



As you enjoy your walk, please remember:

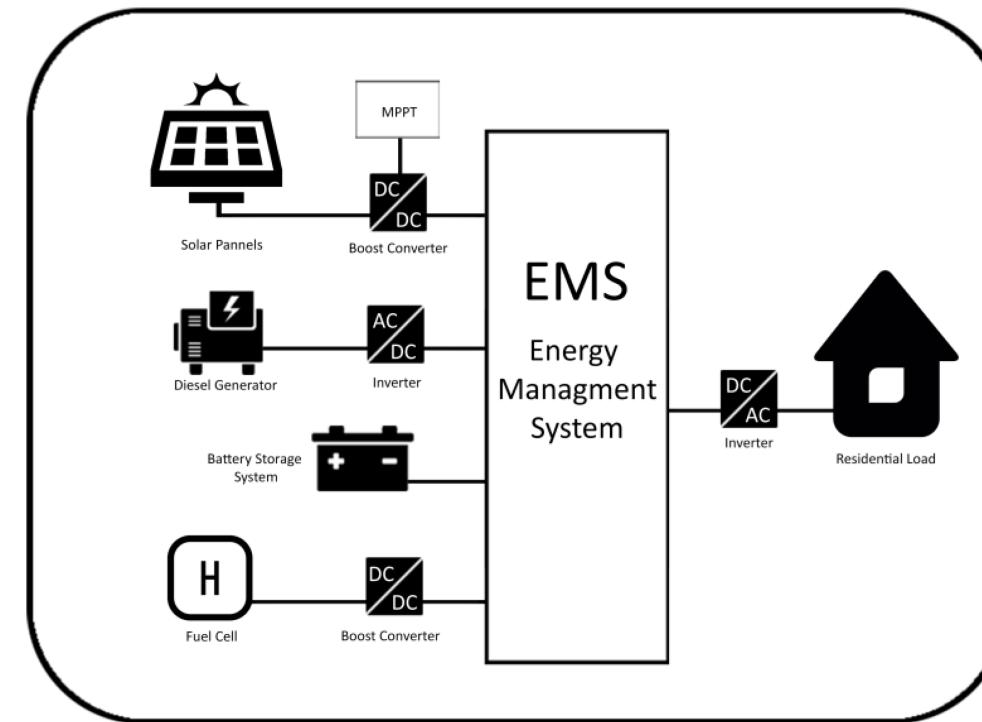
## 05. Last Leg of the Journey



# Multi-Stage Stochastic Optimization

What if we have a sequence of decision stages?

Consider for example an Energy Management System:



- We need to make some decisions (using a generator, buying from the grid...)
- ...Then observe how uncertainty unfolds
- ...Based on that, we make another round of decisions and so on



# Multi-Stage Stochastic Optimization

We will also assume that there are **non-trivial constraints**

- This setup is called multi-stage stochastic optimization
- ...Or also online stochastic optimization, or sequential decision making

**There are a few possible solution approaches**

One approach consist in using scenarios, again

- ...But since there are many stages, the decisions variables branch out
- A solution is called a policy tree, which is **very** expensive to compute

A second approach consists in using anticipatory algorithms

- We iteratively solve an optimization problem with a bit of look-ahead
- Several examples can be found in [1]

[1] Hentenryck, Pascal Van, and Russell Bent. *Online stochastic combinatorial optimization*. The MIT Press, 2006.



# Formalization

**Formally, this setup is well captured by a constrained Markov Decision Process (MDP)**

In particular, we will consider a **constrained**  $\langle X, Z, P^0, P, f, F \rangle$  be an MDP with:

- A set of possible (observable) states  $X$
- A set of possible decisions  $Z$
- A distribution  $P^0(X)$  for the initial state
- A distribution  $P(X | X, Z)$  for the possible state transitions
- A cost function  $f(z, x, x^+)$
- A feasible space  $F(x)$  which depends on the state

**Some comments:**

- The next state depends on the current state and decisions
- The cost depends on the current state and decisions, and on the next state



# Formalization

**Within this framework, we can formalize a multi-stage problem**

Our goal is to define a solution policy  $\pi^*$  from a set of candidates  $\Pi$  s.t.:

$$\pi^* = \operatorname{argmin}_{\pi \in \Pi} \mathbb{E}_{x^0 \sim P^0, x^{t+1} \sim P(X|x^t, z^t)} \left[ \sum_{t=1}^{eoh} f(z^t, x^t, x^{t+1}) \right]$$

subject to:  $z^t = \pi(x^t)$

$z^t \in F(x^t)$

This is very complex problem:

