Topics in Reinforcement Learning: AlphaZero, ChatGPT, Neuro-Dynamic Programming, Model Predictive Control, Discrete Optimization, Applications

Arizona State University
Course CSE 691, Spring 2025

Links to Class Notes, Videolectures, and Slides at
http://web.mit.edu/dimitrib/www/RLbook.html[1]

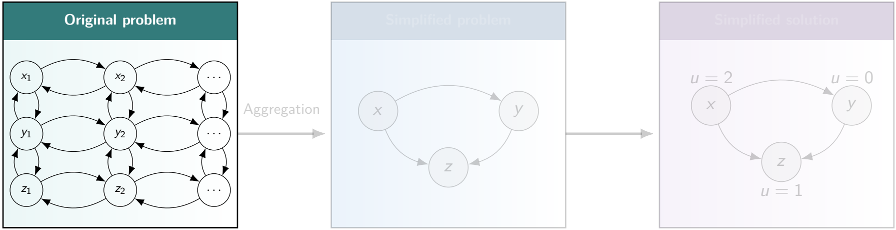
Dr. Kim Hammar (khammar1@asu.edu), Prof. Dimitri P. Bertsekas (dimitrib@mit.edu), and Dr. Yuchao Li (yuchaoli@asu.edu)

Guest Lecture

Approximation in Value Space using Aggregation, with Applications to POMDPs and Cybersecurity

[1] Dimitri P. Bertsekas. *A Course in Reinforcement Learning*. 2nd edition. Athena Scientific, 2025.

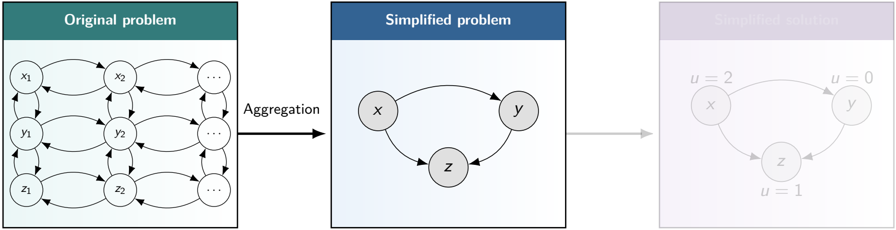
Aggregation is A Form of **Problem Simplification**



The Aggregation Methodology

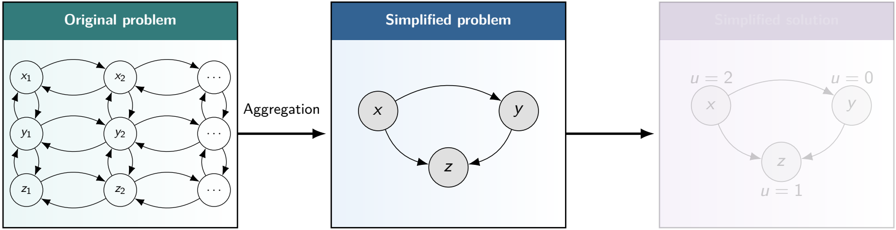
1. Combine groups of similar states into aggregate states.
2. Formulate an aggregate dynamic programming problem based on these states.
3. Solve the aggregate problem using some computational method.
4. Use the solution to the aggregate problem to compute a cost function approximation for the original problem.

Aggregation is A Form of **Problem Simplification**



The Aggregation Methodology

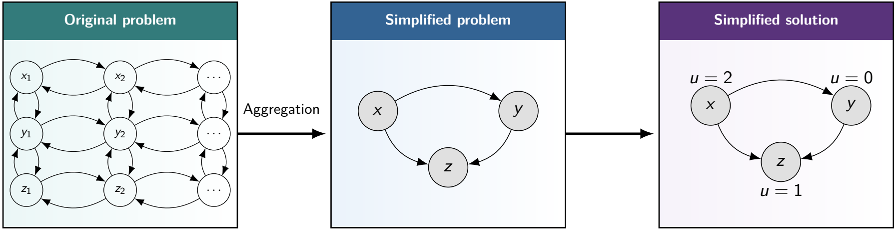
1. Combine groups of similar states into aggregate states.
2. Formulate an aggregate dynamic programming problem based on these states.
3. Solve the aggregate problem using some computational method.
4. Use the solution to the aggregate problem to compute a cost function approximation for the original problem.



The Aggregation Methodology

1. Combine groups of similar states into aggregate states.
2. Formulate an aggregate dynamic programming problem based on these states.
3. Solve the aggregate problem using some computational method.
4. Use the solution to the aggregate problem to compute a cost function approximation for the original problem.

Aggregation is A Form of **Problem Simplification**



The Aggregation Methodology

1. Combine groups of similar states into aggregate states.
2. Formulate an aggregate dynamic programming problem based on these states.
3. Solve the aggregate problem using some computational method.
4. Use the solution to the aggregate problem to compute a cost function approximation for the original problem.

Outline

1. Aggregation with Representative States
2. **Example:** Aggregation with Representative States for POMDPs
3. General Aggregation Methodology
4. **Case study**: Aggregation for Cybersecurity

Outline

1. Aggregation with Representative States
2. **Example:** Aggregation with Representative States for POMDPs
3. General Aggregation Methodology
4. **Case study**: Aggregation for Cybersecurity

Outline

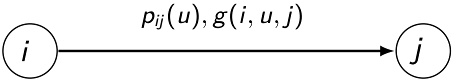
1. Aggregation with Representative States
2. **Example:** Aggregation with Representative States for POMDPs
3. General Aggregation Methodology
4. **Case study**: Aggregation for Cybersecurity

Outline

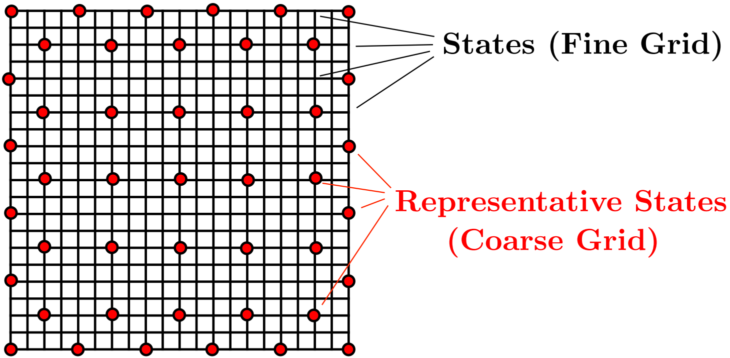
1. Aggregation with Representative States
2. **Example:** Aggregation with Representative States for POMDPs
3. General Aggregation Methodology
4. **Case study**: Aggregation for Cybersecurity

Reminder on Notation

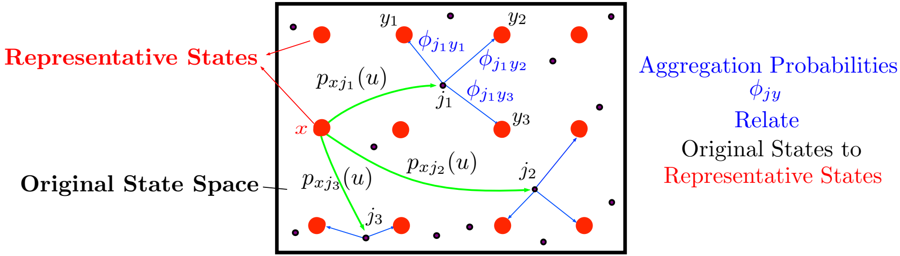
- State space: $X = \{1, \ldots, n\}$, states are denoted by i, j.
- Control constraint set: $U(i)$.
- Cost of transitioning from state i to j given control u: $g(i, u, j)$.
- Cost-to-go from state i: $J(i)$.
- Discount factor: α.
- Probability of transitioning from state i to j given control u: $p_{ij}(u)$.
  - Equivalent formulation: $x_{k+1} = f(x_k, u_k, w_k)$.

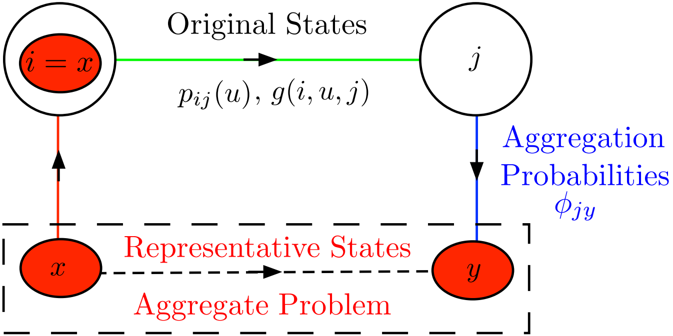


- Introduce a **subset** $\mathcal{A}$ of the original states $1, \ldots, n$, called representative states.
- We use i, j to denote original states and x, y to denote representative states.



- For each state i we define aggregation probabilities $\{\phi_{ix} \mid x \in \mathcal{A}\}$.
- Intuitively, ϕ_{ix} expresses similarity between states i and x, where $\phi_{xx} = 1$.




Original States
$i = x$
j
$p_{ij}(u),\ g(i,u,j)$
Aggregation
Probabilities
ϕ_{jy}
Representative States
x
y
Aggregate Problem

- State space: $\mathcal{A}$ (the set of representative states).
- Control constraint set: $U(i)$ (the original control constraint set).
- Transition probabilities and costs

$$\hat{p}_{xy}(u) = \sum_{i=1}^{n} p_{xi}(u)\phi_{ji}, \qquad \text{for all representative states } (x, y) \text{ and controls } u,$$

$$\hat{g}(x, u) = \sum_{i=1}^{n} p_{xi}(u) g(x, u, i), \qquad \text{for all representative states } x \text{ and controls } u.$$

Formulating the Aggregate Dynamic Programming Problem

- State space: $\mathcal{A}$ (the set of representative states).
- Control constraint set: $U(i)$ (the original control constraint set).
- Transition probabilities and costs

$$\hat{p}_{xy}(u) = \sum_{i=1}^{n} p_{xi}(u)\phi_{ji}, \qquad \text{for all representative states } (x, y) \text{ and controls } u,$$

$$\hat{g}(x, u) = \sum_{i=1}^{n} p_{xi}(u) g(x, u, i), \qquad \text{for all representative states } x \text{ and controls } u.$$

Formulating the Aggregate Dynamic Programming Problem

- State space: $\mathcal{A}$ (the set of representative states).
- Control constraint set: $U(i)$ (the original control constraint set).
- Transition probabilities and costs

$$\hat{p}_{xy}(u) = \sum_{i=1}^{n} p_{xi}(u)\phi_{ji}, \qquad \text{for all representative states } (x, y) \text{ and controls } u,$$

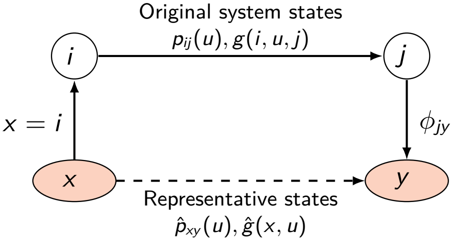
$$\hat{g}(x, u) = \sum_{i=1}^{n} p_{xi}(u) g(x, u, i), \qquad \text{for all representative states } x \text{ and controls } u.$$

Formulating the Aggregate Dynamic Programming Problem

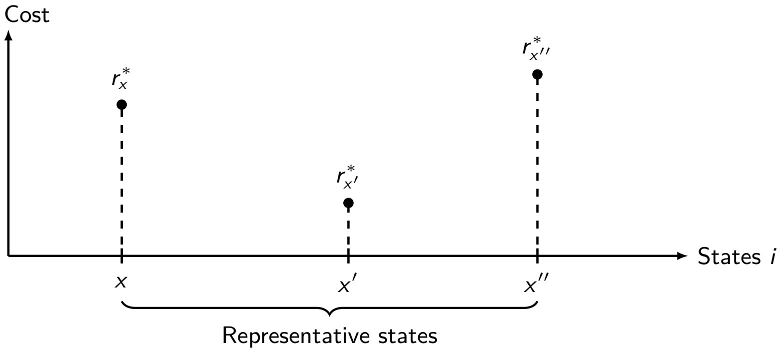
- State space: $\mathcal{A}$ (the set of representative states).
- Control constraint set: $U(i)$ (the original control constraint set).
- Transition probabilities and costs

$$\hat{p}_{xy}(u) = \sum_{i=1}^{n} p_{xi}(u)\phi_{ji}, \qquad \text{for all representative states } (x, y) \text{ and controls } u,$$

$$\hat{g}(x, u) = \sum_{i=1}^{n} p_{xi}(u) g(x, u, i), \qquad \text{for all representative states } x \text{ and controls } u.$$

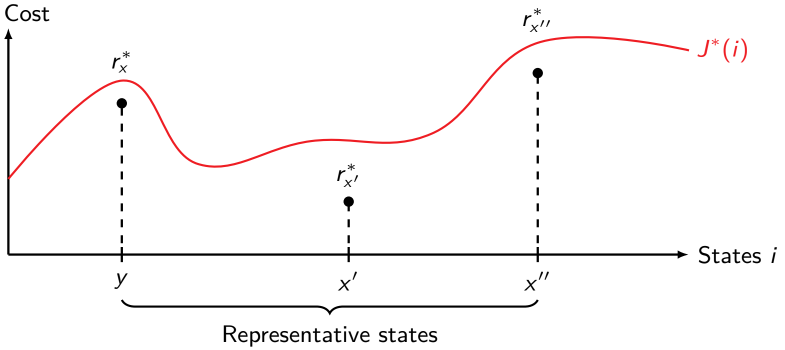


- The aggregate problem can be solved “exactly” using dynamic programming/simulation.
- The optimal cost from a representative state x in this problem is denoted by r_x^*.



