PEGAS: SECURITY ASSESSMENT OF POWER GRID AGENTS THROUGH PHYSICS-INFORMED EVOLUTIONARY GRAPH ADVERSARIAL SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement Learning (RL) agents and neural solvers are becoming the standard for safe & reliable power grid control. Standardized benchmarks and challenges (e.g., L2RPN) show that agents achieve reliable scores, and are resilient to adversaries; driving interest in real-world deployment. This paper argues that current evaluation settings understate real-world adversaries and introduces PEGAS, a query-based, gray-box attack framework that outperforms L2RPN adversarial evaluations. PEGAS scores candidate attacks using a Physics Informed Graph Attention Network surrogate for Optimal Power Flow solvers. PEGAS searches the mixed discrete–continuous, non-convex space with an elitist genetic algorithm. Against static solvers it repairs the selected topology for connectivity; against RL agents it submits line disconnections under the environment's legality rules, which permit islanding. Our convergence analysis shows that PEGAS' best-found fitness converges almost surely to the global optimum. Empirically, on 118-bus grids and standard L2RPN benchmarks (IDF 2023, WCCI 2022), PEGAS substantially reduces defender survival rates compared to existing opponents and beats the winning approaches of these challenges. Specifically, PEGAS achieves a 93.2% success rate against static solvers and decreases the survival of the last L2RPN winner agent from 72% under the challenge's own opponent to 5.3%.

1 INTRODUCTION

Power grids are one of the most critical infrastructures our livelihood depends on. Efficiently operating these grids still requires significant resources and constitutes a real challenge for human operators when optimizing electricity transportation while avoiding blackouts. With the increased complexity of the grids (scale, renewables, electric vehicles, cyber-security threats, etc.), traditional tools (mostly Mixed-Integer Non-Linear Programming solvers) used by electrical engineers to operate the networks and mitigate their issues are becoming increasingly inadequate (Kelly et al., 2020).

Recent advancements in machine learning (ML), such as Graph Neural Networks (GNN) have allowed for finding optimal energy dispatch solutions, known as the Optimal Power Flow Problem for alternating current (AC-OPF), more efficiently for larger grids than traditional solvers (Ghamizi et al., 2024; de Jong et al., 2025). In parallel, Reinforcement Learning (RL) is presenting new opportunities for power grid control with the promise to solve most of these challenges and assist network operators in decision-making (Haarnoja et al., 2018; Sintes & Dang, 2023).

The "Learning to run a power network" (L2RPN) (Marot et al., 2020) challenge is a series of competitions that model the sequential decision-making environments of real-time power network operations (PNO). It is backed by the open-source Grid2op (Donnot, 2020a) core framework and proposes, at each iteration, new challenging scenarios for power grid RL agents. Its latest edition, 2023 TU Delft (Lab, 2023), introduced more realistic scenarios and included an adversarial opponent in addition to the hand-crafted events (e.g., disruptions, power outages, and lightning). The interaction between the opponent and the defender is presented in Fig. 1: The opponent finds an optimal action (topology change or load/dispatch variations, top panels) for which the defender's optimal action (node splitting, rerouting, bottom panels) is not sufficient to recover the grid stability.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Although most solutions proposed for these competitions focused on reducing the action space and learning transfer (van der Sar et al., 2025), we argue that the competitions did not test these PNO agents against malicious behavior. Prior opponents do not actively search for the defender's weakest points; they model the N-1 security criterion (RTE France, 2024b) and sample from a restricted contingency list, weighting each line by its current loading (Marot et al., 2021). Consequently, competition agents achieve high survival rates (72% for (Sintes & Dang, 2023)) in the last L2RPN edition. We keep the native line-disconnection action and attack schedule, but let the attacker choose among 168 lines instead of 22.

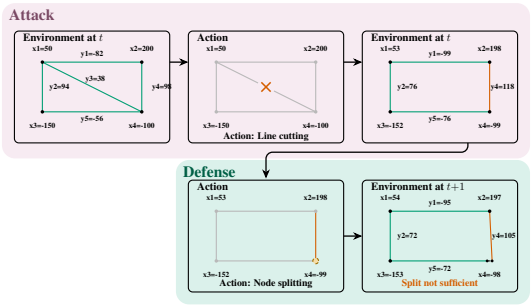


Figure 1: How our adversary agent operates in the top tiles, and how the defender agent can recover the grid in the bottom tiles. The x values on the nodes represent the loads. The y values on the edges represent the % loading. Any y value over 100 (red lines) leads to a grid failure.

To our knowledge, researchers have so far focused on building better defensive agents that survive mild adversarial settings, and we are the first to address PNO competitions (e.g., L2RPN) from the attacker's standpoint. We believe that designing stronger, more realistic adversarial opponents is critical for assessing the robustness of power grid agents. Addressing this problem is non-trivial, and we identify four core challenges that we address to enable systematic adversarial security assessment of power grid agents:

C1: The opponent must apply adversarial actions in near-real-time (i.e., at the same pace as the power grid) and actions must be valid.
C2: The opponent has incomplete knowledge of the grid.
C3: The opponent must handle unknown states of the grid.
C4: The opponent must handle mixed combinatorial spaces.

In this work, we propose Physics-informed Evolutionary Graph Adversarial Search (PEGAS), an adversarial framework that breaks state-of-the-art PNO agents and provides a test-bench for their security assessment. A Physics-Informed Graph Neural Network (PIGNN) surrogate scores the impact of each candidate action and generalizes to perturbed and unseen topologies, addressing C1–C3; EvA+, a mixed discrete–continuous evolutionary algorithm guided by that surrogate, searches the combinatorial action space, addressing C4. In static loop the selected action is projected onto the feasible actions of the environment before it is deployed. The dynamic or live agentic loop sends the selected line under Grid2Op legality.

We evaluate our proposed attack framework in two settings by answering two research questions: ***RQ1:*** *How effective is PEGAS against traditional PNO solvers?* And ***RQ2:*** *How effective is PEGAS against dynamic PNO agents?*

To answer RQ1, we demonstrate the effectiveness of our surrogate and search algorithm against static power grid solvers: PEGAS achieves 93.2% success rate, an improvement of 8.0 percentage points over the best baseline. To answer RQ2, we demonstrate the effectiveness of PEGAS in an RL environment such as Grid2Op: Against PEGAS, all the best agents' survival rate drops from 72% under the standard opponent to at most 6%.

Our contributions are therefore as follows:

- A Novel Physics Informed Power Network Operations surrogate and a mixed discrete–continuous evolutionary algorithm for power grid attacks.
- An extensive evaluation of PEGAS against static OPF solvers, where it significantly outperforms four main baselines and closes the gap with exhaustive search.
- A tooling and agentic integration of PEGAS within Grid2Op RL framework, and its evaluation on the last L2RPN 2023 challenge facing four SoTA defenders.
- A comprehensive generalization study of PEGAS under different topologies: N-1 for static solvers evaluation and the WCCI 2022 challenge for the agentic evaluation.

2 BACKGROUND

We provide in Appendix.A.1 an introduction to the main concepts of power grid operations and the mathematical formulations of the Power Flow problem. Related work is extended in Appendix A.2 for GNNs surrogates and Appendix A.3 for L2RPN competition and agents.

Grid operators rely on the computationally intensive AC Optimal Power Flow (AC-OPF) problem to continuously dispatch power at minimum cost while ensuring the system survives potential single-component failures ($N-1$ contingencies), a process often solved with MINLP solvers such as Ipopt (Wächter & Biegler, 2006; Capitanescu, 2016). While adjusting the grid's topology is a cost-effective corrective action, it creates an intractable combinatorial search space that makes real-time resolution extremely difficult (van der Sar et al., 2025). To bypass these high computational costs, the focus has shifted to Graph Neural Network (GNN) surrogate models to approximate solver outputs efficiently (Ghamizi et al., 2024). Because these models must remain accurate even when the grid's topology unexpectedly changes, recent benchmarks show that combining graph attention architectures with physics-informed losses (Ma et al., 2026)—trained on large-scale collections like OPFData (Lovett et al., 2024)—provides the necessary speed, accuracy, and constraint adherence to manage complex, shifting grid operations safely. The importance of attention and physics-informed losses motivates PEGAS' design.

While surrogate models address the computational challenge of evaluating static grid states, testing the dynamic execution and safety of these topology adjustments requires a sequential framework. The Grid2Op framework models power grid operation as a sequential decision problem where agents manage topology adjustments to prevent overloads and cascading failures within strict physical and temporal limits (Donnot, 2020a). This simulation framework forms the basis of the progressively complex *Learning to Run a Power Network* (L2RPN) competitions (Pavão et al., 2025). However, the standard adversaries deployed in these benchmarks rely on simple heuristics, such as periodically disconnecting heavily loaded lines without simulating the actual systemic impact, meaning current agent robustness is only measured against opponents that lack strategic foresight (Marot et al., 2021; Omnes et al., 2021; RTE France, 2024a). Our work, PEGAS, is the first to introduce adversaries with active strategies to break PNO agents.

Research has focused primarily on approaches to win the competitions. Notable approaches include **OptimCVXPY** (RTE France, 2024c), a convex-optimisation agent that mitigates overloads by redispatching generation and curtailing power when line loadings exceed a critical threshold, returning to the reference dispatch once the grid stabilizes. Another baseline is the **Expert** (Marot et al., 2018) agent. It is a rule-based system that resolves overloads via substation reconfiguration, prioritizing lines based on overflow history and simulating a bounded set of candidate topologies to find a safe state. Finally, the reinforcement-learning approach **LJN**, the winning policy of the L2RPN 2023 competition, relies on simulating candidate reconfigurations to maintain safe line loadings; its neural-network variant, **LJN-TopoNN**, extends this using a PPO policy over a precomputed global action space, relying on heuristic and optimisation-based fallbacks if no learned action successfully secures the grid. We demonstrate through our empirical study that PEGAS breaks all these agents across multiple competition environments.

3 PROBLEM FORMULATION

3.1 SYSTEM MODEL

We consider a transmission grid operated as a cyber-physical system. Substations host buses, generators, loads, and circuit breakers whose switching sets the topology; SCADA collects measurements and executes commands; and a decision-support layer (PF/OPF solvers or RL agents) computes setpoints and topology changes within voltage, thermal, and generator limits. The grid must remain N-1 secure, i.e., withstand the loss of any single line, generator, or transformer.

3.2 ATTACKER MODEL

3.2.1 ATTACKER KNOWLEDGE

Let $G = (V, E)$ be the canonical (undirected, $i < j$) graph of the power system under study, with $m = |E|$ candidate transmission lines (edges) and n_ℓ controllable load nodes. A grid state is a pair $x = (a, L) \in \{0,1\}^m \times \mathbb{R}^{n_\ell \times d}$, where a is the edge-presence indicator vector ($a_e = 1$ iff line e is in service) and L is the load feature matrix (active power draw per load node). The *clean* state is $x^0 = (\mathbf{1}_m, L^0)$.

Attack Characterization. The attacker is query-based gray-box. It knows the buses and the load feature schema of the target grid, but its line parameters come from a reference case (PGLIB case118) rather than the target's live configuration; in RQ2 it observes which lines are in service but not the substation busbar assignments, switch states, or storage units. The attacker also knows the physical bounds $(g^{\min}, g^{\max})$ of the generators (i.e., the minimum and maximum active and reactive power each generator can produce), and the bus voltage bounds $(v^{\min}, v^{\max})$ (i.e., the voltage magnitude range each bus must stay within). Furthermore, the attacker can train a GNN surrogate $f_\theta : x \longmapsto \big(\hat{y}^{\text{gen}}(x),\ \hat{y}^{\text{bus}}(x)\big)$ that predicts optimal generator dispatch and bus voltage state from a grid state x, approximating the AC-OPF solution operator. f_θ is fixed (frozen) at attack time.

3.2.2 ATTACKER CAPABILITIES

The attacker is assumed to access the system to physically damage the power grid (e.g. by cutting a power line (Frymark, 2026)), or strategically changing loads at the consumer nodes (e.g. by compromising high-wattage devices or by coordinating high loads on the network (Shekari et al., 2022)). Based on those two capabilities, the attacker has the following action space:

Topology. The attacker alters the network topology. Formally, a subset $S \subseteq E$, $|S| = k_e$, of lines to remove, encoded as $z \in \{0,1\}^m$ with $\|z\|_0 = k_e$.

Load injection. The attacker alters the consumer bus loads. Formally, a subset of load rows $I \subseteq \{1, \dots, n_\ell\}$, $|I| = k_f$, each perturbed by an additive delta $\delta_i \in \mathbb{R}^d$, bounded coordinate wise by a per-feature L^∞ cap $\varepsilon > 0$: $\|\delta_i\|_\infty \le \varepsilon$ for all $i \in I$, and $\delta_i = 0$ otherwise.

The perturbed state is $x(z, \delta) = \big(a^0 \odot (\mathbf{1} - z),\ L^0 + \delta\big)$, where $\odot$ is a coordinate product on the edge-presence vector restricted to the removed set (edge attributes for removed lines are dropped accordingly). Under a rate-limited attacker (at most one physical change per control step, as in the Grid2Op challenges), the attacker's capability is a single shared budget $k \in \mathbb{N}$: $\|z\|_0 + |I| \le k$.

Feasibility. A candidate (z, δ) is feasible iff it satisfies this budget and the box constraint $\|\delta_i\|_\infty \le \varepsilon$.

While we assume the power grid does not have any monitoring devices to detect anomalies or flag suspicious loads, we enforce the box constraints to restrict the perturbations to the "benign" load patterns: The maximum load variations observed in the grid.

3.2.3 ATTACKER OBJECTIVE

The attacker's goal is to make the grid fail in a way that it cannot be recovered by the solver or by reinforcement learning agents in place.

Let $g^{\min,P}, g^{\max,P}, g^{\min,Q}, g^{\max,Q} \in \mathbb{R}^{n_g}$ be the active (P) and reactive (Q) power bounds of the n_g generators and $v^{\min}, v^{\max} \in \mathbb{R}^{n_b}$ the voltage magnitude bounds of the n_b buses, and let $\tau \ge 0$ be a numerical tolerance. We define the hinge violation of a predicted quantity $\hat{u}$ against bounds (ℓ, u) as $\text{viol}(\hat{u}; \ell, u) = \max(0, \ell - \tau - \hat{u}) + \max(0, \hat{u} - u - \tau)$.

We define the attacker fitness function as

$$\begin{aligned}\Phi_{\text{viol}}(z, \delta) = &\sum_{i \in \mathcal{G}_P} \text{viol}\big(\hat{y}^{\text{gen}}_{p,i}(x(z,\delta));\ g_i^{\min,P}, g_i^{\max,P}\big) \\ &+ \sum_{i \in \mathcal{G}_Q} \text{viol}\big(\hat{y}^{\text{gen}}_{q,i}(x(z,\delta));\ g_i^{\min,Q}, g_i^{\max,Q}\big) \\ &+ \sum_{b \in \mathcal{B}} \text{viol}\big(\hat{y}^{\text{bus}}_{v,b}(x(z,\delta));\ v_b^{\min}, v_b^{\max}\big).\end{aligned}$$

where $\hat{y}^{\text{gen}}_{p,i}, \hat{y}^{\text{gen}}_{q,i}, \hat{y}^{\text{bus}}_{v,b}$ are the predicted active and reactive power of generator i and the voltage magnitude of bus b.

The attacker maximizes Φ_{viol} against its own surrogate f_θ under a feasible perturbation, with no detector to evade, then transfers the chosen action to static solvers (RQ1, Sec. 5) and RL agents (RQ2, Sec. 6); all metrics are computed on these end targets.

3.2.4 Optimization Problem

The attacker's problem $\max_{(z,\delta)\in\Omega_k} \Phi(z,\delta)$ is a mixed discrete–continuous, non-convex, non-differentiable-in-z optimization. Even its discrete restriction ($k_f = 0$) is a subset-selection problem akin to NP-hard network-interdiction problems, with no efficiently checkable optimality certificate for a black-box Φ. Exhaustive enumeration is tractable only at $k = 1$ (m candidates) or small $k = 2$; for the 175 attackable lines of the 118-bus grid, $k = 3$ already gives 877,975 combinations, each requiring a solver call.

This motivates a metaheuristic (genetic) search for $k \geq 2$ and for the mixed discrete–continuous case, where z has no informative gradient at all.

4 Method

We introduce in the following the system-level framework (§4.1), the physics-informed surrogate (§4.2), the evolutionary search that instantiates its attack-selection stage (§4.3), and a convergence analysis of that search (§4.4).

4.1 PEGAS overall framework

PEGAS is a closed-loop framework that observes a state x_t of the power grid and decides whether an attack is admissible under a running attack budget: the deployment grants a total allowance B_0 for the episode, and each executed attack decrements it by its cost, so the opponent must spend a bounded number of attacks over time rather than acting at every step. When an attack is admitted, PEGAS feeds the observation to EvA+, a genetic algorithm that searches for the highest-impact action guided by a physics-informed graph neural network. The solution found by EvA+ is projected, if needed, into the restricted action space of the environment and parsed into an appropriate action and budget update in the environment API (in our study, Grid2Op). We summarize the main components of PEGAS in Fig.2 and report the detailed algorithm of PEGAS in Appendix B.1.

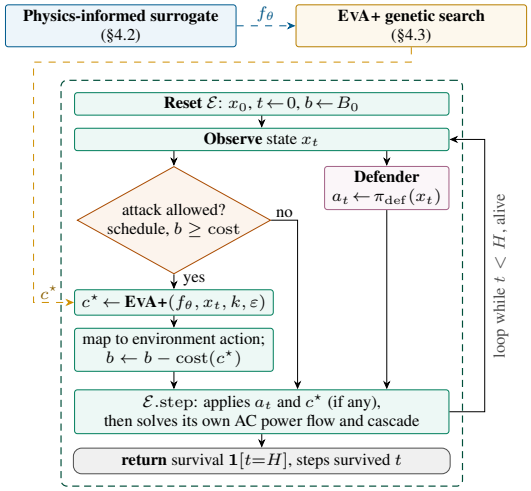


Figure 2: PEGAS closed loop

4.2 Physics-informed surrogate

The surrogate f_θ answers three of the four challenges at once. A supervised loss trains it to mimic the AC-OPF solver, so a candidate attack is scored in one forward pass instead of a solve, fast enough for the control loop (**C1**). It is a graph attention network (Veličković et al., 2018) over the grid graph, so an attack that changes the topology is just a different input graph rather than a model to rebuild (**C2**). A physics-informed Kirchhoff residual added to the loss then keeps its predictions physically consistent on the contingency topologies an attack creates, which it must reach by generalizing from the intact grid it is trained on (**C3**) (Ma et al., 2026).

We train f_θ on solved AC-OPF samples to predict generator setpoints and bus voltages, combining the supervised term with the residual,

$$\mathcal{L} \;=\; \mathcal{L}_{\text{gen}} + \mathcal{L}_{\text{bus}} \;+\; \lambda\, \mathcal{L}_{\text{phys}}, \tag{1}$$

where $\mathcal{L}_{\text{gen}}$ and $\mathcal{L}_{\text{bus}}$ are unweighted per-unit mean squared errors on the generator and bus targets. The residual scores how far the predicted state is from satisfying Kirchhoff's law on the grid's own admittance matrix,

$$\mathcal{L}_{\text{phys}} \;=\; \text{mean}_b\big(\Delta P_b/s_P\big)^2 + \text{mean}_b\big(\Delta Q_b/s_Q\big)^2, \tag{2}$$

where $\Delta P_b, \Delta Q_b$ are the per-bus active and reactive mismatches between the predicted dispatch and the injections its voltages imply, and s_P, s_Q are fixed per-unit scales.

4.3 EvA+: Physics-informed elitist genetic search over Ω_k

EvA+ answers the challenge **C4**: searching Ω_k, the constrained mixed discrete-continuous action space of the attacker's optimization (§3.2.4), which can be too large to enumerate. It is an elitist (μ parents+λ offspring) genetic algorithm, and inherits EvA's (Akhondzadeh et al., 2025) tournament selection and batched candidate evaluation. We extend that search to the mixed space in four ways: a chromosome carries both branch-removal and load genes under one shared budget k, with the load genes drawn from a saliency-ranked pool (i.e., we rank the loads based on their gradient norm), each load variation capped at ε; a *split-shift* mutation moves one gene between the topology and load halves, so their split evolves while k is preserved; a unified crossover recombines both gene types and blends the continuous load deltas; and the fitness is a physical constraint-violation score of the surrogate's prediction, in place of EvA's classification loss. We give the full algorithm in Appendix B.3.

4.3.1 Action validity.

EvA+ Prior work does not guarantee that actions remain valid (as we discuss and show experimentally in Sec. 5.1.) Instead, we overcome this challenge (**C1**), by ensuring that topology candidates are scored unrepaired, the returned RQ1 action is repaired once, and RQ2 does not apply that connectivity repair. We do this by enforcing that a perturbation may (i) only remove lines the grid has, (ii) pick a specific line when several run between the same buses, (iii) not split the grid into more islands than the operator runs, and (iv) keep each perturbed load within its physical limits. The load bound (iv) is enforced at every draw by projection in the feasible space; the topology conditions (i)–(iii) are a single repair of the returned action, since the search runs on the unrepaired topology.

