Physics-Informed Graph Neural Network for Dynamic Reconfiguration of Power Systems

Jules Authier*† Rabab Haider*, Anuradha Annaswamy* and Florian Dörfler†
*Department of Mechanical Engineering, MIT, Cambridge, USA
†Automatic Control Laboratory, ETH Zurich, Zurich, Switzerland

***Abstract*—To maintain a reliable grid we need fast decision-making algorithms for complex problems like Dynamic Reconfiguration (DyR). DyR optimizes distribution grid switch settings in real-time to minimize grid losses and dispatches resources to supply loads with available generation. DyR is a mixed-integer problem and can be computationally intractable to solve for large grids and at fast timescales. We propose GraPhyR, a Physics-Informed Graph Neural Network (GNNs) framework tailored for DyR. We incorporate essential operational and connectivity constraints directly within the GNN framework and train it end-to-end. Our results show that GraPhyR is able to learn to optimize the DyR task.**

***Index Terms*—Graph Neural Network, Dynamic Reconfiguration, Physics Informed Learning.**

I. Introduction

The global energy landscape is rapidly evolving with the transition towards renewable energy generation. This transition brings numerous benefits for the climate, but also presents challenges in effectively controlling and optimizing power systems with high penetration of intermittent renewable generation such as solar and wind. New operating schemes are needed to ensure efficient and reliable grid operations in the presence of intermittent generation. Significant research efforts focus on optimizing resource dispatch and load flexibility towards reducing costs and increasing grid efficiency; however there remains efficiency gains to be had when co-optimizing grid topology. To this end, we propose *Dynamic Reconfiguration* (DyR) in a distribution grid to increase operating efficiency by co-optimizing grid topology and resource dispatch.

The distribution grid reconfiguration problem involves the selection of switch states (open/closed) to meet demand with available generation, while satisfying voltage and operating constraints. Grid reconfiguration can re-route power flows to reduce power losses [1], increase utilization of renewable generation [2], [3], and re-energize grids after contingencies. Presently, DyR is deployed for loss reduction in the EU [2], and for fault conditions in the US using rule-based control schemes. The widespread growth of distributed generation, storage, and electric vehicles creates the opportunity for DyR for loss reduction, whereby topology and dispatch decisions are made *fast and frequently* in response to faster resource timescales; as solar generation varies, the topology is adapted to supply loads in close proximity to generation, thus reducing losses and improving voltage profiles across the grid.

The DyR problem is a mixed integer program (MIP) due to the discrete nature of switch decisions. It is well known that MIPs are NP-hard (i.e. cannot be solved in polynomial time) and thus may be computationally intractable for large-scale problems. A distribution substation may have 10 feeders each with 5 switches, resulting in over 10^{15} possible topologies. If operating constraints and load conditions result in only 1% of these topologies, the search space remains prohibitively large for traditional approaches. One option is to restrict the optimization to a single feeder, however optimizing topology over all feeders permits load transfer and generation exports.

Machine learning (ML) offers an alternative by shifting the computational burden to offline training, thereby making dynamic decision making via the online application of ML algorithms computationally feasible. Recent works propose ML for solving MIPs and combinatorial optimization (CO) [4], either in an end-to-end fashion or to accelerate traditional solvers. Graphs play a central role in formulating many CO problems [5], representing paths between entities in routing problems, or interactions between variables and constraints in a general CO [6], [7]. The use of Graph Neural Networks (GNNs) is also being explored to leverage the underlying graph structure during training and identify common patterns in problem instances. The traveling salesman problem (TSP) is a fundamental problem in CO and a standard benchmark which has been extensively studied with traditional optimization techniques. Recently, GNNs have been used to solve the TSP with good performance and generalizability [8], [9], [10]. In this work we leverage GNNs to learn the power flow representation for reconfiguration.

Grid reconfiguration for distribution grids has been studied with varying solution methodologies including knowledge-based algorithms and single loop optimization [1], [11], heuristic methods [12], [13], and reformulation as a convex optimization problem using big-$\mathcal{M}$ constraints [14], [15], [16]. However, these methods are not computationally tractable for large-scale optimization in close to real-time applications, and may be limited to passive grids (i.e. no local generation). Machine learning approaches for DyR have also been proposed [17], [3]. In [17] the DyR problem is formulated as a Markov decision process and solved using reinforcement learning. In [3] a light-weight physics-informed neural network is proposed as an end-to-end learning to optimize framework

This work is partially supported by the MIT Energy Fellowship.

with guarantee certified satisfiability of the power physics. A physics-informed rounding layer explicitly embeds the discrete decisions within the neural framework. These approaches show potential, but both are limited to a given grid topology and switch locations. Our approach is similar to that of [3] wherein we embed discrete decisions directly within an ML framework.

The main contribution of this paper is **GraPhyR**, a graph neural network (**Gra**) framework employing physics-informed rounding [3] (**PhyR**) for DyR in distribution grids. GraPhyR is an end-to-end framework that learns to optimize the reconfiguration task, enabled by four key architectural components:

(1) A message passing layer that models switches as gates: The gates are implemented as a value between zero and one to model switches over a continuous operating range. Gates control the flow of information through switches in the GNN, modeling the control of physical power flow between nodes.

(2) A scalable local prediction method: We make power flow predictions locally at every node in the grid using local features. The predictors are scale-free and so can generalize to grids of any topology and size.

(3) A physics-informed rounding layer: We embed the discrete open/closed decisions of switches directly within the neural framework. PhyR selects a grid topology for each training instance upon which GraPhyR predicts a feasible power flow and learns to optimize a given objective function.

(4) A GNN that takes the electrical grid topology as input: We treat the grid topology and switch locations as an input which permits GraPhyR to learn the power flow representation across multiple possible distribution grid topologies within and across grids. Thus GraPhyR can optimize topology and generator dispatch: (a) on multiple grid topologies seen during training, and (b) under varying grid conditions such as (un)planned maintenance of the grid.

We demonstrate the performance of GraPhyR in predicting near-optimal and feasible solutions. We also show the effectiveness of GraPhyR in adapting to unforeseen grid conditions.

The remainder of this paper is organized as follows. Section II presents DyR as an optimization problem. Section III presents the GraPhyR method and details the four key architectural components. Section IV presents the simulation results, and conclusions are drawn in Section V.

