



# Optimal Decision-Making under Parameter Uncertainty

STELLA KAPODISTRIA

Summer School Sequential Decision Making 2025

Department of Mathematics and Computer Science

# Real-time data-driven maintenance logistics

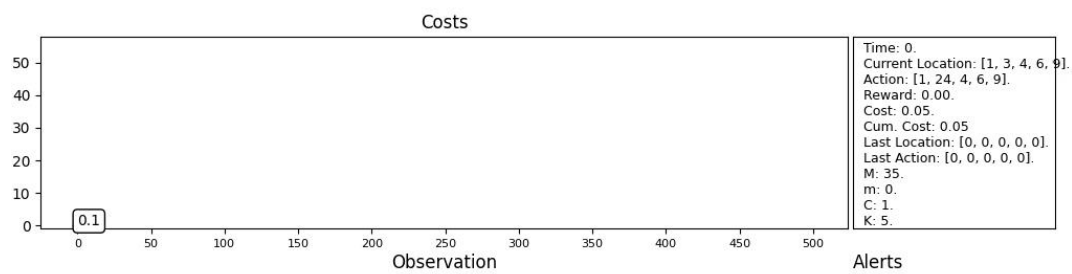
- Jointly with Joachim Arts, Collin Drent, Melvin Drent, Willem van Jaarsvelt, Peter Verleijdsdonk
- Companies involved: ASML, Philips, NS, Fokker
- Problem setting:
  - High-tech systems in a network
  - Real-time condition information
  - Failures and unavailability are costly
  - Maintenance resources are shared over the network
- Question:
  - When to do maintenance?
  - How to optimally dispatch real-time the maintenance resources?



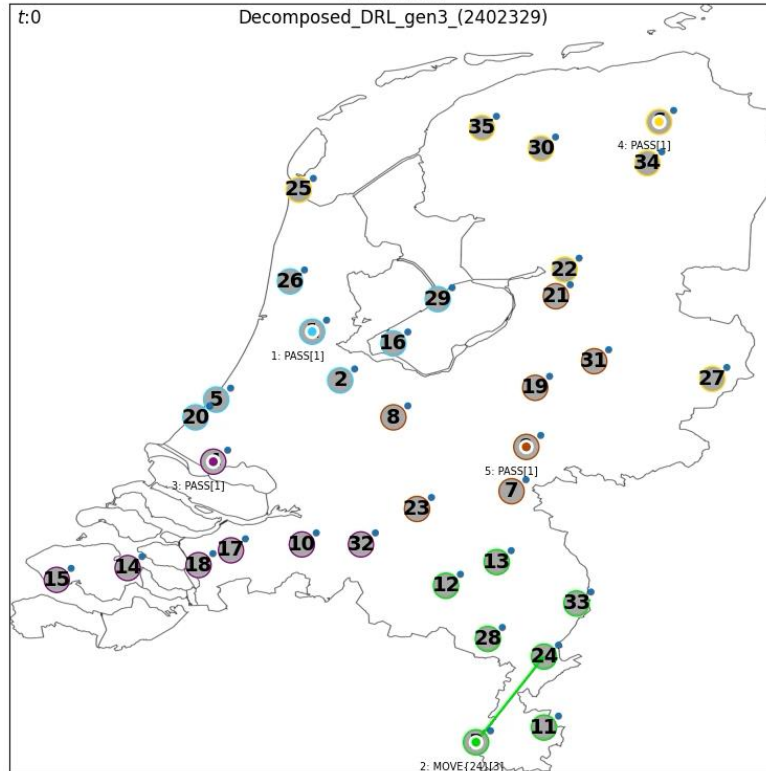
## Real-time data-driven maintenance logistics

- Why is it challenging?
  - Highly dynamic, stochastic environment
  - Under an oracle, the problem reduces to the traveling multi-maintainer problem with response-time dependent costs
- Our A(P)I success:
  - Our solution accounts for real-time information
  - Our solution is interpretable and near-optimal
  - Our solution improves on heuristic state-of-the-art dispatching algorithms





Alerts



# Sketching the idea

Step 1: Solve the *single* maintainer problem for *homogeneous* machines on a *network*

<https://doi.org/10.1016/j.ejor.2022.06.044>.

Step 2: Solve the *multi*-maintainer problem for *homogeneous* machines on a *network*

<https://doi.org/10.1016/j.ejor.2024.05.049>.

<https://doi.org/10.1016/j.ejor.2025.01.026>.

Step 3: Solve the *single* maintainer problem for *heterogeneous* machines on a *network*

<https://pubsonline.informs.org/doi/10.1287/msom.2022.1149>

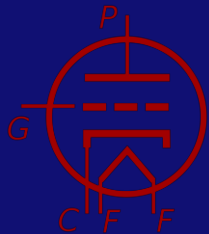
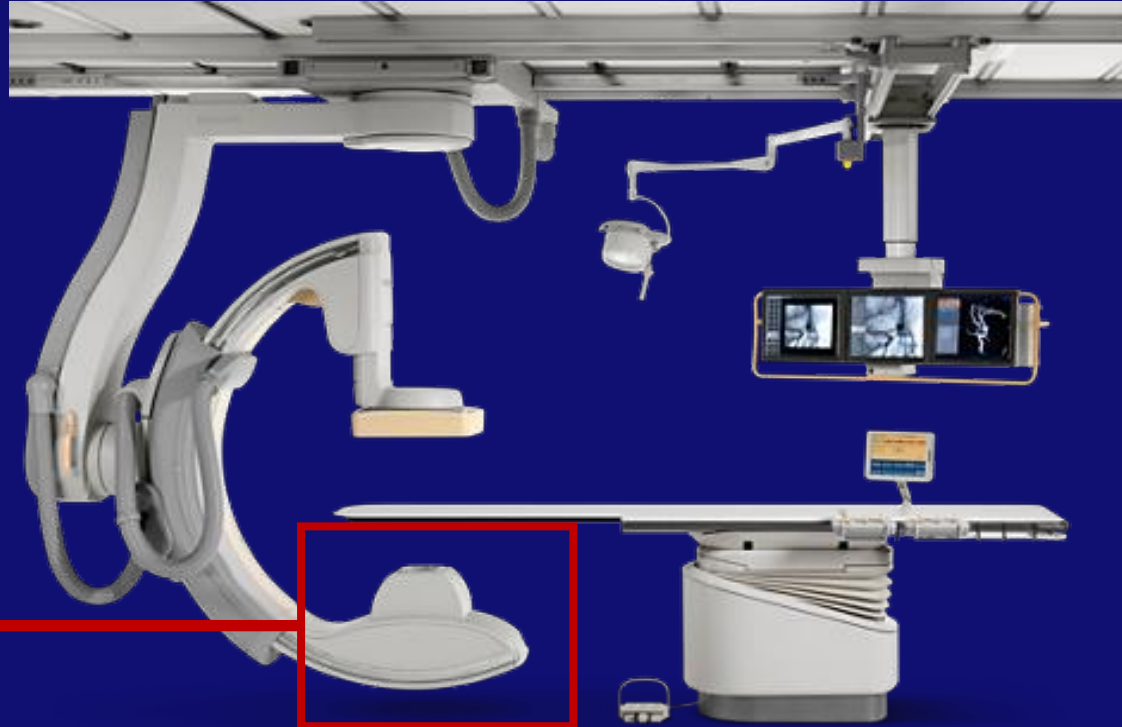
[https://pure.tue.nl/ws/portalfiles/portal/357485182/20250527\\_Verleijsdonk\\_hf.pdf](https://pure.tue.nl/ws/portalfiles/portal/357485182/20250527_Verleijsdonk_hf.pdf) (Chapter 4)

Step 4: Solve the *multi*-maintainer problem for *heterogeneous* machines on a *network*

open problem!

# The challenges of the single component case

iXR: filament

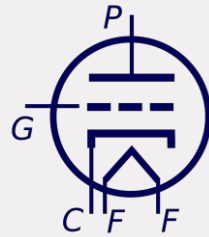


Physical model

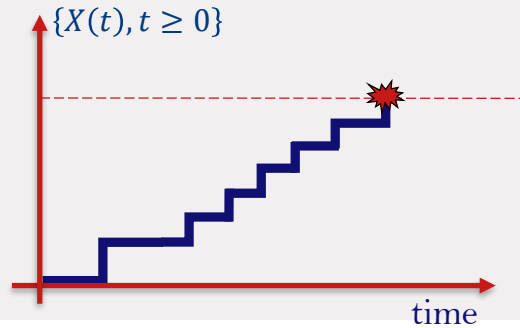
Optimization

Cost optimal  
maintenance  
planning

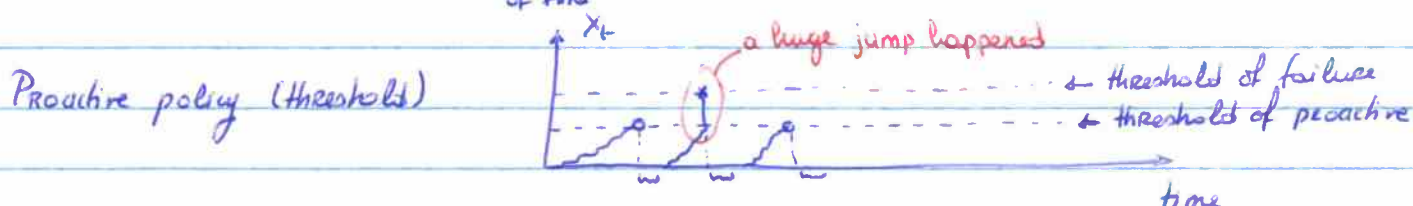
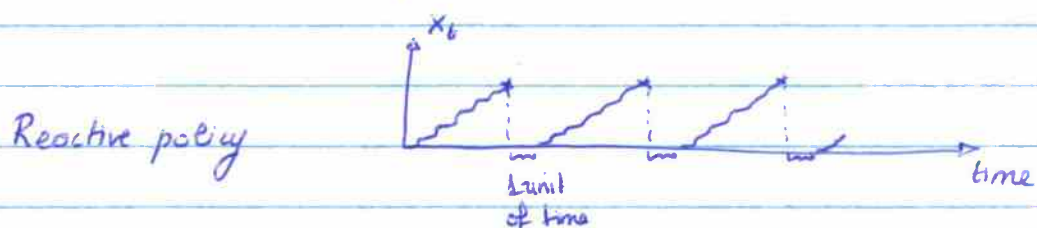
- Population of components



- Condition data to failure



Single engineer - Single machine (known parameters)



Model assumptions

-  $\{X_t : t \geq 0\}$  Compound Poisson process  $(X_t = \sum_{i=1}^{N_t} Y_i)$

- actions  $\{0 \equiv \text{Do nothing}, 1 \equiv \text{Do maintenance}\}$   
 $\rightarrow$  only allowed if  $X_t < f$ , where  $f$  is the failure threshold

- costs = 
$$\begin{cases} c_M & \text{if } X_t \geq f \text{ and } a=1 \quad (c_M > c_P) \\ c_P & \text{if } X_t < f \text{ and } a=1 \\ 0 & \text{if } a=0 \text{ and } X_t < f \end{cases}$$

- replacement with an as-good-as-new machine

- replacement takes one unit of time

Bellman equations

$$V(x, t) = \begin{cases} \min \left\{ 0 + \int_y \sum_{y} p_{xy} V(y, t+1), c_P + \int V(0, t+1) \right\}, & x < f \\ c_M + \int V(0, t+1), & x \geq f \end{cases}$$

The optimal policy is stationary ~~(t)~~

Theorem

The optimal policy is a threshold policy



Proof with contradiction:

Let us assume that the "smallest" state in which we do maintenance is  $x^*$ .

