Adaptive Security Policies via Belief Aggregation and Rollout

NETCON SEMINAR

Kim Hammar, kimham@kth.se

March 3, 2025

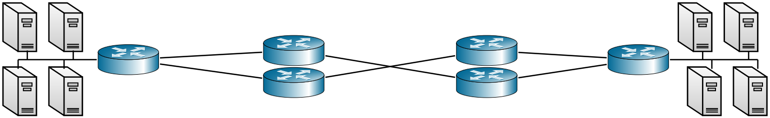
Joint work with Yuchao Li, Tansu Alpcan, Emil Lupu, and Dimitri Bertsekas

Based on work-in-progress papers ⌛:

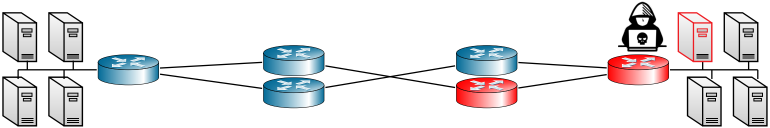
Adaptive Security Policies via Belief Aggregation and Rollout (K.H, Y.L, T.A, E.L)

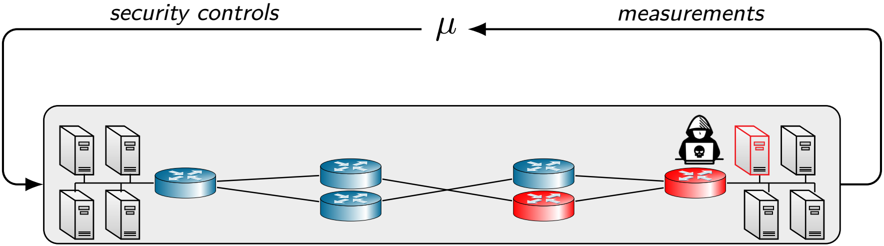
Feature-Based Belief Aggregation for POMDPs (Y.L, K.H, D.B)

Problem

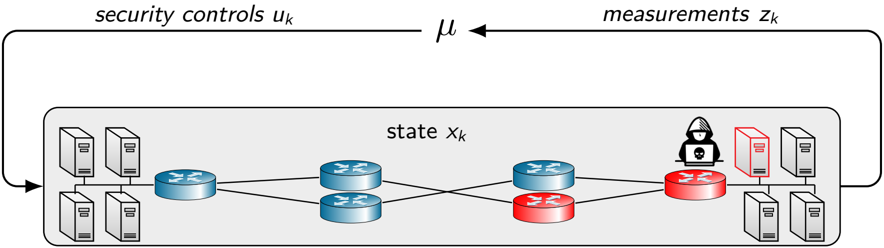


Problem




security controls
μ
measurements

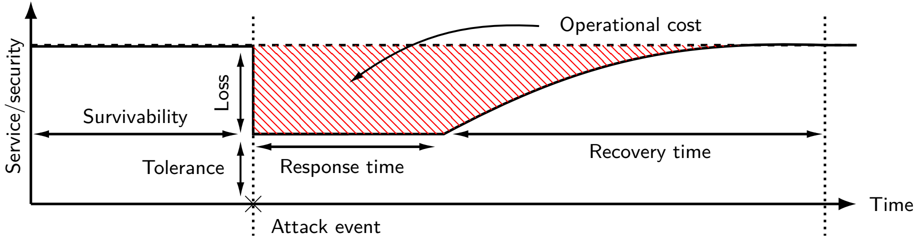
Problem



- Finite partially observed Markovian decision problem (POMDP).
- Hidden states $i \in X = \{1, \dots, n\}$, transition probability $p_{ij}(u)$.
- Observation $z \in Z$ is generated with probability $p(z \mid j, u)$.
- Control $u \in U$.
- **Goal**: find a policy μ that minimizes the discounted cost

$$E\left\{\sum_{k=0}^{\infty} \alpha^k g(x_k, u_k, x_{k+1})\right\}.$$

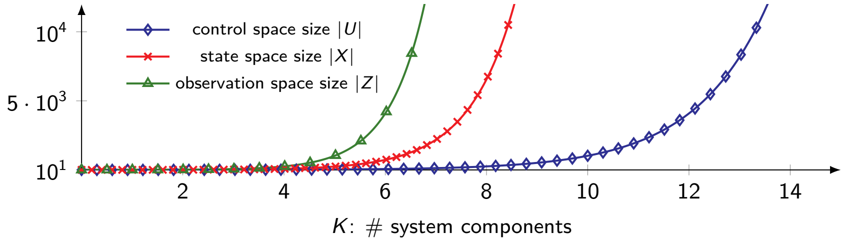
Cost



Operational cost
Service/security
Loss
Survivability
Tolerance
Response time
Recovery time
Time
Attack event

Challenge 1: Curse of Dimensionality

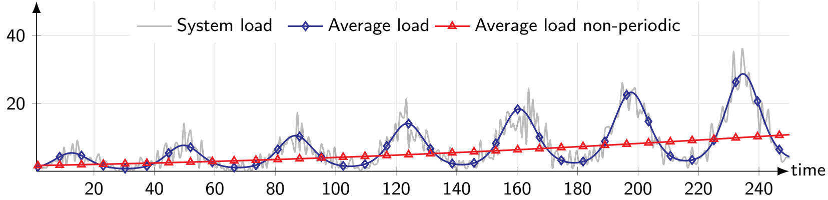
Problem complexity **grows exponentially** with system size.



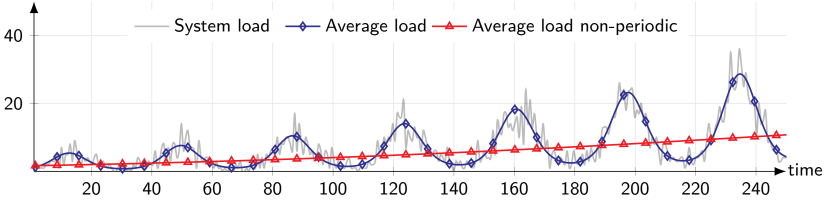
Scalability challenge

Benchmark POMDP (CAGE-2) has over 10^{47} states and 10^{25} observations.

- Networked systems change on a regular basis.
  - Components fail, bandwidth fluctuates, load patterns shift, software is updated, etc.



- Networked systems change on a regular basis.
  - Components fail, bandwidth fluctuates, load patterns shift, software is updated, etc.

