PhML-DyR: A Physics-Informed ML framework for Dynamic Reconfiguration in Power Systems

Rabab Haider
Department of Mechanical Engineering
Massachusetts Institute of Technology
Cambridge, MA 02149
rhaider@mit.edu

Anuradha M. Annaswamy
Department of Mechanical Engineering
Massachusetts Institute of Technology
Cambridge, MA 02149
aanna@mit.edu

Abstract

A transformation of the US electricity sector is underway with aggressive targets to achieve 100% carbon pollution-free electricity by 2035. To achieve this objective while maintaining a safe and reliable power grid, new operating paradigms are needed, of computationally fast and accurate decision making in a dynamic and uncertain environment. We propose a novel physics-informed machine learning framework for the decision of dynamic grid reconfiguration (PhML-DyR), a key task in power systems. Dynamic reconfiguration (DyR) is a process by which switch-states are dynamically set so as to lead to an optimal grid topology that minimizes line losses. To address the underlying computational complexities of NP-hardness due to the mixed nature of the decision variables, we propose the use of physics-informed ML (PhML) which integrates both operating constraints and topological and connectivity constraints into a neural network framework. Our PhML approach learns to simultaneously optimize grid topology and generator dispatch to meet loads, increase efficiency, and remain within safe operating limits. We demonstrate the effectiveness of PhML-DyR on a canonical grid, showing a reduction in electricity loss by 23%, and improved voltage profiles. We also show a reduction in constraint violations by an order of magnitude as well as in training time using PhML-DyR.

1 Introduction

Power systems are an engineering marvel. They are quintessential illustrations of a well functioning large-scale system that safely, reliably, and affordably delivers an essential service, electricity. In recent years, essential properties of this large scale system have expanded to include sustainability and green, precipitated by urgent and intense concerns of climate change [National Academies of Sciences, Engineering, and Medicine, 2021]. The result has been a transformation of the grid edge, with increasing penetration of distributed energy resources (DER), which include, among others, roof-top solar PV, community solar, batteries of various sizes, and electric vehicles.

The opportunity to provide new, green sources of power necessitates new decision-making paradigms. Current practices of decision making in power systems consist of static optimization tools at slower time-scales for applications in planning and markets, and dynamic tools at faster time-scales for applications in prediction, estimation, and control. The proliferation of DERs introduces dynamic signatures across the power system landscape, pushing for dynamic decision making. Dynamic decisions require new tools for computationally faster and accurate decisions in the presence of operating constraints. It is in this regard that machine learning (ML) is being investigated.

An important decision in power systems involves grid reconfiguration, a highly dynamic phenomenon. Grid reconfiguration involves the selection of switch-states (open/closed) in the grid to ensure that
loads are met with available generation resources, while satisfying voltage and line flow operating constraints. At the transmission (bulk) level, reconfiguration is a typical response to contingencies, failure of a generator or line. At the distribution level, reconfiguration can be used to reduce electrical losses involved in carrying power from bulk resources and DERs to individual loads (commercial businesses, residential units). The latter, increasing efficiency in the distribution grid with high DER penetrations, is the focus of this work. Static reconfiguration (StatR) determines a fixed set of switch-states that optimize losses over a long term. But the introduction of DERs necessitates dynamic reconfiguration where local DERs supply loads in closer proximity to them, thus reducing losses and improving voltage profiles across the grid. Dynamic reconfiguration (DyR) identifies the switch-states (i.e. grid topology) which minimizes losses for the given load and generation conditions. This is not possible with either traditional power systems approaches which rely on heuristics and operator experience, or traditional optimization approaches which can be computationally intractable.

This paper demonstrates a novel use of physics-informed ML (PhML) suitably modified to incorporate physical constraints present in the reconfiguration problem, both equality and inequality, and optimize a relevant cost function. More importantly, this optimization involves solving a mixed integer problem (MIP), a combination of both continuous and binary variables. Exactly solving MIPs, even those with only linear constraints, is an NP-hard problem, requiring exponential time to solve. Thus using traditional optimization methods for DyR is intractable, especially when requiring fast decisions and high degree of accuracy to remain within operating constraints. Further, traditional solvers cannot take advantage of the structures present in repeatedly solving an optimization problem, and warm-start techniques may struggle when parameters vary rapidly - such as with solar generation forecasts. Instead, ML can be used to learn the underlying problem structure, and when modified in a physics-informed approach, can provide good and fast solutions which satisfy operating constraints. We incorporate in PhML-DyR, a novel physics-informed rounding heuristic and overcome the challenges of the mixed decision variables. Using PhML-DyR, we carry out a study of a 33-node grid with 7 switches, and 25% DER penetration in the form of community solar. The proposed PhML-DyR framework leads to a 23% reduction in line losses as compared to no reconfiguration, and 2% reduction in voltage violations throughout the year as compared to StatR. Our physics-informed rounding significantly improves the training and performance of the ML framework, with order of magnitude reduction in inequality constraint violation as well as training time.

Our contributions can be summarized as follows: (1) PhML-DyR: An ML framework for dynamic reconfiguration, which carries out mixed integer optimization employing a novel physics-informed rounding heuristic. PhML-DyR is shown to reduce inequality constraint violations by an order of magnitude, eliminate line flow violations, and improve training performance; and (2) Case study on a canonical grid: PhML-DyR results in upwards of 23% reduction in line losses as compared to no reconfiguration, and 2% fewer voltage violations throughout the year as compared to StatR.

2 Related Work

Traditional Methods for Reconfiguration The reconfiguration problem has been extensively studied in the context of transmission grids, where normally closed switches (NCS) and normally open switches (NOS) are used to ensure demand is met without violating line thermal constraints, and during contingencies. Classical literature employ reconfiguration for loss reduction [Baran and Wu, 1989], using single loop optimization [Fan et al., 1996] and heuristics for approximating losses without extensive power flow calculations [Civanlar et al., 1988]. With increased computing abilities and successful commercial MIP software, literature in the 2000’s focused on the modeling of the reconfiguration problem as a convex problem [Taylor and Hover, 2012], with a particular focus on the radiality constraint of the resulting grid topology [Lei et al., 2020, Wang et al., 2020, Ahmad and Martí, 2015, Lavorato et al., 2012], to be solved by classical optimization methods. Despite the advances in MIP solvers, reconfiguration for realistic power grids remained computationally intractable [Silva et al., 2021]; instead literature has proposed using various heuristic methods (including genetic, particle swarm, fuzzy multi-objective, and harmony search algorithms), and other iterative heuristics involving constraint elimination and repeatedly solving smaller sub-optimization problems [Crozier et al., 2022]. However, none of these techniques provide guarantees for optimality, and continue to be computationally prohibitive in dynamic applications.
ML for Power Systems  There is a growing body of literature employing ML techniques for various tasks in power systems, including optimal power flow (OPF) Pan et al. [2019], Zhao et al. [2020], Donti et al. [2021], Zhou et al. [2020], Dobbe et al. [2020], Fioretto et al. [2020], Zamzam and Baker [2020], Chen et al. [2022], probabilistic power flow Yang et al. [2020], security constrained unit commitment Gutierrez-Martinez et al. [2011], Pan et al. [2021], Xavier et al. [2020], fault isolation Li et al. [2019], and reconfiguration Junlakarn and Ilic [2014], Yin et al. [2020], Yang and Oren [2019], Subramanian et al. [2021]. These works all leverage the significant reduction in computational runtimes of ML-based methods as compared to classical methods. ML enables practical implementation of online decision making involving complex mathematics (nonconvex equations versus linear), frequent decision making (dynamic decisions versus static), and higher accuracy (as compared to heuristic methods). Physical constraints of voltage, line flow, and generator limits can be incorporated with loss functions with Lagrangian duals Fioretto et al. [2020], projection algorithms to ensure feasibility Pan et al. [2019], Zhao et al. [2020], and gradient-based algorithms to learn feasibility Donti et al. [2021]. Prior works show that ML-based approaches can often improve upon existing model-based and optimization-based approaches. The order of magnitude improvement with fast ML-based OPF compared to classical methods using state-of-art solvers is of note Pan et al. [2019]. We further argue that physics-informed ML and heuristics have the potential to significantly improve the reliable, affordable, and green operations of power systems, even beyond what systems currently experience. For example, leveraging domain knowledge of decoupled power flow can improve the computational performance of neural network weight updates Yang et al. [2020]. Note that in the papers mentioned above, the underlying decision variables have been continuous variables. Power systems problems frequently experience decisions in the mixed-integer space, with switch-states in reconfiguration being key levers for managing power flow, especially at the grid edge. In this context of reconfiguration, current practice has been to use decision trees and other heuristic classification methods. However, other ML methods such as neural networks have shown to perform better Yang and Oren [2019]. In this paper, we not only use ML but carefully integrate it with physics-informed tools, leading to a novel framework with significant improvement in performance, termed PhML-DyR.

