Learning to Predict Security Constraints for Large-Scale Unit Commitment Problems

Rafael Sterzinger
ETH Zürich
Zürich, Switzerland
rsterzinger@ethz.ch

Jan Poland
Hitachi Energy Research
Baden-Dättwil, Switzerland
jan.poland@hitachienergy.com

Max B. Paulus
ETH Zürich
Zürich, Switzerland
max.paulus@inf.ethz.ch

Didier Chételat
Polytechnique Montréal
Montréal, Canada
didier.chetelat@polymtl.ca

***Abstract*—Ahead-of-time electricity generation and consumption scheduling, also known as unit commitment, is essential to operate power grids. Today, it is usually formulated and solved as a mixed-integer program. To ensure robustness against operational contingencies, a large number of security constraints must be considered, which significantly increases the problem's complexity. However, only a small subset of these constraints is typically *active* in the solution. Conventional solving approaches attempt to eliminate redundant security constraints through a computationally expensive iterative search. In this paper, we study machine learning approaches to alleviate this search by predicting a set of *relevant* security constraints to retain. We propose a new neural network architecture based on graph convolutions and the attention mechanism, which we evaluate on synthetic unit commitment problems for nine power grids. Compared to the conventional iterative approach and previous machine learning-based methods, our architecture improves average and (more importantly) worst-case solving times by a factor of 2 to 3.**

***Index Terms*—Unit Commitment, Security Constraints, Generator Scheduling, Mixed-Integer Programming, Graph Neural Network, Transformer Neural Network, Attention Mechanism**

I. Introduction

Unit commitment (UC), which refers to ahead-of-time scheduling of electricity generation and consumption, is a routinely executed task in the operation of power grids. An important variant of this is security constrained unit commitment (SCUC) in which a cost-optimal schedule needs to be generated that additionally ensures reliable power delivery even in the presence of potential contingencies [1], [2]. Contingencies can be caused by failing transmission lines or components of the grid: the commonly studied setup is that of $N-1$ transmission contingencies. In this setup, any single line is considered to potentially fail, which might result in an unsafe grid operation due to an overload on another line. Hence, in the presence of N lines in the grid, this security requirement gives rise to $O(N^2)$ constraints to be satisfied. As these security constraints can be expressed in terms of power flow equations, it is also common to refer to them as *network constraints*.

Mixed-Integer Programming. Today's industry standard approach to solve UC and SCUC problems is to formulate them as a mixed-integer program (MIP) [3]. Since this strategy was first shown to be capable of solving UC problems around the year 2000 [4], MIP solvers have significantly matured and become the primary strategy. The standard formulation is, in principle, non-linear, but there are common ways to linearize the non-linear terms in the objective function and the constraints such that the problem can be viewed as a mixed-integer *linear* program. In particular, it is common practice to linearize the $O(N^2)$ network constraints by means of injection shift factors (ISFs) [5] so they can easily be integrated into a linear MIP.

Iterative Security Constraint Selection. Despite the successes of this strategy, the $O(N^2)$ complexity of security constraints limit the direct solving of SCUC problems to all but the smaller grids encountered in practice. Indeed, the ISFs lead to dense constraints, so that even stating a MIP with $O(N^2)$ constraints consumes $O(N^3)$ memory. Instead, an *iterative approach* [2] has been established as the industry standard, in which an SCUC problem is solved as a sequence of network constrained unit commitment (NCUC) problems. The process starts with a standard UC problem without any network constraints enforced, and its corresponding solution is analyzed to identify potential security constraint violations. Subsequently, a subset of violated security constraints is added, and the process is iterated until no more violations occur. This approach results in a final NCUC problem with much less than $O(N^2)$ constraints but with the same solution as the complete SCUC problem. In fact, it was observed in [6] that typically a small subset of security constraints is already sufficient to imply the satisfaction of its complement. This work is based on the conventional iterative approach also used in [2], [5], [7] and others.