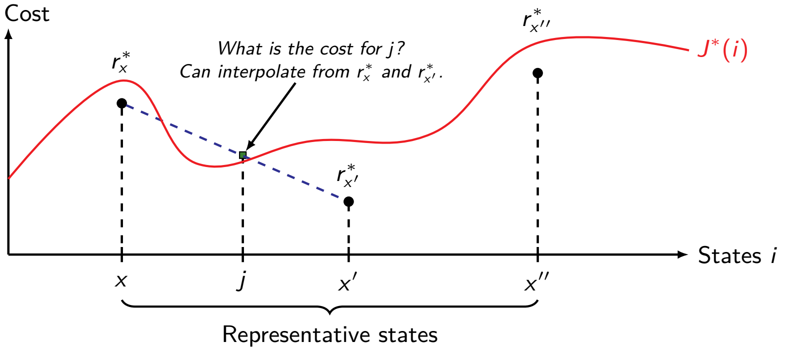
Cost Difference Between the Aggregate and Original Problems

- The aggregate cost function r^*_x is only defined for representative states $x \in \mathcal{A}$.
- The optimal cost function $J^*(i)$ is defined for the entire state space $i = 1, \ldots, n$.
- For a representative state x, we generally have $r^*_x \neq J^*(x)$.



Cost Difference Between the Aggregate and Original Problems

- The aggregate cost function r_x^* is only defined for representative states $x \in \mathcal{A}$.
- The optimal cost function $J^*(i)$ is defined for the entire state space $i = 1, \ldots, n$.
- For a representative state x, we generally have $r_x^* \neq J^*(x)$.



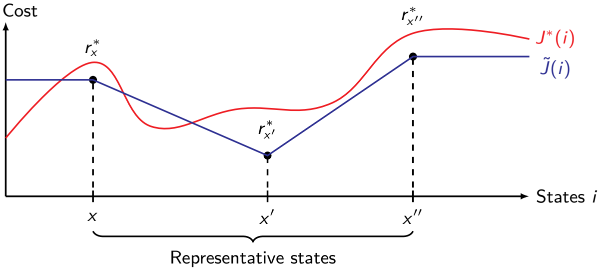
Using the Aggregate Solution to Approximate the Original Problem

- We obtain an approximate cost function $\tilde{J}$ for the original problem via **interpolation**:

$$\tilde{J}(i) = \sum_{x \in \mathcal{A}} \phi_{ix} r_x^*, \qquad i = 1, \ldots, n.$$

- Using this cost function, we can obtain a one-step lookahead policy:

$$\mu(i) \in \arg\min_{u \in U(i)} \left\{ \sum_{j=1}^{n} p_{ij}(u) \left(g(i,u,j) + \alpha \tilde{J}(j) \right) \right\}, \qquad i = 1, \ldots, n.$$



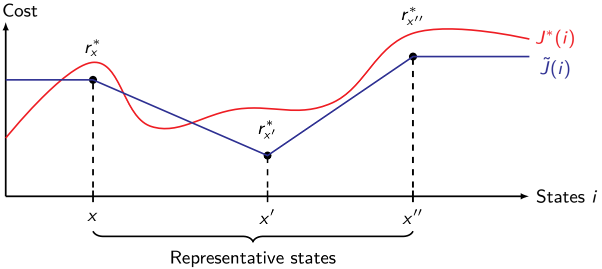
Using the Aggregate Solution to Approximate the Original Problem

- We obtain an approximate cost function $\tilde{J}$ for the original problem via **interpolation**:

$$\tilde{J}(i) = \sum_{x \in \mathcal{A}} \phi_{ix} r_x^*, \qquad i = 1, \dots, n.$$

- Using this cost function, we can obtain a one-step lookahead policy:

$$\mu(i) \in \arg\min_{u \in U(i)} \left\{ \sum_{j=1}^{n} p_{ij}(u) \left(g(i,u,j) + \alpha \tilde{J}(j) \right) \right\}, \qquad i = 1, \dots, n.$$



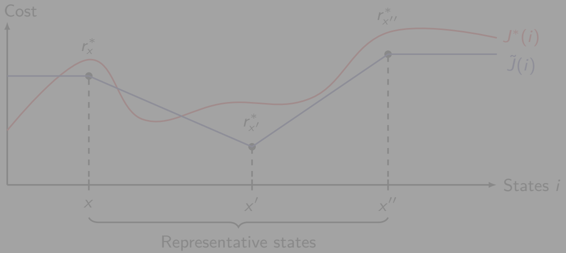
Approximating the Original Problem

- We obtain an approximate cost function $\tilde{J}$ for the original problem via **interpolation**:

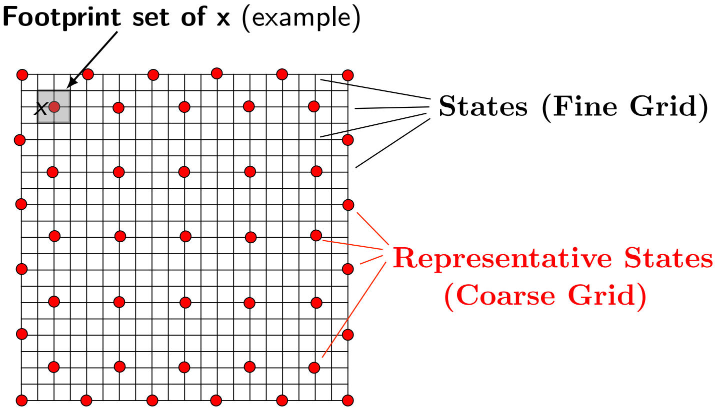
$$\tilde{J}(j) = \sum_{y \in \mathcal{A}} \phi_{jy} r_y^*, \qquad j = 1, \ldots, n.$$

- Using this cost function, we can obtain a one-step lookahead policy:

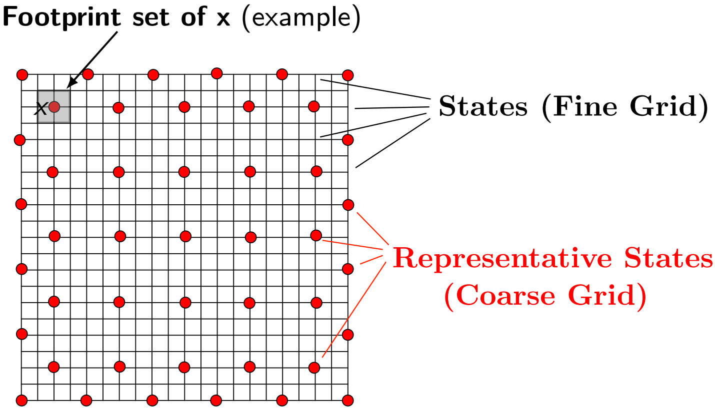
What is the difference between the approximation $\tilde{J}$ and the optimal cost function J^*?



- Consider the case where $\phi_{jy} = 0$ for all representative states x except one.
- Let S_x denote the set of states that aggregate to the representative state x.
  - i.e., the *footprint* of x, where $\{1, \ldots, n\} = \bigcup_{x \in \mathcal{A}} S_x$.



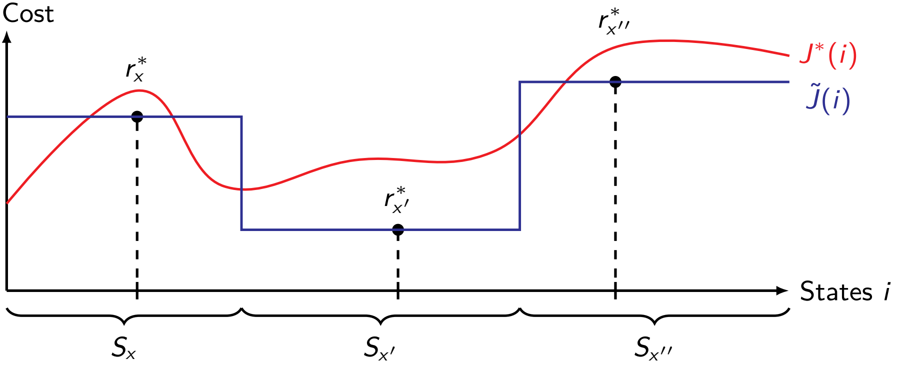
- Consider the case where $\phi_{jy} = 0$ for all representative states x except one.
- Let S_x denote the set of states that aggregate to the representative state x.
  - i.e., the *footprint* of x, where $\{1, \dots, n\} = \bigcup_{x \in \mathcal{A}} S_x$.



- In the case of hard aggregation, $\tilde{J}(i) = \sum_{x \in \mathcal{A}} \phi_{ix} r_x^* = r_y^*$ for all $i \in S_y$.
- Hence, $\tilde{J}$ is piecewise constant.

- In the case of hard aggregation, $\tilde{J}(i) = \sum_{x \in \mathcal{A}} \phi_{ix} r_x^* = r_y^*$ for all $i \in S_y$.
- Hence, $\tilde{J}$ is piecewise constant.

- In the case of hard aggregation, $\tilde{J}(i) = \sum_{x \in \mathcal{A}} \phi_{ix} r_x^* = r_y^*$ for all $i \in S_y$.
- Hence, $\tilde{J}$ is piecewise constant.



Approximation Error Bound in the Case of Hard Aggregation

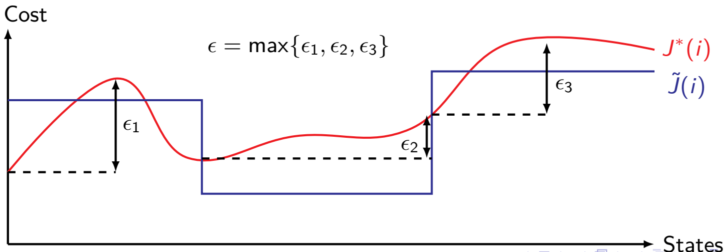
- Let ϵ be the maximum variation of J^* within a footprint set S_x, i.e.,

$$\epsilon = \max_{x \in \mathcal{A}} \max_{i,j \in S_x} |J^*(i) - J^*(j)|.$$

- We refer to the difference $|J^*(i) - \tilde{J}(i)|$ as the *approximation error*.
- This error is bounded as

$$|J^*(i) - \tilde{J}(i)| \leq \frac{\epsilon}{1-\alpha} \qquad i = 1, \ldots, n.$$

- **Takeway**: choose the footprint sets so that ϵ is small.