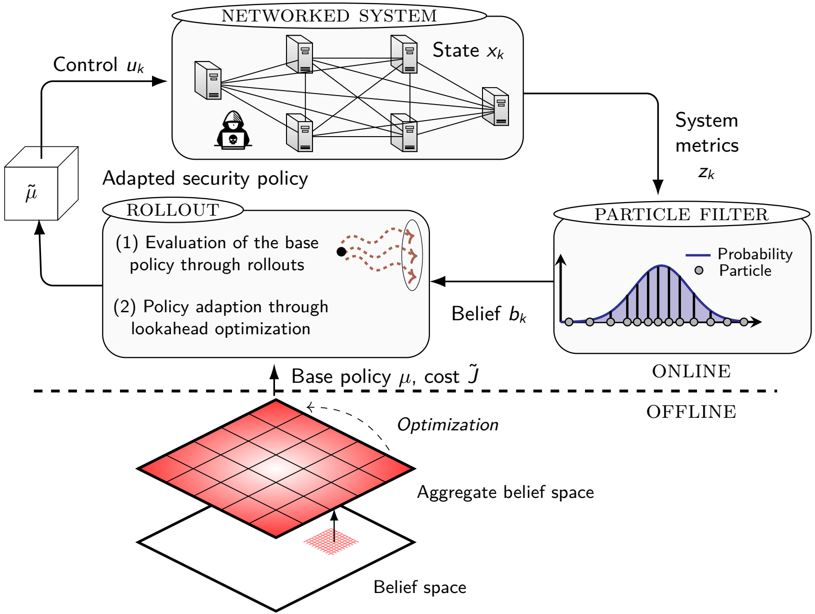


Policy adaption

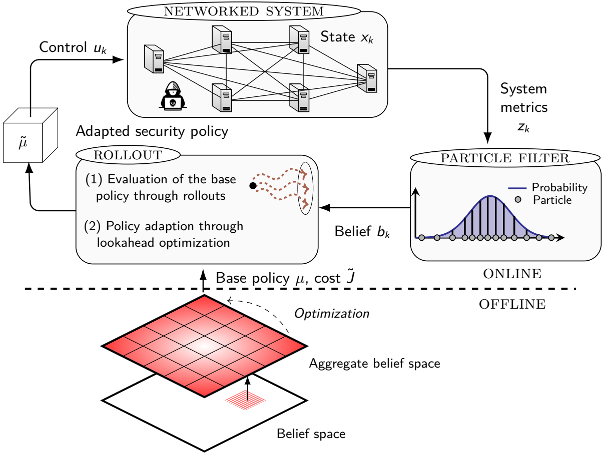
Need an efficient way to adapt the security policy μ when changes occur.

Our Approximation Framework for Large-Scale POMDPs

Our Approximation Framework for Large-Scale POMDPs



Our Approximation Framework for Large-Scale POMDPs



- Achieves state-of-the-art performance on the CAGE-2 benchmark.
  - POMDP with over 10^{47} states and 10^{25} observations.
- Has theoretical **performance guarantees**.
  - Contrasts with other approximation frameworks, e.g., DEEP RL and LLMs.

Belief state

$b = (b(1), \dots, b(n))$ is a probability distribution over the state space.

Let b be the state, we then obtain perfect information problem with **dynamics**

$$b_k = F(b_{k-1}, u_{k-1}, z_k) \qquad \text{(Belief estimator)}$$

$$\hat{g}(b,u) = \sum_{i=1}^{n} b(i) \sum_{j=1}^{n} g(i,u,j) \qquad \text{(Cost)}$$

$$\hat{p}(z \mid b, u) = \sum_{i=1}^{n} b(i) \sum_{j=1}^{n} p_{ij}(u) p(z \mid j, u) \qquad \text{(Disturbance distribution).}$$

Belief state

$b = (b(1), \dots, b(n)) \in B$ is a probability distribution over the state space.

Let b be the state, we then obtain perfect information problem with **dynamics**

$$b_k = F(b_{k-1}, u_{k-1}, z_k) \qquad \text{(Belief estimator)}$$

$$\hat{g}(b, u) = \sum_{i=1}^{n} b(i) \sum_{j=1}^{n} g(i, u, j) \qquad \text{(Cost)}$$

$$\hat{p}(z \mid b, u) = \sum_{i=1}^{n} b(i) \sum_{j=1}^{n} p_{ij}(u) p(z \mid j, u) \qquad \text{(Disturbance distribution).}$$

Challenges

- Computing an optimal policy is PSPACE-hard.

Belief state

$b = (b(1), \dots, b(n)) \in B$ is a probability distribution over the state space.

Let b be the state, we then obtain perfect information problem with **dynamics**

$$b_k = F(b_{k-1}, u_{k-1}, z_k) \qquad \text{(Belief estimator)}$$

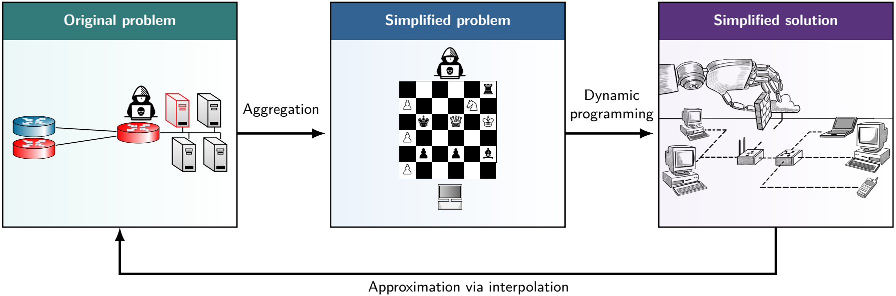
$$\hat{g}(b, u) = \sum_{i=1}^{n} b(i) \sum_{j=1}^{n} g(i, u, j) \qquad \text{(Cost)}$$

$$\hat{p}(z \mid b, u) = \sum_{i=1}^{n} b(i) \sum_{j=1}^{n} p_{ij}(u) p(z \mid j, u) \qquad \text{(Disturbance distribution).}$$

Challenges

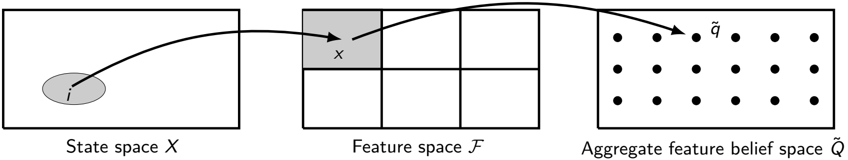
- Computing an optimal policy is PSPACE-hard.
- Enumerating the **dimension** of B is intractable $(|X| \geq 10^{47}$ in CAGE-2$)$.

Offline POMDP Approximation via Problem Simplification



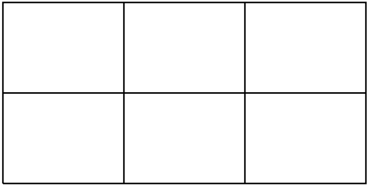
Two-Level Aggregation for Offline Approximation

We use **two-level aggregation** to simplify the POMDP into an aggregate MDP with finite state space, which we solve using dynamic programming.



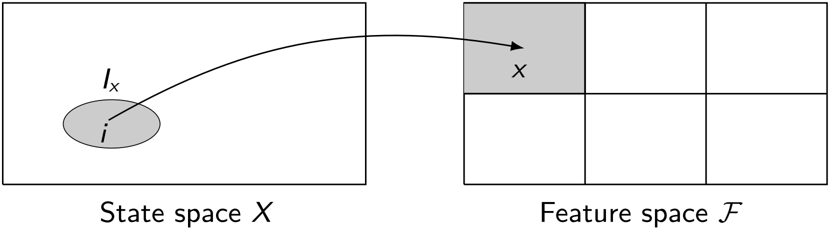
State Aggregation (1/3)

- Two main options to construct the **feature space** $\mathcal{F}$:
  - It can be manually designed based on engineering intuition.
  - It can be automatically constructed via a neural network.

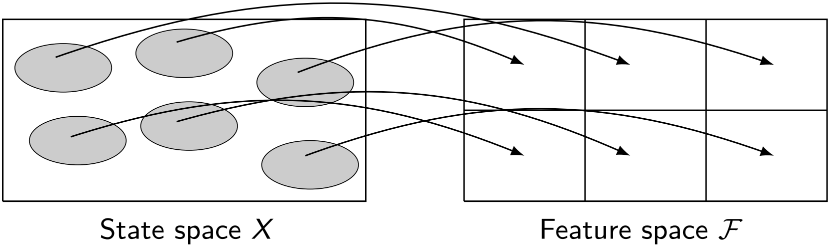


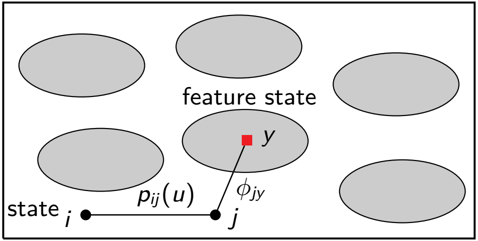
Feature space $\mathcal{F}$

Each ***feature state*** $x \in \mathcal{F}$ is associated with a disjoint subset $I_x \subset X$.



