

Optimizing fault search in power grid outages through Reinforcement Learning

Master's degree in *Data Science and Scientific Computing*

CANDIDATE:

Angela Carraro

SUPERVISOR:

Antonio Celani, ICTP

DSSC - UNITS, SISSA, ICTP

Co-SUPERVISORS:

Andrea Zancola, AcegasApsAmga

Luca Bortolussi, UniTS

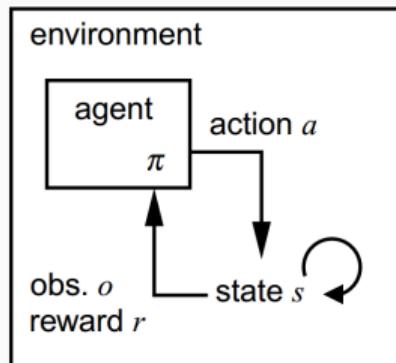
Emanuele Panizon, ICTP



We want to *optimize the fault search* in power grid outages through *Reinforcement learning*.

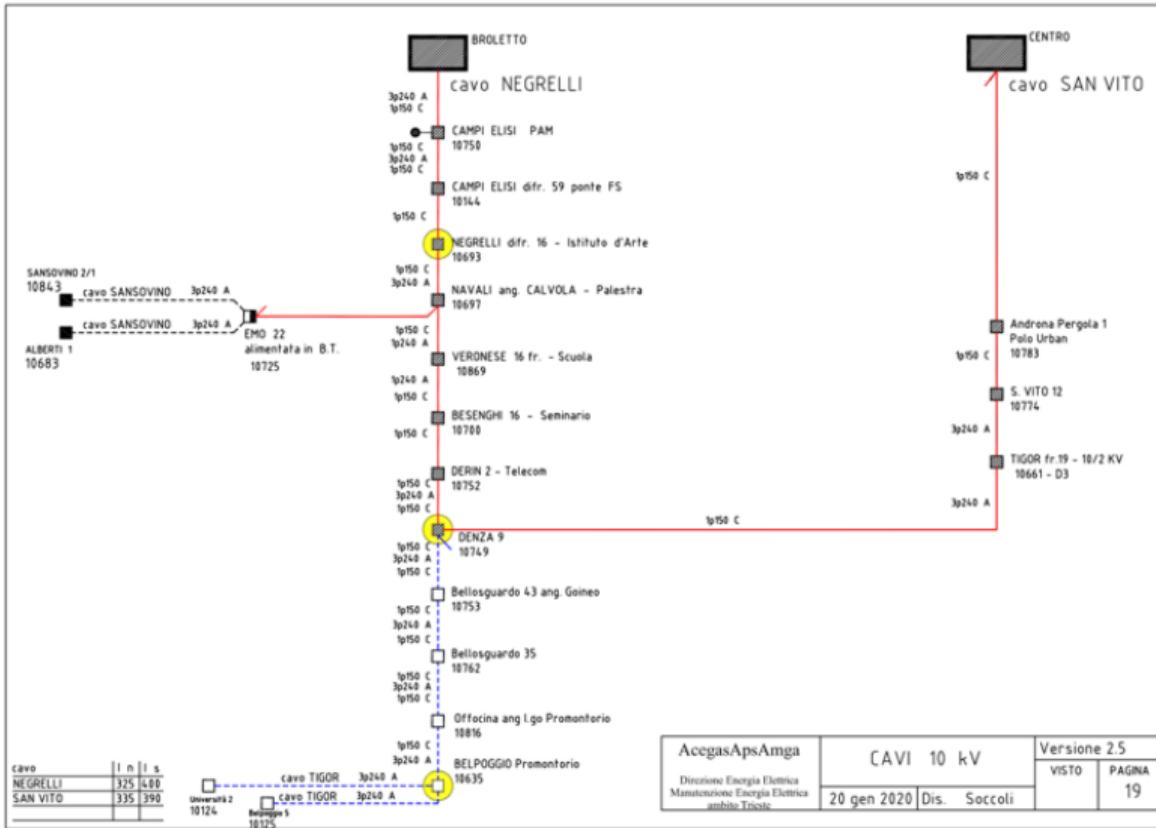
In particular, we use a *policy gradient method* in a *Partially Observable Markov Decision Process*, or POMDP. In particular, we perform a gradient descent in the policy space, using a parametrized policy specifically chosen so to it does not depend on the position of the fault, which is a *hidden state variable*.

Schematic representation of a POMDP agent interacting with the environment.

