Scalable Tree Search over Graphs with Learned Action Pruning for Power Grid Control

Florence Cloutier[1, 2], **Cyrus Neary** [1, 2], **Adriana Hugessen**[1, 2], **Viktor Todosijević**[2], **Zina Kamel**[3], **Glen Berseth**[1, 2]
{florence.cloutier, cyrus.neary, adriana.knatchbull-hugessen, zina.kamel, glen.berseth}@mila.quebec, todosijevicviktor998@gmail.com

[1]**Université de Montréal** [2]**Mila - Quebec AI Institute** [3]**McGill University**

Abstract

As real-world infrastructure systems become increasingly complex and large-scale, there is a growing need for learning-based control strategies that can make informed decisions in complex and dynamic environments. However, large-scale problems — such as power grid control — introduce high-dimensional action spaces and necessitate transferability across varying grid topologies. We introduce **H**ierarchical **E**xpert-Guided **R**econfiguration **O**ptimization for **G**raph **T**opologies, **HERO-GT**, a model-based planning approach that combines a pretrained graph neural network (GNN) for topology-aware action pruning with a Monte Carlo Tree Search (MCTS) planner for targeted, structured exploration. More specifically, the high-level GNN predicts a promising subset of actions, which the low-level MCTS agent uses to focus its search and reduce computational overhead while remaining adaptable to unseen graph structures. Furthermore, the MCTS planner leverages a given *default policy*—which may be defined, for example, by heuristics, problem relaxations, or rule-based methods—to bias the search and prioritize actions that are expected to improve performance over the default. We deploy HERO-GT in power grid environments, demonstrating that it not only improves over a strong default policy, but also scales to a realistic operational setting where exhaustive search becomes computationally infeasible.

1 Introduction

Owing to their ability to learn performant policies that optimize complex objectives directly from interaction data, deep reinforcement learning (RL) algorithms offer tremendous promise for tackling real-world challenges in infrastructure systems such as power grids, traffic networks, and supply chains. However, the scale, complexity, and safety-critical nature of such real-world systems introduce significant practical challenges that prohibit the direct application of existing RL algorithms. For example, controlling power grid line connections necessitates learning over a massive action space that grows exponentially in the grid's size. Moreover, the complex, distributed nature of many large-scale infrastructure systems results in diverse, dynamically varying state and observation spaces. Finally, policies must be reliable; they should, for example, perform at least as well as pre-existing approaches to solving the problem at hand.

To overcome these challenges, algorithms must handle high-dimensional action spaces while also improving the performance of any prior systems. To address high-dimensional action spaces, many planning methods adopt hierarchical control strategies that segment the overall action or exploration space. To ensure monotonic improvement over prior systems, one can apply search starting from a default policy: when initialized with a plan generated by a prior method, the search should only improve upon that plan. We accordingly present *Hierarchical Expert-guided Reconfiguration Op-*

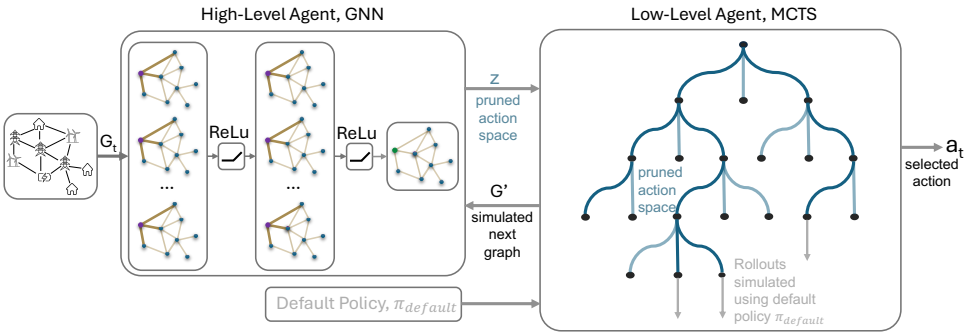

Figure 1: An overview of the proposed hierarchical expert-guided reconfiguration optimization for graph topologies (HERO-GT) algorithm.

timization for Graph Topologies (HERO-GT)—a novel algorithm for sequential decision-making on graphs. Figure 1 illustrates the proposed approach. HERO-GT simplifies exploration in large combinatorial action spaces by using a pre-trained *high-level* agent to propose a set of promising candidate actions at each timestep. A *low-level* agent is then tasked with selecting the optimal action from this reduced set. To support transferability across varying graph topologies and scales, the high-level agent uses a graph neural network (GNN), while the low-level agent operates as a Monte Carlo tree search (MCTS) algorithm that is agnostic to graph structure. To ensure HERO-GT performs at least as well as a given *default policy*—defined, for instance, by heuristics, problem relaxations, rule-based methods, or human intuition—the low-level agent's search procedure is designed to incorporate a default policy as prior knowledge and act as a policy improvement operator over it.