ML for Optimization  The need for fast and repeated solutions to optimization problems has pushed the community towards ML-based approaches over the recent years. While some focus has been on learning new algorithms for optimization Li and Malik [2016] or directly optimizing within a neural framework Donti et al. [2021], the vast majority of effort has been in improving the performance of existing optimization solvers and reducing computational time. This includes hyperparameter optimization Maclaurin et al. [2015], identifying active constraint sets for a general optimization problem Misra et al. [2021] and for OPF Deka and Misra [2019], learning warm start techniques Baker [2019], Xavier et al. [2020], developing an understanding of strategies used by optimization solvers Bertsimas and Stellato [2021], and for the class of MIPs, the design of primal heuristics Wang et al. [2022], Shen et al. [2021], including neural diving Nair et al. [2020] and neural branching Gasse et al. [2019], Gupta et al. [2020]. Recent surveys on advances in ML approaches for MIP include a review of variable branch selection, cutting plane methods, and heuristics such as feasibility pump algorithms Zhang et al. [2022]. Another survey focuses on recent advances in employing graph neural networks for combinatorial optimization (of which MIPs are a member), either directly as solvers or by enhancing exact solves Cappart et al. [2021]. These surveys and references within present a detailed report of the state of art. Within the space of ML for MIPs, few works look at applying techniques to directly optimize (rather than improving existing techniques). In PhML-DyR, we learn how to optimize the reconfiguration MIP directly using an unsupervised neural network, and employ a physics-informed rounding heuristic to significantly improve prediction performance.

3 ML-based Reconfiguration

We propose PhML-DyR, a physics-informed ML-based dynamic reconfiguration framework. It is composed of a neural network which is suitably modified to incorporate grid physics, operating constraints, and salient topological and connectivity constraints. Figure 1 shows the proposed PhML-DyR framework. The unsupervised neural network is composed of 5 key components:

- Neural network: a simple neural network with a sigmoidal output layer. The neural network predicts the independent variables, denoted as $\hat{z}$, from the input data $x$. The output of the
neural network is divided into two sets: predictions for the switch-states $\hat{z}_\tau$, and the other independent variables $\hat{z}_\tau$. Note $\hat{z}_\tau$ consists only of binary variables; $\hat{z}_\tau$ is mixed integer.

- Topology selection layer: the physics-informed rounding heuristic recovers integer solutions to determine switch-states (open/closed). This layer embeds a necessary constraint to maintain topology radiality and node connectivity, as required by grid operators, and permits the neural network to simultaneously learn optimal grid topology and DER dispatch. The input to this layer is the subset of independent variables predicted by the neural network pertaining to switch-states, $\hat{z}_\tau$; the output are the rounded variables now satisfying the integer constraint, denoted as $z_\tau$.

- Inequality constraint layers: the prediction from the neural network $\hat{z}_\tau$ are scaled onto box constraints, using a sigmoidal function mapping, with bounds selected as per the box constraints. This ensures that inequality constraints pertaining to the independent variables are have certified satisfiability. This layer acts in parallel to the topology selection layer. The output of this layer is the scaled variables $z_\tau$.

- Equality constraint layers: leveraging techniques for variable space reduction from optimization literature, the equality constraints describing the power flow across the grid are used to calculate the dependent variables $\phi$ from the independent variables $z = [z_\tau \ z_{\tau}]$.

- Loss function and backpropagation: the neural network is trained to learn to optimize by selecting a loss function which is composed of the objective function to be minimized during optimization (ex. line losses in the grid) and any regularization to bias the solution against violating physical constraints (typically inequality constraints of dependent variables). This is done by way of a soft-loss penalty.

In what follows, we detail the variable decomposition of the independent and dependent variables, the physics-informed rounding heuristic for topology selection, the equality constraints describing power flow, and the loss function for network training.

### 3.1 Variable decomposition

The underlying problem can be cast in the form a standard constrained optimization problem

$$\min_Y f(Y) \quad \text{subject to:} \quad g(Y) = 0; \quad h(Y) \leq 0$$

In (1), the decision variables $Y$ are decomposed as $Y = [x, z, \phi]$, where $x$ is the training data and input to the neural network, $z$ is the reduced variable space predicted by the neural network (independent variables), and $\phi$ are the remaining dependent variables calculated using the equality constraints. In general, this decomposition is non-unique. It critically depends on the structure of the given problem, which determines the relationship between the sets of variables, and requires domain knowledge to exploit the underlying problem structure to produce good solutions. As will be seen below, a further decomposition of the independent and dependent variables is introduced, to accommodate the goal of reconfiguration, into continuous and binary variables, denoted by superscripts C and B, respectively.

The decomposition in the context of reconfiguration is summarized below:

$$x = [\{P_{j}^L, Q_{j}^L \mid j \in B \setminus \#\}]$$

$$z^B = [z_{ji}, \{y_{ij} \mid (i,j) \in T^sw \setminus \{-1\}]$$

$$z^C = [\{v_{j} \mid j \in B \setminus \#\}, P_{j}, P_{ij}, Q_{j}, \{Q_{ij} \mid (i,j) \in T^sw \}, P_{j}^L, Q_{j}^L]$$

---

**Figure 1:** PhML-DyR: A physics-informed ML-based dynamic grid reconfiguration framework.
\[ \varphi^B = \{ z_{ij}, \{ y_{ij} \mid (i, j) \in T_{D_w} \} \} \]
\[ \varphi^C = \{ P^G_j, Q^G_j, \{ Q_{ij} \mid (i, j) \in T_D \} \} \]

In [2], we have denoted a general distribution grid as a graph \( \Gamma(B, T_D) \), where \( B := \{1, \ldots, N\} \) is the set of N nodes, \( T_D := \{(i, j)\} \) is the set of M edges where \( T_{D_w} \subset T_D \) are the set of lines with switches, and \( M_{sw} \) is the number of switches; node \( j^\# \) is the point of common coupling (PCC) of the distribution grid to the bulk transmission grid; \( P^L_i, Q^L_i \) are real and active power loads at every node \( j \); \( P^G_i, Q^G_i \) are real and active power generation at every node \( j \), and \( \phi^j \) indicates import from the bulk transmission grid; \( P^j_i, Q^j_i \) are the directed real and reactive power flows through a distribution line \( (i, j) \in T_D \); \( v \) denotes the squared magnitude of voltage at every node \( j \); \( z_{ij}, z_{ji} \) are binary variables indicating the direction of power flow through a line \( (i, j) \in T_D \); \( y_{ij} \) is a binary variable indicating the switch status \( \forall (i, j) \in T_{D_w} \subset T_D \), where \( y_{ij} = 1 \) is closed and \( y_{ij} = 0 \) is open. The set minus notation \( (i, j) \in T_{D_w} \setminus \{-1\} \) denotes all the switches in the grid except the last one, and the indexing notation \( T_{D_w}^{-1} \) denotes the last switch. The set of independent and dependent variables are \( z = [z^B, z^C] \) and \( \varphi = [\varphi^B, \varphi^C] \) respectively.

We also classify \( z_\tau \) and \( z_\tau \setminus \tau \) as the set of independent switch variables and the set of other variables, respectively. We note that \( z_\tau \subset z^B \), and further, \( z_\tau \subset y_{ij} \). Such a classification enables us to develop a physics-informed rounding, to be applied to the switch variables \( z_\tau \). This is detailed next.

### 3.2 Physics-Informed Rounding

A key challenge in the DyR problem with simultaneous topology selection and DER dispatch problem is the mixed integer nature. Traditional optimization literature deals with mixed integer programming using an array of heuristic methods involved in developing good upper and lower bounds, pruning solution branches, and selecting variables to round to nearest integer solutions. Taking inspiration from the class of rounding heuristics well-established in the MIP literature (Jünger et al. [2010], and others including Goemans and Williamson [1995], Marchand and Wolsey [2001], Hifi and Sadeghsa [2022]), we propose a physics-informed rounding heuristic. In what follows we describe key equations governing grid topology, and then describe our rounding heuristic.

Distribution grids in the US are operated with a radial structure, and during normal operation, all nodes must be connected. Various mathematical formulations of these radiality and connectivity constraints include constraints on the determinant of the branch-to-node incidence matrix or spanning tree constraints and other graph theoretic approaches [Lei et al. [2020], Wang et al. [2020], Ahmadi and Martí [2015], Lavorato et al. [2012]]. However, many of these suffer from high computational requirements and additional complexity, and do not leverage the fact that grid connectivity can be ensured by power flow constraints under normal operation. We integrate these radiality and connectivity constraints in the following manner. We first define \( L = (N - 1) - (M - M_{sw}) \) where \( N \) is the number of nodes, \( M \) the number of lines, and \( M_{sw} \) the number of switches.

\[ \sum_{(i, j) \in T_{D_w}^u} y_{ij} = L \tag{3} \]
\[ \sum_{j: (i, j)} z_{ij} + z_{ji} \geq 1 \quad \forall j \in B \tag{4} \]

Constraint (3) restricts the number of closed switches in the grid so it is radial with \( N - 1 \) total branches, where \( L \) switches must be closed and the remaining \( M_{sw} - L \) must be open. Constraint (4) enforces connectivity by requiring power to flow into or out of a node along at least one line. It should be noted that typical reconfiguration problem statements also include an arborescence constraint [Taylor and Hover [2012]], either explicitly or implicitly in the formulation of the radiality constraint. However, the increasing penetration of DERs voids this assumption, and multiple generating sources (roots of the tree) must be permitted. We have relaxed this arborescence constraint in [4].