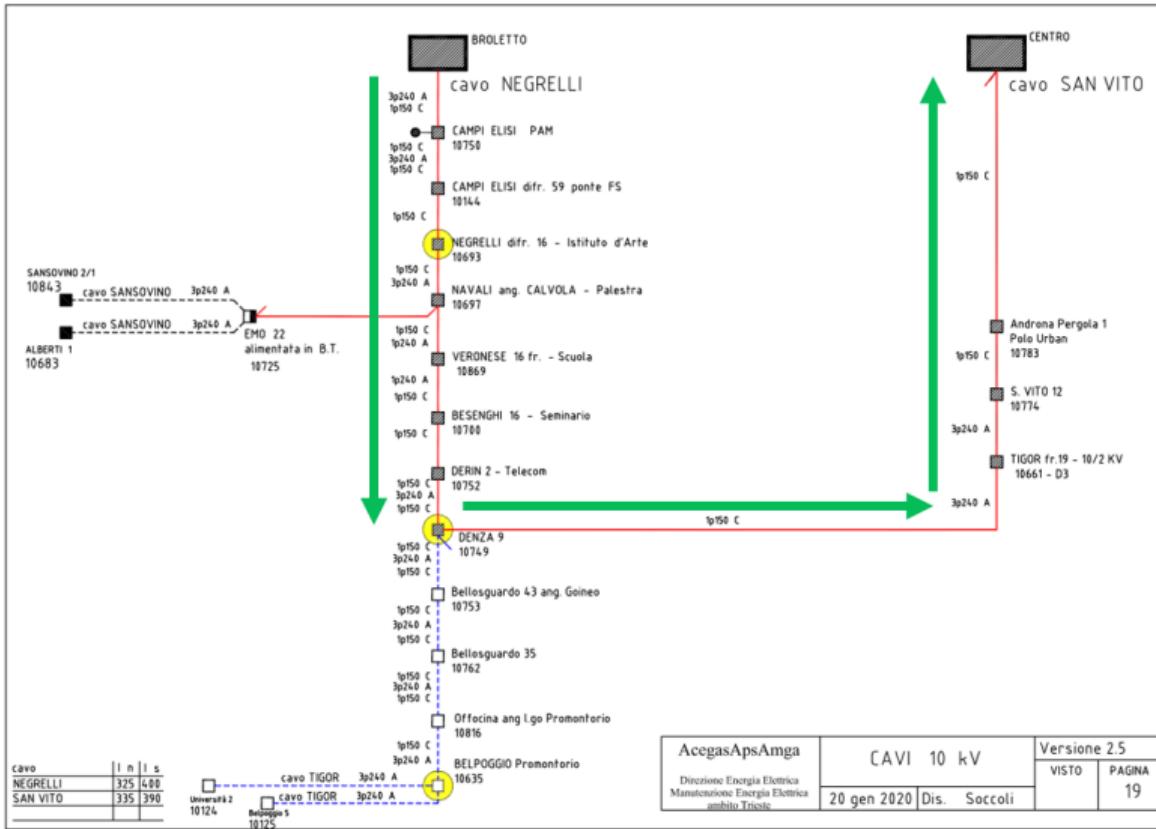


The problem

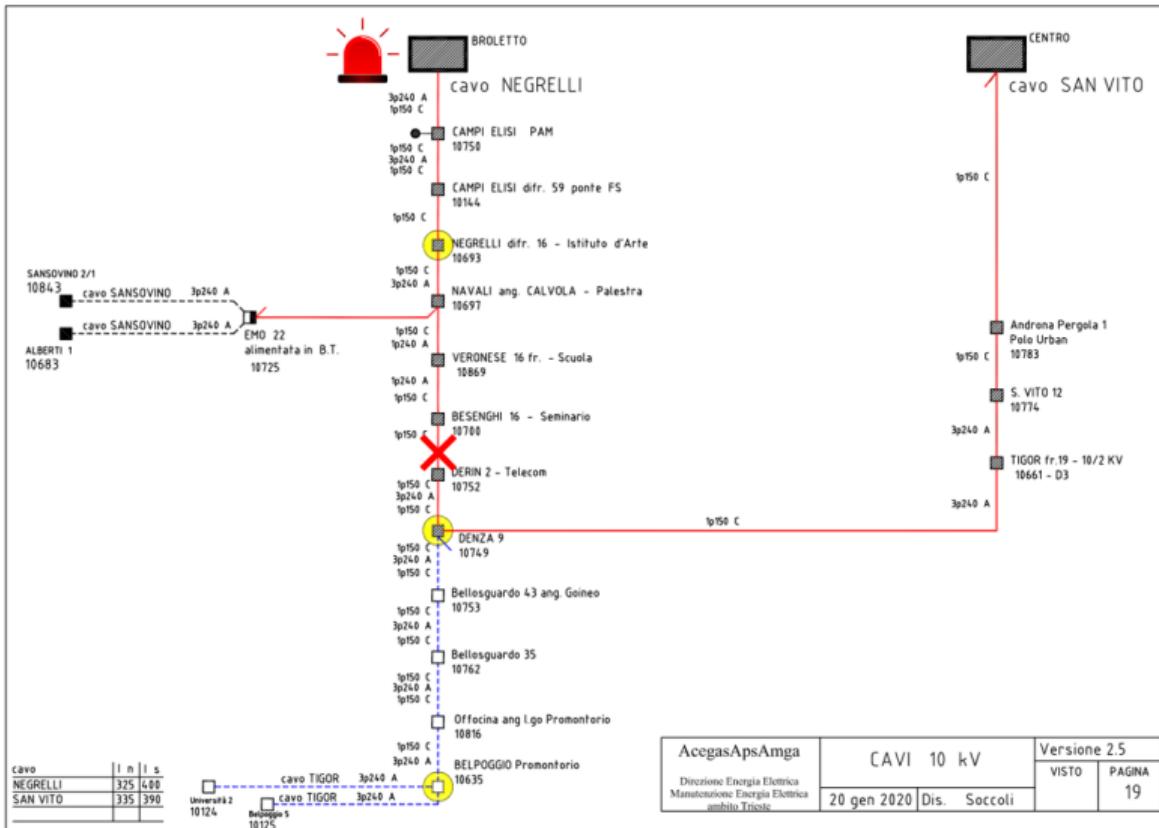
Example of a power line in medium voltage



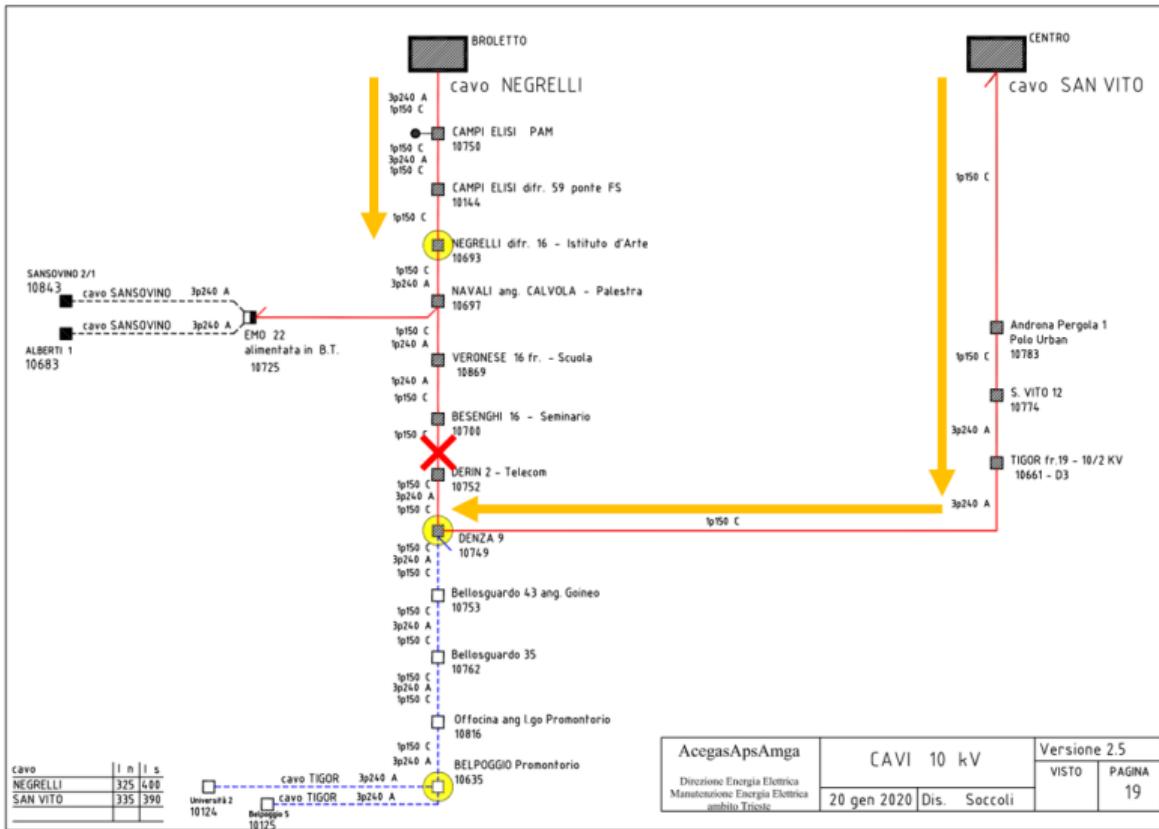
The energy flow in the standard set-up



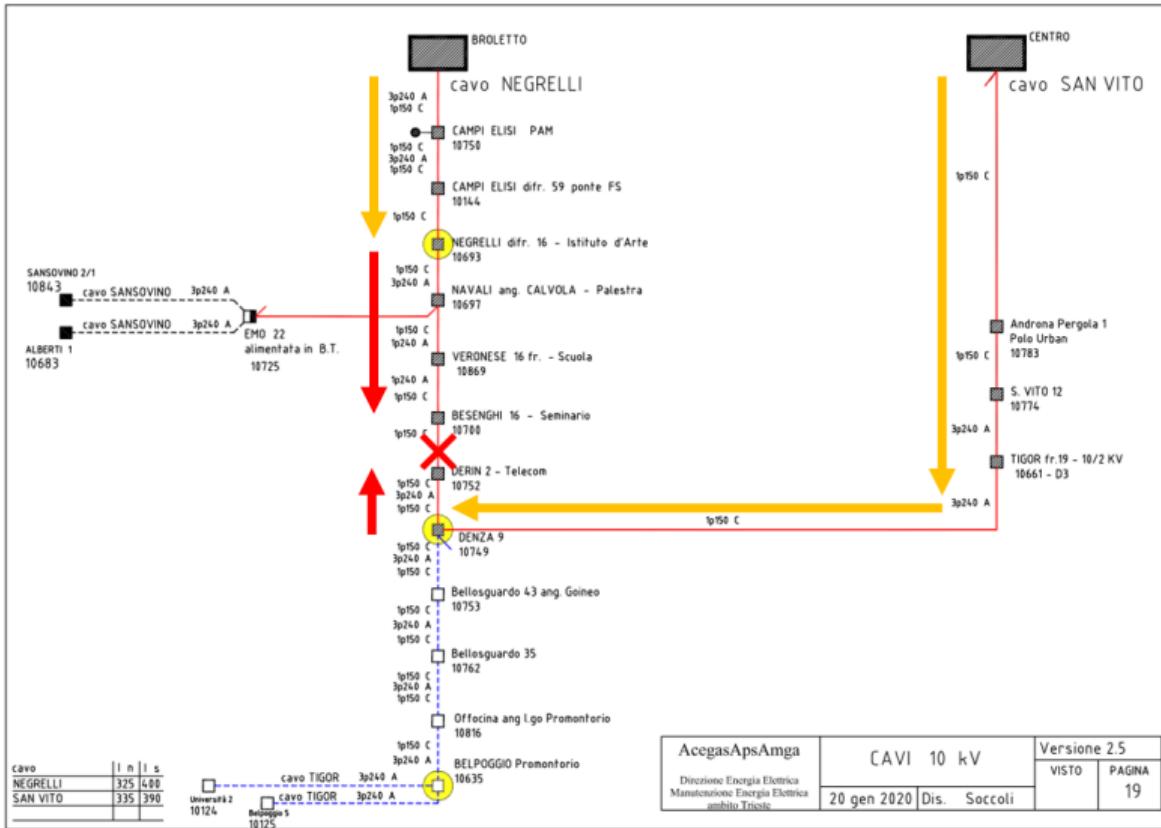
Failure scenario



Failure scenario



Failure scenario



The mathematical framework



Since we don't know where the fault is, we have a problem with **partially observable states**, so we will use a **partially observable Markov decision process (POMDP)**.

We are given a set of disconnected substations, \mathcal{C} , between two remotely controlled substations. In our problem $|\mathcal{C}| < 20$.

- The **state** is

$$s = (x_g, v_k, \{v\})$$

- x_g is the position of the fault (**hidden**);
- $v_k \in \mathcal{C}$ is the substation in which the technician is (**observable**);
- $\{v\}$ is the set of the still disconnected substations after the technician operates in the current substation v_k (**observable**).

