GraphIR: A Canonical Graph Intermediate Representation for Power-Grid Control

Anonymous authors
Paper under double-blind review

Abstract

Graph neural networks are increasingly used for power-grid control, yet Grid2Op observations are typically converted into graphs through model-specific preprocessing. This couples simulator indexing, graph semantics, architecture, and action design, making representation choices difficult to reproduce or compare. We introduce **GraphIR**, a canonical intermediate representation that parses a Grid2Op state once, preserves equipment identity and terminal-level topology, and deterministically projects the state into task-specific graph views. GraphIR exposes representation as a controlled experimental variable while retaining traceability to simulator objects. Our evaluation isolates representation, system, and sampler effects through view ablations, flat and graph baselines, cross-environment validation, runtime measurements, and entity-level diagnostics.

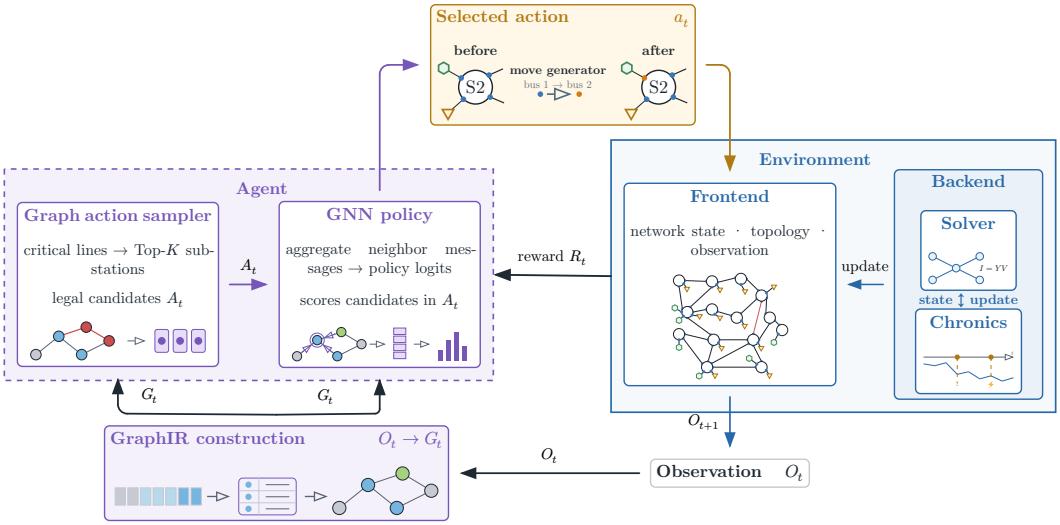


Figure 1: At each Grid2Op step, observation O_t is encoded once as GraphIR G_t. A graph-based sampler constructs legal candidates A_t, and the GNN scores them to select a_t. The environment executes the action, advances the exogenous chronics, and returns reward R_t and O_{t+1} through the same observation interface.

1 Introduction

Power-grid operators must respond to time-varying demand, generation, outages, and thermal limits while preserving reliable service. Grid2Op turns this problem into a sequential control benchmark with realistic network constraints and a combinatorial set of topology and dispatch actions. Graph neural networks (GNNs) are a natural fit because the grid is relational and its energized topology changes with control. However, most graph-based Grid2Op agents define the graph inside model-specific preprocessing. Simulator parsing,

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.

entity identity, feature engineering, message passing, and action design consequently become difficult to separate, and thus difficult to measure the impact of the graph representation.

This coupling creates an experimental problem. When two agents differ, it is unclear whether the difference comes from the graph schema, selected features, neural architecture, action filtering, or reinforcement-learning procedure. It also creates an engineering problem: each graph view or model can require another implementation of Grid2Op's indexed observation semantics. Existing graph agents have shown the representation's power, including homogeneous bus-line graphs, line-graph constructions, heterogeneous object-busbar graphs, and GNN feature extractors for distributed controllers. What remains missing is a stable and defined boundary between the simulator and graph-learning system: a representation layer that preserves Grid2Op semantics before any particular graph view or controller is chosen.

We introduce **GraphIR**, a canonical graph intermediate representation for Grid2Op. Here, **canonical** means with respect to a particular environment: every simulator object has one persistent identity, retained entities have source mappings, topology is complete at the terminal level, static and dynamic state have separate lifetimes, and projections are deterministic and non-mutating. GraphIR is intentionally representation-agnostic at the IR boundary. It can express the full canonical relation set, while any model-facing projection may choose only the relations and features needed by a task. The implementation used in this paper instantiates a compact two-view projection, though different counts and types of projections are possible.

To our knowledge, GraphIR is the first Grid2Op-oriented intermediate representation that explicitly separates canonical simulator semantics from learned graph projections while preserving persistent equipment identities, terminal-to-busbar assignments, source provenance, and projection metadata for representation-controlled RL experiments.

We study the following question: **how should a power-grid state be represented so that graph views can be compared, reused, and interpreted independently of the downstream controller?** We treat graph-conditioned candidate construction as a downstream case study: it tests whether the same semantic state can support action reasoning without another simulator-specific parser.

Our contributions are:

- a precise set of canonicality invariants for a terminal-aware Grid2Op state, including identity uniqueness, source traceability, topology completeness, lifecycle separation, and deterministic projection;
- a representation-agnostic projection interface that lets downstream controllers materialize different graph views from one canonical state rather than reparsing Grid2Op arrays;
- a two-view GraphIR PPO instantiation that fuses an operating-grid view with a typed physical-equipment view; and
- an evaluation design that details action selection and model failure modes

2 Background and Problem Formulation

2.1 Grid2Op as a Markov decision process

Grid2Op models transmission-grid operation as a Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$ (Marot et al., 2020; Sutton and Barto, 2018). At timestep t, the environment exposes an observation o_t containing electrical quantities, equipment status, current bus assignments, cooldowns, forecasts, and time features. The agent selects an action a_t from topology, line-status, redispatch, curtailment, storage, or no-op controls. A power-flow solver and the next exogenous chronic produce o_{t+1} and reward r_t. Episodes terminate on grid failure or the benchmark horizon. Because a single observation may omit latent

exogenous state, we use the standard MDP formulation operationally while reporting the exact observation and forecast features supplied to each agent.

Grid2Op stores observations as typed arrays, where equipment use distinct index spaces, while `topo_vect` records the busbar assignment of equipment terminals. A topology action can therefore change energized connectivity without changing the identity of the affected equipment.

A graph builder must therefore resolve object identity, terminal position, substation membership, line-end orientation, connectivity, and time-varying electrical state. GraphIR preserves persistent identity separately from current busbar assignment.

2.2 Graph neural policies

A graph is $G_t = (V_t, E_t, X_t^V, X_t^E)$, with entities V_t, relations E_t, and node X_t^V and edge X_t^E features. A message-passing GNN updates each node from its own state and aggregated neighbor messages; a readout then yields node-, edge-, or graph-level embeddings (Gilmer et al., 2017). In this work a graph-level embedding feeds policy and value heads. GraphIR defines the input semantics and graph projections, which is invariant to a GNN's message-passing operator or RL optimizer. This separation lets the same policy architecture compare graph views and lets the same graph view support different downstream learners.

3 Related Work

Graph-based Grid2Op agents use GNNs as value or policy encoders, learned power-flow surrogates, imitation classifiers, and components of search or hierarchical controllers (Jong et al., 2025; Taha et al., 2022; Xu et al., 2020; Yoon et al., 2021). Their graphs vary substantially: buses, substations, equipment, or lines may become nodes; relations may represent physical adjacency, active bus topology, or typed equipment incidence. These choices are usually bound to a particular architecture and control pipeline rather than specified as a reusable simulator interface.

Performance also depends on mechanisms outside the GNN. Strong systems restrict action families, construct unitary-action libraries, mask illegal actions, gate intervention to stressed states, search over candidates, or validate actions with simulation (Lehna et al., 2023; Manczak et al., 2023; Marot et al., 2021). Differences in benchmark, chronics, safety logic, and action construction therefore prevent controller score alone from identifying the effect of graph representation.

Recent representation-focused work compares homogeneous and heterogeneous formulations more directly (Jong et al., 2025), but does not define a simulator-level state with explicit identity, traceability, lifecycle, and projection invariants. GraphIR addresses this narrower gap by holding parsing semantics fixed while varying graph projection and downstream learner. Appendix A compares graph representations and action-space reduction from the literature.

