***small ps :)**

this project was created as a part of the term paper assignment of EET-110. given my term paper experience of the previous semester, i was determined to conquer this one alone. if anyone who is going to take this course in the future is reading this, DONOT TRY TO DO YOUR TERM PAPER ALONE. your fire-like motivation will diminish by the third week and thoughts of dropping the course will become common occurence. unfortunately, my nuggets of wisdom for my juniors end here.

the complete assignment was to beat the base paper at a metric of our choice. the base paper i chose was this one and the metric i used was SOI, as defined in the paper. reading the paper, i realized that it was almost impossible to beat it in wall-clock time and, therefore, tried to make a (N-1) topology-generalizable architecture. if the last sentence did not make any sense to you, hopefully by the end of this blog it will.

codebase for the entire project can be found here. you can contact the author here

1 What is Unit Commitment?

Every morning, a grid operator faces a scheduling problem. There are power plants available — some large and cheap to run (coal, gas turbines), some small and expensive (diesel peakers). There is a load forecast telling the operator how much electricity customers will demand, hour by hour, over the next 24 hours. The operator must decide: which plants to turn on, and when? This is the **Unit Commitment** (UC) problem.

And no, running everything all the time is not a solution. Running a generator has a fixed cost even when it produces zero power, the no-load cost covers fuel and maintenance. Across thousands of units on a real grid, poor commitment decisions could potentially cost millions of dollars daily.

The output of UC is a binary matrix $z \in \{0,1\}^{G\times T}$ where G is the number of generators and T is the scheduling horizon (24 hours in our case). $z_{g,t} = 1$ means generator g is ON at hour t; $z_{g,t} = 0$ means it is OFF.

Once the commitment is decided, the **economic dispatch** problem determines how much power each committed generator produces to minimise total fuel cost. In our formulation, these two stages are solved simultaneously as a single MILP.

2 Why is Unit Commitment Hard?

The decision space has size $2^{G\times T}$. As we worked with a 30-bus system, our 6-generator, 24-hour system:

$$2^{6\times 24} = 2^{144} \approx 2.2 \times 10^{43}$$

Real power systems have hundreds of generators, making the problem vastly harder. The difficulty is compounded by physical constraints that couple decisions across time and generators:

- **Minimum uptime:** Once a steam turbine starts, it must run for at least T_g^U hours (typically 3–6 hours) because shutting it down and restarting costs significant fuel and causes thermal stress.
- **Minimum downtime:** After shutdown, the unit must stay off for at least T_g^D hours to cool down safely.
- **Ramp rate limits:** A generator cannot jump from 0 to full power instantly. It increases at most R_g^U MW per hour.
- **Spinning reserve:** At any hour, there must be enough committed capacity beyond current demand to handle sudden load increases.

These constraints mean that the commitment decision at hour 6 depends on what happened at hour 3 and what will happen at hour 10.

3 The Security Constraint: N-1 Criterion

Power grids do not just need to serve normal operating conditions. They must also remain operational after any single component failure. This is the **N-1 criterion**: the system must be secure against the loss of any one element (a line, a transformer, a generator).

For our 30-bus, 41-line system, this means checking 41 N-1 contingencies. For each possible line outage, the committed schedule must allow a feasible power dispatch without violating any remaining line's thermal limit.

The **Security-Constrained UC** (SCUC) embeds all N-1 checks directly into the optimisation problem. This makes it 41× harder: instead of one set of transmission constraints, we have 42 (base case plus 41 outages).

4 The MILP Formulation

The SCUC problem is formulated as a Mixed-Integer Linear Programme (MILP). Presented below is the complete MILP formulation.

4.1 Decision Variables

Variable	Type	Meaning
$z_{g,t}$	Binary	Generator g is ON at hour t
$v_{g,t}$	Binary	Generator g starts up at hour t
$y_{g,t}$	Binary	Generator g shuts down at hour t
$p_{g,t}$	Continuous	Power output of generator g at hour t (MW)
$p_{g,m,t}$	Continuous	Output in segment m (piecewise cost)
$r_{g,t}$	Continuous	Spinning reserve from generator g at hour t
$f_{\ell,t}^{(c)}$	Continuous	Power flow on line ℓ under contingency c
λ	Continuous	Fuzzy satisfaction (Stage 3 only)