- The **observation** is

$$o = (v_k, \{v\})$$

We define it as a function of s : $o(s) : s = (x_g, o) \mapsto o$.

- The **action** is the choice of the specific substation the technician will visit as the next step, so $a \in \mathcal{C}$. Actually, since we visit only disconnected substations, we have that $a \in \{v\}$ if we are in state $s = (x_g, v_k, \{v\})$.

$$a \in \mathcal{A}(s = (x_g, v_k, \{v\})) = \{v\}$$

With every action, we visit a substation and we reconnect it (if lucky, we can reconnect half of the substations). We are therefore positive that the process terminates.

- The **next state** is

$$s' = (x_g, v_{k+1} = a, \{v'\})$$

where $\{v'\}$ is the set of disconnected substations after the technician operates in substation v_{k+1} . We have that the set of disconnected substations decreases after each action, so $\{v'\} \subseteq \{v\} \setminus a$.

- The **expected cost** is the time (*in seconds*) of going in a certain substation multiplied by the number of disconnected users.

$d_{v_k, v_{k+1}}$ → time *in seconds* to go from the substation v_k to the next substation v_{k+1} .

$n_k = \sum_{v \in \{v\}} u_v$ → number of users still disconnected before operating in the substation v_{k+1} , where u_v is the number of users underneath substation v .

$$r(s = (x_g, v_k, \{v\}), a) = d_{v_k, a} \cdot n_k = d_{v_k, a} \cdot \sum_{v \in \{v\}} u_v$$

For now, in the cost we will ignore the cost of establishing if the fault is before or after the substation in which the technician is, which is complicated and might rise the total cost significantly.

When the fault occurs, the technician can be everywhere: at home if it is the middle of the night, at the company, be around, etc. So we introduce an extra dummy substation, called substation 0, that is the position of the technician when the fault occurs.
So the **initial state** is always

$$s_0 = (x_g, o_0 = (0, \mathcal{C}))$$

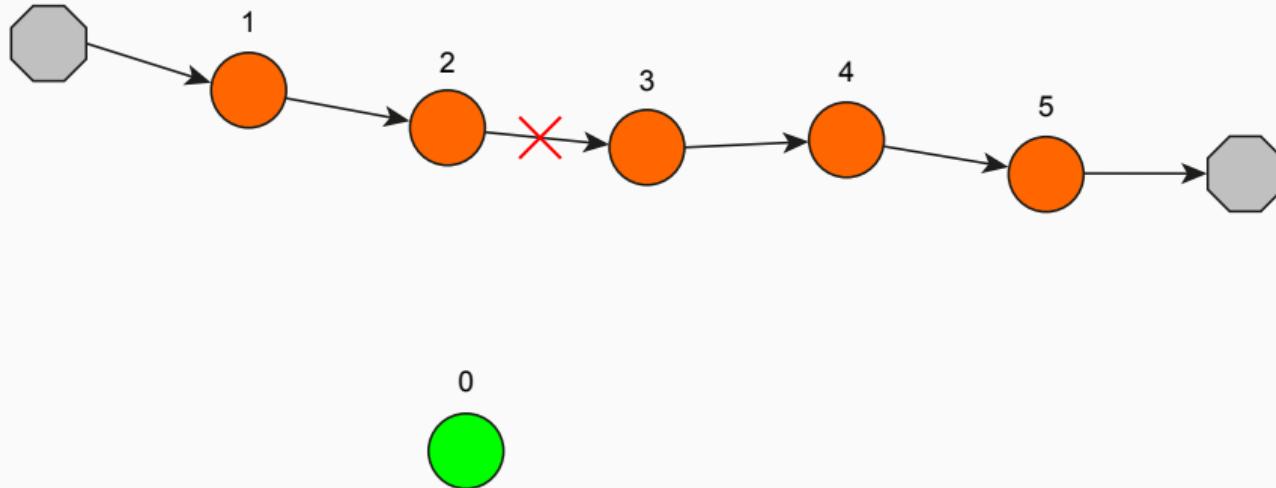
thus we have $|x_g| = 2|\mathcal{C}| + 1$ initial states, one for every possible position of the fault.

Instead, the **terminal state** is of the form

$$s_t = (x_g, v_k, \emptyset)$$

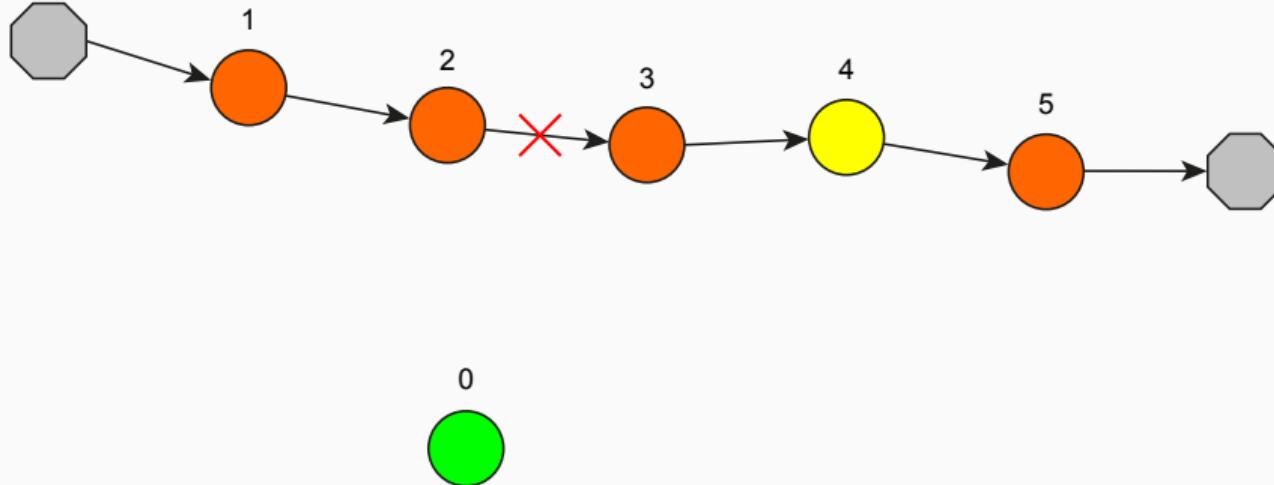
where we have that, if the fault is on a cable, v_k will be one of the two substations at the ends of that faulty cable, so we would have two terminal states, while if the fault is in a substation, v_k would be that exact substation, so the terminal state would be only one.

Example



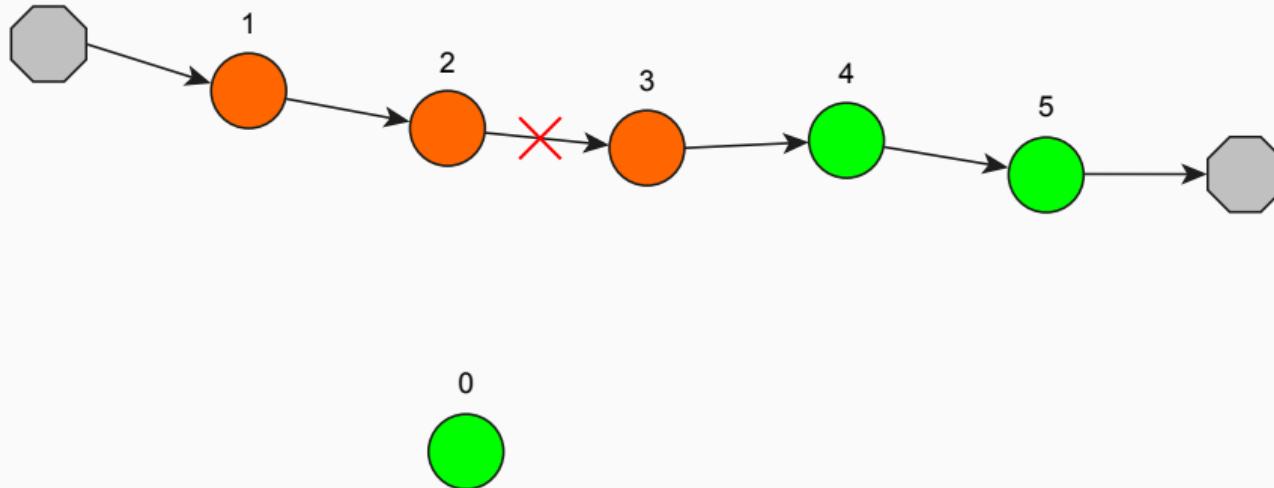
A fault has occurred in 2-3. We are in substation 0 and all the substations are disconnected (orange). Initial state: $s_0 = (2-3, 0, \mathcal{C} = \{1, 2, 3, 4, 5\})$.