Machine Learning. Trends these past decades in energy markets have led generation capacities and grids to increase in volume and granularity, while simultaneous surges in supply volatility have led to pressure to increase the temporal resolution of schedules. In order to keep up the ability to solve UC problems in real-time (e.g., achieving an optimality gap of 0.1% consistently [8]), further technical advancements have become required. Recently, there has been an increase of interest by the combinatorial optimization community in the exploitation of statistical regularities by machine learning (ML) techniques when similar problems are repeatedly solved (see [9] and references therein). As noted in [5], the fact that

979-8-3503-9678-2/23/$31.00 ©2023 IEEE

a grid operator must solve similar SCUC tasks on a regular basis means that they are a prime target for such innovations. Already, ML has been employed to learn generation commitment and warm starts [5], [7], [10], network constraints [5], patterns such as affine subspaces to support efficient solving of MIPs [5], and generative models [11].

Contribution. In the iterative SCUC solving procedure described above, one must select among the most violated constraints at every iteration. Inspired by recent successes of ML approaches in the area of UC, we propose to use ML methods to complement the iterative approach by a first step *relevant* network constraint prediction based on the characteristics of a given SCUC task to reduce its computational cost as much as possible. In detail, we propose the following procedure:

1. Read the specification of the SCUC (compare Table I).
2. Predict a set of *relevant* network constraints and add it to the MIP formulation.
3. Solve the actual MIP iteration.
4. Check if further network constraints are violated:
   - If yes, add constraints as in [2] and go to step 3.
   - If no, terminate and return the UC solution.

In the same vein as recent work, e.g., by [5] or [10], ours explores another way for harnessing ML to benefit SCUC solving which, in particular, improves upon our selected baseline: a k-nearest neighbors (KNN) approach by [5] (see also Section IV).

II. Security Constrained Unit Commitment

Consider an SCUC task defined on a grid of B buses, G generation units, and N lines/branches. Our work is based on an SCUC formulation following the conventional approach: we employ the same MIP model as introduced in [3] and later used, e.g., in [5] and [10]. The SCUC task is formulated as an optimization problem over the entire network and set of generators over a given planning horizon of, e.g., 24 hours. Its core elements are:

1. Continuous decision variables: Generation levels for all generation units during the planning horizon.
2. Binary decision variables: Indicators representing the on/off status of generation units and start-up/shut-down decisions during the planning horizon.
3. Linear demand constraints, ensuring that energy supply meets the predicted demand.
4. Linear reserve constraints, ensuring the production schedule accommodates a reserve necessary to react to deviations from the predicted demand.
5. Linear generator ramp constraints, ensuring that the maximum ramp rates of generation units are respected.
6. Linear minimum up- and downtime constraints, ensuring that generation units remain in a state for at least a minimum required time.
7. Linear generator linking constraints, connecting the generator-related decision variables and ensuring that the logic of running/starting up/shutting down is correct.
8. Linearized network constraints, ensuring that thermal limits of transmission lines are respected.
9. A linear or linearized cost function, driving the optimization towards the minimum total cost of the UC schedule.

For the sake of compactness, we omit the details of the MIP formulation here and refer to [5] instead. The data defining a specific UC instance, which also constitutes the input features for our ML models, are summarized in Table I.

Type	Data	$\forall t$	Description
Gen.	energy_cost_per_mwh	x	marginal cost of generator
	running_cost_per_h	x	running cost of generator
	startup_cost	x	startup cost for generator
	pmin_mw		min. energy offer capacity
	pmax_mw		max. energy offer capacity
	min_on_h		minimum uptime
	min_off_h		minimum downtime
Bus	load_per_h	x	energy demand at bus
Con.	OTDF		contingency related DFs
	flow_lim		branch power flow limit
Time	day_sin		day encoded with sine
	hour_sin		hour encoded with sine
	day_cos		day encoded with cosine
	hour_cos		hour encoded with cosine

TABLE I: Data defining a specific UC task. It consists of generator-related data (costs, power limits, minimum up/down time), demand-related data, network-related data, and time encodings. The contingency-related distribution factors (DFs), derived from ISFs, are a dense tensor in $\mathbb{R}^{N\times N\times B}$ also referred to as *outage transfer distribution factors* (OTDFs) [12]. Each entry in $\mathrm{OTDF}_{b,c}$ is a real vector in $\mathbb{R}^B$ that captures the sensitivity of power flow at transmission line b based on variations of bus power balances under a normal/contingency condition c. Some features are defined per time step over the scheduling horizon; those entries are marked in the $\forall t$ column.