While the HERO-GT framework may be applied generally to a wide range of sequential decision problems on graphs, in this work, we focus primarily on controlling power grid connectivity. More specifically, we consider a sequential decision-making problem in which the decision-making agent may change the connection structure of a power grid at each timestep during operation. The agent's objective is to minimize operational costs and maintain grid reliability under adversarial conditions, such as line faults and demand peaks, without overloading the system. Novel approaches, such as HERO-GT, are necessary to enhance power grid planning, reliability, resilience, and security while accelerating the transition to low-carbon energy sources.

We demonstrate HERO-GT's capabilities through simulated experiments on the Grid2Op platform (Donnot, 2020)—a power grid simulation suite that provides realistic benchmarks for developing AI-based grid management strategies. We observe that even on relatively small grids, HERO-GT achieves higher rewards and improved grid stability in comparison with a proximal policy optimization (PPO) baseline (Schulman et al., 2017), while matching or exceeding the performance of both the default policy and a standard MCTS algorithm that explores the full action space without pruning. Furthermore, by using the high-level agent to restrict the action space, HERO-GT maintains strong performance even on a larger grid where vanilla MCTS becomes computationally intractable.

2 Related Work

Graph-structured environments present unique challenges for planning algorithms due to the environment's complex combinatorics and topology. Recent surveys such as Nie et al. (2023) highlight the growing interest in applying RL to graph domains. Several works leverage RL for graph generation in the context of molecule generation (You et al., 2018), neural architecture search Rupp & Eckert (2024), communication networks (Darvariu et al., 2021), and transit network design (Holliday et al., 2024; Holliday, 2025). While these approaches share the objective of optimizing graph topology, they operate in a generative setting, whereas our method addresses graph reconfiguration, modifying an existing graph in a constrained, dynamic environment. A closer line of work involves

graph rewiring (Peng et al., 2024), which uses RL to adjust graph connectivity to improve GNN performance in learning tasks. In contrast, our method's focus is on sequential decision-making tasks, where topological changes directly impact behaviour and long-term performance of real-world systems. In robotics control, works such as AnyMorph (Trabucco et al., 2022), MetaMorph (Gupta et al., 2022) and One Policy to Control Them All (Huang et al., 2020) aim to train controllers that generalize across a wide variety of robot morphologies. While they address challenges inherent to morphologically diverse agents, they typically focus on policy transfer across agents, whereas our method also aims to deal with combinatorial action spaces, common in large infrastructure settings such as power grids.

RL has been increasingly applied to power system control tasks such as topology optimization. In the L2RPN benchmark (Marot et al., 2021), agents learn to manage grids under faults and demands shifts. Recent work leverages GNNs to capture grid structure and improve generalization (de Jong et al., 2025). Dorfer et al. (2022) use AlphaZero with MCTS and a reduced, greedily defined action space for congestion management, which may limit flexibility and optimality. In contrast, we learn a policy to dynamically select relevant actions based on the current grid state, enabling more adaptive exploration. Prior work like van der Sar et al. (2023) applies hierarchical multi-agent RL to power grid topology optimization, assigning agents to substations. While they also highlight the benefits of hierarchical control, their experiments are limited to a small grid with 5 substations, whereas our work addresses transferring performance to a larger grid.

3 Preliminaries

A **Markov Decision Process** (MDP) is a tuple $\mathcal{M} = (S, \Delta s_0, A, T, R)$. Here, S denotes the MDP's set of states, Δs_0 is a distribution over initial states, A denotes its set of actions, $T(s'|s, a)$ is the probability of transitioning from state $s \in S$ to $s' \in S$ under action $a \in A$, and $R(s, a, s') \in \mathbb{R}$ is the associated scalar reward. A policy $\pi(a|s)$ is a function that maps states s to probability distributions over actions a. We consider finite-horizon decision-making problems, in which the objective is to find a policy $\pi(a|s)$ that maximizes the expected sum of cumulative rewards $\mathbb{E}[\sum_{t=1}^{H} R(s_t, a_t, s_{t+1})]$, for some finite time horizon H.