4.4 Convergence analysis

We analyze EvA+ as a time-homogeneous Markov chain $\{(\mathcal{P}_t, \Phi_t)\}_{t\geq 0}$ on population states, and prove that its best-found fitness converges to the global optimum of Problem Sec. 3.2.4 almost surely as $t \to \infty$. We first treat the finite discrete case (topology-only, $k_f \equiv 0$), where the argument is exact, then extend it to the mixed discrete–continuous (hybrid) case under a mild regularity assumption on the mutation kernel, given fixed split between the discrete and continuous action spaces.

With B_t the best fitness in the population at generation t, $F^\star$ the optimum over the $\binom{m}{k}$ discrete candidates, N the population size, n_{tour} the tournament size, and $\pi_{\min} > 0$ a lower bound on the probability that mutation outputs any given chromosome, elitist survival and full-pool mutation give the following guarantee (Appendix B.4 states and proves the supporting lemmas and theorems).

Theorem 1 (Almost-sure convergence, discrete case)**.** *Let $q = 1 - \big(1 - \frac{1}{N}\big)^{n_{\text{tour}}} \in (0, 1]$ be the probability that the tournament in a single offspring draw selects the current best individual as a parent (present in $\mathcal{P}_{t-1}$ by Lemma 1), and let $\rho = q \cdot \pi_{\min} > 0$. Then for every $T \geq 1$,*

$$\Pr\big[B_t < F^\star \;\forall\, t \leq T\big] \;\leq\; (1-\rho)^{NT},$$

and consequently $B_t \to F^\star$ almost surely as $t \to \infty$.

5 RQ1: How effective is PEGAS against traditional PNO solvers?

5.1 Experimental Protocol

The attacker perturbs a static grid, and an AC power flow evaluates the dispatch the surrogate predicts for it, with non-slack generation held at that prediction and no corrective redispatch. In the

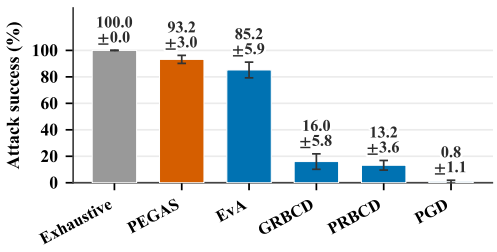


Figure 3: Attack success rate on the intact (N) 118-bus grid, mean ± standard deviation, five seeds (higher is better).

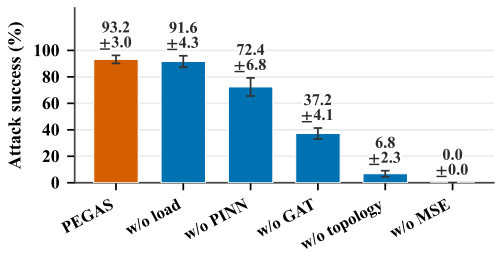


Figure 4: Contribution of PEGAS components: attack success rate on the intact (N) 118-bus grid, mean ± standard deviation over five seeds.

PEGAS loop (Fig. 2), x_t is a sample from the external CASE-118 test set of OPFData (Lovett et al., 2024), the horizon is $H = 1$, there is no defender, and the environment step is one AC power-flow solve.

The surrogates are graph attention networks (Veličković et al., 2018) over the heterogeneous graph. They are trained using 270k samples and validated on 15k samples of OPFData up to 150 epochs with early stopping. Attacks are evaluated on 250 test samples split into disjoint bins over five seeds.

In addition to the EvA (Akhondzadeh et al., 2025) baseline, we compare our approach to three common graph attacks. PRBCD and GRBCD (Geisler et al., 2021) are gradient-based topology attacks: they differentiate the surrogate loss with respect to the adjacency and flip the highest-gradient edges under a block-coordinate scheme. PGD is a projected-gradient attack on the load features under an L_0 node budget (Madry et al., 2018). These four baselines are guided by their traditional surrogate trained without PINN loss.

We report in our study the success rate over all seeds and test runs. An attack is successful on a sample if the attacked state has more generator, bus, or line violations than the sample's clean power-flow state. We also report the RPA (Robust Physical Accuracy) and MSE of buses and generators. RPA is the percentage of buses (respectively generators) with no violation over all test samples.

The original implementation of these baseline attacks all achieved a 0% success rate, given they were not designed to generate valid constrained adversarials. Therefore, we extended all attacks with our Action Validity checker and repair (Sec. 4.3.1) and report in the following the performance of these improved versions. We also report an exhaustive search attack that enumerates all single-branch removals and is the upper bound to any edge-based attack. Detailed protocol and hyperparameters are in Appendix C.5.

5.2 Results

How effective is PEGAS? Figure 3 reports the attack success rate on the target 118-bus grid. PEGAS breaks 93.2% of states, approaching the Exhaustive upper bound (100%) and significantly outperforming the baseline search-attack EvA baseline (85.2%). All the gradient-based baselines (PGD, PRBCD, GRBCD), remain below 16%. We provide in Appendix D.1 exhaustive performance results, including the split per component (bus, generator, and line success rates).

Ablation of PEGAS components. Figure 4 compares ablated configurations on the intact grid (Case N). Removing the supervised term (0%), the GAT backbone (37.2%) or the physics loss (72.4%) lowers success, and a load-only search reaches 6.8%. At k=1 the hybrid search always selected a topology gene, so its 1.6-point gap to topology-only (91.6%) is not due to load changes.

6 RQ2: How effective is PEGAS against dynamic PNO agents?

Now that we have demonstrated the effectiveness of our approach against static solvers, we explore how PEGAS performs when agentic defenders are introduced to mitigate the attacks. The

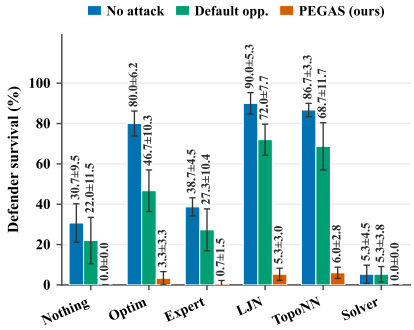


Figure 5: Power grid survival (%) over IDF 2023 by attacker and defender, mean ± standard deviation, five seeds (lower is better).

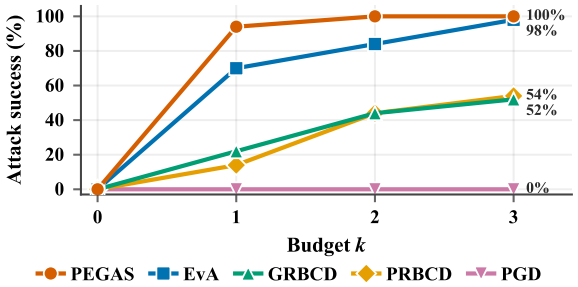


Figure 6: Attack success (%) versus budget k on Grid N, one seed (50 states). PEGAS uses the constraint-violation objective and the baselines the supervised MSE objective; k=0 is the clean state.

target is now a live grid-control loop rather than a static solver. On the Grid2Op (Donnot, 2020a) IDF 2023 and WCCI 2022 benchmarks, a defender and opponent choose from the current observation; Grid2Op applies the actions in the transition, and the defender sees the resulting state at the next step only if the episode survives, to keep it within limits while PEGAS disconnects lines. The defender always acts after the attacker has perturbed the grid and has access to the grid state after the attack. Attacker and defender are allowed one action per step.

6.1 Experimental Protocol

In the PEGAS loop (Fig. 2), the horizon is $H = 2017$ steps and the environment step is Grid2Op's own power flow and cascade; we evaluate 30 scenarios under five seeds. PEGAS uses the same physics-informed surrogate (§4.2) and genetic search as in RQ1, disconnecting one line per attack ($k = 1$); as Grid2Op exposes only line-status actions, it runs topology-only EvA+. It worth to note that in IDF 2023 environment there is 168 versus 22 lines and RQ1 connectivity repair is absent in this environment. The ablation (Sec. 5.2) shows this costs little: the topology-only attack reaches 91.6% against 93.2% for the full attack. We compare PEGAS against *no opponent* and the *default* stochastic opponent of the L2RPN challenge, and confront all three attackers against four SoTA defenders: **OptimCVXPY** (RTE France, 2024c) (convex redispatch), **Expert** (Marot et al., 2018) (rule-based reconfiguration), **LJN** (Pavão et al., 2025) (the RL agent that won L2RPN 2023), and **LJN-TopoNN** (its PPO variant). We also evaluate two baseline scenarios: **DoNothing** where no mitigation strategy is used, and **Solver** where a traditional solver tries to find a single re-dispatch action. The metric is the grid survival rate, i.e., the fraction of the 30 scenarios completed without a blackout, as the mean ± standard deviation over the five seeds; **lower survival is better for the attacker**. The detailed protocol is in Appendix. C.6.

6.2 Results

Figure 5 reports power grid survival for each attacker–defender pair. PEGAS collapses survival to at most 6% against every defender, whereas an unattacked grid or the default opponent leaves 22–90% of episodes surviving for the learned and rule-based defenders.We report **Solver** separately: it survives only 5.3% even without an opponent, so it is already failing before the attack. Across the five defender conditions other than Solver (750 episodes), 613 of the 727 collapsed PEGAS episodes end in the same Grid2Op step as a line 26 disconnection, with logged power-flow divergence;

LJN and its PPO variants are the most robust against PEGAS, with 5.3% and 6% respectively, while the Expert Defender is the weakest of all real defenders with a 0.7% survival rate. The naive baselines **DoNothing** and **Solver** both lead to a 0% survival rate against PEGAS. The detailed survival tables are given in Appendix D.6.

Table 1: Performance of PEGAS vs. the best baseline in ID and OoD settings. RQ1: attacker success rate. RQ2: grid survival rate.

	RQ1 ↑		RQ2 ↓	
Setting	PEGAS	Baseline	PEGAS	Baseline
ID	93.2 ± 3.0	85.2 ± 5.9	6.0 ± 2.8	68.7 ± 11.7
OoD	86.3 ± 5.4	81.5 ± 4.9	32.7 ± 6.4	48.0 ± 6.9

Table 2: Sensitivity of PEGAS to line protection. Mean and std of Survival rate of the power grid with different defenders.

Attacker	**Nothing**	**TopoNN**	**Average**
PEGAS	0.0 ± 0.0	6.0 ± 2.8	3.1 ± 1.1
+ Protect Top 1	2.0 ± 1.8	78.7 ± 8.7	52.3 ± 3.5
+ Protect Top 2	9.3 ± 4.3	88.7 ± 4.5	60.9 ± 3.5

7 Discussion

Generalization. We compare for each of our scenarios in Table 1 the performance of PEGAS and the best baseline for each scenario in In-distribution (ID) and Out-of-distribution settings (OoD). For RQ1, OoD is evaluated using the N-1 OoD validation set of OPFData, and in RQ2, OoD is evaluated using the WCCI 2022 competition environment. Our results show that PEGAS still outperforms the best baseline in static and dynamic evaluation scenarios. However, the gap of performance is tighter: PEGAS' performance degrades to 86.3% success rate against static solvers, but degrades to 32.7% survival rate on the grid with agentic defenders. We report detailed generalization study for RQ1 in Appendix D.5, and for RQ2 in Appendix D.9.

Restricted action space. We evaluate the degradation of performance of PEGAS when lines are removed from PEGAS searche space, the action space of PEGAS, for e.g., by protecting some lines from ever being cut. We report in Table 2 the survival rate of the grid after protecting the top-1 line and the top-2 lines on PEGAS against the best-performing defender (TopoNN). We also report the impact of using no defender, and the average survival rate of all defenders. This study shows that survival rate increases on average from 3.1% to 60.9% when PEGAS is restricted from cutting any of its top-2 candidates.

Scalability to larger combinatorial space. At $k \geq 2$, where exhaustive search becomes intractable, PEGAS stays above the MSE-objective baselines on Grid N (Fig. 6). We provide in Appendix D.4 the detailed results for each k.

Computation overhead. We detail in Appendix D.8 the computation overhead of Eva+ compared to the baseline EvA. Our results show that our new approach costs 1.5× the execution of EvA, but EvA+ remains 9.1 times cheaper than exaustive attack (57.8s for for exhaustive vs 6.34s for EvA+).

Further analyses. Appendix D.9 demonstrates the robustness of PEGAS to variations in the hyper-parameters. We consider variation of the cooldown between attacks, the attack duration, and the scenario attack budget.

Limitations. The search fitness counts generator active and reactive power and bus voltage violations but has no branch thermal term, so it does not directly optimize for the line overloads. However, these overloads are considered in assessing PEGAS and all the baselines: RQ1 scores line limits only after the power-flow projection, and RQ2 measures the trajectory Grid2Op produces.

8 Conclusion

This study proposes PEGAS, an adversarial framework for evaluating the security of power network operation agents. PEGAS uses evolutionary search guided by physics-informed surrogates to identify the actions with maximum impact on the grid, operating under query-based, incomplete knowledge of the system under attack. PEGAS achieves a 93% success rate against static power grid solvers and 94% against the best-performing power grid agents. By defeating every top-performing agent from the L2RPN competitions, PEGAS exposes critical robustness flaws in these agents against malicious operators. It offers a reliable testbed that complements the handcrafted scenarios of the L2RPN competitions, opening new avenues for RL agents to train and refine their policies in adversarial settings, moving the field toward adversarially robust power grid agents.

AI USE STATEMENT

In this work, we used generative AI tools, Claude Code and OpenAI Codex, as software-development and document-preparation assistants. Their use supported code debugging, implementation review, figure-script preparation, and LaTeX document structuring and formatting, and editing parts of this paper for length. We have not used generative AI tools to generate synthetic data sets, develop theoretical models, formulate or prove mathematical claims, propose hypotheses, design the research methodology or experiments, implement the proposed methods, clean or reformat data sets, or interpret results, and translation and qualitative data analysis are not applicable to this work. The outputs of these tools were reviewed, corrected, and integrated selectively by the authors. Experimental results, tables, figures, and scientific claims were verified against the project code, run artifacts, logs, and source files. The AI tools were not used as authoritative sources for numerical results or scientific conclusions. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies attacks on learning-based power network operation agents. Power grids are critical infrastructure, so the attack is dual-use: the same search that evaluates a defender could inform a malicious operator. All experiments use public simulated benchmarks, namely the 118-bus and 2000-bus PGLib-OPF cases, the OPFData dataset, and the Grid2Op `l2rpn_idf_2023` and `l2rpn_wcci_2022` environments. No real grid, operator system, human subject or personal data was involved. The threat model assumes a query-based gray-box attacker with partial knowledge of the topology, whose actions are physical, such as disconnecting a line or shifting a load, rather than a new cyber vulnerability. We consider the evaluation value to outweigh this risk: PEGAS exposes critical robustness flaws in the top-performing agents against malicious operators, which the stochastic opponents of the L2RPN competitions do not reveal. The results also point to defenses. Protecting the two lines that PEGAS attacks most raises the survival rate from 3.1% to 60.9%, and a 100-step attack cooldown raises it to 32.4% (Table 2 and Supp. D). The authors declare no conflicts of interest.

REPRODUCIBILITY STATEMENT

Section 4 and Supp. B describe PEGAS: Algorithm 1 gives the closed loop, Supp. B.2 the physics-informed surrogate, and Algorithm 2 with Supp. B.3 the EvA+ search and the action-validity repair. Supp. B.4 states the assumptions of the convergence results and gives complete proofs (Lemmas 1 and 2, Theorems 1 and 2). Supp. C gives the data splits, the power-flow scoring and the hyperparameters, and Supp. D reports every result as the mean and standard deviation over five seeds (42 to 46), with paired significance tests. The anonymized code supplement holds the code for surrogate training, the static attacks, the closed-loop opponent runs and the analysis. It carries no trained checkpoints, no run artifacts, no logs and no figures, so reproducing a reported number means rerunning the training, campaign and analysis commands that produced it; Supp. D lists these commands. The command `bash scripts/setup/environment.sh all` installs the Python requirements and fetches the PGLIB, Grid2Op and OPFData assets together with the Julia solver stack. The LJN model weights and action-space archives are not included and are restored from the pinned upstream commit under its Mozilla Public License 2.0. Each reported campaign has a documented launcher and seed configuration, and the analysis commands derive the summaries, tables and figures from the resulting artifacts.

REFERENCES

Mohammad Sadegh Akhondzadeh, Soroush H Zargarbashi, Jimin Cao, and Aleksandar Bojchevski. Eva: Evolutionary attacks on graphs. *arXiv preprint arXiv:2507.08212*, 2025.

Sogol Babaeinejadsarookolaee, Adam Birchfield, Richard D. Christie, Carleton Coffrin, Christopher DeMarco, Ruisheng Diao, Michael Ferris, Stephane Fliscounakis, Scott Greene, Renke Huang, et al. The power grid library for benchmarking ac optimal power flow algorithms. *arXiv preprint arXiv:1908.02788*, 2019.

Florin Capitanescu. Critical review of recent advances and further developments needed in ac optimal power flow. *Electric Power Systems Research*, 136:57–68, 2016.

Carleton Coffrin, Russell Bent, Kaarthik Sundar, Yeesian Ng, and Miles Lubin. Powermodels.jl: An open-source framework for exploring power flow formulations. In *2018 Power Systems Computation Conference (PSCC)*, pp. 1–8, June 2018. doi: 10.23919/PSCC.2018.8442948. URL https://arxiv.org/abs/1711.01728.

Matthijs de Jong, Jan Viebahn, and Yuliya Shapovalova. Generalizable graph neural networks for robust power grid topology control. *arXiv preprint arXiv:2501.07186*, 2025.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(1):1–30, 2006. URL http://jmlr.org/papers/v7/demsar06a.html.

Thomas G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, 10 1998. ISSN 0899-7667. doi: 10.1162/089976698300017197. URL https://doi.org/10.1162/089976698300017197.

Benjamin Donnot. Grid2op-a testbed platform to model sequential decision making in power systems. *GitHub repository*, 2020a.

Benjamin Donnot. Lightsim2grid: A C++ backend targeting the Grid2Op platform. https://github.com/Grid2op/lightsim2grid, 2020b. URL https://github.com/Grid2op/lightsim2grid.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019. URL https://arxiv.org/abs/1903.02428.

Stephen Frank, Ingrida Steponavice, and Steffen Rebennack. Optimal power flow: A bibliographic survey i: Formulations and deterministic methods. *Energy systems*, 3:221–258, 2012.

Kamil Frymark. Consequences of the Berlin blackout. https://www.osw.waw.pl/en/publikacje/analyses/2026-01-09/consequences-berlin-blackout, January 2026. Accessed: 2026-07-23.

Simon Geisler, Tobias Schmidt, Hakan Şirin, Daniel Zügner, Aleksandar Bojchevski, and Stephan Günnemann. Robustness of graph neural networks at scale. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 7637–7649. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/3ea2db50e62ceefceaf70a9d9a56a6f4-Paper.pdf.

Salah Ghamizi, Jun Cao, Aoxiang Ma, and Pedro Rodriguez. Powerflowmultinet: Multigraph neural networks for unbalanced three-phase distribution systems. *IEEE Transactions on Power Systems*, 40(1):1148–1151, 2024.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, pp. 1861–1870. PMLR, 2018.

Adrian Kelly, Aidan O'Sullivan, Patrick de Mars, and Antoine Marot. Reinforcement learning for electricity network operation. *arXiv preprint arXiv:2003.07339*, 2020.

Delft AI Energy Lab. Learning to run a power network - delft 2023 competition. https://codalab.lisn.upsaclay.fr/competitions/12420, 2023. CodaLab; Accessed 08-08-2024.

Sean Lovett, Miha Zgubič, Sofia Liguori, Sephora Madjiheurem, Hamish Tomlinson, Sophie Elster, Chris Apps, Sims Witherspoon, and Luis Piloto. Opfdata: Large-scale datasets for ac optimal power flow with topological perturbations, 2024. URL https://arxiv.org/abs/2406.07234.

Aoxiang Ma, Salah Ghamizi, Aleksandar Bojchevski, Pei Zhang, and Jun Cao. Safepowergraph: A safety-aware benchmarking framework for scalable graph neural networks in power flow and optimal power flow. *Energy and AI*, 25:100819, 2026. ISSN 2666-5468. doi: https://doi.org/10.1016/j.egyai.2026.100819. URL https://www.sciencedirect.com/science/article/pii/S266654682600145X.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. URL https://arxiv.org/abs/1706.06083.

A Marot, B Donnot, S Tazi, and P Panciatici. Expert system for topological remedial action discovery in smart grids. In *Mediterranean Conference on Power Generation, Transmission, Distribution and Energy Conversion (MEDPOWER 2018)*, pp. 1–6. IET, 2018.

Antoine Marot, Isabelle Guyon, Benjamin Donnot, Gabriel Dulac-Arnold, Patrick Panciatici, Mariette Awad, Aidan O'Sullivan, Adrian Kelly, and Zigfried Hampel-Arias. L2rpn: Learning to run a power network in a sustainable world neurips2020 challenge design. *Réseau de Transport d'Électricité, Paris, France, White Paper*, 2020.

