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

A linear Bellman surrogate for fast, interpretable repair scheduling

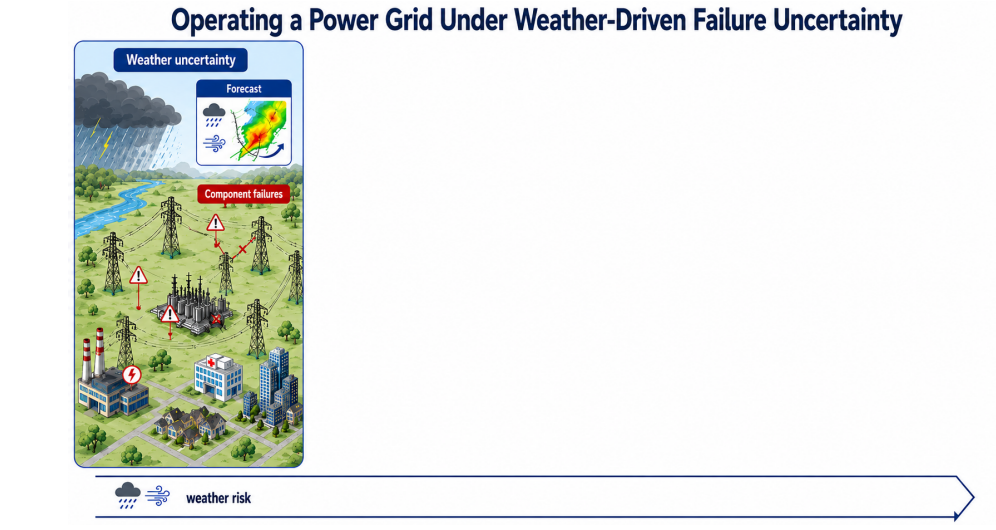
David Krame Kadurha

Alessandro Zocca & Sandjai Bhulai & Robbert van der Burg

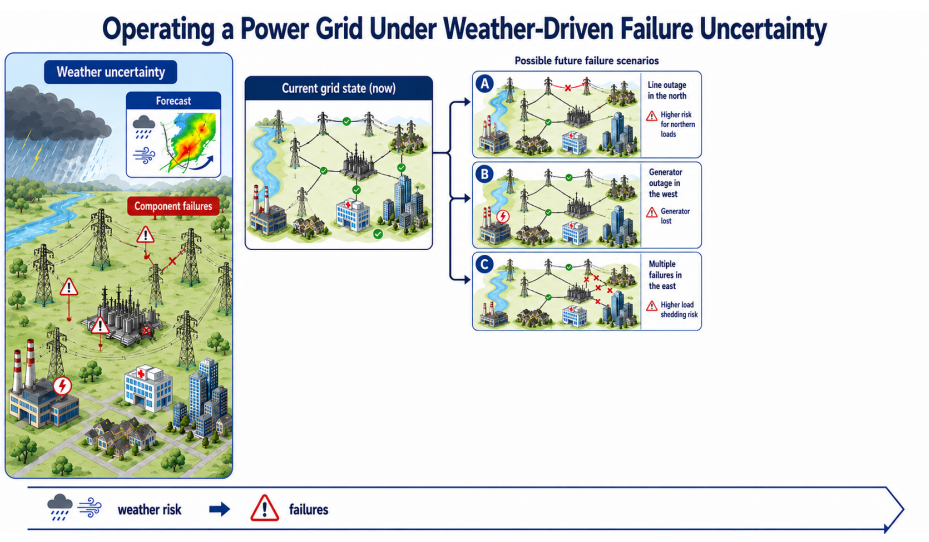
Vrije Universiteit Amsterdam

StochMod 2026

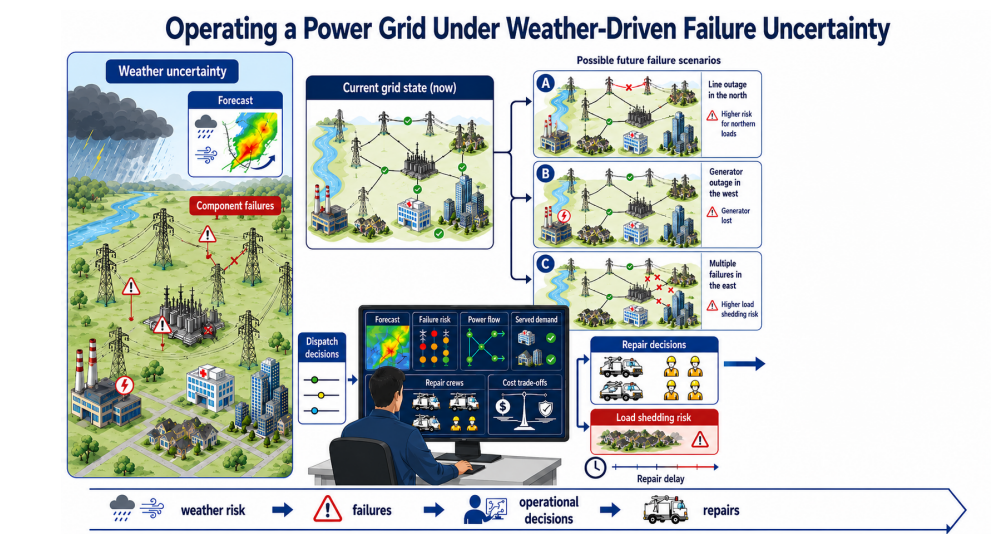
Problem: Operating a Grid Under Weather-Driven Failures



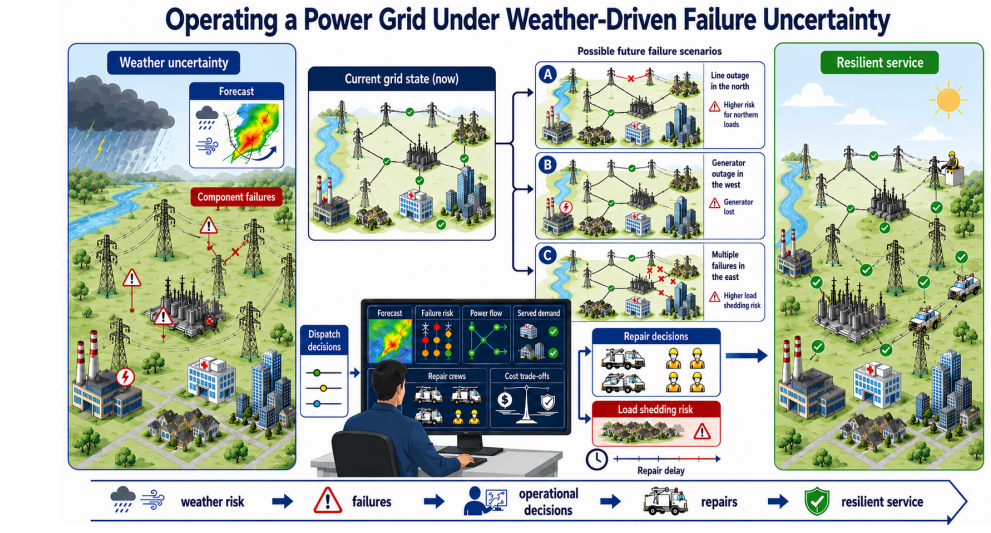
Problem: Operating a Grid Under Weather-Driven Failures