Graph Neural Networks (GNNs) are a class of neural architectures designed to operate directly on graph-structured data (Zhou et al., 2020). The input to a GNN is a graph $G = (V, E, \{\phi_v \mid v \in V\})$, with nodes V, edges $E \subseteq V \times V$, and feature vectors $\phi_v \in \mathbb{R}^{d_v}$ encoding attributes for each node $v \in V$. The GNN embeds these features into latent representations $h_v \in \mathbb{R}^{d_h}$, which are then iteratively updated through a *message passing* process. At every layer l of the GNN inference procedure, each node's latent features h_v^l are updated through two distinct steps: 1) message aggregation, in which information is collected on the latent features $h_{v'}^l$ of all neighboring nodes $v' \in N(v)$, and 2) node updating, in which the collected information is processed along with the current latent feature to obtain the node's latent feature h_v^{l+1} of the next GNN layer. That is, at each layer l, each node v aggregates information from its neighbors $N(v)$ and updates its embedding according to

$$m_v^{(l)} = \text{AGGREGATE}^{(l)}\left(\{h_u^{(l-1)} : u \in N(v)\}\right), \quad h_v^{(l)} = \text{UPDATE}^{(l)}\left(h_v^{(l-1)}, m_v^{(l)}\right), \tag{1}$$

where $\text{AGGREGATE}^{(l)}(\cdot)$ is a permutation invariant function (e.g., convolutions in the case of Graph Convolutional Networks (Kipf & Welling, 2017)) and $\text{UPDATE}^{(l)}(\cdot)$ is a learnable transformation such as a neural network. Through multiple message-passing layers, the latent node features can capture increasingly rich, multihop neighborhood information. A key property of GNNs is their ability to leverage the inductive biases specific to graph topologies—such as locality, permutation invariance, and relational inductive biases—in order to generalize across graphs of different sizes and structures. This property makes GNNs well-suited for tasks where input graphs vary significantly between training and deployment, such as for the power grid problems studied in this work.

Monte Carlo Tree Search (MCTS) is a sequential decision-making algorithm that searches for optimal actions at every timestep by iteratively constructing a search tree via the following four

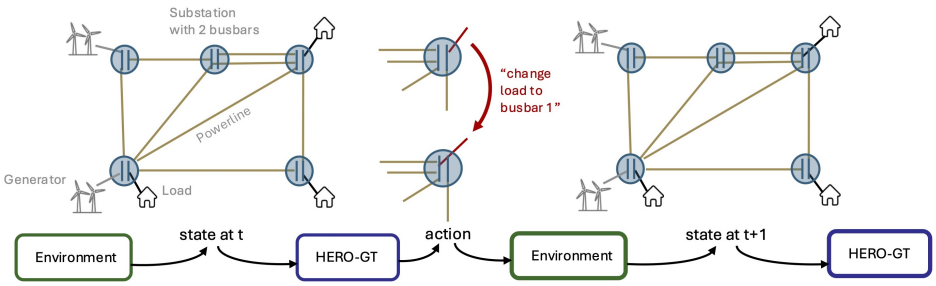

Figure 2: An illustration of a sequential decision-making problem on power grids. Taking a topological action ("change load to busbar 1") on a power grid with 5 substations (blue circles) and 2 busbars (parallel blue vertical lines) per substation results in updated connectivity and power flow.

steps: expansion, selection, simulation, and backpropagation (Świechowski et al., 2023). Each node u of the search tree corresponds to a candidate environment state s, and stores data on the search, such as the number of visits $N(u)$ to each node, the number of times $N(u, a)$ an action has been taken from each node, and estimates of node-action values $Q(u, a)$. During the *selection* phase, the agent traverses the tree from the root to a leaf node by choosing actions that balance exploration and exploitation. It does so by selecting an action a from a node u using a selection criterion, such as the Upper Confidence Bound (UCB) in Equation (2), and moving to the corresponding child node.

$$a^* = \arg\max_a UCB(u, a) = \arg\max_a [Q(u, a) + c\sqrt{\frac{\ln N(u)}{N(u, a)}}] \tag{2}$$

Once a leaf node is reached, the *expansion* phase adds a child node to it by simulating the result of taking an unexplored action from the corresponding environment state. In the *simulation* phase, a so-called rollout policy $\pi_{rollout}$ is used to simulate a trajectory from a newly added child node for a fixed time horizon, or until a terminal node is reached. The rewards of these trajectories are then propagated up the tree in the *backup* phase, updating the value estimates for each of the visited nodes along the trajectory, typically by averaging returns and incrementing visit counts. After a user-specified number of iterations over these four steps, MCTS returns the action at the root node with the highest visit count.

4 Hierarchical Expert-Guided Reconfiguration Optimization for Graph Topologies

We consider a general class of sequential decision-making problems on graphs, which we define in §4.1. To address the large and complex state and action spaces associated with this class of problems, we then introduce HERO-GT, a two-stage framework for efficient and graph-agnostic decision-making. Intuitively, the method splits the decision process into a high-level action pruning stage (§4.2) and a low-level action selection stage (§4.3), in which the high-level agent significantly simplifies the action space that the low-level agent is required to explore. We instantiate this framework by implementing it on a representative problem in power grid control (§4.4).