II. Reconfiguration as an Optimization Problem

We consider DyR of distribution grids with high penetration of distributed generation. We model the power physics using Linearized DistFlow [1] as below:

$$\min_{\boldsymbol{\psi}} f(\mathbf{x}, \boldsymbol{\psi}) = \sum_{(i,j)\in\mathcal{A}} (p_{ij}^2 + q_{ij}^2) R_{ij} \tag{1}$$

$$\text{s.t. } p_j^G - p_j^L = \sum_{k:(j,k)\in\mathcal{A}\cup\mathcal{A}sw} p_{jk} - \sum_{i:(i,j)\in\mathcal{A}\cup\mathcal{A}sw} p_{ij}, \quad \forall j \in \mathcal{N} \tag{2}$$

$$q_j^G - q_j^L = \sum_{k:(j,k)\in\mathcal{A}\cup\mathcal{A}sw} q_{jk} - \sum_{i:(i,j)\in\mathcal{A}\cup\mathcal{A}sw} q_{ij}, \quad \forall j \in \mathcal{N} \tag{3}$$

$$v_i - v_j = 2(R_{ij}p_{ij} + X_{ij}q_{ij}), \quad \forall (i,j) \in \mathcal{A} \tag{4}$$

$$\begin{cases} v_i - v_j = 2(R_{ij}p_{ij} + X_{ij}q_{ij}) & \text{if } y_{ij} = 1 \\ \text{inactive} & \text{if } y_{ij} = 0 \end{cases} \quad \forall (i,j) \in \mathcal{A}_{sw} \tag{5}$$

$$-\mathcal{M}y_{ij} \le p_{ij} \le \mathcal{M}y_{ij}, \quad \forall (i,j) \in \mathcal{A}_{sw} \tag{6}$$

$$-\mathcal{M}y_{ij} \le q_{ij} \le \mathcal{M}y_{ij}, \quad \forall (i,j) \in \mathcal{A}_{sw} \tag{7}$$

$$p_j^G \in \left[\underline{p}_j^G, \overline{p}_j^G\right], \ q_j^G \in \left[\underline{q}_j^G, \overline{q}_j^G\right] \quad \forall j \in \mathcal{N} \tag{8}$$

$$v_j \in [\underline{v}, \overline{v}], \ v_{j\#} = 1, \quad \forall j \in \mathcal{N} \tag{9}$$

$$\sum_{(i,j)\in\mathcal{A}_{sw}} y_{ij} = N - 1 - M \tag{10}$$

$$|\delta_{\mathcal{A}}(j)| + \sum_{j:(i,j)\in A_{sw}} y_{ij} + \sum_{j:(j,i)\in A_{sw}} y_{ji} \ge 1, \quad \forall j \in \mathcal{N} \tag{11}$$

where $\mathbf{x} = [\mathbf{p^L}, \mathbf{q^L}]$ and the decision variable is $\boldsymbol{\psi} = [\mathbf{y}, \mathbf{v}, \mathbf{p_{ij}}, \mathbf{q_{ij}}, \mathbf{p^G}, \mathbf{q^G}]$. The equations above describe a distribution grid as a directed graph $\mathcal{G}(\mathcal{N}, \mathcal{A}, \mathcal{A}_{sw})$, with $\mathcal{N}$ the set of N nodes, $\mathcal{A}$ the set of M directed edges, and $\mathcal{A}_{sw}$ the set of M_{sw} switches. The distribution grid is connected to the transmission grid via the point of common coupling (PCC), node $j^{\#}$, which is the slack bus. The real and reactive power loads at a node i are p_i^L, q_i^L, and generation are p_i^G, q_i^G. Generation at the PCC indicates import from the transmission grid. The squared magnitude of the voltage at node i is v_i. The directed real and reactive power flows across a line (switch) from node i to j are p_{ij} and q_{ij}. The power flows are uniquely defined on the directed graph, where $p_{ij} > 0$ indicates flow from i to j, and $p_{ij} < 0$ indicates flow from j to i. The same applies for q_{ij}. The switch status is given by $y_{ij}, \forall (i,j) \in A_{sw}$ and takes on a binary value, one if the switch is closed and zero is the switch is open. The line resistance and reactance are R_{ij} and X_{ij} respectively. We define $\delta_{\mathcal{A}}(j)$ as the set of edges in $\mathcal{A}$ connected to node j, and $|\delta_{\mathcal{A}}(j)|$ as the number of those edges.

The objective function in (1) linearly approximates electric losses. Eq. (2)-(5) describe the Linearized DistFlow model [1] which assumes lossless power balance (2)-(3), and approximates Ohm's Law as a linear relationship between voltages and power (4)-(5). Eq. (5) accommodates switches in the model with a conditional constraint where Ohm's Law is enforced for closed switches. The big-$\mathcal{M}$ constraint in (6) and (7) enforces power flows through open switches to be zero. Eq. (8) describes the nodal injection constraints. Eq. (9) sets the voltage constraints and the slack bus voltage. Eq. (10)-(11) describe radiality and connectivity constraints required for distribution grids under normal operations. We assume existing protection schemes are used which require radiality of the grid topology; we then enforce radiality in the reconfiguration problem. Note that (11) is not sufficient to enforce connectivity. To maintain a simple problem formulation, we leverage the fact that the power flow constraints implicitly enforce connectivity: a load cannot be supplied if it is disconnected from the grid.

III. GraPhyR: End-to-end learning for dynamic reconfiguration

We propose GraPhyR, a physics-informed machine learning framework to solve (1)-(11). Our framework in Fig. 1 features four architectural components: (A) gated message passing to model switches, (B) local predictions to scale across nodes, (C) physics-informed rounding to handle binary variables, and (A) topology input data for adaptability during online deployment. We embed the physics of the distribution grid and reconfiguration problem within each component of the GraPhyR framework. First, the GNN embeds the topology of the underlying distribution grid, and explicitly models the switches using gated message passing. Second, the topology selection embeds the discrete open/close decision of the switches using the physics-informed rounding. Third, we use the power flow equations to predict a subset of variables (denoted as the independent variables), and compute the remaining variables in a recovery step. The GraPhyR framework uses these physics-informed layers to learn to optimize the reconfiguration task while satisfying equality and binarity constraints. The framework is presented in detail next.

A. Message Passing

The GNN models the distribution grid topology as an undirected graph, with switch embeddings modeling the switches in the electrical grid. The GNN's message passing layers incorporate these embeddings as gates, which enables GraPhyR to learn the representation of linearized Ohm's law of (5) across multiple topologies in a physics-informed way. The input to the GNN are the grid topology and nodal loads, and the output is a set of node and switch embeddings which will be used to make reconfiguration, power flow, and voltage predictions.

