Physics-Informed Graph-Based Learning to Enable Solving Optimal Distribution Switching Problem

Reza Bayani, *Graduate Student Member, IEEE* and Saeed Manshadi, *Senior Member, IEEE*

***Abstract*—This letter introduces a novel graph convolutional neural network (GCN) architecture for solving the optimal switching problem in distribution networks while integrating the underlying power flow equations in the learning process. The switching problem is formulated as a mixed-integer second-order cone program (MISOCP), recognized for its computational intensity making it impossible to solve in many real-world cases. Transforming the existing literature, the proposed learning algorithm is augmented with mathematical model information representing physical system constraints both during and post training stages to ensure the *feasibility* of the rendered decisions. The findings highlight the significant potential of applying predictions from a linearized model to the MISOCP form.**

***Index Terms*—Distribution Switching, Mixed-Integer Programming, Graph Convolutional Networks, Power Flow**

I. Introduction

Distribution system operators execute switching operations to enhance grid's reliability, efficiency, and resilience. These operations include network reconfiguration, fault isolation, loss minimization, load balancing, and voltage regulation, highlighting the importance of optimal switching decisions [1], [2]. The optimal switching problem, characterized by integer (e.g. switching) and non-convex (e.g. power flow) constraints, demands extensive computational effort, imposing a challenge in practical distribution operations where swift and timely decision-making is crucial.

Solving mixed-integer programs (MIPs) with machine learning (ML)-based methods is an active research area. Early research demonstrated promising speed-ups in solving certain MIP classes [3]. Although the learned ML model in [3] outperformed the solver in several instances, it could not beat solver when dealing with electricity grid problems. In addition to the inherent complexity, infeasibility, and sub-optimality are mentioned as challenges of ML prediction tasks in power system problems [4], [5]. To avoid infeasibility issue, researchers often use ML output as *warm start* solutions, rather than directly *fixing* decisions, which limits the predictive power of the ML models. The proposed novel framework here employed a physics-informed GCN network that utilizes MIP features for prediction of optimal integer solutions. The proposed GCN incorporates physical system constraints both during training (through a constraint violation penalty) and after prediction (using a filtering layer) to guarantee feasible solutions.

Reza Bayani is with the University of California San Diego, La Jolla, CA, 92093, USA, and the San Diego State University (email:rbayani@ucsd.edu). Saeed Manshadi is with the San Diego State University, San Diego, CA, 92182, USA. (email: smanshadi@sdsu.edu).

II. Model Background

Consider a power distribution grid with remotely controlled switches on selected lines. The optimal distribution switching problem, presented in (1), includes binary line switching variables and utilizes a second-order cone programming (SOCP)-relaxed power flow. The objective function (1a) minimizes the cost of energy loss, purchase from the upstream grid, battery degradation, and the value of lost load. For lines with a switch, $f_{j,k}$ signifies the operational state of the switch. For lines without a switch, the energization status is inferred from neighboring lines. To maintain the radiality of the network post-switch operation, the single-parent flow model (1b)-(1e) is employed. Two binary variables $y_{j,k}$ and $y_{k,j}$ are introduced to determine the directional flow within each line. Equation (1b) ensures that energy flow remains unidirectional within each line. Here, $\widetilde{\mathcal{B}}$ denotes the set of buses connected to a switch. According to (1c), energy cannot flow towards the substation. The single-parent flow constraint is presented in (1d). By (1e), if a switch is activated, all of its neighboring lines are energized.

$$\min \textstyle\sum_{j,k\in\mathcal{B}} \kappa^{loss} l_{j,k} + \sum_{f\in\mathcal{F}} \kappa^{tou} p_f + \sum_{e\in\mathcal{E}} \kappa^{deg} p_e + \sum_{d\in\mathcal{D}} \kappa^{voll}(p_d^{max} - p_d) \quad (1a)$$

$$y_{j,k} + y_{k,j} = f_{j,k}, \quad \forall j,k \in \widetilde{\mathcal{B}} \quad (1b)$$

$$y_{k,j} = 0, \quad \forall j,k \in \widetilde{\mathcal{B}}_{feeder} \quad (1c)$$

$$\textstyle\sum_{j,k\in\widetilde{\mathcal{B}}_{to}} y_{j,k} + \sum_{j,k\in\widetilde{\mathcal{B}}_{from}} y_{k,j} \le 1, \quad \forall j,k \in \widetilde{\mathcal{B}} \quad (1d)$$