III. Teacher

A network constraint can be expressed by a pair $(b, c) \in \{1 \dots N\} \times \{1 \dots N\}$. Let D be the data defining an SCUC instance – compare Table I. Let $\mathrm{MIP} : D \times \mathcal{C} \times \mathcal{P} \mapsto x$ be the MIP solving process which takes as input the data D, an actual set of network constraints $\mathcal{C} = (b_k, c_k)_{k=1}^K$, and parameters $\mathcal{P}$ comprising, e.g., the required MIP gap, and returns a UC solution x. We call a set of network constraints $\mathcal{C}$ *sufficient* if the solution $x = \mathrm{MIP}(D, \mathcal{C}, \mathcal{P})$ satisfies *all* network constraints $(b, c) \in \{1 \dots N\} \times \{1 \dots N\}$.

The standard iterative SCUC solution approach [2] yields a solved SCUC instance comprising a final sufficient set of network constraints – the *default set*. For training, we take the union of default sets of solved SCUC tasks for which we predict the activeness of a constraint. Consequently, this makes predictions for constraints outside this union impossible, but the chance of an unobserved constraint being active is generally low (see Table IV). Additionally, note that the cardinality K of this union verifies that $K \ll N^2$, which typically ranges between 50 to 300 unique (b, c)-pairs in our setup.

Predicting a good set of network constraints is not a well-defined supervised learning task. It is unclear if default sets are the best possible for efficient and robust MIP solving.

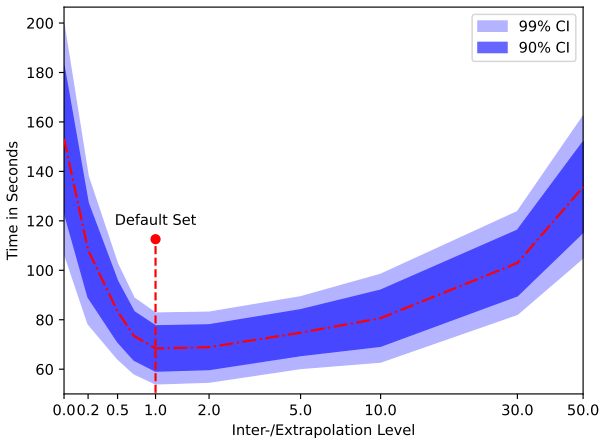

Fig. 1: Study, based on the *ieee118* grid setup, on the one-shot solving time (with SCIP [13]) for different inter-/extrapolated network constraint sets. On average, the default set is roughly the best-performing one, as it gives rise to the lowest one-shot solving time; both larger or smaller constraint sets seem less favorable. The x-axis is normalized such that 0 corresponds to a greedily found minimal set, which, on average, is roughly 55% of the size of the default set.

It might be that:

1. smaller (sufficient) constraint sets are more favorable since they put less load on the MIP solver, and/or
2. larger (sufficient) constraint sets are more favorable since they restrict the search space and thus facilitate faster convergence.

In order to understand the situation, we executed a study on the *ieee118* grid setup (see Section V for details on the different test setups). Starting from a solved SCUC instance and a resulting default set of network constraints, the set is both decreased (*interpolated*) to an approximated minimum set of sufficient network constraints by removing redundant constraints greedily – depending on the SCUC instance a reduction to about 55% of the initial size is possible – and increased (*extrapolated*) by adding random constraints. Illustrated in Fig. 1 are the results of this study; we observe that the default set roughly delivers the best performance.