1) Grid Topology as Input Data for Graph Structure: An input to the GNN is the grid topology described by $\mathcal{G}(\mathcal{N}, \mathcal{A}, \mathcal{A}_{sw})$, using which the GNN models the physical grid topology as an undirected graph $\mathcal{G}(\mathcal{N}, \mathcal{E}, \mathcal{E}_{sw})$ with N nodes, M lines, and M_{sw} switches. Trivially, $\mathcal{E}$ ($\mathcal{E}_{sw}$) represents the undirected communication links along the directed edges $\mathcal{A}$ ($\mathcal{A}_{sw}$) to support message passing and extracting the problem representation in the embeddings. By including $\mathcal{G}(\mathcal{N}, \mathcal{A}, \mathcal{A}_{sw})$ as an input to the GNN, our GraPhyR framework is able to adapt to changing grid conditions, rather than requiring a large training dataset with multiple scenarios.

2) Initial Node, Line, and Switch Embeddings: The second input data to the GNN is the load data $\mathbf{x}^0$ which defines the node embeddings. The load data contains the active and reactive power load p_i^L and q_i^L for each node i in the grid and thus determines the initial node embeddings x_i^0 of every node i in the corresponding graph where $\mathbf{x}^0 = \left[x_1^0, \ldots, x_N^0\right]^T = \left[(p_0^L, q_0^L), \ldots, (p_N^L, q_N^L)\right]^T$. The line embeddings are set to one and are not updated by the message passing layers. The switch embeddings determine the value of the gate and are randomly initialized, similar to randomly initializing weights in a neural network. The switch embeddings are updated through the message passing layers. Initial line and switch embeddings are given by $z_{ij}^0,\ \forall \{i,j\} \in \mathcal{E} \cup \mathcal{E}_{sw}$.

3) Message Passing Layers: In each hidden layer of the GNN the nodes in the graph iteratively aggregate information from their local neighbors. Deeper GNNs have more hidden layers and thus have node embeddings which contain information from further reaches of the graph. For each node embedding x_i^0 in the graph, the first message passing layer is defined in (12) where $\mathcal{N}_i$ denotes the set of neighboring nodes to node i. For each switch embedding z_{ij}^0 in the graph, the first message passing layer is defined in (14).

$$x_i^1 = ReLU(W_1^0 x_i^0 + \sum_{j \in \mathcal{N}_i} \{W_2^0 \cdot f(z_{ij}^0) \cdot x_j^0\}) \tag{12}$$

$$f(z_{ij}^0) = \begin{cases} sig(z_{ij}^0) & \text{if } \{i,j\} \in \mathcal{E}_{sw} \\ 1 & \text{otherwise} \end{cases} \tag{13}$$

$$z_{ij}^1 = ReLU(W_3^0(x_i^0 + x_j^0) + W_4^0 z_{ij}^0), \forall \{i,j\} \in \mathcal{E}_{sw} \tag{14}$$

For the remaining message passing layers, denoted by $l \in \{1, 2, \ldots, \mathcal{L} - 1\}$, a residual connection is added to improve prediction performance and training efficiency [18]. The resulting node and switch embeddings are:

$$x_i^{l+1} = x_i^l + ReLU(W_1^l x_i^l + \sum_{j \in \mathcal{N}_i} \{W_2^l \cdot f(z_{ij}) \cdot x_j^l\}) \tag{15}$$

$$f(z_{ij}^l) = \begin{cases} sig(z_{ij}^l) & \text{if } \{i,j\} \in \mathcal{E}_{sw} \\ 1 & \text{otherwise} \end{cases} \tag{16}$$

$$z_{ij}^{l+1} = z_{ij}^l + ReLU(W_3^l(x_i^l + x_j^l) + W_4^l z_{ij}^l), \forall \{i,j\} \in \mathcal{E}_{sw} \tag{17}$$

The line embeddings are trivially set to one. We omit residual connections in the first message passing layer to expand the input embeddings x_i^0 with dimensions of the input data, to an arbitrarily large hidden embeddings dimension h. This allows the GNN to learn more complex representations by extracting features in a higher dimensional space.

4) Gates: We implement gates in the message passing layer by applying a sigmoid to the switch embeddings, as in (13) and (16). The function $f(z_{ij})$ acts like a filter for the message passing between two neighboring nodes, attenuating the information signal if the switch is closed. The gate models the switches as a continuous switch (ex. a household light dimmer), controlling information flow in the same way a switch controls power flow between two nodes.

5) Global Graph Information: In the final message-passing layer, we calculate a global graph embedding, $x_G^{\mathcal{L}} = \sum_{i=1}^N x_i^{\mathcal{L}}$. This embedding offers information access across the graph and can reduce the need for an excessive number of message passing layers for sparse graphs, such as those in power systems. This improves the computational efficiency.

B. Prediction

After the $\mathcal{L}$ message passing layers, the embeddings extracted from the input data are used to predict the switch open/close status and a subset of the power flow variables, denoted as independent variables.

1) Variable Space Partition: We partition the variable space into independent and dependent variables. The independent variables constitute the active power flows p_{ij}, nodal voltages v_i, and switch open/close status y_{ij}. The dependent variables constitute the reactive power flows q_{ij}, and nodal generation $\{p_i^G, q_i^G\}$. We leverage techniques for variable space reduction

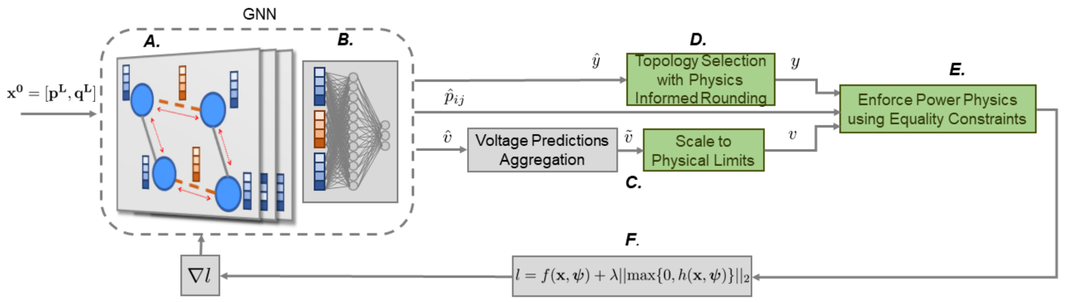
GNN
A.
B.
D.
Topology Selection with Physics Informed Rounding
E.
Enforce Power Physics using Equality Constraints
Voltage Predictions Aggregation
Scale to Physical Limits
C.
F.
$\mathbf{x^0} = [\mathbf{p^L}, \mathbf{q^L}]$
$\hat{y}$
y
$\hat{p}_{ij}$
$\hat{v}$
$\tilde{v}$
v
∇l
$l = f(\mathbf{x}, \boldsymbol{\psi}) + \lambda \|\max\{0, h(\mathbf{x}, \boldsymbol{\psi})\}\|_2$

Fig. 1. GraPhyR: proposed framework to solve the DyR problem.