$$f_{j,k} \le f_{j_n,k_n}, \quad \forall j,k \in \widetilde{\mathcal{B}}_{switch}, \forall j_n,k_n \in \widetilde{\mathcal{B}}_{neighbor} \quad (1e)$$

$$p_a^{min} \le p_a \le p_a^{max}, \quad \forall a \in \mathcal{D}\cup\mathcal{E}\cup\mathcal{F}\cup\mathcal{R} \quad (1f)$$

$$q_a^{min} \le q_a \le q_a^{max}, \quad \forall a \in \mathcal{D}\cup\mathcal{F}\cup\mathcal{S} \quad (1g)$$

$$\textstyle\sum_{a\in\mathcal{F}_b\cup\mathcal{R}_b} p_a = \sum_{a\in\mathcal{D}_b\cup\mathcal{E}_b} p_a + \sum_{k\in\mathcal{B}_b} p_{b,k}, \ \forall b \in \mathcal{B} \quad (1h)$$

$$\textstyle\sum_{a\in\mathcal{F}_b\cup\mathcal{S}_b} q_a = \sum_{a\in\mathcal{D}_b} q_a + \sum_{k\in\mathcal{B}_b} q_{b,k}, \ \forall b \in \mathcal{B} \quad (1i)$$

$$c_{j,k}, s_{j,k} \ge 0, \ (c_{j,k} = c_{k,j}, \ s_{j,k} = -s_{k,j}), \quad \forall j,k \in \mathcal{B} \quad (1j)$$

$$|p_{j,k} + c_{j,j}\mathbf{G}_{j,k} - c_{j,k}\mathbf{G}_{j,k} - s_{j,k}\mathbf{B}_{j,k}| \le M(1-f_{j,k}) \quad (1k)$$

$$|q_{j,k} - c_{j,j}\mathbf{B}_{j,k} + c_{j,k}\mathbf{B}_{j,k} - s_{j,k}\mathbf{G}_{j,k}| \le M(1-f_{j,k}) \quad (1l)$$

$$c_{j,k}^2 + s_{j,k}^2 \le c_{j,j}.c_{k,k}, \quad \forall j,k \in \mathcal{B} \quad (1m)$$

$$v_b^{min\,2} \le c_{b,b} \le v_b^{max2}, \quad \forall b \in \mathcal{B} \quad (1n)$$

$$p_{j,k}^2 + q_{j,k}^2 \le p_{j,k}^{max2} \cdot f_{j,k}, \quad \forall j,k \in \mathcal{B} \quad (1o)$$

$$\sigma_e^{'} = \sigma_e + p_e \cdot \eta_e, \quad \sigma_e^{min} \le \sigma_e \le \sigma_e^{max}, \quad \forall e \in \mathcal{E} \quad (1p)$$

$$l_{j,k} \ge |p_{j,k} + p_{k,j}|, \quad \forall j,k \in \mathcal{B} \quad (1q)$$

The real/reactive power (p/q) limits for demands ($\mathcal{D}$), storage devices ($\mathcal{E}$), feeders ($\mathcal{F}$), renewables ($\mathcal{R}$), and capacitors ($\mathcal{S}$) are modeled in (1f) and (1g). The SOCP power flow is modeled in (1h) - (1m). This SOCP relaxation is proven to

be exact in radial networks [6]. Voltage and line flow limits are enforced in (1n) and (1o). According to (1p), the battery's next hour energy ($\sigma^{'}$) depends on its current energy (σ) and power exchange, considering cycle efficiency η. Energy loss is calculated by (1q). The power flow variables and constraints in MISOCP (1) can be linearized similar to the linearization concept utilized in [7], yielding an MILP as described in (2), incorporated in learning to solve the MISCOP problem.

$$\min {q^*}^T x, \quad s.t. \quad A^* x \leq b^*, \quad x_i \in \mathbb{Z} \quad \forall i \in \mathcal{I}. \tag{2}$$

III. Methodology

Despite its convex nature, the extensive integer decisions present in MISOCP (1) complicate its solution. This section illustrates the ML algorithm designed to expedite decision-making without sacrificing feasibility and optimality. The algorithm's training is guided by a loss function that complements the binary cross entropy (BCE) loss with a violation loss component, refining the neural network's weights to converge on feasible solutions. Final binary predictions are obtained by confidence threshold (CT) selection and applying a filtering layer that adheres to the physical system constraints, as illustrated in Fig. 1.