With the above constraints, we now use this explicit knowledge of \( L \) to inform our topology selection. In particular, we use \( L \) to round \( \tilde{z}_\tau \) to integer values \( z_\tau \). We describe the algorithm next, and formally present it in Algorithm 1. The neural network predicts the status of all but one switch, and is described by \( \tilde{z}_\tau \). Of these \( M_{sw} - 1 \) switches, at least \( L - 1 \) must be closed. The remaining \( L \)-th switch that is closed can belong either to \( \tilde{z}_\tau \) or to \( \varphi_\tau = \{ y_{ij} \mid (i, j) \in T_{D_w}^{-1} \} \). We then close \( L - 1 \) switches in \( \tilde{z}_\tau \) by setting the corresponding \( z_\tau = 1 \), and we open \( M_{sw} - L - 2 \) switches in \( \tilde{z}_\tau \) by setting the
corresponding $z_\tau = 0$. For the remaining two switches, the neural network training guides $y_{ij}$ to integer solutions. This physics-informed rounding heuristic then selects a feasible (read: radial and connected) grid topology upon which the power flow describing the relationship between $z^C$ and $\varphi^C$ are satisfied. This enables simultaneous optimization of grid topology and DER dispatch.

**Algorithm 1:** Physics-Informed Rounding for Topology Selection

**Data:** $\tilde{z}_\tau$

**Result:** Binary variables $z_\tau$ for switch-state prediction and topology selection initialization: $L = (N - 1) - (M - M_{sw})$

Sort $\tilde{z}_\tau$ in ascending order; assign $\hat{I}_\tau$, the sorted indices of $\tilde{z}_\tau$

Assign switches to be closed: $z_\tau \left[ \hat{I}_\tau^{1:L} \right] = \max \{ \tilde{z}_\tau \left[ \hat{I}_\tau^{L} \right], 1 \}$

Assign switches to be open: $z_\tau \left[ \hat{I}_\tau^{L+1:M_{sw}-1} \right] = \min \{ \tilde{z}_\tau \left[ \hat{I}_\tau^{L+1:M_{sw}-1} \right], 0 \}$

### 3.3 Equality constraints

We describe the distinction between $z$ and $\varphi$ variables, which stem from equality constraints from power flow. We use the LinDistFlow model of the distribution grid Baran and Wu [1989], given by:

\[
P_j^G - P_j^L = \sum_{k \in \{j,k\}} P_{jk} - \sum_{i \in \{i,j\}} P_{ij} \quad \forall j \in B \setminus j^# \tag{5}
\]

\[
Q_j^G - Q_j^L = \sum_{k \in \{j,k\}} Q_{jk} - \sum_{i \in \{i,j\}} Q_{ij} \quad \forall j \in B \setminus j^# \tag{6}
\]

\[
v_j - v_i = -2(R_{ij}(P_{ij} - P_{ji}) + X_{ij}(Q_{ij} - Q_{ji})) \quad \forall (i, j) \in T_D \setminus T_D^{uw} \tag{7}
\]

\[
z_{ij} + z_{ji} = 1 \quad \forall (i, j) \in T_D \setminus T_D^{uw} \tag{8}
\]

\[
z_{ij} + z_{ji} = y_{ij} \quad \forall (i, j) \in T_D^{uw} \tag{9}
\]

Constraints (5)-(7) describe power flow using the LinDistFlow model. Constraint (7) is Ohm’s law across all lines which do not have switches, and constraints (5)-(6) are lossless power balance at every node. These equality constraints are used to calculate $z^C$ from the independent variables $z^C$. These correspond to $P_j^G$ from constraint (5), $Q_{ij}$ for non-switch lines from constraint (6), and $Q_j^G$ from constraint (7). A corresponding set of equations describe Ohm’s law across switched line, and are represented as big-M relaxations of the conditional constraints; these are inequality constraints and do not impact the variable decomposition. The full MILP model is in Appendix A for reference. The next set of constraints (8)-(9) describe topological constraints and are composed of integer variables, used to calculate $\varphi^C$ from $z^C$. In particular, the corresponding dependent variables $z_{ij}$ are calculated from constraints (8)-(9), and the status of the last switch is calculated from (3).

It should be noted that the equality constraints of the selected LinDistFlow model are linear, and therefore the dependent variables can be determined trivially with zero error. For problems which involve more complex (potentially nonlinear) equality constraints, such as nonlinear DistFlow, the same variable space reduction techniques can be used, and programs like Newton’s method can be leveraged to solve for the dependent variables. In the backpropagation step through the equality constraint layer, the Jacobians describing the derivatives can be explicitly written out and the implicit function theorem used to backpropagate through the dependent variables.

Here we highlight a key feature of our variable decomposition, and the resulting certified satisfiability of inequality constraints describing voltage, line flow, and generator limits. The particular variable decomposition in (2) and the selection of which variables belong in $z$ and $\varphi$ are critical to satisfying inequality constraints in PhML-DyR. By selecting voltages $v$ as an independent variable, these are scaled onto the box constraints describing operating limits; for a grid operator, this means voltage limits across the grid will always be satisfied, a critically important aspect of power systems operation. This is inherent in our proposed structure, as compared to other methods which rely on projections, clipping, or penalties to enforce voltage constraints.
3.4 Loss function

The loss function for the dynamic reconfiguration problem is chosen to minimize the power loss over the lines, which is approximated as $f_x(z, \varphi) = \sum_{(i,j) \in T_{\text{NOS}}} (P_{ij}^2 + Q_{ij}^2) R_{ij}$. The soft loss penalizes inequality constraint violation on dependent variables with hyperparameter $\lambda_h$. Note that LinDistFlow power flow model is composed of all linear constraints, so the completion step is trivial and results in no equality constraint violations. For nonlinear power flow models, a penalty for equality constraints can also be included. The resulting convex loss function is $l = f_x(z, \varphi) + \lambda_h \max \{0, h_x(z, \varphi)\}$. 

4 Experiments

Dataset: We test the proposed PhML-DyR on a 33-node test system commonly used in the reconfiguration literature [Baran and Wu [1989]]. The grid consists of 33 nodes ($N = 33$), 37 lines ($M = 37$) of which 5 are tie lines (NOS) and the remaining 32 are typically assumed to be switches (NCS). The grid is very lossy, with losses up to 8% of total load, and voltage profile violating voltage limits. These characteristics make the 33-node grid an excellent test case for DyR, with the objective function to reduce line losses. Grid data includes the location of loads and their nominal power demand (P and Q) for a single period. To develop a diverse set of training data, the maximum load perturbation at each node is restricted to 70% deviation from nominal value (i.e. $P^r = \delta P, \delta = [0.3, 1.7]$). The power factor of the loads, which describes the relationship between the real and reactive power as $pf = P/(\sqrt{P^2 + Q^2})$, is kept constant to the pf in the nominal data. To restrict the problem to a simpler test case, only a subset of the lines are considered switchable – these include the 5 tie lines (numbered 33 to 37) and 2 NCS lines (line numbers 10 and 26). We add a range of community solar facilities (each <5MW), up to a penetration of 25.3% of nameplate capacity to baseline load. This is a modest DER penetration compared to that which we would expect in the future grid, and reflects solar uptake now and over the next few years. We use renewable generation profiles to couple generation across the grid with weather conditions, with hourly generation data from the NREL SAM tool [National Renewable Energy Laboratory [2020]]. We divide the grid into sections, based on the location of switches, and vary the location of community solar farms amongst these sections. We denote the distribution of these DERs (DD) as follows: (i) DD-U: uniform distribution of solar with a differentiable relaxation of the step function, using a steep sigmoid activation function at the location of switches, and 2 NCS lines (line numbers 10 and 26). We add a range of community solar farms amongst these sections. (ii-iv) DD-I, DD-II, DD-III: all facilities are in Sections I, II, or III of the grid separately. 