Approximation Error Bound in the Case of Hard Aggregation

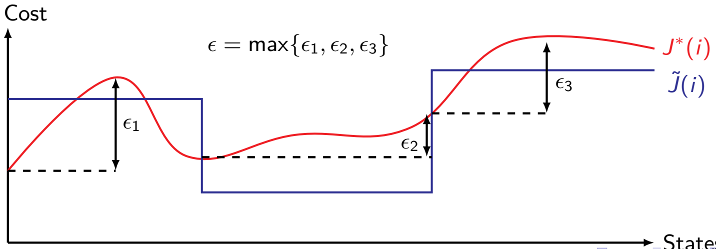
- Let ϵ be the maximum variation of J^* within a footprint set S_x, i.e.,

$$\epsilon = \max_{x \in \mathcal{A}} \max_{i,j \in S_x} |J^*(i) - J^*(j)|.$$

- We refer to the difference $|J^*(i) - \tilde{J}(i)|$ as the *approximation error*.
- This error is bounded as

$$|J^*(i) - \tilde{J}(i)| \leq \frac{\epsilon}{1-\alpha} \qquad i = 1, \dots, n.$$

- **Takeway**: choose the footprint sets so that ϵ is small.

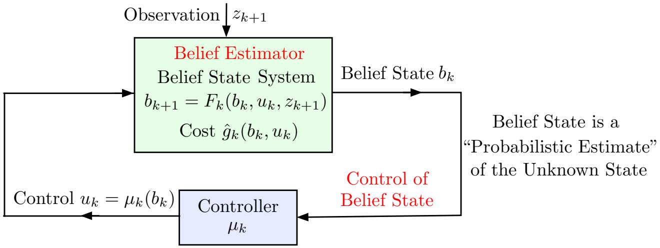


Outline

1. Aggregation with Representative States
2. **Example: Aggregation with Representative States for POMDPs**
3. General Aggregation Methodology
4. **Case study:** Aggregation for Cybersecurity

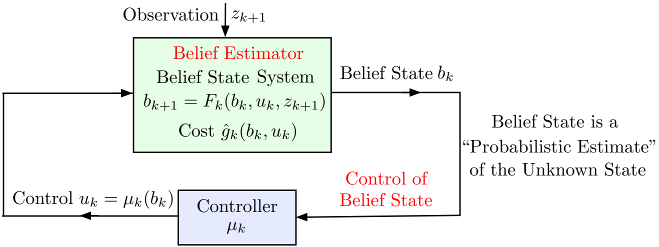
Recap of Partially Observed Markov Decision Problems (POMDPs)

- State space $X = \{1, \dots, n\}$, observation space Z, and control constraint set $U(i)$.
- Each state transition (i, j) generates a cost $g(i, u, j)$;
- and an observation z with probability $p(z \mid j, u)$.
- Let $b(i)$ denote the conditional probability that the state is i, given the history.
- The belief state is defined as $b = \big(b(1), b(2), \dots, b(n)\big)$.
- The belief b is updated using a **belief estimator** $F(b, u, z)$.
- Goal: Find a policy as a function of b that minimizes the cost.



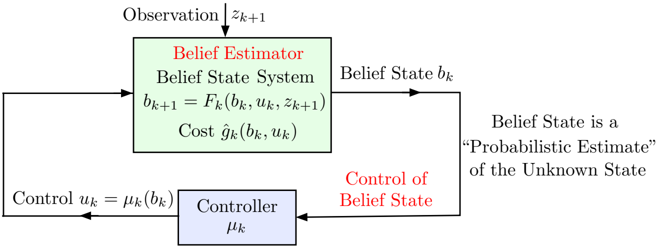
Recap of Partially Observed Markov Decision Problems (POMDPs)

- State space $X = \{1, \dots, n\}$, observation space Z, and control constraint set $U(i)$.
- Each state transition (i, j) generates a cost $g(i, u, j)$;
- and an observation z with probability $p(z \mid j, u)$.
- Let $b(i)$ denote the conditional probability that the state is i, given the history.
- The belief state is defined as $b = \big(b(1), b(2), \dots, b(n)\big)$.
- The belief b is updated using a **belief estimator** $F(b, u, z)$.
- Goal: Find a policy as a function of b that minimizes the cost.



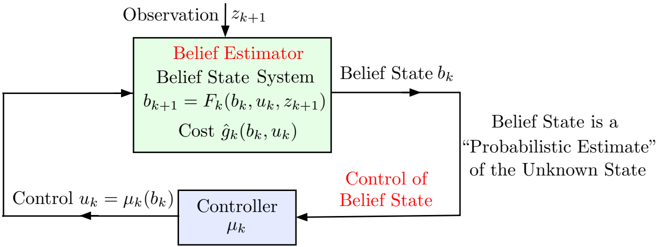
Recap of Partially Observed Markov Decision Problems (POMDPs)

- State space $X = \{1, \dots, n\}$, observation space Z, and control constraint set $U(i)$.
- Each state transition (i, j) generates a cost $g(i, u, j)$;
- and an observation z with probability $p(z \mid j, u)$.
- Let $b(i)$ denote the conditional probability that the state is i, given the history.
- The belief state is defined as $b = \big(b(1), b(2), \dots, b(n)\big)$.
- The belief b is updated using a **belief estimator** $F(b, u, z)$.
- Goal: Find a policy as a function of b that minimizes the cost.



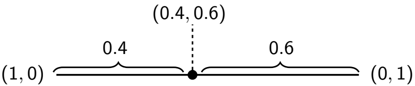
Recap of Partially Observed Markov Decision Problems (POMDPs)

- State space $X = \{1, \ldots, n\}$, observation space Z, and control constraint set $U(i)$.
- Each state transition (i, j) generates a cost $g(i, u, j)$;
- and an observation z with probability $p(z \mid j, u)$.
- Let $b(i)$ denote the conditional probability that the state is i, given the history.
- The belief state is defined as $b = \big(b(1), b(2), \ldots, b(n)\big)$.
- The belief b is updated using a **belief estimator** $F(b, u, z)$.
- Goal: Find a policy as a function of b that minimizes the cost.

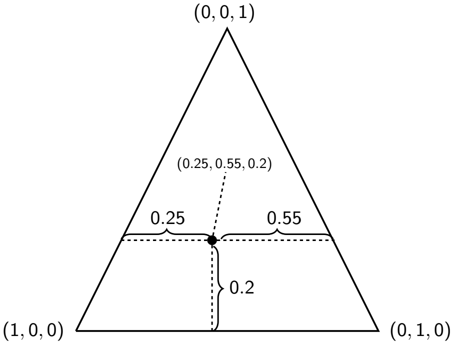


The Belief Space

- The belief b resides in the belief space B, i.e., the $n-1$ dimensional unit simplex.
- For example, if the states are $\{0,1\}$, then $b \in [0,1]$.



(a) *1-dimensional unit simplex.*



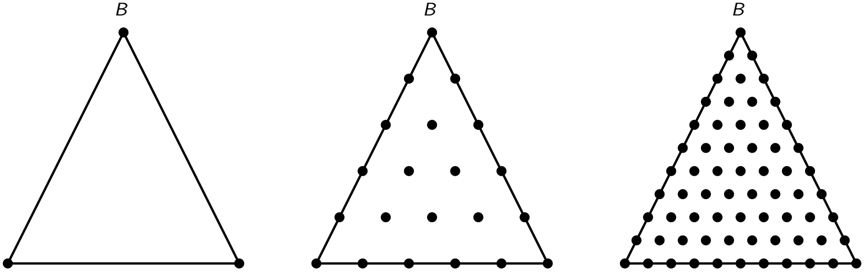
(b) *2-dimensional unit simplex.*

Aggregation of the Belief Space into Representative Beliefs

- We can obtain representative beliefs via uniform discretization of the belief space:

$$\mathcal{A} = \left\{ b \mid b \in B, b(i) = \frac{k_i}{\rho}, \sum_{i=1}^{n} k_i = \rho, k_i \in \{0, \dots, \rho\} \right\},$$

where ρ serves as the discretization resolution.

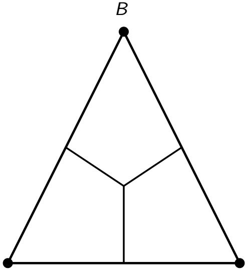


Discretization resolution $\rho = 1$ Discretization resolution $\rho = 5$ Discretization resolution $\rho = 10$

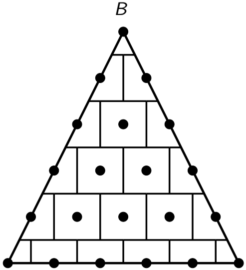
Hard Aggregation of the Belief Space

- We can implement hard aggregation via the nearest neighbor mapping:

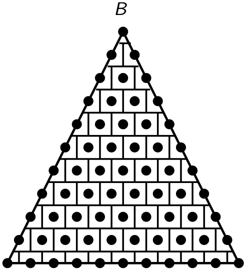
$$\phi_{by} = 1 \text{ if and only if y is the nearest neighbor of b, where } b \in B \text{ and } y \in \mathcal{A}.$$



Discretization resolution $\rho = 1$



Discretization resolution $\rho = 5$

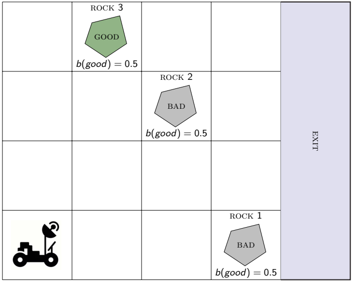


Discretization resolution $\rho = 10$

Example POMDP: **Rocksample (4,3)**

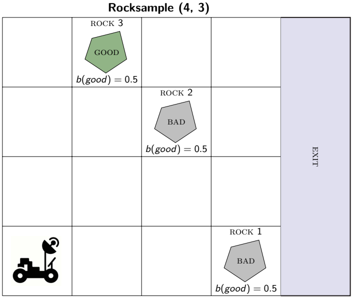
- Problem: rover exploration on Mars to find "good" rocks with high scientific value.
- There are 3 rocks on a 4×4 grid. The rover does not know which rocks are good.
- The controls (north, south, east, west) moves the rover (at cost 0.1).
- The control "sampling" determines the rock quality at the rover position (cost 10 for sampling a bad rock and cost -10 for sampling a good rock).
- Control "check-l" applies a **sensor to check the quality of rock l** (at cost 1).
- Accuracy of the sensor decreases exponentially with Euclidean distance to the rock.
- The rover stops the mission by moving to the right, yielding an exit-cost of -10.

Rocksample (4, 3)

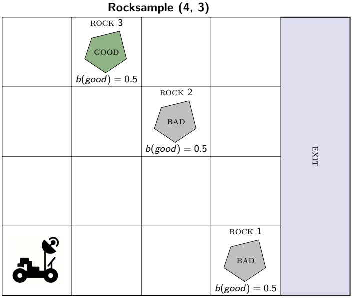


Example POMDP: **Rocksample (4,3)**

- Problem: rover exploration on Mars to find "good" rocks with high scientific value.
- There are 3 rocks on a 4×4 grid. The rover does not know which rocks are good.
- The controls (north, south, east, west) moves the rover (at cost 0.1).
- The control "sampling" determines the rock quality at the rover position (cost 10 for sampling a bad rock and cost -10 for sampling a good rock).
- Control "check-l" applies a **sensor to check the quality of rock** l (at cost 1).
- Accuracy of the sensor decreases exponentially with Euclidean distance to the rock.
- The rover stops the mission by moving to the right, yielding an exit-cost of -10.



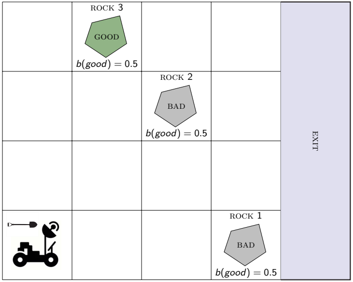
- Problem: rover exploration on Mars to find "good" rocks with high scientific value.
- There are 3 rocks on a 4×4 grid. The rover does not know which rocks are good.
- The controls (north, south, east, west) moves the rover (at cost 0.1).
- The control "sampling" determines the rock quality at the rover position (cost 10 for sampling a bad rock and cost -10 for sampling a good rock).
- Control "check-l" applies a **sensor to check the quality of rock** l (at cost 1).
- Accuracy of the sensor decreases exponentially with Euclidean distance to the rock.
- The rover stops the mission by moving to the right, yielding an exit-cost of -10.