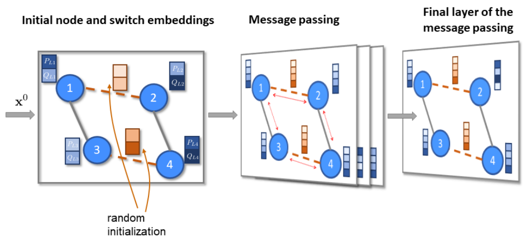
Initial node and switch embeddings
Message passing
Final layer of the message passing
random initialization
$\mathbf{x}^0$

Fig. 2. Message passing layers where switches are denoted by red-dashed lines. The node and switch embeddings are represented by blue and red colored blocks respectively, where the number of squares per-block indicates the dimension of the embeddings h.

TABLE I
DIMENSIONS OF LOCAL AND GLOBAL PREDICTORS

	L-predictor	S-predictor	Global-predictor
Input dimension	$3h$	$4h$	$(N + 2M_{sw})h$
Output dimension	3	4	$N + M + 2M_{sw}$

to calculate the dependent variables from the independent variables, using constraints (2)-(11). This step ensures that the power physics constraints have certified satisfability, as further discussed in Section III-E.

This partition is non-unique. It critically depends on the structure of the given problem which determines the relationship between the sets of variables, and the neural architecture which determines the relationship between inputs, predictions, and consecutive neural layers. **We further advocate that the neural architecture itself must be physics-informed, to embed domain knowledge and physical constraints directly into the neural network, as we have done in GraPhyR.**

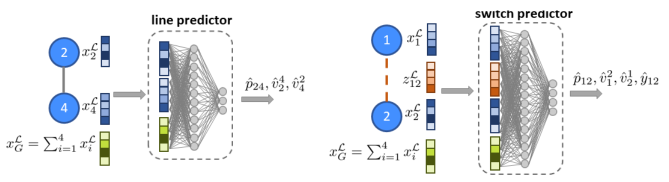
line predictor
switch predictor
$x_2^{\mathcal{L}}$
$x_4^{\mathcal{L}}$
$x_G^{\mathcal{L}} = \sum_{i=1}^4 x_i^{\mathcal{L}}$
$\hat{p}_{24}, \hat{v}_2^4, \hat{v}_4^2$
$x_1^{\mathcal{L}}$
$z_{12}^{\mathcal{L}}$
$x_2^{\mathcal{L}}$
$\hat{p}_{12}, \hat{v}_1^2, \hat{v}_2^1, \hat{y}_{12}$

Fig. 3. Local predictions made by the switch and line predictors use the node and switch embeddings extracted after $\mathcal{L}$ message passing layers.

2) Local Prediction Method: Our prediction method leverages two key observations: (i) the relationship between power flows and voltages are the same for any node-edge pair and are modelled by the physics equations (2)-(5); (ii) the binary nature of switches makes it inherently different from a distribution line. Using these, we define two local prediction methods which use multi-layer perceptrons: a line predictor (L-predictor) and a switch predictor (S-predictor), shown in Fig. 3. The L-predictor in (18) predicts power flow and the voltages of the two nodes connected by the line using the node and global embeddings. The S-predictor also predicts the probability for the switch to be closed, using the switch embeddings $z_{ij}^{\mathcal{L}}$ in addition to the node and global embeddings, as in (19). All predictions are denoted with a hat (i.e. $\hat{v}_i$) and will be processed in subsequent layers to render the final topology and dispatch decisions.

$$[\hat{p}_{ij}, \hat{v}_i^j, \hat{v}_j^i] = \text{L-predictor}[x_i^{\mathcal{L}}, x_j^{\mathcal{L}}, x_G^{\mathcal{L}}], \qquad \forall (i,j) \in \mathcal{A} \tag{18}$$

$$[\hat{p}_{ij}, \hat{v}_i^j, \hat{v}_j^i, \hat{y}_{ij}] = \text{S-predictor}[x_i^{\mathcal{L}}, x_j^{\mathcal{L}}, z_{ij}^{\mathcal{L}}, x_G^{\mathcal{L}}], \ \forall (i,j) \in \mathcal{A}_{sw} \tag{19}$$

Our local predictors exploit the full flexibility of GNNs. They are permutation invariant to the input graph data; are independent of the size of the graph (scale-free); and are smaller than the corresponding global predictor for the same grid. The first feature means our framework is robust to changes in input data. The last two features means our framework is lightweight and scalable. This would not be possible with a global predictor which predicts all independent variables from node and switch embeddings across the graph. The size of the input and output layers of a global predictor would depend on the size of the graph and the number of switches, and is the limitation in [3]. Table I summarizes the size of local and global predictors for the reconfiguration problem, where h is the dimension of the hidden graph embeddings.

C. Voltage Aggregation and Certified Satisfiability of Limits

The local predictions obtained from the L-predictor and S-predictor generate multiple instances of voltage predictions for each node as indicated by a superscript. Specifically, the number of instances corresponds to the degree of the node i, $|\delta_{\mathcal{E}\cup\mathcal{E}_{sw}}(i)|$. We aggregate the voltage predictions to a unique value for each node in the grid as $\tilde{v}_i = \frac{1}{|\delta_{\mathcal{E}\cup\mathcal{E}_{sw}}(i)|} \sum_{j:\{i,j\}\in\mathcal{E}\cup\mathcal{E}_{sw}} \hat{v}_i^j$. The voltage predictions are then scaled onto the box constraints (9) with $v_i = \underline{v}\cdot(1-\tilde{v}_i)+\overline{v}\cdot\tilde{v}_i$. Notably, by selecting voltages as an independent variable in our variable space partition, we certify that voltage limits across the grid will always be satisfied, a critical aspect of power systems operation.