Problem: Operating a Grid Under Weather-Driven Failures

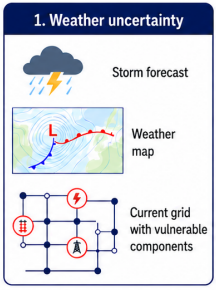


Problem: Operating a Grid Under Weather-Driven Failures



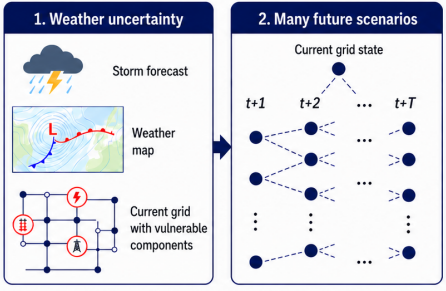
Approach: Motivation & Offline Learning

Weather uncertainty creates many possible futures, so exact look-ahead optimization must be repeated many times.



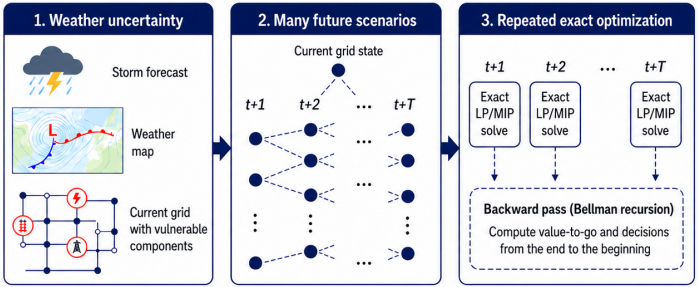
Approach: Motivation & Offline Learning

Weather uncertainty creates many possible futures, so exact look-ahead optimization must be repeated many times.

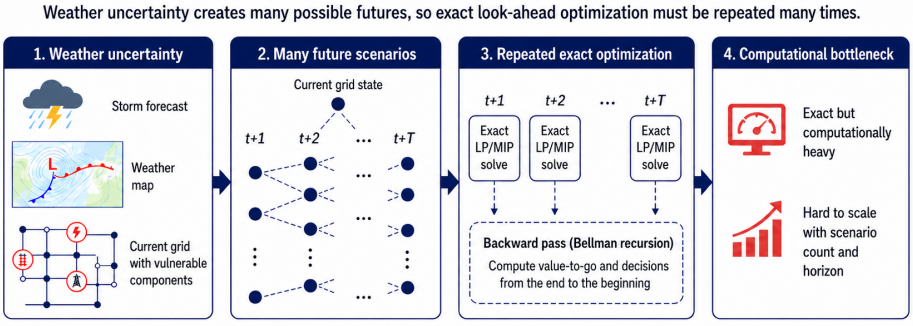


Approach: Motivation & Offline Learning

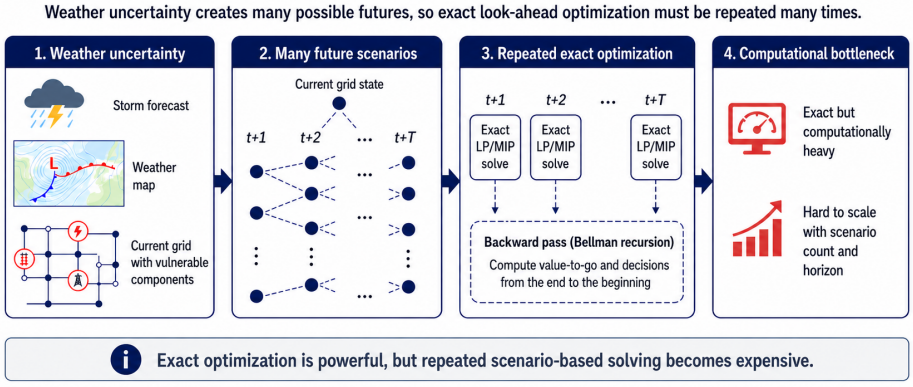
Weather uncertainty creates many possible futures, so exact look-ahead optimization must be repeated many times.



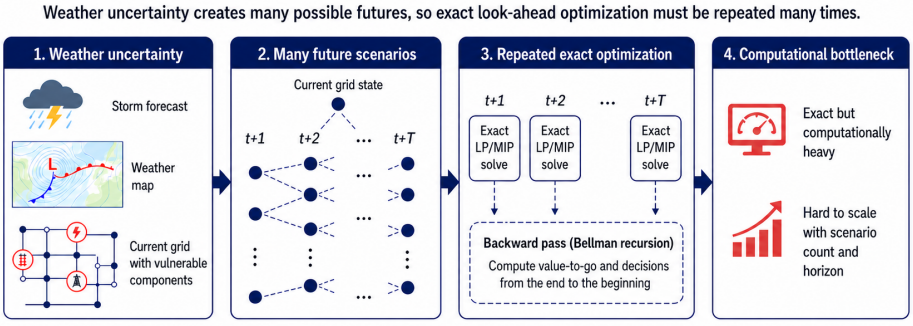
Approach: Motivation & Offline Learning



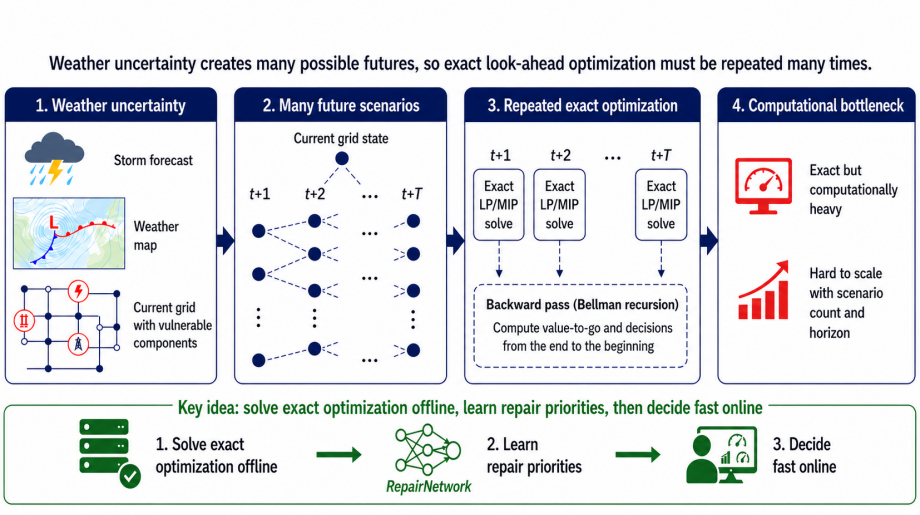
Approach: Motivation & Offline Learning



Approach: Motivation & Offline Learning

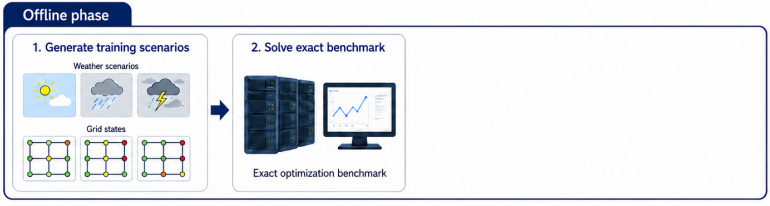


Approach: Motivation & Offline Learning



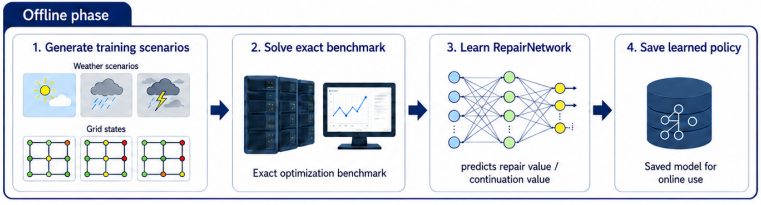
Approach: Online Deployment & Full Pipeline

Use exact optimization offline to train a repair-value policy, then use that policy online.