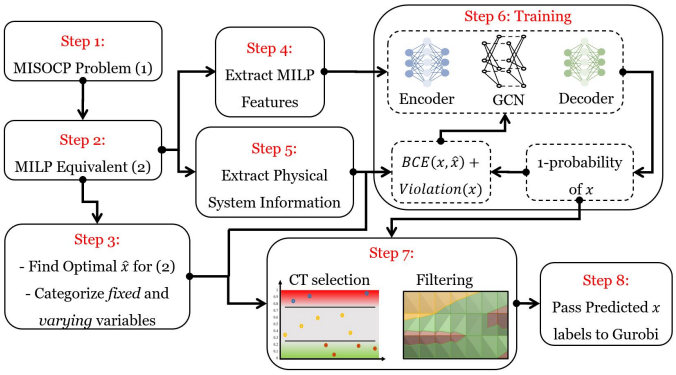

Fig. 1. The illustration of the proposed method

In the first and second steps, the MISOCP (1) is formulated, and its linear equivalent (2) is obtained. The MILP representation is chosen for training due to its computational efficiency. In step 3, the optimal solutions to different instances of (2) are obtained and processed, categorizing the binary variables into *fixed* and *varying* groups. Variables classified as *fixed* are incorporated into the set of predicted x labels. The LP features are extracted in step 4 by considering (2) as a graph comprised of variables and constraint nodes [8]. To extract features for each feasible MIP solution, its LP-relaxed form is considered. Variable features include objective coefficient (f_1^v), lower/upper bounds ($f_{2,3}^v$), type (f_4^v), value (f_5^v), fractionality (f_6^v), reduced cost (f_7^v), basis status (f_8^v), and sensitivity metrics to optimal basis (f_{9-12}^v). Constraint features include sense (f_1^c), right-hand side (f_2^c), slack (f_3^c), dual value (f_4^c), basis status (f_5^c), and sensitivity metrics ($f_{6,7}^c$). The physical system information, i.e. constraints that must be enforced by the mathematical model, are determined and processed in step 5. The constraint violations (denoted by ν) for the set of integer constraints in (2) (denoted by $\mathcal{H}$) are given in (3). This function is called both during GCN training (Step 6) and binary prediction (Step 7).

$$\nu_h = A^* x - b^*, \quad \forall h \in \mathcal{H}. \tag{3}$$

The training algorithm is implemented in step 6. Let $\mathcal{V}_m$ and $\mathcal{C}_n$ respectively denote the encoded normalized features for the m^{th} variable and n^{th} constraint extracted in step 4. In each layer, the graph convolution operation is comprised of a variable-to-constraint and a constraint-to-variable pass, respectively expressed for the l^{th} GCN layer as:

$$\mathcal{C}_m^{l+1} = \gamma^l \left(\mathcal{C}_m^l, \textstyle\sum_{k \in \mathcal{E}_m} \phi^l (\mathcal{C}_m^l, \mathcal{V}_k^l, \xi_{mk}) \right), \text{ and} \tag{4}$$

$$\mathcal{V}_n^{l+1} = \gamma^l \left(\mathcal{V}_n^l, \textstyle\sum_{k \in \tilde{\mathcal{E}}_n} \phi^l (\mathcal{V}_n^l, \mathcal{C}_k^{l+1}, \xi_{nk}) \right). \tag{5}$$

Here, γ and ϕ are differentiable functions. The general idea of a GCN layer is to update node features by incorporating information from adjacent nodes. A Sigmoid function is applied to the decoder output to ensure all predictions are within the (0,1) range. Hence, the final outputs can be treated as *1-probability* of each binary variable. The training loss function is comprised of the BCE and the violation loss. The BCE loss is presented in (6), where α_0 and α_1 denote weights for loss of 0 and 1 components, respectively.

$$BCE(x, \hat{x}) = \textstyle\sum_{i \in \mathcal{I}} \alpha_0 (1 - \hat{x}_i) \log (1 - x_i) + \alpha_1 \hat{x_i} \log x_i \tag{6}$$