Although this study provides some insight into constraint sets, it is not exhaustive enough to entirely rule out the possibility that better sets may exist. For instance, different construction methods or grid setups may yield different results, posing an option for follow-up research. Still, we decided to use the default set as our training targets, justifying a supervised learning approach for our ML models. Subsequently, we refer to the set of network constraints to be predicted – the constraints contained in the default set – as *relevant* network constraints.

	F1 Score	Precision	Recall
initial model	0.617±0.015	0.754±0.023	0.561±0.034
+ normalization	0.667±0.006	0.787±0.005	0.590±0.011
+ time information	**0.745±0.001**	**0.821±0.011**	0.696±0.019
+ weighted	**0.742±0.004**	0.658±0.004	**0.872±0.002**

TABLE II: Ablation study of our GNN architecture on the *ACTIVSg2000* power grid. Initially, our model used layer normalization, which we swapped for batch normalization. By additionally adding feature standardization, our model improved by around 5%. Next, we exploited seasonal trends within a day/year by incorporating time information encoded via trigonometric functions to obtain an additional boost of around 8%. Lastly, we added an overall class weighting, capped at three, to trade off precision with recall.

IV. Learners

A. k-Nearest Neighbors (Data-Driven Baseline)

A KNN approach has been proposed by [5] for the problem of predicting sets of network constraints, which we use as a baseline. Given a pool of solved SCUC instances, the KNN predictor evaluates the k nearest neighbors (we use $k = 25$) and decides, via majority voting, for each network constraint, if it should be enforced in the MIP from the beginning.

As for input features, we extend the list denoted in Table I: before the KNN predictor is employed, we execute the first iteration of the SCUC solving procedure; its solution provides initial violations of network constraints which we add as additional features. The computational cost is generally low as no network constraints are enforced yet. Note that the subsequently described architecture does not see these additional features.

B. Neural Networks with Graph Convolutions and Attention

The principal ML architecture developed in this work is a neural network with graph convolutions and attention, outlined in Fig. 2. We refer to it as a graph neural network (GNN), which takes the input features listed in Table I related to constraints, generators, and buses, to then processes them through a trainable embedding into one common dimension ($d = 64$). Moreover, we include dropout layers with a rate of 20% to counteract the risk of overfitting.

With these embedded feature vectors, we perform two interleaved half-convolutions (as proposed by [9] and used in [14]), i.e., one convolution from buses to generators and one from generators to buses. The resulting bus embeddings are further processed with a multilayer perceptron (MLP).

The following two stages consist of multi-head attention layers [15]: the first is a self-attention on the bus embeddings; the second is a cross-attention between bus embeddings and constraint embeddings. Similar to [15], attention layers are followed by skip connections, dropout layers, and position-wise MLPs. However, unlike their standard *Transformer*, our input does not receive a positional encoding as it does not impose a natural order.

The final step of our GNN involves an MLP to estimate the probability of a constraint being relevant, i.e., being in