Each ***feature state*** $x \in \mathcal{F}$ is associated with a disjoint subset $I_x \subset X$.



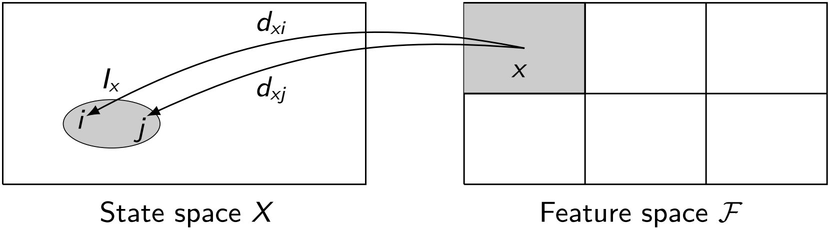


State space X

State aggregation

For each state $j \in X$, we associate

- an *aggregation probability* distribution $\{\phi_{jy} \mid y \in \mathcal{F}\}$, where $\phi_{jy} = 1$ for all $j \in I_y$.



State aggregation

For each state $j \in X$, we associate

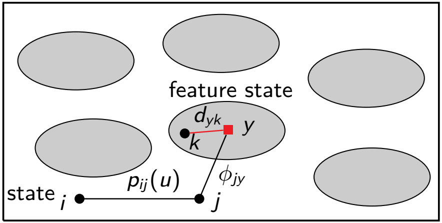
- an *aggregation probability* distribution $\{\phi_{jy} \mid y \in \mathcal{F}\}$, where $\phi_{jy} = 1$ for all $j \in I_y$.

Feature disaggregation

For every *feature state* $x \in \mathcal{F}$, we associate

- a *disaggregation probability* distribution $\{d_{xi} \mid i \in X\}$, where $d_{xi} = 0$ for all $i \notin I_x$.

State Aggregation (2/3)



State space X

State aggregation

For each state $j \in X$, we associate

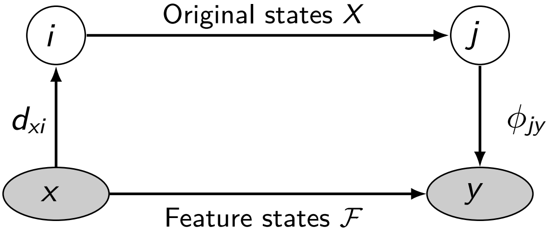
- an *aggregation probability* distribution $\{\phi_{jy} \mid y \in \mathcal{F}\}$, where $\phi_{jy} = 1$ for all $j \in I_y$.

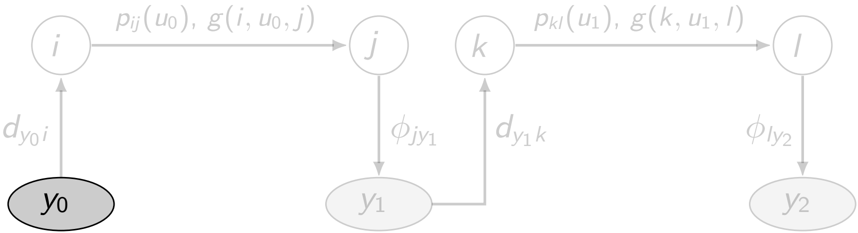
Feature disaggregation

For every *feature state* $x \in \mathcal{F}$, we associate

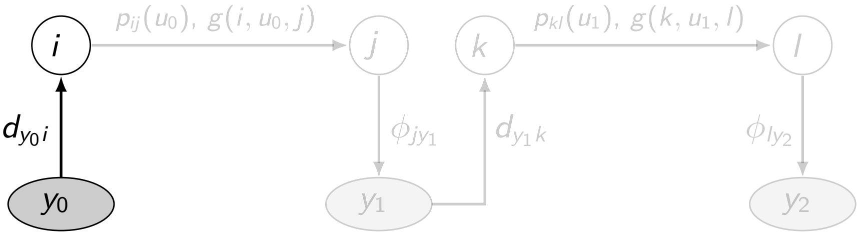
- a *disaggregation probability* distribution $\{d_{xi} \mid i \in X\}$, where $d_{xi} = 0$ for all $i \notin I_x$.

We obtain the **dynamic system**:

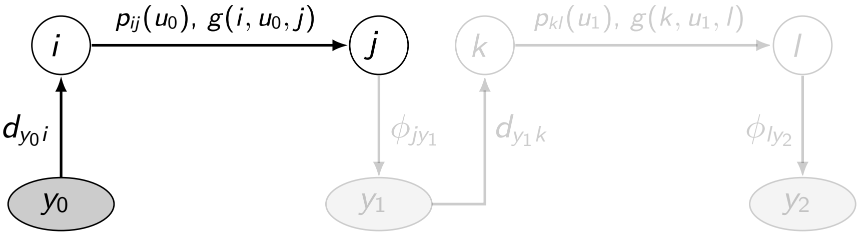




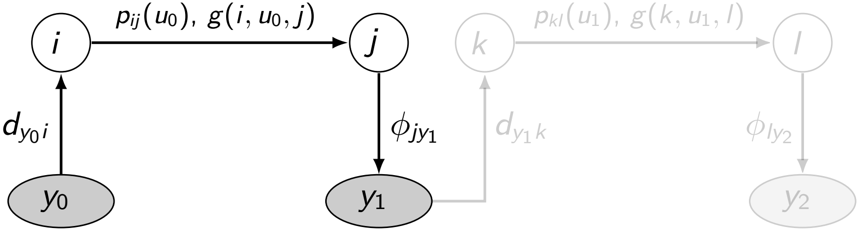
- $\implies$ **Well-defined aggregate problem** with state space $\mathcal{F}$.
  - We obtain the desired dimensionality reduction: $|\mathcal{F}| << |X|$.



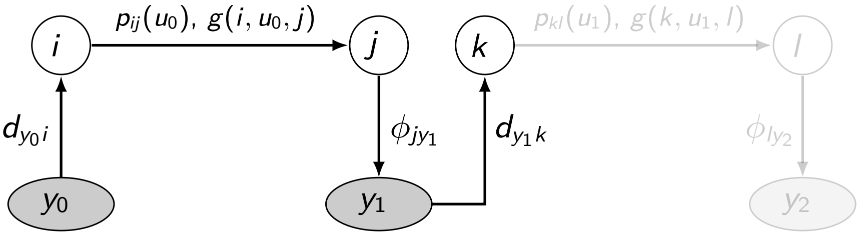
- $\implies$ **Well-defined aggregate problem** with state space $\mathcal{F}$.
  - We obtain the desired dimensionality reduction: $|\mathcal{F}| << |X|$.



- $\implies$ **Well-defined aggregate problem** with state space $\mathcal{F}$.
  - We obtain the desired dimensionality reduction: $|\mathcal{F}| << |X|$.