4.2 Objective Function

Minimise total operating cost over 24 hours:

$$\min_{z,p} \sum_{g=1}^{G} \sum_{t=1}^{T} \left[\underbrace{c_g^z z_{g,t}}_{\text{no-load}} + \underbrace{c_g^{su} v_{g,t}}_{\text{startup}} + \underbrace{c_g^{sd} y_{g,t}}_{\text{shutdown}} + \underbrace{c^r r_{g,t}}_{\text{reserve}} + \underbrace{\sum_{m=1}^{M} c_{g,m}^p \, p_{g,m,t}}_{\text{fuel (piecewise)}} \right] \tag{1}$$

The piecewise fuel cost uses $M = 2$ linear segments to approximate the quadratic fuel curve, keeping the problem purely linear.

4.3 State Transition Constraints

The binary variables $v_{g,t}$ (startup) and $y_{g,t}$ (shutdown) are linked to the commitment $z_{g,t}$:

$$v_{g,t} + y_{g,t} \leq 1 \qquad \forall g, t \tag{2}$$

$$z_{g,t} - z_{g,t-1} = v_{g,t} - y_{g,t} \qquad \forall g, t > 0 \tag{3}$$

Constraint (2) says a unit cannot start and stop in the same hour. Constraint (3) says if z goes from 0 to 1, a startup occurred; if it goes from 1 to 0, a shutdown occurred.

4.4 Minimum Uptime and Downtime

$$z_{g,s} \geq v_{g,t+1} \quad \forall\, s \in [t+2,\, \min(t + T_g^U, T)],\ \forall g, t \tag{4}$$

$$1 - z_{g,s} \geq y_{g,t+1} \quad \forall\, s \in [t+1,\, \min(t + T_g^D, T)],\ \forall g, t \tag{5}$$

If unit g starts at hour $t+1$, it must stay ON for the next T_g^U hours. If it stops at hour $t+1$, it must stay OFF for T_g^D hours.

4.5 Generation Limits and Ramp Rates

$$p_{g,t} = \sum_{m=1}^{M} p_{g,m,t} \tag{6}$$

$$p_{g,m,t} \leq P_{g,m}^U \, z_{g,t} \tag{7}$$

$$P_g^{\min} \, z_{g,t} \leq p_{g,t} \leq P_g^{\max} \, z_{g,t} \tag{8}$$

$$p_{g,t} - p_{g,t-1} \leq R_g^U (1 - v_{g,t}) + P_g^D v_{g,t} \tag{9}$$

$$p_{g,t-1} - p_{g,t} \leq P_g^D y_{g,t} + R_g^D (1 - y_{g,t}) \tag{10}$$

Constraint (8) forces $p_{g,t} = 0$ when $z_{g,t} = 0$ (no generation from offline units) without needing a big-M formulation, since the segment bounds already enforce it.

4.6 Power Balance

At every hour, total generation must equal total load:

$$\sum_{g=1}^{G} p_{g,t} = \sum_{n=1}^{N} P_n^d(t) \qquad \forall t \tag{11}$$

4.7 DC Power Flow and Thermal Limits

Line flows are computed using the DC approximation via Power Transfer Distribution Factors (PTDF):

$$f_{\ell,t}^{(c)} = \sum_{n=1}^{N} \mathrm{PTDF}_{\ell,n}^{(c)} \cdot P_n^{\mathrm{net}}(t) \tag{12}$$

where $\text{PTDF}_{\ell,n}^{(c)}$ is the sensitivity of line ℓ's flow to a unit injection at bus n under contingency c, computed by inverting the reduced susceptance matrix. Thermal limits:

$$\left|f_{\ell,t}^{(c)}\right| \leq F_\ell^U \qquad \forall\, \ell \in \Omega_L \setminus \{\ell_c\},\ \forall t \tag{13}$$

The tripped line ℓ_c is excluded — its flow is forced to zero.

4.8 Spinning Reserve

$$\sum_{g=1}^{G} r_{g,t} \geq \Gamma \cdot \max_g (P_g^{\max} z_{g,t}) \qquad \forall t, \quad \Gamma = 0.25 \tag{14}$$

At every hour, committed headroom must cover 25% of the largest online unit's capacity (NPCC standard).