Antoine Marot, Benjamin Donnot, Gabriel Dulac-Arnold, Adrian Kelly, Aidan O'Sullivan, Jan Viebahn, Mariette Awad, Isabelle Guyon, Patrick Panciatici, and Camilo Romero. Learning to run a power network challenge: a retrospective analysis. In *Proceedings of the NeurIPS 2020*, volume 133 of *PMLR*, pp. 112–132, 06–12 Dec 2021.

Loïc Omnes, Antoine Marot, and Benjamin Donnot. Adversarial training for a continuous robustness control problem in power systems. In *2021 IEEE Madrid PowerTech*, pp. 1–6. IEEE, 2021.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32, pp. 8024–8035. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

Adrien Pavão, Antoine Marot, Jules Sintes, Viktor Eriksson Möllerstedt, Laure Crochepierre, Karim Chaouache, Benjamin Donnot, Van Tuan Dang, and Isabelle Guyon. Ai challenge for safe and low carbon power grid operation. *Energy and AI*, 22:100564, 2025. ISSN 2666-5468. doi: https://doi.org/10.1016/j.egyai.2025.100564. URL https://www.sciencedirect.com/science/article/pii/S2666546825000965.

Ivonne Pena, Carlo Brancucci Martinez-Anido, and Bri-Mathias Hodge. An extended ieee 118-bus test system with high renewable penetration. *IEEE Transactions on Power Systems*, 33(1): 281–289, 2017.

RTE France. Welcome to grid2op's documentation. https://grid2op.readthedocs.io/en/latest/, 2024a. Read the Docs; Accessed 26-08-2024.

RTE France. Opponent modeling — Grid2Op 1.12.4 documentation. https://grid2op.readthedocs.io/en/latest/user/opponent.html, 2024b. Accessed: 23-07-2026.

RTE France. Welcome to l2rpn-baselines's documentation! https://l2rpn-baselines.readthedocs.io/en/latest/index.html, 2024c. Read the Docs; Accessed 26-08-2024.

Tohid Shekari, Alvaro A. Cardenas, and Raheem Beyah. MaDIoT 2.0: Modern High-Wattage IoT botnet attacks and defenses. In *31st USENIX Security Symposium (USENIX Security 22)*, pp. 3539–3556, Boston, MA, August 2022. USENIX Association. ISBN 978-1-939133-31-1. URL https://www.usenix.org/conference/usenixsecurity22/presentation/shekari.

Jules Sintes and Van Tuan Dang. How we built the winning real time autonomous agent for power grid management in the l2rpn challenge 2023. https://lajavaness.medium.com/, 2023. Medium; accessed: 01-08-2024.

R. S. Sutton and A. G. Barto. *Reinforcement learning an introduction*. MIT press, 2018.

Erica van der Sar, Alessandro Zocca, and Sandjai Bhulai. Optimizing power grid topologies with reinforcement learning: A survey of methods and challenges. *Foundations and Trends in Electric Energy Systems*, 9(1):1–119, 2025.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://arxiv.org/abs/1710.10903.

Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106 (1):25–57, 2006. doi: 10.1007/s10107-004-0559-y. URL https://doi.org/10.1007/s10107-004-0559-y.

Deunsol Yoon, Sunghoon Hong, Byung-Jun Lee, and Kee-Eung Kim. Winning the l2rpn challenge: Power grid management via semi-Markov afterstate actor-critic. In *International Conference on Learning Representations*, 2021.

A SUPPLEMENTARY A: BACKGROUND

A.1 THE POWERFLOW OPTIMIZATION

Complex power in AC circuits:

Electric power transmission is more efficient at high voltages because these higher voltages minimize energy loss due to dissipation in the transmission lines. Power grids typically utilize alternating current (AC) because the voltage of AC can be conveniently changed (from high to low) using transformers. Hence, we will begin by introducing some notation and definitions relevant to AC circuits.

A major feature of AC circuits is that, in contrast to direct current (DC) circuits, the currents and voltages fluctuate over time: their magnitude and direction change periodically. Due to various technical benefits such as reduced losses and fewer disturbances, power generators employ sinusoidal alternating quantities, which can be easily represented using complex numbers.

We will consistently use capital and small letters to denote complex and real-valued quantities, respectively. For instance, let us consider two buses, $i, j \in \mathcal{N}$, that are directly connected by a transmission line (i, j) . The complex power flowing from bus to bus is denoted by S_{ij} and it can be decomposed into its active (p_{ij}) and reactive (q_{ij}) components:

$$S_{ij} = p_{ij} + jq_{ij} \tag{3}$$

where $j = \sqrt{-1}$. The complex power flow can be expressed as the product of the complex voltage at bus i, V_i, and the complex conjugate of the current flowing between the buses I^*_{ij},

$$S_{ij} = V_i I^*_{ij}, \tag{4}$$

Transmission lines have power losses due to their resistance $(r_{i,j})$, that indicates the opposition to current flow. In AC circuits, there is an additional dynamic effect due to the line reactance (x_{ij}). Unlike resistance, reactance does not result in power loss but causes temporal delays by storing and then returning power to the circuit. The combined effect of resistance and reactance can be expressed through a complex quantity called impedance: $Z_{ij} = r_{i,j} + jx_{ij}$. We also consider the admittance, which is the reciprocal of the impedance: $Y_{ij} = \frac{1}{Z_{ij}}$. Similarly to the impedance, the admittance can be also decomposed into its real, conductance (g_{ij}), and imaginary, susceptance (b_{ij}), components: $Y_{ij} = g_{ij} + jb_{ij}$.

Following Ohm's law, we can write the current as a function of the line voltage drop and the admittance between the two buses:

$$I_{ij} = Y_{ij}(V_i - V_j), \tag{5}$$

Replacing the above expression for the current in the power flow equation (eq. 4), we get

$$S_{ij} = Y^*_{ij} V_i V^*_i - Y^*_{ij} V_i V^*_j = Y^*_{ij}(|V_i|^2 - V_i V^*_j). \tag{6}$$

This power flow equation can be expressed using the admittance components and the polar form of voltage, i.e. $V_i = v_i e^{j\theta_i} = v_i(cos\theta_i + jsin\theta_i)$ (where v_i and θ_i are the voltage magnitude and angle of bus i, respectively):

$$S_{ij} = (g_{ij} - jb_{ij})(v_i^2 - v_i v_j(cos(\theta_{ij}) + jsin(\theta_{ij}))), \tag{7}$$

where we denote with angle difference as $\theta_{ij} = \theta_i - \theta_j$.

We also simplify the conductance and susceptance components with algebraic identities as follows:

$$g_{ij} - jb_{ij} = \frac{g_{ij}^2 + b_{ij}^2}{g_{ij} + jb_{ij}} = \frac{|Y_{ij}|^2}{Y_{ij}} = \frac{Z_{ij}}{|Z_{ij}|^2} = \frac{r_{i,j} + jx_{ij}}{r_{i,j}^2 + x_{i,j}^2}, \tag{8}$$

hence, the impedance components-based power flow expression becomes:

$$S_{ij} = \frac{r_{i,j} + jx_{ij}}{r_{i,j}^2 + x_{i,j}^2}(v_i^2 - v_i v_j(cos(\theta_{ij}) + jsin(\theta_{ij}))), \tag{9}$$

Finally, the corresponding real equations can be written as:

$$\begin{cases} p_{ij} = \frac{1}{r_{i,j}^2 + x_{i,j}^2}\left[r_{i,j}(v_i^2 - v_i v_j cos(\theta_{ij})) + x_{i,j}(v_i v_j sin(\theta_{ij}))\right] \\ q_{ij} = \frac{1}{r_{i,j}^2 + x_{i,j}^2}\left[x_{i,j}(v_i^2 - v_i v_j cos(\theta_{ij})) - r_{i,j}(v_i v_j sin(\theta_{ij}))\right] \end{cases} \tag{10}$$

A.2 Grid operations and the AC optimal power flow problem

A grid operator must serve demand at minimum cost while keeping every component within its operating limits. This is the AC optimal power flow problem (AC-OPF): choose the active and reactive power output of each generator so as to minimize cost, subject to the network's power flow equations, and to bounds on generator output, bus voltage, and line loading. The power flow equations are non-linear, so AC-OPF is solved numerically. We follow standard practice and formulate it with PowerModels (Coffrin et al., 2018), solving with Ipopt (Wächter & Biegler, 2006); its formulations and solution methods have been surveyed in detail (Frank et al., 2012; Capitanescu, 2016). Operation requires repeated solves. Dispatch is recomputed every few minutes, and each solution must also remain feasible after loss of any single component. We denote by N the network with all branches in service and by $N-1$ the network with one branch removed.

Topology reconfiguration is the operator's least costly corrective action. Switching a line out of service or splitting a substation so that its connected equipment feeds two separate busbars redirects the power flow without altering the setpoints of the generator (Yoon et al., 2021). Alternatives have a direct cost: compensating generators for departing from their schedule, cutting power, or disconnecting the load. Topology control instead exploits the infrastructure that is already in place (van der Sar et al., 2025). The difficulty lies in selecting the configuration rather than in executing the switch. A substation with two busbars admits $2^{|v|}$ ways to partition its connected equipment, where $|v|$ is the number of generators, loads, and lines attached, so the number of options doubles with each additional element (van der Sar et al., 2025). Aggregated on a network, the space becomes intractable: the 36-substation grid used in the L2RPN competition admits roughly $70{,}000$ topologically distinct actions (Yoon et al., 2021). Operators and learned agents alike therefore act over a reduced candidate set rather than the full space. That same dependence is what exposes the grid. If the security margin rests this heavily on which branches are in service, removing branches is a means of attack.

The cost of solving AC-OPF repeatedly has motivated surrogate models trained to reproduce the solver's output from the grid state. Graph neural networks apply directly, since the grid itself is a graph, and they are considerably faster, reaching a 38-fold speedup over a model-based solver on a 906-bus network (Ghamizi et al., 2024). Speed alone is insufficient. A surrogate used for contingency screening must remain accurate when the topology changes, which is precisely when it departs farthest from its training distribution. The SafePowerGraph benchmark evaluates this across five grids of 9 to $13{,}659$ busses under load variation, price variation, and line outages. Graph attention outperforms alternative architectures in every scenario and at every grid scale, and increasing the supervised loss with a physics term reduces constraint violations by up to 80% (Ma et al., 2026). Supervision on the scale such models require is available from OPFData, a public collection of solved AC-OPF instances that includes the $N-1$ variant of each case (Lovett et al., 2024); it is the source of our training data. These two findings, attention and a physics term, motivate the surrogate of PEGAS.

A.3 THE LEARNING TO RUN A POWER NETWORK (L2RPN) CHALLENGE

The objective of these challenges is to learn to operate a power grid so as to ensure continuous electricity delivery to consumers over a specified time horizon while preventing blackouts. In practice, blackouts stem from cascading failures in which overloaded transmission lines are automatically disconnected, potentially interrupting power supply to end users.

Agents must be constructed to minimize the **cost of operations** while respecting safety constraints. This sequential decision-making problem is formulated as a Markov Decision Process (MDP) and tackled using reinforcement learning (RL) (Sutton & Barto, 2018).

The L2RPN competition series instantiates the Grid2OP environment on progressively harder grids. The 2020 WCCI edition used a 36-substation network (Yoon et al., 2021). The NeurIPS 2020 edition ran two tracks, one testing robustness against an adversary and one testing adaptation to unseen renewable mixes (Marot et al., 2021). The 2023 edition, the `l2rpn_idf_2023` environment we attack in RQ2, is built on a 118-node grid divided into three areas, with sixteen years of weekly scenarios, 832 in total, and six competing teams (Pavão et al., 2025).

Objectives and game over condition. The environments are episodic, simulated at a **5-minute** time resolution over **one week** (2016 timesteps). At their core lies a power grid represented as a network of substations (nodes), some of which host consumers (loads) or producers (generation), connected via transmission lines (edges). These lines exhibit varying physical properties, and the entire system is modeled as a **graph**. For this competition, the standard synthetic **IEEE 118** benchmark network was employed (Pena et al., 2017). The network is paired with realistic time series of production and consumption. Within this setting, agents can intervene on the grid, subject to the following constraints:

- Events such as scheduled maintenance (deterministic) and line disconnections (stochastic and potentially adversarial) can take power lines out of service for several hours.
- Power lines have inherent physical capacity limits that constrain the amount of flow they can carry. If a line remains overloaded for too long, it will automatically trip, potentially triggering cascading failures.
- After a line has been disconnected, an agent must wait several hours before it can be reconnected.
- Furthermore, to prevent costly degradation or failure of network assets, an agent is restricted from intervening too frequently on the same asset or executing too many actions simultaneously.

The Game Over condition occurs when the system can no longer fully satisfy total demand, meaning that some consumption is shed at one or more substations. A more detailed discussion of the governing physical laws and the network's physical properties is provided in the white paper by (Kelly et al., 2020).

Action space. The environment offers two categories of actions: inexpensive topology adjustments (discrete, combinatorial moves) that let the agent reconfigure the grid graph, and expensive changes to power production from a fixed set of generators (continuous controls). Generation levels can be adjusted as long as they remain within the physical limits of each plant. Overall, the action space consists of more than $70{,}000$ discrete actions and a 40-dimensional continuous control space.

Observation space. At each time step, agents have access to the full system state of the power grid. This state encompasses, among other things, power flows on every transmission line, the electricity demand and generation at each node, and the remaining disconnection time of lines. Once the previously described constraints are checked, each action is passed to an internal power-flow engine that runs an AC power flow to determine the next grid state. Agents also have the ability to **simulate** the impact of a candidate action on the current state, for example to validate it in a way similar to how a human operator would proceed. However, the future remains uncertain: unforeseen contingencies cannot be predicted, and future production and demand are stochastic.

Reward. The organizers provide a standard scoring function used to rank agents. The main ranking is based on the survival of these agents across all the scenarios of the challenge.

B SUPPLEMENTARY B: ALGORITHMS AND PROOFS

B.1 PEGAS

Algorithm 1 states PEGAS for a generic grid environment $\mathcal{E}$. It is written in terms of four objects that any deployment must supply: a state stream, an admissibility rule that says when an attack may be issued and which actions are legal, a defender policy, and a transition. Nothing in the loop is tied to a particular simulator or to reinforcement learning.

Algorithm 1 PEGAS

Require: perturbation budget k, environment $\mathcal{E}$, defender policy π_{def}, load cap ε, admissibility rule $\mathcal{A}$, episode budget B_0, horizon H

offline, once: fit surrogate f_θ with the physics-informed objective, then freeze θ
$x_0 \leftarrow \mathcal{E}.\text{reset}(); \quad t \leftarrow 0; \quad b \leftarrow B_0$
while $t < H$ and episode not terminated **do**
  observe grid state x_t from $\mathcal{E}$
  $a_t \leftarrow \pi_{\text{def}}(x_t)$ ▷ defender acts on x_t
  if $\mathcal{A}$ admits an attack at t and $b \geq \text{cost}$ **then**
    $z^\star \leftarrow \text{EVA+}(f_\theta, x_t, k, \varepsilon)$ ▷ Algorithm 2
    $\tilde{z} \leftarrow \text{PROJECTFEASIBLE}(z^\star)$ ▷ RQ1 repair; identity in RQ2
    queue $\tilde{z}$ for this transition
    $b \leftarrow b - \text{cost}(\tilde{z})$ ▷ spend from the episode budget
  end if
  $x_{t+1} \leftarrow \mathcal{E}.\text{step}(a_t)$ ▷ applies a_t, then any queued $\tilde{z}$
  $t \leftarrow t + 1$
end while
return survival indicator $\mathbf{1}[t = H]$, steps survived t

Each object is instantiated by the deployment, and several of them degenerate in the simpler settings. A static solver is the case $H = 1$: the state stream is a dataset of grid cases rather than a trajectory, there is no defender, so π_{def} is the identity, and the transition is a single AC-OPF solve. A live control loop supplies a real policy and a real transition. The admissibility rule likewise ranges from an unrestricted action set to a schedule with a cooldown, and the episode budget B_0 caps how many attacks may be issued (it is unbounded in the static setting). The projection is the step that enforces the physical realisability of a candidate, and where the environment already rejects illegal actions itself, it reduces to the identity.

B.2 PHYSICS INFORMED GNN SURROGATE

Given a grid state x, comprising the bus loads, the in-service branch set and the electrical parameters of the network, f_θ predicts the optimal solution returned by an AC-OPF solver, namely the generator active and reactive setpoints and the bus voltage magnitudes and angles. Training and validation data are synthetic samples solved using an exact AC-OPF Solver.

Inputs and outputs. A grid state x is the heterogeneous graph supplied by OPFData, with four node types. Each bus carries 4 features, its base voltage, its type and its voltage magnitude limits; each generator carries 11, its base power, active and reactive limits, voltage setpoint and cost coefficients; each load carries its active demand; and each shunt its susceptance and conductance. Buses are joined by two physical relations, the switchable AC branches (`ac_line`, 175 for the 118-bus case, in canonical undirected form with $i < j$) and the transformers (11), and each generator, load and shunt is attached to its bus by a membership relation. The branch and transformer relations carry the electrical parameters used below, namely series resistance and reactance, shunt susceptances, and tap ratio. The network predicts the quantities an AC-OPF solves would return: per generator, the active and reactive setpoints $(\hat{P}_g, \hat{Q}_g)$, and per bus, the voltage angle and magnitude $(\hat{V}_b^a, \hat{V}_b^m)$, all

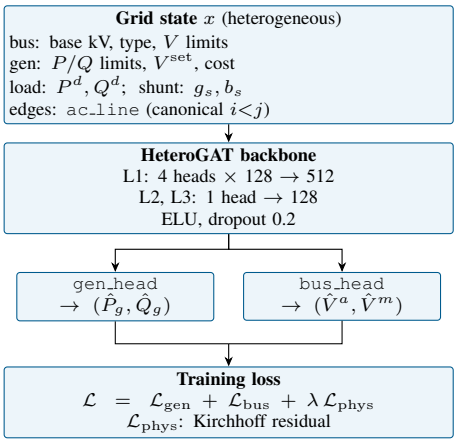


Figure 7: Physics-informed OPF surrogate: heterogeneous grid state, attention backbone, prediction heads, and the training loss.

in per unit and regressed directly, with no target normalization. Training pairs come from OPFData samples for the target case, each pairing a state with its solved optimum.

Architecture. A shared graph attention backbone is replicated per relation and its messages are summed at each node, so every relation carries its own attention parameters. The backbone is three attention layers, the first with 4 heads whose outputs are concatenated to width 512 and the remaining two single-headed at width 128, with ELU activations and dropout 0.2 between layers. Attention is computed from the endpoint embeddings alone: branch electrical parameters enter the physics term below rather than the attention, and no self-loops are added. Two linear heads read the final bus and generator embeddings, each mapping width 128 to its two targets, for 1.37M parameters in total.

Physics-informed objective. The training loss adds an AC power-flow residual $\mathcal{L}_{\text{phys}}$ to the unweighted per-unit generator and bus mean-squared-error terms. The residual is Kirchhoff's law evaluated on the network's own admittance matrix. Let $V_b = \hat{V}_b^m \exp(\mathrm{j}\hat{V}_b^a)$ be the complex voltage implied by the predicted bus outputs, and let Y be the bus admittance matrix assembled from the branch, transformer and shunt parameters of x. The injection implied by those voltages is $S = V \odot \overline{YV}$, whereas the injection implied by the predicted dispatch is $P_b^{\text{inj}} = \sum_{g\to b} \hat{P}_g - \sum_{l\to b} P_l^d$, and likewise for reactive power. The two mismatches are $\Delta P = P^{\text{inj}} - \Re(S)$ and $\Delta Q = Q^{\text{inj}} - \Im(S)$, each divided by its own per-unit scale (s_P for ΔP, s_Q for ΔQ), $s_P = \max\big(s_{\min}, \text{mean}_b|P_b^{\text{inj}}|\big)$ and $s_Q = \max\big(s_{\min}, \text{mean}_b|Q_b^{\text{inj}}|\big)$ with floor $s_{\min} = 10$ p.u., which keeps a lightly loaded graph from producing a vanishing denominator. On the grids we study the mean injection stays below this floor, so the scale is in practice the constant $s_{\min}$ and the residual is a fixed multiple of the raw mismatch.

.

The supervised term teaches f_θ to mimic solver outputs but leaves it free to predict operating points that no network could realize. The residual constrains predictions to be consistent with Kirchhoff's law on the network's own admittance matrix, but says nothing about which feasible point the solver would actually choose. The ablation of Table 8 separates the two: removing the residual costs about a fifth of the attack success on the intact grid, while removing the supervised term costs essentially all of it. The second case is instructive: satisfying the residual alone is still useless as an attack guide, because nothing ties it to the operating point the solver would pick.

