Les Cahiers du GERAD ISSN: 0711–2440

Learning-accelerated mixed-integer quadratic programming for unit commitment

P. Maisonneuve, A. Lesage-Landry

G–2024–08

January 2024

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Learning-accelerated mixed-integer quadratic programming for unit commitment

Philippe Maisonneuve [a]

Antoine Lesage-Landry [a, b, c]

[a] *Department of Electrical Engineering, Polytechnique Montréal, Montréal (Qc), Canada, H3T 1J4*

[b] *GERAD, Montréal (Qc), Canada, H3T 1J4*

[c] *Mila, Montréal (Qc), Canada, H2S 3H1*

philippe.maisonneuve@polymtl.ca
antoine.lesage-landry@polymtl.ca

January 2024
Les Cahiers du GERAD
G–2024–08

Abstract : In this work, we improve the efficiency of Unit Commitment (UC) optimization solvers using a Graph Convolutional Neural Network (GCNN). In power systems, UC is crucial as it entails making essential decisions regarding the scheduling of power generation units to effectively meet the demand within a specific period. The complex nature of UC, arising from the binary nature of the problem and the non-convexity of the power flow constraints, warrants their representation as mixed-integer second-order cone program (MISOCP). Harnessing the inherent structure of these mixed-integer programs, we represent them as variable-constraint k-partite graphs. Utilizing a GCNN, we extract valuable insights from these graphs, learning effective variable selection policies for the branch and bound algorithm, thereby accelerating the overall optimization process. Our method ensures the preservation of solution optimality contrary to end-to-end learning approaches. Our methodology unfolds in two parts: (1) we construct a k-partite graph from the constraints of the optimization problem (2) we subsequently train our GCNN model on this graph representation via imitation learning, using the strong branching expert rule as our guide. Our approach stands out by effectively integrating problem-specific information into the variable selection within the branch and bound process of optimization.

Keywords : Branch and bound method, graph convolutional neural networks, integer programming, unit commitment

Acknowledgements: This work was funded by the Institute for Data Valorization (IVADO) and by the National Science and Engineering Research Council of Canada (NSERC).

1 Introduction

Unit Commitment (UC) is a fundamental decision-making process in power systems, involving the optimal scheduling of power generation units to meet demand over a specified period [32]. It is a complex optimization problem that takes into account numerous factors such as power demand, generation costs, and operational constraints to ensure cost-effective and reliable power delivery. Solving the UC problem quickly and accurately is crucial because it directly impacts the operational efficiency of power systems. Fast, accurate solutions minimizing operational costs, ensure a reliable power supply, and allow for a rapid response to fluctuations in power demand or unexpected events, thus significantly enhancing grid stability and performance [29].

In our work, we focus on the UC problem where the goal is to schedule over a time horizon power generation units to meet power demand at the minimum possible cost. This approach ensures an optimal balance between meeting operational constraints and achieving economic efficiency.

Building on a methodology proven effective in the linear case [24], we propose learning-accelerated mixed-integer convex second-order cone programming (MISOCP) and convex quadratic programming (MICQP) methods. Our approach allows for the inclusion of non-linear constraints and objectives and leads to a more accurate representation of power systems, thus enhancing the decision-making process.

Our methodology unfolds as follows. First, we construct a k-partite graph from the problem formulation, where at least one set of nodes corresponds to variables and another set corresponds to the problems constraints. In a nuanced adaptation of existing methodology [16], we extend the graph representation by converting the variable order into an edge feature, thereby enriching the structure of the graph with additional information. This modification provides a more comprehensive view of the problem, facilitating a more accurate learning process. Then, when considering the power flow within the UC formulation, we rely on a new representation strategy for second-order cone (SOC) constraints using a tripartite graph. This tripartite graph structure divides the problem elements into three interlinked sets: the decision variables, the bilinear components modelling the product of these variables, and the constraints. Finally, we train a graph convolutional neural network (GCNN) model on this graph representation via imitation learning, using the strong branching expert rule as our guide.

Our methodology uniquely synthesizes problem-centric data into variable selection, which differentiate it from conventional heuristics in branch and bound algorithms. Notably in power engineering, the wealth of data is generated daily due to the repetitive nature of problem-solving [25]. This significant and ever-growing trove of data can be employed to improve the resolution speed of these recurring problems using data-driven solvers like ours.