5 The IEEE 30-Bus Test System

All experiments in this project use the IEEE 30-bus system, a standard benchmark in power systems research.

Table 1: IEEE 30-Bus System: Generator Parameters

Gen	Bus	$P^{\min}$ (MW)	$P^{\max}$ (MW)	T^U (h)	T^D (h)	R^U (MW/h)	R^D (MW/h)	c^{su} ($/start)
0	1	10	80	3	3	40	40	200
1	2	10	80	3	3	40	40	180
2	13	5	40	1	1	40	40	90
3	22	5	50	2	2	30	30	110
4	23	3	30	1	1	30	30	70
5	27	8	150	3	3	28	28	130
Total capacity			430 MW	Peak load ≈ 283.5 MW (51% reserve margin)				

The base paper makes a few arguments about machine learning being used as a proxy for optimization problem and I'll be extending their arguments.

6 The Computational Bottleneck

Gurobi solves one instance of the 30-bus SCUC in about 2–3 seconds with a 0.1% optimality gap. This sounds fast. But consider:

- The Polish 2383-bus system used by researchers takes *hours* per solve.
- Real operators re-solve UC every 15 minutes as load forecasts update.
- Each re-solve at the 2383-bus scale would miss its own deadline.

The bottleneck is the binary commitment variables. Relaxing them to continuous (the LP relaxation) is fast, but the binary structure is what makes UC physically meaningful, you cannot commit a generator to 0.7.

7 The ML-UC Idea

The ML-UC pipeline works in three stages:

1. **Offline (training):** Solve SCUC exactly using Gurobi for many historical scenarios. Each solve gives a labelled pair (load profile, optimal commitment). Train a neural network on these.
2. **Online (inference):** Given a new load profile, run the neural network to predict $\tilde{z}$ in milliseconds.
3. **Correction (feasibility):** Use a fuzzy MILP layer to correct $\tilde{z}$ into a guaranteed-feasible schedule.

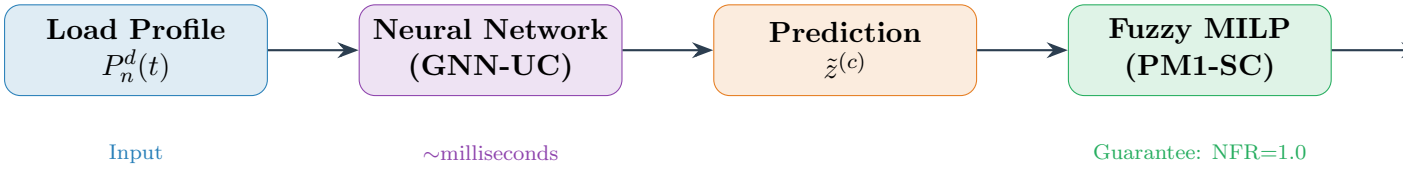

Figure 1: The ML-UC pipeline: load profile enters, feasible schedule exits. The neural network runs in milliseconds; the fuzzy MILP provides a mathematical feasibility guarantee.

8 What Venkatesh et al. Did (The Base Paper)

Venkatesh, Shekeew, and Ma (2025) [**?**] were the first to combine CNN-BiLSTM predictors with fuzzy MILP feasibility layers for UC. Their key contribution was showing that ML predictions could be embedded as *non-binding fuzzy constraints* in a MILP, guaranteeing 100% feasibility even when the ML prediction is wrong.

Their method:

- CNN spatial encoder: treats the grid as a flat feature vector, one entry per bus.
- BiLSTM temporal encoder: processes the 24-hour sequence.
- PM1/PM2 fuzzy MILP: corrects the prediction while guaranteeing feasibility.
- Achieves SOI = 0.060 on unseen load profiles (Case 2).

What they did not do:

- They never tested N-1 line outages at inference time.
- They never reported an N-1 Feasibility Rate metric.

- Their CNN is structurally unable to see which line tripped — it receives the same input regardless of topology.

9 The Gap This Project Fills

Our hypothesis: a Graph Neural Network (GNN) trained on contingency-augmented data, combined with a PM1-SC fuzzy layer with embedded N-1 constraints, will produce commitment schedules that are both topology-aware and guaranteed feasible under N-1 outages.