- We are not searching for a fixed solution, but for a policy
- The decisions can be anything (including discrete and combinatorial)
- ...They affect the state at the next stage (endogenous uncertainty)
- ...And they should be feasible according to hard constraints



# Solution Approach Wanted

**Normally, with an MDP we may turn to Reinforcement Learning**

...But in this case there are a couple of difficulties:

- Handling constraints (hard ones in particular) in RL is challenging
- Handling combinatorial decisions in RL is **very** challenging

**Let's recap our situation**

- Classical approaches from stochastic optimization have poor scalability
- RL approaches have poor support for constraints and combinatorial spaces

**Can we use DFL in this scenario?**

[1] Garcia, Javier, and Fernando Fernández. "A comprehensive survey on safe reinforcement learning." *Journal of Machine Learning Research* 16.1 (2015): 1437-1480.

## DFL and RL (UNIFY)

**Indeed we can, and at this point it's not even that difficult**

The trick is simply to decompose the policy  $\pi$ , leading to:

$$\theta^* = \operatorname{argmin}_{\theta} \mathbb{E}_{x^0 \sim P^0, x^{t+1} \sim P(X|x^t, z^t)} \left[ \sum_{t=1}^{eoh} f(z^t, x^t, x^{t+1}) \right]$$

$$\text{subject to: } z^t = z^*(y^t, x^t)$$

$$y^t = h(x^t, \theta)$$

Intuitively:

- We use a ML model to output a set of virtual parameters  $y$
- ...Then we compute  $z^k$  by solving a constrained optimization problem
- The ML model take care of uncertainty
- The optimization problem take care of the constraints



## DFL and RL (UNIFY)

We use the generalized, surrogate-based approach to compute  $z^*$

In particular, we have:

$$z^*(y, x) = \operatorname{argmin}_z \{\tilde{f}(z, y, x) \mid z \in \tilde{F}(y, x)\}$$

- Depending on our choice for the virtual parameters
- We will need to craft the surrogate cost  $\tilde{f}$  and feasible space  $\tilde{F}$
- The original constraints are satisfied as long as  $z \in F(y, x) \Rightarrow z \in F(x)$

The surrogate terms can usually be designed by tweaking a bit  $f$  and  $F$

The overall idea is that the ML model guides the optimizer,  
exactly as in normal DFL



## DFL and RL (UNIFY)

**For training, we can rely on a simple reformulation**

In particular, we define a new **unconstrained** MDP  $\langle X, \Theta, P^0, P, g \rangle$  such that:

- The set of states is the same as before
- The set of states is the set  $\Theta$  of possible training parameters
- The state transition distributions are the same as before
- The cost function is defined as:

$$g(y, x, x^+) = f(z^*(y), x, x^+)$$

**Intuitively, we treat the solver as part of the environment**

This new MDP can be addressed by **any RL learning approach**

so we can benefit from recent advances in such field



## DFL and RL (UNIFY)

**This setup is the most general we have seen so far**

It can be used to address a wide number of problem types

- Optimization with parameters that need to be estimated
- One-stage stochastic programming
- Two-state stochastic programming
- Sequential decision making with constraints
- In principle, also black-box optimization and parameter tuning
- ...Though it probably would not a good fit for such cases

You can find it described in [1], under the name **UNIFY**

[2] Silvestri, Mattia, et al. "UNIFY: a Unified Policy Designing Framework for Solving Constrained Optimization Problems with Machine Learning." arXiv preprint arXiv:2210.14030 (2022).



# An Example

## Let's consider the Energy Management System example in detail

Every 15 minutes, we need to adjust power flow to/from a set of nodes

- Nodes can be generators, demand points, or the grid
- One special node represents a storage system

### The decisions $z^t$ at time $t$ include:

- A vector of power flows  $z_{nodes}^t$  to/from the main nodes
- A power flow  $z_{storage}^t$  to/from the storage system

### The state $x^t$ at time $t$ is given by:

- The power  $x_{power}^t$  generated by some nodes (e.g. PV plants)
- The demand  $x_{demand}^t$  for some nodes (e.g. production sites or housing)
- The storage charge level  $x_{storage}^t$



# An Example

The transition distribution  $P$  is defined by:

- A distribution  $P_{power}$  of the yield of renewable energy generators
- A distribution  $P_{demand}$  of the demand
- The deterministic transition  $x_{storage}^{t+1} = x_{storage}^t + \eta z_{storage}$

The feasible space  $F(x^t)$  is defined via:

- Flow capacity constraints:  $l \leq z^t \leq t$
- Flow balance constraints:  $1^T z + x_{power} - x_{demand} = 0$
- Storage capacity constraints  $0 \leq x_{storage}^t + \eta z_{storage} \leq C$