1.1 Contributions

In this research, we make the following contributions to the field of combinatorial optimization and machine learning in power systems:

- We develop a novel method that expands upon [16] to accommodate mixed-integer non-linear optimization problems, particularly MISOCPs and MICQPs. Our strategy incorporates additional edge features into the graph representation and k-partite graphs, enabling the model to capture the non-linear characteristics inherent in the problems. This aspect is not covered by existing methodologies.
- We apply our methodology to UC problems, demonstrating its versatility and effectiveness in handling complex, real-world issues in power engineering. Our approach consistently accelerates the computation process. This expedited resolution is particularly evident in formulations with and without power flow considerations, showcasing the broad scope of our method.
- We implement our method for the UC problem in which SOC-relaxed power flow constraints are integrated and subsequently evaluate its efficacy on standard power grid test cases. To further

extend its applicability, we adapt our model to MICQPs. We validate this specialized model on four larger, standard power grid models derived from the French RTE model.

The rest of this paper is organized as follows. In Section 1.2, we review the relevant literature. In Section 2, we present the UC formulation to be solved via the methodology detailed in Section 3. Section 4 provides the results of our numerical experiments. Lastly, we conclude in Section 5.

1.2 Related work

We now review the relevant literature. The UC problem presents a critical challenge in power system operation, involving the optimal scheduling of power generation units to meet the forecasted load demand at the least possible cost, all while adhering to a variety of operational constraints. In this study, we focus on two different formulations of the UC problem: one solely accounting for economic dispatch (ED) and another incorporating power flow (PF) constraints.

The UC problem seeks to minimize the power generation's operational costs by optimally allocating the load among the online units. Solved typically using mixed-integer programming (MIP), our first UC implementation simplifies the problem by excluding network constraints, therefore, by assuming unrestricted power flow within the grid. This abstraction aids in achieving scalability, allowing efficient management of larger and more complex systems [32]. We refer to this implementation as unit commitment-economic dispatch (UC-ED).

Building on UC-ED, the unit commitment with power flow (UC-PF) incorporates grid-specific transmission constraints, such as transmission line limits and bus voltage magnitude bounds, into the UC problem. These constraints become significantly relevant as total load increases and congestion emerges as a critical factor. Although this formulation offers a more accurate model of the power system's operations, it significantly increases the computational burden due to its non-convexity [7]. This problem is typically solved using a convex relaxation, which offers a higher degree of realism in comparison to approximations [32] but also brings with it an increased computational burden. Particularly, our work leverages the SOC relaxation introduced in [9].

Reference [11] presents innovative solutions to the UC problem in large-scale power systems using decomposition techniques. Specifically, Benders' decomposition is employed to solve a SOC relaxation of the UC-PF problem, ensuring power flow feasibility through a sequence of continuous convex optimization problems. Reference [10] proposes a Benders-type decomposition with a dynamically enriched master problem. They further utilize Kron reductions to compact the description of under-contingency operating conditions. These decomposition techniques break down the complex UC problem into manageable subproblems, enabling efficient solutions in large-scale systems.

In recent years, the burgeoning field of machine learning (ML) has been leveraged to tackle complex problems, like the UC. One such approach is outlined by [25] where they demonstrated a simple yet effective strategy using ML to solve the UC problem. Their work suggests that even naive algorithms, guided by past UC instances, can provide optimal or near-optimal solutions with significant speedups. They emphasize that any sophistication in the learning method should correspond to a statistically significant improvement in the results. Some of these approach known as end-to-end learning approximately the optimal solution of those problems sacrificing the guarantee of optimality that would be provided by a more conventional optimization method [33]. Over the years, ML has been adopted in various forms to tackle the UC problem. Notably, an important body of literature focuses on linear formulations, often yielding only approximate solutions [12, 13]. [35] employed machine learning to improve MIP solver performance by predicting constraints and solution areas, demonstrating resilience to data shifts. Reference [27] reformulates the UC problem as a Markov decision process and employs a multi-step deep reinforcement learning-based algorithm. The method proved more computationally efficient than traditional optimization methods, achieving similar levels of optimality but with significantly shorter computation times. However, the size of the tested instances is significantly smaller than the standard problems solved in industry. Our work aims to leverage machine learning to accelerate

the solution process of both UC-PF and UC-ED while maintaining the quality of the solution. We use ML to enhance the branch and bound process, thereby conserving the efficiency while preserving the exactness of the solver. We employ a hybrid method that integrates machine learning to reduce the computation time of classical solvers for UC problems. Specifically, we enhance the variable selection process of the branch and bound algorithm, resulting in more efficient problem-solving. This guarantees the optimality and accuracy of our solution while enabling faster solving.