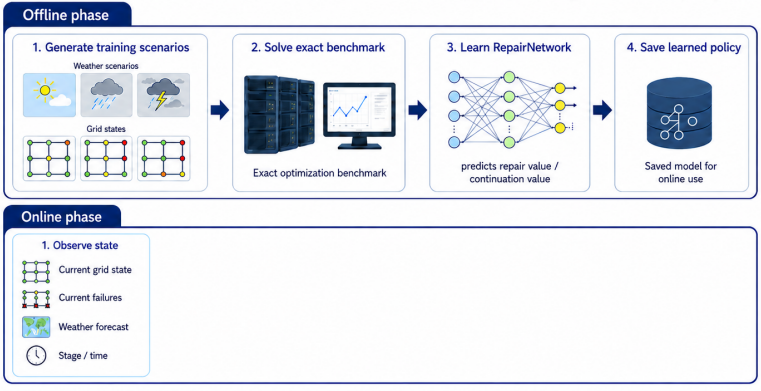
Approach: Online Deployment & Full Pipeline

Use exact optimization offline to train a repair-value policy, then use that policy online.



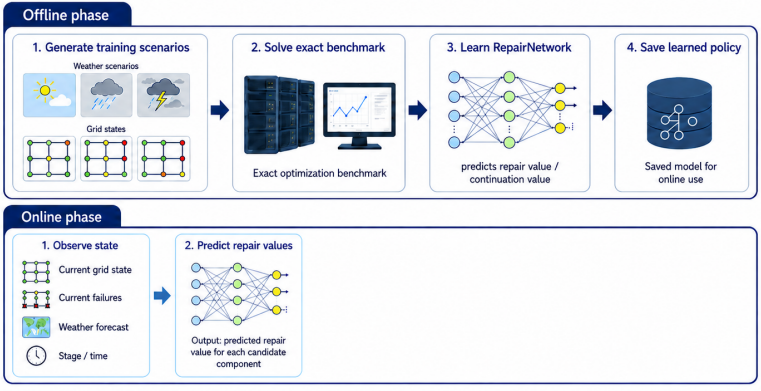
Approach: Online Deployment & Full Pipeline

Use exact optimization offline to train a repair-value policy, then use that policy online.



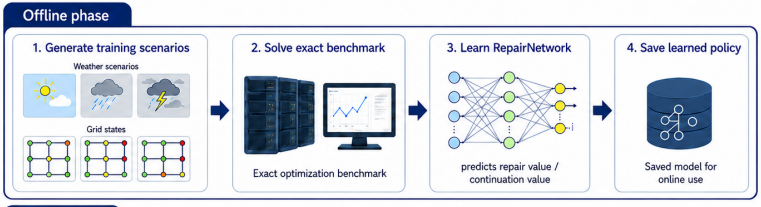
Approach: Online Deployment & Full Pipeline

Use exact optimization offline to train a repair-value policy, then use that policy online.



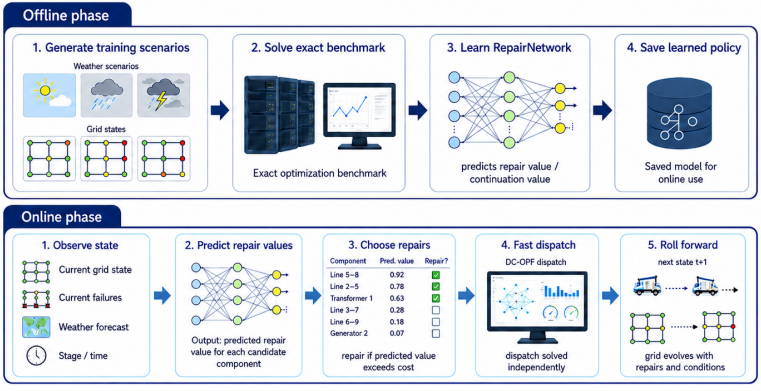
Approach: Online Deployment & Full Pipeline

Use exact optimization offline to train a repair-value policy, then use that policy online.

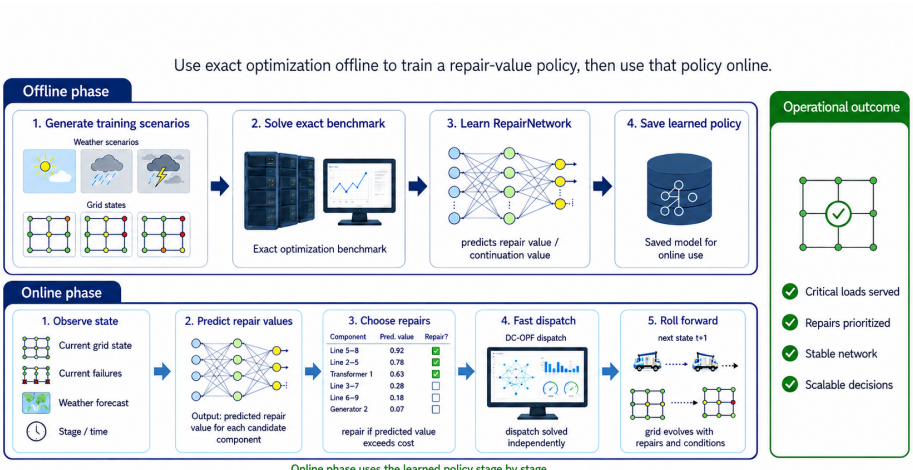


Approach: Online Deployment & Full Pipeline

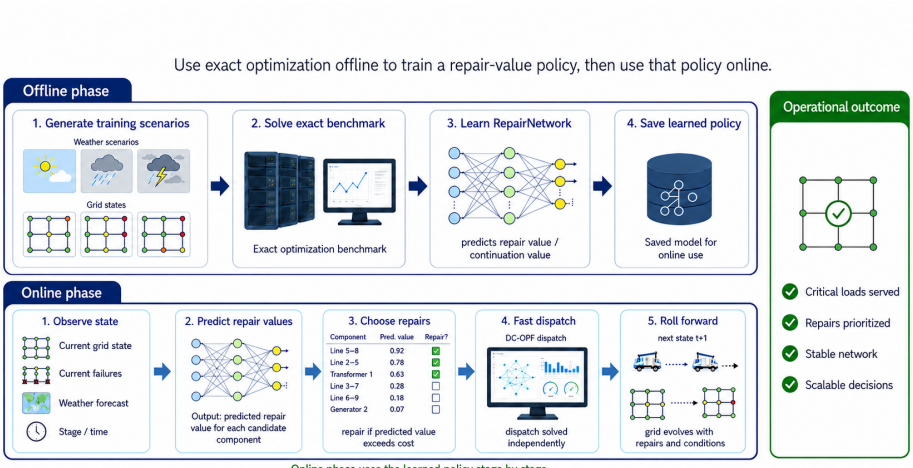
Use exact optimization offline to train a repair-value policy, then use that policy online.