Example



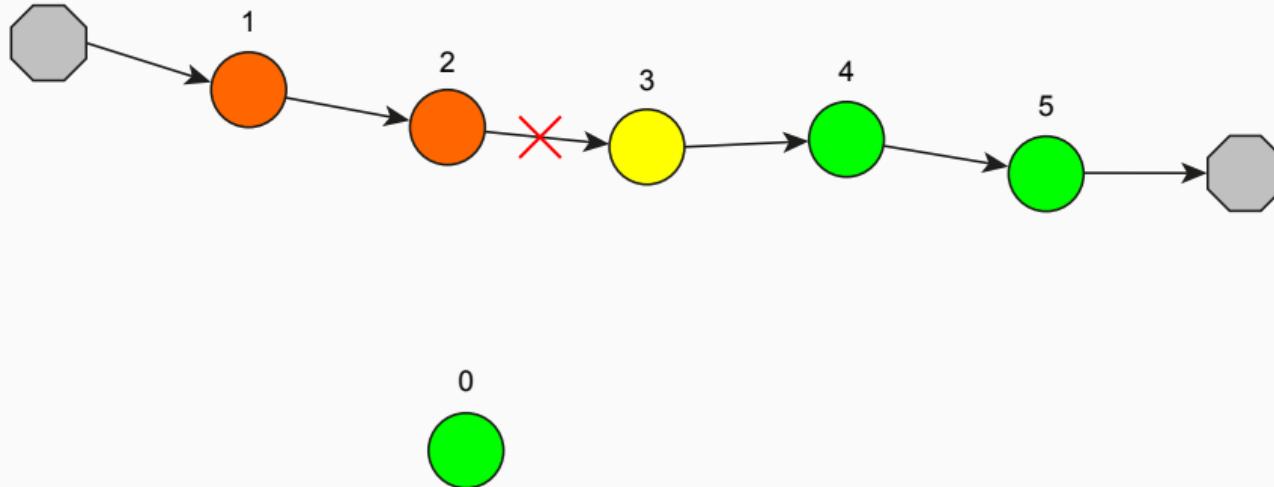
We visit substation 4 (yellow). Action: $a_0 = 4$.

Example



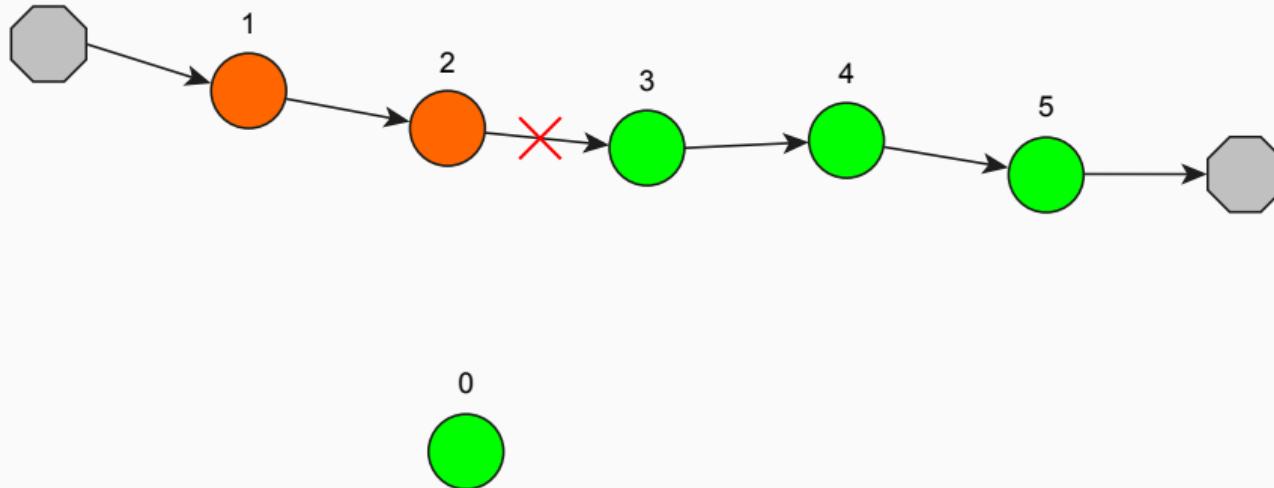
We reconnect substations 4 and 5 (green). State: $s_1 = (2-3, 4, \{1, 2, 3\})$.

Example



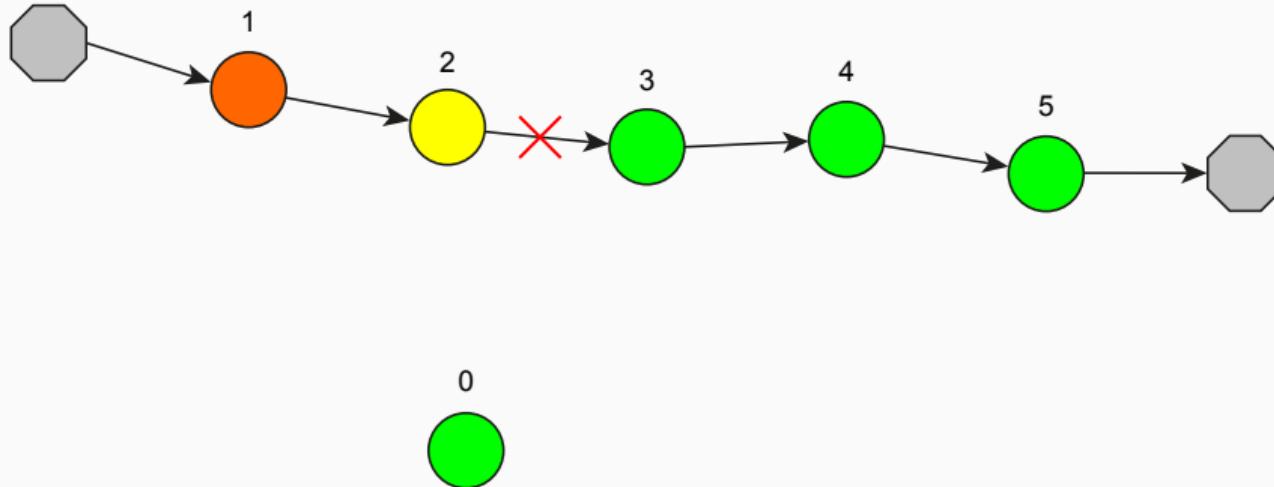
We visit substation 3 (yellow). Action: $a_1 = 3$.

Example



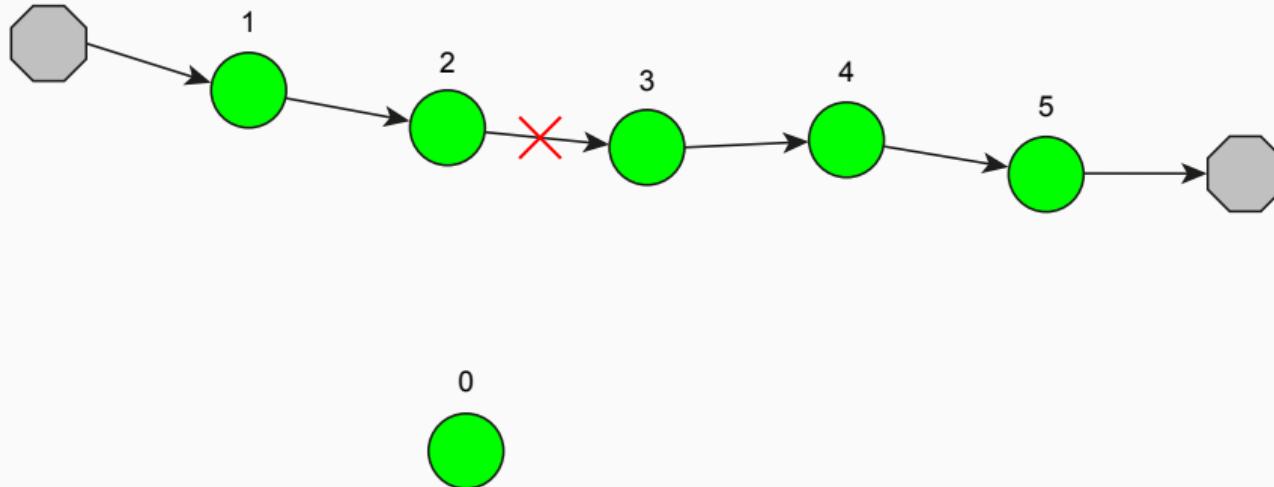
We reconnect substation 3 (green). State: $s_2 = (2-3, 3, \{1, 2\})$.

Example



We visit substation 2 (yellow). Action: $a_2 = 2$.

Example



We reconnected substations 1 and 2 (green). All the substations are reconnected.
Terminal state: $s_3 = (2-3, 2, \emptyset)$.

The model



The system is **deterministic**, so, given an admissible action a , we will surely perform it and end up in the state to which that action leads.