The use of machine learning to improve the variable selection in branch and bound algorithms for solving combinatorial optimization problems is a well-studied research area [6]. One of the common approaches is to use imitation learning [20], a form of supervised learning that learns to replicate the decisions of an expert algorithm. For instance, a key challenge in MIP is how to represent these problems in a way that can be processed by a machine learning algorithm, which led to the distinction between static and dynamic features [2].

Recent work has evolved these techniques further by introducing graph neural networks (GNN) to represent MIPs [21]. The GNN formulation of mixed-integer linear programs (MILPs) led to a more comprehensive representation of problem instances and improved the efficiency of the branch and bound algorithm. However, the practical speed gains from GNNs were contested, with suggestions that hybrid architectures might not be more effective on machines equipped only with CPUs [17].

Machine learning approaches for branch and bound can be trained at different levels of specificity, ranging from instance specific to generalizable across all types of problems. For example, some works focused on developing models that are specialized for problems belonging to the same family [19, 21], while others sought to generalize across all types of instances [4, 14, 31].

Training these models requires amount of information, which often need to be artificially generated due to the limited size of traditional model used for testing optimization problem-solving methods [22, 26]. Collecting the data involves solving numerous instances of the problem and recording the state of the branch and bound tree along with the selected variable.

Metrics for evaluating the performance of these machine learning approaches can be challenging to define, as the real objectives, e.g., computation time, the primal-dual gap, and the size of the explored branch and bound tree are not the same as the ones used to train the machine learning models [21].

Finally, there has been some exploration of reinforcement learning techniques for variable selection. Although theoretically not limited by the performance of the expert heuristics they seek to imitate, these methods currently face several challenges. For example, they require training metrics with a more global view, which is difficult to define, and they are typically slower to train due to the need to solve the optimization problem multiple times per instance [15, 28, 30].

2 UC Formulation

In this section, we present the UC-ED and UC-PF formulations used in this work. The UC-ED formulation is inspired by [32] while the power flow formulation is adapted from [9].

We first introduce the following notation.

Sets	
$\mathcal{N} \subset \mathbb{N}$	is the set of nodes/buses;
$\mathcal{L} \subseteq \mathcal{N} \times \mathcal{N}$	is the set of transmission lines;
$\mathcal{G} \subseteq \mathcal{N}$	is the set of generators;
$\mathcal{T} \subset \mathbb{N}$	is the time horizon.

Parameters	
$a_i, b_i, c_i \geq 0$	are the cost coefficients for generator $i \in \mathcal{G}$;
$\overline{r}_i, \underline{r}_i \geq 0$	are ramp-up/down limits for generator $i \in \mathcal{G}$;
$\overline{T}_i, \underline{T}_i \geq 0$	are minimum up/down times for generator $i \in \mathcal{G}$;
$\overline{v}_i, \underline{v}_i \geq 0$	are voltage magnitude limits at bus $i \in \mathbb{N}$;
$\overline{S}^i, \underline{S}^i \in \mathbb{C}$	are apparent power output limits for generator $i \in \mathcal{G}$;
$S^{\mathrm{d}}_{i,t} \in \mathbb{C}$	is the apparent power demand at bus $i \in \mathbb{N}$ at time $t \in \mathcal{T}$;
$Y_{ij} \in \mathbb{C}$	is the admittance of line $ij \in \mathcal{L}$;
$\overline{S}_{ij} \geq 0$	is the apparent power limit of line $ij \in \mathcal{L}$;
$\theta^{\Delta}_{ij} \in \mathbb{R}$	is the phase angle difference limit of line $ij \in \mathcal{L}$.
Variables	
$p_{i,t}, q_{i,t} \geq 0$	are active and reactive power outputs for generator $i \in \mathcal{G}$ at time $t \in \mathcal{T}$;
$S^{\mathrm{gen}}_{i,t} \in \mathbb{C}$	is the apparent power output for generator $i \in \mathcal{G}$;
$D_t, Q_t \geq 0$	are total active and reactive power demands at time $t \in \mathcal{T}$;
$\xi_{i,t} \in \{0, 1\}$	is the status of generator i $\in \mathcal{G}$ at time $t \in \mathcal{T}$;
$S_{ij,t} \in \mathbb{C}$	is the apparent power flow of line $ij \in \mathcal{L}$ at time $t \in \mathcal{T}$;
$W_{jj,t}, W_{ij,t} \in \mathbb{C}$	are elements of the voltage product matrix at time $t \in \mathcal{T}$;
$x \in \mathbb{R}$	is a slack variable.