4.1 Planning on Graphs

We model decision-making problems on graphs as an MDP $\mathcal{M} = (S, \Delta s_0, A, T, R)$. More specifically, we consider the underlying graph to have a fixed set of nodes V, and we define each MDP state $s \in S$ to be given by a graph $G = (V, E, \{\phi_v | v \in V\})$, with edges E and feature vectors $\phi_v \in \mathbb{R}^{d_v}$. We consider actions $a \in A$ to correspond to direct modifications to the graph itself, which may generally include adding, removing, or altering edges, or updating node features directly. In this

work, however, we restrict our focus to discrete changes in graph connectivity via modifications to the edge set E. The transition function $T(s' \mid s, a)$ encodes the dynamics of the environment, returning a new graph $s' = G' = (V, E', \{\phi'_v | v \in V\})$ with updated edges E' and node features ϕ'_v in response to the action a taken from the previous state s. The reward function $R(s, a, s')$ is defined over states and actions, reflecting task-specific objectives such as minimizing cost or maintaining structural properties of the graph. In this work, we assume that the graph is fully observable to the agent at each timestep. That is, at every time t, the agent may use all information relating to the graph's edges and node features to select an action a.

As an illustrative example, consider the problem of altering the connection structure of a power grid, as is described in the introduction and depicted in Figure 2. In this setting, the nodes $v \in V$ correspond to all of the buses, loads, generators, and transmission line endpoints in the grid, while the edges $E \subseteq V \times V$ describe how those elements are connected at the current timestep. Typically, loads, generators, and transmission line endpoints are connected to buses to enable power flow and system operation. Meanwhile, the node features ϕ_v encode the relevant electrical and structural properties in the grid, such as bus voltage magnitudes and angles, active and reactive power injections, line thermal limits, and generator setpoints. At time t, the agent uses information from the graph G_t to select an action $a_t \in A$ corresponding to the addition, removal, or reconfiguration of an edge in the graph—modeling the connection, disconnection, or switching of loads, generators, or power lines between buses within a power grid substation. As a result of this change to the graph's connectivity, the power flow within the grid will also change accordingly, resulting in updated node features and the graph G_{t+1} modeling the state at the next timestep.

4.2 High-Level Agent

In the general case where actions represent arbitrary re-specifications of the edge set E, the number of possible actions grows exponentially in the number of nodes. This large scale of the action space renders exploration challenging. We accordingly train the high-level agent to take the full graph G_t as input, and to output a candidate subset of actions that is likely to contain the optimal action a_t^*.

More formally, we assume that a problem-specific decomposition of the action space is available in the form of a partition P of the action space A. That is $A = \bigcup_{Z \in P} Z$, with $Z \subseteq A$, $Z \neq \emptyset$, and $Z \cap Z' = \emptyset$, for all $Z, Z' \in P$. In many applications, such a partition arises naturally from the problem structure. In the running power grid scenario, for example, we define each subset $Z \in P$ of the partition to contain all edge modifications occurring at buses within a specific substation. We also assume access to a behavioral cloning dataset $\mathcal{D}$ containing tuples of the form (G_t, a_t^*), where a_t^* denotes the optimal action taken at graph G_t. Such a dataset may be constructed from historical operation data or through simulation on small-scale graphs where it is feasible to exhaustively identify the action with the highest greedy reward. We then train the high-level policy π_{high} as a classifier that maps the current graph G_t to the subset $Z_{a_t^*} \in P$ containing the optimal action a_t^*—i.e., $\pi_{high}(G_t) = Z_{a_t^*}$.

The high-level policy π_{high} is implemented as a GNN. The GNN architecture supports heterogeneous graphs: node types are embedded using dedicated MLP encoders, followed by message-passing layers. This training approach enables π_{high} to transfer across different grid topologies by learning spatial and relational dependencies through the GNN architecture.

4.3 Low-Level Agent

The low-level agent's objective is to select the optimal action a_t^* at each timestep. We implement the low-level policy as a MCTS algorithm that uses the high-level policy π_{high} and a default policy $\pi_{default}$ to inform the *expansion* and *simulation* phases of the search, respectively. The process is illustrated in Figure 1. At every timestep t, the low-level agent instantiates a new search tree beginning with a root node containing information on the current state s_t. It then proceeds with MCTS from this root node, as described in §3, to obtain the action a_t to execute in the environment.

However, during each expansion phase of the MCTS, the low-level agent queries π_{high} with the graph G at the corresponding leaf node to obtain a set of candidate actions $Z = \pi_{high}(G)$. It then only uses actions a from this subset $Z \subseteq A$ to expand child nodes from that leaf. Intuitively, because we train π_{high} to output the subset Z containing the optimal action, this step significantly reduces the tree's branching factor without pruning optimal actions from consideration. We also note that this reduction to the action space is related to the sampling-based reduction of large and complex action spaces proposed by Hubert et al. (2021).