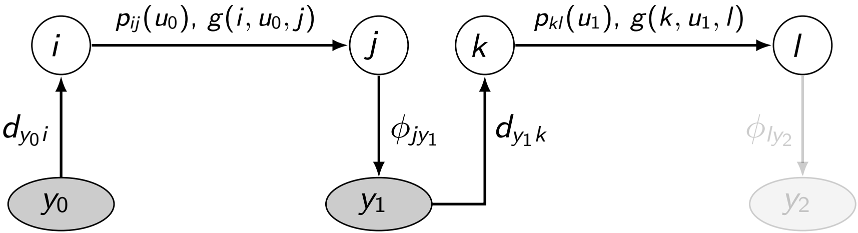


- $\implies$ **Well-defined aggregate problem** with state space $\mathcal{F}$.
  - We obtain the desired dimensionality reduction: $|\mathcal{F}| << |X|$.

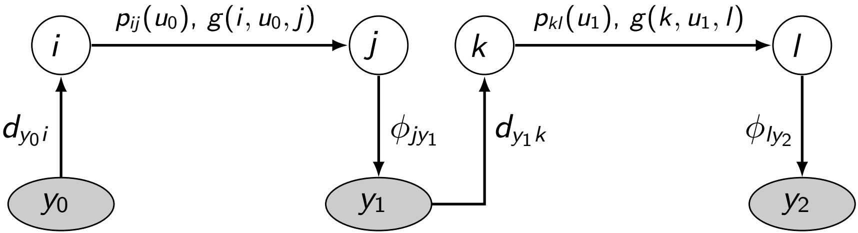


- $\Longrightarrow$ **Well-defined aggregate problem** with state space $\mathcal{F}$.
  - We obtain the desired dimensionality reduction: $|\mathcal{F}| << |X|$.

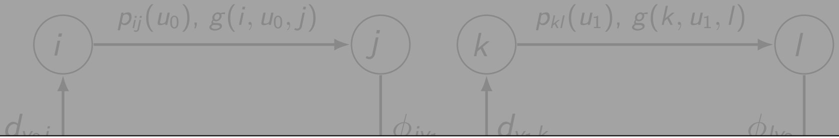
State Aggregation (3/3)



- $\implies$ **Well-defined aggregate problem** with state space $\mathcal{F}$.
  - We obtain the desired dimensionality reduction: $|\mathcal{F}| << |X|$.



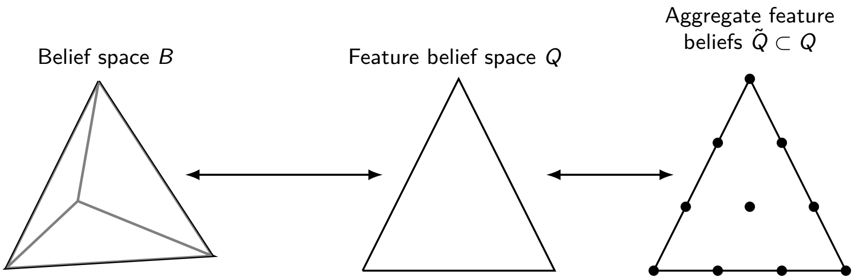
- $\Longrightarrow$ **Well-defined aggregate problem** with state space $\mathcal{F}$.
  - We obtain the desired dimensionality reduction: $|\mathcal{F}| << |X|$.



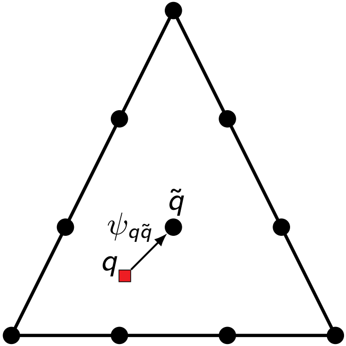
How can we lift this dynamic aggregation system to the **belief space**?

- $\implies$ **Well-defined aggregate problem** with state space $\mathcal{F}$.

Feature Belief Aggregation (1/3)



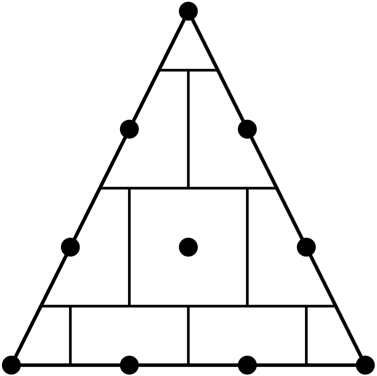
For each **feature belief** $q \in Q$, we associate a *belief aggregation probability* distribution $\{\psi_{q\tilde{q}} \mid \tilde{q} \in \tilde{Q}\}$, where $\psi_{\tilde{q}\tilde{q}} = 1$.



Feature belief space Q

Example (nearest-neighbor aggregation):

$$\psi_{q\tilde{q}} = 1 \iff \tilde{q} = \arg\min_{\tilde{q} \in \tilde{Q}} \|q - \tilde{q}\|.$$

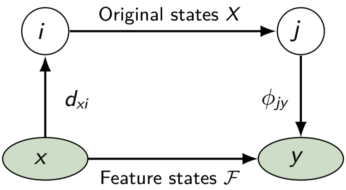


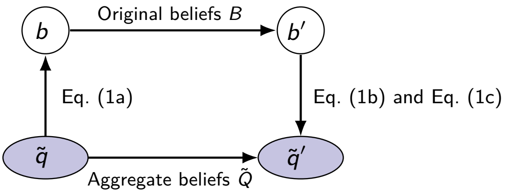
Dynamic belief system:

$$b(i) = \sum_{x \in \mathcal{F}} \tilde{q}(x) d_{xi} \qquad \text{for all } i \in X \quad (b \leftarrow q, \tilde{q}) \tag{1a}$$

$$q(y) = \sum_{j=1}^{n} b(j)\phi_{jy} \qquad \text{for all } y \in \mathcal{F} \quad (b \rightarrow q) \tag{1b}$$

$$\tilde{q} \sim \psi_{q\tilde{q}} \qquad (q \rightarrow \tilde{q}). \tag{1c}$$





POMDP Approximation

- Let $V^{\star}$ be the optimal cost-to-go of the aggregate problem.
- We obtain a cost **approximation of the original POMDP** by

$$\tilde{J}(b) = \sum_{\tilde{q} \in \tilde{Q}} \psi_{\Phi(b)\tilde{q}} V^{\star}(\tilde{q}), \qquad \text{(interpolation formula)},$$

where $\Phi : B \mapsto Q$ is defined as

$$\Phi(b)(y) = \sum_{j=1}^{n} b(j)\phi_{jy} \qquad \text{for all } y \in \mathcal{F}.$$

- Let $V^\star$ be the optimal cost-to-go of the aggregate problem.
- We obtain a cost **approximation of the original POMDP** by

$$\tilde{J}(b) = \sum_{\tilde{q} \in \tilde{Q}} \psi_{\Phi(b)\tilde{q}} V^\star(\tilde{q}), \qquad \text{(interpolation formula)},$$

where $\Phi : B \mapsto Q$ is defined as

$$\Phi(b)(y) = \sum_{j=1}^{n} b(j)\phi_{jy} \qquad \text{for all } y \in \mathcal{F}.$$

- Similarly, a **base policy** μ for the POMDP can be obtained as

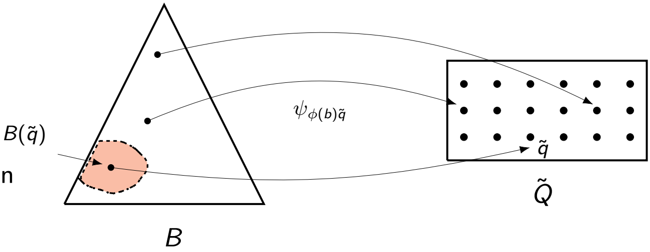
$$\mu(b) \in \arg\min_{u} E_{b'} \left\{ \hat{g}(b,u) + \alpha \tilde{J}(b') \right\} \qquad \text{for all } b \in B.$$