Next, we present the MICQP formulation of the UC-ED. We recall that the goal of UC-ED is to determine the optimal operational schedule of power generation units to meet forecasted electricity demand while minimizing operational costs. The UC objective is given by:

$$\sum_{t\in\mathcal{T}}\sum_{g\in\mathcal{G}} a_g p^2_{g,t} + b_g p_{g,t} + C_S \xi_{g,t},$$

which we reformulate as a quadratic constraint for the minimization problem to be compatible with our method:

$$\sum_{t\in\mathcal{T}}\sum_{g\in\mathcal{G}} a_g p^2_{g,t} + b_g p_{g,t} + C_S \xi_{g,t} \leq x. \tag{1}$$

The problem is subject to the following constraints. The power balance constraint ensures that the total generation meets the demand at each time period. It is expressed as:

$$\sum_{g\in\mathcal{G}} p_{g,t} = D_t \quad \forall t \in \mathcal{T} \tag{2}$$

$$\sum_{g\in\mathcal{G}} q_{g,t} = Q_t \quad \forall t \in \mathcal{T}. \tag{3}$$

The output of each unit, when it is online, is constrained by the minimum and maximum generation limits:

$$\overline{S}_g \xi_{g,t} \leq S^{\mathrm{gen}}_{g,t} \leq \underline{S}_g \xi_{g,t}, \quad \forall g \in \mathcal{G}, \forall t \in \mathcal{T}, \tag{4}$$

where,

$$\overline{S}^{\mathrm{gen}}_{g,t} = p_{g,t} + \jmath q_{g,t}, \quad \forall g \in \mathcal{G}, \forall t \in \mathcal{T}. \tag{5}$$

Generators are subject to the minimum up and down time requirements of the units enforced by:

$$\sum_{k=t}^{\min(\overline{T}_g, |\mathcal{T}|)} \xi_{g,k} \geq \overline{T}_g(\xi_{g,t} - \xi_{g,t-1}), \quad \forall g \in \mathcal{G}, t \in \mathcal{T} \setminus \{1\} \tag{6}$$

$$\sum_{k=t}^{\min(\underline{T}_g, |\mathcal{T}|)} (1 - \xi_{g,k}) \geq \underline{T}_g(\xi_{g,t-1} - \xi_{g,t}), \quad \forall g \in \mathcal{G}, t \in \mathcal{T} \setminus \{1\}. \tag{7}$$

Finally, ramping constraints that limit the rate at which power output can increase (ramp-up) or decrease (ramp-down) between consecutive time periods are considered and provided by.

$$p_{g,t} - p_{g,t-1} \leq \overline{r}_g, \quad \forall g \in \mathcal{G}, t \in \mathcal{T} \setminus \{1\} \tag{8}$$

$$p_{g,t-1} - p_{g,t} \leq \underline{r}_g, \quad \forall g \in \mathcal{G}, t \in \mathcal{T} \setminus \{1\}. \tag{9}$$

The variable $\xi_{g,t}$ is a binary variable indicating whether unit g is ON at time t. The MICQP formulation of UC-ED is:

$$\begin{aligned} \min_{x, S^{\text{gen}}_{g,t}, \xi_{g,t}} \quad & x \\ \text{subject to } & (1)-(9), \\ & \xi_{g,t} \in \{0,1\} \quad \forall g \in \mathcal{G} \quad \forall t \in \mathcal{T}. \end{aligned} \tag{10}$$

Traditionally the UC-ED and the optimal power flow problems arme solved sequentially due to their computational complexity. However, this sequential approach may not yield the global optimum due to the interdependencies among these problems. Therefore, integrating power flow constraints into a unified problem has been a topic of interest in recent years [11].

The UC-PF formulation is designed to optimize the scheduling and dispatch of both real and reactive power, while taking into account the physical network constraints directly in the UC. The UC-PF problem can be formulated by adding the following constraints. Let $V_i \in \mathbb{C}$ be the voltage phasor of node i and $W_{ij} \in \mathbb{C}$ be the product of node i's voltage and the complex conjugate of node j's.

