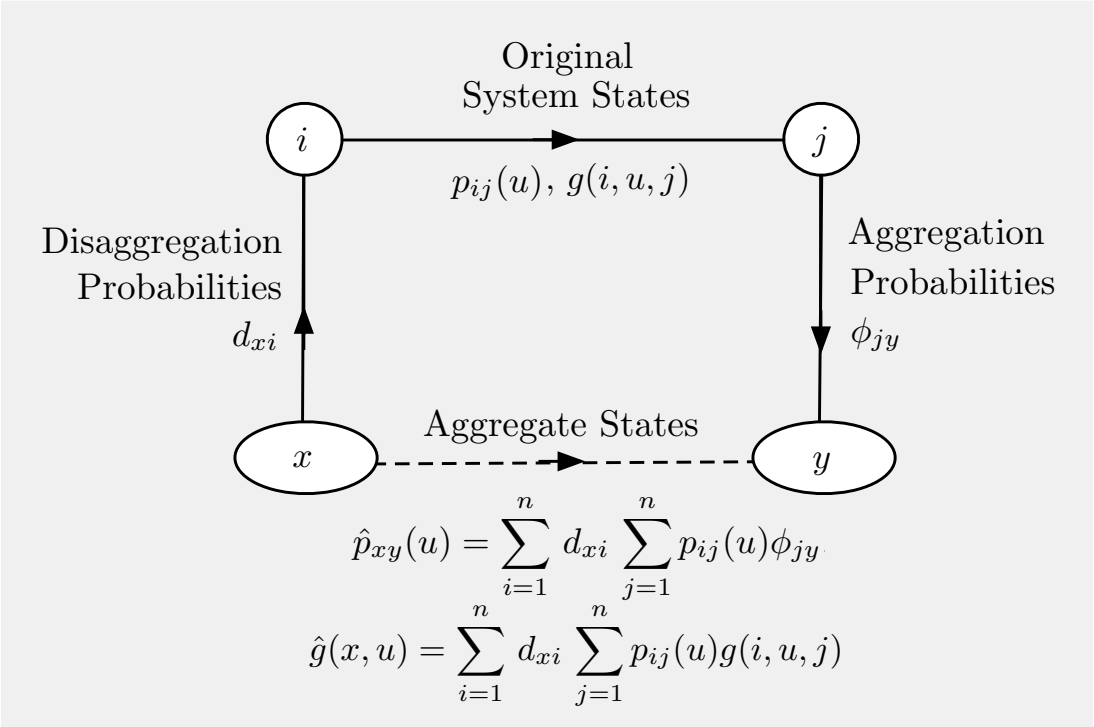



- Restrictions:
  - The aggregate states/subsets are disjoint.
  - The disaggregation probabilities satisfy $d_{xi} > 0$ if and only if $i \in x$.
  - The aggregation probabilities satisfy $\phi_{jy} = 1$ for all $j \in y$.

- Hard aggregation is a special case: $\cup_x S_x = \{1, \ldots, n\}$

- Aggregation with representative states is a special case: $S_x$ consists of just one state

# APPROXIMATE PI BY AGGREGATION




- Consider approximate PI for the original problem, with policy evaluation done by aggregation.
- Evaluation of policy $\mu$: $\tilde{J} = \Phi R$, where $R = DT_\mu(\Phi R)$ ($R$ is the vector of costs of aggregate states for $\mu$). Can be done by simulation.
- Looks like projected equation $\Phi R = \Pi T_\mu(\Phi R)$ (but with $\Phi D$ in place of $\Pi$).
- Advantages: It has no problem with oscillations.
- Disadvantage: The rows of $D$ and $\Phi$ must be probability distributions.