In our case we have that the **transition probability** is equal to 1 only when, starting from state $s = (x_g, v_k, \{v\})$, the new substation v_{k+1} of the next state $s' = (x_g, v_{k+1}, \{v'\})$ is equal to the action $a \in \{v\}$ that we took:

$$p(s' | s, a) = \mathbb{I}(s' = \sigma(s, a)) = \begin{cases} 1 & \text{if } v_{k+1} = a \\ 0 & \text{if } v_{k+1} \neq a \end{cases} .$$



The **policy** depends only on the observable states, so it doesn't know where the fault is. Let's define a parameterized policy using the *soft-max* (i.e., Boltzmann) distribution:

$$\pi(a \mid o = (v_k, \{v\}); \boldsymbol{\theta}) = \frac{e^{\theta_{o,a}}}{\sum_{b \in \{v\}} e^{\theta_{o,b}}}.$$

where $\boldsymbol{\theta}$ are the parameters for each observable o and action a :

$$\boldsymbol{\theta} = (\theta_{o,a})_{o \in \mathcal{O}, a \in \mathcal{A}} = \begin{pmatrix} \theta_{o_1, a_1} & \cdots & \theta_{o_1, a_{|\mathcal{C}|}} \\ \vdots & & \vdots \\ \theta_{o_{|\mathcal{O}|}, a_1} & \cdots & \theta_{o_{|\mathcal{O}|}, a_{|\mathcal{C}|}} \end{pmatrix}.$$

The policy cannot depend on the position of the failure, otherwise we would have automatically solved the problem: the solution would be to go in the substation in which the failure is or at the substations at the ends of the faulty electrical cable.



The **action value function or quality** of the state-action pair, thanks to the Bellman equation, is:

$$Q_\pi(s, a) = \sum_{s'} p(s'|s, a) \left(r(s, a, s') + \sum_{a'} \pi(a'|o(s'); \theta) Q_\pi(s', a') \right).$$

Given that the state is $s = (x_g, o = (v_k, \{v\}))$, the action is $a \in \{v\}$ and the new state is $s' = \sigma(s, a) = (x_g, o' = (v_{k+1} = a, \{v'\}))$, it becomes:

$$Q_\pi(s, a) = \left(d_{v_k, a} \cdot n_k + \sum_{a' \in \{v'\}} \pi(a'|o(\sigma(s, a)); \theta) Q(\sigma(s, a), a') \right).$$



We define $\rho_0(s')$ as the probability of starting in the state s' . In our case

$$\begin{aligned}\rho_0\left(s = (x_g, v_k, \{v\})\right) &= \Pr(x_g) \mathbb{I}(o(s) = o_0 = (0, \mathcal{C})) \\ &= \frac{1}{2|\mathcal{C}| + 1} \delta_{o(s), o_0};\end{aligned}$$

so ρ_0 is uniform in x_g since it doesn't depend on the position of the fault.

Besides, we define the average number of time steps that the agent spends in state s' before the process dies as

$$\begin{aligned}\eta_\pi(s') &:= \rho_0(s') + \sum_s \eta_\pi(s) \sum_a \pi(a|o(s); \boldsymbol{\theta}) p(s'|s, a) \\ &= \frac{1}{2|\mathcal{C}| + 1} \delta_{o(s'), o_0} + \sum_{s \in \text{pa}(s')} \eta_\pi(s) \pi(v_{k+1}|o(s); \boldsymbol{\theta}),\end{aligned}$$

Notice that the equations for both Q_π and η_π are linear systems.



Let's define the **performance measure** $J_\pi(\boldsymbol{\theta})$ as the sum of all the costs we incur, added up over time until the process is concluded:

$$J_\pi(\boldsymbol{\theta}) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} r(s_t, a_t, s_{t+1}) \right] = \sum_{s \in \mathcal{S}} \rho_0(s) \sum_{a \in \mathcal{A}(s)} \pi_{\boldsymbol{\theta}}(a|o(s); \boldsymbol{\theta}) Q_{\pi_{\boldsymbol{\theta}}}(s, a)$$

Thanks to the **policy gradient theorem** we have that

$$\nabla_{\boldsymbol{\theta}} J_\pi(\boldsymbol{\theta}) = \sum_{s \in \mathcal{S}} \eta_{\pi_{\boldsymbol{\theta}}}(s) \sum_{a \in \mathcal{A}(s)} Q_{\pi_{\boldsymbol{\theta}}}(s, a) \nabla_{\boldsymbol{\theta}} \pi(a|o(s); \boldsymbol{\theta}).$$

To optimize J we perform a gradient descent on $\boldsymbol{\theta}$ with *learning rate* α :

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \alpha \nabla_{\boldsymbol{\theta}} J_\pi(\boldsymbol{\theta}_k),$$

Problem: we can compute Q_π and η_π only if we know the position of the fault x_g !



Given the equation for the policy, we have that its derivative is

$$\frac{\partial}{\partial \theta_{o',a'}} \pi(a \mid o = (v_k, \{v\}); \boldsymbol{\theta}) = \delta_{o',o} (\delta_{a',a} - \pi(a'|o; \boldsymbol{\theta})) \pi(a|o; \boldsymbol{\theta})$$

Therefore, the equation of the gradient becomes

$$\begin{aligned} \nabla_{\theta_{o',a'}} J_\pi(\boldsymbol{\theta}) &= \sum_{s \in \mathcal{S}} \eta_\pi(s) \sum_{a \in \mathcal{A}(s)} Q_\pi(s, a) \nabla_{\theta_{o',a'}} \pi(a|o(s); \boldsymbol{\theta}) \\ &= \sum_s \eta_\pi(s) \sum_a Q_\pi(s, a) \delta_{o',o(s)} \left((\delta_{a',a} - \pi(a'|o(s); \boldsymbol{\theta})) \pi(a|o(s); \boldsymbol{\theta}) \right) \\ &= \sum_{x_g} \eta_\pi((x_g, o')) \sum_a Q_\pi((x_g, o'), a) (\delta_{a,a'} - \pi(a'|o'; \boldsymbol{\theta})) \pi(a|o'; \boldsymbol{\theta}), \end{aligned}$$

So, we sum all the values of the gradient that have the same observation but different x_g !



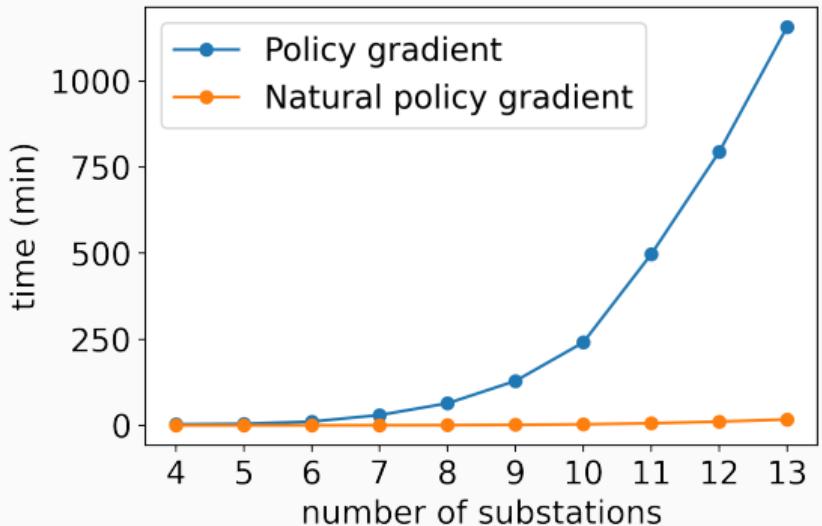
So we **parametrized the policy**, and we imposed that the parameters depend only on the observable variables. Now we try to find the best policy in this subspace in order to optimize the POMDP.

We start from a certain policy, for example *random policy*, in which all the parameters θ are equal to 0: $\theta = \mathbf{0} = (0, 0, \dots, 0)$, so that all actions have an equal probability of being selected

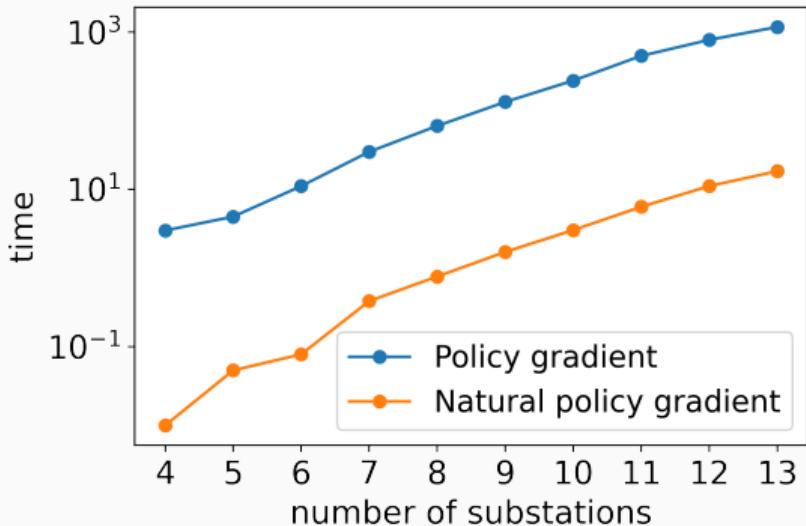
$$\pi(a | o(s); \theta) = \frac{e^\theta}{\sum_{b \in \{v\}} e^\theta} = \frac{e^\theta}{e^\theta \sum_{b \in \{v\}} 1} = \frac{1}{|\{v\}|},$$

Then we perform a step of gradient descent on the parameters θ of the policy. We compute the gradient using the previous formula: the gradients of the policy are simple computations, while to compute Q_π and η_π you have to solve the corresponding linear equations for the current policy, so they have to be solved at each iteration of gradient descend.