4 GraphIR

4.1 Canonical contract and typed schema

GraphIR separates environment-level semantics from observation-dependent state. For an environment specification S, initialization constructs the static schema cache

$$C = C(S) = (V, E^{\text{static}}, \Sigma, M), \quad I_t = U(C, o_t). \tag{1}$$

Here C contains persistent entities V, static typed relations E^{static}, entity/relation/feature schemas Σ, and mappings M to Grid2Op indices or derived identifiers. Binding observation o_t with U yields canonical state I_t: the same identities plus live terminal assignments, active

relations, and node/edge features. This is a logical state; its arrays may be materialized lazily.

Appendix B instantiates these operators on a five-bus fixture: $C(S_5)$ fixes 5 substations, 8 lines, 2 generators, 3 loads, and 21 permanent terminal owners, while $U(C(S_5), o_t)$ supplies their live assignments and electrical state.

$C(S)$ is the immutable `GridIndex` and schema metadata, while I_t is represented lazily by binding that index to the current Grid2Op observation o_t. Projection recipes are compiled against $C(S)$ but are not part of the canonical state. We call I_t canonical with respect to (S, o_t) when it satisfies five invariants: **(i)** identity uniqueness, **(ii)** source traceability, **(iii)** terminal-level topology completeness, **(iv)** separation of static specification from dynamic state, and **(v)** deterministic, non-mutating projection.

For line L_l, $T^{\text{or}}_{L_l}$ and $T^{\text{ex}}_{L_l}$ are its fixed Grid2Op **origin** and **extremity** terminals. They identify measurement sides, regardless of power-flow direction. Identities persist when bus assignments change.

Table 1: Canonical GraphIR entities and relations.

Schema	Canonical members
Entities	Substations, busbars, generators, loads, storage, lines, terminals
Static relations	`contains`, `located_at`, `has_terminal`
Live relation	`assigned_to`: terminal → active busbar

For example, in the five-bus fixture `gen_pos_topo_vect[0] = 4` identifies topology position 4 as belonging to generator 0′s terminal T_{G_0}. If `topo_vect[4] = 2`, GraphIR assigns T_{G_0} to the second local busbar of its substation with relation `assigned_to`. If `topo_vect[4] = -1`, the terminal remains identifiable but receives no active-bus assignment. A projection may subsequently collapse this path, but the canonical state preserves it.

The canonical relation set is intentionally richer than the projection used by any one policy. Relations such as `contains` and `has_terminal` are part of the simulator-facing semantic contract, but they need not become learned message-passing edges. This distinction is central to the IR claim: GraphIR preserves all required identities, ownership, terminal assignments, and source mappings first, then lets downstream projections decide which relations are task-relevant.

4.2 Construction and validation

Static identities, terminal ownership, endpoint roles, substation membership, schemas, and source mappings are initialized once per environment. Each observation supplies assignments, connectivity, electrical quantities, cooldowns, and maintenance state through a lazy binding.

Algorithm 1: GraphIR initialization, observation binding, and projection.

Input: environment S, observations (o_t), requested recipes $\mathcal{H}$.
1 $C \leftarrow C(S)$; validate identities, owners, indices, and endpoint roles.
2 **For each** observation o_t:
(a) $I_t \leftarrow U(C, o_t)$; bind live state to the fixed schema.
(b) Validate assignments and feature dimensions.
(c) **For each** $\eta \in \mathcal{H}$, check its contract; set $G_t^\eta \leftarrow P_{\eta(I_t)}$.
(d) **Yield** I_t and requested G_t^η.

At initialization, GraphIR checks that every `topo_vect` position has exactly one permanent terminal owner, positions are in range, and line origin and extremity terminals remain distinct. During materialization, connected line edges must have valid endpoint bus assignments, disconnected endpoints are represented by masks rather than dropped tensor slots, and feature columns must agree in length. The full schema form additionally validates entity-id uniqueness, terminal ownership, bus-assignment consistency, and relation buckets.

In the running example, topology positions 0 and 4 bind $T^{\text{or}}_{L_0}$ and T_{G_0} to $B_{0,2}$, whereas position 5 binds T_{D_0} to $B_{0,1}$. These assignments change with o_t without changing the terminals' permanent identities in $C(S_5)$.

4.3 Declarative graph projections

To avoid overloading p (active power), let η denote a projection recipe:

$$G_t^{\eta} = P_{\eta(I_t)}. \tag{2}$$

A recipe η specifies retained entity and relation types, selected or derived features, edge orientation, and disconnected-equipment masks. P_η is deterministic and does not alter I_t. The current paper materializes two learned views from one lazy GraphIR binding:

On the five-bus fixture, the same declarations produce an operating view with 10 fixed bus slots and 8 line slots, and a physical view with 15 typed nodes. The appendix places these instantiated outputs directly beside the corresponding schema and binding operations.

- **Operating grid:** a homogeneous fixed-capacity bus-line graph. Nodes are bus slots ordered by substation and local busbar; node features aggregate generator and load $\frac{p}{q}$, aggregate line-side voltages, and substation cooldowns. Line slots retain origin-to-extremity Grid2Op orientation and carry `p_or`, `q_or`, `a_or`, `rho`, overflow, cooldown, and maintenance attributes. `node_mask` and `edge_mask` record which fixed slots are active at t.
- **Physical:** a heterogeneous graph with bus, generator, load, and storage nodes. Equipment nodes carry their own $\frac{p}{q}$ or storage state; entity-to-bus relations use the current terminal assignment; the line relation carries structural availability (`line_status`) and line cooldown only. Electrical flow and loading are not duplicated here because they already live in the operating-grid view.

The recipe does not introduce separate topology, energy-flow, or operational-constraint views. Topology, electrical state, and operational availability are fused in `operating_grid`; typed equipment semantics are preserved in `physical`. Inside the learner, active line edges are compacted and duplicated for bidirectional message passing, while the Grid2Op origin/extremity orientation remains the source of side-specific measurements and fixed indices. The experiment metadata records the selected representation (`flat_mlp`, `operating_gnn`, or `graphir_gnn`), distinguishing relational structure from information availability under the fixed default recipe.

Each projection is therefore a subset instantiation of GraphIR. A fuller projection could materialize substations, explicit terminal nodes, `contains`, `has_terminal`, or all potential busbar attachments for a different ablation. This work uses the smaller two-view projection because the omitted canonical relations are either static relations already encoded by fixed indices and masks, or are reserved for future terminal-level and potential-attachment studies. Appendix B displays only the feature columns needed to illustrate identity, assignment, masking, and projection; the implemented tensors use the complete feature sets specified above.

4.4 Policy interface

The full GraphIR control path is

$$h_t^{\eta} = \text{Readout}_{\eta\left(\text{GNN}_{\varphi_\eta}\left(P_{\eta(I_t)}\right)\right)}, \quad \eta \in \{\text{op}, \text{phys}\}, \tag{3}$$

$$z_t = \text{Fuse}\left(h_t^{\text{op}}, h_t^{\text{phys}}\right), \quad a_t \sim \pi_{\theta(\cdot \mid z_t, \mathcal{A}_t)}. \tag{4}$$

Here φ_{op} and φ_{phys} are separate encoder parameters; h_t^{η} is the graph-level embedding returned by the corresponding readout; z_t is the fused representation; $\mathcal{A}_t$ is the admissible

candidate action set; and a_t is the sampled or selected action. The environment then produces the next observation, reward, and termination indicator. The primary fusion concatenates the two embeddings and applies a small MLP and layer normalization; a learned gate exists only as an optional diagnostic. Shared policy and value heads consume z_t. The `operating_gnn` baseline uses only the operating-grid view. The `flat_mlp` baseline flattens the same fixed-shape GraphIR TensorDict before an MLP, separating relational structure from information availability.