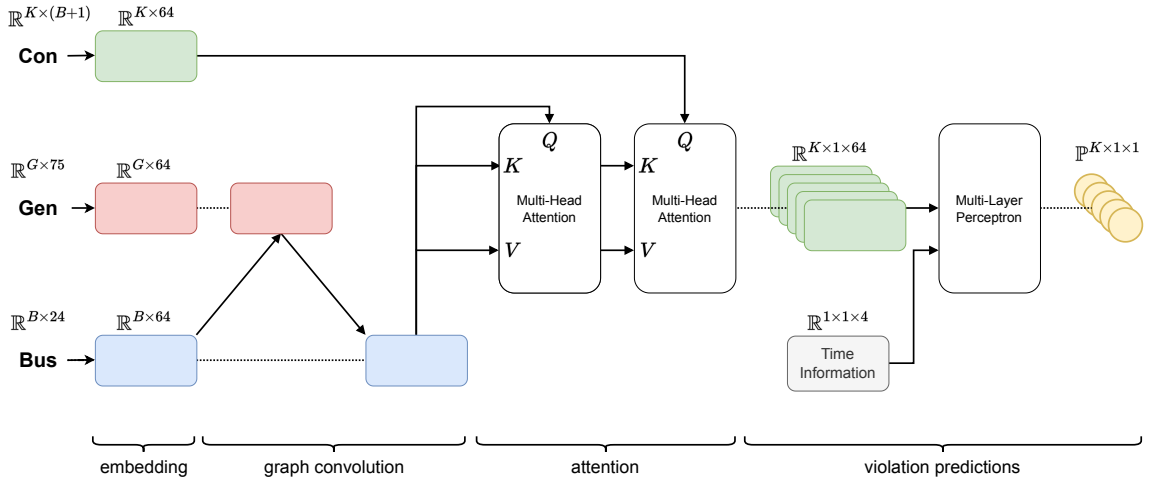

Fig. 2: Neural network architecture comprising graph convolutions and attention layers to predict relevant network constraints.

Grid	# Buses	# Gen.	# Lines	# Instances train/valid/test
ieee118	118	19	186	1024/256/256
sp2736	2,736	289	3,504	512/128/128
case2848rte	2,848	544	3,776	512/128/128
ACTIVSg2000	2,000	542	3,206	512/128/128
sp3120	3,120	483	3,693	512/128/128
pegase2869	2,869	510	4,582	512/128/128
wp2383	2,383	323	2,896	1024/256/256
case6468rte	6,468	1,262	9,000	512/128/128
rte6515	6,515	1,368	9,037	512/128/128

TABLE III: Details of the power grids studied in this work.

Grid	F1 GNN Valid.	F1 GNN Test	F1 KNN Valid.	F1 KNN Test	New Targets
ieee118	0.900	0.894	0.892	0.887	0.0%
sp2736	0.750	0.759	0.811	0.811	2.3%
case2848rte	0.946	0.951	0.928	0.930	0.7%
ACTIVSg2000	0.743	0.743	0.755	0.761	3.9%
sp3120	0.824	0.821	0.837	0.835	2.3%
pegase2869	0.859	0.858	0.820	0.818	2.3%
wp2383	0.704	0.711	0.768	0.776	1.6%
case6468rte	0.829	0.832	0.333	0.334	1.6%
rte6515	0.802	0.804	0.296	0.298	2.3%

TABLE IV: An overview of the predictive performance of our models. In the last column, we denote the percentage of validation and test instances that contained network constraints not encountered in the union of training instances.

the target set. This estimation is based on the final constraint embeddings and time encodings (day, hour) for our SCUC task. Employing the *binary cross-entropy loss*, we calculate an error between these predictions and our ground-truth default set to adjust the weights of our model.

Ablation Study. Our primary ablation study on the GNN architecture is presented in Table II. In addition to the steps documented there, the following variations have been explored but did not yield significant changes in performance: variations of the model width and depth, additional features composed of statistics and peaks of costs and loads, and adjusting the learning rate/batch size.

V. Computational Experiments

A. Test Cases

Our computational studies are based on transmission-level UC tasks with hourly scheduling for a planning horizon of 24 hours, randomly generated by a procedure, described in [10]. A task is defined by the following elements:

- a transmission grid sourced from [16];
- forecasted load curves sampled from a stochastic process model identified from [17];
- generation bids sourced from [17];
- and additional constraints, such as operational reserves.

Table III shows an overview of the grids in our setup with detailed specifications and the number of instances sampled per grid. Each instance in the training set is solved to 1% optimality using the baseline iterative approach. From this data, separate KNN and GNN models are trained based on the security constraints observed. Table IV depicts the predictive performance of these models. Additionally, we note that only a small number of new security constraints were observed during inference.

Finally, Fig. 3 displays the performance of our predictive models employed in the framework of SCUC solving. When evaluating the performance, we focus on the toughest instances of a test setup: the *worst-case* performance is a highly relevant evaluation metric in practice because it determines the computational budget, i.e., if the solving time exceeds the budget, there will be no (updated) schedule available.