10 Power Grid as a Graph

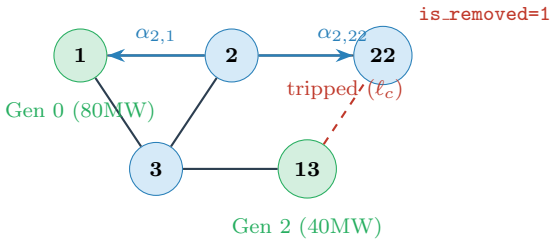

Figure 2: A subgraph of the IEEE 30-bus system illustrating graph concepts. Green nodes host generators. The dashed red edge shows a tripped line (`is_removed=1`). Blue arrows show attention weights α_{ij} that the GAT will learn.

11 Why a CNN Cannot See Topology

A Convolutional Neural Network processes its input as a fixed-size array. For a power grid, this means stacking bus features into a vector: $[P_1^d(t), P_2^d(t), \ldots, P_{30}^d(t), \ldots]$.

When line ℓ_c trips, the power flow equations change completely — currents must reroute through alternative paths. But the CNN's input array looks identical to the base case. Bus 1's load did not change. Bus 22's load did not change. The array is the same. **The CNN is blind to the topology change.**

We verified this empirically: our trained CNN-BiLSTM baseline produces *identical* commitment predictions for all 41 N-1 contingencies of the same scenario. It learned one schedule and applies it everywhere.

12 The Graph Representation

We represent the power system under contingency c as:

$$\mathcal{G}^{(c)} = \left(\mathcal{V},\ \mathcal{E}^{(c)},\ \mathbf{X}(t),\ \mathbf{W}\right) \tag{15}$$

12.1 Node Features

Each bus n at time t has a feature vector:

$$\mathbf{x}_n(t) = \Big[\underbrace{P_n^d(t)}_{\text{net load}}, \quad \underbrace{\bar{c}_n^p}_{\text{avg fuel cost}}, \quad \underbrace{\bar{P}_n^U}_{\text{installed cap.}}, \quad \underbrace{\mathbf{1}_{n\in\Lambda^G}}_{\text{has generator}}\Big]^\top \in \mathbb{R}^4 \tag{16}$$

Let us make this concrete for bus 27 (Gen 5, 150 MW) at a peak hour:

Index	Feature	Formula	Example value
0	Net load $P_{27}^d(t)$	$d_{27}(t) - p_{27}^w(t)$	0 MW (no load at bus 27)
1	Avg fuel cost $\bar{c}_{27}^p$	$\sum_g M_{27,g} c_g^p / \sum_g M_{27,g}$	\$2.60/MWh
2	Installed capacity $\bar{P}_{27}^U$	$\sum_g M_{27,g} P_g^{\max}$	150 MW
3	Has-generator flag	$\mathbf{1}_{27\in\Lambda^G}$	1

12.2 Edge Features

Each transmission line $\ell = (i, j)$ has a feature vector:

$$\mathbf{w}_{ij} = \Big[\underbrace{b_{ij}}_{\text{susceptance}}, \quad \underbrace{F_{ij}^U / S_{\text{base}}}_{\text{norm. rating}}, \quad \underbrace{\mathbf{1}_{\text{removed}}}_{\text{KEY SIGNAL}}\Big]^\top \in \mathbb{R}^3 \tag{17}$$

The **is_removed** flag is the critical design choice. We do not delete the tripped line from the graph. We keep it and mark it. This gives the GNN an explicit signal about what changed topologically.

If you delete the tripped edge, the GNN must infer what changed from the *absence* of a connection — which is hard to learn. If you keep the edge and mark it with `is_removed=1`, the GNN receives a direct signal: "this line is out." The attention mechanism can then learn to down-weight or ignore this edge's contribution to neighbourhood aggregation.

12.3 The Generator–Bus Incidence Matrix

Definition .1. *The generator–bus incidence matrix* $\mathbf{M} \in \{0, 1\}^{N\times G}$*:*

$$M_{n,g} = \begin{cases} 1 & \textit{generator } g \textit{ is at bus } n \\ 0 & \textit{otherwise} \end{cases} \tag{18}$$

For our 30-bus system:

Bus n	G0	G1	G2	G3	G4	G5
1	**1**	0	0	0	0	0
2	0	**1**	0	0	0	0
13	0	0	**1**	0	0	0
22	0	0	0	**1**	0	0
23	0	0	0	0	**1**	0
27	0	0	0	0	0	**1**
⋮	(all other rows are zero)					

This matrix is used in two places: (i) to compute the bus-level fuel cost and capacity features in Equation (16), and (ii) to project bus-level GNN embeddings back to generator-level decisions in the BiLSTM decoder.