D. Topology Selection using Physics-Informed Rounding

The S-predictor provides probabilistic predictions for open/close decisions of each switch. We recover binary decisions using a physics-informed rounding (PhyR) algorithm [3]. We exploit the radiality of distribution grids, which requires $\mathcal{S} = N - 1 - M$ switches to be closed so there are always $N - 1$ conducting lines. The PhyR method selects the $\mathcal{S}$ switches with the largest probabilities $\hat{y}_{ij}$ and closes them by setting the corresponding $y_{ij} = 1$; the remaining switches are opened, $y_{ij} = 0$. This enforces (**??**) and (10). Note that as distribution grid technologies advance, bidirectional and loop flows may be easily incorporated in new protection schemes. This would remove the radiality constraint, which GraPhyR can accommodate with suitable modifications to PhyR.

A note must be made about the practical implementation: PhyR is implemented with $\min$ and $\max$ operators which return gradients of 0, "killing" the gradient information necessary for backpropagation. We preserve these gradients in the computational graph by setting all but one switches to binary values, those with $\mathcal{S} - 1$ largest probabilities. Training guides the remaining switch towards a binary value.

E. Certified Satisfiability of Power Physics

The final neural layer recovers the full variable space and enforces power flow constraints through open switches. The following steps happen sequentially:

1) Given $\mathbf{y}$ and the independent variables we compute the reactive power flows $\tilde{q}_{ij}$ using (4)-(5).
2) Given $\mathbf{y}$ we enforce (6) and (7) as $p_{ij} = (\hat{p}_{ij} - 0.5) \cdot 2\mathcal{M}y_{ij}$ and $q_{ij} = (\tilde{q}_{ij} - 0.5) \cdot 2\mathcal{M}y_{ij}$ respectively. By explicitly setting flows through open switches to zero we enforce the constraints in a hard way. This step also explicitly enforces the condition constraint describing Ohm's Law (5).
3) Active and reactive power nodal generation is calculated using (2) and (3), respectively.

F. Loss Function

The neural network learns to optimize by using an unsupervised framework. It has two objectives: to minimize line losses in (1), and to minimize inequality constraint violations of generation constraint (8) and connectivity constraints (11). Denoting these constraints as $h(\mathbf{x}, \boldsymbol{\psi}) \leq 0$, we regularize the loss function using a soft-loss penalty with hyperparameter λ. The loss function is $l = f(\mathbf{x}, \boldsymbol{\psi}) + \lambda||\max\{0, h(\mathbf{x}, \boldsymbol{\psi})\}||_2$.

Remark 1: The inequality constraints (6), (7), and (9) have certified satisfiability by design of GraPhyR.

Remark 2: Our loss function is unsupervised, and does not need the optimal solutions, which may be unknown or computationally prohibitive to compute.

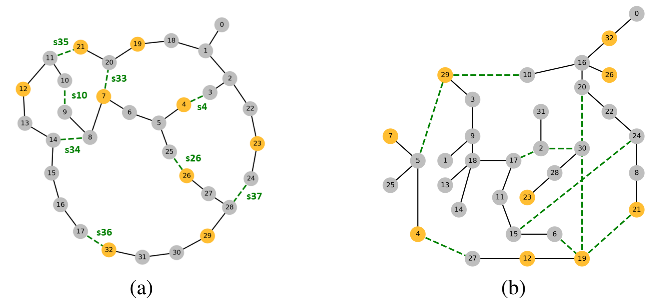

Fig. 4. Grid topology of BW-33 (left) and the synthetic $\mathcal{G}_1$ (right). Switches indicated with green dashed lines. Solar generator locations indicated with yellow nodes.

IV. Experimental Results

A. Dataset and Experiment Setup

We first evaluate GraPhyR on a canonical distribution grid BW-33 [1] with 33 nodes, 29 lines, and 8 switches. We generate a variant of BW-33, called $\mathcal{G}_1$ with 33 nodes, 27 lines, and 10 switches. Second, we evaluate GraPhyR on a a model of a real distribution grid TPC-94 [19] with 94 nodes and 14 switches. The grid has 11 individual distribution feeders which can be connected to one another via switches (i.e. reconfiguration) to share load across feeders. We use the datasets from [3] which introduce distributed solar generation in the grid with a penetration of 25% generation-to-peak load. Loads are perturbed about their nominal value as typically done in literature. The two networks are shown in Fig. 4. The dataset has 8600 data points per grid which are divided as 80/10/10 for training/validation/testing.

We implement GraPhyR using PyTorch and train on the MIT supercloud [20]. GraPhyR has 4 message passing layers each with dimension 8 ($\mathcal{L} = 4, h = 8$). The L-predictor and S-predictor have a single hidden layer with dimension 24 and 32 respectively. We use 10% dropout, batch normalization, and ReLU activation in both predictors. The soft loss hyperparameter is $\lambda = 100$, big-$\mathcal{M}$ relaxation parameter is 0.5 per unit (p.u.). The voltage bounds for BW-33 are $\underline{v} = 0.83, \overline{v} = 1.05$ p.u. to adapt to the lossy behaviour of the grid [1], [3], and for TPC-94 we use the typical $\underline{v} = 0.83, \overline{v} = 1.05$ p.u. limits. We use ADAM optimizer with a learning rate of $\gamma = 5e^{-4}$, a batch size of 200, and train for 1500 epochs. We evaluate the performance of the neural framework using a committee of networks approach. We train 10 models with independent weight initialization and average the predictions across all models.

B. Performance Metrics

We adopt the performance metrics defined in [3] to assess prediction performance. The asterisks notation (i.e. v^*) denotes the optimal solution obtain from a MIP solver.

Dispatch error: optimality metric of mean-squared error (MSE) in optimal generator dispatch: $\frac{1}{N}\sum_{j\in\mathcal{N}} (p^G_j - p^{G*}_j)^2 + (q^G_j - q^{G*}_j)^2$.