- If  $x^* = j$ , then  $c_{PM} \neq c_{CM}$ , but this contradicts the model assumption  $c_{PM} < c_{CM}$ .

- If  $x^* < j$ : Assume that there exists an  $\hat{x}$  with  $\hat{x} > x^*$  and in state  $\hat{x}$ , it is optimal to do nothing. Then from the Bellman equations

$$V(x^*) = c_{PM} + \delta V(0) = \min \{ 0 + \delta \sum p_{x^*y} V(y), c_{PM} + \delta V(0) \} \quad (1)$$

$$V(\hat{x}) = 0 + \delta \sum p_{\hat{x}y} V(y) = \min \{ 0 + \delta \sum p_{\hat{x}y} V(y), c_{PM} + \delta V(0) \} \quad (2)$$

$$\Rightarrow c_{PM} + \delta V(0) \stackrel{(1)}{<} \delta \sum p_{x^*y} V(y) \text{ and } c_{PM} + \delta V(0) \stackrel{(2)}{>} \delta \sum p_{\hat{x}y} V(y)$$

$$\Rightarrow \sum p_{\hat{x}y} V(y) < \sum p_{x^*y} V(y) \text{ for } \hat{x} > x^* \quad (3)$$

$$\text{However } \sum p_{\hat{x}y} V(y) = \mathbb{E} V(\hat{x} + \text{Random increase of the CPP starting from } \hat{x}) \stackrel{(3)}{<} \mathbb{E} V(x^* + \text{Random increase of CPP starting from } x^*)$$

We know that  $\hat{x} + Y \underset{st}{>} x^* + Y$  (in the usual stochastic order)  $\forall Y$

We can recursively prove that  $V(\cdot)$  is increasing, but this contradicts (3) which can only occur if  $V(\cdot)$  is decreasing.

Intuitively  $V(\cdot)$  is increasing due to the higher we start (in  $x$ ) the most likely it is to do maintenance and to pay a higher cost.

One can show that  $V(\cdot)$  is increasing using the value iteration algorithm. Namely, say  $V^{(0)}(x) = 0$  then  $V^{(1)}(x)$  is increasing.

Say  $V^{(n)}(x)$  is increasing in  $x$ . Then  $V^{(n+1)}(x) = \min \left\{ 0 + \delta \mathbb{E} V^{(n)}(x+Y), c_{PM} + \delta V^{(n)}(0) \right\}$

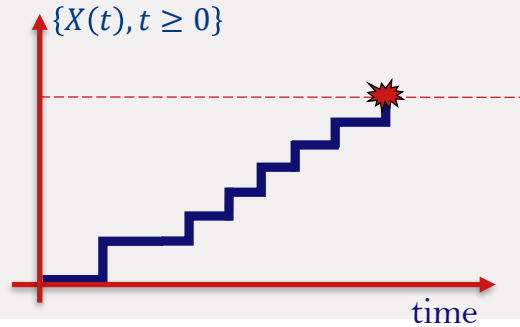
Note that  $V^{(n)}(\cdot)$  is increasing and from the usual stochastic order  $\mathbb{E} V^{(n)}(x+Y)$  is also increasing in  $x$ . As such  $V^{(n+1)}(x)$  is increasing.  $\square$

Physical model

Optimization

Cost optimal  
maintenance  
planning

- Population of components

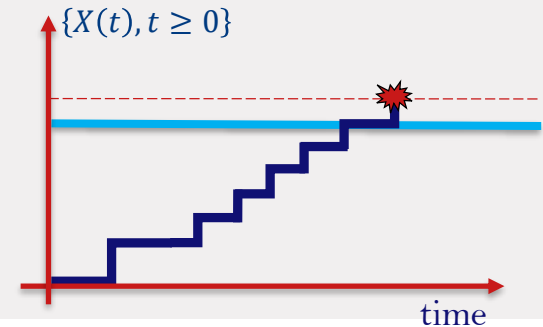


- Condition data to failure

When to do maintenance?

condition ( $\theta$ )

Known from  
historical data



# When to do maintenance?

Drent, Drent, Arts and Kapodistria: *Integrated Learning and Decision Making*

---

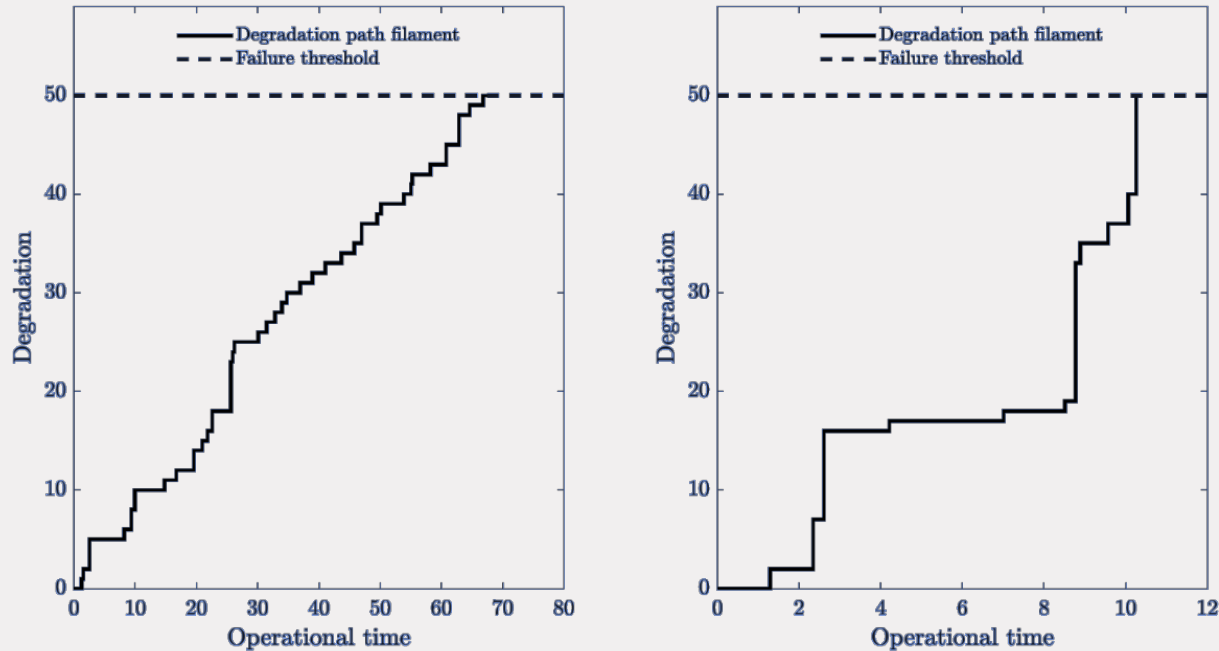


Figure 1 Two historical IXR filament degradation paths.

Data with physical  
model properties

Integration

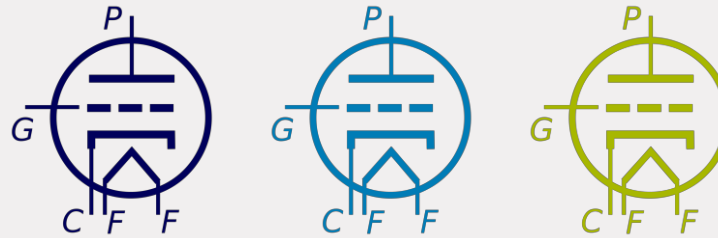
Physical model

Optimization

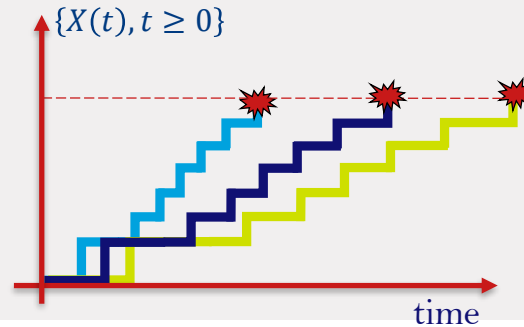
Optimization

Cost optimal  
maintenance  
planning

- Heterogeneous population

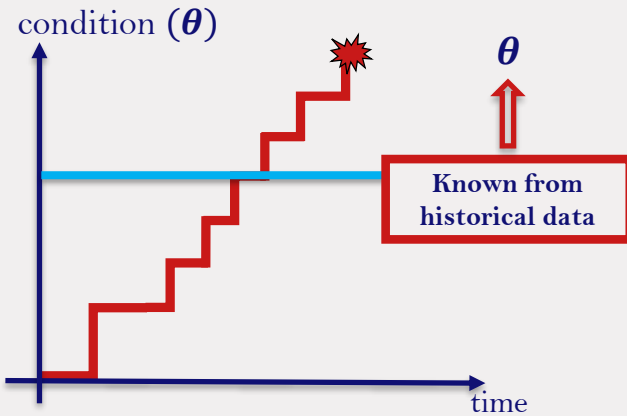


- Condition data to failure

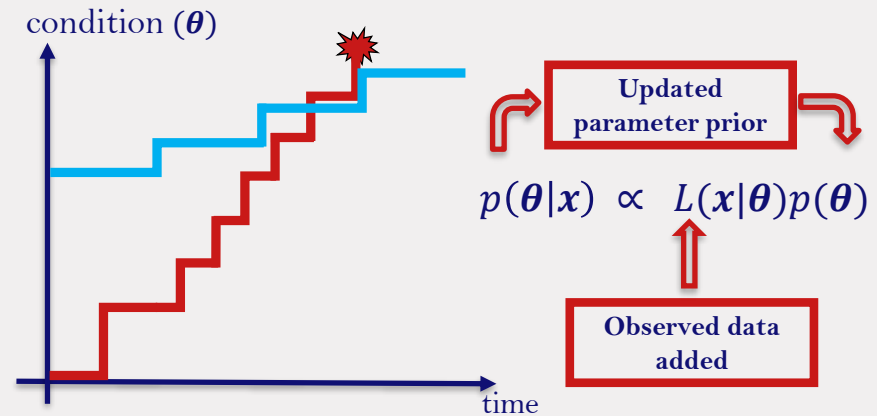


# How to account for a heterogeneous population in maintenance planning?

Traditional condition replacement policy



Integrated condition replacement policy



# Single engineer - Single machine (unknown parameters)

Reactive policy



We assume that the time between jumps is  $\Gamma_{\text{op}}$  but with a parameter that is sampled upon the installation of the machine ( $\uparrow$ ) from a random variable  $\Lambda$ . Similarly for the jump size of the Compound Poisson process, say  $\Phi$ .