13 Stage 1: Graph Attention Network

13.1 Neighbourhood Aggregation

The fundamental operation of a GNN is **neighbourhood aggregation**: to update bus n's representation, collect information from all its electrically connected neighbours and combine it.

13.2 Attention: Not All Neighbours Are Equal

A simple GNN would average all neighbours equally. But in a power grid, the electrical coupling between buses depends on line impedance — a high-susceptance (low-impedance) line creates strong coupling; a transformer creates weak coupling.

Graph Attention Networks (GAT) learn a weight for each neighbour, called an **attention coefficient**. The unnormalised attention from bus j to bus i at layer k, incorporating edge features $\mathbf{w}_{ij}$:

[=] GAT Attention Equation

$$e_{ij}^{(c)} = \text{LeakyReLU}\left(\mathbf{a}^\top \underbrace{\left[\mathbf{W}_n \mathbf{h}_i^{(k)} \,\|\, \mathbf{W}_n \mathbf{h}_j^{(k)} \,\|\, \mathbf{W}_e \mathbf{w}_{ij}\right]}_{\text{concatenate: bus } i\text{, bus } j\text{, line features}}\right) \tag{19}$$

$\mathbf{a} \in \mathbb{R}^{3d_h}$	learnable attention vector
$\mathbf{W}_n \in \mathbb{R}^{d_h \times 4}$	node feature transform
$\mathbf{W}_e \in \mathbb{R}^{d_h \times 3}$	edge feature transform
$\mathbf{h}_i^{(k)}$	current embedding of bus i
$\|$	concatenation

The normalised attention coefficient (softmax over neighbours):

$$\alpha_{ij}^{(c)} = \frac{\exp\left(e_{ij}^{(c)}\right)}{\sum_{k \in \mathcal{N}^{(c)}(i)} \exp\left(e_{ik}^{(c)}\right)} \tag{20}$$

where $\mathcal{N}^{(c)}(i)$ is the set of buses connected to bus i under contingency c.

13.3 Why GATv2 Specifically?

We use **GATv2** (Brody et al., 2022) rather than the original GAT. The difference is subtle but important: original GAT computes attention as $\mathbf{a}^\top[\mathbf{W}\mathbf{h}_i \| \mathbf{W}\mathbf{h}_j]$, where the two terms are computed *separately* before concatenation. GATv2 computes the attention after concatenation, creating a *joint* function of both endpoints. This is called **dynamic attention** — the importance of bus j to bus i depends on both of their states, not just one.

For power systems, this matters: the importance of bus 1's information to bus 22 depends on what *both* buses are doing. A cheap, lightly loaded Gen 0 at bus 1 is very important to an expensive, heavily loaded Gen 3 at bus 22. Dynamic attention can represent this relationship; static attention cannot.

14 Stage 2: BiLSTM Temporal Decoder

14.1 Why Bidirectional?

After the spatial encoder, each bus has a sequence of embeddings $\{\mathbf{Q}_n(t)\}_{t=1}^{24} \in \mathbb{R}^{d_h}$.

A standard LSTM processes this left to right: the hidden state at hour t depends on hours $1, \ldots, t-1$. But unit commitment decisions are *bidirectionally* coupled:

- **Forward dependency:** You cannot decide hour 6 without knowing that the turbine started at hour 4 and must stay on for 3 hours.
- **Backward dependency:** You cannot decide hour 6 without knowing whether the turbine is needed at hour 8 (evening peak starts).

A BiLSTM processes the sequence in both directions:

$$\vec{h}_{n,t} = \tanh\left(\mathbf{W}_c\,\mathbf{Q}_n(t) + \mathbf{W}_{\rightarrow}\,\vec{h}_{n,t-1} + \mathbf{b}_{\rightarrow}\right) \tag{21}$$

$$\overleftarrow{h}_{n,t} = \tanh\left(\mathbf{W}_c\,\mathbf{Q}_n(t) + \mathbf{W}_{\leftarrow}\,\overleftarrow{h}_{n,t+1} + \mathbf{b}_{\leftarrow}\right) \tag{22}$$

$$\tilde{Y}_n(t) = \tanh\left(\mathbf{V}_{\rightarrow}\,\vec{h}_{n,t} + \mathbf{V}_{\leftarrow}\,\overleftarrow{h}_{n,t} + \mathbf{b}_Y\right) \tag{23}$$

The BiLSTM treats the 30 buses as a "batch" dimension — each bus gets its own independent temporal processing, with cross-bus dependencies already encoded by the GAT stage.