B. Results

What follows are the observations from our experiments:

- There is a clear performance ranking: GNN performs better than KNN, and KNN performs better than the iterative baseline, i.e., no predictions being performed.
- The benefits of having a prediction significantly vary between the test setups/power grids. On a single

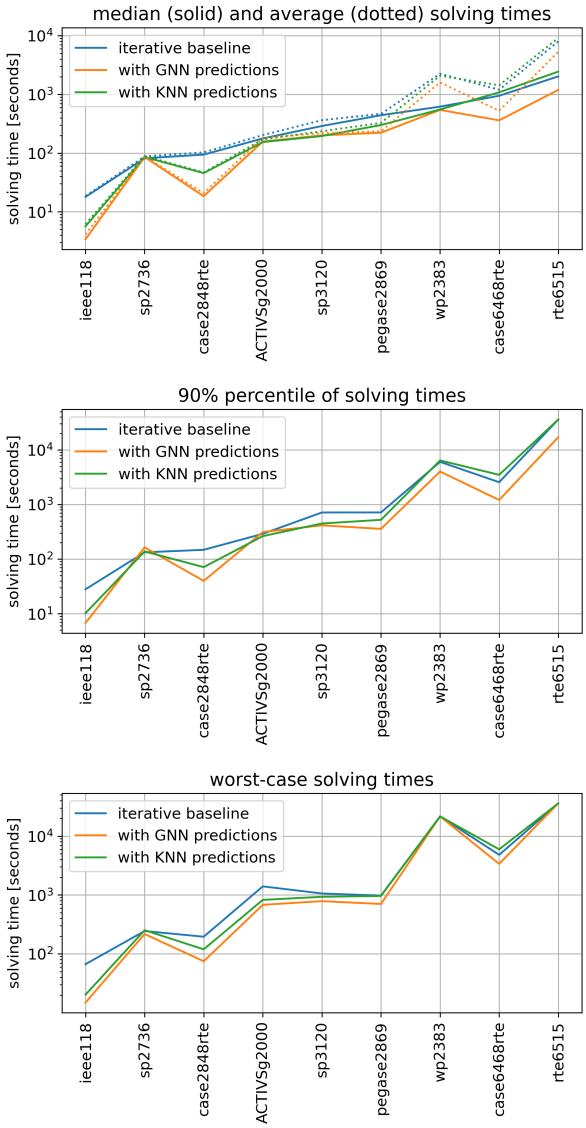

Fig. 3: Performance evaluation of the different network constraint predictors: we denote the computation time (lower is better) to fully solve SCUC tasks to 1% optimality with Gurobi v9.5.1 [18] as our solver. There is the typical performance (median and average, top plot), performance on tough instances (90% percentile, middle plot), and performance on the toughest instances (bottom plot).

grid (*sp2736*), none of the predictive models can decrease the solving time (neither typical nor worst-case).

- For larger grids, even with predictions, the SCUC solving process might run into timeouts (10-hour computational budget). This is visible through the saturating worst-case solving times for *wp2383* and *rte6516*; the instances at the 90% percentile saturate much less frequently.
- A typical speedup of the GNN relative to the iterative baseline is about a factor of 2 to 3 on tough instances (note the logarithmically scaled y-axis of the plots).

VI. Summary

We have introduced a new neural network architecture that combines graph convolutions and the attention mechanism to predict relevant network constraints for SCUC problems. When evaluating our proposed GNN in the setting of nine power grids, we found that it improved average solving time on a per-instance basis compared to our baselines. In particular, our approach significantly reduced the computational budget required for tough instances by a factor of 2 to 3 concerning the conventional iterative approach, the current industry standard. Lastly, we have identified opportunities for follow-up research on discovering even stronger sets of network constraints, e.g., through reinforcement learning.

References

[1] N. Padhy, "Unit Commitment - A Bibliographical Survey," *IEEE Transactions on Power Systems*, vol. 19, no. 2, 2004.