Simulation Parameters: We use PyTorch on a 2.3GHz Dual-Core Intel Core i5, with 8GB RAM. The neural network has two hidden layers, each with 25 neurons. The size of input and output layer is determined by $x$ and $\hat{z}$. Each layer applies a linear transformation with bias, batch normalization, ReLU activation, and 50% dropout. We use He initialization for the weights. Backpropagation uses an adaptive learning rate (Adam), with parameter $\gamma = 0.001$. The soft loss penalty on inequality constraint violation is set to $\lambda_h = 100$, chosen to enforce a high penalty on violating inequality constraints, while still allowing the underlying objective function of minimizing network losses to be improved upon. There are 8760 data points, split as 80% training, and 10% testing and validation each. We use mini-batching with 200 batch size. 

Inequality constraint violation: Violation of inequality constraints are measured as: (i) $error_{\text{ineq-tol}} = \sum I_{h_i(x,z,\varphi)} > \text{tol}$ with tol = 0.01 a tolerance threshold; and (ii) $error_{\text{ineq-mean}} = \frac{1}{n} \sum_h h_i(x, z, \varphi)$ the mean violation of an inequality constraint $h_i$ across $n$ data points. 

Sigmoidal activation for integer variables (SigInt): We compare our physics-informed rounding with a differentiable relaxation of the step function, using a steep sigmoid activation function at the output layer: $\sigma_{\beta}(z) = \left[ \frac{2^{1+\mu}}{\mu+e^{-z/\tau}} - 1 \right]_+$, where $\tau, \mu$ are free parameters [Cao and Zavala [2020]]. This sigmoid activation is used for the binary variables $\hat{z}^B$, while the continuous variables $\hat{z}^C$ still pass through the traditional sigmoid function. We set the parameters $\mu = 1$ and $\tau = 5$, where $\tau$ governs the sharpness of the sigmoidal function - and hence how well the sigmoid approximates the

---

1 Allowable deviation from nominal voltage under normal conditions vary globally. ANSI C84.1 (North America) permits ±5% deviation (0.95pu minimum). IEC and European EN 50160 (Europe) permits ±10% deviation (0.9pu minimum).
Table 1: Loss reduction using StatR and DyR

| DER Locations | StatR vs. no reconfig | PhML-DyR vs. StatR |
|---------------|-----------------------|--------------------|
|               | % Loss reduction, MW saved per year | MW saved per year |
| No DERs       | 23%, 370 MW           | 0 MW               |
| DD-U          | 20%, 300 MW           | 23 MW              |
| DD-I          | 20%, 320 MW           | 31 MW              |
| DD-II         | 19%, 270 MW           | 20 MW              |
| DD-III        | 21%, 310 MW           | 27 MW              |
| DD-II+III     | 20%, 280 MW           | 28 MW              |

*The MW (power) saved is equal to the MWh (energy), as the simulation is run for every hour of the year and loss reduction summed for every test case.

Gradient-based correction (Corr): We implement the gradient-based correction procedure proposed in [Donti et al. 2021] to accommodate inequality constraints after the variable space completion step. The correction layer uses gradient descent with a learning rate of $\alpha_{\text{corr}} = 10^{-6}$, momentum $\beta_{\text{corr}} = 0.5$, and minimizes the inequality constraint violation given by $\|\max \{0, h_z(z, \varphi)\}\|_2^2$ subject to the equality constraints $g_x(z, \varphi) = 0$. The value of $\alpha_{\text{corr}}$ was chosen through simulation to retain stability of the gradient descent algorithm.

4.1 Performance of PhML-DyR

Table 1 shows the significant loss reduction enabled by reconfiguration of the 33-node distribution grid as DER locations are varied, upwards of 23%. The StatR closes tie line 35 and opens NCS 10, and keeps this topology fixed. The DyR selects primarily between two states: (a) closing tie line 35 and opening NCS 10, and (b) closing tie lines 35 and 36 and opening NCS 10 and 26. For a grid without any DERs versus a grid with DERs, the loss reduction from reconfiguration is higher without DERs; this can be attributed to greater losses without leveraging local generators which are located closer to loads and thus incur lower losses when supplying those loads. The second column compares DyR to StatR, showing savings up to 30 MW for a single distribution feeder.

While this is a modest 2.5% improvement of DyR upon StatR, this is a nontrivial reduction for distribution grid operators and utilities. Current industry standard methods which aim to reduce the load on a distribution grid, such as Conservation Voltage Reduction (CVR), typically reduce peak demand by a modest 2-2.5%; for a utility with a peak load of 100 MW, this translates to savings of US$200,000 per year [Ardis and Uluski 2015]. Further, it should be noted that this was obtained with a small test case (33 nodes). As the dimension increases, with increasing penetration of DERs and switches, with disparate patterns and topologies, it is expected that this difference will be significantly larger.

Figure 2 plots the voltage distribution across the grid for an entire year. The ideal distribution is the shape of a short ice cream cone - wide on top and narrow on the bottom, with the tip above the line indicating the ANSI minimum voltage limit of 0.95pu. On each plot is printed the percentage of time the voltages are within the ANSI limits, where higher numbers are better. We make the following key observations: (i) without reconfiguration, the grid performs very poorly, violating ANSI limits 50-60% of the time, and voltages drop to 0.88pu (outside of ANSI limits); (ii) reconfiguration (Stat or Dy) significantly improves the voltages across the grid, with minimum voltage improving to 0.9pu, and ANSI limits satisfied 77-83% of the time (IEC limits are always satisfied); (iii) DyR reduces the number of voltage violations throughout the year by 2%, as compared to StatR, which is a significant improvement as undervoltage can result in brownouts and even lead to blackouts. We anticipate that larger test cases will similarly show greater improvement in the voltage profile with DyR, as with the line losses. We note that since the grid chosen for this test case is very lossy, in our simulations we enforce a lower voltage limit of 0.87pu to ensure feasibility of loading conditions (instead of 0.95pu), which our PhML-DyR framework always satisfies. While it is interesting to note the simple case study of 33-nodes does not imply a preferred DD over others, higher dimensions will expect to naturally suggest an optimal DD.
4.2 Performance of Physics-Informed Rounding

Figure 3 compares the performance of PhML to SigInt-ML for DyR, where PhML employs physics-informed rounding. In Fig. 3a versus 3e, we see an order of magnitude reduction in $\text{error}_{\text{ineq}} - \text{tol}$ when using PhML, with PhML outperforming the SigInt-ML within the first 50 epochs. Fig. 3e also shows that Corr provides little reduction in $\text{error}_{\text{ineq}} - \text{tol}$. Figure 3b and 3f plot $\text{error}_{\text{ineq-mean}}$ averaged over all test data and all epochs, once again showing order of magnitude reductions in error when using PhML, from a maximum error of 10 to 0.06. A similar result is shown in Fig. 3c and 3g which plots $\text{error}_{\text{ineq-mean}}$ for a single test point after the neural network has been trained. With the use of physics-informed rounding, inequality violations of line flow limits are eliminated, and errors in generator limits are significantly reduced. Finally, Fig. 3d and 3h show the test loss over the epochs. From these plots, it’s clear to see that the PhML shows significantly better training performance and reduced training time as compared to SigInt-ML. The interpretation of these results is as follows: at each epoch and for every training point, the physics-informed rounding selects a feasible grid topology, after which the network can optimize the line losses and inequality violation. Notably, without the rounding heuristic, the network struggles to learn both integer and continuous variables, preventing any meaningful progress in satisfying inequality constraints and reducing line losses. We also note that the performance of the PhML-DyR framework is invariant to DD, with the training and test plots looking very similar across all DD scenarios (thus omitted). We also note the difference in computing time: online optimization to solve the reconfiguration MILP using Gurobi with the Yalmip optimizer environment in MATLAB takes 126.1ms (averaged over 500 data points), while PhML-DyR takes 0.422ms (averaged over 876 data points). This incredible speed-up of PhML-DyR over traditional optimization can be attributed to the ability of ML to learn the underlying task of grid reconfiguration, and the reduction of overhead required by optimization solvers in setting up the MILP and executing iterative heuristics to solve for the integer solutions.

5 Conclusion

In this paper we propose a framework, PhML-DyR, for dynamic reconfiguration of the distribution grid. We propose a novel physics-informed rounding to tackle the mixed integer nature of the reconfiguration problem. Our PhML-DyR provides order of magnitude improvement in inequality error and training as compared to other approaches for integer accommodation which do not leverage physics. Our test case on a 33-node grid shows up to 23% reduction in line losses and significant improvements in voltage profile. While our test case is small, leading to modest improvements of DyR over StatR in both loss reduction and voltage violations, we anticipate larger test cases will show significant differences. Future work will further enhance the performance of PhML-DyR by employing representative training datasets for reconfiguration, and case studies on larger grids and with uncertainties in load and generation forecasts.
Figure 3: Results comparing performance of: (top two rows) PhML-DyR which leverages the proposed physics-informed rounding and (bottom two rows) SigInt-DyR with and without Corr.
References

National Academies of Sciences, Engineering, and Medicine. The future of electric power in the United States. Technical report, The National Academies Press, 2021.