Example  $\Lambda \sim \text{Gamma}(\alpha, \beta)$

At time  $t=0$ , we have not seen any realisation so  $f_{\Lambda}^0(\lambda) = \frac{e^{-\lambda\beta} \lambda^{\alpha-1} \beta^{\alpha}}{\Gamma(\alpha)}$

At time  $t=1$ , we see one realisation say  $N(t)=n$  ( $N(t)$  is the PP)  
Then

$$f_{\Lambda}^1(\lambda) = \mathbb{P}(\Lambda = \lambda \mid N(t)=n, \alpha, \beta)$$

$$= \frac{\mathbb{P}(\Lambda = \lambda, N(t)=n \mid \alpha, \beta)}{\mathbb{P}(N(t)=n \mid \alpha, \beta)} = \frac{\mathbb{P}(N(t)=n \mid \Lambda = \lambda) \cancel{\mathbb{P}(\Lambda = \lambda \mid \alpha, \beta)}}{\int_{\Lambda} \mathbb{P}(N(t)=n \mid \Lambda = \lambda) f_{\Lambda}^0(\lambda) d\lambda}$$

$$\propto \frac{e^{-\lambda t} (\lambda t)^n / n! \cdot e^{-\lambda \beta} \lambda^{\alpha-1} \beta^{\alpha} / \Gamma(\alpha)}{\text{normalization}}$$

$$\propto e^{-\lambda(t+\beta)} \lambda^{n+\alpha-1}$$

So  $f_{\Lambda}^1(\lambda)$  is Gamma ( $\alpha+n$ ,  $\beta+t$ ) with  $t=1$

Example  $\Phi \sim \text{Beta}(a, b)$

At time  $t=0$ ,  $f_{\Phi}^0(p) = \frac{p^{a-1} (1-p)^{b-1}}{B(a, b)}$

At time  $t=1$ , we see a jump realisation say  $Y=y$  (assume  $Y$  to be geometric for this example)



(4)

Then

$$f_{\Phi}^{\perp}(p) = \mathbb{P}(\Phi=p \mid Y=y, a, b)$$

$$\propto \mathbb{P}(Y=y \mid \Phi=p) \mathbb{P}(\Phi=p \mid a, b)$$

$$\propto p(1-p)^y \frac{p^{a-1}(1-p)^{b-1}}{B(a, b)}$$

$$\propto p^{a-1} (1-p)^{y+b-1}$$

So  $f_{\Phi}^{\perp}(p)$  is  $\text{Beta}(\underline{a+1}, \underline{y+b})$

Of course in the case of the Compound Poisson process we see both a number of events say  $N(t)=n$  and for each event a jump realisation  $Y_1=y_1, Y_2=y_2, \dots, Y_n=y_n$ . Then the joint Bayes updating rule for  $\Lambda$  and  $\Phi$  can be proved to be

$$\text{Given that at time } t=0, f_{\Lambda}^0(\lambda) = \frac{e^{-\lambda\beta} \lambda^{\alpha-1} \beta^{\alpha}}{\Gamma(\alpha)} \text{ and } f_{\Phi}^0(p) = \frac{p^{a-1}(1-p)^{b-1}}{B(a, b)}$$

At time  $t=1$ , given  $N(t)=n$  and  $\sum_{i=1}^n Y_i = \sum y_i$  ( $\sum Y_i$  becomes negative binomial in distribution as the sum of independent geometrics)

$$f_{\Lambda, \Phi}^{\perp}(\lambda, p) = f_{\Lambda}^{\perp}(\lambda) \cdot f_{\Phi}^{\perp}(p) \quad (*)$$

$$\text{with } f_{\Lambda}^{\perp}(\lambda) \text{ Gamma}(\alpha+n, \beta+1) \text{ and } f_{\Phi}^{\perp}(p) \text{ Beta}(a+n, b+\sum_{i=1}^n y_i).$$

Although  $\Lambda$  and  $\Phi$  are not seemingly independent, due to the same data that drives the update, equation (\*) shows that they are in reality independent.

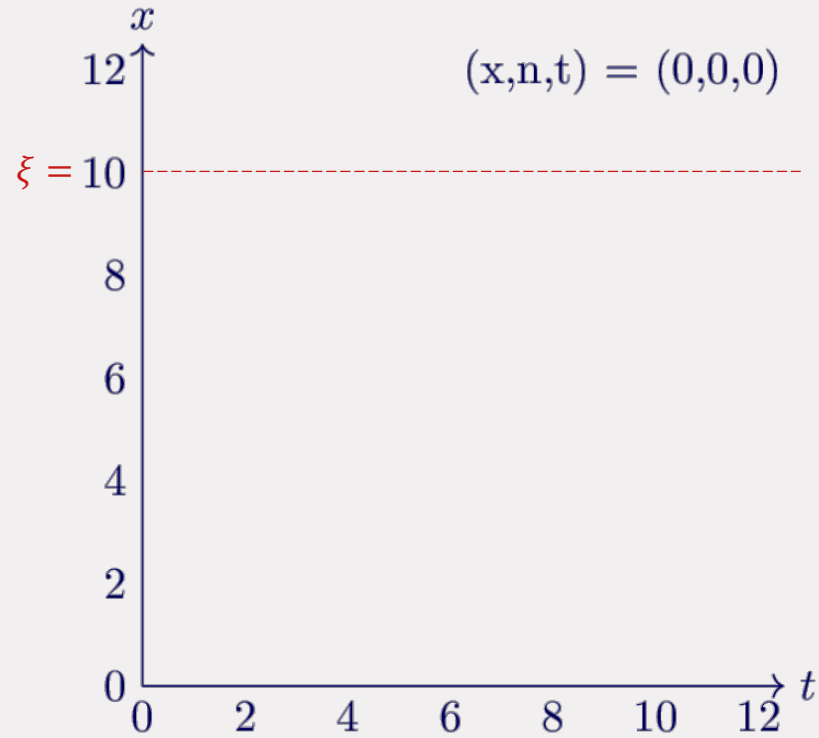
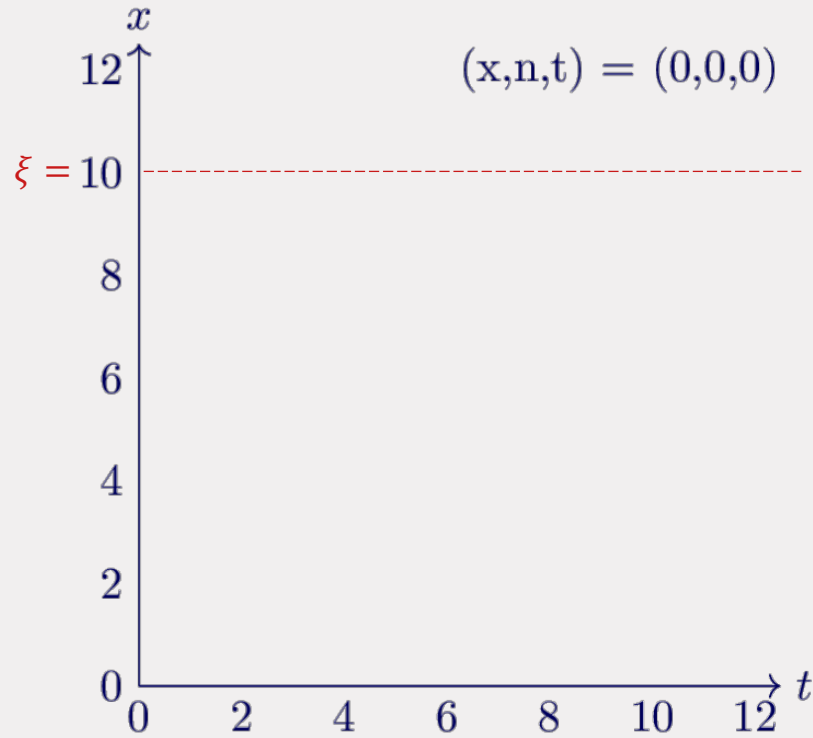
This is due to 1) the prior belief choice

2) the fact that we observe in one time unit ( $t=1$ ) both the number of events ( $N(t)=n$ ) and the total jump ( $\sum_{i=1}^n Y_i = \sum y_i$ )