POMDP Approximation

- Let $V^{\star}$ be the optimal cost-to-go of the aggregate problem.
- We obtain a cost **approximation of the original POMDP** by

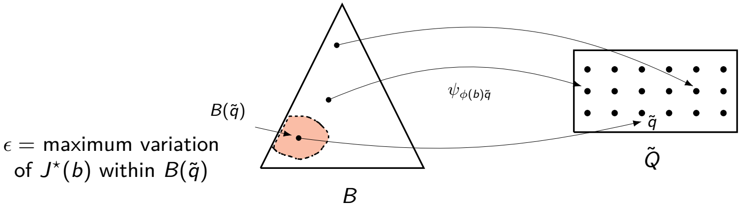
$$\tilde{J}(b) = \sum_{\tilde{q} \in \tilde{Q}} \psi_{\Phi(b)\tilde{q}} V^{\star}(\tilde{q}), \qquad \text{(interpolation formula)}.$$

$\epsilon =$ maximum variation of $J^{\star}(b)$ within $B(\tilde{q})$



- Let $V^\star$ be the optimal cost-to-go of the aggregate problem.
- We obtain a cost **approximation of the original POMDP** by

$$\tilde{J}(b) = \sum_{\tilde{q} \in \tilde{Q}} \psi_{\Phi(b)\tilde{q}} V^\star(\tilde{q}), \qquad \text{(interpolation formula)}.$$

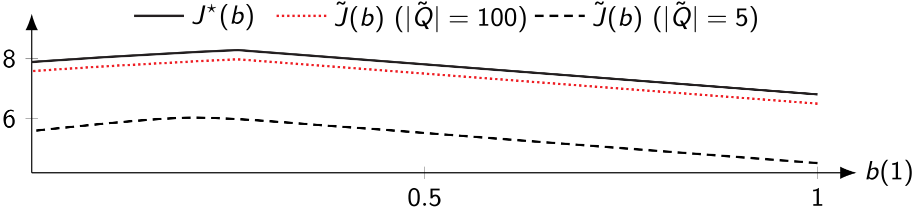


Proposition (Approximation error bound)

Under hard aggregation, the approximation error of $\tilde{J}$ is bounded as

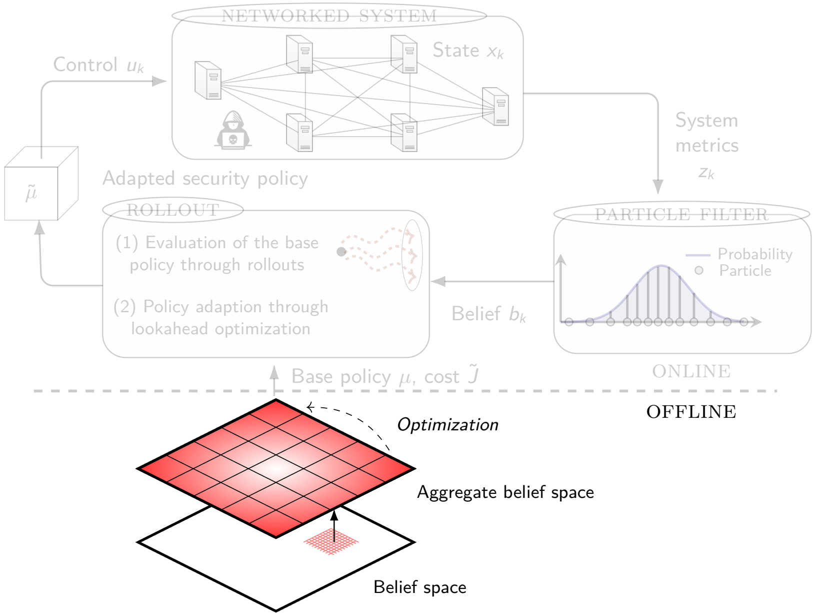
$$|\tilde{J}(b) - J^\star(b)| \leq \frac{\epsilon}{1-\alpha} \qquad \forall b \in B(\tilde{q}), \tilde{q} \in \tilde{Q}.$$

Experimental Illustration of the Error Bound

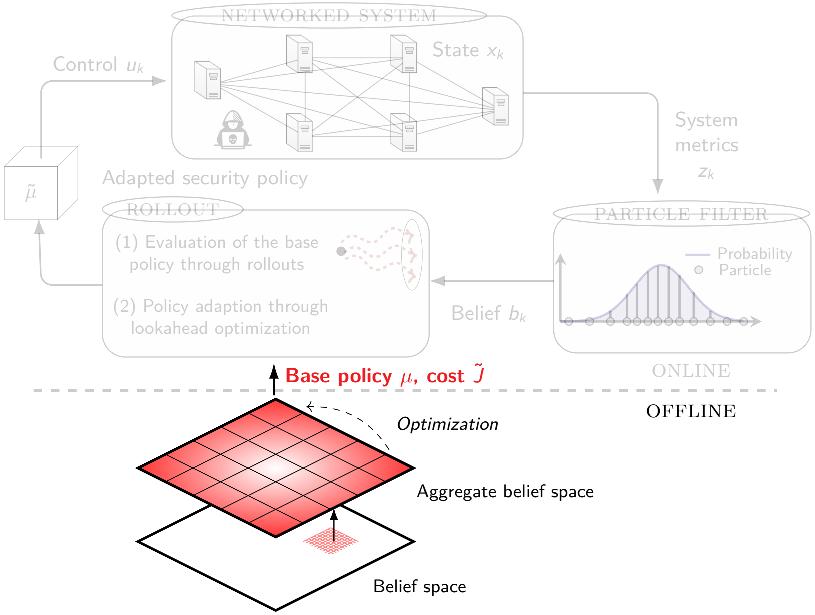


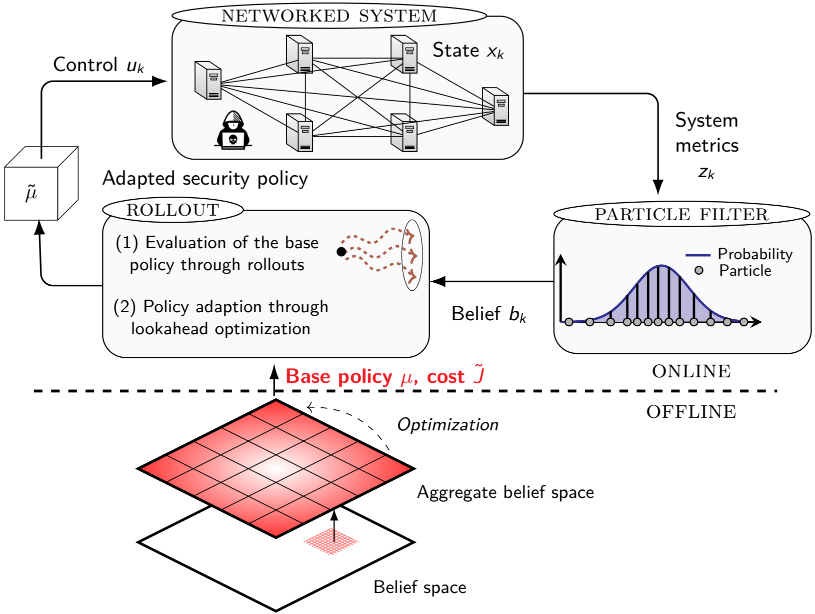
Comparison between the optimal cost-to-go $J^\star$ of the POMDP and the approximate cost-to-go $\tilde{J}$ for varying $|\tilde{Q}|$. The numerical results are based on an example POMDP with $|X| = 2$, $\mathcal{F} = X$, $\tilde{Q}$ defined via grid points, and ψ based on the nearest-neighbor mapping.