We also introduce the idea of violation loss, where the GCN outputs are assessed against the physical system model. Considering (3), the violation loss is defined as:

$$Violation(x) = \textstyle\sum_{h \in \mathcal{H}} \max(0, \nu_h). \tag{7}$$

In step 7, binary predictions are initially processed by the CT selection layer, which utilizes a quantified strategy to discard assignments that do not meet the confidence criteria. The CT selection is governed by the parameter α, which can range between 50% to 100%. For each integer variable x_i, its corresponding binary assignment is determined based on its proximity to either 0 or 1 value, delineated as follows:

$$CT(\alpha) = \begin{cases} \text{predict 1} & x_i \geq \alpha/100, \\ \text{predict 0} & x_i \leq (100 - \alpha)/100, \\ \text{do not predict} & otherwise. \end{cases} \tag{8}$$

A filtering layer then evaluates these predictions against the physical system constraints to guarantee feasibility. Particularly, the filtering rule obtains violations for each constraint in $\mathcal{H}$ based on (3). Binary assignments that result in non-negative violations are filtered out to ensure feasibility of the predictions. Combining CT selection together with the filtering layer ensures that predictions not only adhere to statistical confidence but also align with practical requirements. Ultimately, in step 8, the predicted decisions are passed as fixed-value constraints to the solver (e.g. Gurobi).

Numerical example: Consider the MILP: $\min x_1 - 2x_2$, $c_1 : x_1 + x_2 \geq 1$, $x_1, x_2 \in \{0, 1\}$, with the optimal solution $[0, 1]$. For the feasible solution $[1, 1]$, the first 5 elements in the feature vector of x_2 (i.e. $f_{1-5}^{x_2}$) are $[-2, 0, 1, B, 1]$. For the constraint c_1, the first 3 elements (i.e. $f_{1-3}^{c_1}$) are $[\geq, 1, -1]$. After applying one-hot encoding to non-numerical features, the resulting vectors are normalized. If the GCN output (i.e., 1-probability) for these features is $[0.15, 0.75]$, a constraint violation loss of 0.1 is considered. Additionally, the filtering layer will only predict x_1 to avoid infeasible predictions.

IV. Results and Discussion

In this part, the performance of the proposed GCN solver is investigated. Comparisons are presented for solving randomized instances of the optimal switching problem in both its MILP and MISOCP forms. The GCN solver is implemented as part of the Gurobi solver. The test scenario involves a modified IEEE 33 bus distribution network, featuring 37 lines (10 with switching capability) and 2 solar units, each paired with a battery unit and a shunt capacitor. The resulting MIP contains 4,438 continuous variables and 924 binary variables. As a result of solving 200 MILP instances for 15 minutes each, 1994 samples were extracted as training dataset. The MILPs were generated randomly by adjusting demand and renewable generation curves by multipliers drawn from $\mathcal{N}(1, 0.03)$ for temporal values and $\mathcal{N}(1, 0.1)$ for spatial values. Training 100 episodes took 1442 seconds. To perform tests on unseen MIP instances, the average processor time for extracting features and returning GCN predictions was 4.47 and 1.69 seconds, respectively.

A. Learning to Solve the Linearized Switching Problem

In our analysis, we compare the performance of the GCN solver with the Gurobi solver across 10 randomly selected MILP instances. It is clear from Figure 2 that the proposed framework will quickly lead to small duality gaps. Figure 3 reveals that the GCN solver significantly reduces solution times compared to Gurobi and achieves a very small gap. The superiority of the GCN solver is highlighted by its ability to identify high-quality solutions in a fraction of the time needed by Gurobi.

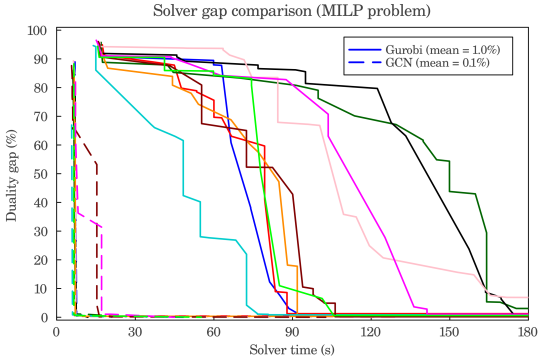

Fig. 2. Duality gap for Gurobi vs. GCN solver (MILP case)