The following non-convex constraint ensures that the voltage product W_{ij} represents the above definition, aligning with the power flow equations in the network:

$$W_{ij,t} = V_{i,t} V^*_{j,t}, \quad \forall i, j \in \mathcal{N}, \forall t \in \mathcal{T}. \tag{11}$$

The voltage magnitude at each node must be kept in an acceptable interval:

$$(\underline{v}_i)^2 \leq W_{ii,t} \leq (\overline{v}_i)^2, \quad \forall i \in \mathcal{N}, \forall t \in \mathcal{T}. \tag{12}$$

In UC-PF, the constraints (2)–(3) are removed and the power balance at each node is ensured by:

$$S^{\text{gen}}_{i,t} - S^{\text{d}}_{i,t} = \sum_{ij \in \mathcal{L}} S_{ij,t}, \quad \forall i \in \mathcal{N}, \forall t \in \mathcal{T}, \tag{13}$$

where $S^{\text{gen}}_{i,t}$ and $S^{\text{d}}_{i,t}$ are the total power generation and demand at node i, respectively. Based on (11), the power flowing in each line is modelled by:

$$S_{ij} = (W_{ii,t} - W_{ij,t}) Y^*_{ij,t}, \quad ij \in \mathcal{L}, \tag{14}$$

where Y_{ij} is the admittance of line ij. Line ij's power flow is limited by its capacity expressed as $\overline{S}_{ij,t}$, yielding the constraint:

$$|S_{ij,t}|^2 \leq \left(\overline{S}_{ij}\right)^2, \quad \forall ij \in \mathcal{L}, \forall t \in \mathcal{T}. \tag{15}$$

The phase difference between two adjacent nodes must satisfy the following constraint to be within the physical limits imposed by the transmission line:

$$\tan\left(-\theta^{\Delta}_{ij}\right) \text{Re}(W_{ij,t}) \leq \text{Im}(W_{ij,t}) \leq \tan\left(\theta^{\Delta}_{ij}\right) \text{Re}(W_{ij,t}), \quad \forall ij \in \mathcal{L}, \forall t \in \mathcal{T}. \tag{16}$$

Lastly, we use the SOC relaxation of [9] to obtain a mixed-integer convex UC-PF formulation and impose:

$$|W_{ij,t}|^2 \leq W_{ii,t} W_{jj,t} \ \forall ij \in \mathcal{L}, \forall t \in \mathcal{T}, \tag{17}$$

instead of (11).

Altogether, we obtain the MISOCP formulation of the UC-PF:

$$\begin{aligned} \min_{x, S^{\text{gen}}_{g,t}, \xi_{g,t}} \quad & x \\ \text{subject to } & (1), (4)-(9), (12)-(17) \\ & \xi_{g,t} \in \{0,1\} \quad \forall g \in \mathcal{G} \quad \forall t \in \mathcal{T}. \end{aligned} \tag{18}$$

3 Methodology

We now present our machine learning-accelerated methodology for MISOCP. We then specialized our approach to MICQP. The non-linearity of MISOCP and MICQP introduces complexities which require us to extend and modify [16]'s approach for MILP. We introduce additional edge features to account for the non-linear components in the problem formulations. Through these adaptations, we successfully apply a GCNN-based approach to solve MISOCPs and MICQPs for UC with and without power flow constraints, respectively, in reduced time. In what follows, we assume that all constraints have been rewritten as $g_k(z) \leq 0$, where g_k represents the k^{th} constraint function and z collects all optimization variables. Our methodology is summarized in Figure **??**.

3.1 K-partite graph representation

Given the complex, relational nature of MIPs, we represent them using a graph. In such a representation, variables and constraints can be modelled as nodes in a graph, with edges connecting variables and constraints, creating a k-partite graph. The advantage of this representation is that it captures the inherent structure of the MIP, while being flexible and scalable to problems of different sizes and complexity.

3.2 MISOCP

A tripartite graph, as the name suggests, involves three disjoint sets of nodes with edges connecting nodes from different sets. The benefit of utilizing a tripartite graph lies in its capability to handle the more complex structure of SOC constraints. We note that in this work, we express all SOC constraints in their equivalent hyperbolic form.

For the graph representation of MISOCPs, the three sets of nodes in our tripartite graph are defined as: (1) variables, (2) bilinear terms representing the multiplicative relationships between variables, and (3) the constraints. The edges in this graph indicate the relationships between these three node sets. An edge is drawn between two variables and a bilinear node if and only if the variables are multiplied together in the SOC constraint. The degree of the variable is indicated on this edge. Simultaneously, an edge is drawn between a bilinear node and a constraint node if the latter belongs to that particular constraint. The weight of the variable pair involved in the multiplication operation is encoded on this edge. Figure 1 depicts a tripartite graph for constraints (1) and (17).