14.2 Bus-to-Generator Projection

The BiLSTM produces a bus-level embedding $\tilde{Y}_n(t)$. We need generator-level predictions. The incidence matrix $\mathbf{M}$ performs this projection:

$$\tilde{Y}_g(t) = \sum_{n=1}^{N} M_{n,g}\, \tilde{Y}_n(t) = \tilde{Y}_{\text{bus}(g)}(t) \tag{24}$$

Since $\mathbf{M}$ has exactly one non-zero per column, this simply selects the embedding of generator g's host bus. **We do not learn this mapping** — the physical location of generators is known and must not be learned away.

14.3 Per-Generator Output Heads

Each generator has its own linear output head:

$$\text{logit}_g(t) = \mathbf{w}_g^\top \tilde{Y}_g(t) + b_g, \qquad \mathbf{w}_g \in \mathbb{R}^{d_h} \tag{25}$$

The binary commitment prediction:

$$\tilde{z}_g^{(c)}(t) = \begin{cases} 1 & \text{if } \text{logit}_g(t) \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{26}$$

15 Training the Model

15.1 Dataset

For each of 210 scenarios, we solve SCUC under the base topology and all 41 N-1 contingencies using Gurobi, giving $210 \times 42 = 8{,}820$ labelled instances. The dataset is split at the *scenario level*:

Split	Scenarios	Samples	Purpose
Train	147	5,498	Model fitting
Validation	31	1,158	Early stopping
Test	32	1,203	Final evaluation
Total	210	8,820	

Why scenario-level splitting? If you split at the sample level, your model sees scenario 47 under the base case in training and scenario 47 under contingency 3 in test. The model memorises load profiles rather than generalising. Scenario-level splitting ensures test scenarios are genuinely unseen.

Each scenario has randomised load scaling ($\mu = 1.1$, $\sigma = 0.15$), per-bus spatial noise (3%), fuel cost perturbation (10%), and 30 MW solar PV generation at bus 5. This diversity prevents the model from memorising specific patterns.

15.2 Loss Function

BCEWithLogitsLoss with positive class weights:

$$\mathcal{L} = -\frac{1}{TGS(1+C)} \sum_{c,s,t,g} w_g^+ \left[z_{t,g}^{(c)} \log \sigma(\text{logit}_{t,g}) + (1 - z_{t,g}^{(c)}) \log(1 - \sigma(\text{logit}_{t,g})) \right] \quad (27)$$

with $\mathbf{w}^+ = [1, 1, 5, 5, 5, 5]$. Generators 0 and 1 are almost always ON (weight 1 is sufficient). Generators 2–5 are ON only a fraction of hours — class imbalance means the loss would ignore their minority-class errors without boosted weights.

15.3 Early Stopping

Early stopping tracks $\min_{g \in \{2,3,4,5\}} \text{acc}_g$ (minimum accuracy across hard generators). This is better than tracking mean accuracy: a model that gets Gen 0 and Gen 1 perfectly but fails on Gen 3 would score 83% mean accuracy while failing operationally.

Hyperparameter	Value
GAT layers	3
Embedding dimension d_h	64
Attention heads	4 (intermediate), 1 (last)
LSTM layers	2 (bidirectional)
LSTM hidden size	64
Total parameters	208,102
Optimiser	AdamW, lr=3×10^{-4}, wd=10^{-4}
LR schedule	ReduceLROnPlateau (factor=0.5, patience=10)
Dropout	0.10
Early stopping patience	40 epochs

16 The Problem the Fuzzy Layer Solves

Our GNN predicts the commitment schedule with 89.7% accuracy. That means 10.3% of bits are wrong. In a 6-generator, 24-hour schedule, that is about 15 wrong bits per sample (mean Hamming distance = 14.86).