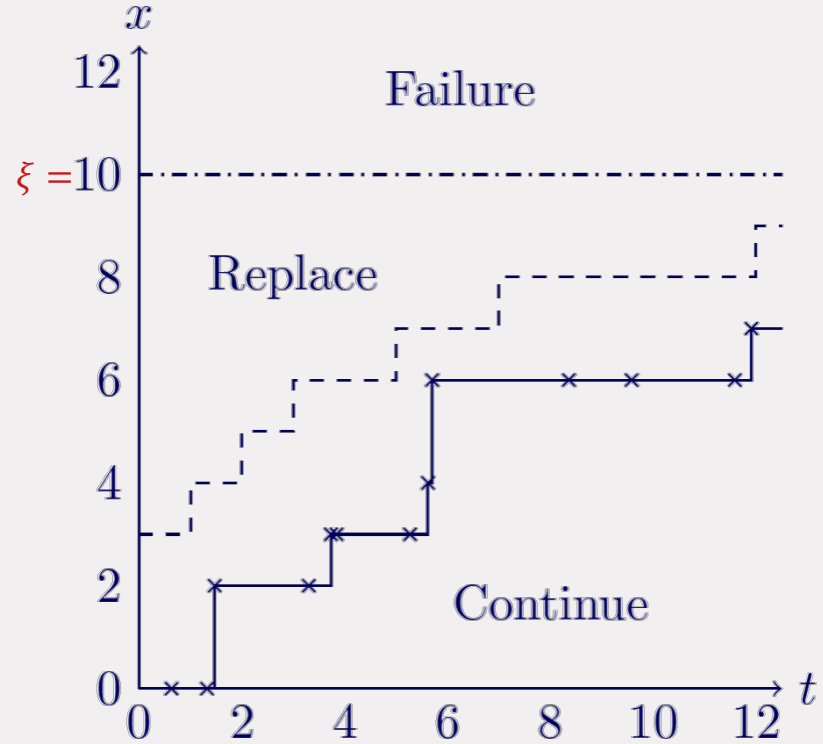
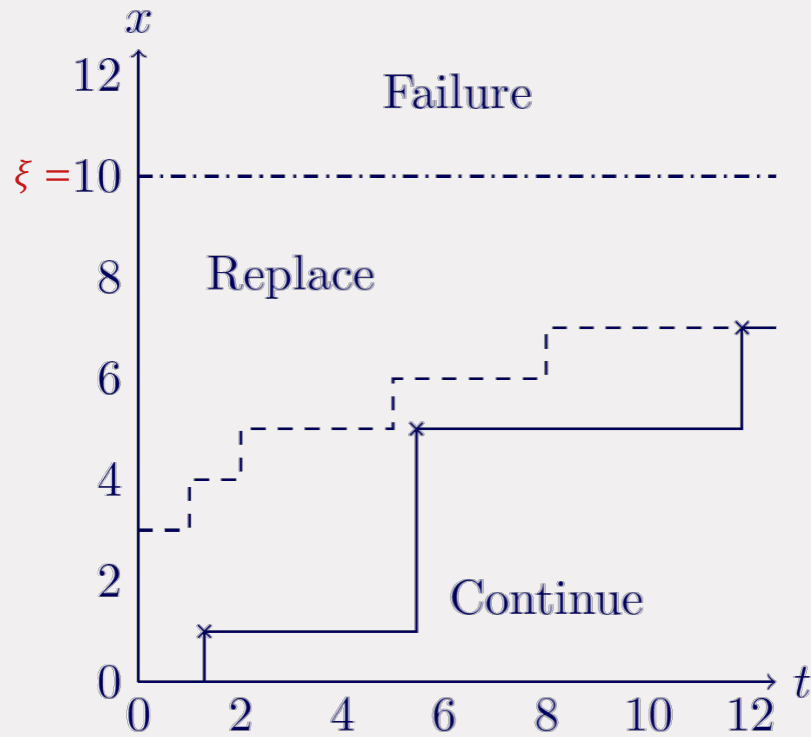




# Optimal policy



# Optimal policy



# Building blocks of the theory

Let  $\{X_t, t \geq 0\}$  denote the stochastic process that drives the condition with unknown parameters  $\theta$ .

Consider  $V(h_t, s_t, t)$  with  $h_t = ((s_0, a_1), (s_1, a_2), \dots, (s_{t-1}, a_t))$

**Step 1:** State space collapse  $V(h_t, s_t, t) \mapsto V(\widehat{\theta}_t, s_t, t)$  with  $\widehat{\theta}_t$  denoting the efficient statistic

E.g., if  $\{X_t, t \geq 0\}$  is CPP then  $(h_t, s_t, t) \mapsto (n, x_t, t)$

**Step 2:** Stochastic ordering of  $\{X_t, t \geq 0\}$  with respect to  $(\widehat{\theta}_t, s_t, t)$

E.g., if  $\{X_t, t \geq 0\}$  is CPP, then non-increasing in  $t$  and non-decreasing in  $x$

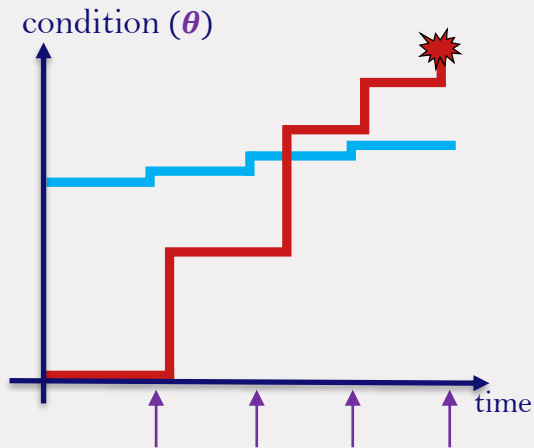
**Step 3:** Properties of the policy with respect to  $(\widehat{\theta}_t, s_t, t)$

**THEOREM 1.** *At each component age  $t \in \mathbb{N}_0$ , for a given number of shock arrivals  $n \in \mathbb{N}_0$ , there exists a control limit  $\delta^{(n,t)} \leq \xi$ , such that the optimal action is to carry out a preventive replacement if and only if  $x \geq \delta^{(n,t)}$ . The control limit  $\delta^{(n,t)}$  is monotonically non-decreasing in  $t$ , for all  $n$ .*

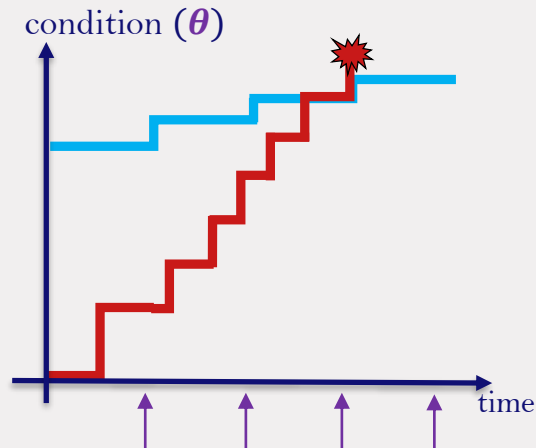
# Cost savings

Oracle < Bayes cost gap < Myopic  $\theta$  updating cost gap < Historical  $\theta$  cost gap

“Few large jumps”



“Many small jumps”



# Cost saving

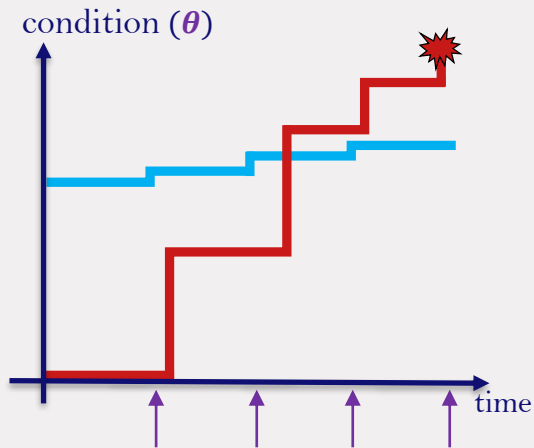
0,6%

7.08%

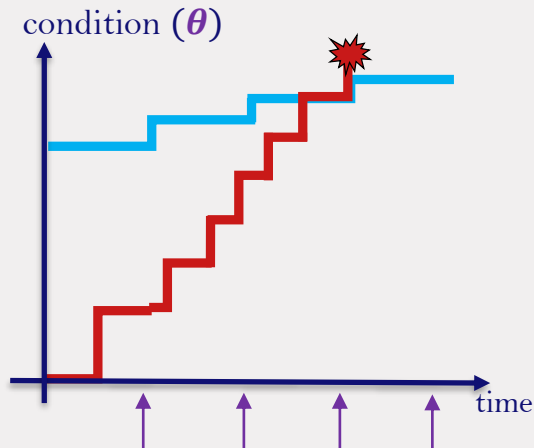
15.02%

Oracle < Bayes cost gap < Myopic  $\theta$  updating cost gap < Historical  $\theta$  cost gap

“Few large jumps”



“Many small jumps”



Intermittent degradation signal gap = 15%

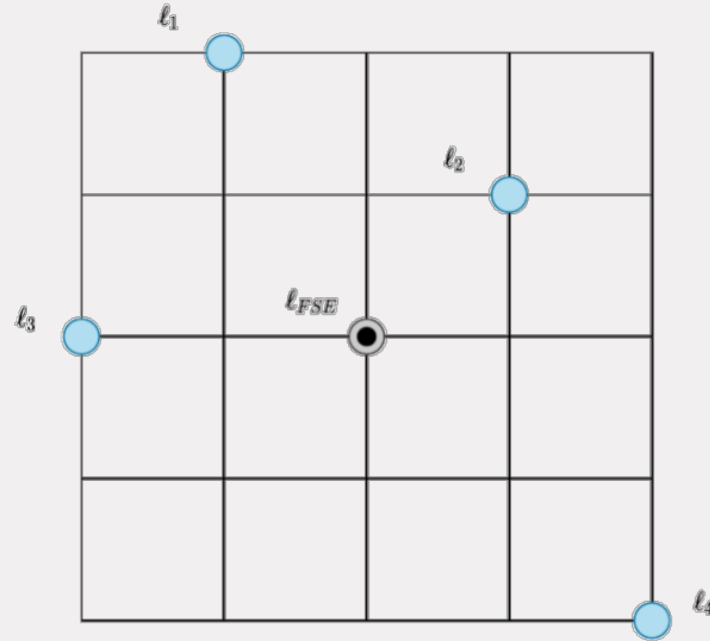
Imperfect degradation signal gap = 6%

# The challenges of the network case



## Goal: Maintenance planning using IoT data in a network of machines

- Field service engineers FSE (●) travel in a network of machines
- Strategic location (○)
- Machine location (●)
- Degradation of a machine raises an **alert** (|)
- Machine failure (○)



# Goal: Maintenance planning using IoT data in a network of multi-component machines

## Challenges

- Traveling salesman problem
- Real time scheduling problem
- Maintenance optimization problem

Preventive/Corrective/Downtime cost and degradation



High-tech  
equipment



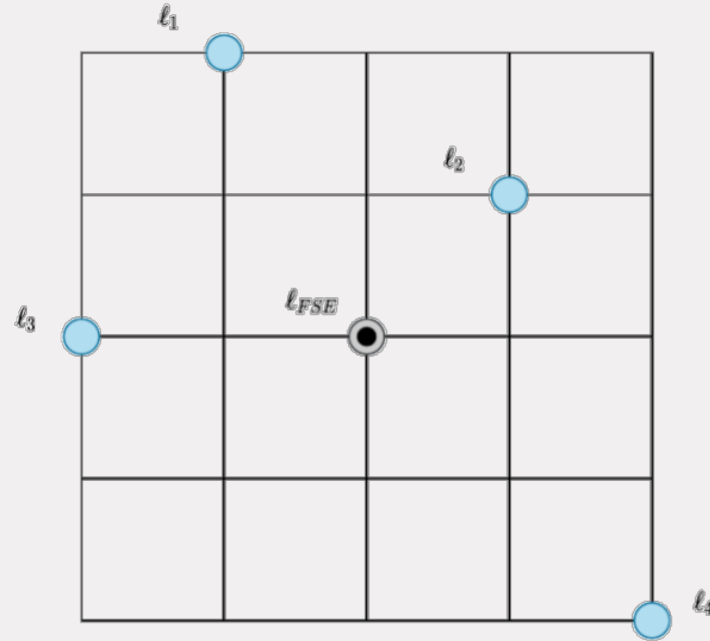
Enable exact  
optimization  
for small  
instances