Meanwhile, during the simulation phase of the MCTS, the agent uses $\pi_{default}$, instead of a uniform random policy, to compute rollout-based estimates of the values of newly expanded child nodes. By using $\pi_{default}$ to evaluate newly expanded nodes, the search procedure ensures that value estimates reflect the default policy's performance. This enables the low-level agent to identify higher-return actions relative to $\pi_{default}$, effectively refining upon $\pi_{default}$, given sufficient search.

4.4 An Instantiation of HERO-GT for Power Grid Control

We apply HERO-GT to topological control over power networks, where the agent reconfigures grid connections at each timestep to mitigate faults, avoid overloads, and maintain reliability. The environment is modeled using Grid2Op (Donnot, 2020), which simulates realistic constraints, stochastic events and cost-based rewards.

Each state G_t is a dynamic graph with nodes V (e.g. generators, loads, powerline extremities) and edges E capture their connectivity via shared busbars in substations. Actions $a \in A$ correspond to modifying the connection status between these components and the busbars they are assigned to, reconfiguring the grid topology. With multiple busbars per substation and many switchable components, the action space grows exponentially, motivating action-restriction strategies like HERO-GT.

We train the high-level agent via behavior cloning on a small grid (14 substations and 57 nodes). Supervision labels are generated using the greedy search from de Jong et al. (2024), which selects the actionsimulates all available topological actions $a \in A$ at each timestep and selects $a_t^* \in A$, that minimizes maximum line loading at each timestep. The substation affect by a_t^* is used as the training label. In power grids, a substation groups components(e.g. transmission lines, generators, loads), connected by two busbars. The GNN-based high-level agent performs node-level classification to identify the most relevant substation for intervention, reducing the action space to those within that substation, i.e. $\pi_{high}(G_t) = Z_{a_t^*}$ from §4.2. This greedy labeling approach is tractable only for small grids because we are exhaustively simulating all possible actions, but we note that the trained agent is designed to ensure scalability to larger topologies without needing additional supervision.

At inference time, the predicted substation is passed to the low-level agent, restricting the action space to $Z \subseteq A$, i.e., actions associated with that substation §4.3. To estimate leaf node values during MCTS, we use rollouts with a default policy $\pi_{default}$, defined as the no-operation policy that keeps the current topology unchanged. This reflects a key intuition in power grid operation that under standard conditions, the system remains stable and interventions are required only in response to contingencies. If Z remains too large for efficient MCTS, the low-level agent falls back on $\pi_{default}$, ensuring an action can still be selected, even under limited computational resources.

5 Experiments

We design experiments to evaluate whether HERO-GT can improve performance over a default policy by incorporating MCTS while still retaining scalability to a larger environment in which MCTS is computationally infeasible.

Environment We evaluate our algorithm in the Learning to Run a Power Network (L2RPN) Challenge environment (Dorfer et al., 2022), built on the Grid2Op platform (Donnot, 2020), which provides a realistic RL environment for power grid management. Agents must maintain grid reliability under adversarial conditions, such as line faults and demand peaks, without overloading the system.

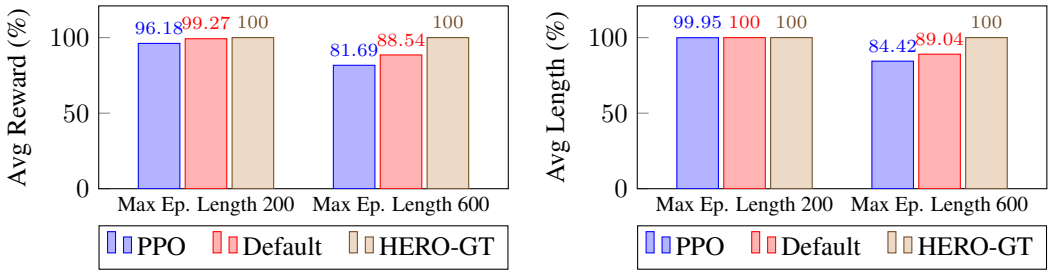

Figure 3: Performance of all methods on single held-out test scenario in `Small Grid A`. (Left) Average cumulative reward on the power grid connectivity control problem as a percentage of the reward achieved by an MCTS algorithm that expands all possible actions. (Right) Average episode length as a percentage of the maximum episode time horizon.

Simulations in Grid2Op can be varied across (1) number of substations (2) initial edge configurations (3) fixed scenarios for historical load and generation, which are needed to simulate realistic operation of the grid over finite horizons. We perform evaluation over two grid sizes (1) `Small Grid` which contains 14 substations (with a number of nodes |V| = 57), and (2) `Large Grid` which contains 36 substations (with |V| = 177). We consider two initial edge configurations for both `Small Grid` and `Large Grid`, which we refer to as `Small Grid A` and `B` and `Large Grid A` and `B`, (see Appendix A for the corresponding Grid2Op graph names). For each initial edge configuration, Grid2Op provides multiple load and generation profiles, three in the `Small Grid` and ten in the `Large Grid`.