Voltage error (VoltErr): optimality metric of MSE in nodal voltage prediction: $\frac{1}{N}\sum_{j\in\mathcal{N}}(v_j - v_j^*)^2$.
Topology error: optimality metric of the Hamming distance [21] between two topologies, calculated as the ratio of switch decisions not in the optimal position: $\frac{1}{M_{sw}}\sum_{(i,j)\in\mathcal{A}_{sw}}(y_{ij} - y_{ij}^*)^2$.
Inequality violation: feasibility metric of the magnitude of violations in constraint set, measuring the mean and maximum as $\frac{1}{|h|}\sum_k \max\{0, h^k(\mathbf{x},\boldsymbol{\psi})\}$ and $\max_k\{\max\{0, h^k(\mathbf{x},\boldsymbol{\psi})\}\}$.
Number of violations exceeding a threshold: feasibility metric of the number of inequality constraints which are violated by more than an ϵ threshold: $\sum_k \mathbb{I}_{\max\{0,h^k(\mathbf{x},\boldsymbol{\psi})\}>\epsilon}$.

C. Benchmark models

We benchmark our results against multiple solution techniques, including a traditional MIP solver and different ML frameworks, as described below.
Optimizer: Traditional optimization solver, Gurobi, a state-of-art commercial solver for MIPs.
GraPhyR: The proposed method described in Section III, with a GNN with switch embeddings, local predictors (the L-predictor and S-predictor), and PhyR layer to recover binary decisions.
Global-GraPhyR: A modification of the proposed GraPhyR method which uses a global predictor which predicts all independent variables from node and switch embeddings.
SiPhyR: An physics-informed method introduced in [3] which uses a lightweight fully connected neural network with a sigmoidal output layer to predict the independent variables (rather than the GNN). The topology selection uses the PhyR algorithm, and the variable space decomposition is modified to additionally include integer variables indicating the directionality of power flow in the lines.
InSi: A simple neural network without the proposed PhyR layer [3]. Integer solutions for the switch status are encouraged (read: not enforced) by using a differentiable relaxation of the step function, the integer sigmoid (InSi): $\sigma_{InSi}(z) = \left[2\frac{1+\mu}{\mu+e^{-\tau z}} - 1\right]_+$, where τ, μ are free parameters [22].

D. Case (a). GraPhyR with Local vs. Global Predictors

We first compare GraPhyR with local predictors to a variant with a global predictor, termed Global-GraPhyR. The global predictor determines all independent variables (real power flows, voltages, switch probabilities) using all node and line embeddings. We implement the global predictor with a single hidden layer of the same size as the input dimension. The global predictor has input/output dimensions of 328/78 as compared to the L-predictor and S-predictor with dimensions of 24/3 and 32/4 respectively. Note that the global predictor predicts one voltage per node so voltage aggregation is not needed. We also compare the performance of GraPhyR with that of prior work which use a simple neural network with two hidden layers [3]: *SiPhyR* which employs PhyR; and *InSi* which approximates a step function.

Table II-(a) shows the prediction performance for these methods. We first observe that **the GNN frameworks achieve lower dispatch error**, with Global-GraPhyR outperforming SiPhyR by two orders of magnitude. The GNN uses topological information to optimize the dispatch and satisfy loads. Second, **the PhyR-based frameworks achieve lower topology errors** by up to 10%, by embedding the discrete decisions directly within the ML framework. However, the topology error remains high ($> 30\%$), demonstrating the challenge in learning to optimize this combinatorial task. Finally, **SiPhyR and Global-GraPhyR achieve the best performance across feasibility metrics**, with lower magnitude and number of inequality violations. Notably, the maximum inequality violation is an order of magnitude higher for InSi which does not benefit from PhyR, and GraPhyR which makes local predictions. This is expected. First, PhyR explicitly accounts for binary variables within the training loop to enable the end-to-end learning: PhyR selects a feasible topology upon which the neural framework predicts a near-feasible power flow solution. Second, GraPhyR sacrifices some prediction performance for the flexibility to train and predict on multiple graphs. Figure 5 plots the mean inequality violation of the 10 trained GraPhyR models, for the sets of inequality constraints, namely (8), (9), and (11). The constraints are always respected for voltage (by design) and connectivity (by constraint penalty). Nodal generation constraints are frequently violated as the lowest cost (lowest line losses) solution is to supply all loads locally.

We next test the limits of topology prediction within our ML framework by comparing with a semi-supervised approach. The loss function includes a penalty on the switch status:

$$l_{sm} = f(\mathbf{x},\boldsymbol{\psi}) + \lambda||\max\{0, h(\mathbf{x},\boldsymbol{\psi})\}||_2 + \mu\|\mathbf{y} - \mathbf{y}^*\|_2 \quad (20)$$

Table II-(a) shows the performance of the semi-supervised GraPhyR. We also include results of Supervised-SiPhyR from [3] which uses a regression loss for voltages, generation, and switch statuses, and an inequality constraint violation penalty:

$$\begin{aligned} l_{sup}(z,\varphi) = {} & \|(\mathbf{v}-\mathbf{v}^*)^2 + (\mathbf{p}^{\mathbf{G}} - \mathbf{p}^{\mathbf{G}*})^2 + (\mathbf{q}^{\mathbf{G}} - \mathbf{q}^{\mathbf{G}*})^2\|_2^2 \\ & + \|(\mathbf{y}-\mathbf{y}^*)^2\|_2^2 + \lambda||\max\{0, h(\mathbf{x},\boldsymbol{\psi})\}||_2 \end{aligned} \quad (21)$$

The results show that Semi-supervised GraPhyR outperforms Supervised-SiPhyR on topology error, achieving near-zero error. This substantial difference can be attributed to the GNN which embeds topological data directly within the framework. Although these (semi-)supervised approaches achieve good performance, they are not practicable. They require access to the optimal solutions, which may be computationally prohibitive to generate across thousands of training data points.

A note must be made on computational time. Solving the DyR problem using Gurobi takes on average 201 milliseconds for BW-33, and 18 seconds for a 205-node grid per instance. Actual computational times vary significantly with varying load conditions which stress grid voltages (Ex. 17-fold increase for the 205-node grid during high load periods [3]). In contrast, the inference time of GraPhyR is only 84 milliseconds for a batch of 200 instances.

TABLE II
SIMULATION RESULTS ON THE BW-33, $\mathcal{G}_1$, AND TPC-94 GRIDS TESTED ON 860/8640 INSTANCES. LOWER VALUES ARE BETTER FOR ALL METRICS.