Approach: Online Deployment & Full Pipeline



Approach: Online Deployment & Full Pipeline



Model: Multistage Stochastic DC-OPF with Failures

Objective: minimise expected operating cost

$$\min \ \mathbb{E}\left[\sum_{t=1}^{T}\Big(\underbrace{c^{\top} p_g^t}_{\text{generation}} + \underbrace{\alpha \mathbf{1}^{\top}(p_l^t - d^t)}_{\text{load shedding}} + \underbrace{\beta_g^{\top} r_g^t + \beta_e^{\top} r_e^t}_{\text{repairs}}\Big)\right]$$

Key notation

$c,\ \alpha,\ \beta_g,\ \beta_e$	cost parameters
$p_g^t,\ d^t$	dispatch & demand served
$r_g^t,\ r_e^t \in \{0,1\}$	repair decisions
$a_g^t,\ a_e^t \in \{0,1\}$	availability state
$\xi_\gamma^t[k] \in \{0,1\}$	failure shock
$\lambda_\gamma^t[k] \in (0,1)$	failure probability
$\tau,\ T$	repair delay, horizon

Model: Multistage Stochastic DC-OPF with Failures

Objective: minimise expected operating cost

$$\min \ \mathbb{E}\left[\sum_{t=1}^{T}\Big(\underbrace{c^\top p_g^t}_{\text{generation}} + \underbrace{\alpha \mathbf{1}^\top (p_l^t - d^t)}_{\text{load shedding}} + \underbrace{\beta_g^\top r_g^t + \beta_e^\top r_e^t}_{\text{repairs}}\Big)\right]$$

At each stage t we decide

How much to generate (p_g^t) and **how much load to serve** (d^t), subject to DC power flow and generation limits.

Whether to start a repair for each failed component ($r_g^t, r_e^t \in \{0,1\}$). Repairs become effective after τ stages.

Power flow & repair feasibility constraints in Backup A.

Key notation

$c,\ \alpha,\ \beta_g,\ \beta_e$	cost parameters
$p_g^t,\ d^t$	dispatch & demand served
$r_g^t,\ r_e^t \in \{0,1\}$	repair decisions
$a_g^t,\ a_e^t \in \{0,1\}$	availability state
$\xi_\gamma^t[k] \in \{0,1\}$	failure shock
$\lambda_\gamma^t[k] \in (0,1)$	failure probability
$\tau,\ T$	repair delay, horizon

Model: Multistage Stochastic DC-OPF with Failures

Objective: minimise expected operating cost

$$\min \ \mathbb{E}\left[\sum_{t=1}^{T}\Big(\underbrace{c^\top p_g^t}_{\text{generation}} + \underbrace{\alpha \mathbf{1}^\top (p_l^t - d^t)}_{\text{load shedding}} + \underbrace{\beta_g^\top r_g^t + \beta_e^\top r_e^t}_{\text{repairs}}\Big)\right]$$

At each stage t we decide

How much to generate (p_g^t) and **how much load to serve** (d^t), subject to DC power flow and generation limits.

Whether to start a repair for each failed component ($r_g^t, r_e^t \in \{0,1\}$). Repairs become effective after τ stages.

Power flow & repair feasibility constraints in Backup A.

Uncertainty: availability evolves stochastically

Component k can be **knocked out** by a Bernoulli failure shock $\xi_k^t \sim \mathrm{Bern}(\lambda_k^t)$, or **restored** by a completed repair.

λ_k^t is weather-calibrated: peaks shortly after the event, then decays (Backup B).

Key notation

$c,\ \alpha,\ \beta_g,\ \beta_e$	cost parameters
$p_g^t,\ d^t$	dispatch & demand served
$r_g^t,\ r_e^t \in \{0,1\}$	repair decisions
$a_g^t,\ a_e^t \in \{0,1\}$	availability state
$\xi_\gamma^t[k] \in \{0,1\}$	failure shock
$\lambda_\gamma^t[k] \in (0,1)$	failure probability
$\tau,\ T$	repair delay, horizon

Bellman Reformulation: What Gets Approximated?

Dynamic programming recursion

$$V_t(x^{\text{in}}) = \min_{u^t,\, x^{\text{out}}} \Big\{ \underbrace{c^\top p_g^t + \alpha \mathbf{1}^\top (p_l^t - d^t) + \beta_g^\top r_g^t + \beta_e^\top r_e^t}_{\text{stage cost } f_t} + \underbrace{\mathbb{E}\big[V_{t+1}(x^{\text{out}})\big]}_{\text{cost-to-go} \,\leftarrow\, \text{usually intractable}} \Big\}$$

■ generation ■ load shedding ■ repair ■ cost-to-go (intractable)

State x and control u

Availability + repair queue $\Rightarrow$ x^{in}

Dispatch + repair decisions $\Rightarrow$ u^t

Why hard: x is binary; 2^{14} states on the toy example. $\mathcal{Q}_{t+1}$ cannot be computed exactly.

Two approaches to approximate $\mathcal{Q}_{t+1}$

Bellman Reformulation: What Gets Approximated?

Dynamic programming recursion

$$V_t(x^{\text{in}}) = \min_{u^t,\, x^{\text{out}}} \Big\{ \underbrace{c^\top p_g^t + \alpha \mathbf{1}^\top (p_l^t - d^t) + \beta_g^\top r_g^t + \beta_e^\top r_e^t}_{\text{stage cost } f_t} + \underbrace{\mathbb{E}\big[V_{t+1}(x^{\text{out}})\big]}_{\text{cost-to-go} \leftarrow \text{usually intractable}} \Big\}$$

■ generation ■ load shedding ■ repair ■ cost-to-go (intractable)

State x and control u

Availability + repair queue $\Rightarrow$ x^{in}

Dispatch + repair decisions $\Rightarrow$ u^t

Why hard: x is binary; 2^{14} states on the toy example. $\mathcal{Q}_{t+1}$ cannot be computed exactly.

Two approaches to approximate $\mathcal{Q}_{t+1}$

SDDiP: linear lower bounds from the dual

Accumulates cuts over forward/backward passes.

Converges to the true $\mathcal{Q}_{t+1}$. **Exact but expensive.**

Bellman Reformulation: What Gets Approximated?