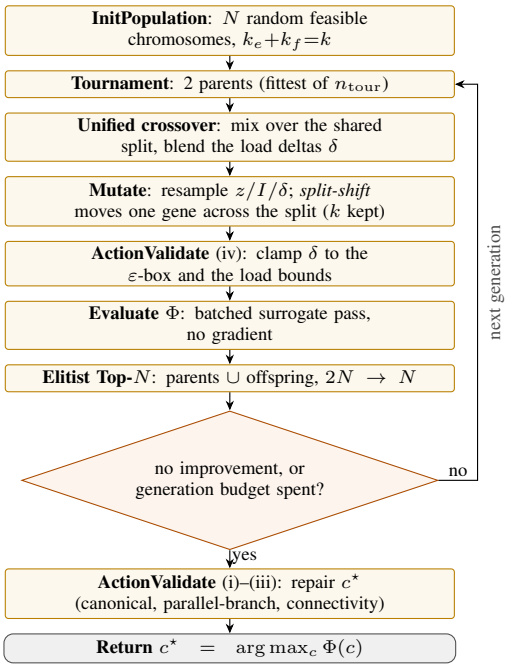


Figure 8: EVA+ elitist genetic search. ACTIONVALIDATE (iv) clamps the load deltas at every draw; ACTIONVALIDATE (i)–(iii) repairs the returned chromosome once, after the search.

B.3 EVA+

B.3.1 MAIN LOOP

We present in Algorithm 2 the complete loop of our evolutionary algorithm. It involves the following internal procedures:

- TOURNAMENT: sample n_{tour} individuals uniformly (with replacement) from the population and return the fittest; this is repeated to obtain each of the two parents.
- CROSSOVER: the operator differs by mode. *(a) Topology genes.* Multi-point crossover with k_{cross} swap points, followed by duplicate repair against the edge pool. When $k_{\text{cross}} \geq k_e$ this degenerates into an independent per-slot swap with probability $1/2$; since $k_{\text{cross}} = 30$ exceeds every budget we run, this degenerate branch is the one exercised in all our experiments. *(b) Load genes (feature-only mode).* Index swap plus arithmetic blend $\delta_{\text{child}} = \alpha\,\delta_{p_1} + (1-\alpha)\,\delta_{p_2}$ with $\alpha \sim \text{Unif}[0.3, 0.7]$, re-clipped to the ε-box. *(c) Hybrid-unified mode,* in which the child inherits p_1's (k_e, k_f) split and genes. Within each half only the first $\min(k_{p_1}, k_{p_2})$ positions are eligible, where k_p counts that parent's slots in the half; each eligible slot is taken from p_2 with probability $1/2$, the remaining positions keep p_1's genes, and p_2's surplus genes are ignored, so a $(1,0)$ parent crossed with a $(0,1)$ parent returns a copy of p_1. A swapped load slot takes p_2's load index and the *unweighted mean* of the two parents' deltas at that position, which may belong to different loads, re-clipped to the ε-box. The split itself is never changed by crossover; only mutation moves genes across it.
- MUTATE: each topology slot is resampled uniformly from the edge pool independently w.p. p_{mut} and each load slot resamples its index or its delta w.p. p_{mut}; in unified mode an additional *split-shift* move (probability p_{mut}) migrates one gene between the topology half and the load half, with the replacement gene resolved *before* the old one is dropped so the budget k is never momentarily under-spent.
- TOP-N: elitist $(\mu+\lambda)$ replacement — pool parents and offspring ($2N$ individuals) and keep fittest N.

Algorithm 2 EvA+ — elitist genetic attack search

Require: surrogate f_θ, clean state x^0, budget k, cap ε, population size N, generations T, tournament size n_{tour}, crossover points k_{cross}, mutation rate p_{mut}, patience P

$\mathcal{P}_0 \leftarrow \textsc{InitPopulation}(N, k, \varepsilon)$ ▷ i.i.d. random feasible-support chromosomes
$\Phi_0 \leftarrow \big(\Phi(c)\big)_{c\in\mathcal{P}_0}$
$B_0 \leftarrow \max \Phi_0$; stall $\leftarrow 0$
for $t = 1, \dots, T$ **do**
  for $j = 1, \dots, N$ **do** ▷ generate N offspring
    $p_1 \leftarrow \textsc{Tournament}(\mathcal{P}_{t-1}, \Phi_{t-1}, n_{\text{tour}})$
    $p_2 \leftarrow \textsc{Tournament}(\mathcal{P}_{t-1}, \Phi_{t-1}, n_{\text{tour}})$
    $c_j \leftarrow \textsc{Crossover}(p_1, p_2, k_{\text{cross}})$ ▷ mode-dependent; see text
    $c_j \leftarrow \textsc{Mutate}(c_j, p_{\text{mut}})$ ▷ resample z-slots / I, δ-slots / split-shift
    $c_j \leftarrow \textsc{ActionValidate}_{(\text{iv})}(c_j)$ ▷ clamp load deltas to ε-box $\cap$ load bounds;
  end for
  $\Phi^{\text{off}} \leftarrow \big(\Phi(c_j)\big)_{j=1}^{N}$
  $\big(\mathcal{P}_t, \Phi_t\big) \leftarrow \textsc{Top-}N\big(\mathcal{P}_{t-1} \cup \{c_j\}_{j=1}^{N},\ \Phi_{t-1} \cup \Phi^{\text{off}}\big)$ ▷ elitist $(\mu+\lambda)$ survivor selection
  $B_t \leftarrow \max \Phi_t$
  stall $\leftarrow (B_t \le B_{t-1})\,?\,\text{stall} + 1 : 0$
  if stall $\ge P$ **then break**
  end if
end for
$c^\star \leftarrow \arg\max_{c\in\mathcal{P}_t} \Phi(c)$
$c^\star \leftarrow \textsc{ActionValidate}_{(\text{i})-(\text{iii})}(c^\star)$ ▷ repair topology once: canonical, parallel-branch, connectivity (this step is step conditional on RQ1);
return $c^\star$

Action validity in RQ1. In RQ1, validity is enforced in two tiers. Validity is enforced in two tiers. The first is the action space itself. The original EvA (Akhondzadeh et al., 2025) perturbs an adjacency matrix, where adding an edge is as meaningful as deleting one; on a transmission grid an "added" line is not a physical object, so EvA+ draws its topology genes from the columns of the in-service branch set. An addition is therefore never proposed in the first place, and addressing candidates by branch column rather than by endpoint pair is what preserves the identity of parallel lines, which share an endpoint pair but carry distinct impedances. The second tier repairs, or discards, whatever is still not deployable.

In RQ1, ActionValidate applies the topology conditions to the canonical undirected AC-line graph, stored once in the $i < j$ orientation (175 branches for case118), and bounds load perturbations separately. **(i) Canonical topology:** proposed edges with no counterpart in the original template are dropped and logged, so the realized action space consists of removals of existing branches. This condition is a guard rather than a routine correction in the production path: the restricted EvA+ pool and the branch-index PRBCD and GRBCD adaptations both draw from existing branch columns, so neither can propose such an edge.**(ii) Parallel-branch identity:** power-grid cases may contain parallel branches that share an endpoint pair but carry distinct impedances, so the repair operates column-wise on the edge index rather than on the set of endpoint pairs. When such a pair exists, the removal of one branch preserves the identity of the surviving branch. **(iii) Connectivity in RQ1:** After search, each selected AC-line removal is kept only if its endpoints remain reachable through the remaining AC lines; otherwise that removal is reverted. This is the endpoint-reachability check implemented by the repair. The intact AC-only case118 graph has two components among buses incident to AC lines (four if the two buses with no AC-line edge are counted), whereas the full graph including transformers is connected. Of the 175 possible single AC-line cuts, this AC-only check rejects 15; eight of those leave the full graph connected through transformer paths. RQ2 does not apply this repair. **(iv) Load bounds:** every load delta is sampled uniformly on the ε-box and clamped to the physical load bounds at draw time and again when applied, so the evaluated load state lies in $\mathcal{B}(z, I) = [-\varepsilon, \varepsilon]^d \cap [\ell - x,\ u - x]$; crossover re-clips its blended deltas to the ε-box only, and the clamp at application time is what guarantees feasibility.

Because conditions (i)–(iii) repair only the best RQ1 chromosome returned after the search, the fitness Φ optimized in Algorithm 2 is evaluated on unrepaired chromosomes; therefore, the conver-

Table 3: EvA as published versus EvA+. Rows marked *inherited* are used unchanged, so they are not claimed as contributions. Active-power scoring excludes zero-$P_{\max}$ synchronous compensators.

Component	**EvA (as published)**	**EvA+ (ours)**
Edge action	*Flip*: $\tilde{A} = A \oplus P$, so an entry may be added or removed	*Removal* of an in-service branch; additions are never proposed
Candidate set	Node pairs of the graph	The grid's existing branches, addressed by branch column so parallel lines keep their identity
Genes	Edge indices only	Branch removals *and* load genes (I, δ) under one shared budget k, with I drawn from a gradient-ranked load pool
Search space	Discrete	Mixed discrete–continuous
Fitness	Task score $\mathrm{fit}(s) = \mathcal{L}(X, A \oplus P, y)$, instantiated by classification accuracy	Physical constraint violation Φ (or Φ_{mse} below for the unconstrained baseline)
Guide model	Node classifier under attack	Frozen AC-OPF surrogate: PEGAS uses the physics-informed guide; the surrogate-guided baselines use an otherwise identical guide trained without the physics residual
Mutation	Random index resampling; targeted variants restrict an endpoint to the attacked set	Uniform resampling from the branch pool, load index/delta resampling, and a *split-shift* move that migrates one gene across the topology/load split at fixed k
Crossover	Joint concatenation $s_1[:j] \cdot s_2[j{+}1{:}]$	Unified recombination over the shared split, blending the continuous load deltas
Validity	Flip budget, plus an optional local degree cap	Budget *and* the grid conditions (i)–(iv)
Parent selection	Tournament of size n_{tour}	*inherited*
Survivor selection	Elitist: the best individuals are kept across generations	*inherited*
Batched evaluation	Population scored in one batch	*inherited*

gence results of Supp. B.4 characterize the optimum of that unrepaired objective, and the repair can reduce the realized fitness of the deployed attack. The load component has no comparable gap, since the load state is clamped in $\mathcal{B}(z, I)$ whenever it is evaluated. The guarantee the two tiers provide is therefore about what is *deployed*: The RQ1 projection prevents a new split in the modeled AC-only graph; it does not guarantee AC power-flow convergence. For a topology-only $k = 1$ attack, a rejected cut yields no applied topology change. The reports record both requested and applied budgets. The RQ2 opponent submits its selected line without this projection.

B.3.2 From EvA to EvA+

EvA+ keeps the evolutionary skeleton of EvA (Akhondzadeh et al., 2025) and changes the parts that are specific to attacking a graph classifier rather than a power grid. Table 3 states each difference; the components not listed are inherited unchanged.

Action space. The move from edge flips to branch removals is not a restriction of convenience but the first tier of validity, together with the branch-column addressing that keeps parallel lines distinct.

Objective. EvA optimizes a classification score, which has no direct counterpart for a regression surrogate that outputs setpoints and voltages. The closest transposition, used for the unconstrained baselines of RQ1, maximizes the deviation of the prediction from its clean-state value,

$$\begin{aligned}\Phi_{\mathrm{mse}}(z, \delta) = \tfrac{1}{2}\,\mathrm{MSE}\big(\hat{y}^{\mathrm{gen}}(x(z,\delta)),\, \hat{y}^{\mathrm{gen}}(x^0)\big) \\ + \tfrac{1}{2}\,\mathrm{MSE}\big(\hat{y}^{\mathrm{bus}}(x(z,\delta)),\, \hat{y}^{\mathrm{bus}}(x^0)\big),\end{aligned} \tag{11}$$

which rewards any large deviation, whether or not it is physically harmful. EvA+ instead maximizes the constraint-violation fitness, repeated here so the two objectives can be read side by side,

$$\begin{aligned}\Phi(z,\delta) = &\sum_{i\in\mathcal{G}_P} \mathrm{viol}\big(\hat{y}^{\mathrm{gen}}_{i,P}(x(z,\delta));\, P^{\min}_i, P^{\max}_i\big)\\ &+ \sum_{i\in\mathcal{G}_Q} \mathrm{viol}\big(\hat{y}^{\mathrm{gen}}_{i,Q}(x(z,\delta));\, Q^{\min}_i, Q^{\max}_i\big)\\ &+ \sum_{b\in\mathcal{B}} \mathrm{viol}\big(\hat{y}^{\mathrm{bus}}_{b,V_m}(x(z,\delta));\, V^{\min}_b, V^{\max}_b\big).\end{aligned} \tag{12}$$

where $\mathrm{viol}(\hat{u};\ell,u) = \max(0,\ell-\tau-\hat{u}) + \max(0,\hat{u}-u-\tau)$. Here $\hat{y}^{\mathrm{gen}}_{i,P}$ and $\hat{y}^{\mathrm{gen}}_{i,Q}$ are the active- and reactive-power components of the generator prediction, while $\hat{y}^{\mathrm{bus}}_{b,V_m}$ is the voltage-magnitude component of the bus prediction. $\mathcal{G}_P = \{i : P^{\max}_i - P^{\min}_i > \tau\}$ excludes generator rows with no active-power capability (the zero-$P_{\max}$ synchronous compensators in the case118 reports), whereas $\mathcal{G}_Q$ retains the reactive-power rows. The code computes each component as the maximum of the lower- and upper-bound hinge terms; this is identical to the viol definition above because, for valid bounds, the two hinge terms cannot both be positive. The two objectives read the *same* prediction and differ only in what they reward: Φ_{mse} pays for moving it anywhere, Φ pays only for moving it outside the operating limits, which is what steers the search toward states the grid cannot sustain. Both are scored on the frozen surrogate inside the loop; the power-flow projected metrics we report are computed after the search and never enter it.

B.4 Convergence analysis

We analyze Algorithm 2 as a time-homogeneous Markov chain $\{(\mathcal{P}_t, \Phi_t)\}_{t\geq 0}$ on population states, and prove that its best-found fitness converges to the global optimum of the Constrained Problem almost surely as $t \to \infty$. We first treat the finite discrete case (topology-only, $k_f \equiv 0$), where the argument is exact, then extend it to the mixed discrete–continuous (hybrid) case under a mild regularity assumption on the mutation kernel.

B.4.1 Discrete case: topology-only search

Fix $k_f = 0$. The chromosome space $\Omega = \{\, S \subseteq E : |S| = k \,\}$ is finite, $|\Omega| = \binom{m}{k}$. Let $F^\star = \max_{S\in\Omega} \Phi(S)$ and let $z^\star \in \Omega$ attain it. Let $B_t = \max_{c\in\mathcal{P}_t} \Phi(c)$.

Lemma 1 (Elitist monotonicity)**.** *B_t is non-decreasing: $B_{t+1} \geq B_t$ for all $t \geq 0$.*

Proof. $\mathcal{P}_t$ is a subset of the pooled parent–offspring set at step $t+1$ (line 13, Algorithm 2), so $\max\big(\Phi_t \cup \Phi^{\mathrm{off}}\big) \geq \max \Phi_t = B_t$, where Φ^{off} is the set of offspring fitnesses. Top-N selection retains the single fittest individual of the pooled set among its N survivors (since $N \geq 1$), hence $B_{t+1} = \max \Phi_{t+1} = \max\big(\Phi_t \cup \Phi^{\mathrm{off}}\big) \geq B_t$. □

Lemma 2 (Positive mutation reachability)**.** *There is a constant $\pi_{\min} = \pi_{\min}(m,k,p_{\mathrm{mut}}) > 0$, independent of t and of the parent chromosome, such that for* every *parent $z \in \Omega$ and* every *target $z' \in \Omega$, the probability that* Mutate *applied to any chromosome (regardless of what* Crossover *produced) outputs exactly z' is at least $\pi_{\min}$.*

Proof. Condition on the event E_{all} that all k topology slots are selected for mutation, which occurs with probability $p^k_{\mathrm{mut}} > 0$ since $p_{\mathrm{mut}} \in (0,1)$. On E_{all}, every slot is resampled i.i.d. uniformly over the edge pool (size $\geq m \geq k$), independent of the pre-mutation genotype — hence independent of whatever Crossover produced. The k slots are drawn uniformly and are then passed through duplicate repair, which resamples colliding slots from the unused remainder of the pool; the resulting joint law is therefore not a product measure, but it is supported on all of $\binom{m}{k}$ and assigns each target set a probability bounded below by a constant $\pi_0(m,k) > 0$ that depends only on m and k (it suffices to bound the probability of the collision-free event, on which the k draws are a uniformly random ordered k-tuple of distinct edges). This bound is independent of the parent and of t. Hence $\Pr[\textsc{Mutate}(\cdot) = z'] \geq p^k_{\mathrm{mut}} \cdot \pi_0(m,k) =: \pi_{\min} > 0$. □

Theorem 1 (Almost-sure convergence, discrete case; restates Theorem 1)**.** *Let* $q = 1 - \big(1 - \frac{1}{N}\big)^{n_{\text{tour}}} \in (0, 1]$ *be the probability that the tournament in a single offspring draw selects the current best individual as a parent (present in* $\mathcal{P}_{t-1}$ *by Lemma 1), and let* $\rho = q \cdot \pi_{\min} > 0$*. Then for every* $T \geq 1$*,*

$$\Pr\left[B_t < F^\star \ \forall\, t \leq T\right] \ \leq \ (1-\rho)^{NT},$$

and consequently $B_t \to F^\star$ *almost surely as* $t \to \infty$*.*

Proof. Fix generation t. The elite chromosome achieving B_{t-1} is physically present in $\mathcal{P}_{t-1}$ (Lemma 1's proof). Because it strictly dominates any tournament it participates in, it is chosen as parent p_1 of a given offspring slot j with probability at least q (probability the elite is among the n_{tour} uniformly-drawn tournament members). Conditioned on p_1 = elite, Lemma 2 gives $\Pr[c_j = z^\star \mid p_1 = \text{elite}] \geq \pi_{\min}$ regardless of p_2 and of CROSSOVER's output (mutation overwrites it on E_{all}). Hence $\Pr[c_j = z^\star] \geq q\pi_{\min} = \rho$ for every offspring slot j, independently across $j = 1, \ldots, N$ (independent randomness per offspring). So

$$\Pr\left[\text{no offspring} = z^\star \text{ at generation } t\right] \leq (1-\rho)^N,$$

and if some offspring equals $z^\star$ at generation t, elitist TOP-N selection guarantees $z^\star \in \mathcal{P}_t$, hence $B_t \geq \Phi(z^\star) = F^\star$, i.e. $B_t = F^\star$ (as $F^\star$ is the global max). Iterating the conditional bound and applying the tower property,

$$\Pr\left[B_t < F^\star \ \forall\, t \leq T\right] \leq \prod_{t=1}^{T} (1-\rho)^N = (1-\rho)^{NT} \xrightarrow[T\to\infty]{} 0.$$

By Lemma 1, once $B_t = F^\star$ it remains so for all larger t; the above shows $\Pr[\exists t : B_t = F^\star] = 1$, hence $B_t \to F^\star$ almost surely. □

Corollary 1 (Expected hitting time)**.** *Let* $T_{\text{hit}} = \min\{t : B_t = F^\star\}$*. Then*

$$\mathbb{E}[T_{\text{hit}}] \ \leq \ \frac{1}{1-(1-\rho)^N}.$$

Proof. T_{hit} is $\mathbb{N}$-valued and a.s. finite by Theorem 2, so $\mathbb{E}[T_{\text{hit}}] = \sum_{T\geq 0} \Pr[T_{\text{hit}} > T]$. For $T \geq 1$, $\{T_{\text{hit}} > T\} = \{B_t < F^\star \ \forall t \leq T\}$ has probability at most $(1-\rho_N)^{NT}$ by Theorem 2; at $T = 0$ the bound is trivial. Summing the geometric series gives the claim. □

Remark 1 (Effect of the population size)**.** *ρ_N depends on N through the tournament term, and $\rho_N \to 0$ as $N \to \infty$. The operative quantity is $N\rho_N$, which is non-decreasing in N (difference quotient at 0 of the concave map $x \mapsto 1-(1-x)^{n_{\text{tour}}}$) and increases to $n_{\text{tour}}\pi_{\min}$. Hence $(1-\rho_N)^N \to e^{-n_{\text{tour}}\pi_{\min}}$, and $(1-\rho_N)^N \geq \exp\big(-n_{\text{tour}}\pi_{\min}/(1-\pi_{\min})\big)$ for every N. The generation bound therefore saturates near $\big(1-e^{-n_{\text{tour}}\pi_{\min}}\big)^{-1} \approx (n_{\text{tour}}\pi_{\min})^{-1}$, improving on the $N = 1$ value $\pi_{\min}^{-1}$ by at most a factor n_{tour} ($= 2$ in our configuration), while the query bound $N \cdot \mathbb{E}[T_{\text{hit}}]$ grows linearly in N. Corollary 1 thus does not justify a large population; our choice of $N = 1024$ rests on the fact that a generation's N evaluations are batched into one GPU forward pass.*

Corollary 2 (Explicit hitting-time bound)**.** *In the setting of Theorem 2 with $k_f = 0$, the constant of Lemma 2 satisfies $\pi_{\min} \geq p_{\text{mut}}^k\, k!/m^k$, and consequently*

$$\mathbb{E}[T_{\text{hit}}] \ \leq \ 1 + \frac{1}{N\rho_N} \ \leq \ 1 + \frac{1}{\pi_{\min}} \ \leq \ 1 + \frac{1}{p_{\text{mut}}^k} \cdot \frac{m^k}{k!},$$

and since $m^k/k! \geq \binom{m}{k} = |\Omega|$, the bound is at least $|\Omega|\, p_{\text{mut}}^{-k}$ generations.

