Decision-Focused Learning for Multistage DC-OPF Repair under Weather-Driven Failures

A linear Bellman surrogate for fast, interpretable repair scheduling.

David Krame Kadurha, Alessandro Zocca, Sandjai Bhulai | *Vrije Universiteit Amsterdam* | **DTU PES Summer School 2026**

d.kramekadurha@vu.nl

TL;DR. A small neural network learning a **linear Bellman surrogate** recovers a competitive multistage repair policy: it matches or outperforms AlwaysRepair on `toy` and `case14`, and reveals that **immediate repair is not always optimal** – repair pays only when the predicted forward value beats the marginal repair cost.

10× – 100× faster

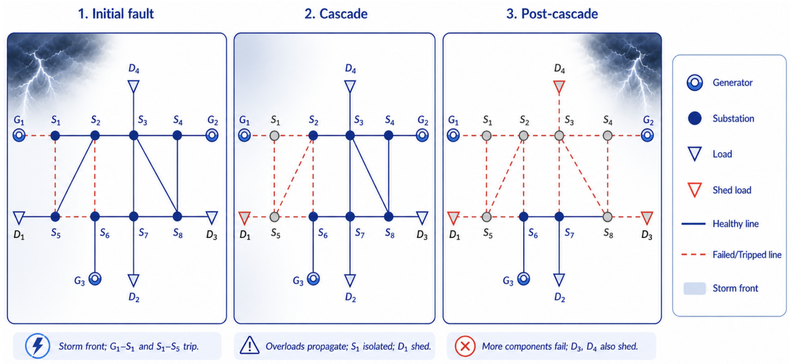
than SAA / SDDiP overall (training + deployment)

01 PROBLEM & MODEL

From a weather-driven outage to a tractable repair model.

1. Motivation: weather, repair, and a learned policy

Extreme weather increasingly cascades through power grids into wide-area load shedding. **Flexibility alone is not enough**: storage, redispatch and curtailment act on minutes, while weather-driven outages unfold over days, and lost connectivity returns only when failed components are **repaired**.



Repair scheduling is therefore the binding decision: over a T-stage horizon, decide **which** components to repair and **when**, given a τ-stage delay and uncertain future failures. Classical SAA / SDDiP do not scale well here – the binary availability state grows as $2^{(1+\tau)(N_G+E)}$.

2. Optimization model: decisions, dynamics, uncertainty

Stage-t decisions. Generation *dispatch* p_g^t, *served load* p_l^t (shed $= d^t - p_l^t$), and binary *repair activations* r_g^t, r_e^t on failed generators / lines, with a τ-stage delay.

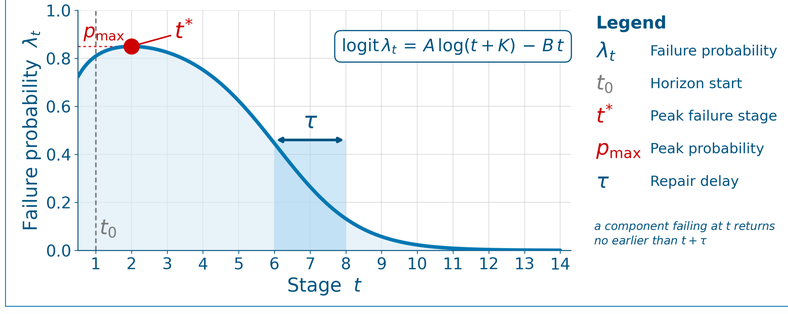
Objective.

$$\min \mathbb{E}\Big[\sum_{t=1}^{T}\Big(c^\top p_g^t + \alpha\,\mathbf{1}^\top(d^t - p_l^t) + \beta_g^\top r_g^t + \beta_e^\top r_e^t\Big)\Big]$$

DC-OPF + load-shedding [5], coupled to availability maps a_g^t, a_e^t:

$$CBC^\top\theta^t = \big[a_g^t \odot \overline{p}_g,\ -p_l^t\big]^\top, \quad \mathbf{0} \le p_g^t \le a_g^t \odot \overline{p}_g.$$

Stochastic failure model. Components fail *independently* but with non-constant probabilities driven by the same **log-linear logit** curve, parameterized by peak stage t^* and peak probability $p_{\max}$ – a common weather hazard hitting the whole grid at once. It is possible that a *repaired* component may fail again later in the horizon.