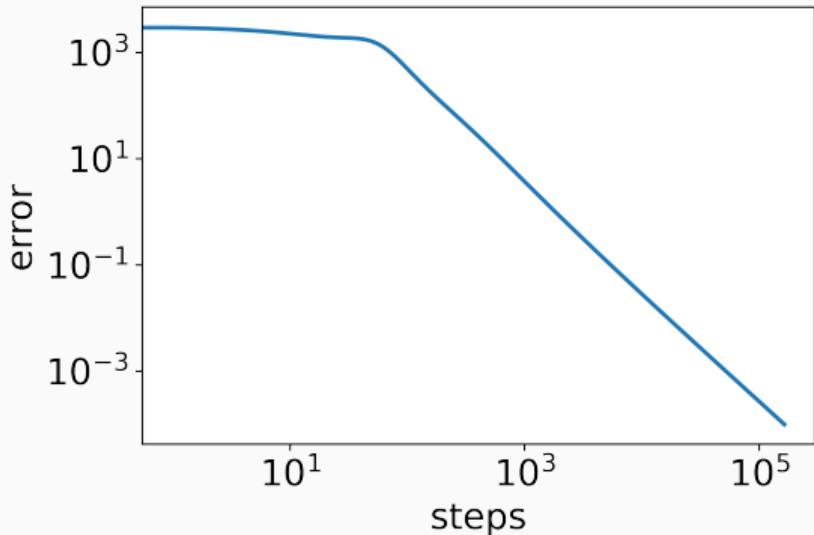
Convergence times



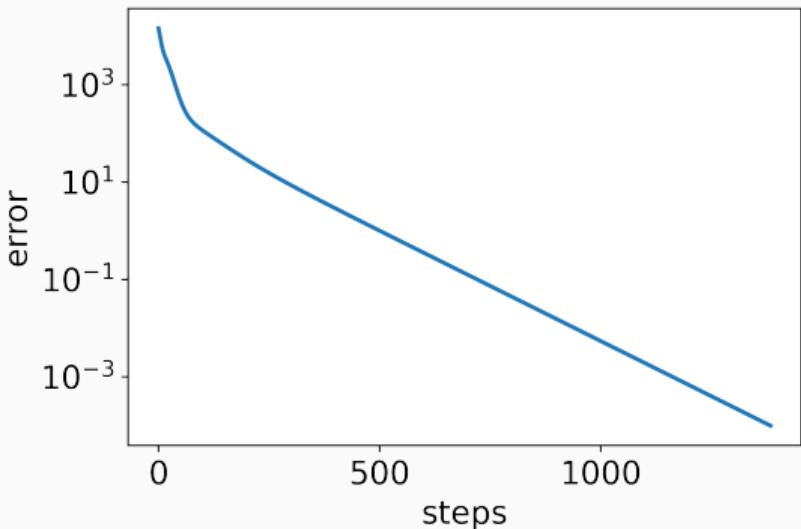
Convergence times (semi-log plot)



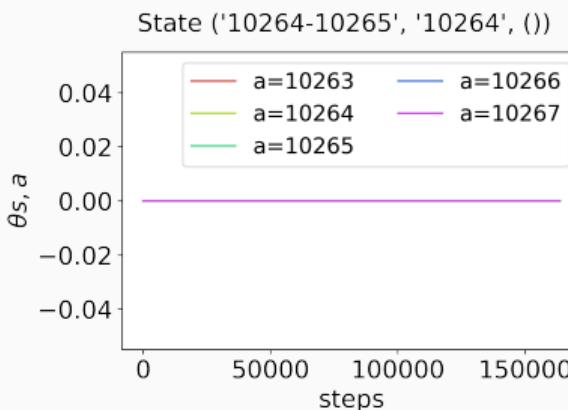
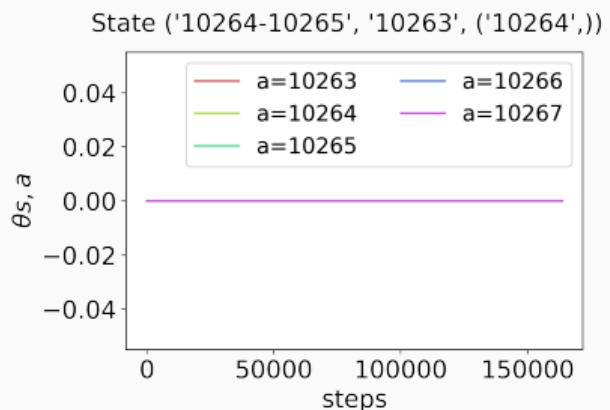
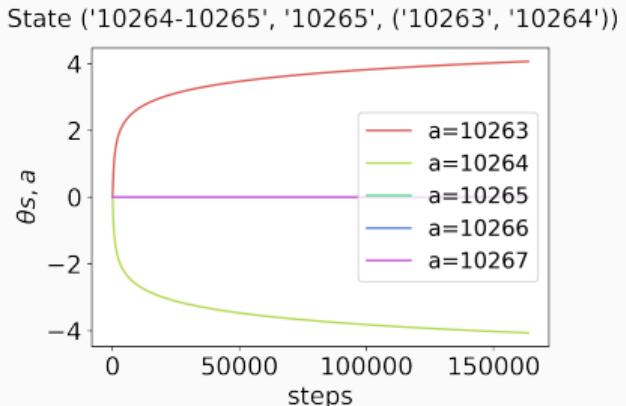
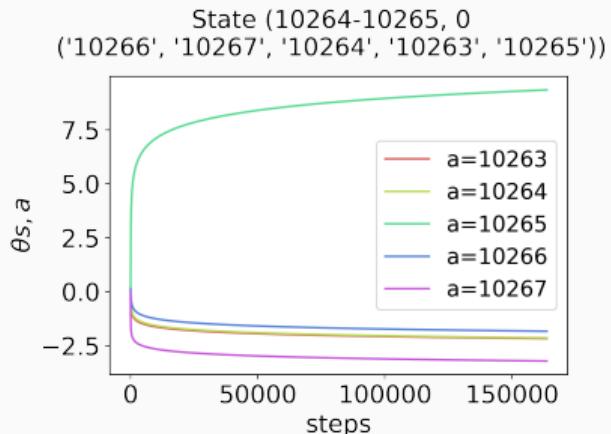
PG error trend (log-log plot)



NPG error trend (semi-log plot)



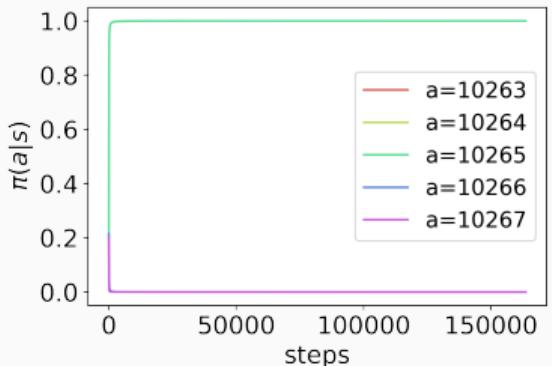
Trajectories of the parameters θ for PG



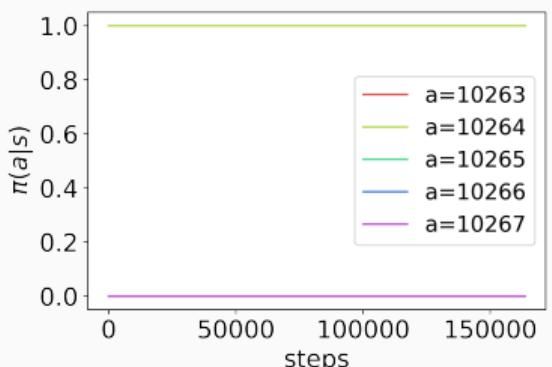
Trajectories of the policies π_θ for PG



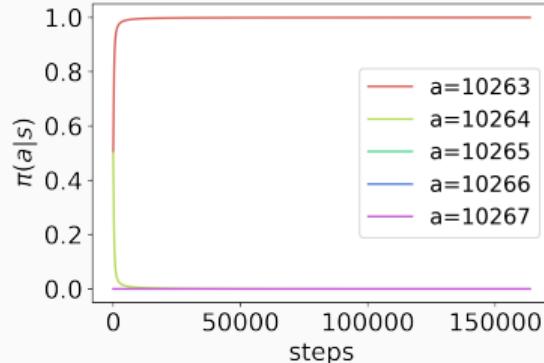
State (10264-10265, 0
('10266', '10267', '10264', '10263', '10265'))