M.E. Baran and F.F. Wu. Network reconfiguration in distribution systems for loss reduction and load balancing. *IEEE Transactions on Power Delivery*, 4(2):1401–1407, 1989. doi: 10.1109/61.25627.

Ji-Yuan Fan, Lan Zhang, and J.D. McDonald. Distribution network reconfiguration: single loop optimization. *IEEE Transactions on Power Systems*, 11(3):1643–1647, 1996. doi: 10.1109/59.535709.

S. Civanlar, J.J. Grainger, H. Yin, and S.S.H. Lee. Distribution feeder reconfiguration for loss reduction. *IEEE Transactions on Power Delivery*, 3(3):1217–1223, 1988. doi: 10.1109/61.193906.

Joshua A. Taylor and Franz S. Hover. Convex models of distribution system reconfiguration. *IEEE Transactions on Power Systems*, 27(3):1407–1413, 2012. doi: 10.1109/TPWRS.2012.2184307.

Shunbo Lei, Chen Chen, Yue Song, and Yunhe Hou. Radiality constraints for resilient reconfiguration of distribution systems: Formulation and application to microgrid formation. *IEEE Transactions on Smart Grid*, 11(5):3944–3956, 2020. doi: 10.1109/TSG.2020.2985087.

Ying Wang, Yin Xu, Jiaxu Li, Jinghan He, and Xiaojun Wang. On the radiality constraints for distribution system restoration and reconfiguration problems. *IEEE Transactions on Power Systems*, 35(4):3294–3296, 2020. doi: 10.1109/TPWRS.2020.2991356.

Hamed Ahmadi and José R. Martí. Mathematical representation of radiality constraint in distribution system reconfiguration problem. *International Journal of Electrical Power & Energy Systems*, 64:293–299, 2015. ISSN 0142-0615. doi: https://doi.org/10.1016/j.ijepes.2014.06.076. URL https://www.sciencedirect.com/science/article/pii/S014206151400475X.

Marina Lavorato, John F. Franco, Marcos J. Rider, and Rubén Romero. Imposing radiality constraints in distribution system optimization problems. *IEEE Transactions on Power Systems*, 27(1):172–180, 2012. doi: 10.1109/TPWRS.2011.2161349.

Filipe F. C. Silva, Pedro M. S. Carvalho, and Luís A. F. M. Ferreira. Improving pv resilience by dynamic reconfiguration in distribution grids: Problem complexity and computation requirements. *Energies*, 14(4):830, Feb 2021. doi: 10.3390/en14040830.

Constance Crozier, Kyri Baker, and Bridget Toomey. Feasible region-based heuristics for optimal transmission switching. *Sustainable Energy, Grids and Networks*, 30:100628, 2022. ISSN 2352-4677. doi: https://doi.org/10.1016/j.segan.2022.100628. URL https://www.sciencedirect.com/science/article/pii/S2352467722000157.

Xiang Pan, Tianyu Zhao, and Minghua Chen. Deepopf: Deep neural network for dc optimal power flow. In 2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), pages 1–6, 2019. doi: 10.1109/SmartGridComm.2019.8909795.

Tianyu Zhao, Xiang Pan, Minghua Chen, Andreas Venzke, and Steven H. Low. Deepopf+: A deep neural network approach for dc optimal power flow for ensuring feasibility. In 2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), pages 1–6, 2020. doi: 10.1109/SmartGridComm47815.2020.9303017.

Priya L. Donti, David Rolnick, and J Zico Kolter. DC3: A learning method for optimization with hard constraints. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=V1ZHVxJ6dSS.

Yuhao Zhou, Bei Zhang, Chunlei Xu, Tu Lan, Ruisheng Diao, Di Shi, Zhiwei Wang, and Wei-Jen Lee. A data-driven method for fast ac optimal power flow solutions via deep reinforcement learning. *Journal of Modern Power Systems and Clean Energy*, 8(6):1128–1139, 2020. doi: 10.35833/MPCE.2020.000522.
Roel Dobbe, Oscar Sondermeijer, David Fridovich-Keil, Daniel Arnold, Duncan Callaway, and Claire Tomlin. Toward distributed energy services: Decentralizing optimal power flow with machine learning. *IEEE Transactions on Smart Grid*, 11(2):1296–1306, 2020. doi: 10.1109/TSG.2019.2935711.

Ferdinando Fioretto, Terrence W.K. Mak, and Pascal van Hentenryck. Predicting ac optimal power flows: Combining deep learning and lagrangian dual methods. In *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence*, AAAI 2020 - 34th AAAI Conference on Artificial Intelligence, pages 630–637. AAAI Press, 2020.

Ahmed S. Zamzam and Kyri Baker. Learning optimal solutions for extremely fast ac optimal power flow. In *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pages 1–6, 2020. doi: 10.1109/SmartGridComm47815.2020.9303008.

Yexiang Chen, Subhash Lakshminarayana, Carsten Maple, and H. Vincent Poor. A meta-learning approach to the optimal power flow problem under topology reconfigurations. *IEEE Open Access Journal of Power and Energy*, 9:109–120, 2022. doi: 10.1109/OAJPE.2022.3140314.

Yan Yang, Zhifang Yang, Juan Yu, Baosen Zhang, Youqiang Zhang, and Hongxin Yu. Fast calculation of probabilistic power flow: A model-based deep learning approach. *IEEE Transactions on Smart Grid*, 11(3):2235–2244, 2020. doi: 10.1109/TSG.2019.2950115.

Victor J. Gutierrez-Martinez, Claudio A. Cañizares, Claudio R. Fuerte-Esquivel, Alejandro Pizano-Martinez, and Xueping Gu. Neural-network security-boundary constrained optimal power flow. *IEEE Transactions on Power Systems*, 26(1):63–72, 2011. doi: 10.1109/TPWRS.2010.2050344.

Xiang Pan, Tianyu Zhao, Minghua Chen, and Shengyu Zhang. Deepopf: A deep neural network approach for security-constrained dc optimal power flow. *IEEE Transactions on Power Systems*, 36(3):1725–1735, 2021. doi: 10.1109/TPWRS.2020.3026379.

Álison S. Xavier, Feng Qiu, and Shabbir Ahmed. Learning to solve large-scale security-constrained unit commitment problems. *INFORMS Journal on Computing*, 33(2):739–756, 2020. doi: 10.1287/ijoc.2020.0976.

Wenting Li, Deepjyoti Deka, Michael Chertkov, and Meng Wang. Real-time faulted line localization and pmu placement in power systems through convolutional neural networks. *IEEE Transactions on Power Systems*, 34(6):4640–4651, 2019. doi: 10.1109/TPWRS.2019.297794.

Siripha Junlakarn and Marija Ilić. Toward implementation of the reconfiguration for providing differentiated reliability options in distribution systems. In *2014 IEEE PES General Meeting | Conference Exposition*, pages 1–5, 2014. doi: 10.1109/PESGM.2014.6939082.

Ziyang Yin, Xingquan Ji, Yumin Zhang, Qi Liu, and Xingzhen Bai. Data-driven approach for real-time distribution network reconfiguration. *IET Generation, Transmission & Distribution*, 4(13), March 2020. doi: 10.1049/iet-gtd.2019.1733.

Zhu Yang and Shmuel Oren. Line selection and algorithm selection for transmission switching by machine learning methods. In *2019 IEEE Milan PowerTech*, pages 1–6, 2019. doi: 10.1109/PTC.2019.8810867.

Medha Subramanian, Jan Viebahn, Simon H. Tindemans, Benjamin Donnot, and Antoine Marot. Exploring grid topology reconfiguration using a simple deep reinforcement learning approach. In *2021 IEEE Madrid PowerTech*, pages 1–6, 2021. doi: 10.1109/PowerTech46648.2021.9494879.

Ke Li and Jitendra Malik. Learning to optimize, 2016. URL https://arxiv.org/abs/1606.01885.

Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2113–2122, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/maclaurin15.html.
Sidhant Misra, Line Roald, and Yeessian Ng. Learning for constrained optimization: Identifying optimal active constraint sets. *INFORMS Journal on Computing*, 34(1), September 2021. doi: 10.1287/ijoc.2020.1037. URL https://doi.org/10.1287/ijoc.2020.1037

Deepjyoti Deka and Sidhant Misra. Learning for dc-opf: Classifying active sets using neural nets. In *2019 IEEE Milan PowerTech*, pages 1–6, 2019. doi: 10.1109/PTC.2019.8810819.

Kyri Baker. Learning warm-start points for ac optimal power flow. In *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2019. doi: 10.1109/MLSP.2019.8918690.

Dimitris Bertsimas and Bartolomeo Stellato. The voice of optimization. *Mach. Learn.*, 110(2): 249–277, Feb 2021. ISSN 0885-6125. doi: 10.1007/s10994-020-05893-5. URL https://doi.org/10.1007/s10994-020-05893-5

Akang Wang, Linxin Yang, Sha Lai, Xiaodong Luo, Xiang Zhou, Haohan Huang, Shengcheng Shao, Yuanming Zhu, Dong Zhang, and Tao Quan. Efficient primal heuristics for mixed-integer linear programs, 2022. URL https://arxiv.org/abs/2202.02725.