02 METHOD

Replacement of the Bellman recursion with a learned linear surrogate.

3. SAA / SDDiP / DFL: idea, complexity, scaling

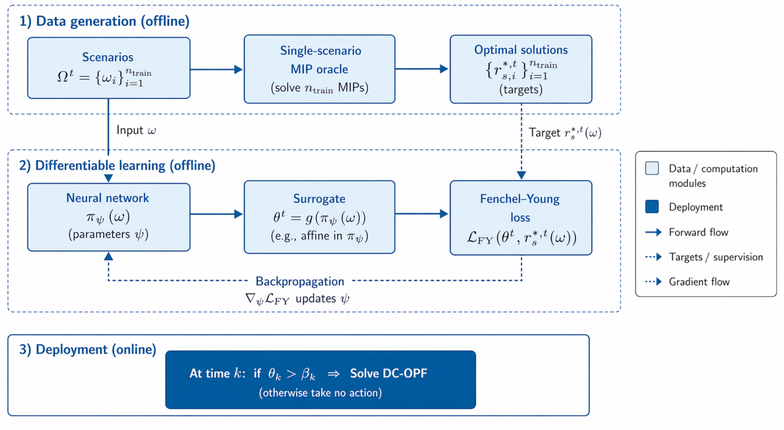
	Idea	Complexity / scaling
SAA	One extensive-form MIP over n_{scen} scenarios.	Exact in the limit; combinatorial in scenarios *and* stages.
SDDiP [1]	Bellman recursion on the binary state, refined by Lagrangian cuts.	Finite convergence; **exponential backward pass** in $N_G{+}E$.
DFL *(ours)*	Learn a **linear surrogate** of the cost-to-go; deploy one MIP per stage.	Forward pass + DC-OPF LP; **10×–100× faster**.

More on SDDiP. Forward passes sample paths; backward passes add integer *Lagrangian cuts* on the expected future cost V_{t+1}. Each cut needs a relaxation exact on $\{0,1\}^{N_G+E}$, so the backward pass is **exponential in $N_G{+}E$**; we use SDDiP only for *LB / UB* brackets. Within a **48 h** budget, SAA returns essentially **NoRepair-quality solutions** on every network (**2.51 M** on toy and **3.62 M** on case14) – This intractability motivates trying DFL.

4. DFL in detail: linear surrogate & Fenchel–Young loss

We replace the unknown expected future cost by a **linear surrogate** $\hat{V}_{t+1} = -\theta_g^\top r_g^t - \theta_e^\top r_e^t$, whose coefficients $\theta^t = (\theta_g^t, \theta_e^t)$ are predicted by a neural network $\pi_\psi(\Omega^t)$ from the information set $\Omega^t = (a, q, \xi, \lambda, t)$.

Training. The per-stage MIP is treated as a *perturbed argmax* [4], which makes it differentiable. The **Fenchel–Young loss** [3] $\mathcal{L}_{\text{FY}}(\theta^t, r^{*,t})$ is then a convex, differentiable surrogate that pushes π_ψ toward predictions whose perturbed optimum agrees with the target $r^{*,t}$. Gradients flow *through* the optimizer: the LP becomes a trainable layer, end-to-end.



Why isn't immediate repair always optimal? The per-stage decision collapses to a closed-form test **repair k $\iff$ $\theta_k^t > \beta_k$**: repair only when the **predicted forward value** θ_k^t beats the **immediate repair cost** β_k. **Late in the horizon, or when a line is unlikely to be reused, DFL declines to repair** – an option **AlwaysRepair** cannot exploit.