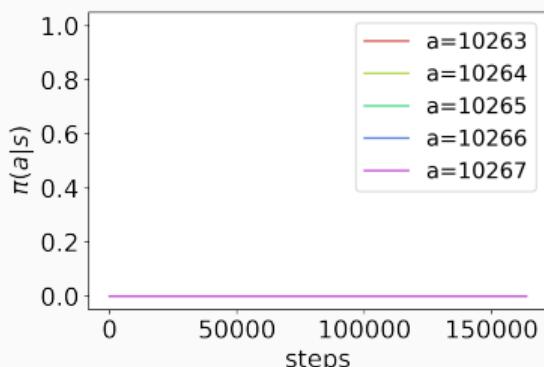
State ('10264-10265', '10263', ('10264',))



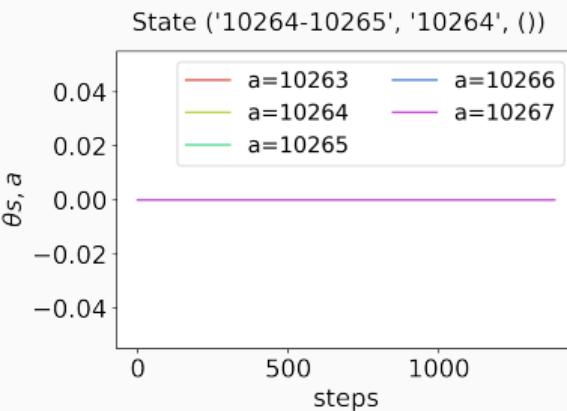
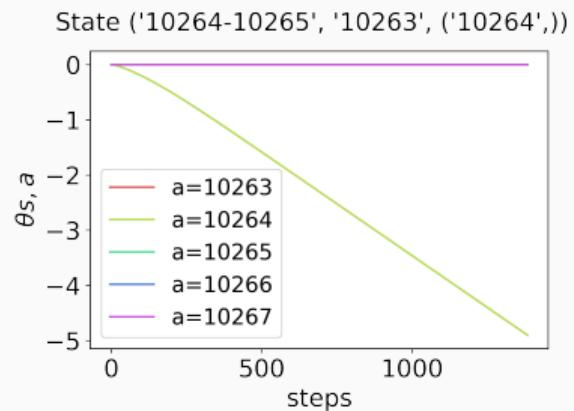
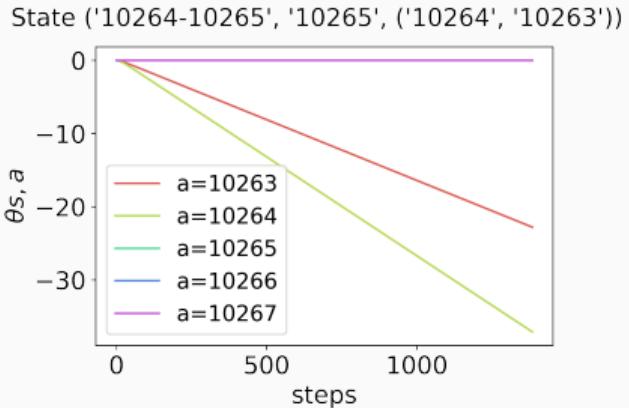
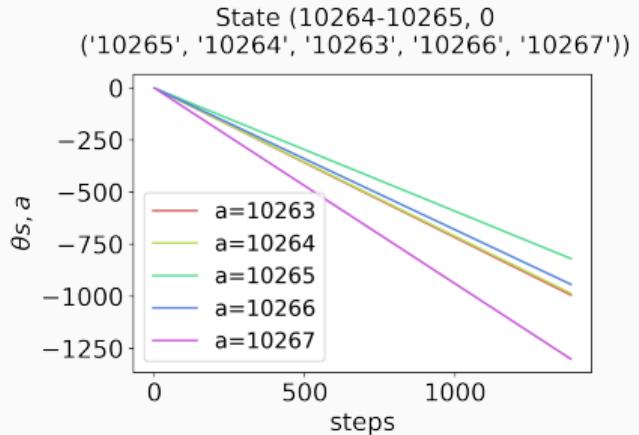
State ('10264-10265', '10265', ('10263', '10264'))



State ('10264-10265', '10264', ())



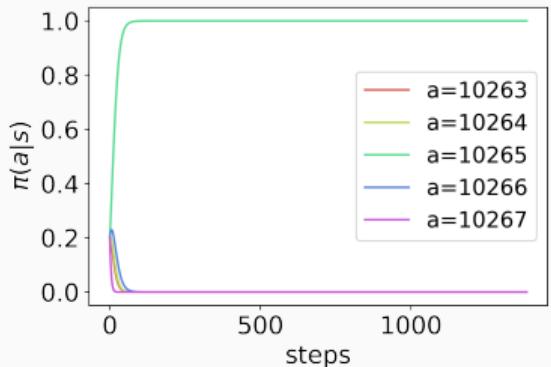
Trajectories of the parameters θ for NPG



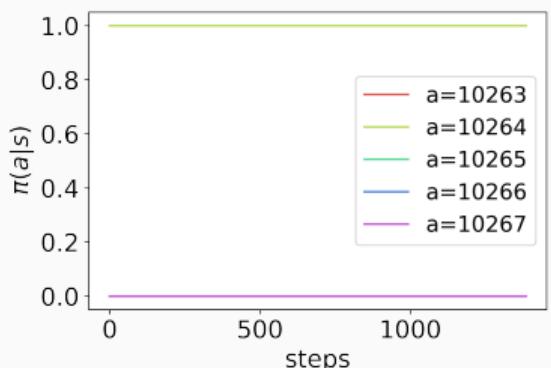
Trajectories of the policies π_θ for NPG



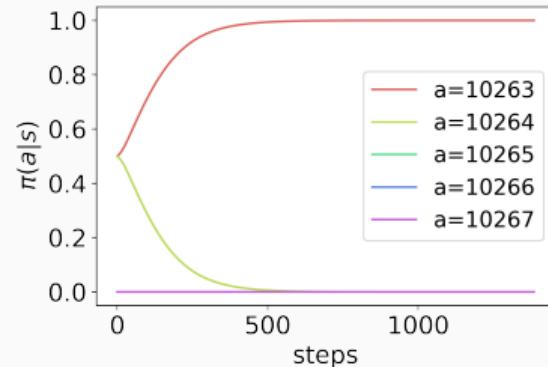
State (10264-10265, 0)
('10265', '10264', '10263', '10266', '10267')



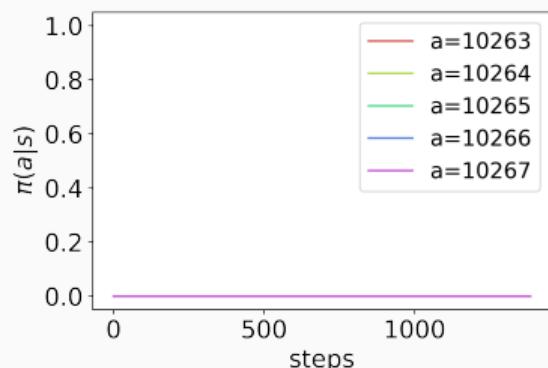
State ('10264-10265', '10265', ('10264', '10263'))