Assumptions  
based on  
literature



Online  
decision  
making





# Sketching the idea

Step 1: Solve the single maintainer problem

- Compare API with MDP - Exact method  
Solve exactly MDP for small problems

Step 2

Step 3

# Iterative methods

## Approximate policy iteration

1. Choose an initial policy  $\pi_0$
2. Select a subset of states  $S' \subseteq S$
3. For all  $h \in S'$ : compute the best action  $a^*$  using simulation given the policy  $\pi_0$  is used after  
$$\rightarrow a^* = \arg \max_{a_0 \in A(h_0)} \{C(h_0, a_0) + \gamma \mathbb{E}^{\pi_0}[Q(H_1)|h_0, a_0]\}, h_0 \in S'$$
4. Train a neural network classifier on the constructed data set  $\rightarrow$  induces a policy  $\pi_1$ 
  - Input: feature representation  $f(h)$  of a state  $h \in S$ .
  - Output: probability distribution over the action space
5. If  $\pi_1$  improves upon  $\pi_0$ : set  $\pi_0 = \pi_1$  and return to step 2, else: terminate

**Requires a suitable choice of  $\pi_0$  and  $S'$ !**

# Approximate policy iteration

## Subset of states $S'$

- Must depend on  $\pi_0$
- Idea: Given  $\pi_0$ , construct a data set  $\mathcal{D} = \{(h, a_{\pi_0}^*)\}$  containing “optimal”, “most-likely” state-action pairs for the MDP reformulation using simulation.

## Initial policy $\pi_0$

- Display (some) desired behavior
- Must be a fast algorithm
  - I.e., polynomial time complexity
- Idea: Equip existing greedy/reactive heuristics
  - Nearest neighbour
  - Greedy based on distance/cost/time
  - Deterministic problem by first order approximation

# Approximate policy iteration

## Training the neural network classifier

- Split the data set in a training set and a test set
- Minimize (cross-entropy) training loss  $L(\theta)$  on the training set
  - Loss function measures the distance between the neural network policy and the simulation-based policy for the feature representation of the states in the training set
  - Fitting the neural network parameters  $\theta$  is an iterative, gradient-based process: In each step, the gradient of  $L(\theta)$  is estimated with respect to  $\theta$ . Subsequently,  $\theta$  is updated by taking a step in the opposite direction.

## Feature representation

$$f(h) = \left( x_1, n_1^{\text{av}}, n_1^{\text{ua}}, \hat{t}_1^\nu, t_1^{\Theta_1}, t_1^{\Theta_2} \xi_1, \dots, x_M, n_M^{\text{av}}, n_M^{\text{ua}}, t_M^\nu, t_M^{\Theta_1}, t_M^{\Theta_2}, \xi_M, \sum_{m \in \mathcal{M}} n_m^{\text{av}} \right)$$

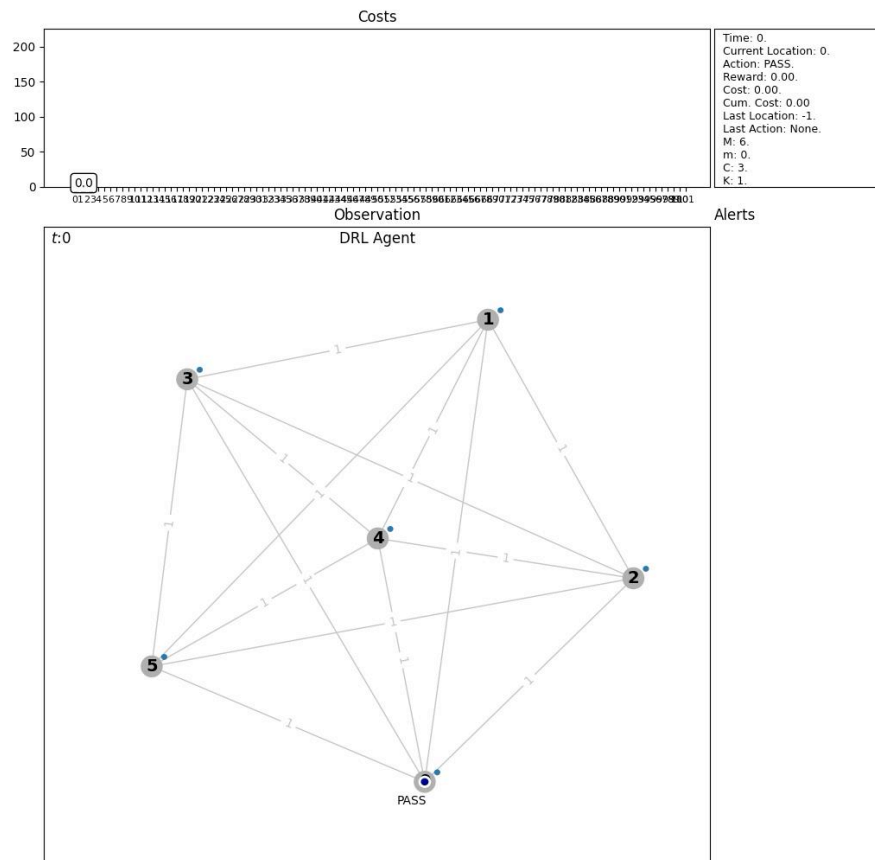
# Sketching the idea

Step 1: Solve the single maintainer problem

- Compare API with MDP - Exact method  
Solve exactly MDP for small problems

Step 2

Step 3



# Sketching the idea

Step 1: Solve the single maintainer problem

- Compare API with MDP - Exact method  
Solve exactly MDP for small problems

Step 2: Solve the multi-maintainer problem for homogeneous machines

Key elements

- Reformulation of the action space
- Choosing a smart, suitable initial solution
  - Prioritize machines based on proximity, urgency, and economic risk
  - Incorporating dispatching and relocation
- Benchmark against heuristic policies
- Robust against network modifications (removing asset or engineer) or yield a suitable initial solution

Step 3

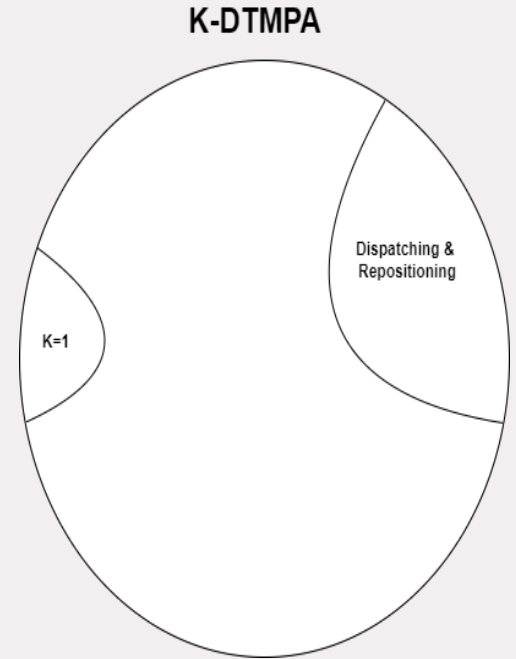
## Single maintainer ( $K=1$ )

- Developed heuristics yield competitive policies
- Deep reinforcement learning outperforms the heuristics

## Dispatching & repositioning ( $K>1$ )

### General insights

- ✓ API can solve single maintainer instances up to optimality within few iterations and produces state-of-the-art improved policies for multi-maintainer instances.
- ✓ Smart dispatching heuristics are superior initial solutions for solving multi-maintainer instances.
- ✓ The trained policies are robust against removing an asset or engineer or yield a suitable initial solution.
- ✓ Deep reinforcement learning can be applied to industrial cases to provide cost-efficient and scalable solutions.





# Sketching the idea

Step 1: Solve the single maintainer problem

- Compare API with MDP - Exact method  
Solve exactly MDP for small problems

Step 2: Solve the multi-maintainer problem for homogeneous machines

Key elements

- Reformulation of the action space
- Choosing a smart, suitable initial solution
  - Prioritize machines based on proximity, urgency, and economic risk
  - Incorporating dispatching and relocation
- Benchmark against heuristic policies
- Robust against network modifications (removing asset or engineer) or yield a suitable initial solution

Step 3

# Iterative methods

## Approximate policy iteration

1. Choose an initial policy  $\pi_0$
2. Select a subset of states  $S' \subseteq S$
3. For all  $h \in S'$ : compute the best action  $a^*$  using simulation given the policy  $\pi_0$  is used after  
$$\rightarrow a^* = \arg \max_{a_0 \in A(h_0)} \{C(h_0, a_0) + \gamma \mathbb{E}^{\pi_0}[Q(H_1)|h_0, a_0]\}, h_0 \in S'$$
4. Train a neural network classifier on the constructed data set  $\rightarrow$  induces a policy  $\pi_1$ 
  - Input: feature representation  $f(h)$  of a state  $h \in S$ .
  - Output: probability distribution over the action space
5. If  $\pi_1$  improves upon  $\pi_0$ : set  $\pi_0 = \pi_1$  and return to step 2, else: terminate

**Requires a suitable choice of  $\pi_0$  and  $S'$ !**

# Approximate policy iteration

## Subset of states $S'$

- Must depend on  $\pi_0$
- Idea: Construct a data set  $\mathcal{D} = \{(h_i^{(a_i^{(k-1)})}, a_i^k) \mid i \in I\}$  containing “optimal” state-action pairs for the MDP reformulation using simulation.