5 Graph-Conditioned Candidate Construction

As a downstream case study, we use the same graph state available to GraphIR to construct state-dependent action candidates without introducing another simulator parser. Grid2Op's nominal action space is combinatorial, although few interventions are relevant in a given state. We factor control into deterministic candidate construction and policy selection:

$$I_t \to G_t^i \to E_t^{\text{crit}} \to V_t^K \to A_t^K \to \pi_{\theta(a \mid G_t^i, A_t^K)}. \tag{5}$$

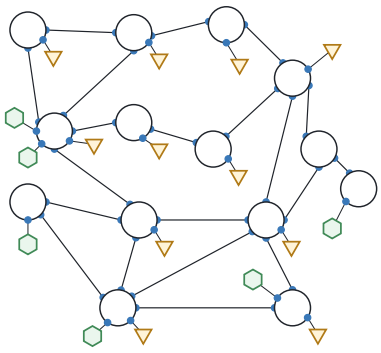
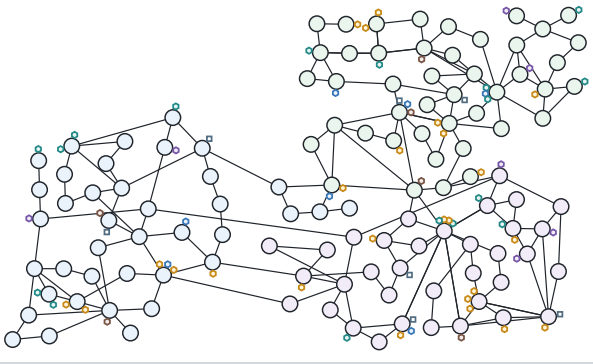
When $\max_e \rho_e < \tau_{\text{act}}$, the intervention task can return no-op. Otherwise, branches above τ_{crit} define E_t^{crit}. The implemented case118 sampler deterministically restricts the action family to topology-set actions, expands a k-hop neighborhood around critical-line endpoints on the physical grid graph, rejects fragile local bus assignments with structural checks, ranks actions that affect warning or unsafe lines, and exact-checks Grid2Op legality only on a bounded ranked refill stream.

6 Experimental Evaluation

We evaluate whether graph-structured observation encodings change policy behavior under matched training and evaluation conditions on the Case 14 and Case 118 environments. The study compares three policy architectures: a flat multilayer perceptron (**MLP**), the proposed multi-view **GraphIR** policy, and a graph policy restricted to the bus-topology view (**bus topology**). The comparison asks whether the representations improve recovery behavior and the ability to exit a controlled unsafe state.

6.1 Experimental protocol

Table 2: **Evaluation environments.** Structural counts describe `l2rpn_case14_sandbox` and `l2rpn_idf_2023`; run-dependent configuration is reported in Appendix B.

Case 14	**Case 118**
14 substations · 20 lines · 11 loads 6 generators · 0 storage	118 substations · 186 lines · 99 loads 62 generators · 7 storage
	

We report a controlled unsafe-state exit probe, where immediate loading relief is $\Delta\rho_t = \max_e \rho_{t,e} - \max_e \rho_{t+1,e}$. The probe reports whether maximum loading falls below the exit threshold within H steps. Entries are mean $\pm$ across-seed standard error over five seeds.

Capacity- and margin-recovery scores remain bounded task summaries whose final equations should accompany the evaluation code.

All three learned policies use the same evaluation chronics, action set, reward definition, and termination rules within each case. Differences cannot be attributed solely to relational structure without feature parity, matched training interactions, checkpoint selection, and model-specific tuning controls.

Table 3: **Controlled unsafe-state exit across cases.** All entries are mean ± standard error over five seeds. $\Delta\rho$ is immediate maximum-loading relief; ρ after is the mean immediately after action. Case 14 includes the shared no-op/heuristic baseline; Case 118 uses the common deterministic graph-localized action mask.

Policy	Exit by H ↑	$\Delta\rho$ ↑	ρ after ↓
Case 14 · controlled exit			
No-op / heuristic	0.000 ± 0.000	0.005 ± 0.000	1.002 ± 0.000
MLP	1.000 ± 0.000	0.103 ± 0.000	0.904 ± 0.000
Bus topology	0.800 ± 0.200	0.089 ± 0.014	0.918 ± 0.014
GraphIR	1.000 ± 0.000	0.103 ± 0.000	0.904 ± 0.000
Case 118 · controlled exit			
MLP	0.822 ± 0.046	-0.002 ± 0.017	1.057 ± 0.055
Bus topology	0.867 ± 0.024	0.024 ± 0.016	1.092 ± 0.029
GraphIR	0.826 ± 0.035	-0.008 ± 0.022	1.094 ± 0.016

6.2 Controlled unsafe-state exit

The controlled probe isolates that local corrective behavior. Both the MLP and GraphIR exit the unsafe region in all evaluated seeds after one action, reducing mean maximum loading from $\rho = 1.007$ to 0.904 (Table 3). The bus-topology policy exits in 0.800 ± 0.200 of runs and reaches $\rho = 0.918 \pm 0.014$. The shared no-op/heuristic baseline never exits and reduces loading by only 0.005. No learned policy produces an illegal action or terminal transition in this probe.

Case 118 is harder and uses the deterministic graph-localized action mask described above. In the five-seed, 1000-state test probe, the bus-topology policy has the largest observed exit rate, Exit@H $= 0.867 \pm 0.024$, followed by GraphIR at 0.826 ± 0.035 and the MLP at 0.822 ± 0.046 (Table 3). The paired graph-vs-flat deltas remain uncertain: GraphIR improves exit rate by only 0.003 with bootstrap interval $[-0.041, 0.058]$, while the bus-topology view improves it by 0.045 with interval $[-0.057, 0.152]$.

The bus-topology advantage is plausible, as the Case 118 task is topology-only recovery under a strong candidate constructor: the mask already restricts the policy to about 127 local actions near stressed equipment plus a do-nothing action. In this scenario, the operating-grid view may be the cleaner as it emphasizes the bus connectivity and electrical state that directly determine whether a topology action relieves the overload. GraphIR supplies additional typed physical context, but additional context is useful only if the task and training budget let the policy exploit it. Here the extra view appears at least partly redundant with the operating graph and may increase optimization difficulty. Consistent with that interpretation, the bus-topology policy has fewer terminal-after-action outcomes, 0.030 ± 0.026 versus 0.052 ± 0.020 for GraphIR, higher action-valid rate, 0.970 ± 0.026 versus 0.948 ± 0.020, and more immediate loading relief, 0.024 ± 0.016 versus -0.008 ± 0.022.

The location of selected actions also differs across representations (Figure 2). The three maps show where each policy intervenes and the observed exit rate conditional on acting at that substation. Figure 3 shows a single instance, with line overloads and each model's decision and what substation they are targeting. Appendix E provides additional single instance examples.

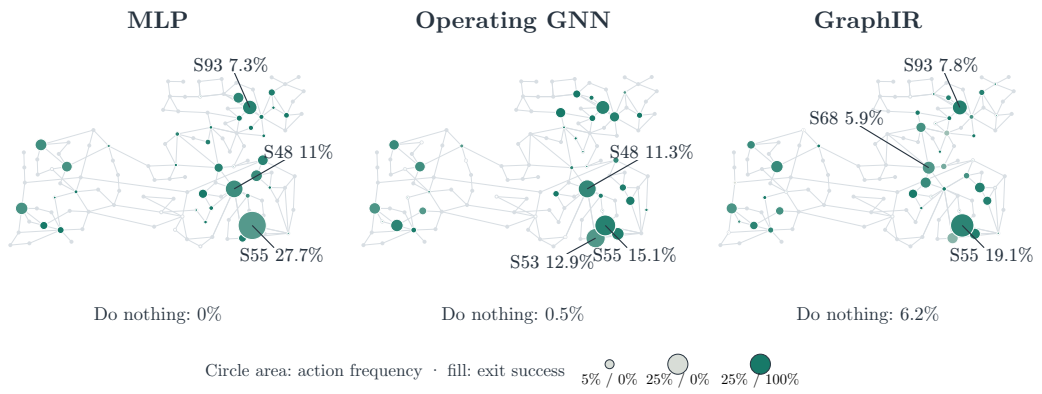


Figure 2: **Action locations of three policies on case 118.** Each map shows the fraction of 5,000 test scenario decisions targeting each substation; circle area indicates action frequency and the neutral-to-teal fill indicates the fraction of selected actions that achieve the exit-success criterion within the configured horizon. Do-nothing actions are included in the 5,000-decision denominator but have no mapped substation. Success rates for rarely selected actions have high uncertainty. The three maps share one frequency-size and success-color scale.