Baselines We conduct experiments to evaluate how our method **HERO-GT** is able to improve over a **Default** policy. In this case, the **Default** policy is set to a no-operation agent. Surprisingly, as we demonstrate, this no-op agent provides a reasonably good default policy for power grids, which are generally operational without intervention under normal conditions. In the `Small Grid` environment, we report performance as a percentage of standard **MCTS** without the high-level agent action-pruning, which operates as an upper bound on the performance of our method. In the larger environment `Large Grid`, MCTS is infeasible to run, so we report raw results versus the **Default**. Finally, we compare against **PPO**, a standard RL algorithm.

Evaluation Metrics We evaluate **Default**, **MCTS** and **HERO-GT** on the 3 scenarios in `Small Grid` and 10 in `Large Grid`. As both methods and environments are deterministic, we run a single evaluation per scenario. We report average episode length (grid stability) and average reward (percentage below maximum thermal load on each transmission line). Longer episodes reflect more stable control while higher rewards demonstrate a measure of overall control performance. In all experiments, we evaluate over two episode lengths,

	Ep. Length		Reward	
Small Grid A		**% MCTS**		**% MCTS**
T=200				
HERO-GT	200	100	**3382.38**	**100**
Default	200	100	3357.76	99
T=600				
HERO-GT	**575**	**100**	**9497.38**	**100**
Default	512	89	8409.38	89
Small Grid B				
T=200				
HERO-GT	200	100	**3419.56**	**100**
Default	200	100	3309.66	97
T=600				
HERO-GT	575	100	**9516.16**	**100**
Default	**600**	**104**	9221.92	97
Large Grid A				
T=200				
HERO-GT	200	-	10231.32	-
Default	200	-	10231.32	-
T=600				
HERO-GT	**600**	-	**30477.12**	-
Default	568	-	28822.24	-
Large Grid B				
T=200				
HERO-GT	185.7	-	9847.016	-
Default	185.7	-	9847.016	-
T=600				
HERO-GT	416.3	-	21644.09	-
Default	416.3	-	21644.09	-

Table 1: Performance of **HERO-GT** and **Default** policy on all grid topologies. Results are averaged over all evaluation episodes: three load/generation scenarios for the `Small Grid` and ten for the `Large Grid`. Best results are shown in **bold**. **HERO-GT** matches or improves the **Default** performance across all topologies and horizons.

200 and 600. Except for in Figure 3, we report all results as an average over the available load/generation scenarios. In Figure 3, to compare against **PPO** which requires historical scenarios for training, we select two scenarios for training and one holdout scenario for testing. We provide results on this single scenario across all methods for a fair comparison. Since the PPO algorithm is stochastic, we train over three seeds and evaluate over 100 iterations and average the result. All training hyperparameters can be found in Appendix A

HERO-GT Improves Performance over a Strong Default A core assumption of our method is that, for certain applications like power grids, a strong default policy exists that can be relied upon to provide default actions on the nodes not selected by the high-level policy. In Figure 3, we can see that indeed the **Default** no-operation method does provide a strong baseline for **HERO-GT**. In particular, it is not trivial to beat this baseline, as can be observed by the lower performance of **PPO** which is a strong general-purpose RL method.

Provided with a strong **Default** control policy, we see in Table 1 that our method **HERO-GT** is able to consistently improve over **Default** both in terms of episode length and total rewards, particularly over the longer episode horizon. This suggests that actively exploring topological actions, even when significantly pruning the action space, is able to provide benefit over strong **Default** policies. In fact, we see in Table 1 that **HERO-GT** is able to match the performance of **MCTS** in both `Small Grid` environments, even with significantly less computation, as we discuss next.

HERO-GT allows MCTS to scale **MCTS** is infeasible in `Large Grid` due to the large search space. Without action pruning, expanding all candidate actions (in this case over 65 000 actions) becomes extremely expensive, with each tree expansion step requiring evaluation of thousands of possibilities. In contrast, by pruning actions to be restricted to within the high-level agent's predicted substation, **HERO-GT**, significantly reduces the effective branching factor of MCTS. In `Small Grid` the number of nodes expanded in **HERO-GT** is only 7.1% of the nodes expanded by **MCTS**. In the `Large Grid` this percentage drops to only 2.8%. This enables scalable decision-making even on larger grids. Moreover, HERO-GT still offers performance enhancement over **Default** while remaining computationally efficient, as can be seen in the results in Table 1. This is due to its focused search guided by the high-level policy and structural transferability from the GNN.