## Initial policy $\pi_0$

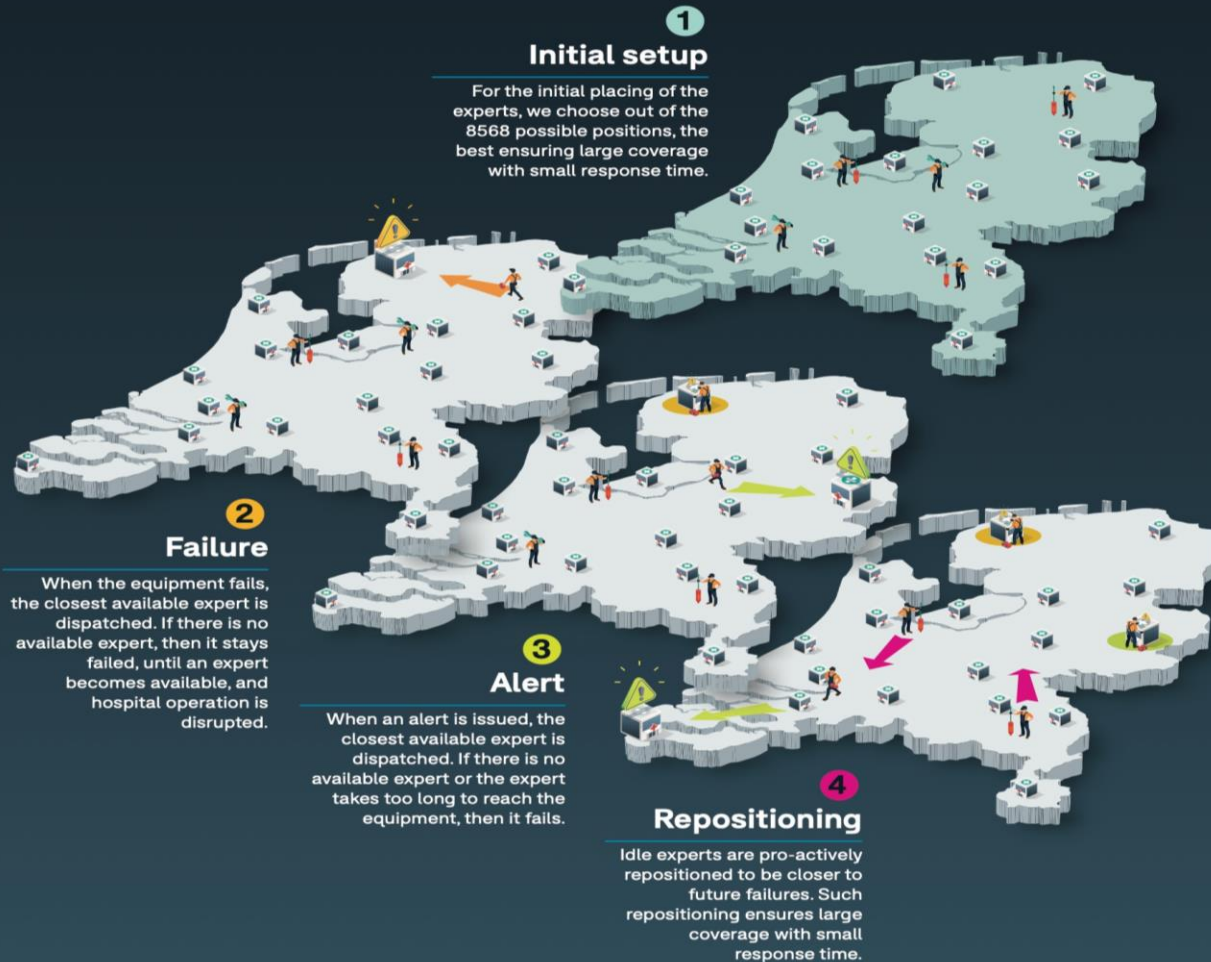
- Display (some) desired behavior
- Must not ‘self-correct’
  - E.g., a partitioned solution will not learn to share resources over the network
- Must be a fast algorithm
  - I.e., polynomial time complexity
- Idea: Equip existing greedy/reactive heuristics with state-of-the-art dispatching algorithm

# Approximate policy iteration

## Initial policies $\pi_0$

- Ranking heuristics with sequential dispatching: First assign engineer 1, then engineer 2, ...
  - Does not include cooperative behavior
- Ranking heuristics with simultaneous dispatching: Shortest pair first
  - Includes suboptimal cooperative behavior
- Ranking heuristics with Hungarian dispatching: Construct and solve an assignment problem
  - Includes optimal cooperative behavior
  - Assignment problems are solvable in polynomial time complexity

# DISPATCHING EXPERTS TO DO MAINTENANCE

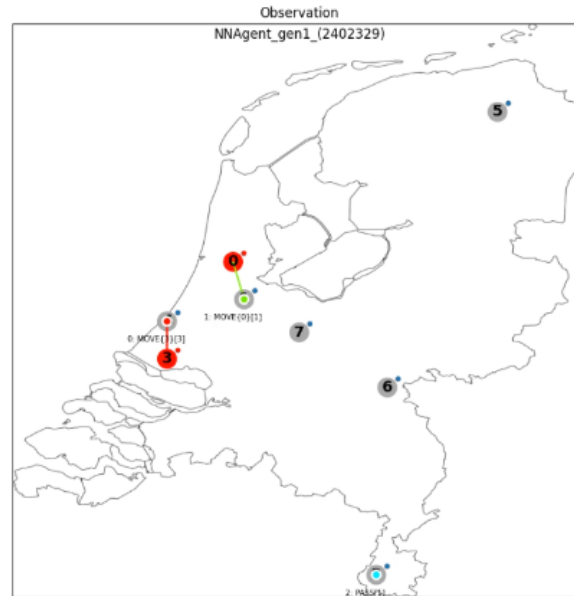


Nowadays surgical operations require advanced robotic equipment. Such equipment can help save lives. Unfortunately, such equipment deteriorates with usage and it can eventually fail. When it fails, it requires maintenance from an expert engineer. Until it is maintained, it cannot be used and hospital operation is disrupted. Thankfully failures are often predicted before they happen. When a failure is predicted, we issue an alert and we plan how to dispatch an expert to do maintenance with minimal disruption of the hospital operation. Combining mathematics and Artificial Intelligence (AI), we design algorithms for the dispatching of the experts ensuring as few and as short as possible disruptions at a low cost. The biggest challenge in designing these algorithms is the computational complexity as there are typically hundreds of experts and thousands of equipment.

# Results



(a) Strategic initial positioning.



(b) Efficient dispatching.



(c) Tactical repositioning.

# Approximate policy iteration

## Training the neural network classifier

- Split the data set in a training set and a test set
- Minimize training loss  $L(\theta)$  on the training set
  - Loss function measures the distance between the neural network policy and the simulation-based policy for the feature representation of the states in the training set
  - Fitting the neural network parameters  $\theta$  is an iterative, gradient-based process: In each step, the gradient of  $L(\theta)$  is estimated with respect to  $\theta$ . Subsequently,  $\theta$  is updated by taking a step in the opposite direction.

## Feature representation

$$f(h) = \left( x_1, n_1^{\text{av}}, n_1^{\text{ua}}, \hat{t}_1^\nu, t_1^{\Theta_1}, t_1^{\Theta_2} \xi_1, \dots, x_M, n_M^{\text{av}}, n_M^{\text{ua}}, t_M^\nu, t_M^{\Theta_1}, t_M^{\Theta_2}, \xi_M, \sum_{m \in \mathcal{M}} n_m^{\text{av}} \right)$$

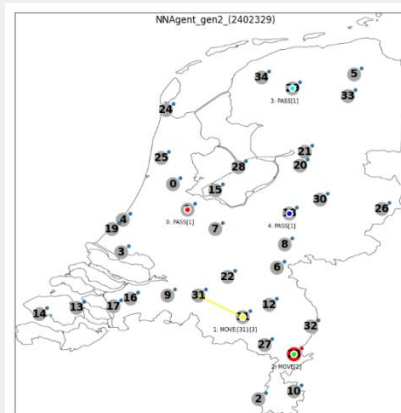
# Dispatching & Repositioning: Dutch academic hospitals (M35K5-Q1C1)

[Source](#)

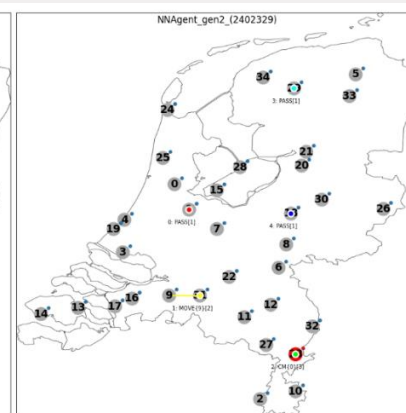
$M = 35$	assets
$K = 5$	engineers
$N_m \equiv 2$	states
$c_{PM} = 0$	cost PM
$c_{CM} = 0$	cost CM
$c_{DT} = 1$	cost DT
$c_T = 0.05$	cost travel
$\theta_{ij} \in \{1, \dots, 16\}$	time travel
$t_{PM} = 4$	time PM
$t_{CM} = 4$	time CM

Policy	Performance
Benchmark (Reactive)	65.612 $\pm$ 0.074
$\pi_1$ API ( $\pi_0$ : Reactive)	63.178 $\pm$ 0.070
$\pi_2$ API ( $\pi_0$ : Reactive)	<b>62.101 <math>\pm</math> 0.069</b>

**$\Rightarrow$  5.35% improvement!**





(a) Tactical repositioning from Eindhoven to Tilburg.