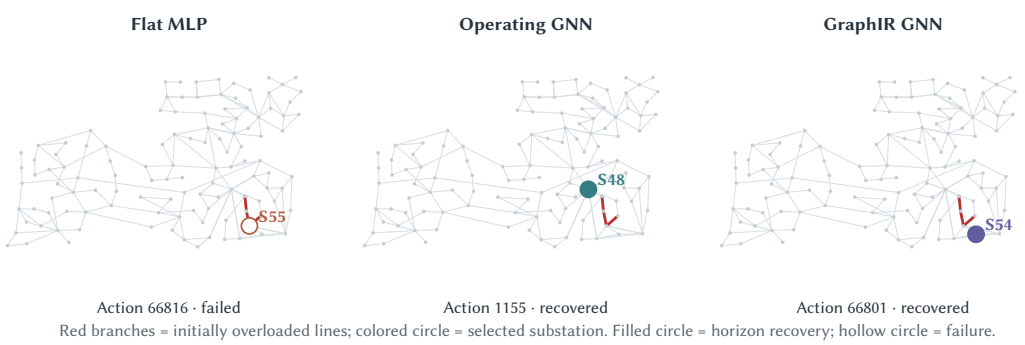


Figure 3: Case 118 matched state (seed 3, chronic 39, step 139). Flat MLP chooses S55 and fails; Operating GNN chooses S48 and recovers; GraphIR GNN chooses S54 and recovers. Red branches are the three initially overloaded lines (IDs 146, 148, 149). The recovery outcome is measured over multiple steps. Appendix E provides additional examples.

6.3 Interpretation and limitations

The stress-conditioned view in Figure 4 shows that unsafe-state exit changes with initial maximum loading, with different shapes across the three policies. In Case 118, GraphIR falls from 91.7% in the $\rho \in [1, 1.05)$ bin to 76.8% in $[1.1, 1.2)$, then reaches 79.6% for $\rho \geq 1.2$. The bus-topology policy reaches 87.4% in the highest bin, while the MLP reaches 72.9%. These are pooled decisions within loading bins, and the GraphIR-labelled cohort contains many more decisions than the comparator cohorts. They describe the sampled stress distribution and cannot establish that one representation is more robust to severe overload.

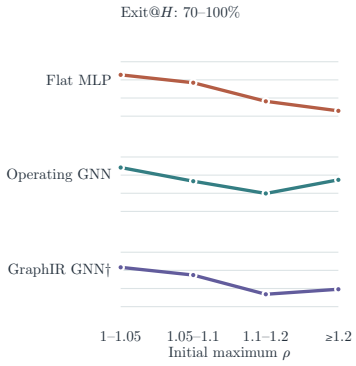


Figure 4: Case 118 unsafe-state exit across initial loading bins, shown as three separate model trajectories. Each point pools overload-count strata by decision count. The marks denote 70%, 80%, 90%, and 100%.

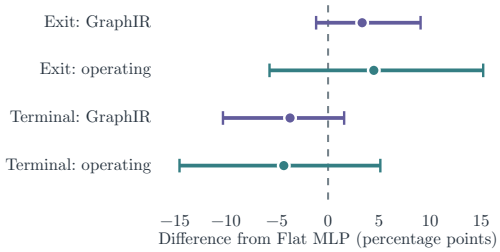


Figure 5: Case 118 paired-seed differences from Flat MLP for horizon exit and terminal rate (five paired seeds). Positive Exit@H and negative terminal rate are favorable.

The paired Case 118 **intervention-recovery** comparison in Figure 5 is a separate evaluation from the controlled exit probe above. Relative to the MLP, its mean Exit@H difference is +3.3 percentage points for GraphIR and +4.5 points for bus topology. The corresponding terminal-rate differences are −3.7 and −4.3 points. These directions favor the graph policies, but all four supplied intervals cross zero.

The bus-topology policy has the highest observed Case 118 rate in the controlled exit probe (Table 3), while GraphIR and the MLP are close. Operating-grid connectivity appears sufficient for many actions under this graph-localized candidate mask. The added physical view has no demonstrated benefit here. Its utility under a different action mask or stress distribution remains an open experiment.

Failure geometry suggests one reason for the remaining difficulty (Table 4). States where all three policies fail have 3.46 overloaded lines on average, compared with 1.94 where all succeed. Separated overloads occur in 60% versus 34%, and multi-area overloads in 34% versus 23%. This is consistent with a challenge for policies choosing local topology actions when stress is distributed across the grid. Future work to address the current restriction of a single action would be to introduce multi-agent training.

Table 4: **Overload geometry by exit pattern.** Lines = mean overloaded lines; Op. = bus-topology policy; GIR = GraphIR. **Sep.** is the share with overloaded lines separated by at least two hops; **Areas** is the share spanning more than one of the three area partitions.

Exit pattern	Lines	Sep.	Areas
All succeed	1.94	34%	23%
MLP + Op.	1.94	45%	27%
MLP + GIR	2.44	7%	6%
Op. + GIR	1.99	8%	3%
MLP only	2.41	46%	21%
Op. only	1.46	19%	6%
GIR only	1.98	50%	10%
All fail	3.46	60%	34%

7 Conclusion

GraphIR defines a stable boundary between Grid2Op's indexed observation interface and model-facing graph construction. It creates a terminal-aware state with explicit identity, traceability, lifecycle, validation, and projection invariants. This boundary makes graph representation inspectable and allows multiple views and downstream controllers to reuse the same simulator semantics.

Graph representation does show improvement over the base feature vector across different grid sizes, though topology views have shown thus far to largely contribute to this improvement. Relational encoding is compatible with corrective control, but additional heterogeneous structure does not automatically improve learning at this scale, and will need additional investigation.

AI use statement

In this work, we used generative AI tools for paper styling (tables, figures, equations), writing polish, reference verification, assistance with vector-based diagrams, and sections of code for initial implementation road mapping and test verification. We have not used generative AI tools for written context (unless for above writing polish and editing), core code and contributions, and any takeaways or discussion from results. All AI-assisted work was manually reviewed by at least one of the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Reproducibility statement

Appendix B gives an example detailing the GraphIR methodology for one of the smaller power grid scenarios. Appendix D gives experimental details of each model trained in this paper. Appendix E provides training plots and additional snapshots of inference decisions of each model.

References

Fabrizio, C., Losapio, G., Mussi, M., Metelli, A. M., and Restelli, M. Power Grid Control with Graph-Based Distributed Reinforcement Learning. *Arxiv Preprint Arxiv:2509.02861*, 2025. https://doi.org/10.48550/arXiv.2509.02861

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural Message Passing for Quantum Chemistry. *Proceedings of the 34th International Conference on Machine Learning*, *70*, 1263–1272, 2017. https://proceedings.mlr.press/v70/gilmer17a.html

Hassouna, M., Holzhüter, C., Lehna, M., Jong, M. de, Viebahn, J., Sick, B., and Scholz, C. Learning Topology Actions for Power Grid Control: A Graph-Based Soft-Label Imitation Learning Approach. *Machine Learning and Knowledge Discovery in Databases: Applied Data Science Track*, 2025. https://doi.org/10.1007/978-3-032-06129-4_8

Jong, M. de, Viebahn, J., and Shapovalova, Y. Graph Neural Networks for Transmission Grid Topology Control: Busbar Information Asymmetry and Heterogeneous Representations. *Arxiv Preprint Arxiv:2501.07186*, 2025. https://doi.org/10.48550/arXiv.2501.07186

Lehna, M., Viebahn, J., Marot, A., Tomforde, S., and Scholz, C. Managing Power Grids Through Topology Actions: A Comparative Study Between Advanced Rule-Based and Reinforcement Learning Agents. *Energy and AI*, *14*, 100276, 2023. https://doi.org/10.1016/j.egyai.2023.100276

Liu, S., Zhou, Y., Song, M., Bu, G., Guo, J., and Chen, C. Progressive Decision-Making Framework for Power System Topology Control. *Expert Systems with Applications*, *235*, 121070, 2024. https://doi.org/10.1016/j.eswa.2023.121070

Manczak, B., Viebahn, J., and Hoof, H. van. Hierarchical Reinforcement Learning for Power Network Topology Control. *Arxiv Preprint Arxiv:2311.02129*, 2023. https://doi.org/10.48550/arXiv.2311.02129

Marot, A., Donnot, B., Dulac-Arnold, G., Kelly, A., O'Sullivan, A., Viebahn, J., Awad, M., Guyon, I., Panciatici, P., and Romero, C. Learning to Run a Power Network Challenge: A Retrospective Analysis. *Proceedings of the Neurips 2020 Competition and Demonstration Track*, *133*, 112–132, 2021.