6 Conclusion

In this work, we proposed **HERO-GT** (**H**ierarchical **E**xpert-Guided **R**econfiguration **O**ptimization for **G**raph **T**opologies), a model-based sequential decision making framework designed to address two key challenges in real-world control tasks such as power grid topology optimization: **navigating large combinatorial action spaces** and **transferability across different graph topologies**. By combining a pretrained graph neural network (GNN) for structure aware action subset prediction with a Monte Carlo Tree Search (MCTS) planner for targeted search, HERO-GT enables more efficient sequential decision-making by reducing the search burden compared to conventional MCTS, while maintaining adaptability across diverse environments. Simulated experiments in power grid environments show that HERO-GT consistently improves cumulative reward over the baseline policies used for comparison. Furthermore, the results empirically show that HERO-GT scales to a larger and more complex environment than the one used for training.

Future work will integrate policy and value networks into HERO-GT to learn from search-generated data. While HERO-GT currently reduces the action space by operating at the substation level, it relies on a default policy when that subset is still too large. A finer node-level decomposition could improve efficiency in such cases.

Overall, HERO-GT is a promising, practical, scalable framework for real-world systems, offering valuable insights into real-world deployment of hierarchical, graph-informed planning, such as power grid operation.

Acknowledgments and Disclosure of Funding

We want to acknowledge funding support from NSERC and FRQNT, as well as compute support from Mila IDT.

References

Victor-Alexandru Darvariu, Stephen Hailes, and Mirco Musolesi. Goal-directed graph construction using reinforcement learning. *Proceedings of the Royal Society A*, 477(2254):20210168, 2021.

Matthijs de Jong, Jan Viebahn, and Yuliya Shapovalova. Imitation learning for intra-day power grid operation through topology actions. *arXiv preprint arXiv:2407.19865*, 2024.

Matthijs de Jong, Jan Viebahn, and Yuliya Shapovalova. Generalizable graph neural networks for robust power grid topology control, 2025. URL https://arxiv.org/abs/2501.07186.

B. Donnot. Grid2op- a testbed platform to model sequential decision making in power systems., 2020. URL https://GitHub.com/Grid2Op/grid2op.

Matthias Dorfer, Anton R. Fuxjäger, Kristian Kozak, Patrick M. Blies, and Marcel Wasserer. Power grid congestion management via topology optimization with alphazero, 2022. URL https://arxiv.org/abs/2211.05612.

Agrim Gupta, Linxi Fan, Surya Ganguli, and Li Fei-Fei. Metamorph: learning universal controllers with transformers. In *International Conference on Learning Representations*. ICLR, 2022.

Andrew Holliday. Applications of deep reinforcement learning to urban transit network design. 2025.

Andrew Holliday, Ahmed El-Geneidy, and Gregory Dudek. Learning heuristics for transit network design and improvement with deep reinforcement learning. *arXiv preprint arXiv:2404.05894*, 2024.

Wenlong Huang, Igor Mordatch, and Deepak Pathak. One policy to control them all: Shared modular policies for agent-agnostic control. In *International Conference on Machine Learning*, pp. 4455–4464. PMLR, 2020.

Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Mohammadamin Barekatain, Simon Schmitt, and David Silver. Learning and planning in complex action spaces. In *International Conference on Machine Learning*, pp. 4476–4486. PMLR, 2021.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

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 *NeurIPS 2020 Competition and Demonstration Track*, pp. 112–132. PMLR, 2021.

Mingshuo Nie, Dongming Chen, and Dongqi Wang. Reinforcement learning on graphs: A survey. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 7(4):1065–1082, 2023.

Tianhao Peng, Wenjun Wu, Haitao Yuan, Zhifeng Bao, Zhao Pengru, Xin Yu, Xuetao Lin, Yu Liang, and Yanjun Pu. Graphrare: Reinforcement learning enhanced graph neural network with relative entropy. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pp. 2489–2502. IEEE, 2024.

Florian Rupp and Kai Eckert. G-pcgrl: Procedural graph data generation via reinforcement learning. In *2024 IEEE Conference on Games (CoG)*, pp. 1–8. IEEE, 2024.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Maciej Świechowski, Konrad Godlewski, Bartosz Sawicki, and Jacek Mańdziuk. Monte carlo tree search: A review of recent modifications and applications. *Artificial Intelligence Review*, 56(3): 2497–2562, 2023.

Brandon Trabucco, Mariano Phielipp, and Glen Berseth. AnyMorph: Learning transferable polices by inferring agent morphology. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 21677–21691. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/trabucco22b.html.

Erica van der Sar, Alessandro Zocca, and Sandjai Bhulai. Multi-agent reinforcement learning for power grid topology optimization, 2023. URL https://arxiv.org/abs/2310.02605.

Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. *Advances in neural information processing systems*, 31, 2018.

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020.

A Experiment Details

A.1 Grid2Op Environment Names

The grids used in the experiments §5 refer to the following environments in Grid2Op:
`Small Grid A` refers to `l2rpn_case14_sandbox` environment.
`Small Grid B` refers to `l2rpn_case14_sandbox_diff_grid` environment.
`Large Grid A` refers to the `l2rpn_wcci_2020` environment.
`Large Grid B` refers to the `l2rpn_icaps_2021_small` environment.