Proof. On the collision-free event of Lemma 2's proof the k resampled slots form a uniformly random ordered k-tuple of distinct edges, so any prescribed target set is produced by exactly $k!$ of the m^k equally likely tuples, giving $\pi_0(m,k) \geq k!/m^k$ and $\pi_{\min} \geq p_{\text{mut}}^k k!/m^k$. The first inequality is the sharpened form of Corollary 1 ($1/(1-e^{-x}) \leq 1 + 1/x$ with $x = N\rho_N$), the second uses $N\rho_N \geq \pi_{\min}$, and the third substitutes the bound on $\pi_{\min}$. □

B.4.2 EXTENSION: HYBRID DISCRETE–CONTINUOUS SEARCH

We now allow $k_f > 0$. Under the shared budget the split is itself part of the chromosome: an individual carries $|S| = k_e$ topology genes and $|I| = k_f$ load genes with $k_e + k_f = k$, and the pair (k_e, k_f) varies across the population. The chromosome space is therefore

$$\Omega_{\text{hyb}} = \bigsqcup_{(S,I):\ |S|+|I|=k} \{S\} \times \{I\} \times \mathcal{B}(S, I),$$

a finite union of compact continuous boxes $\mathcal{B}(S, I) \subseteq [-\varepsilon, \varepsilon]^{|I|\times d}$ as defined in Action Validity (i.e., restricted to realized search space) , taken over the discrete choices (S, I), which include the choice of split.

Assumption 1 (Continuity of the objective)**.** *$\Phi(z, \delta)$ is continuous in δ for every fixed z.* (True by construction: f_θ is a continuous neural network, and Φ composes it with the hinge functions $\max(0, \cdot)$, themselves continuous.)

Assumption 2 (Mutation kernel lower bound)**.** *Conditional on all k_f load slots being resampled by* MUTATE *(probability $p_{\text{mut}}^{k_f} > 0$), each resampled delta is drawn independently from a law κ that dominates Lebesgue measure on the* feasible *region $\mathcal{B}(z, I) = \prod_{i\in I} \big([-\varepsilon, \varepsilon]^d \cap [\ell_i - x_i,\ u_i - x_i]\big)$: there is $c_0 > 0$ with $\kappa(U) \geq c_0 \operatorname{vol}(U)$ for every measurable $U \subseteq \mathcal{B}(z, I)$.*

This is the weakest form the argument needs, and it is the form the implementation satisfies. EVA+ draws each delta uniformly on the ε-box and then clamps it to the physical load bounds. The resulting law is not absolutely continuous, since clamping deposits atoms on $\partial\mathcal{B}(z, I)$, so no pointwise density bound holds on all of $\mathcal{B}(z, I)$. Clamping only transports mass from outside $\mathcal{B}(z, I)$ onto its boundary and never removes mass from a set U contained in $\mathcal{B}(z, I)$, so $\kappa(U) \geq \operatorname{vol}(U)/(2\varepsilon)^d$ and the assumption holds with $c_0 = (2\varepsilon)^{-d}$. Since $\delta^\star$ is feasible it lies in $\mathcal{B}(z, I)$, and the argument of Theorem 2 goes through with all neighbourhoods taken relative to $\mathcal{B}(z, I)$.

Theorem 2 (Almost-sure convergence, hybrid case at a fixed split)**.** *Fix a split $(k_e^\star, k_f^\star)$ with $k_e^\star + k_f^\star = k$ and let $\Omega_{\text{hyb}}^{(k_e^\star, k_f^\star)} \subseteq \Omega_{\text{hyb}}$ be the corresponding sub-space. Under Assumptions 1–2, with $F^\star = \sup\{\Phi(z, \delta) : (z, \delta) \in \Omega_{\text{hyb}}^{(k_e^\star, k_f^\star)}\}$ (attained, by continuity and compactness), the elitist* EVA+ *of Algorithm 2 restricted to that split satisfies $B_t \uparrow F^\star$ almost surely as $t \to \infty$.*

Proof. Fix $\eta > 0$ and let $(z^\star, \delta^\star)$ attain $F^\star$. By Assumption 1, there is $r = r(\eta) > 0$ such that $\Phi(z^\star, \delta) \geq F^\star - \eta$ for every δ with $\|\delta - \delta^\star\|_\infty \leq r$ (continuity at $\delta^\star$, on the compact box). Let $(z^\star, I^\star, \delta^\star)$ denote the target, where $I^\star$ is its set of $k_f^\star$ load indices. We bound the probability that a single offspring realises it, treating the topology and load genes separately, as the mutation operator does.

Topology genes. Each edge slot is resampled uniformly from the edge pool with probability p_{mut}, so conditioning on all $k_e^\star$ edge slots being resampled (probability $p_{\text{mut}}^{k_e^\star}$) the topology part realises $z^\star$ with probability at least $\pi_0(m, k_e^\star) > 0$, by Lemma 2's argument applied to the discrete component.

Load genes. Load mutation acts per slot: with probability p_{mut} the slot is mutated, and then with probability $\frac{1}{2}$ its index is resampled from the load pool *and* its delta is redrawn at the new index, the alternative branch keeping the index and redrawing only the delta. Reaching a prescribed index set $I^\star$ therefore requires the index-resampling branch on every load slot, which conditions on an event of probability $(p_{\text{mut}}/2)^{k_f^\star}$. Conditioned on that event, the slots realise $I^\star$ with probability at least a positive constant $\pi_I(n_{\text{load}}, k_f^\star) > 0$ that depends only on the load-pool size and $k_f^\star$ (the sequential index draws avoid duplicates but are supported on every $k_f^\star$-subset of the pool, exactly as in Lemma 2); and, conditioned in turn on $I^\star$ being realised, each delta is redrawn at its target index by the bounds-respecting sampler, so by Assumption 2 every resampled coordinate of δ lands within r of $\delta_i^\star$ with probability at least $\big(c_0 \cdot \min(2r, 2\varepsilon)\big)^{k_f^\star d} =: \pi_\delta(r) > 0$, the neighbourhood being taken relative to $\mathcal{B}(z^\star, I^\star)$.

Since the split is held fixed at $(k_e^\star, k_f^\star)$ by hypothesis, it suffices that no split-shift fires in the generation considered, which has probability $1 - p_{\text{mut}}$. Conditioning as before on the elite parent winning

its tournament (probability q), and multiplying the independent events above, the per-offspring probability of producing an individual with fitness $\geq F^\star - \eta$ is at least

$$\begin{aligned}\rho(\eta) &= q\,(1-p_{\mathrm{mut}})\,p_{\mathrm{mut}}^{k_e^\star}\,\pi_0(m,k_e^\star) \\ &\quad\cdot\left(\tfrac{p_{\mathrm{mut}}}{2}\right)^{k_f^\star}\,\pi_I(n_{\mathrm{load}},k_f^\star)\,\pi_\delta\big(r(\eta)\big) \\ &> 0,\end{aligned}$$

so $\Pr[B_t < F^\star - \eta\ \forall t \leq T] \leq (1-\rho(\eta))^{NT} \to 0$, hence $\Pr[\limsup_t B_t \geq F^\star - \eta] = 1$. Since B_t is non-decreasing (Lemma 1, whose proof did not use finiteness of Ω) and bounded above by $F^\star$, $B_t \uparrow B_\infty \leq F^\star$ almost surely for some limit B_∞; the above gives $\Pr[B_\infty \geq F^\star - \eta] = 1$ for every $\eta > 0$. Intersecting over a countable sequence $\eta_n \downarrow 0$ yields $\Pr[B_\infty = F^\star] = 1$. □

C Supplementary C: Experimental Protocol

We evaluate PEGAS in two settings. In RQ1 the attacker perturbs a static grid and the outcome is scored by an AC optimal power flow solver; in RQ2 the attacker acts as a Grid2Op opponent against a defender that controls the grid over time. The surrogate guides the attacker in both settings but is never the victim: the victim is the grid together with its solver in RQ1, and the grid together with its defender in RQ2. The two settings use different data and metrics, described separately below. All values reported here are read from the training and run artifacts (checkpoint configuration dictionaries, training configuration dataclasses, launcher-captured defaults, and the per-run JSON reports); the launch scripts are used only as secondary context. The complete verbatim configurations, including data-schema column indices and logging flags that are fixed and not tuned, are listed in the supplementary material.

Instantiating Algorithm 1. The Algorithm 1 is stated for a generic environment; the two settings realize its objects differently, and several of them degenerate. In RQ1 the environment is a dataset of grid cases and a solver, so the horizon is $H = 1$, an observation is one OPFData sample, there is no defender, and π_{def} is the identity, nd the transition is one AC power-flow projection of the surrogate's predicted dispatch (Supp. C.4), with no corrective redispatch. The admissibility rule does not place restrictions beyond the budget k, the validity conditions apply in full, and every reported number is computed after the power-flow projection of §C.4. In RQ2, the environment is a Grid2Op episode over chronological load and generation data, π_{def} is the defender under test, and the transition is one environment step. The admissibility rule is the native opponent schedule described in the following. Here we apply neither the RQ1 connectivity repair nor a separate power-flow projection. Grid2Op checks line-action legality and advances its solver, but legality does not require the resulting network to stay connected. An accepted line cut may therefore island the grid and end the episode through power-flow failure. The attack interface also differs. RQ1 accepts arbitrary perturbations of the state, so PEGAS runs EvA+ there with both gene types, whereas the Grid2Op opponent slot accepts only line-status actions and admits no load perturbation, so PEGAS runs the topology-only EvA+ in RQ2. The same frozen `phys_cosine` surrogate guides both settings.

C.1 RQ1: attacks against static AC-OPF solvers

Data. Grid states come from the OPFData dataset (Lovett et al., 2024) for the 118-bus IEEE case (`pglib_opf_case118`), loaded as 20 OPFData groups. Each group contains 15,750 grid states, split 13,500/750/750/750 into training, validation, N test, and $N-1$ test, so the 20-group case118 build holds 270,000 training, 15,000 validation, 15,000 N-test, and 15,000 $N-1$-test states; the four-group cross-grid builds (case30 and case2000) hold 54,000/3,000/3,000/3,000 each. We attack the OPFData *test* split, evaluating both the intact grid (denoted N) and its single-branch contingencies ($N-1$). Attacks are represented on the canonical undirected AC-line graph, stored once in the $i < j$ orientation with 175 branches. For each of five seeds $\{42, 43, 44, 45, 46\}$ we draw 50 samples with a disjoint seeded selection. The implemented logic is deterministic and recorded in each report. For split x and sorted seed cohort $C = (42, 43, 44, 45, 46)$, the runner first checks that $50|C| \leq |\mathcal{D}_x|$. It then builds a split-specific random permutation π_x of the split indices using a seed sequence formed from the global selection seed 42 and a 32-bit salt derived from the SHA-256 hash of the split name. If active seed s has cohort position j, it receives $\pi_x[50j : 50(j+1)]$. Thus cohorts

are non-overlapping within each split and the five seeds cover 250 samples from the N test split and 250 samples from the $N-1$ test split. The two split cohorts are selected independently rather than index-paired. Within a selected cohort, stochastic attack components are also deterministic: for sample index i, the attack seed is $s + i$. Per-generator active and reactive bounds are taken from the PGLIB case file (Babaeinejadsarookolaee et al., 2019) at scoring time; zero-$P_{\max}$ synchronous compensators are excluded from generator scoring.

Surrogate architecture. The surrogate is a graph attention network (Veličković et al., 2018) over the heterogeneous (`bus`, `ac_line`, `bus`) graph. It has 3 attention layers with hidden dimension 128 and dropout 0.2; the first layer uses 4 attention heads and the other two are single-headed. The convolution accepts an optional per-branch availability in $[0, 1]$, whose logarithm is added to the attention logit before the softmax. At availability 1 this is an exact no-op, so it is inactive during training and whenever EvA+ queries the network; EvA+ instead applies a candidate topology by removing the corresponding branches from the input graph. The gate is used only where a continuous relaxation of a branch removal is required, namely by the PRBCD and GRBCD baselines, described below.

Surrogate training. All case118 surrogates in the main RQ tables use the same training budget, taken from the checkpoint configuration dictionaries. Unless a row states otherwise, the backbone is GAT with the architecture above. Training uses the Adam optimiser with learning rate 10^{-3}, weight decay 10^{-5}, a cosine annealing schedule down to 10^{-5}, gradient clipping at 1.0, batch size 128, and up to 150 epochs with early stopping (patience 30, minimum improvement 10^{-5}); the seed is 42 and training uses the intact-grid data (case N). Generator and bus predictive terms carry equal weight. The physics-informed guide (`phys_cosine`), which PEGAS uses, adds the Kirchhoff residual of Supp. B.2 to that predictive loss with weight $\lambda = 0.05$, evaluated every fifth optimisation step so its cost stays small next to the supervised pass; the predictive guide (`mse_cosine`), which the surrogate-guided baselines use, is identical in every other respect but omits the residual.

Training reproducibility. The dataset split is fixed by OPFData: training uses `split=train`, validation uses `split=val`, and reporting uses `split=test` with and without $N-1$ perturbations. The training run fixes the Python, NumPy, PyTorch, and CUDA random seeds to 42, records the full configuration and software versions, and stores the checkpoint used for all reported evaluations. The training loader shuffles the training split with multi-worker loading, and we do not enable bitwise deterministic CUDA kernels. Therefore retraining the same recipe reproduces the protocol but can yield small numerical differences; evaluating the released checkpoints reproduces the tables.

Attackers. Table 5 lists the attack configurations, all read from the per-run reports. Every attack in this main comparison uses a single-perturbation budget ($k = 1$) and evaluates the selected perturbation with the AC-OPF solver. EvA and EvA+ are evolutionary, gradient-free attacks: they evolve a population of candidate perturbations over generations, score each candidate by the surrogate's predicted constraint violation, and keep the fittest, so they need only forward queries of the surrogate. PEGAS runs EvA+ in hybrid mode (topology and load genes under one shared budget) guided by `phys_cosine`; our EvA baseline (Akhondzadeh et al., 2025) runs the same genetic search in topology-only mode guided by `mse_cosine`. PRBCD and GRBCD (Geisler et al., 2021) are gradient-based topology attacks. In our grid implementation they are branch-index multigraph adaptations: they optimize over existing AC-branch columns, preserve parallel-line identity, and commit removals rather than non-physical edge additions. PGD is a projected-gradient attack on the load features under an L_0 node budget (Madry et al., 2018). These three are guided by `mse_cosine`. Exhaustive enumerates all $\binom{175}{1} = 175$ single-branch removals and scores each with the solver, so it is model-agnostic; its reported fidelity is dispatched through `mse_cosine`. The two component ablations reuse the `phys_cosine` guide rather than a new checkpoint: *w/o load* reruns it with topology-only EvA+, and *w/o topology* reruns it in load-only mode, isolating the contribution of each gene type. The same budget k governs every mode, so the ablation rows are directly comparable: at k=1 the topology-only arm removes one branch, the load-only arm perturbs one load bus, and the hybrid arm spends its single perturbation on whichever gene type the search prefers. Load genes are restricted to the buses the surrogate is most sensitive to: a single gradient pass on the clean state ranks loads by the L_2 norm of a predictive-loss gradient at each load row, the

Table 4: RQ1 attack configurations, read from the per-run reports. Tournament size is the only entry the reports do not serialize; it is the configuration default, which no launcher overrides. k is the perturbation budget; the load perturbation cap is $\varepsilon = 0.02$ pu, recorded per sample in the reports.

Attack	Configuration
PEGAS (EvA+)	pop. 1024, gen. 500, tourn. 2, k_{cross} 30, p_{mut} 0.01, patience 30, k=1, hybrid, ε=0.02; guide `phys_cosine`
EvA	as above, topology-only; guide `mse_cosine`
PRBCD / GRBCD	block 64, epochs 125, resampling 100, step 10^3, k=1; guide `mse_cosine`
PGD	L_0, node budget 1, 30 steps, 3 restarts, ε=0.02; guide `mse_cosine`
Exhaustive	all 175 single-branch removals; solver-scored

$4k$ most sensitive form a candidate pool, and the search selects which k of that pool to perturb and by how much, additively, capped at ε and clipped to the load bounds.

Optional line-thermal fitness. The implementation includes an opt-in diagnostic only surrogate-derived line-thermal fitness term. It is left at its default weight 0 in the RQ1 attacker configurations above, so line constraints enter the static tables through the post-search power-flow projected evaluation, not through the search objective.

Unconstrained baseline diagnostics. The OPF-specific constraints that EvA+ adds are exposed as explicit switches, so the unconstrained behaviour of the published attacks can be reproduced from the same code path. `--original-implementation` restores the topology-only search over complete-graph flips with the MSE objective and no repair; it fails loudly unless the run is topology-only and uses that objective. `--skip-projection` disables static topology repair; in EvA+ feature/hybrid runs it also disables load-bound clipping during the search. `--skip-bounds` disables load clipping for the gradient feature attack. Every run records which switches were active, so a report states its own configuration. These are diagnostics for the baselines of RQ1;PEGAS never uses them.

Evaluation. The AC-OPF oracle is solved with PowerModels (Coffrin et al., 2018) and Ipopt (Wächter & Biegler, 2006) in Julia. The headline metric is the power-flow projected success rate, which counts a sample as broken when the post-attack state exceeds the sample's clean power-flow baseline on the generator, bus, or line checks. We also report power-flow projected robust physical accuracy and mean squared error as descriptive context; generator accuracy scores active power only, with reactive-power exceedances kept as a diagnostic. Every figure is the mean $\pm$ standard deviation over the five seeds.

C.2 RQ2: ATTACKS AGAINST GRID2OP DYNAMIC AGENTS

Environments. We use the Grid2Op environments (Donnot, 2020a) `l2rpn_idf_2023` and `l2rpn_wcci_2022` with the LightSim2Grid backend (Donnot, 2020b). For each of five seeds $\{42, \ldots, 46\}$ we evaluate 30 chronics selected by the baseline-random protocol used by our Grid2Op baseline scripts: . The same selected chronic IDs are used for every defender and opponent mode under a fixed environment and seed, so comparisons are paired within seed. Different seeds are sampled independently and may overlap. Each chronic receives the deterministic execution seed $(\text{seed} + \text{chronic_id}) \bmod (2^{31} - 1)$. The episode horizon is 2017 steps. All six defenders are available on the IDF environment; only the three transfer-compatible defenders (DoNothing, OptimCVXPY, Expert) run on WCCI.

Defenders. Each defender is a Grid2Op control agent that observes the grid at every step and may act to keep it within limits; we construct them with their library defaults. A defender's survival under an attacker measures how well it absorbs the attack.

- **DoNothing** is the null baseline: it never acts, so its survival reflects the grid's passive resilience to the opponent.
- **Solver** is a traditional AC-OPF baseline that computes a single re-dispatch action; it survives only 5.3% of episodes even without an opponent, so it is already failing before any attack.
- **OptimCVXPY** (RTE France, 2024c) is a convex-optimisation redispatch agent. It solves a small optimisation each time the maximum line loading ρ reaches 0.95, redispatching generation and curtailment to relieve the overload, and returns toward the reference dispatch once ρ falls to 0.85. Its defaults are: thermal-limit margin 0.9, power-flow error weight 0.5, unsafe penalties on redispatch, curtailment, and storage of 0.03, 0.1, and 0.3 respectively (all safe-state penalties 0), unit target weights 1.0, and rounding and sparsity margins of 0.01 and 0.005.
- **Expert** (Marot et al., 2018) is a rule-based expert system that clears line overloads by substation reconfiguration; it has no learned parameters and is instantiated for the IEEE-118 grid. Each step it ranks overloaded lines ($\rho \geq 1$) by loading, prioritising those whose overflow counter has reached the environment's allowance, and treats at most 3 overloads. For each it simulates candidate substation topologies, at most 25 overall and 5 per node, scoring each from 0 to 4 (4 clears all overloads, 3 clears the target overload, 2 partially clears it, 1 clears it but worsens others, 0 fails) and ranking them with the `MinMargin` reward $\min(1-\rho)$. It accepts an action when the simulated state puts every ρ below the safe threshold 0.95. Its remaining defaults are a maximum of 2 unused lines, flow-direction reconsideration ratio 0.75, loop-retention ratio 0.25, minimum loop power 0.1, and line-report threshold 0.2.
- **LJN** is the reinforcement-learning topology agent that won the L2RPN 2023 competition (Pavão et al., 2025), used as its released pretrained policy. It intervenes when the maximum line loading exceeds $\rho_{\text{danger}} = 0.99$, evaluates substation reconfigurations by simulating candidates, and keeps one that holds the maximum loading below $\rho_{\text{safe}} = 0.9$; once the grid is safe it moves back toward the reference topology.
- **LJN-TopoNN** is the topology-network variant of that agent. It keeps the same $\rho_{\text{danger}} = 0.99$ and $\rho_{\text{safe}} = 0.9$ thresholds and adds a PPO policy over a Gym-wrapped box observation of line loadings and a precomputed global topology action space of 352 actions. Each step it ranks the top $\text{topk} = 20$ candidates (the class default is 10; our runs set 20) and applies the best feasible one, falling back to an $N{-}1$ heuristic and then an optimisation-based recovery when no learned action is safe. Every episode report records the top-k, the action-space size, and the checkpoint hash.