Marot, A., Donnot, B., Romero, C., Veyrin-Forrer, L., Lerousseau, M., Donon, B., and Guyon, I. Learning to Run a Power Network Challenge for Training Topology Controllers. *Electric Power Systems Research*, *189*, 106635, 2020. https://doi.org/10.1016/j.epsr.2020.106635

Moradi, M., Panahi, S., Zhai, Z.-M., Weng, Y., Dirkman, J., and Lai, Y.-C. Heterogeneous Reinforcement Learning for Defending Power Grids Against Attacks. *APL Machine Learning*, *2*(2), 26121, 2024. https://doi.org/10.1063/5.0216874

Sar, E. van der, Zocca, A., and Bhulai, S. Multi-Agent Reinforcement Learning for Power Grid Topology Optimization. *Arxiv Preprint Arxiv:2310.02605*, 2023. https://doi.org/10.48550/arXiv.2310.02605

Sutton, R. S., and Barto, A. G. *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press, 2018. http://incompleteideas.net/book/the-book-2nd.html

Taha, S., Poland, J., Knezovic, K., and Shchetinin, D. Learning to Run a Power Network Under Varying Grid Topology. *2022 IEEE 7th International Energy Conference (ENERGYCON)*, 1–6, 2022. https://doi.org/10.1109/ENERGYCON53164.2022.9830198

Tang, F., Wang, Y., Zhang, Y., Guo, H., Gao, S., Shen, C., and Zhang, Z. Weather-Aware Reinforcement Learning for Power Grid Topology Control Under Extreme Events. *IET Smart Grid*, 2026. https://doi.org/10.1049/stg2.70099

Xu, P., Duan, J., Zhang, J., Pei, Y., Shi, D., Wang, Z., Dong, X., and Sun, Y. Active Power Correction Strategies Based on Deep Reinforcement Learning—Part I: A Simulation-Driven Solution for Robustness. *CSEE Journal of Power and Energy Systems*, *8*(4), 1122–1133, 2022. https://doi.org/10.17775/CSEEJPES.2020.07090

Xu, P., Pei, Y., Zheng, X., and Zhang, J. A Simulation-Constraint Graph Reinforcement Learning Method for Line Flow Control. *2020 IEEE 4th Conference on Energy Internet and Energy System Integration (Ei2)*, 319–324, 2020. https://doi.org/10.1109/EI250167.2020.9347305

Yoon, D., Hong, S., Lee, B.-J., and Kim, K.-E. Winning the L2RPN Challenge: Power Grid Management via Semi-Markov Afterstate Actor-Critic. *International Conference on Learning Representations*, 2021. https://openreview.net/forum?id=LmUJqB1Cz8

Zhao, Y., Liu, J., Liu, X., Yuan, K., Ren, K., and Yang, M. A Graph-Based Deep Reinforcement Learning Framework for Autonomous Power Dispatch on Power Systems with Changing Topologies. *2022 IEEE Sustainable Power and Energy Conference (Ispec)*, 1–5, 2022. https://doi.org/10.1109/iSPEC54162.2022.10033001

Zhou, B., Zeng, H., Liu, Y., Li, K., Wang, F., and Tian, H. Action Set Based Policy Optimization for Safe Power Grid Management. *Machine Learning and Knowledge Discovery in Databases. Applied Data Science Track*, 151–167, 2021.

A Literature Review

The comparison in this appendix is organized around the **representation layer**, rather than around controller performance. Prior studies differ in task, action space, learning algorithm, and benchmark, so their reported scores are not directly comparable. The table instead asks whether each method makes the graph construction explicit and reusable: whether it defines a schema (**S**), decouples schema from any one model (**D**), supports multiple graph views (**V**), and evaluates representation choice as an experimental variable (**A**).

Work	Method		Evidence	
Work	**Task and controller**	**Graph representation**	**Evaluation**	**Representation support**
(Xu et al., 2020)	Flow Topology **Controller** DQN Simulation	GCN Homogeneous	Grid2Op IEEE 14-bus	**S** — **D** — **V** — **A** —
(Taha et al., 2022)	Topology **Controller**	Residual GCN Surrogate	Grid2Op IEEE 118-bus	**S** — **D** △ **V** — **A** —

Work	Method		Evidence	
Work	**Task and controller**	**Graph representation**	**Evaluation**	**Representation support**
	MCTS			
(Xu et al., 2022)	Topology Redispatch **Controller** Dueling DDQN MCTS	GAT	Grid2Op IEEE 118-bus L2RPN	**S** — **D** — **V** — **A** —
(Yoon et al., 2021)	Topology **Controller** SMAAC	Graph attention Transformer	Grid2Op IEEE 5/14/118-bus WCCI 2020	**S** — **D** — **V** — **A** —
(Zhao et al., 2022)	Redispatch Curtailment Storage **Controller** PPO Pretraining	GraphSAGE Dynamic	Power-system simulator IEEE 118-bus	**S** — **D** △ **V** — **A** △
(Sar et al., 2023)	Bus switching **Controller** Hierarchical MARL PPO SACD	Non-graph	Grid2Op IEEE 5-bus	**S** — **D** — **V** — **A** —
(Manczak et al., 2023)	Topology **Controller** Hierarchical RL	Non-graph	Grid2Op IEEE 14-bus	**S** · **D** · **V** · **A** ·
(Fabrizio et al., 2025)	Topology **Controller** Distributed MARL DQfD	GNN Homogeneous	Grid2Op IEEE 14-bus l2rpn_case14_sandbox	**S** — **D** — **V** — **A** —
(Jong et al., 2025)	Topology prediction Robust control **Controller** Supervised Imitation	Homogeneous Heterogeneous	Grid2Op IEEE 14-bus + N-1 rte_case14_realistic	**S** — **D** △ **V** △ **A** ✓
(Hassouna et al., 2025)	Topology selection **Controller** Soft-label IL	GNN classifier	Grid2Op IEEE 118-bus WCCI 2022	**S** — **D** — **V** — **A** —
GraphIR (ours)	Topology **Controller** Model-agnostic	Canonical IR Projected views	Grid2Op Multiple GraphIR evaluation	**S** ✓ **D** ✓ **V** ✓ **A** ✓

S: explicit schema; **D**: model-decoupled; **V**: multiple graph views; **A**: representation ablation. ✓ yes △ partial — no · not applicable.

A.1 Takeaway

Graph structure is already well established in power-grid learning, but it is usually embedded inside a particular controller. Across the surveyed work, graphs serve as policy encoders, learned surrogates, or structures for search and multi-agent decomposition. In most cases, however, the mapping from simulator arrays to graph entities and relations is specific to that model and task. The graph representation is therefore part of the controller implementation rather than an independently specified interface that can be reused across architectures or compared through controlled view ablations.

The closest representation-focused comparison is (Jong et al., 2025), which evaluates homogeneous and heterogeneous graphs and studies their effect on generalization. This provides important evidence that graph design matters, but it does not define a persistent

simulator-level intermediate representation from which several task-specific views can be projected. Conversely, hierarchical and non-graph controllers demonstrate that useful action decomposition does not itself require a canonical graph schema.

B GraphIR Worked Example

This appendix pairs the formal construction with one Grid2Op observation from Grid2op's `rte_case5_example`.

B.1 Canonical schema and persistent identity

The five-bus instance $C(S_5)$ contains 5 substations, 8 lines, 2 generators, 3 loads, no storage, and 21 topology positions. It fixes entity identities, terminal ownership, endpoint roles, schemas, and source mappings before an observation is bound. For line L_l, or and ex mean its fixed Grid2Op **origin** and **extremity** endpoints. They identify measurement sides, not the current direction of power flow. The permanent owner of each topology position is shown below.