Example POMDP: **Rocksample (4,3)**

- Problem: rover exploration on Mars to find "good" rocks with high scientific value.
- There are 3 rocks on a 4×4 grid. The rover does not know which rocks are good.
- The controls (north, south, east, west) moves the rover (at cost 0.1).
- The control "sampling" determines the rock quality at the rover position (cost 10 for sampling a bad rock and cost -10 for sampling a good rock).
- Control "check-l" applies a **sensor to check the quality of rock** l (at cost 1).
- Accuracy of the sensor decreases exponentially with Euclidean distance to the rock.
- The rover stops the mission by moving to the right, yielding an exit-cost of -10.

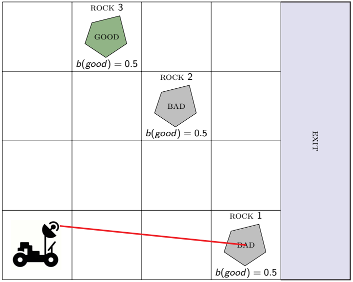
Rocksample (4, 3)



Example POMDP: **Rocksample (4,3)**

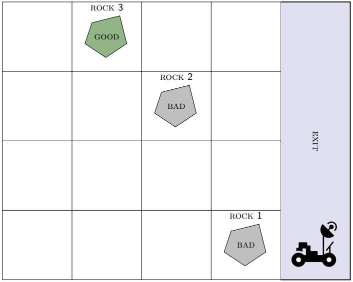
- Problem: rover exploration on Mars to find “good” rocks with high scientific value.
- There are 3 rocks on a 4×4 grid. The rover does not know which rocks are good.
- The controls (north, south, east, west) moves the rover (at cost 0.1).
- The control “sampling” determines the rock quality at the rover position (cost 10 for sampling a bad rock and cost -10 for sampling a good rock).
- Control “check-l” applies a **sensor to check the quality of rock l** (at cost 1).
- Accuracy of the sensor decreases exponentially with Euclidean distance to the rock.
- The rover stops the mission by moving to the right, yielding an exit-cost of -10.

Rocksample (4, 3)



- Problem: rover exploration on Mars to find "good" rocks with high scientific value.
- There are 3 rocks on a 4×4 grid. The rover does not know which rocks are good.
- The controls (north, south, east, west) moves the rover (at cost 0.1).
- The control "sampling" determines the rock quality at the rover position (cost 10 for sampling a bad rock and cost -10 for sampling a good rock).
- Control "check-l" applies a **sensor to check the quality of rock** l (at cost 1).
- Accuracy of the sensor decreases exponentially with Euclidean distance to the rock.
- The rover stops the mission by moving to the right, yielding an exit-cost of -10.

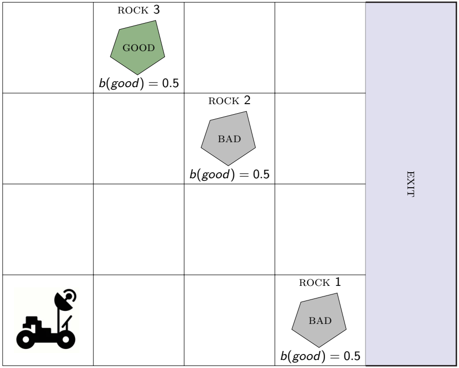
Rocksample (4, 3)



Approximating Rocksample (4,3) via Representative Aggregation

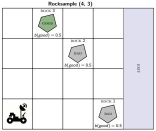
- The Rocksample (4,3) POMDP has a 127-dimensional belief space.
- We **discretize the belief space** with three different resolutions:
  - $\rho = 1$ leads to an aggregate problem with 128 representative beliefs.
  - $\rho = 2$ leads to an aggregate problem with 8256 representative beliefs.
  - $\rho = 3$ leads to an aggregate problem with 357760 representative beliefs.

Rocksample (4, 3)



Approximating Rocksample (4,3) via Representative Aggregation

- The Rocksample (4,3) POMDP has a 127-dimensional belief space.
- We **discretize the belief space** with three different resolutions:
  - $\rho = 1$ leads to an aggregate problem with 128 representative beliefs.
  - $\rho = 2$ leads to an aggregate problem with 8256 representative beliefs.
  - $\rho = 3$ leads to an aggregate problem with 357760 representative beliefs.

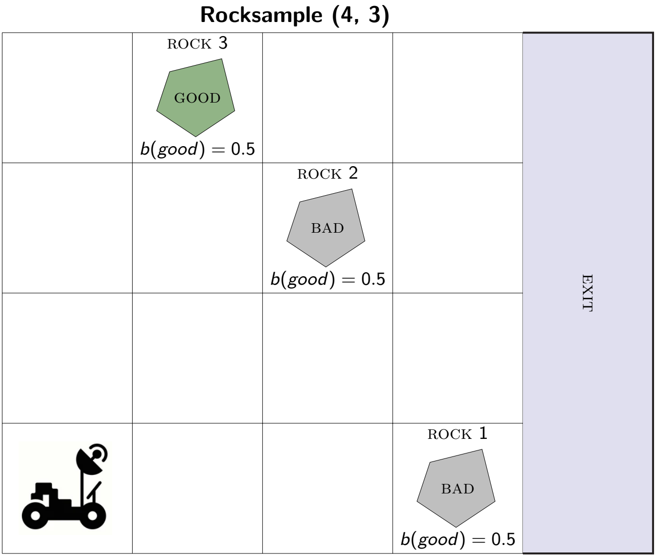


Aggregation Methodology

The representative beliefs define an aggregate dynamic programming problem, which we solve using value iteration to obtain $\tilde{J}$. We then use $\tilde{J}$ to obtain a one-step lookahead policy as

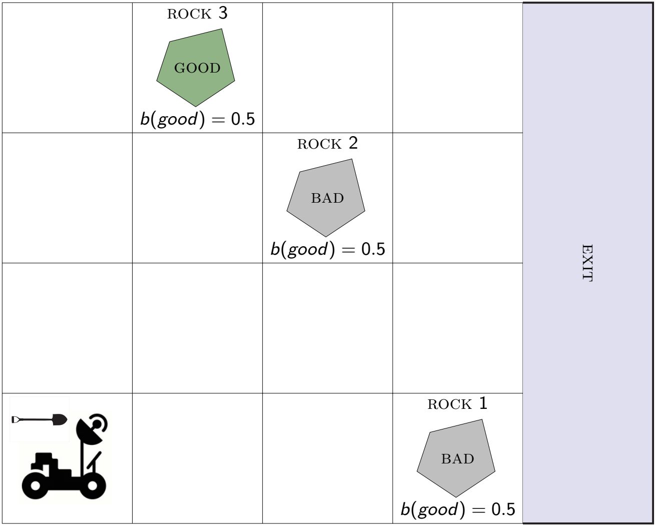
$$\mu(b) \in \arg\min_{u \in U(i)} \left\{ \hat{g}(b,u) + \alpha \sum_{z \in Z} \hat{p}(z \mid b,u) \tilde{J}(F(b,u,z)) \right\}, \quad b \in B.$$

Animation Setup

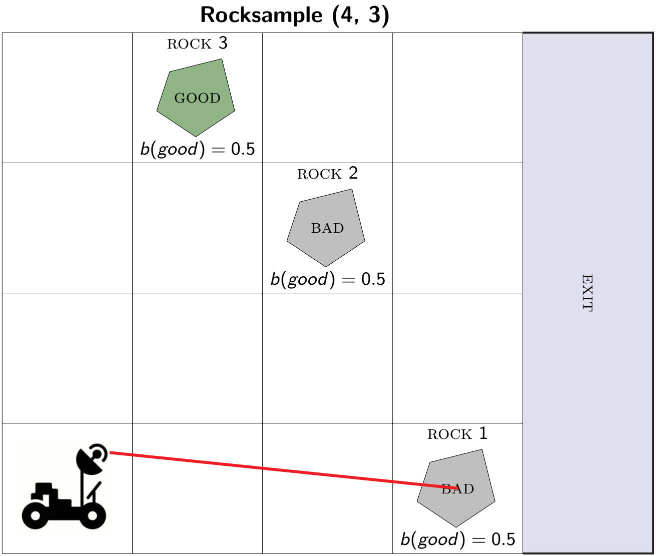


Animation Setup

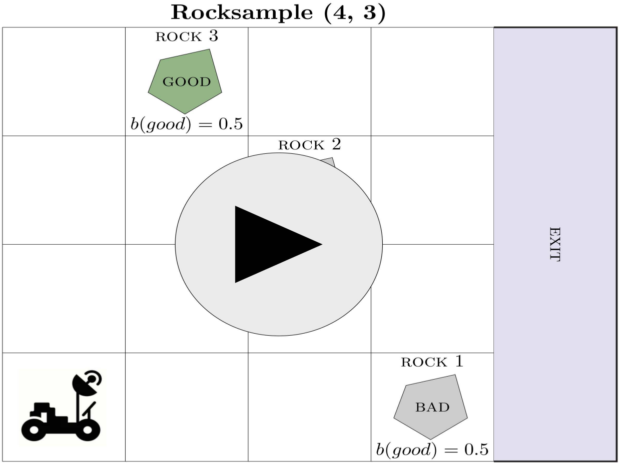
Rocksample (4, 3)



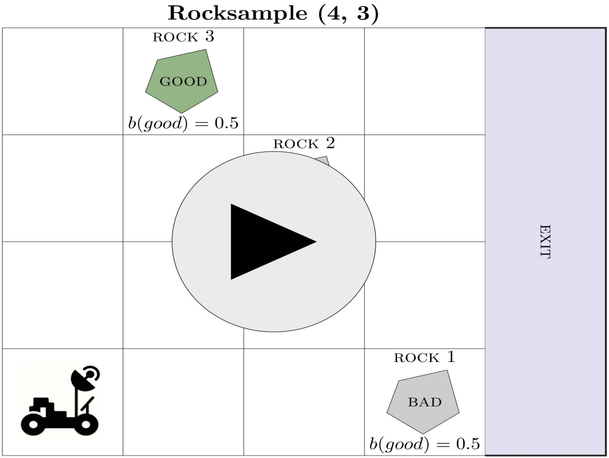
Animation Setup

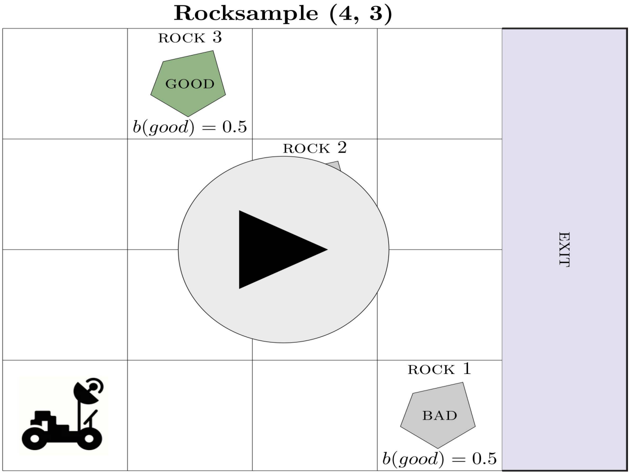


Animation of Control Policy Based on Aggregation with $\rho = 1$



Animation of Control Policy Based on Aggregation with $\rho = 2$




Rocksample (4, 3)
ROCK 3
GOOD
$b(good) = 0.5$
ROCK 2
ROCK 1
BAD
$b(good) = 0.5$
EXIT

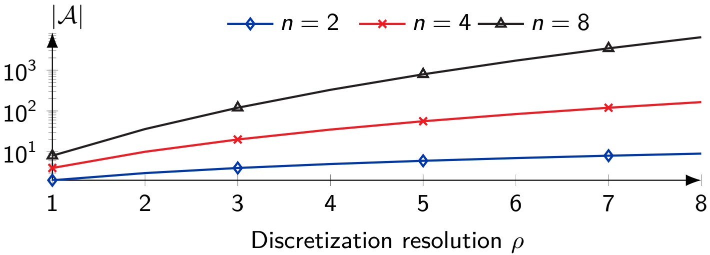
- The animations show that performance improves with the discretization resolution ρ.
- This is not surprising. One can show that $\lim_{\rho \to \infty} \|J^* - \tilde{J}\| = 0$.
- However, the computational complexity increases with the resolution ρ.