The cost  $f(z^t, x^t, x^{t+1})$  is given by:

$$f(z^t, x^t, x^{t+1}) = c^T z_{nodes}$$

- There is no cost associate to demands, renewable generators, and the storage

# The Optimization Problem

We can compute  $z^*(y, x)$  by solving the following LP

$$\operatorname{argmin}_z c^T z_{nodes} + yz_{storage}$$

subject to:  $l \leq z^t \leq t$

$$1^T z + x_{power} - x_{demand} = 0$$

$$0 \leq x_{storage}^t + \eta z_{storage} \leq C$$

The main alteration is that a virtual cost is associated to the storage system

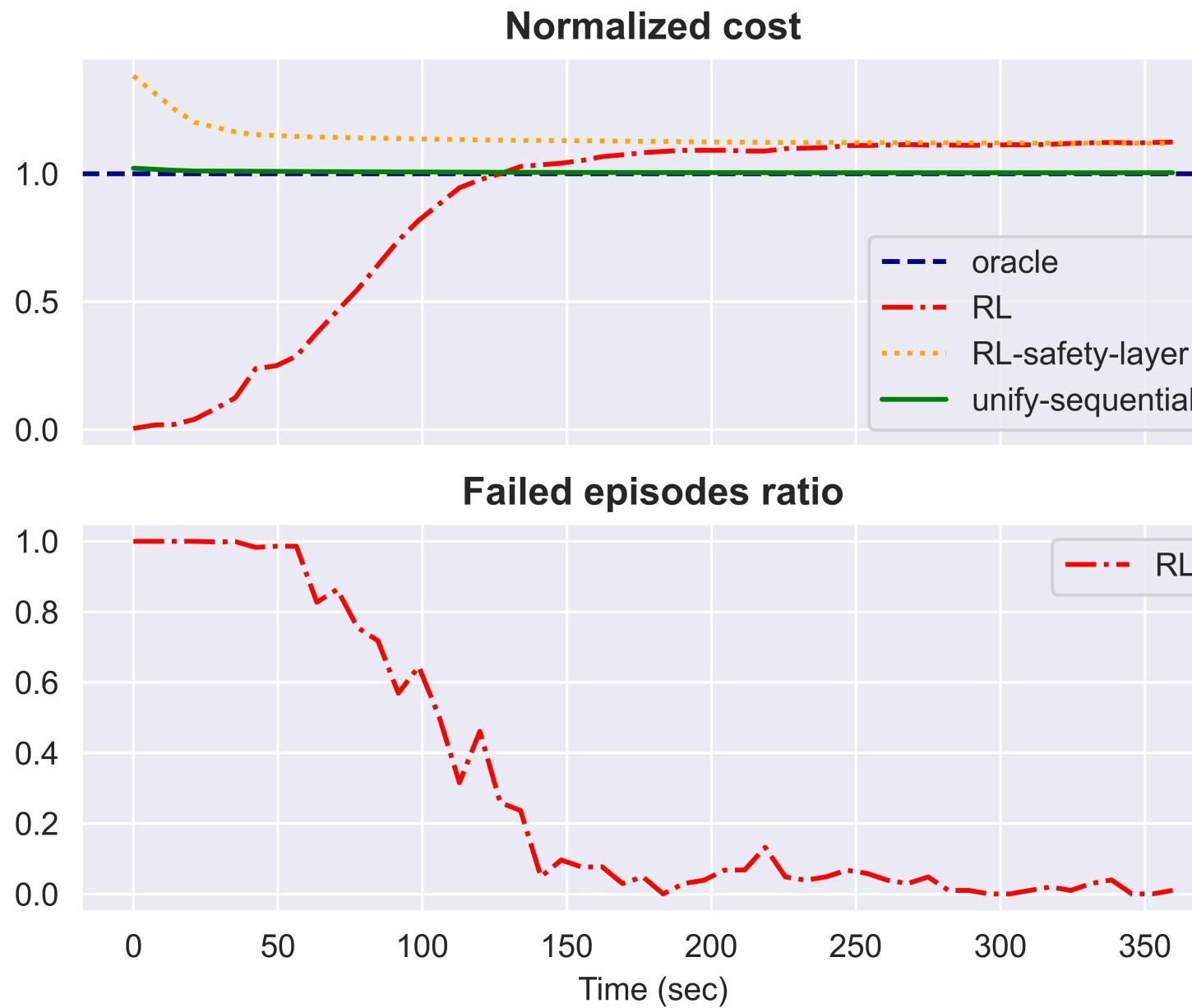
- If  $y > 0$ , the solve will tend to charge the storage
- If  $y < 0$ , the solve will tend to draw power from the storage
- ...So that the ML model can alter the decisions