Sub.	`topo_vect` positions	Permanent owners in order
S_0	0–5	L_0^{or} L_1^{or} L_2^{or} L_3^{or} G_0 D_0
S_1	6–8	L_0^{ex} L_4^{or} G_1
S_2	9–12	L_1^{ex} L_4^{ex} L_5^{or} L_6^{or}
S_3	13–17	L_2^{ex} L_5^{ex} L_6^{ex} L_7^{or} D_1
S_4	18–20	L_3^{ex} L_7^{ex} D_2

B.2 Observation binding and dynamic state

For the displayed snapshot, we begin from `env.reset()` and apply the legal, convergent action set_bus$(S_0) = [2, 1, 1, 1, 2, 1]$. Thus $T_{L_0}^{or}$ and T_{G_0} move to $B_{0,2}$ while all other terminals remain on local busbar 1.

Bound topology and electrical state at t

S_0: [211121]
S_1: [111]
S_2: [1111]
S_3: [11111]
S_4: [111]

$p^G = [0.90, 23.53]$
$p^D = [8.90, 7.90, 7.50]$
cooldowns are zero.

$o_t^{raw} = \text{to_vect}(o_t) \in \mathbb{R}^{192}$

time 0:5	**G: p, q, v** 6:11	**D: p, q, v** 12:20
line origin: p, q, v, a 21:52	**line extremity: p, q, v, a** 53:84	**ρ, status, overflow** 85:108
`topo_vect` 109:129	**cooldown + maintenance** 130:158	**dispatch + curtailment** 159:170
alarm + step + margins 171:183	**protection** 184:191	

GraphIR resolves these named families through the persistent owners in $C(S_5)$ rather than reconstructing the schema at each PPO step. Let $s(k)$ be the permanent substation of the terminal owning topology position k, and let n_b be the number of busbar slots per substation. Its current assignment is the partial function

$$\alpha_{t(k)} = \begin{cases} B_{s(k),\text{topo_vect}_{t[k]}} & \text{if } \text{topo_vect}_{t[k]} \in \{1, ..., n_b\} \\ \perp & \text{if } \text{topo_vect}_{t[k]} = -1. \end{cases} \tag{6}$$

Thus $T_k \overset{\text{assigned_to}}{\rightarrow} \alpha_{t(k)}$ only when $\alpha_{t(k)} \neq \perp$; $\perp$ denotes no active bus assignment, while the terminal itself remains identifiable. A fixed bus slot $i = n_b s + (b - 1)$ preserves tensor capacity across topology actions. In this fixture $n_b = 2$, so $i = 2s + (b - 1)$. For the

displayed observation, $0 \to T^{\text{or}}_{L_0} \to B_{0,2} \to i = 1$, $4 \to T_{G_0} \to B_{0,2} \to i = 1$, and $5 \to T_{D_0} \to B_{0,1} \to i = 0$.

B.3 Declarative projections instantiated

To avoid overloading p, which denotes active power, let η denote a projection recipe:

$$\eta = (\mathcal{T}_\eta, \mathcal{R}_\eta, F_\eta, d_\eta, \delta_\eta). \tag{7}$$

Here $\mathcal{T}_\eta$ and $\mathcal{R}_\eta$ select retained entity and relation types; F_η specifies selected, derived, and aggregated node and edge features; d_η specifies edge orientation; and δ_η specifies disconnected- item treatment. The recipe induces $P_\eta : I_t \to G_t^\eta$. Applying the two paper recipes to the same bound state I_t yields the operating and physical views below; neither projection mutates the canonical state. For legibility, the matrices show only the fields needed to illustrate identity, assignment, masking, and projection, not every implemented model feature.

Data / tensor form | **Rendered structure**

bus 1 bus 2 dashed/gray = inactive or structural

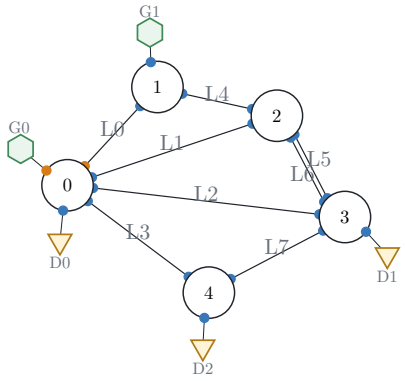
Source o_t

$$\text{topo_vect} = \begin{bmatrix} 2\,1\,1\,1\,2\,1\,1 \\ 1\,1\,1\,1\,1\,1\,1 \\ 1\,1\,1\,1\,1\,1\,1 \end{bmatrix} \quad (8)$$

1x21 matrix, wrapped into three rows for display only

$p^G = [0.90, 23.53]$ $p^D = [8.90, 7.90, 7.50]$

$\rho =$
$[.079, .211, .141, .061, .230, .160, .160, .209]$



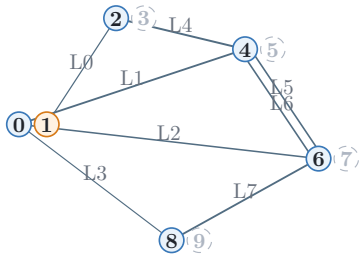
Operating G_t^{op}

$X^{\text{op}} = [i, p^G, p^D, m_V]$

0	0	8.90	1
1	.90	0	1
2	23.53	0	1
3	0	0	0
4	0	0	1
5	0	0	0
6	0	7.90	1
7	0	0	0
8	0	7.50	1
9	0	0	0

$E^{\text{op}} = [u, v, \rho]$

1	2	.079
0	4	.211
0	6	.141
0	8	.061
2	4	.230
4	6	.160
4	6	.160
6	8	.209



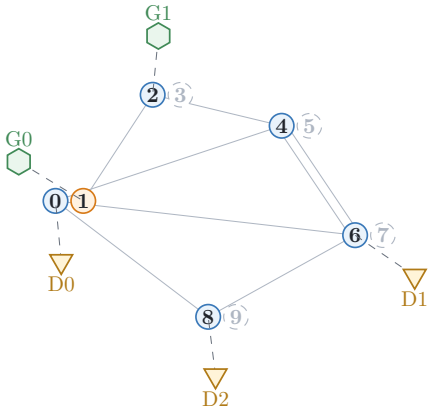
Physical G_t^{phys}

$X^{\text{phys}} = [v, b, p, q]$

G_0	$B_{0,2}$	.90	−8.29
G_1	$B_{1,1}$	23.53	6.14
D_0	$B_{0,1}$	8.90	6.20
D_1	$B_{3,1}$	7.90	5.50
D_2	$B_{4,1}$	7.50	5.30

$E^{\text{attach}} = [v, b]$

G_0	$B_{0,2}$
G_1	$B_{1,1}$
D_0	$B_{0,1}$
D_1	$B_{3,1}$
D_2	$B_{4,1}$



The operating projection preserves 10 fixed bus slots and 8 line slots, including parallel L_5 and L_6; the physical projection instead retains typed equipment nodes and their current bus attachments.

Fixture checks. Each topology position has one permanent terminal owner; retained entities preserve their Grid2Op source indices and endpoint roles; inactive slots are masked rather than deleted; and both $P_{\eta_{\text{op}}}(I_t)$ and $P_{\eta_{\text{phys}}}(I_t)$ leave I_t unchanged. These checks establish construction correctness for the fixture, not policy effectiveness.

C C. Action Sampling

C.1 Action-space reduction strategies

There are two primary mechanisms to address action-space reduction: An *intervention gate* decides **whether** the learned controller should act in state s_t. A *state-conditioned action mask* decides **which** actions the controller may choose after it is activated.

Mechanism	Question and map	Role
Intervention gate	Should the policy act? $g : \mathcal{S} \to \{0, 1\}$	Routes routine states to no-op/baseline control and distressed states to the learned policy.
Action mask	What may it choose? $M : \mathcal{S} \to 2^{\mathcal{A}}$	Returns a legal, relevant candidate set $\mathcal{A}_t = M(s_t) \subseteq \mathcal{A}$ for policy evaluation.

Event-triggered control is the closest established control-theoretic term for the first mechanism; *initiation set* is also appropriate when the learned controller is treated as a temporally extended option. In this work, the gate is

$$g_{\tau(s_t)} = \mathbb{1}\left[\rho_{\max(s_t)} \geq \tau\right], \quad \rho_{\max(s_t)} = \max_{\ell \in \mathcal{L}} \rho_{\ell,t}. \tag{9}$$

States with $g_{\tau(s_t)} = 0$ retain the grid topology; states with $g_{\tau(s_t)} = 1$ activate the recovery policy. Training only from such activated states is therefore *intervention-conditioned recovery training*, distinct from masking the policy's actions. This separation prevents the dominant routine regime from turning recovery learning into an overwhelmingly no-op classification problem, while preserving no-op as a valid safety fallback.