Dynamic programming recursion

$$V_t(x^{\text{in}}) = \min_{u^t,\, x^{\text{out}}} \Big\{ \underbrace{c^\top p_g^t + \alpha \mathbf{1}^\top (p_l^t - d^t) + \beta_g^\top r_g^t + \beta_e^\top r_e^t}_{\text{stage cost } f_t} + \underbrace{\mathbb{E}\big[V_{t+1}(x^{\text{out}})\big]}_{\text{cost-to-go} \,\leftarrow\, \text{usually intractable}} \Big\}$$

■ generation ■ load shedding ■ repair ■ cost-to-go (intractable)

State x and control u

Availability + repair queue $\Rightarrow$ x^{in}

Dispatch + repair decisions $\Rightarrow$ u^t

Why hard: x is binary; 2^{14} states on the toy example. $\mathcal{Q}_{t+1}$ cannot be computed exactly.

Two approaches to approximate $\mathcal{Q}_{t+1}$

SDDiP: linear lower bounds from the dual

Accumulates cuts over forward/backward passes.

Converges to the true $\mathcal{Q}_{t+1}$. **Exact but expensive.**

DFL: linear surrogate

The intractable cost-to-go $\mathbb{E}\big[V_{t+1}(x^{\text{out}})\big]$ is replaced by a **linear function of repairs**:

$$-(\theta_g^t)^\top r_g^t - (\theta_e^t)^\top r_e^t$$

$\theta^t = \pi_\psi(\Omega^t)$: RepairNetwork output.

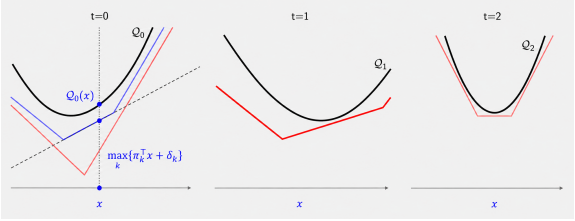
Repair k iff $\theta_k^t > \beta_k$; one LP per stage, no backward pass.

SDDiP: Building the Cost-to-Go via Cutting Planes

Algorithm sketch

1. **Forward pass:** sample a scenario path $\omega^1, \ldots, \omega^T$; solve each stage subproblem to get a candidate state trajectory $x^1, \ldots, x^T$.
2. **Backward pass:** compute a **linear lower bound on** $\mathcal{Q}_{t+1}(x)$ from the dual at each stage and add it to the subproblem.
3. **Iterate** until the optimality gap $\overline{V}_1 - \underline{V}_1 \leq \varepsilon$.

Cutting-plane approximation of $\mathcal{Q}(x)$



Source: V. Leclerc, *Introduction to SDDP*, École des Ponts.

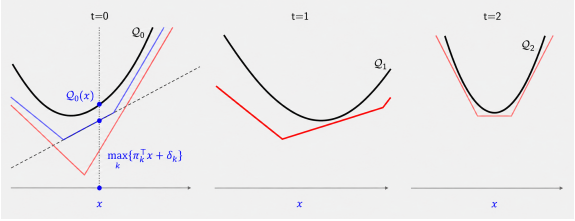
`https://cermics.enpc.fr/~leclerev/courses/Saclay/Saclay-6.pdf`

SDDiP: Building the Cost-to-Go via Cutting Planes

Algorithm sketch

1. **Forward pass:** sample a scenario path $\omega^1, \dots, \omega^T$; solve each stage subproblem to get a candidate state trajectory $x^1, \dots, x^T$.
2. **Backward pass:** compute a **linear lower bound on** $\mathcal{Q}_{t+1}(x)$ from the dual at each stage and add it to the subproblem.
3. **Iterate** until the optimality gap $\overline{V}_1 - \underline{V}_1 \le \varepsilon$.

Cutting-plane approximation of $\mathcal{Q}(x)$

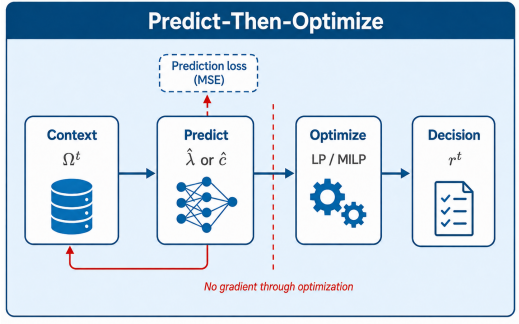


Source: V. Leclerc, *Introduction to SDDP*, École des Ponts.

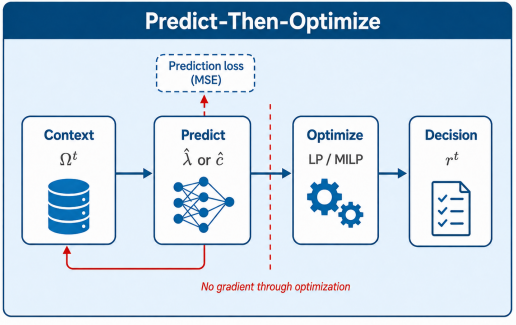
`https://cermics.enpc.fr/~leclerev/courses/Saclay/Saclay-6.pdf`

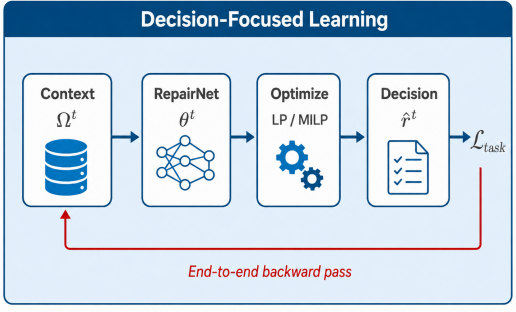
Binary state space: $2^{(1+\tau)(N_G+E)}$ configurations. Already $2^{14} = 16{,}384$ states on the toy example. SDDiP exploits the integer structure to guarantee finite convergence.

“DFL” Intuition



“DFL” Intuition





DFL: Replacing the Cost-to-Go with a Learned Surrogate

Phase 1 – Data generation (offline)

For each training scenario, solve one deterministic MIP and record the optimal repairs $r_s^{*,t}$ at every stage t.

DFL: Replacing the Cost-to-Go with a Learned Surrogate

Phase 1 – Data generation (offline)

For each training scenario, solve one deterministic MIP and record the optimal repairs $r_s^{*,t}$ at every stage t.

Phase 2 – Training (offline)

Learn **RepairNetwork** $\pi_\psi : \Omega^t \to \theta^t$ so that its LP produces repairs matching $r_s^{*,t}$.

The intractable cost-to-go $\mathcal{Q}_{t+1}$ is replaced by a **linear surrogate**:

$$-(\theta_g^t)^\top r_g^t - (\theta_e^t)^\top r_e^t$$

DFL: Replacing the Cost-to-Go with a Learned Surrogate

Phase 1 – Data generation (offline)

For each training scenario, solve one deterministic MIP and record the optimal repairs $r_s^{*,t}$ at every stage t.

Phase 2 – Training (offline)

Learn **RepairNetwork** $\pi_\psi : \Omega^t \to \theta^t$ so that its LP produces repairs matching $r_s^{*,t}$.