Yunzhuan Shen, Yuan Sun, Andrew Eberhard, and Xiaodong Li. Learning primal heuristics for mixed integer programs. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2021. doi: 10.1109/IJCNN52387.2021.9533651.

Vinod Nair, Sergey Bartunov, Felix Gimeno, Ingrid von Glehn, Pawel Lichocki, Ivan Lobov, Brendan O’Donoghue, Nicolas Sonnerat, Christian Tjandraatmadja, Pengming Wang, Ravichandra Addanki, Tharindi Hapuarachchi, Thomas Keck, James Keeling, Pushmeet Kohli, Ira Ktena, Yujia Li, Oriol Vinyals, and Yori Zwols. Solving mixed integer programs using neural networks, 2020. URL https://arxiv.org/abs/2012.13349

Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. *Exact Combinatorial Optimization with Graph Convolutional Neural Networks*. 2019.

Prateek Gupta, Maxime Gasse, Elias B. Khalil, M. Pawan Kumar, Andrea Lodi, and Yoshua Bengio. Hybrid models for learning to branch. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS’20, 2020. ISBN 9781713829546.

Jiayi Zhang, Chang Liu, Junchi Yan, Xijun Li, Hui-Ling Zhen, and Mingxuan Yuan. A survey for solving mixed integer programming via machine learning, 2022. URL https://arxiv.org/abs/2203.02878

Quentin Cappart, Didier Chételat, Elias B. Khalil, Andrea Lodi, Christopher Morris, and Petar Veličković. Combinatorial optimization and reasoning with graph neural networks. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4348–4355. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/595. URL https://doi.org/10.24963/ijcai.2021/595 Survey Track.

Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, Nov 1995. ISSN 0004-5411. doi: 10.1145/227683.227684. URL https://doi.org/10.1145/227683.227684

Hugues Marchand and Laurence A. Wolsey. Aggregation and mixed integer rounding to solve mips. *Operations Research*, 49(3), June 2001. doi: 10.1287/opre.49.3.363.11211. URL https://doi.org/10.1287/opre.49.3.363.11211

Mhand Hifi and Shohre Sadeghsa. A rounding strategy-based algorithm for the k-clustering minimum biclique completion problem. *Journal of the Operational Research Society*, February 2022. doi: 10.1080/01605682.2022.2035272. URL https://doi.org/10.1080/01605682.2022.2035272
National Renewable Energy Laboratory. System advisory model. https://sam.nrel.gov/ 2020.

Yankai Cao and Victor M. Zavala. A sigmoidal approximation for chance-constrained nonlinear programs, 2020. arXiv:2004.02402 [math.OC].

Rob Ardis and Robert Uluski. CVR is here to stay. T & D World, 2015.
A Grid reconfiguration problem

The grid reconfiguration problem using the LinDistFlow model of the distribution grid [Baran and Wu 1989] is formulated as

\[
\begin{align*}
\min_y & \quad f(y) \\
\text{s.t.} & \quad v_j - v_i = -2(R_{ij}(P_{ij} - P_{ji}) + X_{ij}(Q_{ij} - Q_{ji})) & \forall (i, j) \in \mathcal{T}_D \setminus \mathcal{T}^{sw}_D \quad (10a) \\
& \quad v_j - v_i \leq -2(R_{ij}(P_{ij} - P_{ji}) + X_{ij}(Q_{ij} - Q_{ji})) + M(1 - y_{ij}) & \forall (i, j) \in \mathcal{T}^{sw}_D \quad (10b) \\
& \quad v_j - v_i \geq -2(R_{ij}(P_{ij} - P_{ji}) + X_{ij}(Q_{ij} - Q_{ji})) - M(1 - y_{ij}) & \forall (i, j) \in \mathcal{T}^{sw}_D \quad (10c) \\
& \quad P_j^G - P_j^L = \sum_{k:(j,k)} P_{jk} - \sum_{i:(i,j)} P_{ij} & \forall j \in \mathcal{B} \setminus j^# \quad (10d) \\
& \quad Q_j^G - Q_j^L = \sum_{k:(j,k)} Q_{jk} - \sum_{i:(i,j)} Q_{ij} & \forall j \in \mathcal{B} \setminus j^# \quad (10e) \\
& \quad z_{i,j} \geq 0 & \forall (i, j), (j, i) \in \mathcal{T}_D \quad (10f) \\
& \quad z_{i,j} + z_{j,i} = 1 & \forall (i, j) \in \mathcal{T}_D \setminus \mathcal{T}^{sw}_D \quad (10g) \\
& \quad z_{i,j} + z_{j,i} = y_{i,j} & \forall (i, j) \in \mathcal{T}^{sw}_D \quad (10h) \\
& \quad \sum_{j:(i,j)} y_{i,j} = (N - 1) - (M - \#sw) & \forall i \in \mathcal{B} \quad (10i) \\
& \quad \sum_{j:(i,j)} z_{i,j} + z_{j,i} \geq 1 & \forall i \in \mathcal{B} \quad (10j) \\
& \quad y_{i,j} \in \{0, 1\} & \forall (i, j) \in \mathcal{T}^{sw}_D \quad (10k) \\
& \quad z_{i,j}, z_{j,i} \in \{0, 1\} & \forall (i, j) \in \mathcal{T}_D \quad (10l) \\
& \quad 0 \leq P_{i,j} \leq M z_{i,j} & \forall (i, j), (j, i) \in \mathcal{T}_D \quad (10m) \\
& \quad 0 \leq Q_{i,j} \leq M z_{i,j} & \forall (i, j), (j, i) \in \mathcal{T}_D \quad (10n) \\
& \quad P_j^L = P_j^{L,0} & \forall j \in \mathcal{B} \quad (10o) \\
& \quad Q_j^L = Q_j^{L,0} & \forall j \in \mathcal{B} \quad (10p) \\
& \quad P_j^G \leq P_j^L \leq \overline{P}_j & \forall j \in \mathcal{B} \quad (10q) \\
& \quad Q_j^G \leq Q_j^L \leq \overline{Q}_j & \forall j \in \mathcal{B} \quad (10r) \\
& \quad v_j \leq v_j \leq \overline{v}_j & \forall j \in \mathcal{B} \setminus j^# \quad (10s) \\
& \quad v_j \# = 1 & \forall j \in \mathcal{B} \setminus j^# \quad (10t)
\end{align*}
\]
\[
\begin{align*}
  z_{i,j#} &= 0 & \text{(no export allowed)} \\
  \sum_{j:(j#)} P_{j#} &= P^G_{j#} & \text{(no export allowed)} \\
  \sum_{j:(j#)} Q_{j#} &= Q^G_{j#} & \text{(no export allowed)}
\end{align*}
\]

In [10], we have denoted a general distribution grid as a graph \( \Gamma(\mathcal{B}, \mathcal{T}_D) \), where \( \mathcal{B} := \{1, ..., N\} \) is the set of N nodes, \( \mathcal{T}_D := \{(i, j)\} \) is the set of M edges where \( \mathcal{T}_D^w \subset \mathcal{T}_D \) are the set of lines with switches, and \( M_{sw} \) is the number of switches; node \( j# \) is the point of common coupling (PCC) of the distribution grid to the bulk transmission grid; \( y = [P^G, P^L, Q^G, Q^L, v, P_{ij}, Q_{ij}, y_{ij}, z_{ij}, z_{ji}] \) is the decision vector for the reconfiguration problem; \( P^L, Q^L \) are real and reactive power loads at every node \( j \); \( P^G, Q^G \) are real and reactive power generation at every node \( j \), and generation at \( j# \) indicates import from the bulk transmission grid; \( P_{ij}, P_{ji}, Q_{ij}, Q_{ji} \) are the directed real and reactive power flows through a distribution line \( (i, j) \in \mathcal{T}_D \); \( v \) denotes the squared magnitude of voltage at every node \( j \); \( z_{ij}, z_{ji} \) are binary variables indicating the direction of power flow through a line \( (i, j) \in \mathcal{T}_D \); \( y_{ij} \) is a binary variable indicating the switch status \( y_{ij} \in \mathcal{T}_D^w \subset \mathcal{T}_D \), where \( y_{ij} = 1 \) is closed and \( y_{ij} = 0 \) is open. The set minus notation \((i, j) \in \mathcal{T}_D^w \setminus \{-1\}\) denotes all the switches in the grid except the last one, and the indexing notation \( \mathcal{T}_D^{sw}_{sw} \) denotes the last switch.

Constraints \([10b]-[10f]\) describe power flow using the LinDistFlow model. Constraints \([10b]-[10d]\) describe Ohm’s law across all lines. The big-M relaxation is used to describe the conditional constraints for line \((i, j) \in \mathcal{T}_D^w\). Constraints \([10e]-[10f]\) describe lossless power balance at every node.