The second mechanism addresses combinatorial control. A substation containing k elements has on the order of 2^{k-1} distinct two-bus configurations, and arbitrary simultaneous changes multiply these choices across substations. Operational limits make the one-step space much smaller than the number of global topologies, but still too large for efficient exhaustive policy scoring.

Grid2Op case	Global topologies	One-step topology	Line actions
Case 14	1.40×10^{9}	179	20
Case 118	3.88×10^{76}	72,150	186

Global counts are unconstrained topology configurations; one-step counts impose Grid2Op's usual single-substation operational limit. Counts exclude combinations between topology and line-status actions.

Grid2Op studies therefore rarely expose the full combinatorial control space directly to a policy. The two roles appear explicitly in the control flow:

$$s_t \to g_{\tau(s_t)}\{0 \to a_{\text{base}}, 1 \to \mathcal{A}_{\text{family}} \to \mathcal{A}_{\text{local}}(s_t) \to \mathcal{A}_{\text{ranked}}(s_t) \to \mathcal{A}_{\text{admit}}(s_t) \to \mathcal{A}_t \to \pi_{\theta(\cdot \mid s_t, \mathcal{A}_t)}\} \to a_t \tag{10}$$

Work	Before filtering		Conditional action filter	Decision
Work	**Control scope**	**Intervention gate**	**Candidate-action construction**	**Policy interface**
(Yoon et al., 2021)	Topology	Hazard states	Scope: Goal topology Check: Low-level realization	SMAAC **Returns** Goal-topology choice
(Sar et al., 2023)	Topology	—	Scope: Local configurations Scope: Deduplication Locate: Substation Check: Legality mask	Hierarchical MARL **Returns** Configs at one substation
(Liu et al., 2024)	Topology	Risk threshold	Scope: Local configurations Locate: Substation Check: Legality mask	Progressive policy **Returns** One local configuration
(Xu et al., 2020)	Topology	—	Scope: Fixed candidate library Check: Power-flow simulation	Q ranking **Returns** Reduced topology library

Work	Before filtering		Conditional action filter	Decision
Work	**Control scope**	**Intervention gate**	**Candidate-action construction**	**Policy interface**
(Taha et al., 2022)	Topology	—	Scope: Unitary actions Check: Learned surrogate	MCTS **Returns** Dynamic search frontier
(Xu et al., 2022)	Topology Redispatch	—	Scope: Sub-action spaces Locate: Action subspace Check: Search	DDQN MCTS **Returns** One specialist subspace
(Zhou et al., 2021)	Topology Redispatch	—	Scope: Action set Check: Power-flow simulation	Set-based policy **Returns** Small state-conditioned set
(Marot et al., 2021)	Topology Redispatch	Risk threshold	Scope: Unitary library Check: Power-flow simulation	Rank / rules **Returns** ~10^3 vs. >7×10^4 raw
(Lehna et al., 2023)	Topology	Risk threshold	Scope: Unitary actions Scope: Curated library Check: Power-flow simulation	Priority + fallback **Returns** Small curated library
(Manczak et al., 2023)	Topology	Risk threshold	Scope: Local configurations Locate: Substation Check: Greedy evaluation	Hierarchical RL **Returns** Configs at selected substation
(Zhao et al., 2022)	Restricted families	—	Scope: Reduced variants Locate: Action family Check: Environment constraints	PPO **Returns** Task-specific reduced space
(Moradi et al., 2024)	Multiple families	—	Scope: Separate specialists Locate: Action family Check: Legality mask	Family selector **Returns** One specialist space
(Tang et al., 2026)	Topology	Risk threshold	Scope: Controllable elements Locate: Substation Check: Legality mask	RL policy **Returns** Legal topology candidates
Graph-local sampler (ours/ current)	Topology	rho threshold	Scope: Local configurations Locate: Critical-line k-hop neighborhood Check: Bounded legality Check: Offline simulation diagnostics	Masked PPO logits **Returns** legal local topology candidates

The literature supports a consistent conceptual ordering. First, an intervention gate separates **when to act** from **how to correct**. Conditional on activation, the action pipeline restricts the action family (bus topology, power-line set, re-dispatch, etc.), then may localize around stressed equipment, order actions on a criteria, apply simulator legality, or apply diversity or cardinality limits. Heavier masking variants include one-step physics ranking, learned surrogates, and large searches.

We recommend future works report **intervention gate** and **candidate-action construction** as separate columns, as this decomposition can help with cross author comparison. A stronger action library or simulation-based admission stage can significantly affect model effectiveness over a change in model architecture. Our current case118 implementation is: topology-family restriction; critical-neighborhood localization; locality-based candidate ordering; legality check; diversity balancing across substations; and a final 128 sub-sampler. Ideally legality would be applied earlier in the chain, but this is an expensive one-step operation. With case 118 having over 70,000 topology actions, this would mean simulating every action for legality within a single episode. Though caching could potentially help, the grid state could change and so would potential legal moves.

The implemented policies use a fixed Grid2Op action catalog and a boolean mask over action IDs. A future candidate-policy variant could instead encode a local topology action by its structural delta ΔG_a, score each candidate jointly with the state, $\ell_a =$

$\text{Policy}_{\theta(h_G,\ \text{ActionEncoder}(\Delta G_a))}$, and normalize logits only over A_t^K. That variant would avoid a fixed output layer tied to one environment's categorical action IDs.

D Experimental Details for Case14 and Case118

This appendix describes the training and evaluation protocol used for the Case14 and Case118 experiments. All results use the same train/test split across model representations and random seeds within a grid case. Reported values are computed over five random seeds (`0--4`) for each of the three policy representations: Flat MLP, Operating GNN, and GraphIR.

D.1 Tasks and Grid Partitions

Grid	Control task	Reward	Action set	Train/test partition
Case14	Intervention-recovery	PoWRL threshold reward	Unary line-set and topology-set actions	Deterministic chronic split, seed 0; 90% train, 10% held-out test
Case118	Intervention-recovery	PoWRL threshold reward	Topology-set actions only	Deterministic chronic split, seed 0; 90% train, 10% held-out test

For intervention-recovery runs, a decision state is any state with maximum line loading at least `rho = 0.95`; unsafe-exit tests start from states with maximum line loading at least `rho = 1.0`. The intervention index is built separately for the training and test splits so that training never samples held-out test chronics. Case118 repeats each indexed training start four times to improve sample efficiency on the larger grid.

D.2 Model Variants

Model	Case14 width / heads	Case118 width / heads	Inputs and inductive bias
Flat MLP	64 / 2	64 / 2	Dense observation encoder without explicit graph message passing
Operating GNN	48 / 1	48 / 1	Graph policy over the operating electrical topology
GraphIR	48 / 1	48 / 1	Graph policy using the GraphIR physical and operating relational schema

All models use actor-critic PPO with a categorical action head over the available action catalog after any task-specific masking. The reported GraphIR runs use the uncorrupted `correct` GraphIR variant.

D.3 PPO Training Hyperparameters

Grid	Frames	Batch	Epochs	Mini-batch	LR	Gamma / lambda	Clip	Entropy / critic	Learner
Case14	1.05M	8192	3	1024	3e-4	0.99 / 0.95	0.2	0.01 / 1.0	1 PPO learner
Case118	2.10M	4096	6	2048	3e-4	0.99 / 0.95	0.2	0.01 / 1.0	8 CPU-DDP learners

All experiments use advantage normalization and gradient clipping at norm `0.5`. Training uses 32 parallel environment workers, CPU execution, and one Torch thread per learner. Checkpoints are selected at the final training step for the primary comparisons.