Opponent modes. Each RQ2 row uses one of three opponent modes on a given environment. *No opponent* runs the defenders with no attacks and measures baseline survival. The *default* opponent is Grid2Op's native stochastic line-attack opponent, the standard adversary shipped with each L2RPN environment. PEGAS (next paragraph) replaces that stochastic opponent with our surrogate-guided line selection while keeping the same native budget and scheduling machinery. The default opponent differs by environment. On `l2rpn_idf_2023` it is a `GeometricOpponentMultiArea`, which partitions attackable lines into three areas and attacks each area independently: it samples timing geometrically (roughly one attack every 32 hours of simulated time, mean duration 2 hours, minimum 1 hour, generator $p_{\max}/p_{\min}$ ratio 4) from an initial budget of 1000 regenerating at 0.51 per step, with attack duration 96 steps and cooldown 0. On `l2rpn_wcci_2022` it is a single-area `GeometricOpponent` over 23 attackable lines (roughly one attack every 24 hours, mean duration 4 hours, minimum 1 hour) from a smaller initial budget of 144 regenerating at 0.17 per step, with attack duration 96 steps and cooldown 12.

Attacker. PEGAS acts as the environment's native opponent. Every setting below is recorded in the run plan written beside each campaign, which also stores the SHA-256 of the guide checkpoint. PEGAS is guided by `phys_cosine` and selects line disconnections with the genetic selector

(`eva_ga`) in topology-only mode under budget $k = 1$, using the same `paper` preset as RQ1: population 1024, 500 generations, $k_{\text{cross}} = 30$, $p_{\text{mut}} = 0.01$, and early stopping after 30 stalled generations. Grid2Op exposes only line-status actions to the opponent, so PEGAS here runs topology-only EvA+ with no load genes, rather than the hybrid RQ1 variant. The final line is chosen by `argmax` over the top 5 scored candidates, with selection temperature 1.0, selection $\epsilon = 0.1$ and policy seed 0; the surrogate-derived thermal term is disabled (weight 0, `topk_mean` form, derived-bus-voltage source), so the score is the constraint-violation objective alone. Of the 175 canonical branches, 168 map onto distinct Grid2Op lines and form the candidate set at every attack, which the per-attack candidate counts in each episode report confirm; the remaining seven are parallel branches sharing endpoints with a mapped branch and are not separately targetable, and transformer branches are never attacked. The `selected_disconnecting` field is diagnostic telemetry computed from the fixed AC-line template; it is not a live RQ2 action filter and does not account for transformer paths.

Opponent schedule. The opponent operates under Grid2Op's native budget schedule: initial budget 1000, regeneration 0.51 per step, attack duration 96 steps, and cooldown 0. The sensitivity study varies one axis at a time from this default: initial budget in $\{500, 2000\}$, attack duration in $\{48, 192\}$, cooldown in $\{10, 100\}$, and candidate line masking that removes line 26, or lines 70 and 26, from the attacker's reachable set.

Evaluation. The metric is the defender survival rate, computed per seed as the fraction of the 30 chronics the defender completes without a blackout, then averaged as the mean $\pm$ standard deviation over the five seeds. Lower survival indicates a stronger attack.

C.3 Compute

Experiments ran on three remote servers. Two are identical: each has two Intel Xeon Silver 4416+ CPUs (40 physical cores, 80 threads), four NVIDIA L40S GPUs (46 GB each), and 503 GB of RAM. The third has two Intel Xeon Gold 5222 CPUs at 3.80 GHz (8 physical cores, 16 threads), three NVIDIA RTX A6000 GPUs (48 GB each), and 503 GB of RAM. Jobs ran in parallel across GPU slots or separately, according to the task: CPU-bound work such as the attack search used a single GPU with many parallel worker jobs, whereas GPU-bound work such as surrogate training used a full GPU. Surrogates and attacks are implemented in PyTorch (Paszke et al., 2019) and PyTorch Geometric (Fey & Lenssen, 2019). Static attacks solve AC-OPF with PowerModels and Ipopt in Julia; Grid2Op uses the LightSim2Grid backend.

Choice of the load perturbation cap ε

We set $\varepsilon = 0.02$ pu, anchored on how much the load naturally moves between two consecutive control steps rather than on a round number. Rolling a do-nothing agent through the IDF 2023 chronics and recording $\Delta P_l^d = P_l^d(t{+}1) - P_l^d(t)$ for every load at every step yields 658,152 per-load samples over 6,648 steps of operation. The natural per-step change is small: its 95th percentile is 0.006 pu and its 99th percentile is 0.010 pu, about 1 MW, or 2.13% of the load's own magnitude. Two independent readings of that tail agree on the same cap: twice the absolute 99th percentile is 0.02 pu, and the relative 99th percentile of 2.13% is 0.0213 pu. The percentile is also stable, moving only between 0.010 and 0.011 pu as the sample grows from chronic to twenty-four, so the anchor is not an artifact of the sample size. A cap at 0.02 pu therefore sits just above natural variation: large enough that the perturbation is not lost in it, since the 95th percentile is more than three times smaller, yet small enough to remain a plausible abuse of demand response rather than an infrastructure-scale event. For scale, a 0.5 pu cap would be comparable to a typical load itself rather than to its variation. Anchoring the cap on *unattacked* operation keeps the threat model realistic: the injection the attacker may command stays on the scale of the demand swings the grid already absorbs between two control steps, so a successful attack cannot be dismissed as the product of an implausibly large disturbance.

C.4 SCORING: PROJECTION TO A PHYSICAL STATE

The score maximized during the search is not the number we report; this subsection states how the physical state we actually score is obtained in each setting.

Static setting. The surrogate predicts a dispatch, but nothing constrains that prediction to satisfy Kirchhoff's laws, so scoring it directly would measure the surrogate's own error as much as the damage. We therefore project the prediction onto the physics: on the attacked grid, the predicted active power of every non-slack generator is held fixed and an AC power flow is solved for the remaining quantities, namely the bus voltage magnitudes and angles, the generator reactive power, and the slack generator's active power. The result satisfies the network equations and still reflects the dispatch the surrogate chose. Every number reported in the static setting is computed on that projected state rather than on the raw prediction, including the headline power-flow projected success rate of §5.1. The projection is a step of the *evaluation*, not of the attack: the search never sees it, and it is invoked once per sample after EVA+ returns, because a solve per candidate would dominate the cost of the search.

Agentic-loop setting. Against a Grid2Op defender the surrogate serves only to select the line to disconnect; its predicted dispatch never enters the environment, so there is no prediction to project. The environment instead runs its own AC power flow on the actual operating point, the generation and load of the scenario together with the defender's dispatch, and then plays out the cascading-failure dynamics: a line whose flow exceeds its thermal limit is disconnected, at once on a hard overflow and after a fixed number of steps on a sustained one, the flow is re-solved after each disconnection, and this repeats until no further line trips or the grid fails. The static projection differs on both counts, since its input is the surrogate's predicted dispatch rather than the real one and its behaviour is a single solve rather than a time-evolving cascade, so the physical state we score in the Grid2Op setting is the one the simulator produces, not one we compute.

Search proxy versus reported metric. The objective the search maximizes and the metric we report are not identical. EVA and EVA+ maximize a surrogate-space violation score that aggregates bus voltage, generator active-power and generator reactive-power exceedances; the reported ranking is computed only after the power-flow projection, and its per-element generator accuracy scores active power alone, with reactive-power exceedances kept as a diagnostic. A surrogate trained with a predictive loss cannot be expected to respect tight per-generator reactive limits, so the search objective is richer than the reported metric and is best read as a proxy that orders candidate perturbations rather than as the quantity we report.

We evaluate PEGAS in two settings. In RQ1 the attacker perturbs a static grid and the outcome is scored by an AC optimal power flow solver; in RQ2 the attacker acts as a Grid2Op opponent against a defender that controls the grid over time. The surrogate guides the attacker in both settings but is never the victim: the victim is the grid together with its solver in RQ1, and the grid together with its defender in RQ2. The two settings use different data and metrics, described separately below. All values reported here are read from the training and run artifacts (checkpoint configuration dictionaries, training configuration dataclasses, launcher-captured defaults, and the per-run JSON reports); the launch scripts are used only as secondary context. The complete verbatim configurations, including data-schema column indices and logging flags that are fixed and not tuned, are listed in the supplementary material.

Instantiating Algorithm 1. The Algorithm 1 is stated for a generic environment; the two settings realize its objects differently, and several of them degenerate. In RQ1 the environment is a dataset of grid cases and a solver, so the horizon is $H = 1$, an observation is one OPFData sample, there is no defender, and π_{def} is the identity, nd the transition is one AC power-flow projection of the surrogate's predicted dispatch (Supp. C.4), with no corrective redispatch. The admissibility rule does not place restrictions beyond the budget k, the validity conditions apply in full, and every reported number is computed after the power-flow projection of §C.4. In RQ2, the environment is a Grid2Op episode over chronological load and generation data, π_{def} is the defender under test, and the transition is one environment step. The admissibility rule is the native opponent schedule described in the following. Here we apply neither the RQ1 connectivity repair nor a separate power-flow projection. Grid2Op checks line-action legality and advances its solver, but legality does not require the resulting network

to stay connected. An accepted line cut may therefore island the grid and end the episode through power-flow failure. The attack interface also differs. RQ1 accepts arbitrary perturbations of the state, so PEGAS runs EvA+ there with both gene types, whereas the Grid2Op opponent slot accepts only line-status actions and admits no load perturbation, so PEGAS runs the topology-only EvA+ in RQ2. The same frozen `phys_cosine` surrogate guides both settings.

C.5 RQ1: ATTACKS AGAINST STATIC AC-OPF SOLVERS

Data. Grid states come from the OPFData dataset (Lovett et al., 2024) for the 118-bus IEEE case (`pglib_opf_case118`), loaded as 20 OPFData groups. Each group contains 15,750 grid states, split $13{,}500/750/750/750$ into training, validation, N test, and $N-1$ test, so the 20-group case118 build holds 270,000 training, 15,000 validation, 15,000 N-test, and 15,000 $N-1$-test states; the four-group cross-grid builds (case30 and case2000) hold $54{,}000/3{,}000/3{,}000/3{,}000$ each. We attack the OPFData *test* split, evaluating both the intact grid (denoted N) and its single-branch contingencies ($N-1$). Attacks are represented on the canonical undirected AC-line graph, stored once in the $i < j$ orientation with 175 branches. For each of five seeds $\{42, 43, 44, 45, 46\}$ we draw 50 samples with a disjoint seeded selection. The implemented logic is deterministic and recorded in each report. For split x and sorted seed cohort $C = (42, 43, 44, 45, 46)$, the runner first checks that $50|C| \leq |\mathcal{D}_x|$. It then builds a split-specific random permutation π_x of the split indices using a seed sequence formed from the global selection seed 42 and a 32-bit salt derived from the SHA-256 hash of the split name. If active seed s has cohort position j, it receives $\pi_x[50j : 50(j+1)]$. Thus cohorts are non-overlapping within each split and the five seeds cover 250 samples from the N test split and 250 samples from the $N-1$ test split. The two split cohorts are selected independently rather than index-paired. Within a selected cohort, stochastic attack components are also deterministic: for sample index i, the attack seed is $s + i$. Per-generator active and reactive bounds are taken from the PGLIB case file (Babaeinejadsarookolaee et al., 2019) at scoring time; zero-$P_{\max}$ synchronous compensators are excluded from generator scoring.

Surrogate architecture. The surrogate is a graph attention network (Veličković et al., 2018) over the heterogeneous (`bus`, `ac_line`, `bus`) graph. It has 3 attention layers with hidden dimension 128 and dropout 0.2; the first layer uses 4 attention heads and the other two are single-headed. The convolution accepts an optional per-branch availability in $[0, 1]$, whose logarithm is added to the attention logit before the softmax. At availability 1 this is an exact no-op, so it is inactive during training and whenever EvA+ queries the network; EvA+ instead applies a candidate topology by removing the corresponding branches from the input graph. The gate is used only where a continuous relaxation of a branch removal is required, namely by the PRBCD and GRBCD baselines, described below.

Surrogate training. All case118 surrogates in the main RQ tables use the same training budget, taken from the checkpoint configuration dictionaries. Unless a row states otherwise, the backbone is GAT with the architecture above. Training uses the Adam optimiser with learning rate 10^{-3}, weight decay 10^{-5}, a cosine annealing schedule down to 10^{-5}, gradient clipping at 1.0, batch size 128, and up to 150 epochs with early stopping (patience 30, minimum improvement 10^{-5}); the seed is 42 and training uses the intact-grid data (case N). Generator and bus predictive terms carry equal weight. The physics-informed guide (`phys_cosine`), which PEGAS uses, adds the Kirchhoff residual of Supp. B.2 to that predictive loss with weight $\lambda = 0.05$, evaluated every fifth optimisation step so its cost stays small next to the supervised pass; the predictive guide (`mse_cosine`), which the surrogate-guided baselines use, is identical in every other respect but omits the residual.

Training reproducibility. The dataset split is fixed by OPFData: training uses `split=train`, validation uses `split=val`, and reporting uses `split=test` with and without $N-1$ perturbations. The training run fixes the Python, NumPy, PyTorch, and CUDA random seeds to 42, records the full configuration and software versions, and stores the checkpoint used for all reported evaluations. The training loader shuffles the training split with multi-worker loading, and we do not enable bitwise deterministic CUDA kernels. Therefore retraining the same recipe reproduces the protocol but can yield small numerical differences; evaluating the released checkpoints reproduces the tables.

Table 5: RQ1 attack configurations, read from the per-run reports. Tournament size is the only entry the reports do not serialize; it is the configuration default, which no launcher overrides. k is the perturbation budget; the load perturbation cap is $\varepsilon = 0.02$ pu, recorded per sample in the reports.

Attack	Configuration
PEGAS (EvA+)	pop. 1024, gen. 500, tourn. 2, k_{cross} 30, p_{mut} 0.01, patience 30, k=1, hybrid, ε=0.02; guide `phys_cosine`
EvA	as above, topology-only; guide `mse_cosine`
PRBCD / GRBCD	block 64, epochs 125, resampling 100, step 10^3, k=1; guide `mse_cosine`
PGD	L_0, node budget 1, 30 steps, 3 restarts, ε=0.02; guide `mse_cosine`
Exhaustive	all 175 single-branch removals; solver-scored

Attackers. Table 5 lists the attack configurations, all read from the per-run reports. Every attack in this main comparison uses a single-perturbation budget ($k = 1$) and evaluates the selected perturbation with the AC-OPF solver. EvA and EvA+ are evolutionary, gradient-free attacks: they evolve a population of candidate perturbations over generations, score each candidate by the surrogate's predicted constraint violation, and keep the fittest, so they need only forward queries of the surrogate. PEGAS runs EvA+ in hybrid mode (topology and load genes under one shared budget) guided by `phys_cosine`; our EvA baseline (Akhondzadeh et al., 2025) runs the same genetic search in topology-only mode guided by `mse_cosine`. PRBCD and GRBCD (Geisler et al., 2021) are gradient-based topology attacks. In our grid implementation they are branch-index multigraph adaptations: they optimize over existing AC-branch columns, preserve parallel-line identity, and commit removals rather than non-physical edge additions. PGD is a projected-gradient attack on the load features under an L_0 node budget (Madry et al., 2018). These three are guided by `mse_cosine`. Exhaustive enumerates all $\binom{175}{1} = 175$ single-branch removals and scores each with the solver, so it is model-agnostic; its reported fidelity is dispatched through `mse_cosine`. The two component ablations reuse the `phys_cosine` guide rather than a new checkpoint: *w/o load* reruns it with topology-only EvA+, and *w/o topology* reruns it in load-only mode, isolating the contribution of each gene type. The same budget k governs every mode, so the ablation rows are directly comparable: at k=1 the topology-only arm removes one branch, the load-only arm perturbs one load bus, and the hybrid arm spends its single perturbation on whichever gene type the search prefers. Load genes are restricted to the buses the surrogate is most sensitive to: a single gradient pass on the clean state ranks loads by the L_2 norm of a predictive-loss gradient at each load row, the $4k$ most sensitive form a candidate pool, and the search selects which k of that pool to perturb and by how much, additively, capped at ε and clipped to the load bounds.

Optional line-thermal fitness. The implementation includes an opt-in diagnostic only surrogate-derived line-thermal fitness term. It is left at its default weight 0 in the RQ1 attacker configurations above, so line constraints enter the static tables through the post-search power-flow projected evaluation, not through the search objective.

Unconstrained baseline diagnostics. The OPF-specific constraints that EvA+ adds are exposed as explicit switches, so the unconstrained behaviour of the published attacks can be reproduced from the same code path. `--original-implementation` restores the topology-only search over complete-graph flips with the MSE objective and no repair; it fails loudly unless the run is topology-only and uses that objective. `--skip-projection` disables static topology repair; in EvA+ feature/hybrid runs it also disables load-bound clipping during the search. `--skip-bounds` disables load clipping for the gradient feature attack. Every run records which switches were active, so a report states its own configuration. These are diagnostics for the baselines of RQ1;PEGAS never uses them.

Evaluation. The AC-OPF oracle is solved with PowerModels (Coffrin et al., 2018) and Ipopt (Wächter & Biegler, 2006) in Julia. The headline metric is the power-flow projected success rate, which counts a sample as broken when the post-attack state exceeds the sample's clean power-flow baseline on the generator, bus, or line checks. We also report power-flow projected robust physical accuracy and mean squared error as descriptive context; generator accuracy scores active power only, with reactive-power exceedances kept as a diagnostic. Every figure is the mean $\pm$ standard deviation over the five seeds.

C.6 RQ2: ATTACKS AGAINST GRID2OP DYNAMIC AGENTS

Environments. We use the Grid2Op environments (Donnot, 2020a) `l2rpn_idf_2023` and `l2rpn_wcci_2022` with the LightSim2Grid backend (Donnot, 2020b). For each of five seeds $\{42, \ldots, 46\}$ we evaluate 30 chronics selected by the baseline-random protocol used by our Grid2Op baseline scripts: . The same selected chronic IDs are used for every defender and opponent mode under a fixed environment and seed, so comparisons are paired within seed. Different seeds are sampled independently and may overlap. Each chronic receives the deterministic execution seed $(\text{seed} + \text{chronic_id}) \bmod (2^{31} - 1)$. The episode horizon is 2017 steps. All six defenders are available on the IDF environment; only the three transfer-compatible defenders (DoNothing, OptimCVXPY, Expert) run on WCCI.

Defenders. Each defender is a Grid2Op control agent that observes the grid at every step and may act to keep it within limits; we construct them with their library defaults. A defender's survival under an attacker measures how well it absorbs the attack.

- **DoNothing** is the null baseline: it never acts, so its survival reflects the grid's passive resilience to the opponent.
- **Solver** is a traditional AC-OPF baseline that computes a single re-dispatch action; it survives only 5.3% of episodes even without an opponent, so it is already failing before any attack.
- **OptimCVXPY** (RTE France, 2024c) is a convex-optimisation redispatch agent. It solves a small optimisation each time the maximum line loading ρ reaches 0.95, redispatching generation and curtailment to relieve the overload, and returns toward the reference dispatch once ρ falls to 0.85. Its defaults are: thermal-limit margin 0.9, power-flow error weight 0.5, unsafe penalties on redispatch, curtailment, and storage of 0.03, 0.1, and 0.3 respectively (all safe-state penalties 0), unit target weights 1.0, and rounding and sparsity margins of 0.01 and 0.005.
- **Expert** (Marot et al., 2018) is a rule-based expert system that clears line overloads by substation reconfiguration; it has no learned parameters and is instantiated for the IEEE-118 grid. Each step it ranks overloaded lines ($\rho \geq 1$) by loading, prioritising those whose overflow counter has reached the environment's allowance, and treats at most 3 overloads. For each it simulates candidate substation topologies, at most 25 overall and 5 per node, scoring each from 0 to 4 (4 clears all overloads, 3 clears the target overload, 2 partially clears it, 1 clears it but worsens others, 0 fails) and ranking them with the `MinMargin` reward $\min(1 - \rho)$. It accepts an action when the simulated state puts every ρ below the safe threshold 0.95. Its remaining defaults are a maximum of 2 unused lines, flow-direction reconsideration ratio 0.75, loop-retention ratio 0.25, minimum loop power 0.1, and line-report threshold 0.2.
- **LJN** is the reinforcement-learning topology agent that won the L2RPN 2023 competition (Pavão et al., 2025), used as its released pretrained policy. It intervenes when the maximum line loading exceeds $\rho_{\text{danger}} = 0.99$, evaluates substation reconfigurations by simulating candidates, and keeps one that holds the maximum loading below $\rho_{\text{safe}} = 0.9$; once the grid is safe it moves back toward the reference topology.
- **LJN-TopoNN** is the topology-network variant of that agent. It keeps the same $\rho_{\text{danger}} = 0.99$ and $\rho_{\text{safe}} = 0.9$ thresholds and adds a PPO policy over a Gym-wrapped box observation of line loadings and a precomputed global topology action space of 352 actions. Each step it ranks the top $\text{topk} = 20$ candidates (the class default is 10; our runs set 20) and

applies the best feasible one, falling back to an $N-1$ heuristic and then an optimisation-based recovery when no learned action is safe. Every episode report records the top-k, the action-space size, and the checkpoint hash.