- The animations show that performance improves with the discretization resolution ρ.
- This is not surprising. One can show that $\lim_{\rho\to\infty} \|J^* - \tilde{J}\| = 0$.
- However, the computational complexity increases with the resolution ρ.

Computational Trade-Offs

- The animations show that performance improves with the discretization resolution ρ.
- This is not surprising. One can show that $\lim_{\rho \to \infty} \|J^* - \tilde{J}\| = 0$.
- However, the computational complexity increases with the resolution ρ.

- The animations show that performance improves with the discretization resolution ρ.
- This is not surprising. One can show that $\lim_{\rho\to\infty} \|J^* - \tilde{J}\| = 0$.
- However, the computational complexity increases with the resolution ρ.



Comparison Between Aggregation and Other POMDP Methods

POMDP	States n	Observations $\|Z\|$	Controls $\|U\|$	Discount factor α
RS (4,4)	257	2	9	0.95
RS (5,5)	801	2	10	0.95
RS (5,7)	3201	2	12	0.95
RS (7,8)	12545	2	13	0.95
RS (10,10)	102401	2	15	0.95

Table: POMDPs used for the experimental evaluation.

Method	Aggregation	Point-based	Heuristic search	Policy-based	Exact DP
Our method	✓				
IP					✓
PBVI		✓			
SARSOP		✓			
POMCP			✓		
HSVI			✓		
AdaOPS			✓		
R-DESPOT			✓		
POMCPOW			✓		
PPO				✓	
PPG				✓	

Table: Methods used for the experimental evaluation; all methods are based on approximation schemes except IP, which uses exact dynamic programming.

Comparison Between Aggregation and Other POMDP Methods

Method \ POMDP	RS (4,4)	RS (5,5)	RS (5,7)	RS (7,8)	RS (10, 10)
Aggregation	-17.15/2.4	-18.12/125.5	-17.51/189.1	-14.71/202	-11.59/500
IP	N/A	N/A	N/A	N/A	N/A
PBVI	-8.24/300	-9.05/300	N/A	N/A	N/A
SARSOP	-**17.92**/10^{-2}	-**19.24**/58.5	N/A	N/A	N/A
POMCP	-8.64/1.6	-8.80/1.6	-9.81/1.6	-9.46/1.6	-**8.98/1.6**
HSVI	-**17.92**/10^{-2}	-**19.24**/6.2	-**24.69**/721.3	N/A	N/A
PPO	-8.57/300	-8.15/300	-8.76/300	-7.35/300	-4.59/1000
PPG	-8.57/300	-8.24/300	-8.76/300	-7.35/300	-4.41/1000
AdaOPS	-16.95/1.6	-17.39/1.6	-16.14/1.6	-15.99/1.6	-15.29/1.6
R-DESPOT	-12.07/1.6	-12.09/1.6	-12.00/1.6	-13.14/1.6	-10.41/1.6
POMCPOW	-8.60/1.6	-8.47/1.6	-8.26/1.6	-8.14/1.6	-7.88/1.6

Table: Evaluation results on the benchmark POMDPs; the first number in each cell is the total discounted cost; the second is the compute time in minutes (online methods were given 1 second planning time per control); cells with N/A indicate cases where a result could not be obtained for computational reasons. RS(m,l) stands for an instance of Rocksample with an $m \times m$ grid and l rocks.

A Fifteen-Minute Break

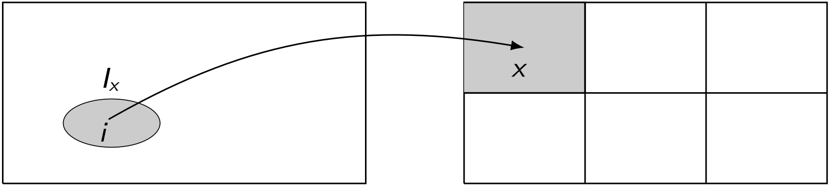
- Digest the first half of the lecture.
- The next half will cover
  - General aggregation; and
  - a case study of using aggregation for network security.

Outline

1. Aggregation with Representative States
2. **Example:** Aggregation with Representative States for POMDPs
3. **General Aggregation Methodology**
4. **Case study:** Aggregation for Cybersecurity

General Aggregation: Replace Representative States with Subsets

- Introduce a finite set of aggregate states $\mathcal{A}$.
- Each aggregate state $x \in \mathcal{A}$ is associated with a disjoint subset $I_x \subset \{1, \ldots, n\}$.



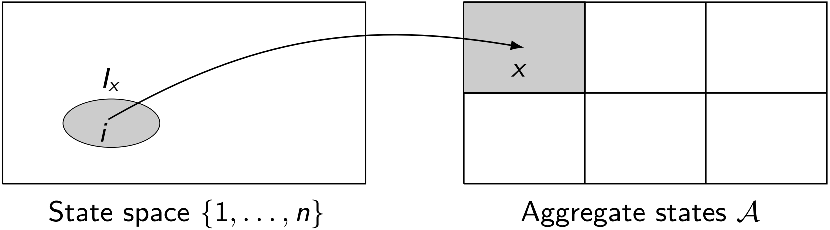
State space $\{1, \ldots, n\}$ Aggregate states $\mathcal{A}$

Question

Why is aggregation with rep. states a special case of general aggregation?

General Aggregation: Replace Representative States with Subsets

- Introduce a finite set of aggregate states $\mathcal{A}$.
- Each aggregate state $x \in \mathcal{A}$ is associated with a disjoint subset $I_x \subset \{1, \dots, n\}$.

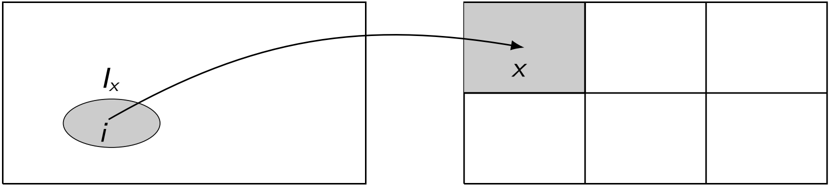


Question

Why is aggregation with rep. states a special case of general aggregation?

General Aggregation: Replace Representative States with Subsets

- Introduce a finite set of aggregate states $\mathcal{A}$.
- Each aggregate state $x \in \mathcal{A}$ is associated with a disjoint subset $I_x \subset \{1, \dots, n\}$.



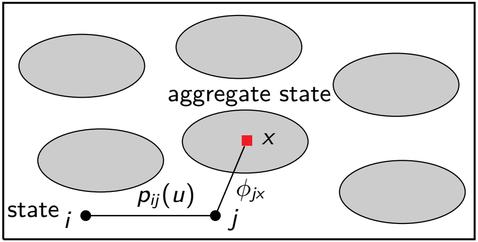
State space $\{1, \dots, n\}$ Aggregate states $\mathcal{A}$

Question

Why is aggregation with rep. states a special case of general aggregation?

Answer

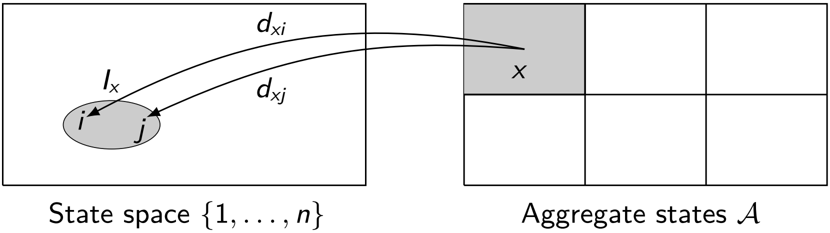
If each I_x contains a single state, we obtain the representative states framework.



State space $\{1, \dots, n\}$

For each state $j \in \{1, \dots, n\}$, we associate

- *aggregation probabilities* $\{\phi_{jx} \mid x \in \mathcal{A}\}$, where $\phi_{jx} = 1$ for all $j \in I_x$.

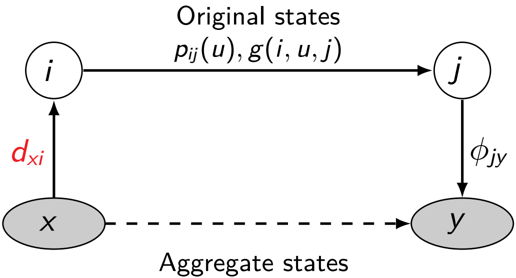


For every *aggregate state* $x \in \mathcal{A}$, we associate

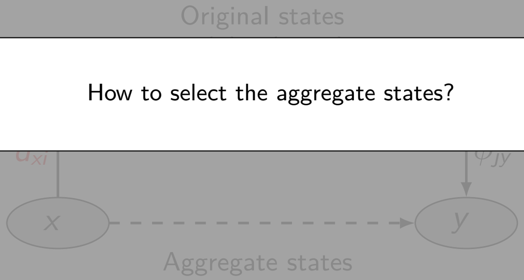
- *disaggregation probabilities* $\{d_{xi} \mid i = 1, \ldots, n\}$, where $d_{xi} = 0$ for all $i \notin I_x$.

Dynamic System of General Aggregation

- Similar to the earlier case, the aggregation leads to a dynamic system.
- This system can be understood through the following transition diagram.



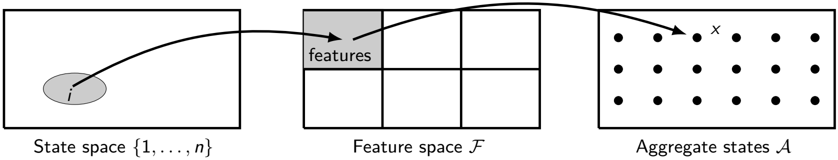
- Similar to the earlier case, the aggregation leads to a dynamic system.
- This system can be understood through the following transition diagram.



How to select the aggregate states?

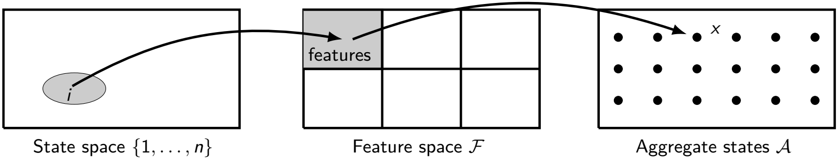
Feature-Based Aggregation

- Suppose that we have a feature mapping $F(i)$ that maps states into feature vectors.
- The mapping F can be obtained using engineering intuition or deep learning.
- We can then form the aggregate states I_x by grouping states with similar features.



Feature-Based Aggregation

- Suppose that we have a feature mapping $F(i)$ that maps states into feature vectors.
- The mapping F can be obtained using engineering intuition or deep learning.
- We can then form the aggregate states I_x by grouping states with similar features.