A.2 Hyperameters

PPO Training Setup	
Topology	Case 14
Seeds	3
Env. interactions (Depth 600)	1,860,000
Env. interactions (Depth 200)	1,460,000

PPO Hyperparameters	
Batch size	128
Actor learning rate	3×10^{-4}
Critic learning rate	3×10^{-4}
γ (discount factor)	0.99
GAE λ	0.95
Max gradient norm	0.5
Update epochs	1
Clip coefficient	0.2
Entropy coefficient	0.01
Clip value loss coefficient	0.2
Value function coefficient (v_f coef)	0.5

MCTS Parameters	
Number of simulations	10 / 50 / 100
Rollout depth	200 / 600
Episode depth	200 / 600
Exploration parameter (c)	100

While we tested 10, 50 and 100 simulations with MCTS, we found 10 to be sufficient after simulating once every expanded node. Therefore, we report the results in §5 to be with 10 simulations.

A.3 Additional Environment Figures

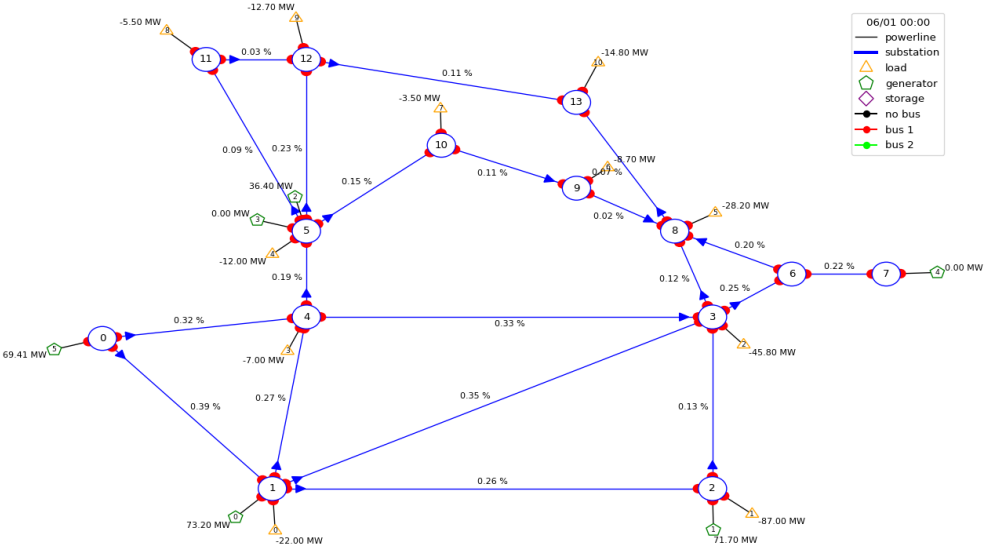

Figure 4: Small Grid Environment with 14 Substations

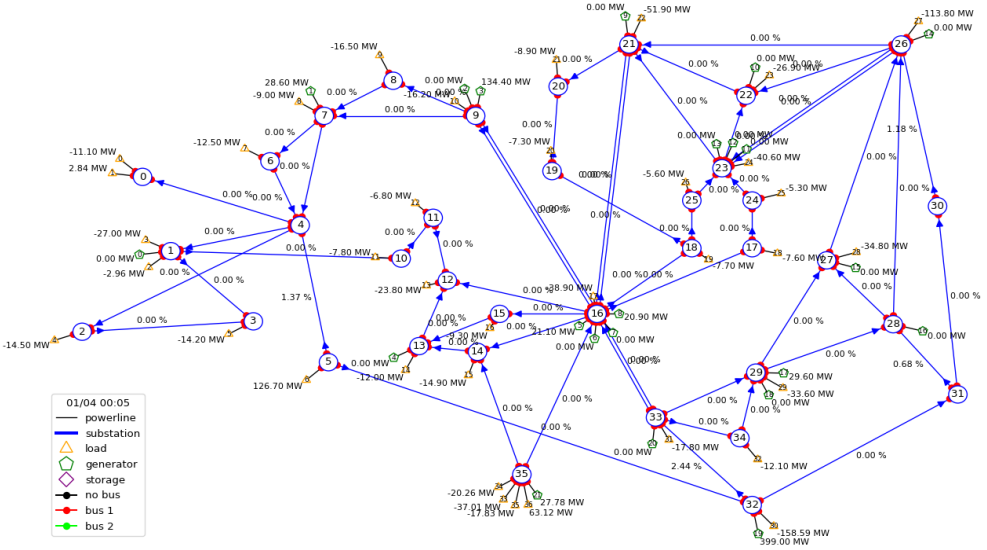

Figure 5: Large Grid Environment with 36 Substations