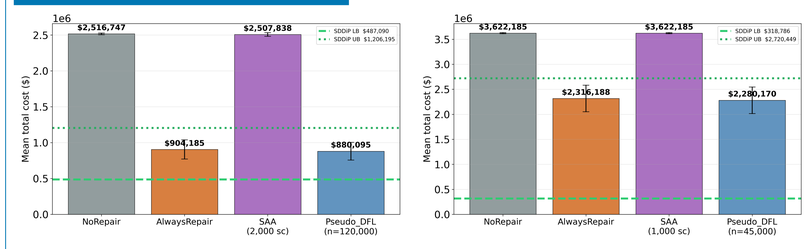
5. Experimental setting

- **Networks.** `toy` (5-bus / 2-gen / 5-line), IEEE `case14`, `case30`; T=14, τ=1.
- **Costs.** α=1,000 m.u./MWh shed; $\beta_g \approx 550$ m.u., $\beta_e \approx 350$ m.u..
- **Failures.** logit hazard, t^*=1, K=1.5; 50/50 mix of $p_{\max} \in [0.55, 0.70]$ and $[0.80, 0.90]$.
- **Baselines.** NoRepair, AlwaysRepair, SAA, SDDiP.
- **Training.** MLP 128→256→128; Adam lr=10^{-3}; FY loss ε=0.1; $n_{\text{train}} \in [25, 10^5]$; 500 held-out scenarios.

03 RESULTS

DFL generally works better and 10×–100× faster than classical approaches.

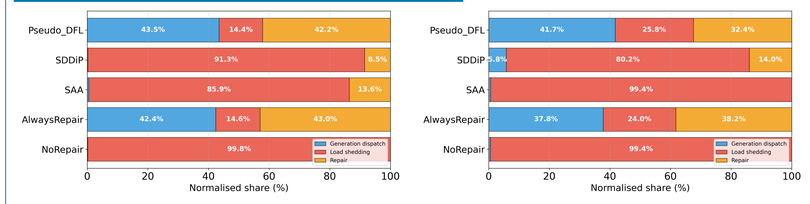
6. Cost: how good is the policy?



toy *case14*

DFL is the lowest-cost method on `toy` (**880 k** vs. AlwaysRepair's 904 k) and on `case14` (**2.28 M** vs. 2.31 M); on `case30` **DFL** ≈ 10% above AlwaysRepair.

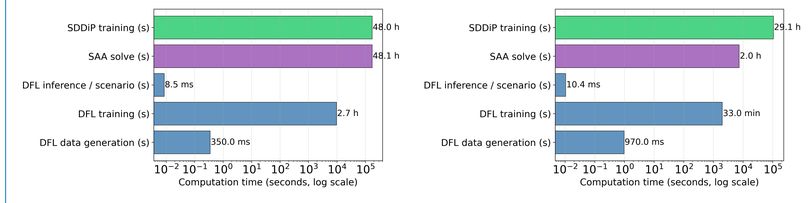
7. Cost composition: where the budget goes



toy *case14*

DFL spends roughly the same repair budget as AlwaysRepair, but on the *right* components, leading to a lower overall cost.

8. Speed: orders of magnitude at deployment



toy *case14*

After training, at deployment, we have one forward pass plus a per-stage DC-OPF LP. – the runtime gap to SAA / SDDiP spans several orders of magnitude, and DFL parallelises trivially across scenarios.

9. What's next

- **`case30` ongoing work.** Best test cost at n_{train}=100, degrading thereafter – a sign that linear surrogates lose expressiveness as networks grow.
- **Richer neural network architectures.** A trade-off between simplicity and expressiveness is to be investigated.
- **Distributionally-robust training of π_ψ.** To stay safe under hazard misspecification (e.g. wrong t^* or $p_{\max}$).
- **Richer surrogates.** Piecewise-linear or other types of cost-to-go functions trained with the same Fenchel-Young pipeline.
- **Reinforcement-learning techniques.** We still need to explore further how well RL will perform in situations with more difficult actions.
- **Scaling.** Larger benchmarks (`case57` and beyond) and more challenging weather scenarios will also be investigated.

Acknowledgements. This work is part of the project *Power Network Optimization in the Age of Climate Extremes*, funded by NWO via the **VIDI** grant (`nwo.nl/projecten/gooel09973`) awarded to Alessandro Zocca.

References. [1] Zou, Ahmed, Sun (2019) *Stochastic Dual Dynamic Integer Programming.* [2] Dowson, Kapelevich (2021) *SDDP.jl.* [3] Blondel et al. (2020) *Learning with Fenchel–Young Losses.* [4] Dalle et al. (2022) *InferOpt.jl.* [5] Zhou, Anderson, Low (2020) *The OPF Operator.*