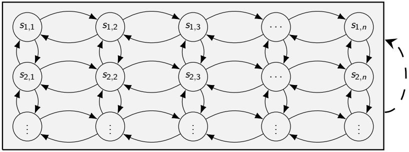


Outline

Traditional/Current Network Security Operations

Adaptive Control Methodology for Network Security Operations

Simulation System

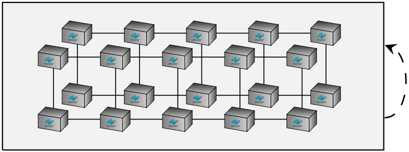


POMDP Model & Approximate DP

Policy Mapping μ

System Identification

Emulation System

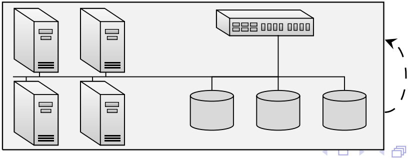


Policy Evaluation & Model Estimation

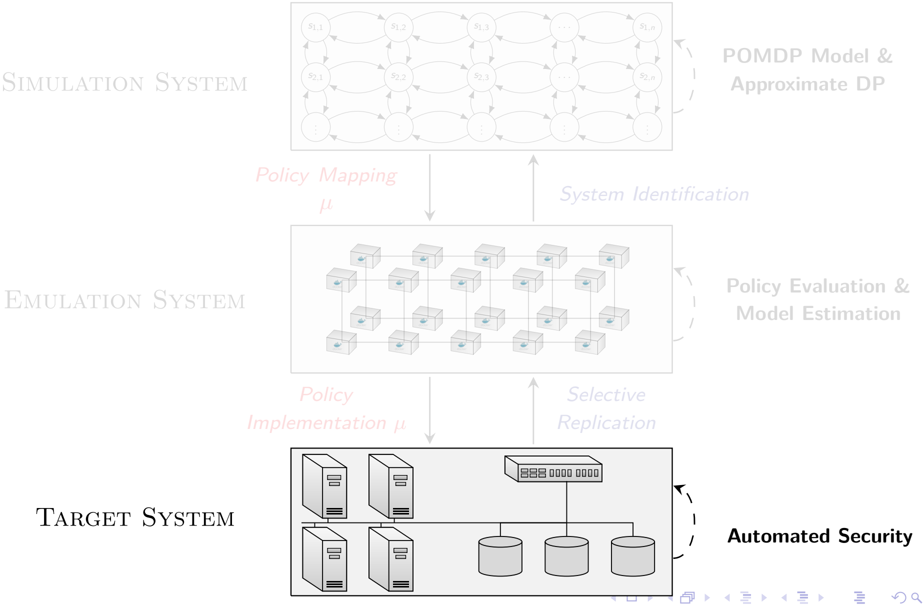
Policy Implementation μ

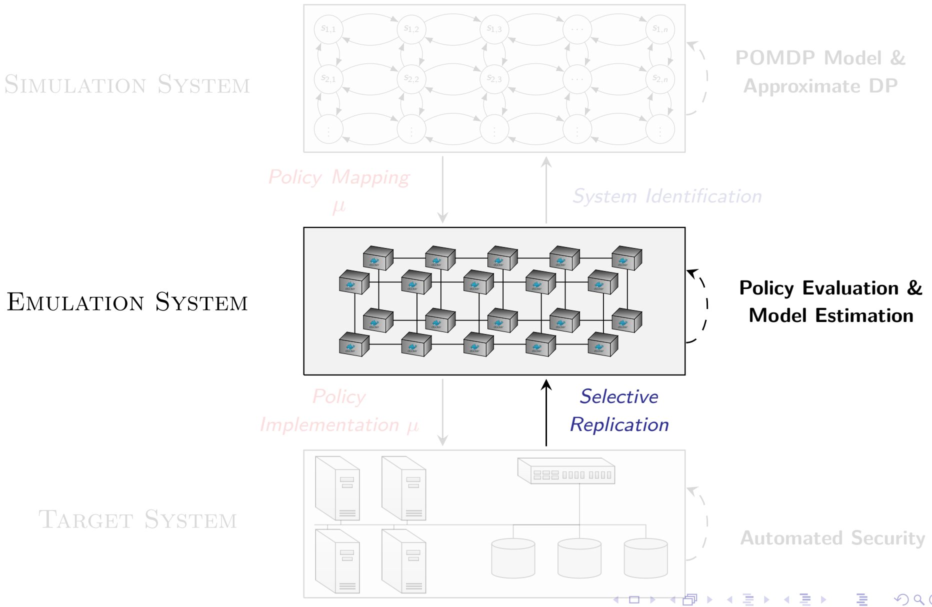
Selective Replication

Target System

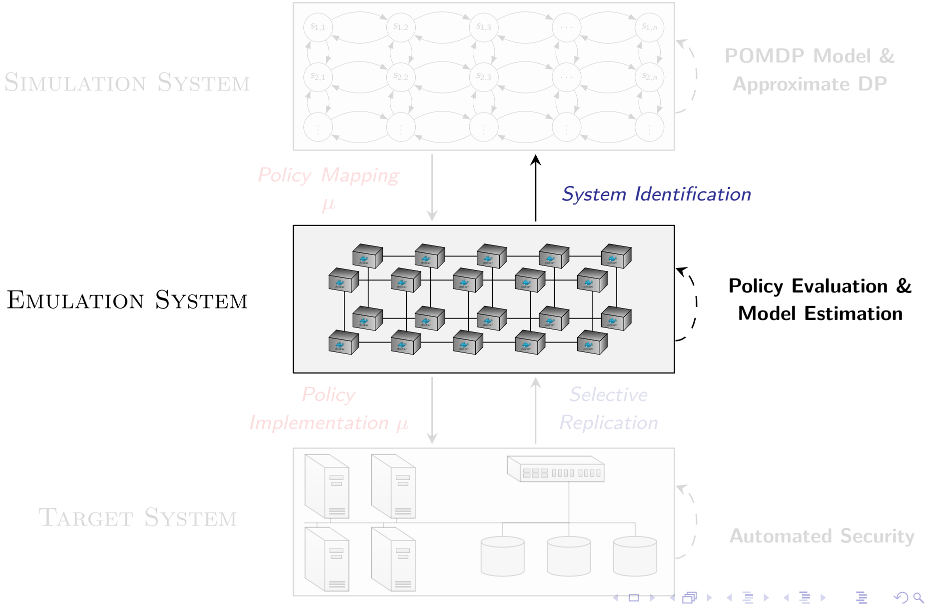


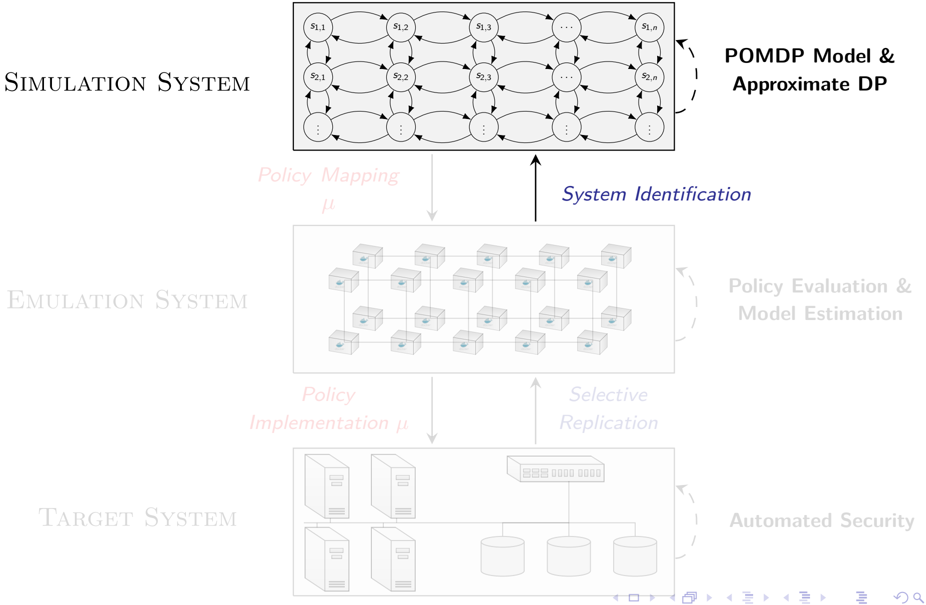
Automated Security


SIMULATION SYSTEM
POMDP Model & Approximate DP
Policy Mapping μ
System Identification
EMULATION SYSTEM
Policy Evaluation & Model Estimation
Policy Implementation μ
Selective Replication
TARGET SYSTEM
Automated Security

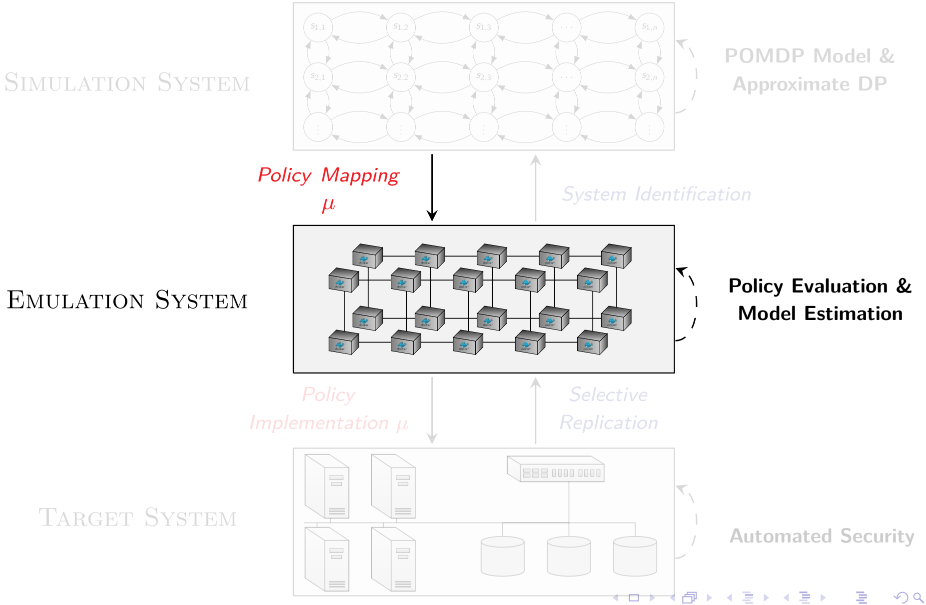

SIMULATION SYSTEM
POMDP Model & Approximate DP
Policy Mapping μ
System Identification
EMULATION SYSTEM
Policy Evaluation & Model Estimation
Policy Implementation μ
Selective Replication
TARGET SYSTEM
Automated Security

Adaptive Control Methodology for Network Security

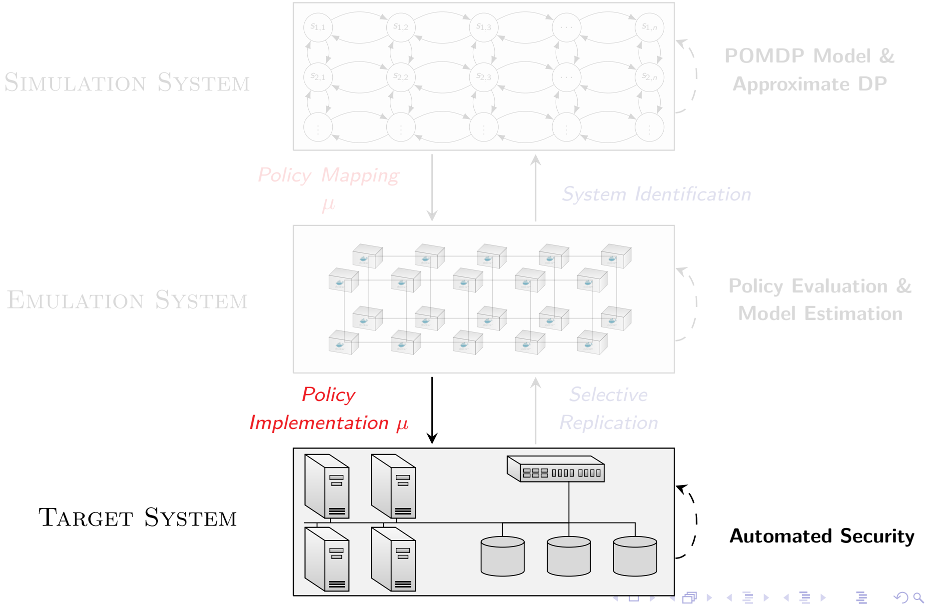



Simulation System
$s_{1,1}$ $s_{1,2}$ $s_{1,3}$ $\dots$ $s_{1,n}$
$s_{2,1}$ $s_{2,2}$ $s_{2,3}$ $\dots$ $s_{2,n}$
POMDP Model & Approximate DP
Policy Mapping μ
System Identification
Emulation System
Policy Evaluation & Model Estimation
Policy Implementation μ
Selective Replication
Target System
Automated Security