# ADDITIONAL ISSUES OF AGGREGATION

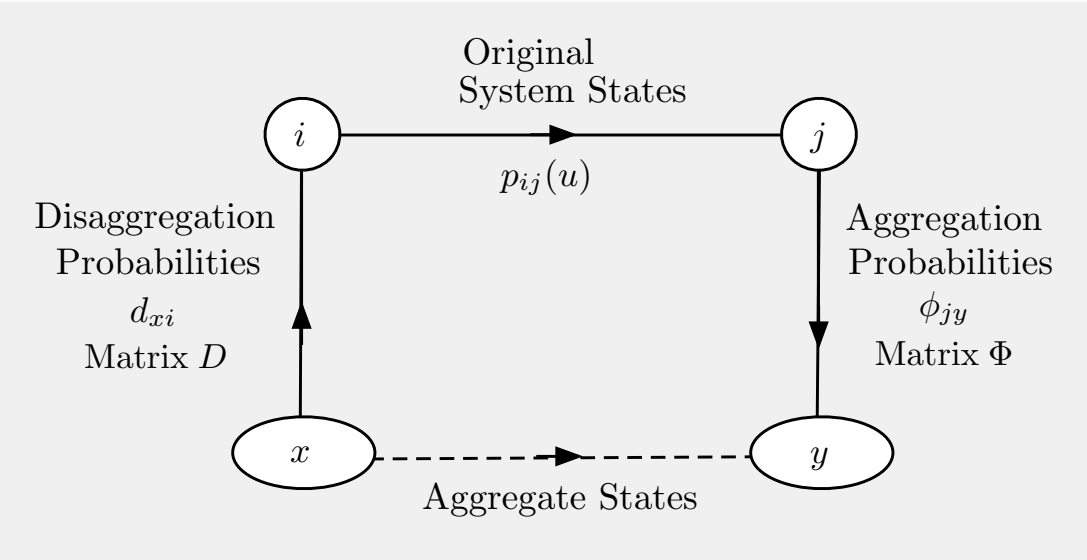
# ALTERNATIVE POLICY ITERATION

- The preceding PI method uses policies that assign a control to each aggregate state.
- An alternative is to use PI for the combined system, involving the Bellman equations:

$$R^*(x) = \sum_{i=1}^{n} d_{xi}\tilde{J}_0(i), \qquad x \in \mathcal{A},$$

$$\tilde{J}_0(i) = \min_{u\in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j)+\alpha\tilde{J}_1(j)\big), \quad i = 1,\ldots,n,$$

$$\tilde{J}_1(j) = \sum_{y\in\mathcal{A}} \phi_{jy}R^*(y), \qquad j = 1,\ldots,n.$$




- Simulation-based PI and VI are still possible.

# RELATION OF AGGREGATION/PROJECTION

- Compare aggregation and projected equations

$$\Phi R = \Phi D T(\Phi R), \qquad \Phi r = \Pi T(\Phi r)$$

- If $\Phi D$ is a projection (with respect to some weighted Euclidean norm), then the methodology of projected equations applies to aggregation

- Hard aggregation case: $\Phi D$ can be verified to be projection with respect to weights $\xi_i$ proportional to the disaggregation probabilities $d_{xi}$

- Aggregation with representative features case: $\Phi D$ can be verified to be a semi-norm projection with respect to weights $\xi_i$ proportional to $d_{xi}$

- A (weighted) Euclidean semi-norm is defined by $\|J\|_\xi = \sqrt{\sum_{i=1}^n \xi_i \big(J(i)\big)^2}$, where $\xi = (\xi_1, \ldots, \xi_n)$, with $\xi_i \geq 0$.

- If $\Phi' \Xi \Phi$ is invertible, the entire theory and algorithms of projected equations generalizes to semi-norm projected equations [including multistep methods such as LSTD/LSPE/TD($\lambda$)].

- Reference: Yu and Bertsekas, "Weighted Bellman Equations and their Applications in Approximate Dynamic Programming," MIT Report, 2012.

# DISTRIBUTED AGGREGATION I

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

    $J(i)$ 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 $\alpha$, 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).