The Big Picture



The Big Picture





NETWORKED SYSTEM
State x_k
Control u_k
System metrics
z_k
$\tilde{\mu}$
Adapted security policy
ROLLOUT
(1) Evaluation of the base policy through rollouts
(2) Policy adaption through lookahead optimization
PARTICLE FILTER
Probability
Particle
Belief b_k
ONLINE
Base policy μ, cost $\tilde{J}$
OFFLINE
Optimization
Aggregate belief space
Belief space

Challenge

Exact computation of the belief b has complexity $O(|X|^2)$, which is **intractable for realistic systems**. (In CAGE-2, $|X| \geq 10^{47}$.)

Particle Filtering

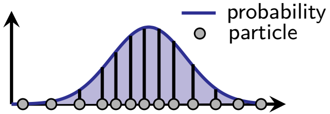
Challenge

Exact computation of the belief b has complexity $O(|X|^2)$, which is **intractable for realistic systems**. (In CAGE-2, $|X| \geq 10^{47}$.)

To manage this complexity, we use a **particle filter** to estimate b as

$$\widehat{b}_k(x_k) = \frac{1}{M}\sum_{i=1}^{M} \mathbb{1}_{x_k = \widehat{x}_k^{(i)}}, \tag{2}$$

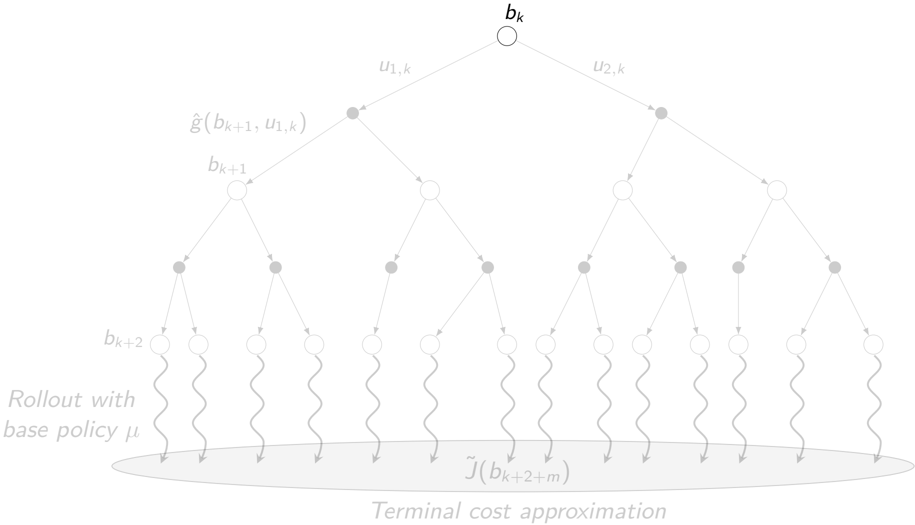
where $\{\widehat{x}_k^1, \ldots, \widehat{x}_k^M\}$ are particles sampled with probability proportional to $p(z_k \mid \widehat{x}_k^i, u_{k-1})$.



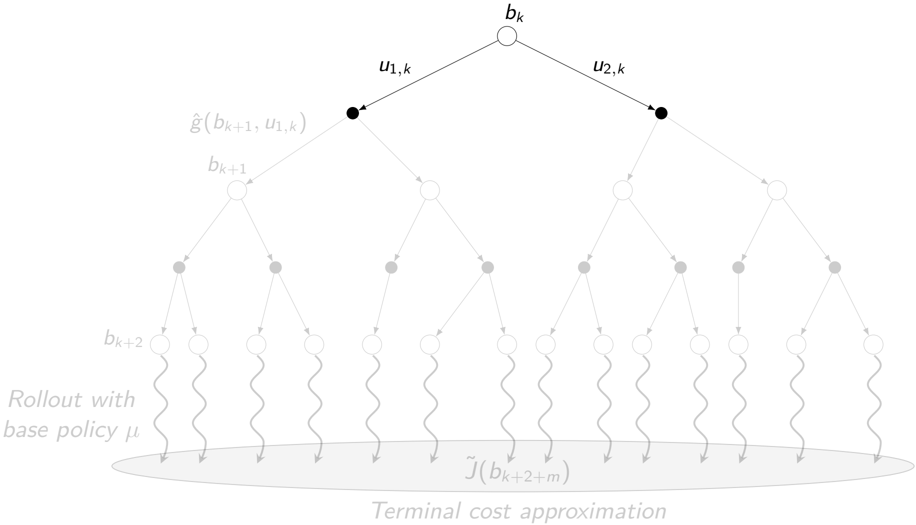
The complexity of Eq. (2) **can be adjusted to available compute resources** by tuning M. Strong law of large numbers implies asymptotic consistency,

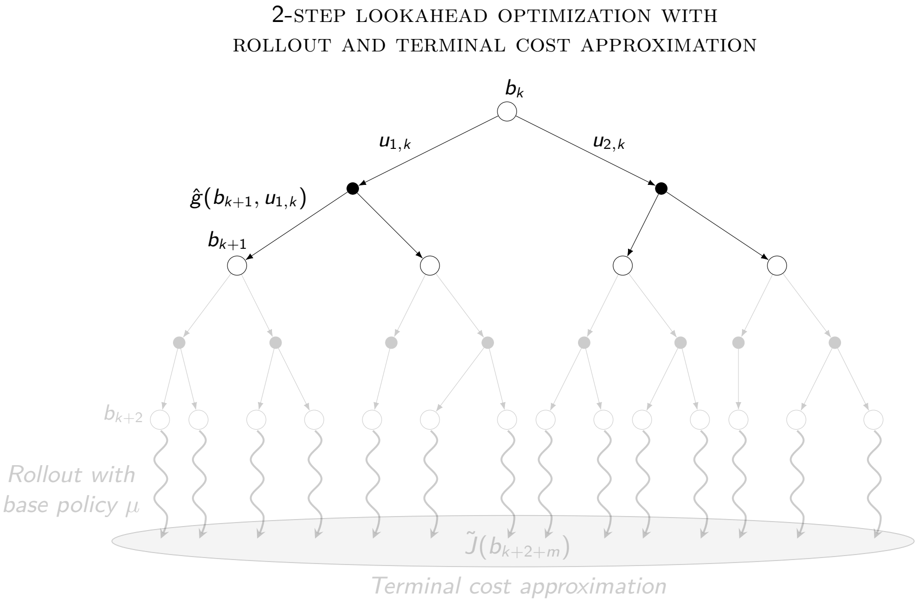
$$\lim_{M\to\infty} \widehat{b} = b.$$

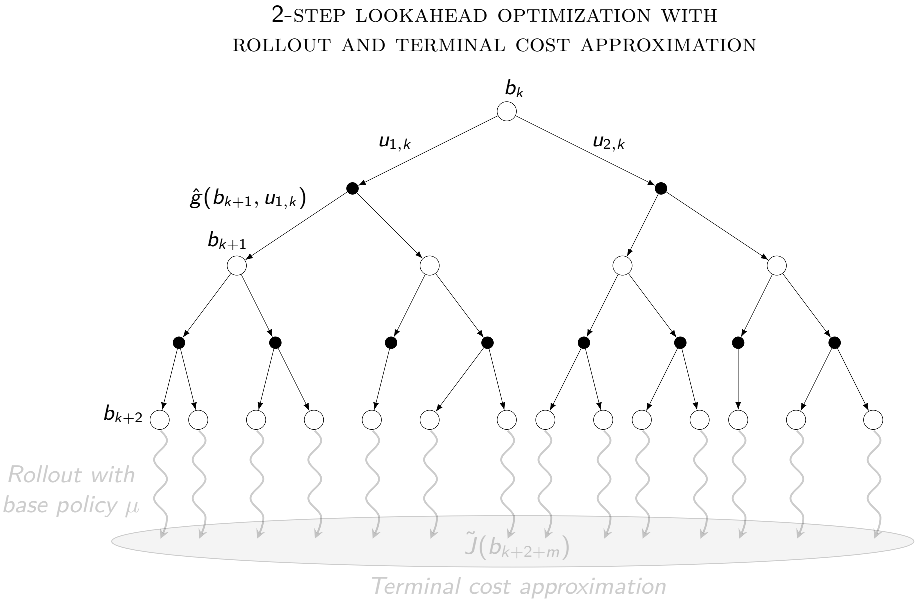
2-step lookahead optimization with rollout and terminal cost approximation