Adaptive Control Methodology for Network Security



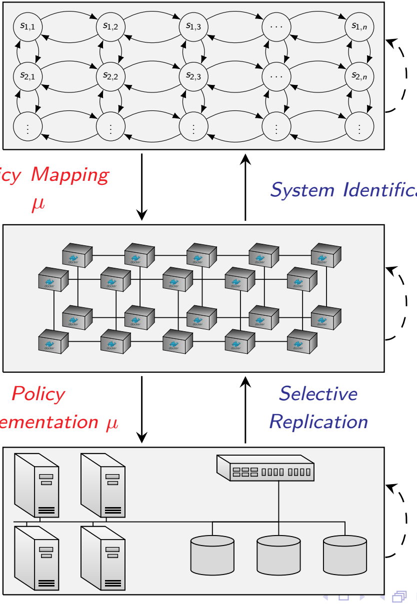
Adaptive Control Methodology for Network Security



Adaptive Control Methodology for Network Security

SIMULATION SYSTEM

POMDP Model & Approximate DP



Policy Mapping μ

System Identification

EMULATION SYSTEM

Policy Evaluation & Model Estimation

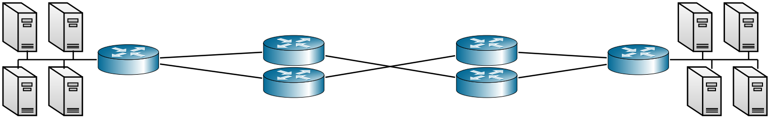
Policy Implementation μ

Selective Replication

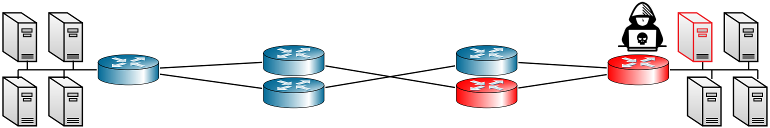
TARGET SYSTEM

Automated Security

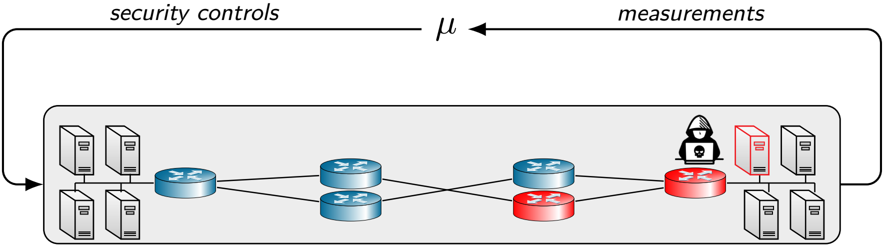
Problem: Finding an Effective Network Security Policy



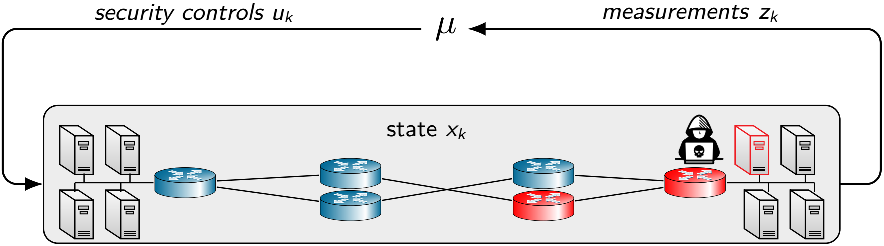
Problem: Finding an Effective Network Security Policy



Problem: Finding an Effective Network Security Policy

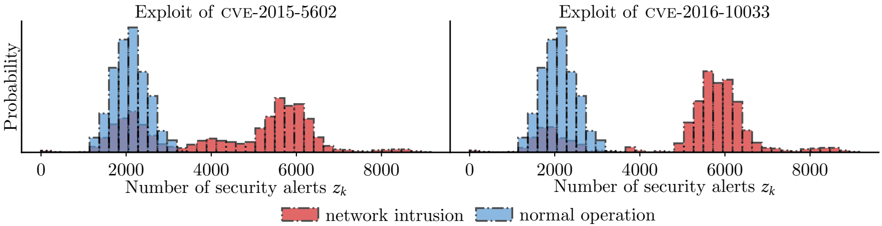


Problem: Finding an Effective Network Security Policy



- Mathematically, the problem can be formulated as a POMDP.
- Hidden states: the security status of each network component.
- **For example,** $x_k = (x_{k,1}, \dots, x_{k,M})$, where
  - M is the number of components.
  - $x_{k,i} = 1$ if component i is compromised at stage k, 0 otherwise.

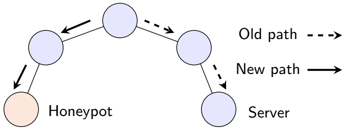
Observations



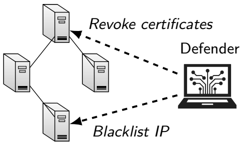
- The system emits observations in the form of logs, performance metrics, and alerts.
- These observations give **partial information about the security** of the system.

Controls

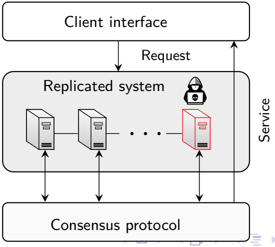
Flow control
By redirecting traffic, the defender can isolate malicious behavior.



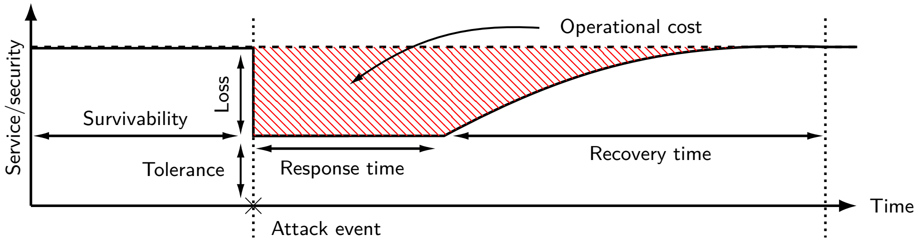
Access control
By adjusting resource permissions, defenders can prevent attackers from compromising critical assets.



Replication control
Replication can ensure that multiple replicas of services remain available even when some are compromised.



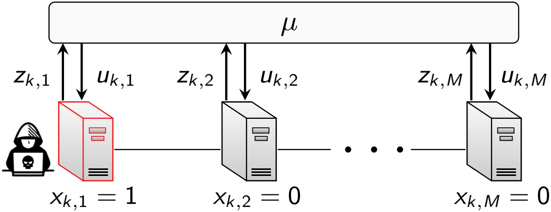
Cost



- The defender wants to minimize the impact of potential attacks.
- At the same time, the defender wants to maintain operational services to clients.

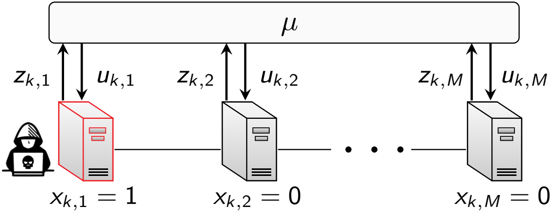
Example POMDP in The Context of Network Security

- **Networked system with M components.**
- Each component can be in two states: 1 (compromised) or 0 (safe).
- Each component logs security alerts z in real-time.
- Two controls per component: 1 (recovery) and 0 (wait).
- Compromised components and unnecessary recoveries incur costs.



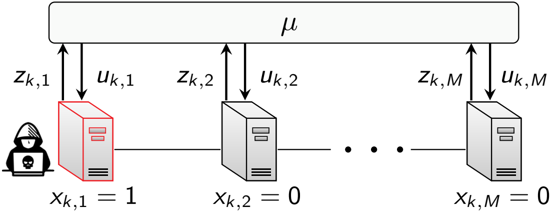
Example POMDP in The Context of Network Security

- **Networked system with M components.**
- Each component can be in two states: 1 (compromised) or 0 (safe).
- Each component logs security alerts z in real-time.
- Two controls per component: 1 (recovery) and 0 (wait).
- Compromised components and unnecessary recoveries incur costs.



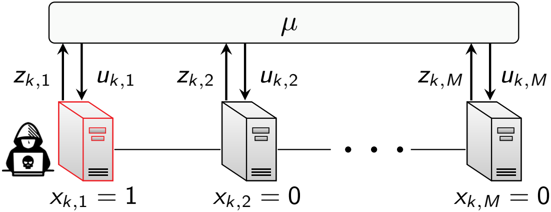
Example POMDP in The Context of Network Security

- **Networked system with M components.**
- Each component can be in two states: 1 (compromised) or 0 (safe).
- Each component logs security alerts z in real-time.
- Two controls per component: 1 (recovery) and 0 (wait).
- Compromised components and unnecessary recoveries incur costs.



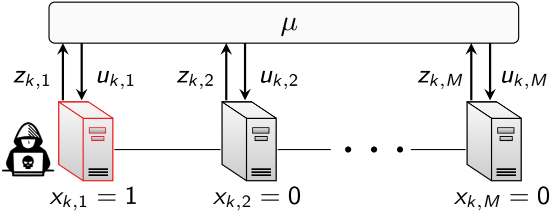
Example POMDP in The Context of Network Security

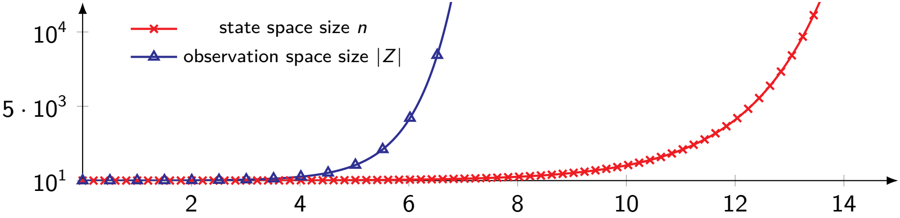
- **Networked system with M components.**
- Each component can be in two states: 1 (compromised) or 0 (safe).
- Each component logs security alerts z in real-time.
- Two controls per component: 1 (recovery) and 0 (wait).
- Compromised components and unnecessary recoveries incur costs.



Example POMDP in The Context of Network Security

- **Networked system with M components.**
- Each component can be in two states: 1 (compromised) or 0 (safe).
- Each component logs security alerts z in real-time.
- Two controls per component: 1 (recovery) and 0 (wait).
- Compromised components and unnecessary recoveries incur costs.





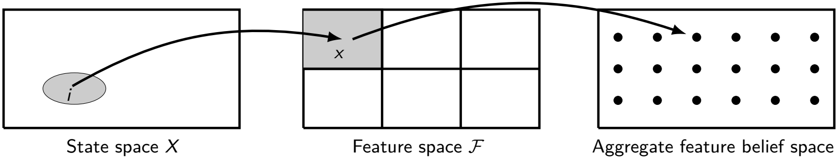
M: number of system components. (Assuming 4 observations per component.)

Scalability challenge

Problem complexity **grows exponentially** with the system size.

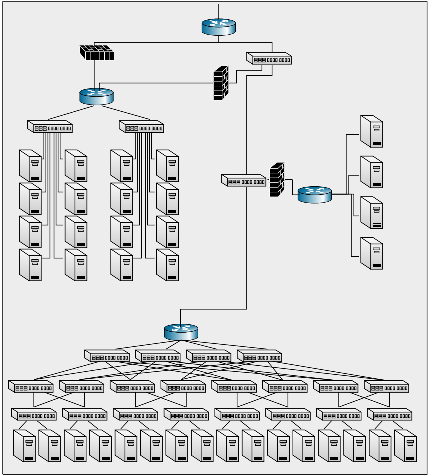
- For the networked systems that we consider, the number of (hidden) states is in the order of 10^{50}.
- We manage the computational complexity using feature-based aggregation.

- For the networked systems that we consider, the number of (hidden) states is in the order of 10^{50}.
- We manage the computational complexity using feature-based aggregation.