		Metric					
	Method	Dispatch error (MSE)	Voltage error (MSE)	Topology error	Ineq Viol (mean)	Ineq viol (max)	Num ineq viol > 0.01
(a)	GraPhyR (our method)	2.22e-03	5.55e-03	39.6%	2.33e-03	2.74e-02	20.8
(a)	Global-GraPhyR	1.93e-04	1.03e-03	38.8%	9.86e-04	2.00e-02	4.1
(a)	SiPhyR [3]	2.89e-02	1.69e-03	41.5%	4.79e-04	4.23e-02	5.72
(a)	InSi [3]	3.24e-02	2.30e-03	49.7%	1.53e-03	0.148	16.3
(a)	Semi-supervised GraPhyR	2.48e-03	5.66e-03	1.33%	2.45e-03	3.48e-02	21.2
(a)	Supervised-SiPhyR [3]	5.78e-04	1.35e-03	33.6%	5.49e-04	4.84e-02	7.26
(b)	GraPhyR [BW-33, $\mathcal{G}_1$]	2.64e-03	8.37e-03	42.4%	2.47e-03	2.95e-02	21.9
(c)	sw10 closed GraPhyR	2.48e-03	8.36e-03	31.7%	3.46e-03	7.89e-02	28.6
(c)	sw35 closed GraPhyR	2.53e-03	5.57e-03	37.4%	3.78e-03	7.57e-02	32.5
(c)	sw36 closed GraPhyR	2.37e-03	1.61e-02	45.0%	3.39e-03	5.98e-02	27.9
(c)	sw10 opened GraPhyR	3.90e-03	5.44e-03	32.5%	7.06e-03	1.75e-01	37.2
(c)	sw35 opened GraPhyR	3.45e-03	8.46e-03	53.9%	6.44e-03	1.50e-01	37.3
(c)	sw36 opened GraPhyR	4.25e-03	5.64e-03	28.2%	7.35e-03	1.71e-01	38.8
(d)	GraPhyR [TCP-94]	1.57e-02	1.49e-02	41.9%	1.13e-02	2.56e-01	140
(d)	SiPhyR [TCP-94] [3]	1.12e-02	1.51e-02	45.4%	7.27e-04	4.25e-02	37.5
(d)	InSi [TCP-94] [3]	1.41	3.31e-02	44.3%	3.21e-02	3.01	162

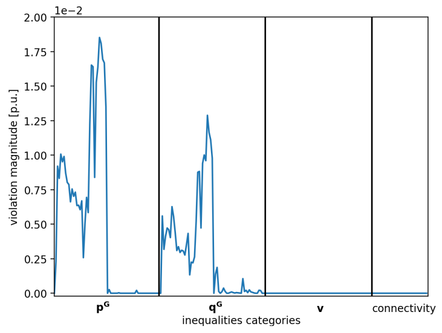

Fig. 5. Magnitude of the inequality constraint violations for GraPhyR. The constraint sets on nodal generation ($\mathbf{p}^{\mathbf{G}}$ and $\mathbf{q}^{\mathbf{G}}$, as in Eq. (8)), voltage limits ($\mathbf{v}$, as in Eq. (9)), and connectivity constraints (Eq. (11)) are separated by black vertical lines.

E. Case (b). Prediction Performance on Multiple Grids

A key feature of GraPhyR is its ability to solve the DyR problem across multiple grid topologies. We construct a dataset using problem instances from both the BW-33 and $\mathcal{G}_1$ grids in Fig. 4. These grids have the same number of nodes, but different topologies, number of lines and switches, and location of switches. We train and test GraPhyR on this dataset. Table II-(b) shows these results, showing the average errors and inequality constraint violations across problem instances of both grids. The performance of GraPhyR on the two grids is similar to that of GraPhyR on a single grid, showing that GraPhyR can learn the power flow representation across multiple topologies and across multiple grids at the same time.

F. Case (c). Adapting to Changing Grid Conditions

We next test GraPhyR on changing grid conditions, such as (un)planned maintenance by the grid operator or switch failure. Since power flows are highly correlated with the grid topology, changes in the set of feasible topologies due to maintenance or equipment failure can significantly change the prediction accuracy. Rather than training on multiple scenarios, we train only on the BW-33 grid for normal operating conditions and test on cases where a switch is required to be open or closed.

Results are shown in Table II-(c). Generally the dispatch error, voltage error, and average inequality violation magnitudes remain similar to cases of normal operation. However, there is a notable increase in the number of inequality violations, and when forcing a switch open, an order of magnitude increase in the maximum inequality violations. Forcing a switch open removes an edge from the GNN graph. The resulting graph is more sparse, reducing access to information during message passing and changing the information contained in the node and switch embeddings.

The topology error is more nuanced. When switch 36 is closed, there is an increase in voltage and topology error. This is because without any operator requirements on switch statuses, switch 36 remains optimally open for all load conditions. Thus, when switch 36 is required to be open, there is a significant decrease in topology error, by almost 10%. Since we did not trained on other scenarios, GraPhyR struggles to optimize the topology and predict voltages when the grid conditions deviate significantly from the training data – such as when switch 36 is closed. Similar performance degradation happens when switch 35 is required to be open; this switch is optimally closed for all load conditions. Interestingly, the status of switch 10 (open or closed) does not affect the topology error, although this switch is typically closed in the training data. There may be multiple (near-)optimal topologies with similar objective value. Regularizing the dataset or performance metrics against these multiple solutions may be necessary to improve prediction performance.

G. Case (d). GraPhyR on a larger grid

We test GraPhyR on the larger TPC-94 grid, using the same GNN parameters ($\mathcal{L} = 4, h = 8$) as in BW-33. Compared to the SiPhyR framework, our GraPhyR approach has comparable performance on optimality metrics and slightly lower topology error. The inequality constraint violations are higher than SiPhyR, but lower than InSi. This result is expected, since the GNN parameters were kept the same as the for $BW - 33$ while the complexity of the problem increased drastically with the grid size and the number of switches. In comparison, the results for SiPhyR and InSi use a larger neural network for TPC-94 than for BW-33. Parameter tuning of the GNN should be done for each network.

H. Utility perspective on DyR

A typical distribution substation may have two to nine distribution feeders per substation. A feeder model may consist of 60 nodes for a medium-sized feeder. The resulting distribution grid model for a single substation may consist of 200-600 nodes. Larger grids may exist in urban load centers. A typical distribution grid may have 10-40 switches (two to nine feeders, with four switches per feeder). These consist of both normally closed switches (NCS) which connect feeders to the substation and across feeders, and normally open switches (NOS) which are typically used for fault location, isolation, and service restoration (FLISR) activities. In this work, we assume both NCS and NOS are available for optimization through DyR. Our GraPhyR framework enables an automated approach to DyR, that can reduce losses as compared to a rule-based approach by 2.5% [3]; for a utility with a 100MW peak load, this translates to savings of US$200,000 per year [23].