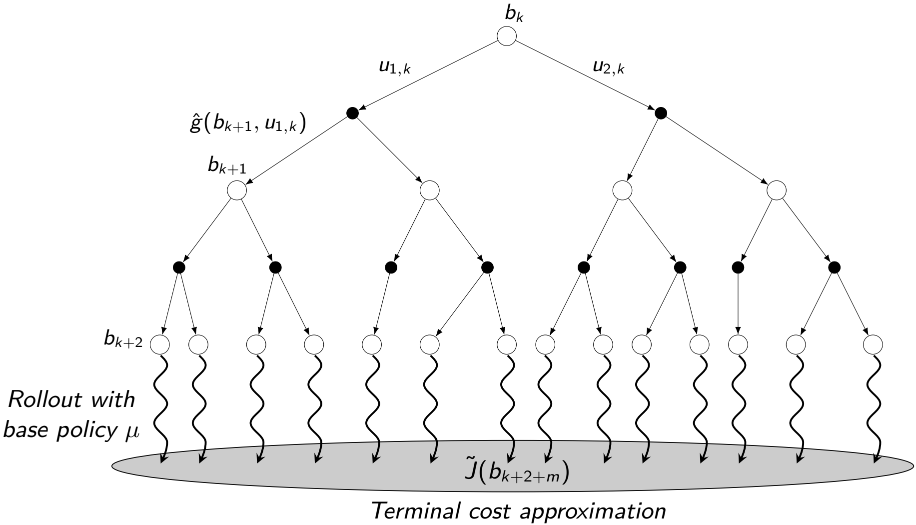
2-STEP LOOKAHEAD OPTIMIZATION WITH ROLLOUT AND TERMINAL COST APPROXIMATION



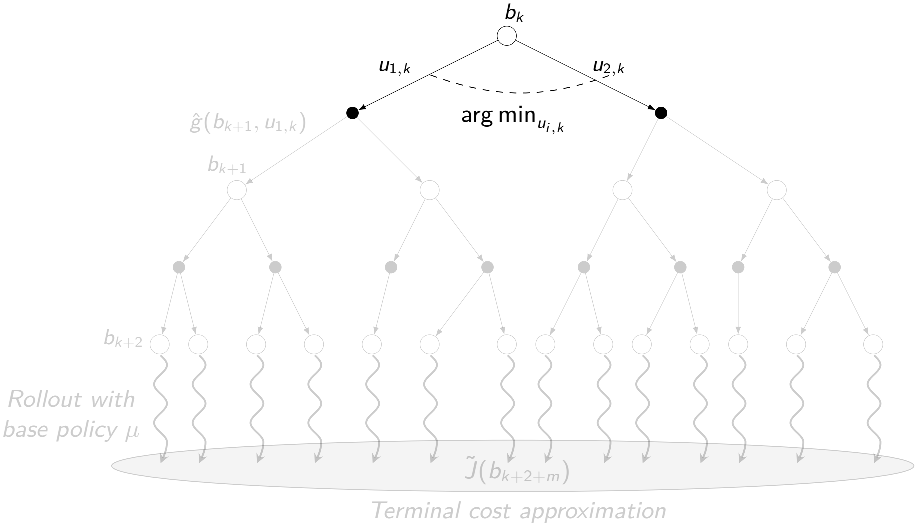

2-STEP LOOKAHEAD OPTIMIZATION WITH ROLLOUT AND TERMINAL COST APPROXIMATION
b_k
$u_{1,k}$
$u_{2,k}$
$\hat{g}(b_{k+1}, u_{1,k})$
b_{k+1}
b_{k+2}
Rollout with base policy μ
$\tilde{J}(b_{k+2+m})$
Terminal cost approximation


2-STEP LOOKAHEAD OPTIMIZATION WITH
ROLLOUT AND TERMINAL COST APPROXIMATION
b_k
$u_{1,k}$
$u_{2,k}$
$\hat{g}(b_{k+1}, u_{1,k})$
b_{k+1}
b_{k+2}
Rollout with
base policy μ
$\tilde{J}(b_{k+2+m})$
Terminal cost approximation

2-STEP LOOKAHEAD OPTIMIZATION WITH ROLLOUT AND TERMINAL COST APPROXIMATION



2-STEP LOOKAHEAD OPTIMIZATION WITH ROLLOUT AND TERMINAL COST APPROXIMATION



Lookahead optimization

We transform the base policy to an **adapted *rollout policy*** $\tilde{\mu}$ via ℓ-step lookahead

$$\tilde{\mu}(b_k) \in \underset{u_k,\mu_{k+1},\ldots,\mu_{k+\ell-1}}{\arg\min} E_{z_{k+1},\ldots,z_{k+\ell}}\left\{\hat{g}(b_k,u_k) + \sum_{j=k+1}^{t+\ell-1} \alpha^{j-k}\hat{g}(b_j,\mu_j(b_j)) + \alpha^{\ell}\tilde{J}_{\mu}(b_{k+\ell})\right\}.$$

Rollout

The cost-to-go in the lookahead minimization is **estimated via** m**-step rollout** with the base policy μ and **terminal cost approximation** $\tilde{J}$:

$$\tilde{J}_{\mu}(b_k) = \frac{1}{L}\sum_{j=1}^{L}\sum_{l=k}^{k+m}\alpha^{l-k}\hat{g}(b_l^j,\mu(b_l^j)) + \alpha^{m-k}\tilde{J}(b_{k+m}^j).$$

Proposition (Bertsekas, 2019)

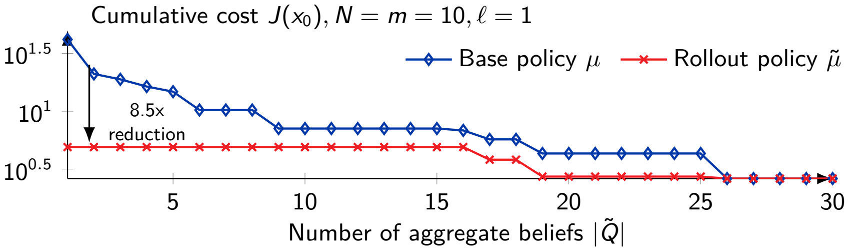
1. *If the rollout policy evaluation is exact, i.e., if $\tilde{J}_{\mu} = J_{\mu}$, then the rollout policy is guaranteed to improve the base policy.*
2. *The **sub-optimality of the rollout policy $\tilde{\mu}$ is bounded** as*

$$\|J_{\tilde{\mu}} - J^{\star}\| \leq \frac{2\alpha^{\ell}}{1-\alpha}\|\tilde{J}_{\mu} - J^{\star}\|.$$