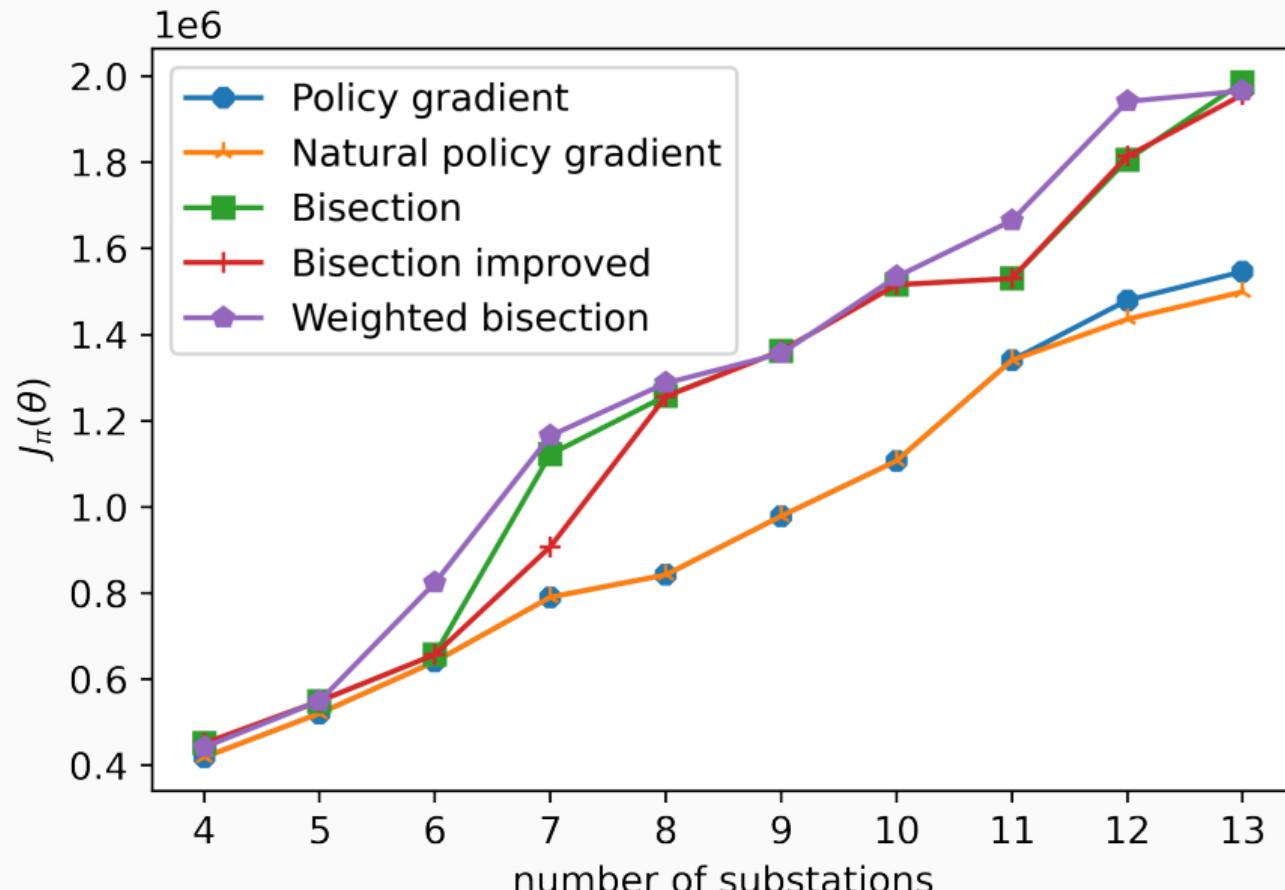
State ('10264-10265', '10265', ('10264', '10263'))

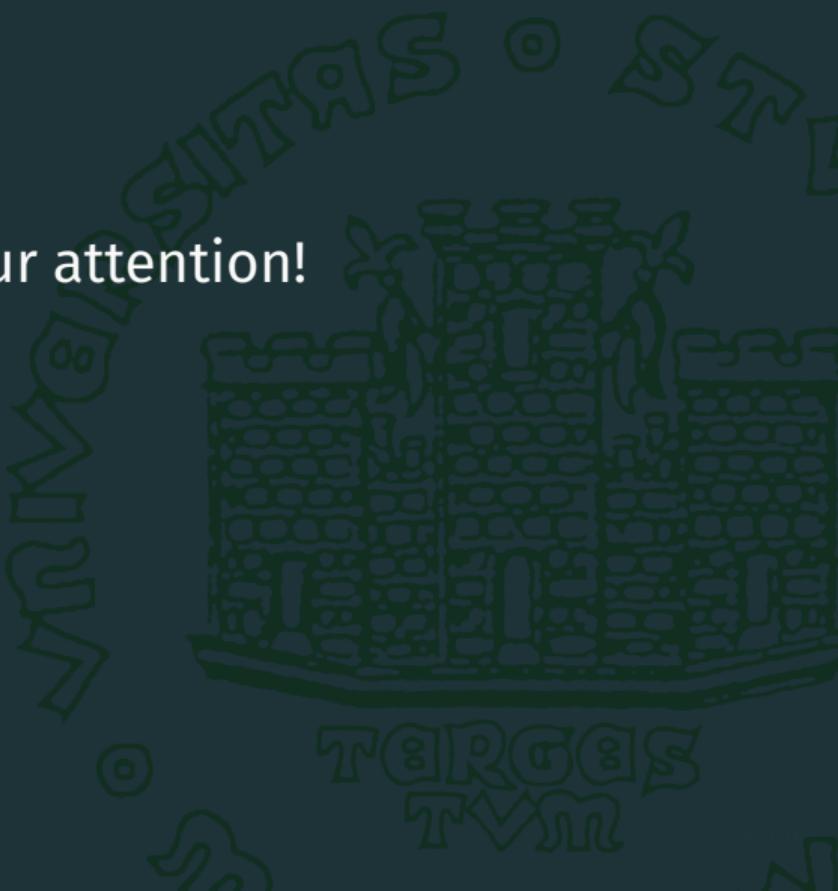


State ('10264-10265', '10264', ())



Performance comparison



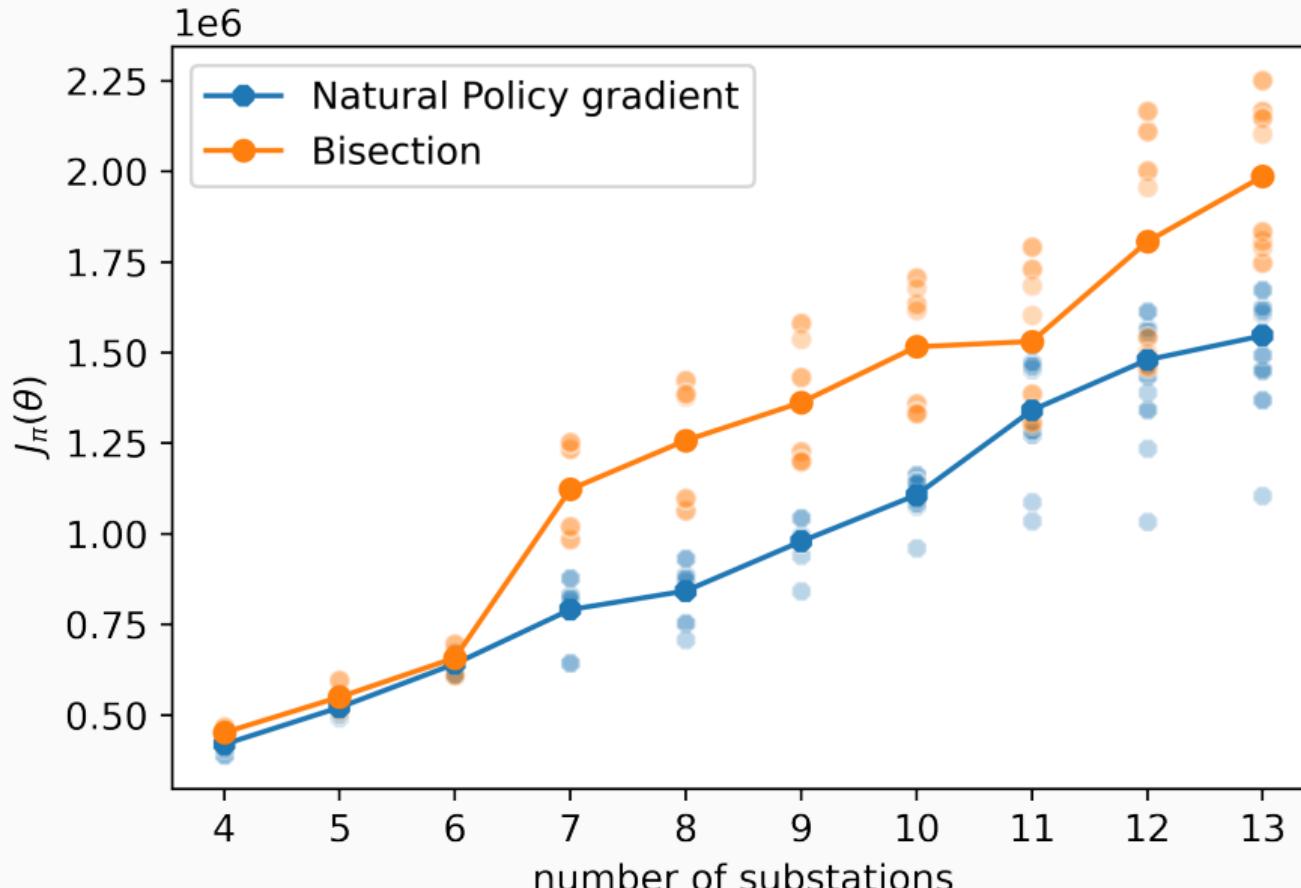


Thank you for your attention!



Backup slides

Performance comparison



11 substations