Constraints \([10g]-[10o]\) describe the topology selection through switch-status, grid radiality and connectivity constraints, enforce binary constraints on the \( y_{ij}, z_{ij}, z_{ji} \) variables, and restrict power flow through a line based on switch-status. In particular, constraint \([10j]\) restricts the number of closed switches in the grid so it is radial with \( N - 1 \) total branches, where \( L \) switches must be closed and the remaining \( M_{sw} - L \) must be open. Constraint \([10k]\) enforces connectivity by requiring power to flow into or out of a node along at least one line. Note that the connectivity constraint must be relaxed when a fault has occurred and islanding of sections of the grid is permitted, and for system restoration. It should be noted that typical reconfiguration problem statements also include an arborescence constraint \([Taylor \ and \ Hover\ 2012]\), either explicitly or implicitly in the formulation of the radiality constraint. However, the increasing penetration of DERs voids this assumption, and multiple generating sources (roots of the tree) must be permitted. We have relaxed this arborescence constraint in \([10k]\). Various other mathematical formulations of radiality and connectivity constraints include constraints on the determinant of the branch-to-node incidence matrix or spanning tree constraints and other graph theoretic approaches \([Lei \ et \ al.\ 2020, \ Wang \ et \ al.\ 2020, \ Ahmadi \ and \ Martí\ 2015, \ Lavorato \ et \ al.\ 2012]\). However, many of these suffer from high computational requirements and additional complexity, and do not leverage the fact that grid connectivity can be ensured by power flow constraints under normal operation. Our formulation accounts for this.

Constraints \([10p]-[10a]\) set the loads at node \( j \), which are assumed to be inflexible. Constraints \([10b]-[10s]\) describe generator operating limits, and \([10t]-[10a]\) describes grid voltage limits where the voltage at the PCC is assumed to be fixed at 1pu, as is common practice in power systems. Finally, constraints \([10v]-[10w]\) describe “no export” limits on the PCC, where net generation excess of net load in the distribution grid cannot be injected into the transmission grid. For regions where distribution grids are permitted to export power to the bulk grid, these constraints can be removed.

The MILP detailed above is solved to minimize an objective function \( f(y) \). For a modern distribution grid with high DER penetration, various objectives are sought after by grid operators.

Some such objectives include minimizing electrical line losses (maximizing grid efficiency), minimizing costs for power generation, minimizing congestion, improving voltage profiles across the distribution feeder, reducing peak power demand, ensuring reliability of service (e.g. higher capacity margins for feeders and supply transformers), and balancing load. Depending on the types of switches in the grid, operators may also minimize the cost incurred by actuating switches. In general, these
objectives can be formulated using a convex function, thus retaining the uniqueness and global optimality of the MILP solution. Further extensions to the reconfiguration problem include distinctions between hard and soft constraints, considering the optimal switch change order to go from topology A to topology B, and considering grid outage conditions and subsequent generator restart and load recovery. Note that soft constraints can include lines which can exceed thermal limits for short periods of time during a reconfiguration activity.

B Dataset: 33-node grid

The 33-node grid presented in [Baran and Wu 1989] is a canonical grid used in the reconfiguration literature. The grid consists of 33 nodes ($N = 33$), 37 lines ($M = 37$) of which 5 are tie lines (NOS) and the remaining 32 are typically assumed to be switches (NCS). The grid is very lossy, with losses up to 8% of total load, and voltage profile violating voltage limits. These characteristics make the 33-node grid an excellent test case for dynamic reconfiguration. Available grid data includes the grid topology and line parameters, as presented in Table 2 and location of loads and their nominal power demand (P and Q) for a single period as presented in Table 3. The network is shown in Fig. 4a.

Table 2: 33-node grid topology data and line parameters

| Branch No. | Upstream Node | Downstream Node | R [ohm]  | X [ohm]  |
|------------|---------------|-----------------|----------|----------|
| 1          | 1             | 2               | 0.0922   | 0.0470   |
| 2          | 2             | 3               | 0.4930   | 0.2511   |
| 3          | 3             | 4               | 0.3660   | 0.1864   |
| 4          | 4             | 5               | 0.3811   | 0.1941   |
| 5          | 5             | 6               | 0.8190   | 0.707    |
| 6          | 6             | 7               | 0.1872   | 0.6188   |
| 7          | 7             | 8               | 0.7114   | 0.2351   |
| 8          | 8             | 9               | 1.030    | 0.7400   |
| 9          | 9             | 10              | 1.0440   | 0.7400   |
| 10         | 10            | 11              | 0.1966   | 0.0650   |
| 11         | 11            | 12              | 0.3744   | 0.1238   |
| 12         | 12            | 13              | 1.4680   | 1.1550   |
| 13         | 13            | 14              | 0.5416   | 0.7129   |
| 14         | 14            | 15              | 0.5910   | 0.5260   |
| 15         | 15            | 16              | 0.7463   | 0.5450   |
| 16         | 16            | 17              | 1.2890   | 1.7210   |
| 17         | 17            | 18              | 0.7320   | 0.5740   |
| 18         | 19            | 2               | 0.1640   | 0.1565   |
| 19         | 19            | 20              | 1.5042   | 1.3554   |
| 20         | 21            | 2               | 0.4095   | 0.4784   |
| 21         | 22            | 2               | 0.7089   | 0.9373   |
| 22         | 23            | 2               | 0.4512   | 0.3083   |
| 23         | 24            | 2               | 0.8980   | 0.7091   |
| 24         | 25            | 2               | 0.8960   | 0.7011   |
| 25         | 26            | 2               | 0.2030   | 0.1034   |
| 26         | 27            | 2               | 0.2842   | 0.1447   |
| 27         | 28            | 2               | 1.0590   | 0.9337   |
| 28         | 29            | 2               | 0.8042   | 0.7006   |
| 29         | 30            | 2               | 0.5075   | 0.2585   |
| 30         | 31            | 2               | 0.9744   | 0.9630   |
| 31         | 32            | 2               | 0.3105   | 0.3619   |
| 32         | 33            | 2               | 0.3410   | 0.5302   |
| 33         | 34            | 2               | 2.00     | 2.00     |
| 34         | 35            | 15              | 2.00     | 2.00     |
| 35         | 12            | 22              | 2.00     | 2.00     |
| 36         | 18            | 33              | 0.500    | 0.500    |
| 37         | 25            | 29              | 0.500    | 0.500    |
Table 3: 33-node grid load data

| j  | $P^L$ [kW] | $Q^L$ [kVAR] | j  | $P^L$ [kW] | $Q^L$ [kVAR] | j  | $P^L$ [kW] | $Q^L$ [kVAR] |
|----|------------|-------------|----|------------|-------------|----|------------|-------------|
| 2  | 100        | 60          | 13 | 60         | 35          | 24 | 420        | 200         |
| 3  | 90         | 40          | 14 | 120        | 80          | 25 | 420        | 200         |
| 4  | 120        | 80          | 15 | 60         | 10          | 26 | 60         | 25          |
| 5  | 60         | 30          | 16 | 60         | 20          | 27 | 60         | 25          |
| 6  | 60         | 20          | 17 | 60         | 20          | 28 | 60         | 20          |
| 7  | 200        | 100         | 18 | 90         | 40          | 29 | 120        | 70          |
| 8  | 200        | 100         | 19 | 90         | 40          | 30 | 200        | 600         |
| 9  | 60         | 20          | 20 | 90         | 40          | 31 | 150        | 70          |
| 10 | 60         | 20          | 21 | 90         | 40          | 32 | 210        | 100         |
| 11 | 45         | 30          | 22 | 90         | 40          | 33 | 60         | 40          |
| 12 | 60         | 35          | 23 | 90         | 50          |    |            |             |

B.1 Community Solar Dataset

We add a range of community solar facilities (each <5MW), up to a penetration of 25.3% of nameplate capacity to baseline load. We divide the grid into sections, based on the location of switches, and vary the location of community solar farms amongst these sections. We denote the distribution of these DERs (DD) as follows: (i) DD-U: uniform distribution of solar throughout the grid; (ii-iv) DD-I, DD-II, DD-III: all facilities are in Sections I, II, or III of the grid respectively; (v) DD-II+III: all facilities are in Sections II and III of the grid. The DD and location of each community solar facility is shown in Fig. 4. Different DDs are used to consider effect on grid reconfiguration, line losses, and voltage profiles.