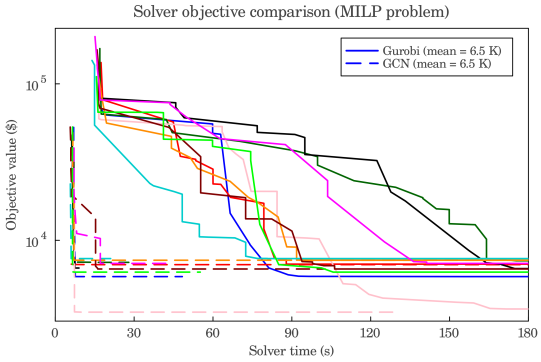

Fig. 3. Objective value for Gurobi vs. GCN solver (MILP case)

B. Applying the GCN Solver for Solving the MISOCP

In this part, the GCN trained on MILP instances is utilized to solve MISOCP problems. Remarkably, the GCN solver's superiority over Gurobi is even more pronounced in MISOCP instances than in MILP cases. As demonstrated in Fig. 4, the GCN solver quickly attains high-quality solutions within minutes, in stark contrast to Gurobi's struggle even after 30 minutes across 10 instances. Notably, the GCN solver achieves an average objective value of \$4,555, whereas Gurobi's average stands at \$41,046 at the same solve time, highlighting the GCN solver's substantial advantage. More importantly, all of the rendered solutions to the GCN-empowered solver are feasible given the physics-informed architecture implemented.

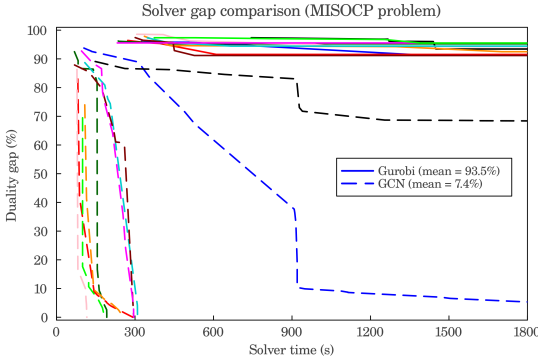

Fig. 4. Duality gap for Gurobi vs. GCN solver (MISOCP case)

The results underscore the GCN's crucial capacity for rapidly identifying high-quality solutions. Table I presents the objective values at various stages of the solving process for both solvers, with values in parentheses indicating the number of problem instances where a feasible solution was achieved at that stage. After 120 seconds, Gurobi has yet to find any feasible solutions, whereas the GCN solver has identified 9, achieving an average objective of 17,304. Table II summarizes the gap progress across 10 test instances highlighting the superiority of GCN solver in attaining very low gaps compared to Gurobi. Notably, in the MISOCP scenario, Gurobi fails to achieve an 80% gap within 1800 seconds, while GCN, on average, reaches this gap in just 271 seconds.

TABLE I
Objective value at certain time (MISOCP case)

Time (s)	120	180	300	600	900	1200	1800
Gurobi ($)	∞ (0)	∞ (0)	66,942 (3)	57,116 (9)	55,537 (10)	54,108 (10)	41,047 (10)
GCN ($)	17,304 (9)	12,002 (10)	8,795 (10)	6,061 (10)	5,872 (10)	4,643 (10)	4,556 (10)

TABLE II
Time to reach specific duality gap in seconds (count of instances reached to gap out of 10 test instances)

Problem	Solver	95%	80%	65%	50%	30%	5%
MILP	Gurobi	17 (10)	80 (10)	98 (10)	107 (10)	111 (10)	141 (10)
MILP	GCN	7 (10)	7 (10)	14 (10)	15 (10)	16 (10)	16 (10)
MISOCP	Gurobi	1,509 (7)	∞ (0)	∞ (0)	∞ (0)	∞ (0)	∞ (0)
MISOCP	GCN	128 (10)	271 (10)	264 (9)	273 (9)	277 (9)	277 (8)
MISOCP	Gurobi*	174 (10)	197 (9)	187 (6)	187 (6)	195 (6)	∞ (0)
MISOCP	GCN*	39 (10)	67 (10)	77 (9)	91 (9)	91 (9)	178 (6)

In our baseline experiments, Gurobi's pre-solve and heuristic features were disabled. To demonstrate the GCN's ability to learn patterns and guide the search process in ways traditional solvers might not, the bottom rows in Table II (distinguished with *) are designated to additional comparisons when Gurobi* is used in its full capacity. It is observed that here, GCN* still outperforms Gurobi* by a considerable margin. For example, it reaches 80% and 30% gap solutions approximately 2.9 and 2.1 times faster,

respectively. At the same time, GCN* is 11% and 50% more successful than Gurobi* in reaching these gaps.