The intractable cost-to-go $\mathcal{Q}_{t+1}$ is replaced by a **linear surrogate**:

$$-(\theta_g^t)^\top r_g^t - (\theta_e^t)^\top r_e^t$$

Phase 3 – Deployment (online)

Predict $\theta^t = \pi_\psi(\Omega^t)$; repair k iff $\theta_k^t > \beta_k$; solve one LP for dispatch.

One LP per stage, no backward pass, no cuts. Scales to large networks.

DFL: Replacing the Cost-to-Go with a Learned Surrogate

Phase 1 – Data generation (offline)

For each training scenario, solve one deterministic MIP and record the optimal repairs $r_s^{*,t}$ at every stage t.

Phase 2 – Training (offline)

Learn **RepairNetwork** $\pi_\psi : \Omega^t \to \theta^t$ so that its LP produces repairs matching $r_s^{*,t}$.

The intractable cost-to-go $\mathcal{Q}_{t+1}$ is replaced by a **linear surrogate**:

$$-(\theta_g^t)^\top r_g^t - (\theta_e^t)^\top r_e^t$$

Phase 3 – Deployment (online)

Predict $\theta^t = \pi_\psi(\Omega^t)$; repair k iff $\theta_k^t > \beta_k$; solve one LP for dispatch.

One LP per stage, no backward pass, no cuts. Scales to large networks.

Stage-t subproblem (surrogate substituted)

$$\min_{u^t} \underbrace{c^\top p_g^t + \alpha \mathbf{1}^\top (p_l^t - d^t)}_{\text{dispatch (unchanged)}} + \underbrace{(\beta - \theta^t)^\top r^t}_{\text{cost}-\text{value: net incentive}}$$

Repair k iff $\theta_k^t > \beta_k$. $\theta^t = \pi_\psi(\Omega^t)$: RepairNetwork output.

Training signal (Fenchel-Young loss)

Goal: θ^t such that the LP matches r^*.

$$\nabla_{\theta^t} \ell_{\mathrm{FY}} = \underbrace{\hat{r}(\theta^t)}_{\text{network's decision}} - \underbrace{r^*}_{\text{optimal}}$$

Missed repair $\to$ θ_k^t pushed up $\to$ next time $\theta_k^t > \beta_k \to$ repair. Full derivation in Backup C.

Gain at deployment

No backward pass, no cut accumulation, no scenario tree. Scales to large networks and rolling-horizon reuse.

Experimental Setup

Test networks

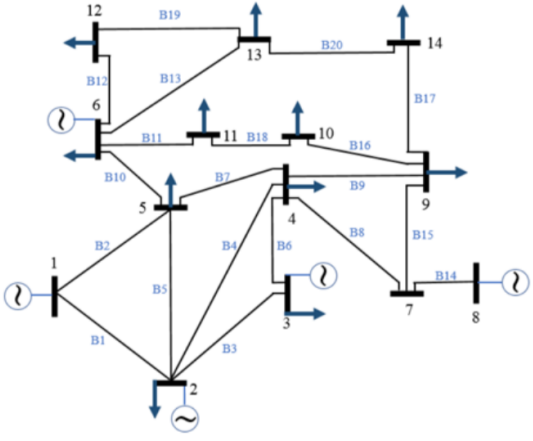
Toy	$N_G = 2,\ E = 5,\ N_L = 3$
IEEE-14	$N_G = 5,\ E = 20,\ N_L = 11$
IEEE-30	$N_G = 6,\ E = 41,\ N_L = 24$

Horizon $T = 14$ stages. Repair delay $\tau = 1$.

Five methods compared

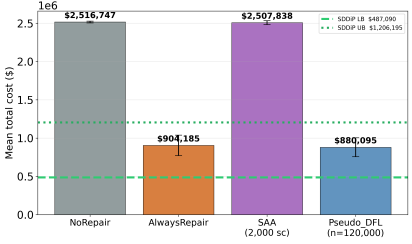
- **NoRepair**: never repair (baseline).
- **AlwaysRepair**: greedy heuristic.
- **SAA**: sample-based approach.
- **SDDiP**: cutting-plane DP, reference.
- **DFL**: one LP per stage, fast deployment.

More on this in Backup B.

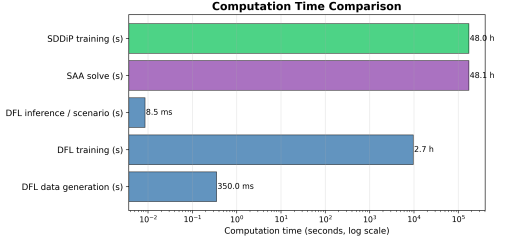


Results: DFL Performance and Speed

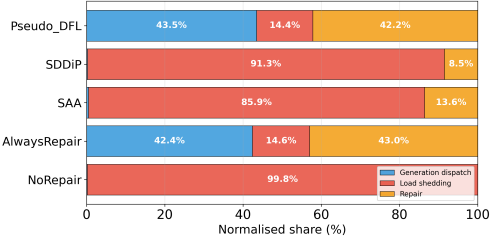
Toy: mean total cost ($n = 120$k, $T = 14$, $N_{\text{eval}} = 500$)



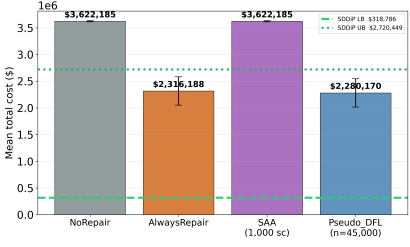
Wall-clock time per method (DFL: one LP/stage; SDDiP: full cutting-plane loop)



Toy: cost share per method, normalised (generation / load shedding / repair)



IEEE-14: mean total cost ($n = 45$k; SDDiP LB/UB gap 88%)

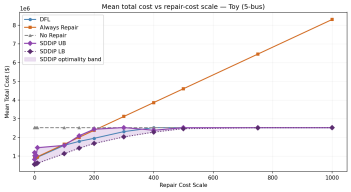


Results: Sensitivity to Repair-Cost Scale

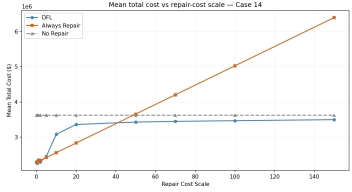
NOTE: The scale multiplies all repair costs uniformly, which simultaneously changes the ratio between load-shedding cost (fixed at 1000 m.u./MWh in our implementations) and repair cost - controlling how expensive it is to let components stay failed versus repairing them.

Each scale trained independently ($n_{\text{train}} = 120{,}000$ scenarios, $T = 14$, $N_{\text{eval}} = 500$). X-axis: repair-cost multiplier. Shaded band: SDDiP optimality interval (toy only).

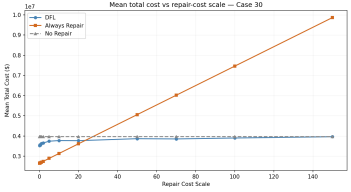
Toy example



IEEE-14



IEEE-30



Toy: DFL robust across all scales. Case-14 & Case-30: DFL's advantage over AlwaysRepair grows with repair cost; in the moderate-to-high cost regime, DFL's selectivity yields significant savings.

Discussion and Conclusion

What we did

- ✓ Built and implemented a multistage stochastic DC-OPF repair model driven by weather uncertainty.