A single critical error can make the schedule infeasible. If Gen 5 is predicted OFF at an hour when it is actually needed (load is high, other generators are at capacity), there is no feasible dispatch — load cannot be met. The LP feasibility check fails.

We need a method that:

1. Tries to match the GNN prediction as closely as possible.
2. Guarantees feasibility regardless of prediction quality.
3. Satisfies all N-1 security constraints.

The PM1-SC fuzzy MILP achieves all three.

17 The Fuzzy Satisfaction Parameter λ

We introduce a scalar $\lambda \in [0, 1]$ that measures how closely the fuzzy solution z^F agrees with the GNN prediction $\tilde{z}$:

- $\lambda = 1$: Perfect agreement. Every predicted-ON generator is ON in z^F, and every predicted-OFF generator is OFF.
- $\lambda = 0$: Complete override. The solver ignored the prediction and found any feasible solution.

18 The PM1-SC Formulation

[=] PM1-SC: Individual Fuzzy Security-Constrained Model

$$\max_{z^F, \lambda} \quad \lambda$$

subject to:

$$\lambda \leq z^F_{g,t}, \quad \forall (g,t) \in \Upsilon^u \qquad \text{(PM1-SC-a)}$$

$$\lambda + z^F_{g,t} \leq 1, \quad \forall (g,t) \in \Upsilon^d \qquad \text{(PM1-SC-b)}$$

$$\frac{\bar{C} - \underline{C}}{K} \lambda + C(z^F) \leq \bar{C} \qquad \text{(PM1-SC-c)}$$

$$G^{SC}(x^F) \leq H^{SC} \qquad \text{(PM1-SC-d}\star\text{)}$$

$$0 \leq \lambda \leq 1, \quad x^F = [z^F, p^F, \lambda] \qquad \text{(PM1-SC-e)}$$

where:

- $\Upsilon^u = \{(g,t) : \tilde{z}_{g,t} = 1\}$ — predicted ON
- $\Upsilon^d = \{(g,t) : \tilde{z}_{g,t} = 0\}$ — predicted OFF
- $\bar{C}$ — cost upper bound (all units at max generation)

- $\underline{C}$ — LP relaxation lower bound
- $K \in (0, 1]$ — cost satisfaction tuning factor

18.1 Understanding Each Constraint

(PM1-SC-a): ON prediction constraint. For every (g, t) where the GNN predicted ON: the fuzzy solution must also be ON, or λ must drop below $z^F_{g,t}$. Since $z^F_{g,t} \in \{0, 1\}$, if the solver sets $z^F_{g,t} = 0$ (overriding the ON prediction), then $\lambda \leq 0$, driving λ to zero.

(PM1-SC-b): OFF prediction constraint. For every (g, t) where the GNN predicted OFF: the fuzzy solution must also be OFF, or λ must be small. If the solver needs to turn a predicted-OFF generator ON (for feasibility), $\lambda \leq 1 - z^F_{g,t} = 0$.

(PM1-SC-c): Cost satisfaction. The fuzzy solution's cost cannot be much worse than the optimal. The K-factor $K \in (0, 1]$ controls how much cost degradation is acceptable. $K = 1$ is standard.

(PM1-SC-d$\star$): N-1 security — the key extension. $G^{SC}(x^F) \leq H^{SC}$ embeds the *full* security-constrained MILP directly: power balance, all surviving-line thermal limits, ramping, uptime/downtime, and reserve constraints for every contingency.

[!] Concept: Key extension over Venkatesh et al.

Venkatesh et al.'s PM1 uses only $G^{(0)}(x^F) \leq H^{(0)}$ — base-case constraints only. Our PM1-SC uses $G^{SC}(x^F) \leq H^{SC}$ — all 41 N-1 contingency constraints embedded simultaneously. This is why our method guarantees N-1 security; theirs does not measure it at all.

(PM1-SC-e): Domain and variable definition. λ is bounded between 0 and 1. The joint decision vector x^F includes the binary commitments z^F, continuous dispatch p^F, and the fuzzy objective λ.

19 The Feasibility Guarantee

Proposition .1 (Feasibility of PM1-SC)**.** *If the base SCUC problem $G^{SC}(x) \leq H^{SC}$ has a feasible solution, then PM1-SC has a feasible solution for any GNN prediction $\tilde{z}$ and any $K \in (0, 1]$.*

Proof. We show that $\lambda = 0$ with any feasible SCUC solution x^F is always a feasible point of PM1-SC.