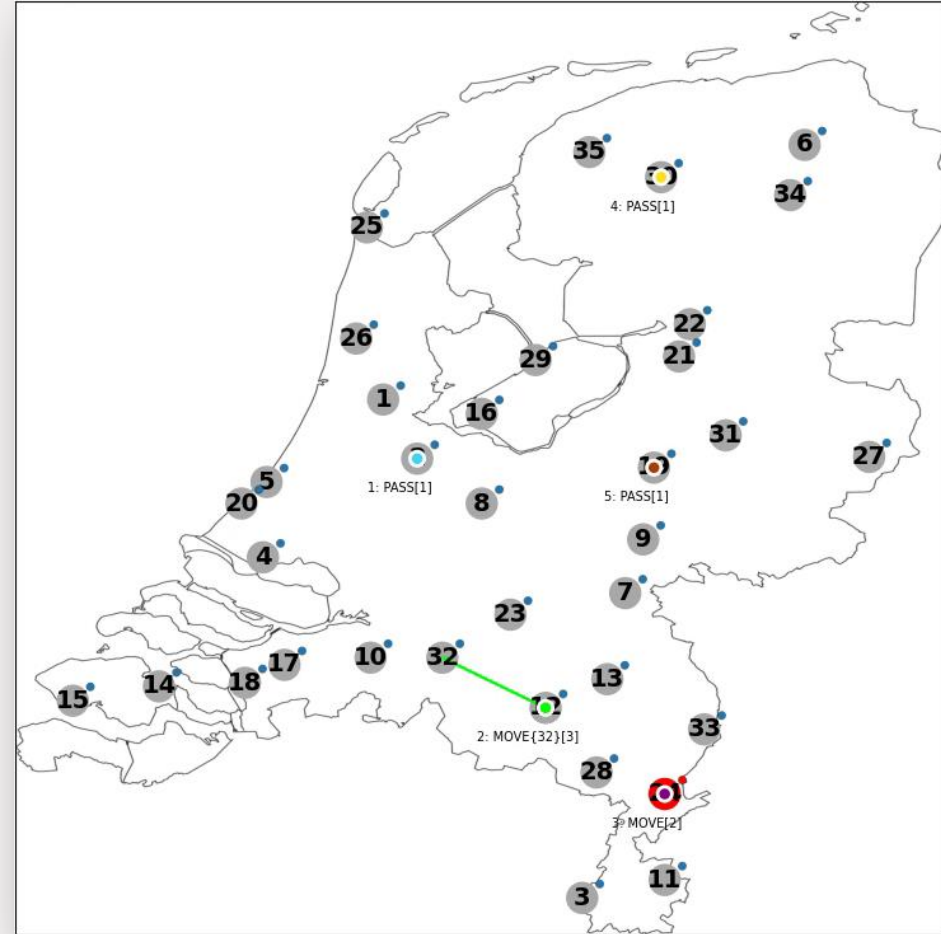


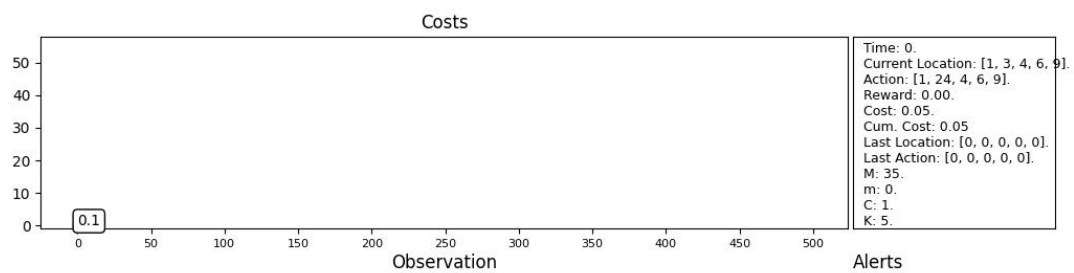
(b) Tactical repositioning from Tilburg to Breda.



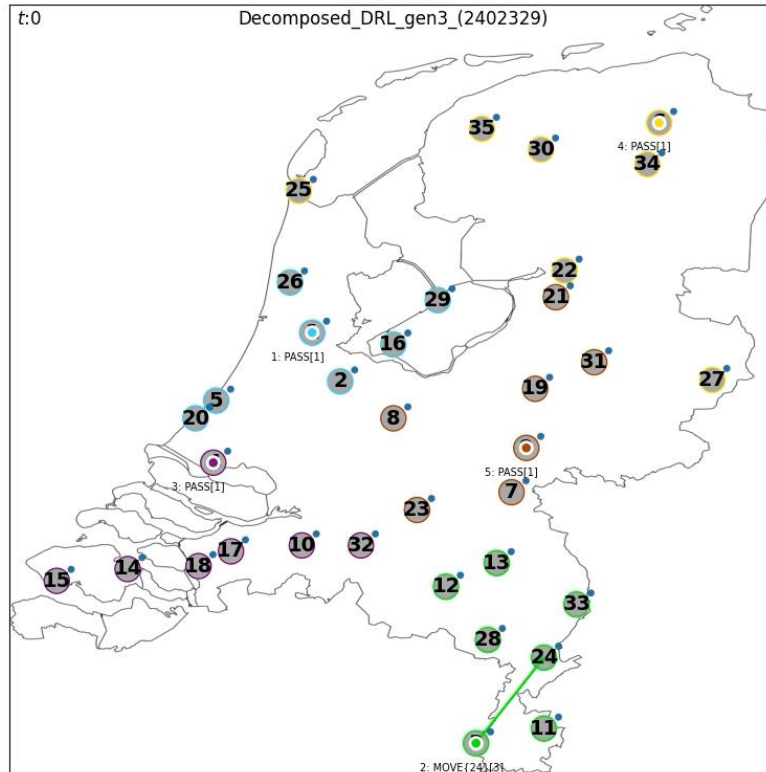
## Results

An exemplary showcase of learned behavior from a neural network policy trained via API: Upon dispatching an engineer to perform corrective maintenance in Roermond , the neural network policy repositions an engineer from Eindhoven  to Tilburg to improve coverage in the network in anticipation of future alerts and failures.





Alerts



# Sketching the idea

Step 1: Solve the single maintainer problem

- Compare API with MDP - Exact method  
Solve exactly MDP for small problems

Step 2: Solve the multi-maintainer problem for homogeneous machines

Key elements

- Reformulation of the action space
- Choosing a smart, suitable initial solution
  - Prioritize machines based on proximity, urgency, and economic risk
  - Incorporating dispatching and relocation
- Benchmark against heuristic policies
- Robust against network modifications (removing asset or engineer) or yield a suitable initial solution

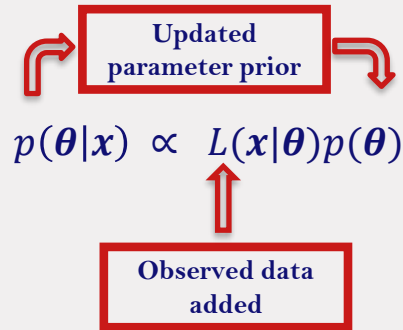
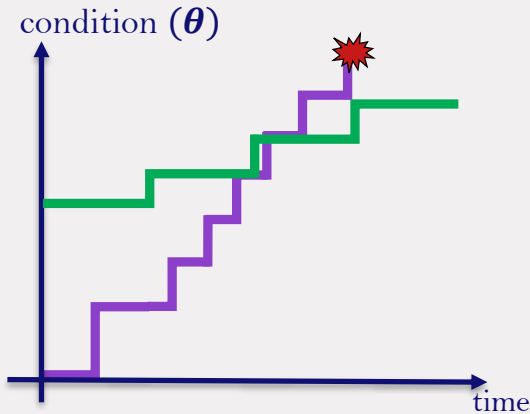
Step 3: Solve the single maintainer problem for heterogeneous machines

Key elements

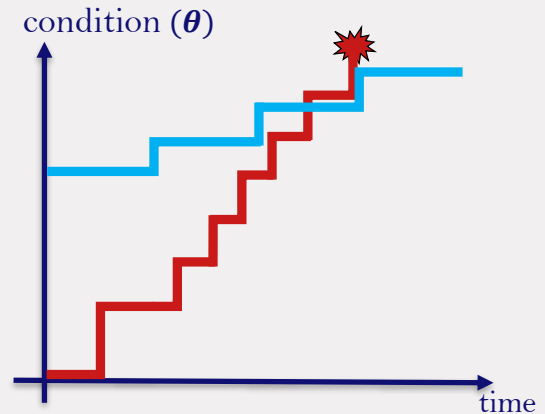
- Incorporate Bayes updating in initial solution