Open questions & next steps

Discussion and Conclusion

What we did

- ✓ Built and implemented a multistage stochastic DC-OPF repair model driven by weather uncertainty.
- ✓ Showed how SDDiP is computationally intensive even for small systems but is useful as a reference for the cost-to-go.

Open questions & next steps

Discussion and Conclusion

What we did

- ✓ Built and implemented a multistage stochastic DC-OPF repair model driven by weather uncertainty.
- ✓ Showed how SDDiP is computationally intensive even for small systems but is useful as a reference for the cost-to-go.
- ✓ Showed how DFL replaces the exact cost-to-go with a learned linear surrogate; experiments on a toy example and IEEE test cases 14 and 30 show promising results.

Open questions & next steps

Discussion and Conclusion

What we did

- ✓ Built and implemented a multistage stochastic DC-OPF repair model driven by weather uncertainty.
- ✓ Showed how SDDiP is computationally intensive even for small systems but is useful as a reference for the cost-to-go.
- ✓ Showed how DFL replaces the exact cost-to-go with a learned linear surrogate; experiments on a toy example and IEEE test cases 14 and 30 show promising results.

Open questions & next steps

- ▷ Scale to larger networks (IEEE 118-bus and beyond).

Discussion and Conclusion

What we did

- ✓ Built and implemented a multistage stochastic DC-OPF repair model driven by weather uncertainty.
- ✓ Showed how SDDiP is computationally intensive even for small systems but is useful as a reference for the cost-to-go.
- ✓ Showed how DFL replaces the exact cost-to-go with a learned linear surrogate; experiments on a toy example and IEEE test cases 14 and 30 show promising results.

Open questions & next steps

- ▷ Scale to larger networks (IEEE 118-bus and beyond).
- ▷ Richer weather models: correlated failures, multiple events,...

Discussion and Conclusion

What we did

- ✓ Built and implemented a multistage stochastic DC-OPF repair model driven by weather uncertainty.
- ✓ Showed how SDDiP is computationally intensive even for small systems but is useful as a reference for the cost-to-go.
- ✓ Showed how DFL replaces the exact cost-to-go with a learned linear surrogate; experiments on a toy example and IEEE test cases 14 and 30 show promising results.

Open questions & next steps

- ▷ Scale to larger networks (IEEE 118-bus and beyond).
- ▷ Richer weather models: correlated failures, multiple events,...
- ▷ Theoretical guarantees (regret bounds under distribution shift, for instance).

Discussion and Conclusion

What we did

- ✓ Built and implemented a multistage stochastic DC-OPF repair model driven by weather uncertainty.
- ✓ Showed how SDDiP is computationally intensive even for small systems but is useful as a reference for the cost-to-go.
- ✓ Showed how DFL replaces the exact cost-to-go with a learned linear surrogate; experiments on a toy example and IEEE test cases 14 and 30 show promising results.

Open questions & next steps

- ▷ Scale to larger networks (IEEE 118-bus and beyond).
- ▷ Richer weather models: correlated failures, multiple events,...
- ▷ Theoretical guarantees (regret bounds under distribution shift, for instance).

Decision-focused learning is a promising path toward scalable, data-driven methods for multistage stochastic optimization.

Thank You!

Questions & discussion welcome

David Krame Kadurha

d.kramekadurha@vu.nl

Supervisors: Alessandro Zocca & Sandjai Bhulai

Vrije Universiteit Amsterdam

StochMod 2026

Scan for slides
& contact details

References

- Zhou, F., Anderson, J., and Low, S. H. (2020). "The Optimal Power Flow Operator: Theory and Computation." *arXiv preprint arXiv:1907.02219v2*.
- Blondel, M., Martins, A. F. T., and Niculae, V. (2020). "Learning with Fenchel-Young Losses." *Journal of Machine Learning Research*, 21(35):1–69. *arXiv:1901.02324*.
- Dalle, G., Baty, L., Bouvier, L., and Parmentier, A. (2022). "Learning with Combinatorial Optimization Layers: a Probabilistic Approach." *arXiv preprint arXiv:2207.13513*. DOI: 10.48550/arXiv.2207.13513.

Backup A: Linearization (for Bellman reformulation and SDDiP implementation)

Four-step linearisation (generator i)

Step 1: pipeline shift

$$q_g^{(\tau)}.\text{out} = r_g^t, \quad q_g^{(k)}.\text{out} = q_g^{(k+1)}.\text{in}, \ k < \tau$$

Step 2: completion signal

$$R_g^t = q_g^{(1)}.\text{in}$$

Step 3: repair activation (McCormick)

$$\chi_g^t \le 1 - a_g.\text{in}, \quad \chi_g^t \le R_g^t, \quad \chi_g^t \ge (1 - a_g.\text{in}) + R_g^t - 1$$

Step 4: availability update

$$\bar{a}_g^t = (1 - \xi_g^t) \cdot a_g.\text{in} + \chi_g^t$$

State dimension

$\dim(x) = (1 + \tau)(N_G + E)$ binary variables.
Toy example ($\tau = 1$, $N_G = 2$, $E = 5$): **14 binaries**.

Why strengthened cuts?

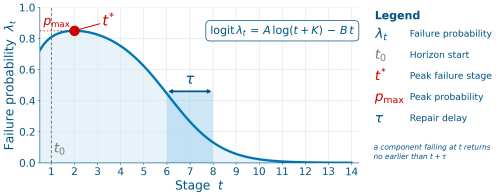
Standard Benders cuts may not be tight for binary states.
SDDiP uses **Lagrangian dual** cuts:

$$\pi^\top x + \delta \le \mathcal{Q}(x) \ \ \forall x \in \{0,1\}^n$$

computed by relaxing the non-anticipativity constraint.

Same thing for lines...

Backup B: Weather-Driven Failure Probability Model



The curve peaks at t^* with probability $p_{\max}$, then decays. Repair delay τ shown.

Cost parameters

Gen. cost	$c \in [20, 25]$ m.u./MWh
Load-shedding	$\alpha = 1{,}000$ m.u./MWh
Gen. repair	$\beta_g \in [440, 660]$ m.u.
Line repair	$\beta_e \in [350, 395]$ m.u.

Log-linear logit specification

$$\lambda_\gamma^t[k] = \sigma\Big(\underbrace{A[k]\ln(t+K)}_{\text{rise}} - \underbrace{B[k]\,t}_{\text{decay}}\Big)$$

Parameters from $(t^*, p_{\max})$

Peak at $t^* \Rightarrow A[k] = B[k](t^* + K)$

$$B[k] = \frac{\ln \frac{p_{\max}}{1-p_{\max}}}{(t^*+K)\ln(t^*+K) - t^*}$$

Valid for $t^* + K > e$ and $p_{\max} > 0.5$ (and also ensures decay to 0).

Vulnerability classes

Class	$p_{\max}[k]$	Behaviour
Robust	$[0.05, 0.20]$	low risk
Standard	$[0.20, 0.50]$	moderate
Vulnerable	$[0.50, 0.80]$	high risk
Critical	> 0.80	severe

Backup C: Fenchel-Young Loss for Decision-Focused Training

Why not use mean-squared error?