Substitute $\lambda = 0$ into each constraint:

(PM1-SC-a): $0 \leq z^F_{g,t}$ for all $(g,t) \in \Upsilon^u$. Trivially satisfied since $z^F_{g,t} \in \{0,1\} \geq 0$.

(PM1-SC-b): $z^F_{g,t} \leq 1$ for all $(g,t) \in \Upsilon^d$. Trivially satisfied since $z^F_{g,t} \in \{0,1\} \leq 1$.

(PM1-SC-c): $C(z^F) \leq \bar{C}$. Satisfied by construction since $\bar{C}$ is defined as an upper bound on the cost over all feasible points.

(PM1-SC-d$\star$): $G^{SC}(x^F) \leq H^{SC}$. Satisfied by the base SCUC feasibility assumption.

(PM1-SC-e): $0 \leq 0 \leq 1$. Trivially satisfied.

Therefore $\lambda = 0$ is always feasible, and the PM1-SC feasible set is non-empty whenever the base SCUC is feasible. □ □

Remark .1. *The proof is* topology-agnostic*: it holds for any contingency* $c \in \{0\} \cup \Omega^{(c)}_L$*, including contingencies not seen during training. NFR = 1.0 is therefore a* mathematical guarantee*, not an empirical observation.*

20 Per-Generator Accuracy

Table 2: Per-Generator Accuracy on N-1 Test Samples (1,171 samples)

Generator	Bus	$P^{\max}$	CNN-BiLSTM	GNN-UC
Gen 0	1	80 MW	1.0000	1.0000
Gen 1	2	80 MW	1.0000	1.0000
Gen 2	13	40 MW	0.6080	**0.8390** (+23.1 pp)
Gen 3	22	50 MW	**0.3490**	0.7890
Gen 4	23	30 MW	**0.9020**	0.8550
Gen 5	27	150 MW	0.4580	**0.8970** (+43.9 pp)
Mean accuracy			0.7196	**0.8968**
Cap-weighted accuracy			0.8013	**0.9146**
Mean Hamming distance			40.4 bits	**14.9 bits**

Bold = winner for that generator. pp = percentage points.

Reading this table:

- Gen 0 and Gen 1 are trivial — both models get them right. These are large base-load units near the slack bus that are almost always ON.
- **Gen 5 (150 MW)** is the most important generator by capacity. The CNN gets it right only 46% of the time — barely better than random. The GNN gets it right 90% of the time. This 44 pp gap on the largest generator dominates the capacity-weighted accuracy.
- Gen 3 and Gen 4: the CNN wins. But the CNN wins by predicting the base-case pattern for every contingency — empirically, it produces identical schedules for all

41 contingencies. Its accuracy on Gen 3 is 0.349% — essentially it got lucky by over-committing.

21 NFR: The Feasibility Metric

[*] Key Result

Method	NFR	SOI
CNN-BiLSTM alone	1.0000†	0.1501
GNN-UC alone	0.9488	**0.0561**⋆
GNN-UC + PM1-SC	**1.0000**‡	**0.0561**
Venkatesh et al. PM1	N/A	0.0600

† Via over-commitment (SOI 2.7× worse)

⋆ Beats Venkatesh et al. on harder N-1 test

‡ Guaranteed by Proposition .1

22 Topology Sensitivity: Direct Evidence

We ran the CNN baseline and GNN on five consecutive test samples from the same scenario with different contingencies. The CNN produced:

```
                Gen:  [G0, G1, G2, G3, G4, G5]
Contingency -1 (base):  ON-hours = [24, 24, 23, 20, 16, 24]
   Contingency 0 :  ON-hours = [24, 24, 23, 20, 16, 24]
   Contingency 1 :  ON-hours = [24, 24, 23, 20, 16, 24]
   Contingency 2 :  ON-hours = [24, 24, 23, 20, 16, 24]
   Contingency 3 :  ON-hours = [24, 24, 23, 20, 16, 24]
```

Identical for all contingencies. The GNN produced distinct schedules for each contingency, adapting its commitment decisions based on which line was tripped.

This is empirical confirmation of the structural limitation we claimed.