# Heterogeneous multi-component systems

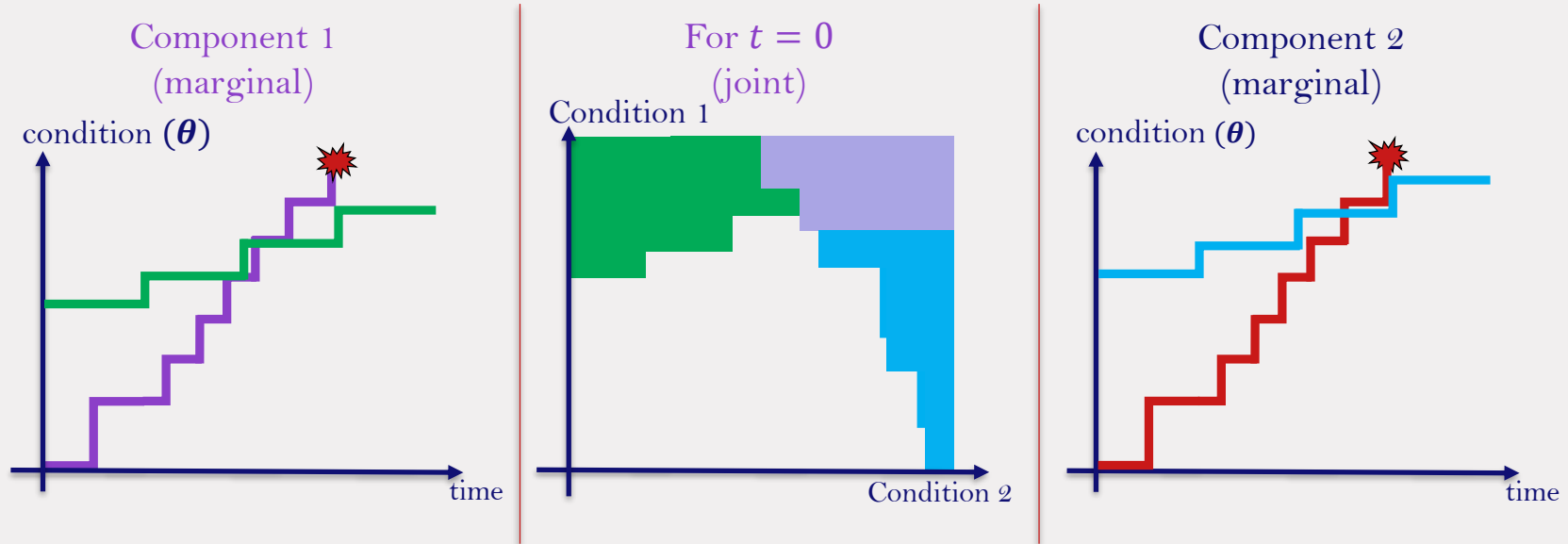
Component 1



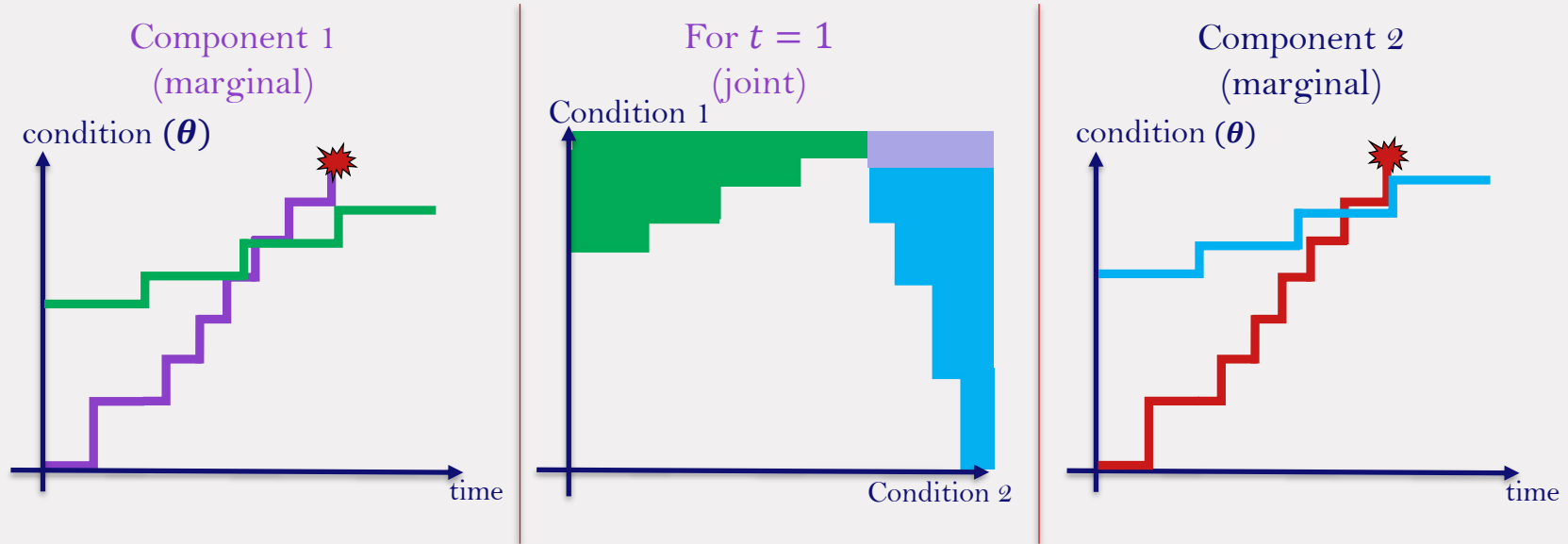
Component 2

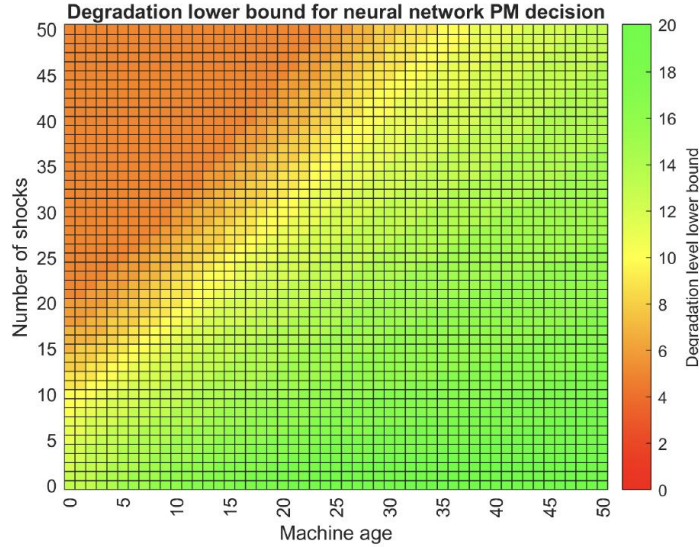


# Heterogeneous multi-component systems

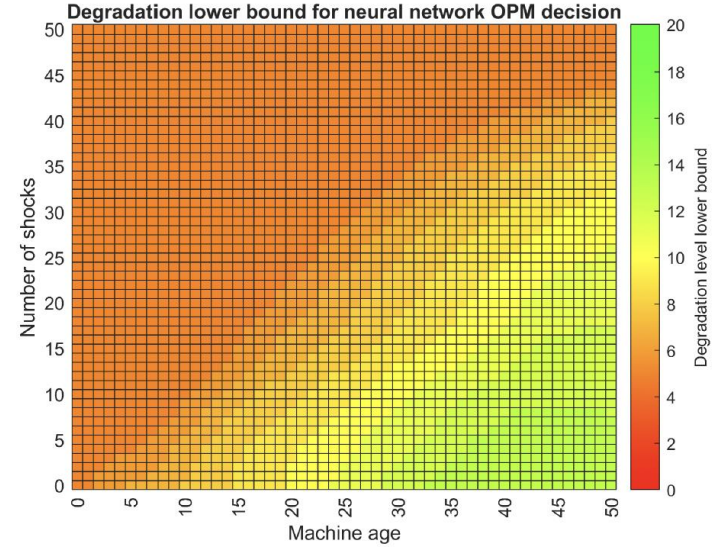


# Heterogeneous multi-component systems





(a) Heatmap of the minimum degradation level  $x_1$  at which maintenance is initiated, given  $k_1(t)$  and  $t_1(t)$ , assuming the other machine is in the healthy state  $(x_2(t), k_2(t), t_2(t)) = (0, 0, 0)$ .



(b) Heatmap of the minimum degradation level  $x_1$  at which maintenance is initiated, given  $k_1(t)$  and  $t_1(t)$ , assuming the other machine is in the failed state  $(x_2(t), k_2(t), t_2(t)) = \left(\xi_2, \frac{\xi_2 \cdot \mu_\Phi}{1 - \mu_\Phi}, \frac{\xi_2 \cdot \mu_\Phi}{(1 - \mu_\Phi) \cdot \mu_\Lambda}\right)$ .

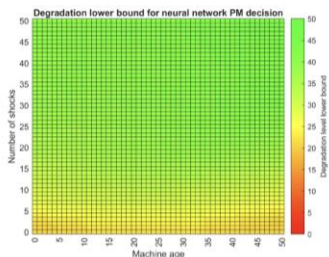
Figure 3: Two policy slices from the best-performing neural network policy for instance I.1, illustrating the complex transformation from PM decisions to OPM decisions.

Instance	$M$	$\xi_m$	$\mu_\Lambda$	$CV_\Lambda$	$\mu_\Phi$	$CV_\Phi$	$\alpha$	$1/\beta$	$a$	$b$	$c_m^{\text{PM}}$	$c_m^{\text{CM}}$	$c^{\text{ST}}$	$\gamma$
CS.1	1	50	1.414	0.157	0.487	0.234	40.696	28.779	8.924	9.405	1	5	0	0.99
CS.2	2	50	1.414	0.157	0.487	0.234	40.696	28.779	8.924	9.405	1	5	1	0.99
CS.3	5	50	1.414	0.157	0.487	0.234	40.696	28.779	8.924	9.405	1	5	1	0.99

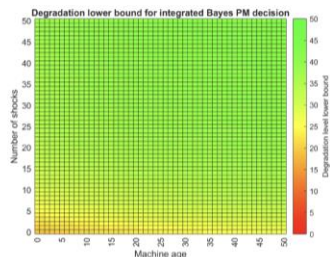
Table 5: Hyperparameter settings and cost structures considered in the case study.

Instance	$\tau_*^{\text{PM}}$	$\tau_*^{\text{OPM}}$	$J(\pi_{\mathcal{N}}^{\text{L}_1})$	$J(\pi_{\mathcal{R}}^{\text{L}_1})$	$J(\pi_{\mathcal{I}}^{\text{L}_1})$
CS.1	40	—	$3.146 \pm 0.002$	$11.071 \pm 0.006$	$2.974 \pm 0.002$
CS.2	41	28	$11.381 \pm 0.005$	$26.516 \pm 0.010$	$11.558 \pm 0.005$
CS.3	41	29	$26.405 \pm 0.007$	$65.876 \pm 0.016$	$28.270 \pm 0.007$

Table 6: Summary of the heuristic solution calibration results for the case study instances CS.1–3.



(a) Heatmap of the minimum degradation level  $x_1$  at which maintenance is initiated according to the neural network policy, given  $k_1(t)$  and  $t_1(t)$ .



(b) Heatmap of the minimum degradation level  $x_1$  at which maintenance is initiated according to the integrated Bayes heuristic, given  $k_1(t)$  and  $t_1(t)$ .

Figure 5: Policy visualization of the best-performing neural network policy for instance CS.1 and the integrated Bayes heuristic, illustrating their similarity.

		CS.1	CS.2	CS.3
Gen 0	$\pi_0^{\text{L}}$	$\pi_{\mathcal{N}}^{\text{L}_1}$	$\pi_{\mathcal{N}}^{\text{L}_1}$	$\pi_{\mathcal{N}}^{\text{L}_1}$
	$J(\pi_0^{\text{L}})$	$3.146 \pm 0.002$	$11.381 \pm 0.005$	$26.405 \pm 0.007$
Gen 1	$J(\pi_{\theta_1}^{\text{L}_2}(f_1^{\text{L}_2}(h)))$	$2.932 \pm 0.002$	$10.564 \pm 0.004$	$24.020 \pm 0.006$
	$J(\pi_{\theta_1}^{\text{L}_2}(f_2^{\text{L}_1}(\tilde{h})))$	$3.936 \pm 0.003$	$12.663 \pm 0.006$	$28.498 \pm 0.008$
Gen 2	$J(\pi_{\theta_2}^{\text{L}_2}(f_1^{\text{L}_2}(h)))$	$2.90 \pm 0.002$	$10.471 \pm 0.004$	$23.660 \pm 0.006$
	$J(\pi_{\theta_2}^{\text{L}_2}(f_2^{\text{L}_1}(\tilde{h})))$	<b><math>3.767 \pm 0.003</math></b>	<b><math>12.527 \pm 0.006</math></b>	<b><math>28.185 \pm 0.008</math></b>
Gen 3	$J(\pi_{\theta_3}^{\text{L}_2}(f_1^{\text{L}_2}(h)))$	$2.901 \pm 0.002$	$10.516 \pm 0.004$	$23.507 \pm 0.006$
	$J(\pi_{\theta_3}^{\text{L}_2}(f_2^{\text{L}_1}(\tilde{h})))$	$3.804 \pm 0.003$	$13.009 \pm 0.007$	$29.336 \pm 0.009$

Table 7: One-step policy improvement results for case study instances CS.1–3. *Gray rows*: The performance of the neural network policy  $\pi_{\theta}^{\text{L}_2}$  in the  $\text{L}_2$  setting, trained on the underlying MDP. *White rows*: The performance of the neural network policy  $\pi_{\theta}^{\text{L}_2}$  applied in the  $\text{L}_1$  setting using the open-loop feedback approach. **Bold**: Indicates the lowest cost for each instance under  $\text{L}_1$  across neural network generations.





# Optimal Decision-Making under Parameter Uncertainty

STELLA KAPODISTRIA

Summer School Sequential Decision Making 2025

Department of Mathematics and Computer Science