V. Conclusion

We developed GraPhyR, an end-to-end physics-informed Graph Neural Network framework to solve the dynamic reconfiguration problem. We model switches as gates in the GNN message passing, embed discrete decisions directly within the framework, and use local predictors to provide scalable predictions. Our simulation results show GraPhyR outperforms methods without GNNs in learning to predict optimal solutions, and offers significant speed-up compared to traditional MIP solvers. Further, our approach adapts to unseen grid conditions, enabling real-world deployment. Future work will investigate the scalability of GraPhyR to larger grids (200+ nodes), approaches to reduce inequality constraint violations, and regularization strategies to improve topology prediction. Finally, further efforts are needed in developing good datasets with representative timeseries data in distribution grids.

References

[1] M. Baran and F. Wu, "Network reconfiguration in distribution systems for loss reduction and load balancing," *IEEE Transactions on Power Delivery*, vol. 4, no. 2, pp. 1401–1407, 1989.

[2] C. Lueken, P. M. Carvalho, and J. Apt, "Distribution grid reconfiguration reduces power losses and helps integrate renewables," *Energy Policy*, vol. 48, pp. 260–273, 2012, special Section: Frontiers of Sustainability. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0301421512004351

[3] R. Haider and A. M. Annaswamy, "Grid-siphyr: An end-to-end learning to optimize framework for combinatorial problems in power systems," 2023, arXiv:2206.06789v3 [eess.SY].

[4] V. Nair, S. Bartunov, F. Gimeno, I. von Glehn, P. Lichocki, I. Lobov, B. O'Donoghue, N. Sonnerat, C. Tjandraatmadja, P. Wang, R. Addanki, T. Hapuarachchi, T. Keck, J. Keeling, P. Kohli, I. Ktena, Y. Li, O. Vinyals, and Y. Zwols, "Solving Mixed Integer Programs Using Neural Networks," 12 2020. [Online]. Available: http://arxiv.org/abs/2012.13349

[5] Q. Cappart, D. Chételat, E. Khalil, A. Lodi, C. Morris, and P. Veličković, "Combinatorial optimization and reasoning with graph neural networks," *Journal of Machine Learning Research*, 2 2023.

[6] J. Zhang, C. Liu, X. Li, H.-L. Zhen, M. Yuan, Y. Li, and J. Yan, "A survey for solving mixed integer programming via machine learning," *Neurocomputing*, vol. 519, pp. 205–217, 2023.

[7] P. Gupta, M. Gasse, E. B. Khalil, P. K. Mudigonda, A. Lodi, and Y. Bengio, "Hybrid models for learning to branch," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[8] C. K. Joshi, T. Laurent, and X. Bresson, "An Efficient Graph Convolutional Network Technique for the Travelling Salesman Problem," 6 2019. [Online]. Available: http://arxiv.org/abs/1906.01227

[9] C. K. Joshi, Q. Cappart, L.-M. Rousseau, and T. Laurent, "Learning the Travelling Salesperson Problem Requires Rethinking Generalization," 6 2020. [Online]. Available: http://arxiv.org/abs/2006.07054 http://dx.doi.org/10.4230/LIPIcs.CP.2021.33

[10] M. Prates, P. H. C. Avelar, H. Lemos, L. C. Lamb, and M. Y. Vardi, "Learning to solve np-complete problems: a graph neural network for decision tsp," in *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, ser. AAAI'19/IAAI'19/EAAI'19. AAAI Press, 2019. [Online]. Available: https://doi.org/10.1609/aaai.v33i01.33014731

[11] J.-Y. Fan, L. Zhang, and J. McDonald, "Distribution network reconfiguration: single loop optimization," *IEEE Transactions on Power Systems*, vol. 11, no. 3, pp. 1643–1647, 1996.

[12] H. J. Wang, J. S. Pan, T. T. Nguyen, and S. Weng, "Distribution network reconfiguration with distributed generation based on parallel slime mould algorithm," *Energy*, vol. 244, 4 2022.

[13] J. Mendoza, R. Lopez, D. Morales, E. Lopez, P. Dessante, and R. Moraga, "Minimal loss reconfiguration using genetic algorithms with restricted population and addressed operators: real application," *IEEE Transactions on Power Systems*, vol. 21, no. 2, pp. 948–954, 2006.

[14] J. A. Taylor and F. S. Hover, "Convex models of distribution system reconfiguration," *IEEE Transactions on Power Systems*, vol. 27, no. 3, pp. 1407–1413, 2012.

[15] N. V. Kovački, P. M. Vidović, and A. T. Sarić, "Scalable algorithm for the dynamic reconfiguration of the distribution network using the Lagrange relaxation approach," *International Journal of Electrical Power and Energy Systems*, vol. 94, pp. 188–202, 1 2018.

[16] Z. N. Popović and N. V. Kovački, "Multi-period reconfiguration planning considering distribution network automation," *International Journal of Electrical Power & Energy Systems*, vol. 139, p. 107967, 7 2022.

[17] O. B. Kundačina, P. M. Vidović, and M. R. Petković, "Solving dynamic distribution network reconfiguration using deep reinforcement learning," *Electrical Engineering*, vol. 104, no. 3, pp. 1487–1501, 6 2022.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[19] C.-T. Su, C.-F. Chang, and J.-P. Chiou, "Distribution network reconfiguration for loss reduction by ant colony search algorithm," *Electric Power Systems Research*, vol. 75, no. 2, pp. 190–199, 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378779605001021

[20] A. Reuther, J. Kepner, C. Byun, S. Samsi, W. Arcand, D. Bestor, B. Bergeron, V. Gadepally, M. Houle, M. Hubbell, M. Jones, A. Klein, L. Milechin, J. Mullen, A. Prout, A. Rosa, C. Yee, and P. Michaleas, "Interactive supercomputing on 40,000 cores for machine learning and data analysis," in *2018 IEEE High Performance extreme Computing Conference (HPEC)*. IEEE, 2018, pp. 1–6.

[21] R. W. Hamming, *Coding and Information Theory (2nd Ed.)*. USA: Prentice-Hall, Inc., 1986.

[22] Y. Cao and V. M. Zavala, "A sigmoidal approximation for chance-constrained nonlinear programs," 2020, arXiv:2004.02402 [math.OC].

[23] R. Ardis and R. Uluski, "CVR is here to stay," *T & D World*, 2015.