Figure 4: 33-node distribution grid from Baran and Wu [1989]. The switches are highlighted in green, sections of the grid highlighted and labelled in orange, and location of community solar DERs noted in yellow squares.
Table 4: Locations and capacity of community solar facilities under each DD. The generating capacity is in kW

|        | DD-U |       | DD-I | DD-II | DD-III | DD-II+III |
|--------|------|-------|------|-------|---------|-----------|
|        | \(\hat{P}^{L,j}\) | \(\hat{P}^{L,j}\) | \(\hat{P}^{L,j}\) | \(\hat{P}^{L,j}\) | \(\hat{P}^{L,j}\) | \(\hat{P}^{L,j}\) |
| 4      | 60   | 100   | 12   | 185   | 29      | 180      |
| 7      | 100  | 4     | 14   | 160   | 30      | 100      |
| 12     | 120  | 7     | 15   | 200   | 31      | 180      |
| 14     | 80   | 19    | 16   | 280   | 29      | 160      |
| 19     | 110  | 23    | 16   | 210   | 31      | 190      |
| 23     | 80   | 110   | 15   | 200   | 14      | 100      |
| 26     | 70   | 29    | 18   | 150   | 15      | 0        |
| 29     | 60   | 32    | 19   | 110   | 16      | 80       |
| 32     | 150  | 23    | 17   | 70    | 21      | 110      |

The solar generation data is taken from NREL’s System Advisory Model (SAM) tool. The data is of a 185kW distributed commercial solar PV facility, located in Phoenix, AZ, using the SunPower SPR-E19-310-COM module, and SMA America (STP 60-US-10, 400V) inverter. The DC to AC ratio is set to the default of 1.2. The desired array size is set to 220kWdc, giving a total AC capacity of 179,580kWac. All other parameters are left unchanged in the SAM setup. Figure 5 shows a sample 24-hour generation profile of the solar facility.

Figure 5: Sample solar PV generation profile for community PV facility, queried for May 14, 2019.

C Gradient-based correction

The correction layer looks to minimize the inequality constraint violation:

\[
\min_{\hat{z}, \hat{\varphi}} \| \max \{0, h_x(z, \varphi)\} \|_2^2 \quad \text{subject to: } g_x(z, \varphi) = 0
\]

We can use a gradient descent with momentum to iteratively solve this algorithm, with each iteration being a corrective step:

Initialize: \(z_0 = \hat{z}, \varphi_0 = \hat{\varphi}\)

\[d_k = \nabla_z \| \max \{0, h_x(z, \varphi)\} \|_2^2 \bigg|_{z_k, \varphi_k}
\]

\[\Delta z_{k+1} = \alpha_{corr,k} d_k + \beta_{corr,k} \Delta z_k
\]

\[z_{k+1} = z_k - \Delta z_{k+1}
\]

\[\varphi_{k+1} = G_x(z_{k+1})
\]
where the final step is equivalently $G_x(z_{k+1}) = \frac{\partial \varphi}{\partial x} \Delta z_{k+1}$. This relationship can be calculated using the implicit function theorem and the constraints. Namely:

$$\frac{\partial \varphi}{\partial x} = \left( \frac{\partial g_x}{\partial \varphi} \right)^{-1} \frac{\partial g_x}{\partial x}$$

$$\frac{\partial \varphi}{\partial z} = \left( \frac{\partial g_x}{\partial \varphi} \right)^{-1} \frac{\partial g_x}{\partial z}$$

These Jacobians are also used in the backpropagation step. Various tools and methods exist for efficiently calculating the Jacobians.

D  Ablation study of architectures for constrained learning

Dataset: We perform experiments for the grid reconfiguration problem on a simple 4-node network with two switches. Node 1 is treated as the point of common coupling to the transmission grid, and hence is a slack bus (i.e. the voltage is fixed 1pu; assume that there exists an infinite generator or load to meet the demand of the distribution network). Loads are located at nodes 2 and 4. A generator is located at node 3. The network has two switches, between nodes 1 and 4, and nodes 2 and 4. As with the rest of the paper, we employ the Linear DistFlow equations to model the power physics of the grid [Baran and Wu (1989)], as described in Appendix A. Data is generated using the same methodology as the 33-node grid.

Simulation Parameters: We use PyTorch on a 2.3GHz Dual-Core Intel Core i5, with 8GB RAM. The neural network has two hidden layers, each with 20 neurons. The size of the input and output layer is determined by $x$ and $\hat{z}$. Each layer applies a linear transformation with bias, batch normalization, ReLU activation, and 50% dropout. We use He initialization for the weights. Backpropagation uses an adaptive learning rate (Adam), with parameter $\gamma = 0.001$. There are 4000 data points, split as 80% training, and 10% testing and validation each. We use mini-batching with 200 batch size.

Inequality constraint violation: Violation of inequality constraints is measured as: $error_{ineq-mean}$, the mean violation of all inequality constraints $h_i \in \mathcal{H}$ per epoch.

Integer constraint: We employ a steep sigmoidal activation function at the output layer of the neural network: $\sigma_B(z) = \left[ \frac{2}{\mu + e^{-\tau z}} - 1 \right]_+$, with $\mu = 1$ and $\tau = 5$, from [Cao and Zavala (2020)]. We do not employ the physics-informed rounding.

Gradient-based correction (Corr): The correction layer employs the algorithm in Appendix C with $\alpha_{corr} = 0.001$ and momentum $\beta_{corr} = 0.5$.

Architectures: To better understand the contribution of different components of the framework on constraint violation, we consider the following different architectures (see Fig. 6):

- **Corr**: We use the gradient-based correction in the training loop, to reduce inequality constraint violation on dependent variables, along with a soft loss penalty, with parameter $\lambda_h$. We use 20 gradient steps in training, and 5 gradient steps in testing/validation.
- **Post-proc**: We use only a soft loss penalty, with parameter $\lambda_h$ in the training loop. We employ the gradient-based correction as a post-processing method in the testing and validation step, with 200 gradient steps.
- **Adaptive**: We allow for an adaptive soft loss penalty, parameter $\lambda_h$, in the training loop. The update rule is based on dual-ascent algorithms, and is as below:

  Initialize: $\lambda_{h,0}$

  $$d_k = \| \max \{ 0, h_x(z, \varphi) \} \|_2$$

  $$\lambda_{h,k+1} = \lambda_{h,k} + \alpha_{soft,k} d_k$$

  where $\alpha_{soft,k}$ is a potentially time-varying step-size. Note that $\lambda_{h,k}$ is still a scalar parameter.

We run the following experiments with the architectures described above:

1. Vary the value of $\lambda_h$ to determine impact of soft loss parameter on $error_{ineq-mean}$
2. Compare the performance of Corr and Post-proc architectures against training with only a soft loss penalty, called Penalty architecture.

3. Employ the Adaptive architecture with and without post-processing.

For each test case, the neural network is trained 10 times, and the train/test/validation performance is averaged over the 10 runs. This is done to account for stochasticity in weight initialization and mini-batching. The results are show in Figs. 7 thru 8.

The results in Fig. 7 show little impact of the value of $\lambda_h$ on $\text{error}_{\text{ineq-mean}}$, across values from 5 to 500, and with the different architectures. The Post-proc architecture outperforms both Corr and Penalty, however this may be attributed to the large number of corrective gradient steps (200 steps) taken in the post-processing. Interestingly, the Corr and Penalty architectures perform similarly, despite the additional constraint satisfaction step employed in Corr. Further investigation is needed as to why correction steps in the training loop underperforms, such as with the larger 33-node test case. Table 5 further shows how expensive the correction step can be in the training loop of the Corr architecture (>8min spent training for simple 4-node network and 300 epochs). In comparison, the Post-proc results in lower inequality violation, with significantly reduced training burden, and slightly longer online testing.

Comparing the results in Fig. 7 and 8 shows comparable performance for scalar and Adaptive $\lambda_h$ values, with only a slight reduction in constraint violation with Adaptive. Figure 8b plots the normalized time-evolution of $\lambda_{h,k}$, showing a maximum of 0.8% change in the value. These results suggests that perhaps (a) our problem has limited constraint violation so does not benefit from adaptive weights; or (b) a larger step size or alternate adaptive algorithm is needed.
(a) **Corr:** Neural architecture with gradient-based correction to reduce inequality constraint violation during training.

(b) **Post-proc:** Neural architecture with gradient-based correction to reduce inequality constraint violation as a post-processing step, during the testing and validation.

(c) **Adaptive:** Neural architecture with adaptive soft-loss penalty on inequality constraint violation, $\lambda_h$.

Figure 6: Different training and testing architectures, using gradient-based correction and post-processing, and time-varying adaptive loss penalties.
Figure 7: Plot of $\text{error}_{\text{ineq-mean}}$ in validation, for different soft loss values $\lambda_h$

(a) Corr (solid lines) and Post-proc (dashed lines), both with Adaptive, and initialization of $\lambda_{h,0} = 5$ and 50 in blue and orange respectively

(b) Time evolution of $\lambda_{h,k}$ normalized by $\lambda_{h,0}$ for Post-proc + Adaptive

Figure 8: Plot of $\text{error}_{\text{ineq-mean}}$ in validation and time evolution of $\lambda_{h,k}$ for varying $\lambda_{h,0}$ and architectures