[2] Á. S. Xavier, F. Qiu, F. Wang, and P. R. Thimmapuram, "Transmission Constraint Filtering in Large-Scale Security-Constrained Unit Commitment," *IEEE Transactions on Power Systems*, vol. 34, May 2019.

[3] G. Morales-España, C. Gentile, and A. Ramos, "Tight and Compact MILP Formulation for the Thermal Unit Commitment Problem," *IEEE Transactions on Power Systems*, vol. 28, no. 4, 2013.

[4] B. Hobbs, M. Rothkopf, R. O'Neill, and H. Chao, *The Next Generation of Electric Power Unit Commitment Models*. New York: Springer, 2001.

[5] Á. S. Xavier, F. Qiu, and S. Ahmed, "Learning to Solve Large-Scale Security-Constrained Unit Commitment Problems," *INFORMS Journal on Computing*, vol. 33, May 2021.

[6] F. Bouffard, F. D. Galiana, and J. M. Arroyo, "Umbrella Contingencies in Security-Constrained Optimal Power Flow," *15th Power systems computation conference, PSCC*, vol. 5, 2005.

[7] S. Schmitt, I. Harjunkoski, M. Giuntoli, J. Poland, and X. Feng, "Fast Solution of Unit Commitment Using Machine Learning Approaches," in *2022 IEEE 7th International Energy Conference*, May 2022.

[8] Y. Chen, A. Casto, F. Wang, Q. Wang, X. Wang, and J. Wan, "Improving Large Scale Day-Ahead Security Constrained Unit Commitment Performance," *IEEE Transactions on Power Systems*, vol. 31, Nov. 2016.

[9] M. Gasse, D. Chételat, N. Ferroni, L. Charlin, and A. Lodi, "Exact Combinatorial Optimization with Graph Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 32*, 2019.

[10] J. Poland, I. Harjunkoski, S. Schmitt, M. Giuntoli, and X. Feng, "Machine Learning Assisted Optimization for Unit Commitment," in *CIGRE Kyoto Symposium*, 2022.

[11] V. Nair, S. Bartunov, F. Gimeno, I. von Glehn, P. Lichocki, I. Lobov, B. O'Donoghue, N. Sonnerat, C. Tjandraatmadja, P. Wang, R. Addanki, T. Hapuarachchi, T. Keck, J. Keeling, P. Kohli, I. Ktena, Y. Li, O. Vinyals, and Y. Zwols, "Solving Mixed Integer Programs Using Neural Networks," July 2021.

[12] Y. C. Chen, A. D. Dominguez-Garcia, and P. W. Sauer, "Measurement-Based Estimation of Linear Sensitivity Distribution Factors and Applications," *IEEE Transactions on Power Systems*, vol. 29, May 2014.

[13] G. Gamrath, D. Anderson, K. Bestuzheva, W.-K. Chen, L. Eifler, M. Gasse, P. Gemander, A. Gleixner, L. Gottwald, K. Halbig, G. Hendel, C. Hojny, T. Koch, P. Le Bodic, S. J. Maher, F. Matter, M. Miltenberger, E. Mühmer, B. Müller, M. E. Pfetsch, F. Schlösser, F. Serrano, Y. Shinano, C. Tawfik, S. Vigerske, F. Wegscheider, D. Weninger, and J. Witzig, "The SCIP optimization suite 7.0," ZIB-Report 20-10, Zuse Institute Berlin, Mar. 2020.

[14] M. B. Paulus, G. Zarpellon, A. Krause, L. Charlin, and C. J. Maddison, "Learning to Cut by Looking Ahead: Cutting Plane Selection via Imitation Learning," in *Proc. International Conference on Machine Learning (ICML)*, July 2022.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," Dec. 2017.

[16] S. Babaeinejadsarookolaee et al., "The Power Grid Library for Benchmarking AC Optimal Power Flow Algorithms," Jan. 2021.

[17] "Data Miner 2." https://dataminer2.pjm.com/list, Mar. 2020.

[18] Gurobi, "Optimizer Reference Manual," 2023.