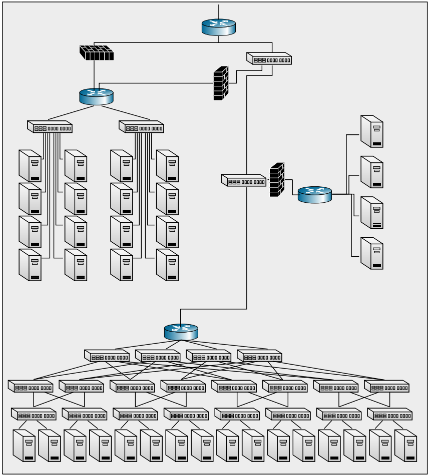
Designing Features for Aggregation

- Consider the infrastructure to the right.
- It comprises 64 components.
- Each component has a binary state:
  - State 1 means compromised.
  - State 0 means safe.
- $\implies 2^{64}$ states.
- $\implies (2^{64}-1)$-dimensional belief space.



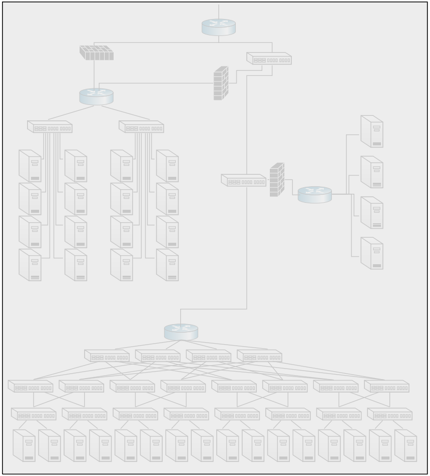
Designing Features for Aggregation

- Consider the infrastructure to the right.
- It comprises 64 components.
- Each component has a binary state:
  - State 1 means compromised.
  - State 0 means safe.
- $\implies 2^{64}$ states.
- $\implies (2^{64}-1)$-dimensional belief space.



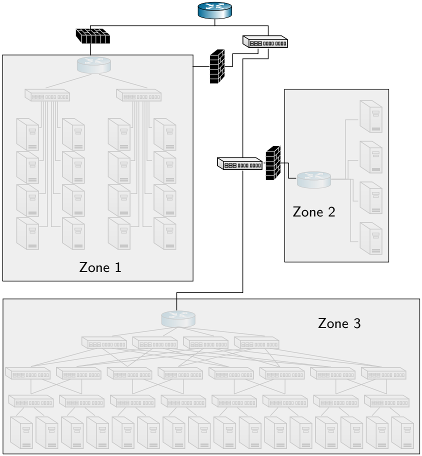
Designing Features for Aggregation

- We manage the complexity using feature-based aggregation.
- Introduce a set of features $\mathcal{F}$.
- For example, $\mathcal{F} = \{\text{Infrastructure-compromised}\}$.
- Then we **only have** 2 **feature combinations:**
  1. Safe
  2. Compromised
- $\implies$ 1-dimensional belief space.



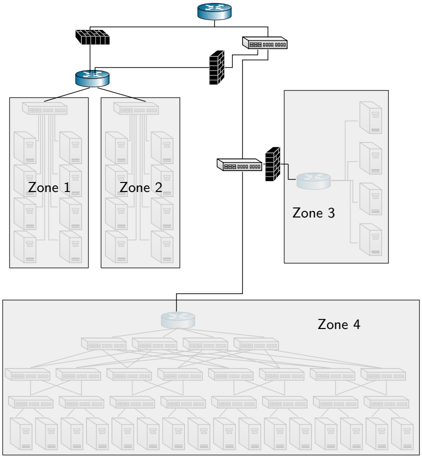
Designing Features for Aggregation

- Finer aggregations can be obtained by segmenting the network into zones.
- Let Zone-k represent the compromised state of zone k.
- If $\mathcal{F} = \{\text{Zone-1}, \text{Zone-2}, \text{Zone-3}\}$.
- Then we **have** 8 **feature combinations**.



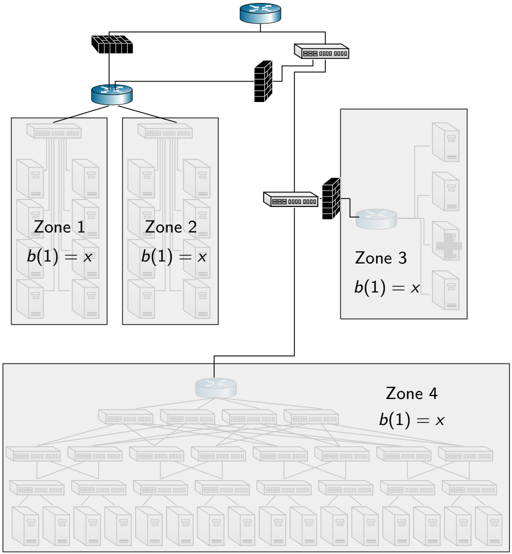
Designing Features for Aggregation

- If $\mathcal{F} = \{\text{Zone-1, Zone-2, Zone-3, Zone-4}\}$.
- Then we **have** 16 **feature combinations**.
- etc.
- Discretize the belief space over features into $\approx 200,000$ representative beliefs.
- $\implies$ Solve the aggregate problem to obtain the cost-function approximation $\tilde{J}$.
- Use $\tilde{J}$ to define a policy μ, e.g., a one-step lookahead policy.



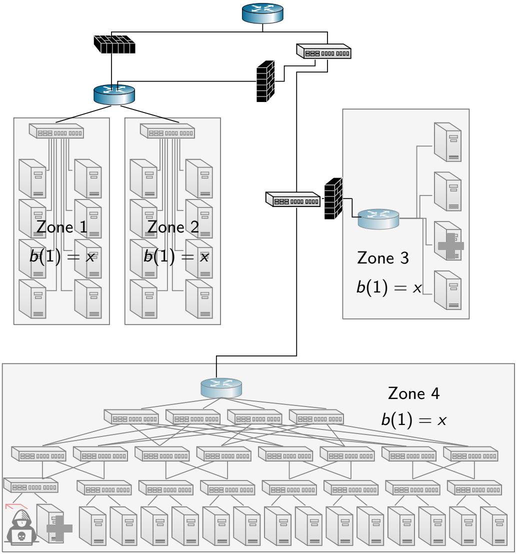
Aggregation allows to control the trade-off between computational cost and performance.

Animation: Security Policy Obtained From Aggregation



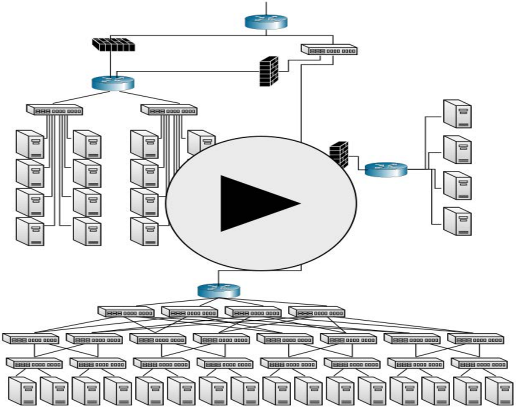
Time step=x, Cost: y, Avg. cost per time step = y,
Control:

Animation: Security Policy Obtained From Aggregation

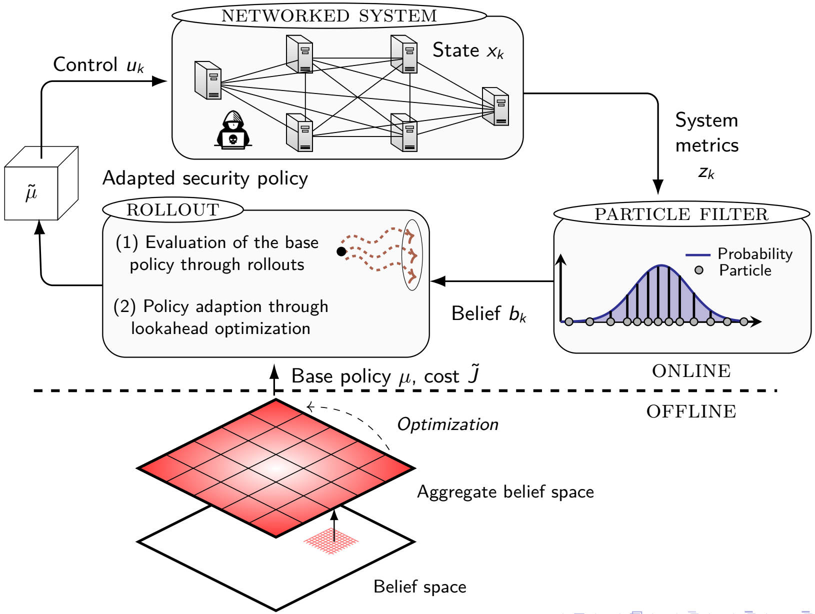


Time step=x, Cost: y, Avg. cost per time step = y,
Control:

Animation: Security Policy Obtained From Aggregation



Combining Aggregation with Rollout for On-line Policy Adaptation



Experimental Results On A Standard Benchmark Security Problem

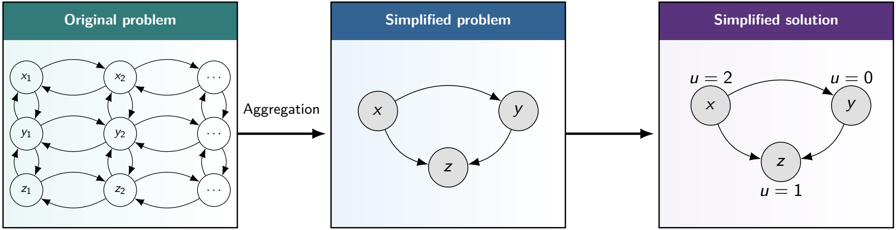
Method	*Offline/Online compute (min/s)*	*State estimation*	*Cost*
AGGREGATION	8.5/0.01	PARTICLE FILTER	61.72 ± 3.96
PPO	1000/0.01	LATEST OBSERVATION	341 ± 133
PPO	1000/0.01	PARTICLE FILTER	326 ± 116
PPG	1000/0.01	LATEST OBSERVATION	328 ± 178
PPG	1000/0.01	PARTICLE FILTER	312 ± 163
DQN	1000/0.01	LATEST OBSERVATION	516 ± 291
DQN	1000/0.01	PARTICLE FILTER	492 ± 204
PPO+ACTION PRUNING	300/0.01	LATEST OBSERVATION	57.45 ± 2.44
PPO+ACTION PRUNING	300/0.01	PARTICLE FILTER	56.45 ± 2.81
POMCP	0/15	PARTICLE FILTER	53.08 ± 3.78
POMCP	0/30	PARTICLE FILTER	53.18 ± 3.42
AGGREGATION+ROLLOUT	8.5/14.80	PARTICLE FILTER	$\mathbf{37.89 \pm 1.54}$

Numbers indicate the mean and the standard deviation from 1000 evaluations. We use 427500 representative beliefs, lookahead horizon $\ell = 2$, and truncated rollout with horizon $m = 20$.

Conclusion

- Aggregation provides a general methodology for approximate dynamic programming.

  1. Combine groups of similar states into aggregate states.
  2. Formulate an aggregate dynamic programming problem.
  3. Solve the aggregate problem using some computational method.
  4. Use the aggregate solution to approximate a solution to the original problem.



About the Next Lecture

- Dynamic programming for mini-max problems
- Rollout and approximation in value space for mini-max problems
- A meta algorithm for computer chess based on reinforcement learning.

Please review Section 2.12 of the "Course in RL" textbook.

Recommended videolecture (Computer chess with model predictive control and reinforcement learning) at https://www.youtube.com/watch?v=88LDkHaf1sU.

About the Next Lecture

- Dynamic programming for mini-max problems
- Rollout and approximation in value space for mini-max problems
- A meta algorithm for computer chess based on reinforcement learning.

Please review Section 2.12 of the “Course in RL" textbook.

Recommended videolecture (Computer chess with model predictive control and reinforcement learning) at https://www.youtube.com/watch?v=88LDkHaf1sU.

About the Next Lecture

- Dynamic programming for mini-max problems
- Rollout and approximation in value space for mini-max problems
- A meta algorithm for computer chess based on reinforcement learning.

Please review Section 2.12 of the “Course in RL" textbook.

Recommended videolecture (Computer chess with model predictive control and reinforcement learning) at https://www.youtube.com/watch?v=88LDkHaf1sU.