D.4 Action Filtering and Controllers

Suite	Controller	Action availability	Selection rule
Case14 intervention	Learned actor-critic intervention policy triggered at `rho >= 0.95`	Full configured catalog: line-set and topology-set unary actions	Sample from policy during training; learned/no-op/heuristic policies evaluated

Suite	Controller	Action availability	Selection rule
Case118 intervention	Learned actor-critic intervention policy triggered at `rho >= 0.95`	Topology-set actions masked to critical-line neighborhoods; legal candidates capped at 128 from a pool of 256	Masked logits during training; learned argmax for the final unsafe-exit test

Case118 uses action masking to make the large topology action catalog tractable. The mask first restricts to topology actions, then keeps actions within one hop of critical lines above `rho = 0.95`, and finally applies exact legality and simulation validation under a fixed candidate budget. If no candidate survives, the controller falls back to do-nothing.

D.5 Compute and Training Time

Grid	Model	Seeds	Batches	Wall-clock time per seed	Throughput
Case14	Flat MLP	5	128	4.70 h (4.27–5.03)	62.2 decisions/s
Case14	Operating GNN	5	128	4.76 h (4.42–5.20)	61.4 decisions/s
Case14	GraphIR	5	128	5.09 h (4.62–5.90)	57.6 decisions/s
Case118	Flat MLP	5	512	22.55 h (19.69–23.34)	26.0 decisions/s
Case118	Operating GNN	5	512	24.20 h (21.17–25.05)	24.2 decisions/s
Case118	GraphIR	5	512	27.17 h (23.71–28.22)	21.5 decisions/s

D.6 Evaluation Protocols

Suite	Protocol	Trials	Decision cap	Primary criteria
Case14	Unsafe-exit	100 indexed unsafe starts	1 action	Exit@H from rho >= 1.0 starts; illegal and terminal actions reported separately
Case118	Unsafe-exit	1000 indexed unsafe starts per checkpoint	1 action	Exit@H, action validity, terminal rate, rho relief, sampler candidate counts, and rejection diagnostics

The unsafe-exit protocol tests whether a single learned intervention can move the grid from an unsafe state into a state that recovers within the evaluation horizon. Illegal or terminal actions count as failures.

E Additional Visualizations

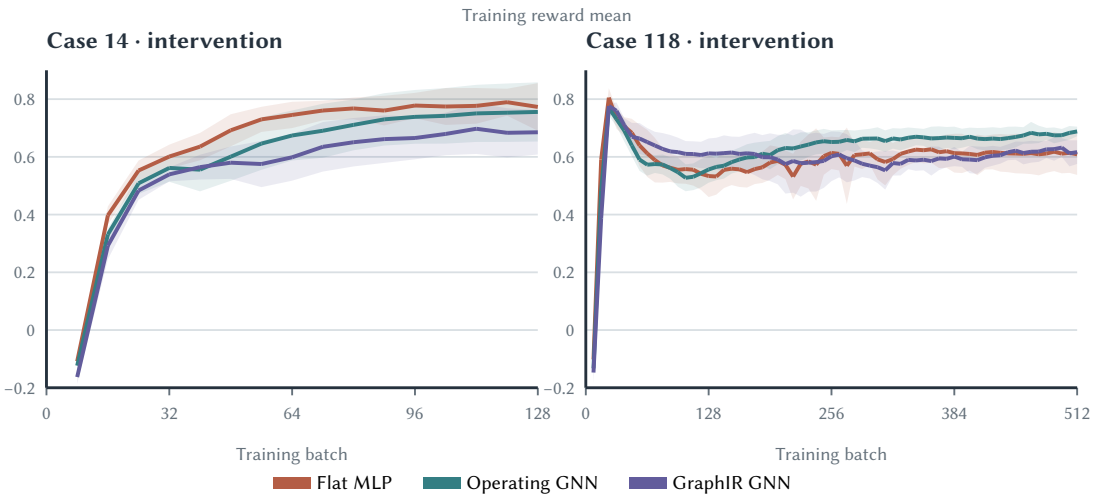
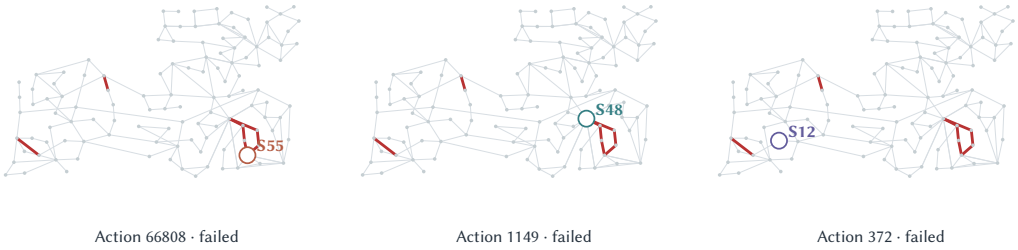
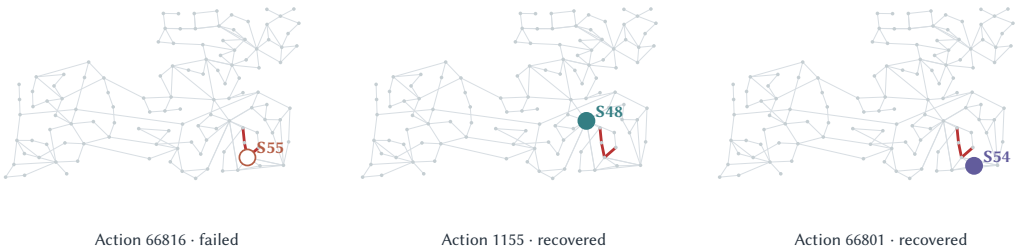


Figure 6: Training reward over 128 (case 14) and 512 (case 118) batches for the intervention task. Lines are means across five seeds; shaded bands show ±1 standard deviation.

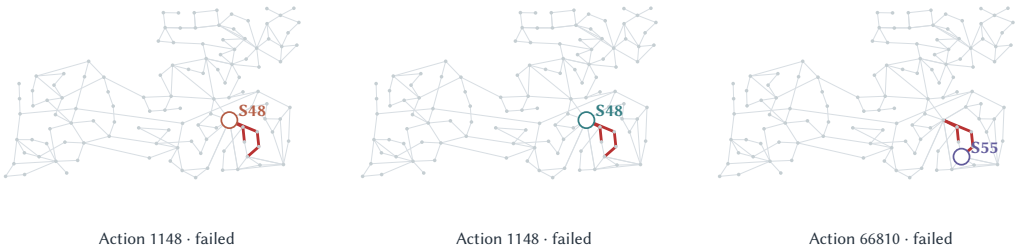
Eight overloads; all fail · seed4__chronic4__t29 · initial max $\rho = 1.352$ · 8 overloaded lines



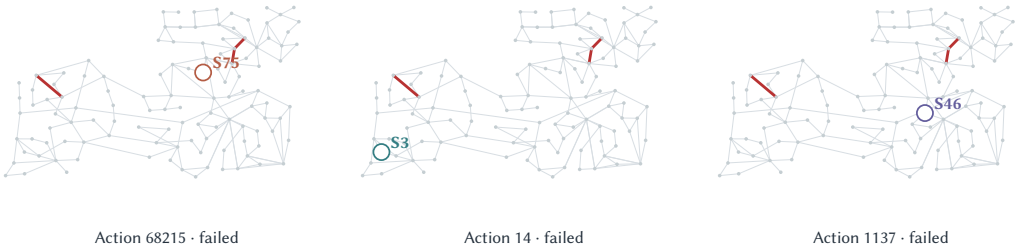
Both GNNs recover · seed3__chronic39__t139 · initial max $\rho = 1.472$ · 3 overloaded lines



Five overloads; all fail · seed3__chronic72__t29 · initial max $\rho = 1.188$ · 5 overloaded lines



Three overloads; all fail · seed4__chronic73__t113 · initial max $\rho = 1.324$ · 3 overloaded lines



Same substation, different outcomes · seed3__chronic43__t139 · initial max $\rho = 1.917$ · 4 overloaded lines



Red branches = initially overloaded lines; colored circle = selected substation. Filled circle = horizon recovery; hollow circle = failure.

Figure 7: Five illustrative case 118 states on a network layout. Rows contrast three, five, and eight initial overloads, recovery by both GNNs at different action locations, and different outcomes despite choosing the same substation. Red branches show the overloaded pre-action lines; circles show each model's selected substation. Outcomes are measured over the evaluation horizon. These examples are diagnostic illustrations only and not estimates of outcome frequency. Note that agents can pick the same substation, but a different bus-topology augmentation, which can result in different outcomes.