Proposition (Bertsekas, 2019)

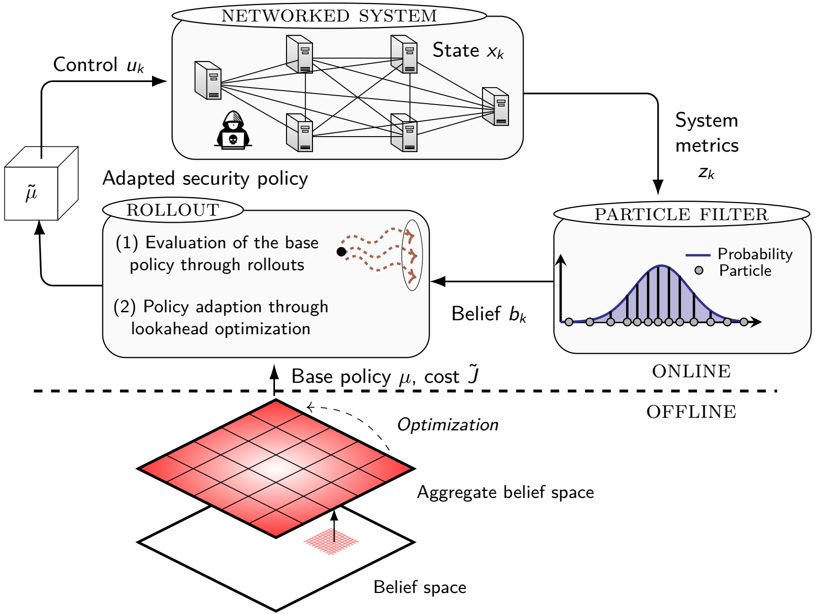
1. *If the rollout policy evaluation is exact, i.e., if $\tilde{J}_\mu = J_\mu$, then the rollout policy is guaranteed to improve the base policy.*
2. *The **sub-optimality of the rollout policy $\tilde{\mu}$ is bounded** as*

$$\|J_{\tilde{\mu}} - J^\star\| \leq \frac{2\alpha^\ell}{1-\alpha}\|\tilde{J}_\mu - J^\star\|.$$

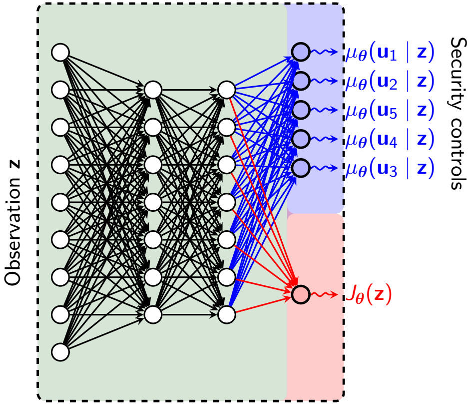


Performance of rollout for an example POMDP.

Framework Summary



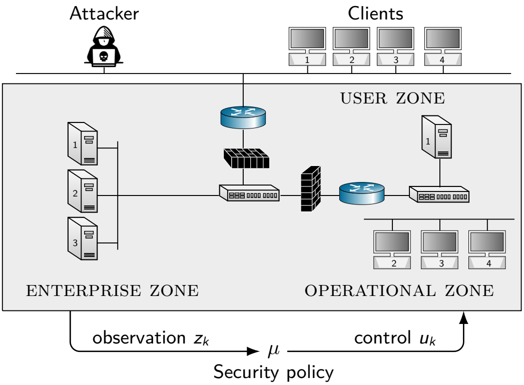
- Standard benchmark for comparing methods: **CAGE-2**.
  - *Problem: find an effective security policy to network intrusions.*
  - POMDP with over 10^{47} states and 10^{25} observations.
  - **Leaderboard** with more than 35 different methods.
- Current state-of-the-art: deep reinforcement learning (variants of PPO).



Parameterized security policy μ_θ

Instantiation of Our Framework for CAGE-2

- We define $\tilde{Q}$ based on intuition, where $|\tilde{Q}| = 427500$.
- We define ψ based on the nearest-neighbor mapping.
- We solve the aggregate problem using **value iteration.**
  - This gives us the base policy μ and cost $\tilde{J}$.
- We use $\ell = 2$ **lookahead steps** and $M = 50$ particles.



The CAGE-2 system.

Experimental Results (1/2)

Method	*Offline/Online compute (min/s)*	*State estimation*	*Cost*
μ	8.5/0.01	PARTICLE FILTER	15.19 ± 0.82
PPO	1000/0.01	LATEST OBSERVATION	280 ± 114
PPO	1000/0.01	PARTICLE FILTER	119 ± 58
PPG	1000/0.01	LATEST OBSERVATION	338 ± 147
PPG	1000/0.01	PARTICLE FILTER	299 ± 108
DQN	1000/0.01	LATEST OBSERVATION	479 ± 267
DQN	1000/0.01	PARTICLE FILTER	462 ± 244
CARDIFF	300/0.01	LATEST OBSERVATION	$\mathbf{13.69 \pm 0.53}$
CARDIFF	300/0.01	PARTICLE FILTER	$\mathbf{13.31 \pm 0.87}$
POMCP	0/15	PARTICLE FILTER	30.88 ± 1.41
POMCP	0/30	PARTICLE FILTER	29.51 ± 2.00
OURS $(m = 0)$	8.5/0.95	PARTICLE FILTER	$\mathbf{13.24 \pm 0.57}$
OURS $(m = 10)$	8.5/8.29	PARTICLE FILTER	$\mathbf{13.23 \pm 0.62}$
OURS $(m = 20)$	8.5/14.80	PARTICLE FILTER	$\mathbf{13.23 \pm 0.57}$

Numbers indicate the mean and the standard deviation from 1000 evaluations.

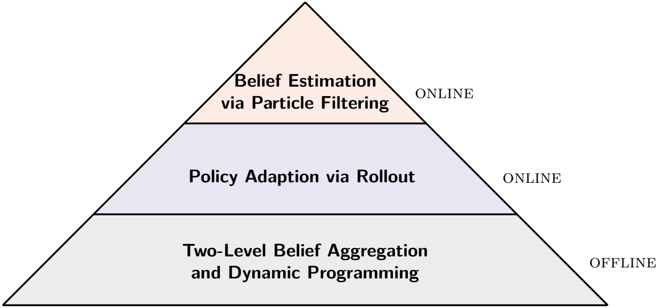
Experimental Results (2/2)

Method	*Offline/Online compute (min/s)*	*State estimation*	*Cost*
μ	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
CARDIFF	300/0.01	LATEST OBSERVATION	57.45 ± 2.44
CARDIFF	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
OURS $(m = 0)$	8.5/0.95	PARTICLE FILTER	51.87 ± 1.42
OURS $(m = 10)$	8.5/8.29	PARTICLE FILTER	$\mathbf{38.81 \pm 1.68}$
OURS $(m = 20)$	8.5/14.80	PARTICLE FILTER	$\mathbf{37.89 \pm 1.54}$

Numbers indicate the mean and the standard deviation from 1000 evaluations.

Conclusion

- We present a **scalable framework** for computing adaptive security policies, which has formal performance guarantees and achieves state-of-the-art performance.
- It consists of three components:



Work in progress! ⌛

Theoretical and experimental details will be available in preprints soon.
Source code is available at:

- `https://github.com/Limmen/rollout_aggregation`; and
- `https://github.com/Limmen/csle`