Opponent modes. Each RQ2 row uses one of three opponent modes on a given environment. *No opponent* runs the defenders with no attacks and measures baseline survival. The *default* opponent is Grid2Op's native stochastic line-attack opponent, the standard adversary shipped with each L2RPN environment. PEGAS (next paragraph) replaces that stochastic opponent with our surrogate-guided line selection while keeping the same native budget and scheduling machinery. The default opponent differs by environment. On `l2rpn_idf_2023` it is a `GeometricOpponentMultiArea`, which partitions attackable lines into three areas and attacks each area independently: it samples timing geometrically (roughly one attack every 32 hours of simulated time, mean duration 2 hours, minimum 1 hour, generator $p_{\max}/p_{\min}$ ratio 4) from an initial budget of 1000 regenerating at 0.51 per step, with attack duration 96 steps and cooldown 0. On `l2rpn_wcci_2022` it is a single-area `GeometricOpponent` over 23 attackable lines (roughly one attack every 24 hours, mean duration 4 hours, minimum 1 hour) from a smaller initial budget of 144 regenerating at 0.17 per step, with attack duration 96 steps and cooldown 12.

Attacker. PEGAS acts as the environment's native opponent. Every setting below is recorded in the run plan written beside each campaign, which also stores the SHA-256 of the guide checkpoint. PEGAS is guided by `phys_cosine` and selects line disconnections with the genetic selector (`eva_ga`) in topology-only mode under budget $k = 1$, using the same `paper` preset as RQ1: population 1024, 500 generations, $k_{\text{cross}} = 30$, $p_{\text{mut}} = 0.01$, and early stopping after 30 stalled generations. Grid2Op exposes only line-status actions to the opponent, so PEGAS here runs topology-only EvA+ with no load genes, rather than the hybrid RQ1 variant. The final line is chosen by `argmax` over the top 5 scored candidates, with selection temperature 1.0, selection $\epsilon = 0.1$ and policy seed 0; the surrogate-derived thermal term is disabled (weight 0, `topk_mean` form, derived-bus-voltage source), so the score is the constraint-violation objective alone. Of the 175 canonical branches, 168 map onto distinct Grid2Op lines and form the candidate set at every attack, which the per-attack candidate counts in each episode report confirm; the remaining seven are parallel branches sharing endpoints with a mapped branch and are not separately targetable, and transformer branches are never attacked. The `selected_disconnecting` field is diagnostic telemetry computed from the fixed AC-line template; it is not a live RQ2 action filter and does not account for transformer paths.

Opponent schedule. The opponent operates under Grid2Op's native budget schedule: initial budget 1000, regeneration 0.51 per step, attack duration 96 steps, and cooldown 0. The sensitivity study varies one axis at a time from this default: initial budget in $\{500, 2000\}$, attack duration in $\{48, 192\}$, cooldown in $\{10, 100\}$, and candidate line masking that removes line 26, or lines 70 and 26, from the attacker's reachable set.

Evaluation. The metric is the defender survival rate, computed per seed as the fraction of the 30 chronics the defender completes without a blackout, then averaged as the mean $\pm$ standard deviation over the five seeds. Lower survival indicates a stronger attack.

C.7 Compute

Experiments ran on three remote servers. Two are identical: each has two Intel Xeon Silver 4416+ CPUs (40 physical cores, 80 threads), four NVIDIA L40S GPUs (46 GB each), and 503 GB of RAM. The third has two Intel Xeon Gold 5222 CPUs at 3.80 GHz (8 physical cores, 16 threads), three NVIDIA RTX A6000 GPUs (48 GB each), and 503 GB of RAM. Jobs ran in parallel across GPU slots or separately, according to the task: CPU-bound work such as the attack search used a single GPU with many parallel worker jobs, whereas GPU-bound work such as surrogate training used a full GPU. Surrogates and attacks are implemented in PyTorch (Paszke et al., 2019) and PyTorch Geometric (Fey & Lenssen, 2019). Static attacks solve AC-OPF with PowerModels and Ipopt in Julia; Grid2Op uses the LightSim2Grid backend.

D SUPPLEMENTARY D: SUPPLEMENTARY RESULTS

D.1 RQ1: SUCCESS RATE AGAINST STATIC PNO

Reporting conventions. Unless stated otherwise, every RQ1 table in this appendix is computed on the 118-bus grid with a single-perturbation budget (k=1). Each attack is scored on its surrogate dispatch after AC power-flow projection (Supp. C.4). We report the relative prediction accuracy (RPA) of generators, buses and lines, the generator and bus MSE, and the attack success rate (SR, `success_rate_v2_pf`) in %; a higher SR indicates a stronger attack. Values are mean ± standard deviation over five seeds. *Grid* N denotes the intact grid and *Grid* $N-1$ the grid under a single contingency.

Main comparison. Table 6 compares all attacks on both grids. Exhaustive, which scores every single-line removal by power flow, is the brute-force upper reference and succeeds on every state. Among the search-based attacks, PEGAS reaches 93.2% on Grid N and 86.3% on Grid $N-1$, ahead of EvA (85.2% and 81.5%), while the gradient baselines PRBCD, GRBCD and PGD stay at or below 16.0%.

Table 6: Attack performance on the 118-bus grid (k=1, five seeds).

	RPA			MSE		
Attack	Gen	Bus	Line	Gen	Bus	**SR (%)** ↑
Grid N						
Exhaustive	0.789 ± 0.000	1.000 ± 0.000	0.971 ± 0.001	0.2356 ± 0.0066	0.0272 ± 0.0007	100.0 ± 0.0
PEGAS	0.742 ± 0.001	1.000 ± 0.000	0.971 ± 0.001	0.2867 ± 0.0110	0.0330 ± 0.0017	93.2 ± 3.0
EvA	0.786 ± 0.001	1.000 ± 0.000	0.977 ± 0.001	0.2349 ± 0.0058	0.0268 ± 0.0007	85.2 ± 5.9
PRBCD	0.789 ± 0.001	1.000 ± 0.000	0.989 ± 0.001	0.1542 ± 0.0042	0.0047 ± 0.0002	13.2 ± 3.6
GRBCD	0.790 ± 0.000	1.000 ± 0.000	0.989 ± 0.001	0.1534 ± 0.0041	0.0047 ± 0.0002	16.0 ± 5.8
PGD	0.789 ± 0.000	1.000 ± 0.000	0.990 ± 0.001	0.1544 ± 0.0040	0.0046 ± 0.0002	0.8 ± 1.1
Grid $N-1$						
Exhaustive	0.788 ± 0.003	0.999 ± 0.000	0.968 ± 0.002	0.2729 ± 0.0094	0.0328 ± 0.0046	100.0 ± 0.0
PEGAS	0.744 ± 0.003	1.000 ± 0.000	0.970 ± 0.001	0.2942 ± 0.0233	0.0318 ± 0.0027	86.3 ± 5.4
EvA	0.787 ± 0.003	1.000 ± 0.000	0.976 ± 0.001	0.2417 ± 0.0092	0.0264 ± 0.0014	81.5 ± 4.9
PRBCD	0.788 ± 0.002	1.000 ± 0.000	0.987 ± 0.001	0.1814 ± 0.0121	0.0068 ± 0.0011	12.9 ± 3.1
GRBCD	0.788 ± 0.003	1.000 ± 0.000	0.987 ± 0.001	0.1810 ± 0.0142	0.0069 ± 0.0011	14.9 ± 3.4
PGD	0.789 ± 0.003	1.000 ± 0.000	0.988 ± 0.001	0.1793 ± 0.0118	0.0067 ± 0.0011	0.4 ± 0.9

D.2 ORIGINAL IMPLEMENTATIONS OF THE BASELINES

Table 7 evaluates the baselines as designed in their original papers, as a single-seed diagnostic (seed 42, 50 samples per grid). These runs use the plain `mse_cosine` guide and the MSE objective, with EvA as published, PRBCD/GRBCD without static feasibility projection, and PGD without OPF load lower/upper-bound clipping. PF reporting remains enabled, and the table keeps the same PF-projected columns as Table 6 so that it reads as a direct comparison. Invalid or unusable actions remain in the denominator and count as failures. The *PF conv.* column gives the number of samples whose projected power flow converged; EvA has no converged sample on either grid, so its metrics are undefined (–). Under these original settings every baseline fails on every Grid N state, and only PGD succeeds at all on Grid $N-1$ (6.0%).

Table 7: Baselines in their original implementations (seed 42, 50 samples per grid).

		RPA			MSE		
Attack	**PF conv.**	Gen	Bus	Line	Gen	Bus	**SR (%)** ↑
Grid N							
EvA	0/50	–	–	–	–	–	0.0
PRBCD	24/50	0.792	1.000	0.990	0.1487	0.0041	0.0
GRBCD	1/50	0.789	1.000	0.984	0.1421	0.0049	0.0
PGD	50/50	0.791	1.000	0.990	0.1467	0.0040	0.0
Grid $N-1$							
EvA	0/50	–	–	–	–	–	0.0
PRBCD	23/50	0.785	1.000	0.991	0.1630	0.0050	0.0
GRBCD	3/50	0.803	1.000	0.989	0.1948	0.0065	0.0
PGD	50/50	0.786	1.000	0.987	0.1971	0.0123	6.0

D.3 ABLATION OF COMPONENTS

Table 8 isolates the contribution of each PEGAS component on Grid N. Each row removes a single factor from the PEGAS recipe. *w/o load* removes the load genes, giving the topology-only PEGAS that is used against the dynamic agents in RQ2; *w/o topology* removes the topology genes, giving a load-only attack. The remaining rows remove a single surrogate factor (the GAT backbone, the physics-informed loss, or the MSE term). All rows share the same budget (k=1), so the comparison is budget-matched. Removing the topology genes or the MSE term is the most damaging (SR falls to 6.8% and 0.0%), while removing the load genes costs only 1.6 points.

Table 8: Ablation of PEGAS components on Grid N (k=1, five seeds).

	RPA			MSE		
Variant	Gen	Bus	Line	Gen	Bus	**SR (%)** ↑
PEGAS	0.742 ±0.001	1.000 ±0.000	0.971 ±0.001	0.2867 ±0.0110	0.0330 ±0.0017	93.2 ±3.0
w/o load	0.745 ±0.003	1.000 ±0.000	0.971 ±0.001	0.2827 ±0.0082	0.0319 ±0.0016	91.6 ±4.3
w/o topology	0.791 ±0.002	1.000 ±0.000	0.987 ±0.000	0.1550 ±0.0056	0.0036 ±0.0003	6.8 ±2.3
w/o GAT	0.609 ±0.004	1.000 ±0.000	0.989 ±0.000	0.2337 ±0.0021	0.0032 ±0.0001	37.2 ±4.1
w/o PINN	0.857 ±0.004	1.000 ±0.000	0.972 ±0.002	0.2596 ±0.0149	0.0328 ±0.0032	72.4 ±6.8
w/o MSE	0.737 ±0.000	1.000 ±0.000	0.966 ±0.001	2.2316 ±0.0229	0.0620 ±0.0013	0.0 ±0.0

D.4 ATTACK-BUDGET SCALABILITY

Table 9 sweeps the attack budget $k \in \{1, 2, 3\}$ on the 118-bus grid as a single-seed study (seed 42, 50 states per split). Rows are compared against one another across k to show how each attack converts additional budget into damage; they are not a restatement of Table 6, which is a five-seed comparison at k=1. The **mean** column averages the three budgets and summarizes an attack's typical success across the k=1, 2, 3 regime.

Two blocks are reported because the baselines were run under both attack objectives: the supervised MSE objective of the original implementations, and the constraint-violation objective used throughout this work. PEGAS optimizes the violation objective in both blocks – the blocks vary the *baselines'* objective, not ours – so its row (marked †) is repeated as the common reference and each block reads as PEGAS against the baselines under that block's objective.

PEGAS is the only attack that saturates: it reaches 100% at k=2 on Grid N and gains nothing from k=3, so a single additional line is already sufficient on essentially every state. The gradient baselines scale roughly linearly and remain far below saturation at k=3. PGD is the outlier: its budget is spent on load perturbations bounded by ε=0.02 pu, and additional budget buys almost nothing, which is consistent with the topology genes dominating the ablation in Table 8.

Table 9: Success rate (%) versus attack budget k on the 118-bus grid (seed 42, 50 states per split). †Repeated reference row.

	Grid N				**Grid $N-1$**			
Attack	k=1	k=2	k=3	**mean**	k=1	k=2	k=3	**mean**
Baseline objective: supervised MSE								
PEGAS†	94.0	100.0	100.0	**98.0**	84.0	94.0	94.0	**90.7**
EvA	70.0	84.0	98.0	84.0	80.0	88.0	92.0	86.7
GRBCD	22.0	44.0	52.0	39.3	10.0	38.0	42.0	30.0
PRBCD	14.0	44.0	54.0	37.3	8.0	42.0	56.0	35.3
PGD	0.0	0.0	0.0	0.0	6.0	8.0	8.0	7.3
Baseline objective: constraint violation								
PEGAS†	94.0	100.0	100.0	**98.0**	84.0	94.0	94.0	**90.7**
EvA	84.0	86.0	96.0	88.7	80.0	88.0	96.0	88.0
GRBCD	22.0	38.0	58.0	39.3	10.0	48.0	70.0	42.7
PRBCD	8.0	34.0	50.0	30.7	2.0	26.0	36.0	21.3
PGD	2.0	2.0	8.0	4.0	4.0	4.0	4.0	4.0

D.5 GENERALIZATION STUDIES

Larger grids. Table 10 applies PEGAS to a different static solver grid, the 2000-bus system, and compares it with the 118-bus results on Grid N (five seeds, k=1). Success decreases from 93.2% to 71.6% on the larger grid.

Table 10: Generalization of PEGAS to a larger static grid (Grid N, five seeds).

	RPA			**MSE**		
Grid	Gen	Bus	Line	Gen	Bus	**SR (%)** ↑
118-bus	0.742 ±0.001	1.000 ±0.000	0.971 ±0.001	0.2867 ±0.0110	0.0330 ±0.0017	93.2 ±3.0
2000-bus	0.550 ±0.001	0.998 ±0.000	0.993 ±0.000	1.1007 ±0.0179	0.3257 ±0.0123	71.6 ±5.9

Surrogate backbone versus search strategy. Table 11 crosses two surrogate guides with two search strategies on the 118-bus grid. EvA perturbs the topology only, whereas EvA+ is the hybrid load-and-topology search; the PI-GAT/EvA+ cell is therefore PEGAS. *GCN* denotes a GCN backbone trained with the MSE + cosine learning-rate recipe. Replacing the PI-GAT guide with the GCN guide lowers success from 91.6–93.2% to 19.2–25.6%, whereas switching the search strategy changes it by at most 6.4 points.

Table 11: Surrogate guide versus search strategy (118-bus, five seeds).

		RPA			
Guide	**Search**	Gen	Bus	Line	**SR (%)** ↑
PI-GAT	EvA	0.745 ± 0.003	1.000 ± 0.000	0.971 ± 0.001	91.6 ± 4.3
PI-GAT	EvA+	0.742 ± 0.001	1.000 ± 0.000	0.971 ± 0.001	93.2 ± 3.0
GCN	EvA	0.737 ± 0.000	1.000 ± 0.000	0.993 ± 0.000	19.2 ± 1.8
GCN	EvA+	0.737 ± 0.000	1.000 ± 0.000	0.992 ± 0.000	25.6 ± 3.6

Table 12: Defender survival rate (%, ↓) by attacker and defender on IDF 2023 (five seeds).

Attacker	Nothing	Optim	Expert	LJN	TopoNN	Solver	Mean
Nothing	30.7 ±9.5	80.0 ±6.2	38.7 ±4.5	90.0 ±5.3	86.7 ±3.3	5.3 ±4.5	55.2 ±3.4
Default	22.0 ±11.5	46.7 ±10.3	27.3 ±10.4	72.0 ±7.7	68.7 ±11.7	5.3 ±3.8	40.3 ±5.5
PEGAS	0.0 ±0.0	3.3 ±3.3	0.7 ±1.5	5.3 ±3.0	6.0 ±2.8	0.0 ±0.0	2.6 ±0.9

D.6 RQ2: Success rate against agentic PNO

Reporting conventions. All RQ2 tables report the defender survival rate (%), i.e. the fraction of episodes that the defender completes; **a lower survival indicates a stronger attack**. Values are mean ± standard deviation over five seeds of 30 chronics each. The **Mean** column is computed within each seed over the displayed defender columns and then averaged over seeds; which defenders it includes is stated for each table.

Full survival matrix. Table 12 gives the complete IDF 2023 survival matrix summarized in the main text. Its **Mean** column averages all six defenders, including Solver. PEGAS lowers the mean survival from 55.2% without an opponent and 40.3% under the default opponent to 2.6%.

PEGAS trajectory outcomes. For the five defender conditions used in the masking analysis (excluding Solver), five seeds of 30 chronics give 750 PEGAS episodes: 23 survive and 727 collapse. The opponent records 1,799 line choices: 613 for line 26 and 1,186 for line 70. The 613 line 26 choices occur in distinct episodes, and every one of those episodes ends on that attack step with logged power-flow divergence (84.3% of collapses). Line 26 is a bridge in the nominal IDF grid, and RQ2 has no connectivity veto. Of the other 114 collapsed episodes, 87 have a preceding line 70 choice and 27 end before any PEGAS attack.

D.7 Candidate access versus line selection

PEGAS attacks on the native schedule but chooses among the 168 lines its surrogate represents, while the default opponent draws from 22. Its lower survival could therefore come from access to more lines rather than from how it selects among them. We equalize the candidate sets in both directions with two additional arms, run on the same five seeds and 30 chronics per seed as Table 12, and report them in Table 13 next to the Default and PEGAS rows of that table (the 22-line default and 168-line PEGAS rows are repeated from it). (i) *PEGAS restricted to the default pool*: its other 147 candidates are removed before search, which leaves 21 lines, because line 180 of the default pool is a transformer outside PEGAS's candidate set. (ii) *Default opponent with PEGAS's candidates*: the same 147 lines are added to the pools of the areas that own them (53, 53 and 63 lines), giving 169 lines, and every other opponent setting is unchanged.

Every arm keeps the native attack schedule, duration and budget; the rows differ only in the lines the attacker may cut (**Lines**) and in how it chooses among them. The default opponent is Grid2Op's load-weighted draw within each area. All rows use the same (seed, chronic) episodes, 150 per cell except Solver (149), and **Mean** averages all six defenders. The upper block of Table 13 reports survival; the lower block reports the mean number of attacks per episode, which also reflects episode length, since shorter episodes receive fewer attacks.

Given PEGAS's candidates, the default opponent becomes considerably stronger, but PEGAS remains far stronger: mean survival is 21.4% against the expanded default opponent, compared with 40.3% against the native one and 2.6% against PEGAS, and PEGAS gives the lower survival for every defender. Restricted to the default pool, PEGAS is not stronger than the default opponent: mean survival is 45.7% against 40.3%. It leaves fewer DoNothing runs alive (3.3% against 22.0%) but more for Optim, Expert, LJN and TopoNN. The five-defender mean of this arm (Solver excluded), 54.2%, is close to the 52.3% obtained when only line 26 is masked (Table 15). No bridge of the intact nominal IDF grid, line 26 included, belongs to the default pool, and most collapses under PEGAS follow a cut of line 26 (§D.6). PEGAS therefore needs both access to lines outside the default pool and its guided choice among them.

Table 13: Candidate access versus line selection on IDF 2023: survival rate (%, ↓, five seeds) and mean attacks per episode.