**Without the virtual cost, the storage system would never be charged**



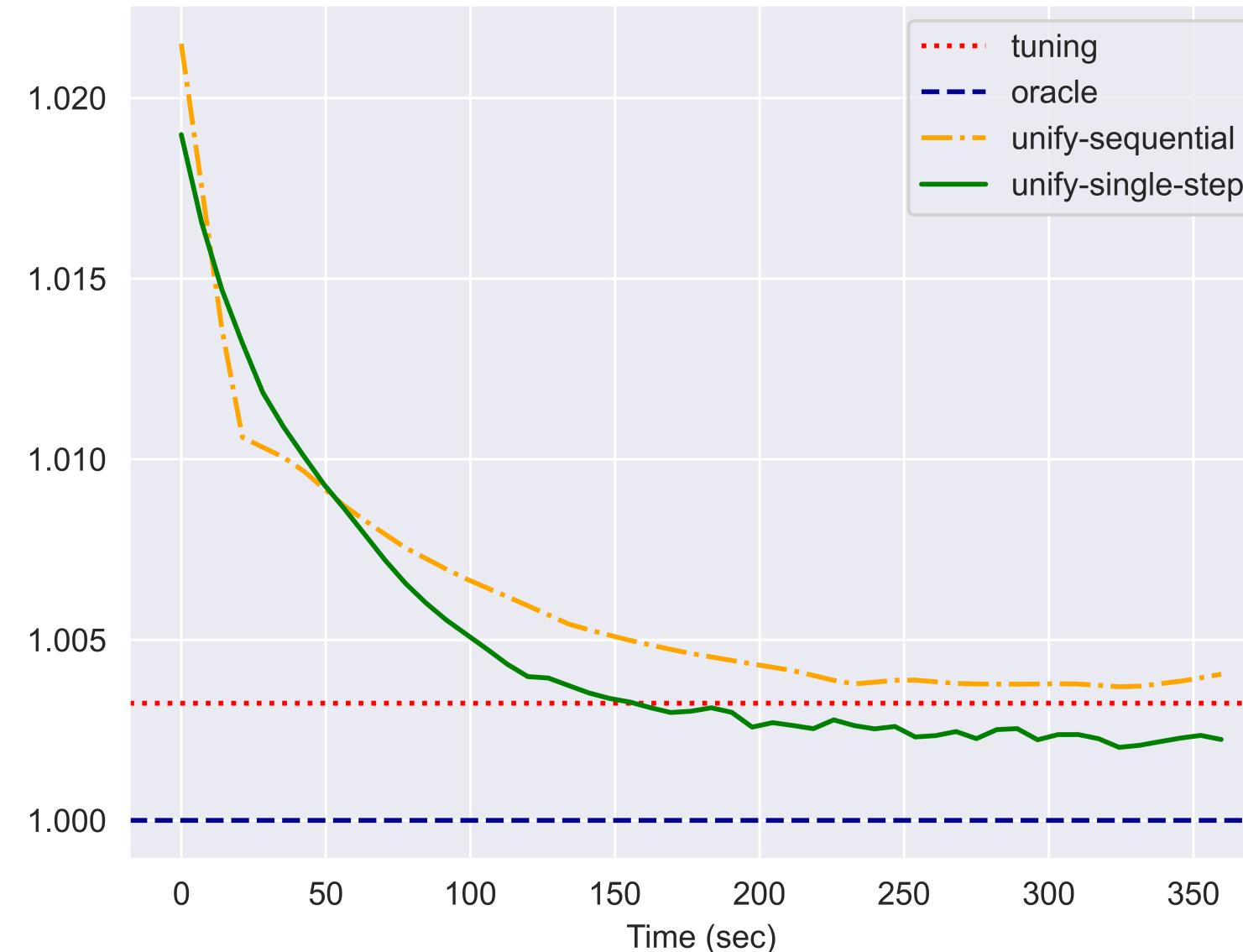
# Some Results

Here's a comparison with some constrained RL methods



# Some Results

And here's a comparison with a specialized stochastic optimization approach



# Some Final Thoughts

If you retain one idea from our ramble, makes sure it is this:

**DFL can be used for **way** more than one purpose!**

You just need to stretch it a little bit ;-)

## Where next?

- We can reap what we haven't sowed! Let's test more RL algos (spoiler: started)
- Scalability is still a big issue
- We need more (and more realistic) applications
- ...



**Thanks for your patience! Any question?**