C. Scalability Tests on a Larger Problem

To demonstrate the scalability of the proposed framework, GCN was trained for the MILP problem for a modified IEEE 123-bus feeder. The resulting problem consisted of 31,152 variables and 53,952 constraints. The GCN training time for 100 episodes was 1673 seconds, with an average GCN prediction time of 3.96 seconds. The objective value of each solver at different stages of the solution process is displayed in Table III. Notably, although Gurobi ultimately reaches a solution with a 0.24% smaller gap, GCN finds low-cost solutions much faster. For example, GCN reaches a 30% optimality gap in an average of 97 seconds, whereas Gurobi achieves the same gap in 676 seconds, demonstrating a 7.0-fold speed-up.

TABLE III
OBJECTIVE VALUE AT CERTAIN TIME (LARGE SYSTEM CASE)

Time (s)	120	180	300	450	750	900
Gurobi ($)	∞ (0)	∞ (0)	124,766 (6)	37,244 (10)	29,341 (10)	24,001 (10)
GCN ($)	28,333 (10)	25,958 (10)	24,059 (10)	24,059 (10)	24,059 (10)	24,059 (10)

V. CONCLUSIONS

This letter explores using GCNs, reinforced with physical system information. It is shown that augmenting system constraints into the learning framework directs GCN predictions towards feasibility. The model trained based on the MILP problem is utilized to make meritorious predictions for the MISOCP problem. The comparative analysis shows significant enhancements in both solution speed and quality with the GCN solver against the Gurobi solver for the MILP and MISOCP formulations of the problem.

REFERENCES

[1] Z. Li, S. Jazebi, and F. De Leon, "Determination of the optimal switching frequency for distribution system reconfiguration," *IEEE Transactions on Power Delivery*, vol. 32, no. 4, pp. 2060–2069, 2016.

[2] T. Zhang, C. Wang, F. Luo, P. Li, and L. Yao, "Optimal design of the sectional switch and tie line for the distribution network based on the fault incidence matrix," *IEEE Transactions on Power Systems*, vol. 34, no. 6, pp. 4869–4879, 2019.

[3] V. Nair, S. Bartunov, F. Gimeno, I. Von Glehn, P. Lichocki, I. Lobov, B. O'Donoghue, N. Sonnerat, C. Tjandraatmadja, P. Wang *et al.*, "Solving mixed integer programs using neural networks," *arXiv preprint arXiv:2012.13349*, 2020.

[4] Á. S. Xavier, F. Qiu, and S. Ahmed, "Learning to solve large-scale security-constrained unit commitment problems," *INFORMS Journal on Computing*, vol. 33, no. 2, pp. 739–756, 2021.

[5] T. Wu, Y.-J. A. Zhang, and S. Wang, "Deep learning to optimize: Security-constrained unit commitment with uncertain wind power generation and besss," *IEEE Transactions on Sustainable Energy*, vol. 13, no. 1, pp. 231–240, 2021.

[6] A. F. Soofi, S. D. Manshadi, G. Liu, and R. Dai, "A socp relaxation for cycle constraints in the optimal power flow problem," *IEEE Transactions on Smart Grid*, vol. 12, no. 2, pp. 1663–1673, 2020.

[7] R. Bayani, M. Bushlaibi, and S. D. Manshadi, "Short-term operational planning problem of the multiple-energy carrier hybrid ac/dc microgrids," in *2021 IEEE Power & Energy Society General Meeting (PESGM)*. IEEE, 2021, pp. 1–5.

[8] Q. Cappart, D. Chételat, E. B. Khalil, A. Lodi, C. Morris, and P. Veličković, "Combinatorial optimization and reasoning with graph neural networks," *Journal of Machine Learning Research*, vol. 24, no. 130, pp. 1–61, 2023.