Attacker	Lines	Nothing	Optim	Expert	LJN	TopoNN	Solver	Mean
Survival rate (%)								
Default	22	22.0 ±11.5	46.7 ±10.3	27.3 ±10.4	72.0 ±7.7	68.7 ±11.7	5.3 ±3.8	40.3 ±5.5
PEGAS	21	3.3 ±2.4	75.8 ±11.2	31.3 ±8.7	81.3 ±3.8	79.3 ±5.5	3.3 ±4.1	45.7 ±2.8
Default	169	21.3 ±8.7	26.7 ±9.1	13.3 ±4.1	35.3 ±9.0	30.0 ±7.1	2.0 ±1.8	21.4 ±4.0
PEGAS	168	0.0 ±0.0	3.3 ±3.3	0.7 ±1.5	5.3 ±3.0	6.0 ±2.8	0.0 ±0.0	2.6 ±0.9
Attacks per episode								
Default	22	1.79	9.14	7.35	13.06	12.71	1.51	
PEGAS	21	1.69	10.53	6.54	12.49	12.45	1.64	
Default	169	1.29	7.41	5.68	9.30	9.09	1.23	
PEGAS	168	1.31	2.38	2.22	3.05	3.03	1.19	

The expanded default opponent differs from PEGAS in more than its choice of line. Grid2Op cancels a scheduled attack whenever any line of the area pool is out of service, and with 53 to 63 lines per area this happens more often; PEGAS keeps the native abort rule, which watches only the 22 default lines. The expanded opponent therefore attacks fewer times per episode than the native one, which reflects both the cancelled attacks and its shorter episodes. The effect is largest for DoNothing, which never reconnects a line: once a line of an area is out of service, the enlarged pool blocks every later attack in that area. Against DoNothing the expanded opponent averages 1.29 attacks per episode, against 1.79 for the native one, and gains almost nothing from the wider pool (21.3% against 22.0%). Against Optim, Expert, LJN and TopoNN, which reconnect lines, the wider pool lowers survival from 46.7, 27.3, 72.0 and 68.7% to 26.7, 13.3, 35.3 and 30.0%, still well above PEGAS's 3.3, 0.7, 5.3 and 6.0%.

D.8 ATTACK SEARCH OVERHEAD

Table 14 and Figure 9 report the per-sample core-search wall time on case118 ($k = 1$, five test states, five seeds, one GPU). Every attack is timed the same way — a `perf_counter` span around the core search, excluding one-time model and dataset loading — so the four modes are directly comparable. The cost is set by the attack *mode*, not by the surrogate: the `mse_cosine` and `phys_cosine` guides share the same GAT architecture and matched within noise (per-generation cost ratio 0.96–1.00, identical for topology-only), so a physics-informed guide is free at inference; we report the `mse_cosine` numbers. Extending the topology-only search of the original EvA to the hybrid load–topology space of EvA+ adds only $1.5\times$ ($4.2 \rightarrow 6.3$ s), since both search under the same small budget (k=1) with a batched surrogate. The EvA+ load-only attack is an order of magnitude more expensive (63 s, $14.9\times$), as its continuous perturbations demand far more surrogate evaluations; a one-time CUDA/JIT warmup on the first evaluated state dominates its standard deviation. Exhaustive brute force, the model-agnostic reference, scores every one of the 175 single-line removals by power flow, costing 57.8 s per state ($13.7\times$ the topology search); its scan is deterministic (tiny standard deviation) and, unlike the guided search, grows combinatorially with the budget (a k=2 sweep is $\sim$15,000 candidates per state).

Table 14: Per-sample core-search wall time on case118 (k=1; lower is cheaper).

Attack mode	Time (s)	vs. EvA
EvA (topology-only)	4.23 ± 0.65	$1.0\times$
EvA+ (hybrid)	6.34 ± 0.96	$1.5\times$
EvA+ (load-only)	63.0 ± 52.9	$14.9\times$
Exhaustive (all k=1 removals)	57.8 ± 0.7	$13.7\times$

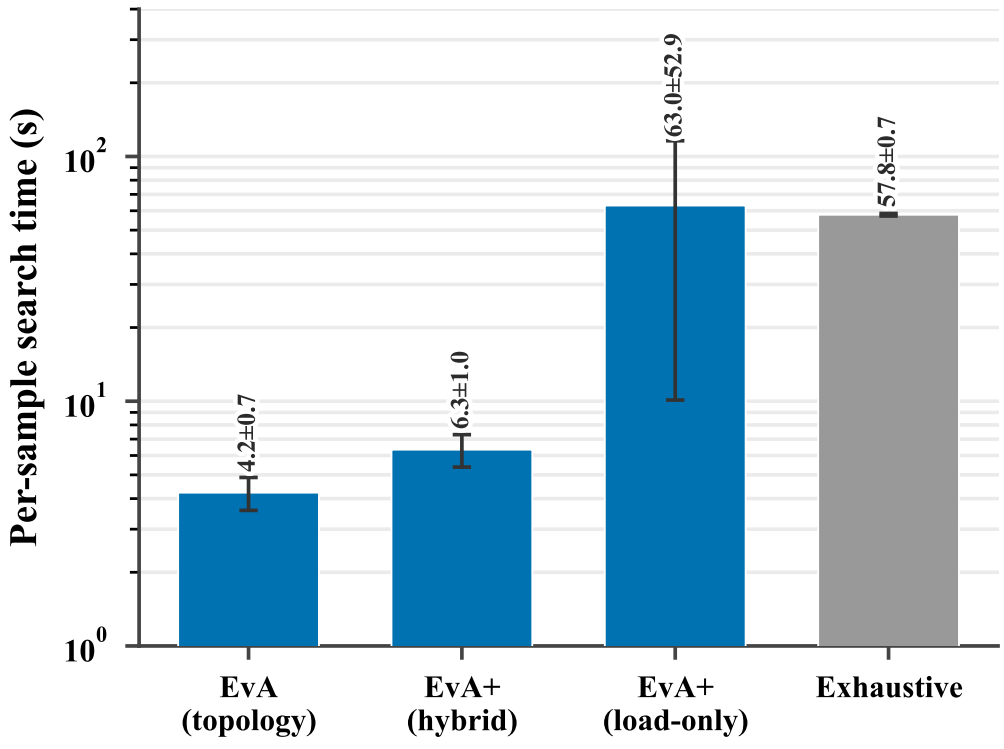


Figure 9: Per-sample search time by attack mode (case118, k=1, log scale, error bars = std). Exhaustive (grey) is the brute-force reference.

Table 15: Sensitivity of PEGAS to opponent settings on IDF 2023: survival rate (%, ↓, five seeds); Mean excludes Solver.

Attacker	Nothing	Optim	Expert	LJN	TopoNN	Mean
PEGAS (reference)	0.0 ±0.0	3.3 ±3.3	0.7 ±1.5	5.3 ±3.0	6.0 ±2.8	3.1 ±1.1
Line masking						
+ mask line 26	2.0 ±1.8	78.7 ±8.7	23.3 ±5.8	78.7 ±3.8	78.7 ±8.7	52.3 ±3.5
+ mask lines {70, 26}	9.3 ±4.3	78.0 ±6.9	38.7 ±5.1	90.0 ±7.1	88.7 ±4.5	60.9 ±3.5
Cooldown (default 0)						
+ 10 steps	0.0 ±0.0	13.3 ±7.5	7.3 ±2.8	4.7 ±3.8	6.0 ±3.7	6.3 ±2.9
+ 100 steps	3.3 ±4.1	44.0 ±13.0	26.7 ±3.3	45.3 ±7.3	42.7 ±10.4	32.4 ±5.4
Budget (default 1000)						
+ budget ×2	0.0 ±0.0	3.3 ±3.3	0.7 ±1.5	5.3 ±3.0	6.0 ±2.8	3.1 ±1.1
+ budget /2	0.0 ±0.0	3.3 ±3.3	0.7 ±1.5	5.3 ±3.0	6.0 ±2.8	3.1 ±1.1
Attack duration (default 96 steps)						
+ duration /2	0.0 ±0.0	3.3 ±3.3	0.7 ±1.5	5.3 ±3.0	6.0 ±2.8	3.1 ±1.1
+ duration ×2	0.0 ±0.0	3.3 ±3.3	0.7 ±1.5	5.3 ±3.0	6.0 ±2.8	3.1 ±1.1

Table 16: Defender survival rate (%, ↓) on WCCI 2022 (five seeds).

Attacker	Nothing	Optim	Expert	Mean
Nothing	4.7 ±3.0	77.3 ±10.9	14.0 ±6.4	32.0 ±5.6
Default	2.7 ±1.5	48.0 ±6.9	9.3 ±4.3	20.0 ±3.1
PEGAS	1.3 ±1.8	32.7 ±6.4	8.7 ±5.6	14.2 ±3.2

D.9 Sensitivity analysis of PEGAS

Opponent settings. Table 15 varies one opponent setting at a time on IDF 2023, starting from the PEGAS configuration of Table 12. Here **Mean** averages the five displayed defenders and excludes Solver, which is why the PEGAS reference reads 3.1% rather than the 2.6% of Table 12. The four blocks are:

- *Line masking*: the listed lines are removed from the attacker's candidate set, so they can no longer be selected for disconnection. Masking line 26 alone raises mean survival from 3.1% to 52.3%.
- *Cooldown*: the minimum number of steps between consecutive attacks (default 0). Unlike the budget and duration, the cooldown binds directly: limiting how often the attacker may strike is what degrades it.
- *Budget*: the opponent's initial attack budget (default 1000). Halving or doubling it leaves every cell unchanged: episodes end long before the budget is exhausted, so this constraint never binds.
- *Attack duration*: how long a disconnected line stays out (default 96 steps). As with the budget, the setting does not bind: the grid fails within the first outage window, so extending or shortening it changes nothing.

Transfer to WCCI 2022. Table 16 repeats the attacker comparison on the WCCI 2022 benchmark. Only the three transfer-compatible defenders (Nothing, Optim and Expert) are available on this environment, and **Mean** averages these three. PEGAS again gives the lowest mean survival (14.2%, against 20.0% for the default opponent and 32.0% without an opponent).

Table 17: Paired survival difference (pp, arm − PEGAS) on IDF 2023. *: $p < 0.05$, exact McNemar with Holm correction.

Attacker	Nothing	Optim	Expert	LJN	TopoNN	Solver
Nothing	$+30.7^*$	$+76.7^*$	$+38.0^*$	$+84.7^*$	$+80.7^*$	$+5.4^*$
Default	$+22.0^*$	$+43.3^*$	$+26.7^*$	$+66.7^*$	$+62.7^*$	$+5.4^*$

Table 18: Paired survival difference (pp, arm − PEGAS) on WCCI 2022. *: $p < 0.05$, exact McNemar with Holm correction.

Attacker	Nothing	Optim	Expert
Nothing	$+3.3$	$+44.7^*$	$+5.3^*$
Default	$+1.3$	$+15.3^*$	$+0.7$

D.10 Statistical significance of the RQ2 comparisons

The survival rates reported above are proportions of episodes, and every attacker is evaluated on the *same* episodes: for a fixed environment, seed and defender the selected chronic IDs are identical across opponent modes, so each attacker pair is matched episode by episode. We therefore treat the outcome as a paired binary variable and use an **exact McNemar test** on the discordant pairs, the standard choice when two methods are compared on one shared evaluation set, as recommended by Dietterich (1998). Pairing is on (seed, chronic) rather than chronic alone: seeds draw their 30 chronics independently and a chronic that recurs under a second seed is a different episode, with its own execution seed.

PEGAS is the control and every other attacker is compared against it, which gives $k-1$ comparisons per defender instead of all $k(k-1)/2$ pairs. The resulting p-values are corrected within each defender by the **Holm** step-down procedure, following Demšar (2006), who recommends comparing a proposed method against the alternatives rather than exhaustively, and prefers Holm to Bonferroni–Dunn on power grounds.

We report Wilcoxon signed-rank tests over the five per-seed survival rates only as a secondary check in the released artifacts, never as the primary evidence. With $n = 5$ paired observations the smallest attainable two-sided p is $2/2^5 = 0.0625$, so a seed-level Wilcoxon test cannot reach $\alpha = 0.05$ for *any* effect size, however large.

Reading the significance tables. Tables 17 and 18 are the companions of Tables 12 and 16. Each cell is the difference in defender survival, in percentage points, of the listed attacker minus PEGAS, paired by (seed, chronic) within a defender. Since survival is the defender's metric, a positive value means the attacker left more defenders alive and is therefore weaker than PEGAS. A * marks $p < 0.05$ after Holm correction. Each column pairs 150 (seed, chronic) units, except Solver on IDF 2023 (149). On IDF 2023, every comparison is significant. On WCCI 2022, three comparisons do not reach significance: attacker Nothing against defender Nothing ($p_{\mathrm{Holm}} = 0.125$), attacker Default against defender Nothing ($p_{\mathrm{Holm}} = 0.500$), and attacker Default against defender Expert ($p_{\mathrm{Holm}} = 1.000$).

D.11 USING THE RELEASED CODE PACKAGE

The anonymized code supplement is an executable companion to the paper. It contains the source code for surrogate training, static AC-OPF attacks, Grid2Op opponent evaluation, PF-based reporting, significance tests, and figure/table generation. It does not include trained checkpoints, raw run outputs, logs, figures, or paper source files; these are regenerated by the package commands. The package README gives the full command list. This section records the main path needed to recover the RQ1 and RQ2 results.

Setup. First create the environment and fetch the datasets and solver assets:

```
python -m venv .venv
source .venv/bin/activate
bash scripts/setup/environment.sh all
```

The setup command installs the Python requirements, downloads the PGLIB and OPFData assets used by RQ1, downloads the Grid2Op environments used by RQ2, and prepares the Julia/PowerModels stack used for AC power-flow reporting. The default examples use `GPUS=0`; users with more devices can override `GPUS` and `JOBS_PER_GPU` without changing the experiment definitions.

Third-party defenders. The `l2rpn-baselines` package is installed from PyPI and is not vendored in the supplement. `OptimCVXPYAgent` and `ExpertAgent` are imported from that installed package. Before importing `OptimCVXPYAgent`, the runtime shim in `run_grid2op_attack_eval.py` handles the `lightsim2grid` and `cvxpy` API changes required by the upstream agent; `ExpertAgent` imports unpatched.

The LJN defender source is included with the compatibility patch used in our runs, but not the upstream binary archives it loads at runtime. Because `LJNAgent` builds several topology-search modules from separate action archives, copy the upstream `models/` and `assets/` directories whole:

```
LJN=docs/research/task0/agents/l2rpn-2023-ljn-agent
git clone https://github.com/lajavaness/l2rpn-2023-ljn-agent.git /tmp/ljn
git -C /tmp/ljn checkout ca0637eab9f098be7f206ed0e46a3900cd4deec0
cp -r /tmp/ljn/models /tmp/ljn/assets "$LJN/"
```

Without those LJN assets, the non-LJN defenders still run. In that case restrict the defender list, for example `DEFENDERS=DoNothingAgent,OptimCVXPYAgent,ExpertAgent`.

Main RQ1 static results. The main static table uses the case118 guides trained with the physics-informed and MSE recipes. Train the two guide families first:

```
ARMS="base mse_cosine pinn_cosine gcn" \
  bash scripts/train/surrogate_ablations.sh

RECIPE_FAMILY=mse_cosine ARMS="base gcn" \
  bash scripts/train/surrogate_ablations.sh
```

Static campaigns use the OPFData test split, both Grid N and Grid $N-1$, and the disjoint-seeded cohorts used in the paper. Set the common options once:

```
export SEEDS=42,43,44,45,46
export SAMPLE_SELECTION=disjoint_seeded
export SAMPLE_SELECTION_SEED=42
export SAMPLE_SEEDS=42,43,44,45,46
export MAX_SAMPLES=50
export SPLIT=both
export BUDGET_K=1
export ATTACK_LOSS_MODES=violation
```

```
export GPUS=0
export JOBS_PER_GPU=1
```

Then run the four campaign families behind the main RQ1 attack comparison:

```
CHECKPOINTS=phys_cosine \
STATIC_RUN_GROUP=surrogate-seed-consistency \
CAMPAIGN_TAG=rq1_pegas_phys_cosine_hybrid_k1_s50_5seed \
PHASES=eva_strong \
EVA_ATTACK_MODE=hybrid_unified \
bash scripts/attack/pegas_sweep.sh

CHECKPOINTS=mse_cosine \
STATIC_RUN_GROUP=surrogate-seed-consistency \
CAMPAIGN_TAG=rq1_eva_mse_cosine_topology_k1_s50_5seed \
PHASES=eva_strong \
EVA_ATTACK_MODE=topology \
bash scripts/attack/pegas_sweep.sh

CHECKPOINTS=mse_cosine \
STATIC_RUN_GROUP=surrogate-seed-consistency \
CAMPAIGN_TAG=rq1_mse_cosine_topology_feature_k1_s50_5seed \
PHASES=topology,feature_l0 \
TOPOLOGY_ATTACK=both \
FEATURE_ATTACK=pgd \
bash scripts/attack/pegas_sweep.sh

CHECKPOINTS=mse_cosine \
STATIC_RUN_GROUP=surrogate-seed-consistency \
CAMPAIGN_TAG=rq1_exhaustive_k1_s50_5seed \
PHASES=brute_force \
BRUTE_FORCE_MODE=exhaustive \
BRUTE_FORCE_SCOPE=shared \
BRUTE_FORCE_CHECKPOINT=mse_cosine \
bash scripts/attack/pegas_sweep.sh
```

The corresponding PF-projected summaries are generated from the run artifacts, not from hardcoded table entries. For example, the PEGAS row is summarized by:

```
python docs/research/task1/analyze_static_seed_sweep.py \
  --run-base outputs/runs/static-attacks/case118/surrogate-seed-consistency \
  --campaign-tag rq1_pegas_phys_cosine_hybrid_k1_s50_5seed \
  --checkpoints phys_cosine \
  --phases eva_strong \
  --seeds 42,43,44,45,46 \
  --attack-loss-mode violation \
  --budget-k 1 \
  --require-complete \
  --require-disjoint-samples
```

Use the same analyzer pattern for the EvA, gradient-baseline, and exhaustive campaign tags above. The result figure and its CSV/Markdown sidecar table are then produced with `make_paper_figures.py`, passing the four summary JSON files listed in the README.

The analyzer reports the same columns used in the paper: PF-projected generator RPA, bus RPA, line RPA, generator MSE, bus MSE, and `success_rate_v2_pf`. The Appendix-D ablation, scalability, overhead, and case2000 studies use the same runners with the campaign tags listed in the README.

Main RQ2 Grid2Op results. RQ2 reports defender survival. Lower survival means the attack is stronger. Set the common five-seed configuration:

```
export SEEDS=42,43,44,45,46
export N_CHRONICS=30
export GPUS=0
export JOBS_PER_GPU=1
export B=outputs/runs/grid2op/l2rpn_idf_2023
export CKPT=outputs/checkpoints/phys_cosine/best_model.pt
```

Run the no-opponent and default baselines and PEGAS. The PEGAS runs use `eva_native.sh`, whose default defender set includes TopoNN:

```
RUN_PREFIX=$B/baselines/master_idf \
MODES=no_opponent,default \
bash scripts/grid2op/baseline_sweep.sh

RUN_ROOT=$B/surrogate-oracle-arms/master_idf_phys_cosine \
CHECKPOINT_PATH=$CKPT bash scripts/grid2op/eva_native.sh
```

The SolverAgent column and TopoNN's no-opponent and default rows are separate campaigns:

```
O=$B/opponent-baselines
DEFENDERS=SolverAgent MODES=no_opponent,default \
RUN_PREFIX=$O/master_idf_solveragent_physcosine \
bash scripts/grid2op/baseline_sweep.sh

E=$B/native-eva-defender-eval
DEFENDERS=SolverAgent CHECKPOINT_PATH=$CKPT \
RUN_ROOT=$E/master_idf_solveragent_physcosine_native_eva \
bash scripts/grid2op/eva_native.sh

SEEDS=42,43,44 PAPER_ROOTS=1 SHARD_TAG=a \
MODES=no_opponent,default \
RUN_PREFIX=$B/ljn-toponn/master_idf_toponn \
bash scripts/grid2op/ljn_toponn_eval.sh

SEEDS=45,46 PAPER_ROOTS=1 SHARD_TAG=b \
MODES=no_opponent,default \
RUN_PREFIX=$B/ljn-toponn/master_idf_toponn \
bash scripts/grid2op/ljn_toponn_eval.sh
```

The RQ2 significance tables and survival figure are then rebuilt with:

```
python scripts/analysis/grid2op_significance.py --family rq2_idf
python scripts/analysis/grid2op_significance.py --family rq2_wcci

python scripts/analysis/make_paper_figures.py --only rq2_survival \
  --rq2-env-root outputs/runs/grid2op/l2rpn_idf_2023 \
  --out-dir outputs/figures
```

Analysis outputs. The analysis scripts compute the numerical tables from artifacts rather than hardcoding values. `analyze_static_seed_sweep.py` summarizes PF-projected RQ1 metrics, `grid2op_significance.py` computes the paired RQ2 significance tests, and `make_paper_figures.py` writes each figure together with CSV and Markdown sidecar tables. The notebook `paper_results.ipynb` is an unexecuted hub that calls the same scripts, so it is optional rather than the only way to recover the reported values. The package also includes `reproduce/run_target.py` and `reproduce/targets.yaml` as a convenience wrapper around the same README commands; it can list and dry-run the targets, but the direct commands above are the authoritative reproduction path.