MSE trains θ^t to match a prediction target. But we do not have a ground-truth $\theta^{*,t}$; only the **repair decisions** r^* that result from a good θ^t.

We need a loss that penalises **decision errors**, not prediction errors. This is the "predict-then-optimise" problem.

Fenchel-Young loss (Blondel et al., 2020)

For a parameterised solver $\mathcal{S}(\theta) = \arg\min_{r \in \mathcal{F}} (\beta - \theta)^\top r$:

$$\ell_{\mathrm{FY}}(\theta, r^*) = \Omega_\varepsilon^*(\theta) - \theta^\top r^*$$

where Ω_ε^* is the Fenchel conjugate of the perturbation-based regulariser.

Key properties: convex in θ; non-negative; zero iff $\mathcal{S}(\theta) = r^*$.

Gradient and perturbed optimiser

Since $\mathcal{S}(\theta)$ is piecewise-constant (not differentiable), we use the **perturbed optimiser**:

$$\mathcal{S}^\varepsilon(\theta) = \mathbb{E}_{\eta \sim \mathcal{N}(0,I)}\big[\mathcal{S}(\theta + \varepsilon\eta)\big]$$

For our separable repair problem this simplifies to a **smooth threshold**:

$$\mathcal{S}_k^\varepsilon(\theta) = \Phi\left(\frac{\theta_k - \beta_k}{\varepsilon}\right) \cdot \mathbf{1}[a_k = 0]$$

Gradient: $\nabla_\theta \ell_{\mathrm{FY}} = \mathcal{S}^\varepsilon(\theta) - r^*$

Interpretation: **predicted repairs minus optimal solution**. Component k under-repaired $\Rightarrow$ θ_k is pushed up.

References

Blondel et al. (2020). *Learning with Fenchel-Young Losses.* JMLR.
Dalle et al. (2022). *Learning with Combinatorial Optimization Layers.*

Backup D: Full Model – Variables & Constraints

Variables (per stage t)

Decisions: $p_g^t \in \mathbb{R}^{N_G}$, $\theta^t \in \mathbb{R}^N$, $d^t \in \mathbb{R}^{N_L}$, $r_g^t \in \{0,1\}^{N_G}$, $r_e^t \in \{0,1\}^E$

States: $a_g^t \in \{0,1\}^{N_G}$ (gen. avail.), $a_e^t \in \{0,1\}^E$ (line avail.)

Stochastic: $\xi_g^t \in \{0,1\}^{N_G}$, $\xi_e^t \in \{0,1\}^E$ (failure events)

Network: $C \in \mathbb{R}^{N \times E}$ incidence matrix, $B = \mathrm{diag}(b_1, \ldots, b_E)$ susceptances, $N = N_G + N_L$

Constraints (each stage t, each scenario)

Slack bus: $\theta_1^t = 0$

Power balance: $CBC^T\theta^t = \begin{bmatrix} a_g^t \odot p_g^t \\ -d^t \end{bmatrix}$

Generation limits: $\mathbf{0} \le p_g^t \le a_g^t \odot \bar{p}_g$

Line flow limits: $a_e^t \odot \underline{f} \le a_e^t \odot (BC^T\theta^t) \le a_e^t \odot \bar{f}$

Load shedding: $\mathbf{0} \le d^t \le p_l^t$

Repair feasibility: $r_g^t \le \mathbf{1} - a_g^t, \quad r_e^t \le \mathbf{1} - a_e^t$

Bilinear elimination (SDDP reformulation)

The generation limits force $p_g^t[i] = 0$ whenever $\bar{a}_g^t[i] = 0$, so in every feasible solution $a_g^t \odot p_g^t = p_g^t$. The power balance simplifies to:

$$CBC^T\theta^t = \begin{bmatrix} p_g^t \\ -d^t \end{bmatrix}$$

avoiding the bilinear term while preserving equivalence.

Assumptions

Demand: p_l^t is deterministic (perfect forecast).

Info. structure: ξ^t revealed at start of stage t; decisions at t depend only on information up to t (non-anticipativity).

Repairs: initiated at t, effective at $t + \tau$; perfect, no load repair decisions.

Topology: incidence C and susceptances B are constant.

Parameters

$c \in \mathbb{R}_+^{N_G}$	unit generation cost
$\alpha \in \mathbb{R}_+$	load-shedding penalty
β_g, β_e	repair costs (gen/line)
$\underline{p}_g, \bar{p}_g$	generation capacity limits
$\underline{f}, \bar{f}$	line flow limits
$\tau \in \mathbb{Z}_+$	repair delay (stages)

Backup E: Full State Dynamics & Availability Evolution

State vector

$$x = \left(a_g \in \{0,1\}^{N_G},\ a_e \in \{0,1\}^{E},\ q_g^{(1:\tau)},\ q_e^{(1:\tau)}\right)$$

$q_\gamma^{(k),t}[k']$: repair pipeline; $k = \tau$ is the entry (newly initiated repair), $k = 1$ is the exit (repair completes).

$\dim(x) = (1+\tau)(N_G + E)$ binary variables.
Toy ($\tau = 1$, $N_G = 2$, $E = 5$): **14** binaries.

Availability dynamics ($t = 2, \ldots, T$, component-wise)

For $\gamma \in \{g, e\}$ and each component k:

$$a_\gamma^t[k] = \begin{cases} 0 & \text{if } a_\gamma^{t-1}[k] = 1 \text{ and } \xi_\gamma^t[k] = 1 \\ 1 & \text{if } a_\gamma^{t-1}[k] = 0,\ t \geq \tau + 1,\ \sum_{s=t-\tau}^{t-1} r_\gamma^s[k] \geq 1 \\ a_\gamma^{t-1}[k] & \text{otherwise} \end{cases}$$

Initial conditions: $a_g^1 = \mathbf{1}_{N_G}$, $a_e^1 = \mathbf{1}_E$ (all operational at $t = 1$; stochastic start possible).

Failure event model

$\xi_\gamma^t[k] \sim \text{Bernoulli}(\lambda_\gamma^t[k])$ conditionally on $a_\gamma^{t-1}[k] = 1$.

$\xi_\gamma^t[k] = 0$ deterministically if $a_\gamma^{t-1}[k] = 0$ (a component cannot fail if it is already in failed mode).

All failure events independent across components and stages.

Once repaired, component faces $\lambda_\gamma^{t'}[k]$ again with no failure history retained (so, it can fail again).

Stochastic realisation ω^t

$\omega^t = (\xi_g^t, \xi_e^t) \in \{0,1\}^{N_G+E}$, $|\Omega_t| = 2^{N_G+E}$.

Toy ($N_G = 2$, $E = 5$): $2^7 = 128$ realisations per stage.

$$\mathbb{P}(\omega^t) = \prod_{\alpha \in \{e,g\}} \left[\prod_{i=1}^{N_\alpha} (\lambda_\alpha^t[i])^{\xi_\alpha^t[i]} (1 - \lambda_\alpha^t[i])^{1 - \xi_\alpha^t[i]} \right]$$

$\lambda_\gamma^t[k]$ deterministic in $t \Rightarrow$ stagewise independence holds $\Rightarrow$ valid cut sharing in SDDP.