We remark that variables not involved in multiplicative operations are linked alone to a bilinear node. This approach aids to preserve the simplicity of the graph embedding while accurately reflecting the structure of the MISOCPs.

3.3 MICQP

A simplification of this model can be made when representing MICQPs, reducing the graph to a bipartite graph.

In a bipartite graph, nodes are divided into two disjoint sets, and edges can only connect nodes from different sets. To represent MICQPs, we define the two sets of nodes as variables and constraints, respectively. An edge is drawn between a variable node and a constraint node if and only if the variable appears in the constraint. In this way, the graph structure captures the incidence relation between variables and constraints of the MICQP. The order of a given variable is given as a feature for the same edge as its weights in this representation. Figure 2 provides an example of the bipartite graph for constraints (1) evaluated at $t = |\mathcal{T}|$ and (6) at $t = |\mathcal{T}| - 1$.

These graphical representations not only captures the structure of the optimization problem but also provide a rich source of information that a GCNN can leverage to learn effective branch and bound variable selection policies.

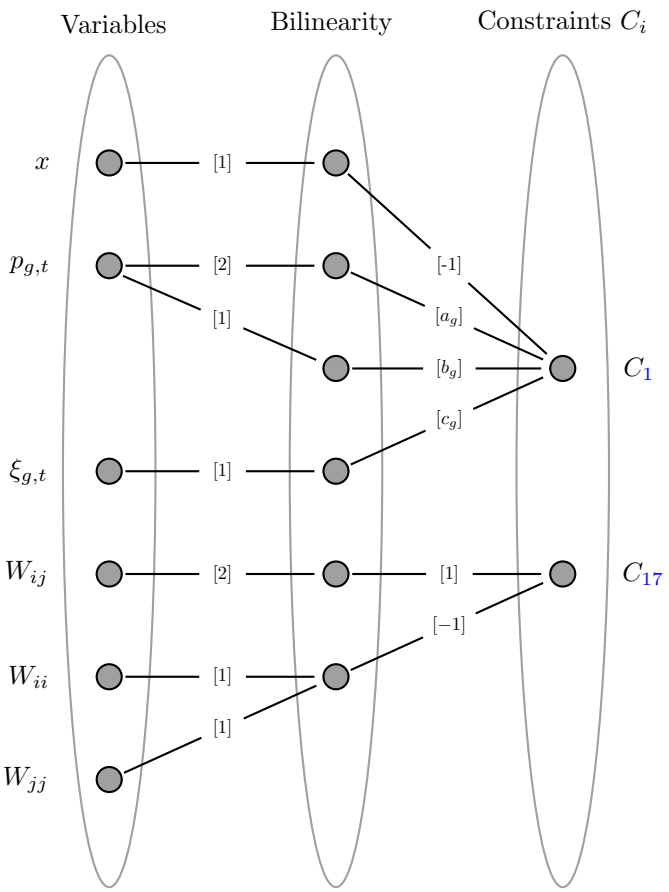

Figure 1: Tripartite graph representation of constraints (1) and (17)

3.4 Graph Convolutional Neural Networks

GCNNs are a variant of CNNs designed to process graph-structured data. By applying transformation functions to the local neighbourhoods of nodes, GCNNs are capable of focusing on local graph structures and of propagating information across the graph, enabling the effective handling of irregular and dynamic structures frequent in graph data [34].

GCNNs have been successfully used for various tasks on graph-structured data, such as node classification, link prediction, and graph classification. GCNNs have also demonstrated significant promise in addressing MILPs effectively. In the context of our work, we utilize GCNNs to process the tripartite graph representations of MISOCPs or the bipartite graph representations of MICQPs and learn effective variable selection policies for the branch and bound algorithm.

The central component of our methodology is the GCNN, which is trained to process the k-partite graph representation of each MIP. The GCNN functions by utilizing the structured data in the form of the k-partite graph and extracts valuable information that is inherent in the graph topology.

In the context of the tripartite graph representation, the GCNN use three distinct convolution passes: from variables to bilinear terms, from the bilinear terms to constraints, and finally from constraints back to bilinear terms. This differs from the bipartite representation for which the GCNN applies two distinct convolution passes: one moving from variables to constraints, and another from constraints to variables.

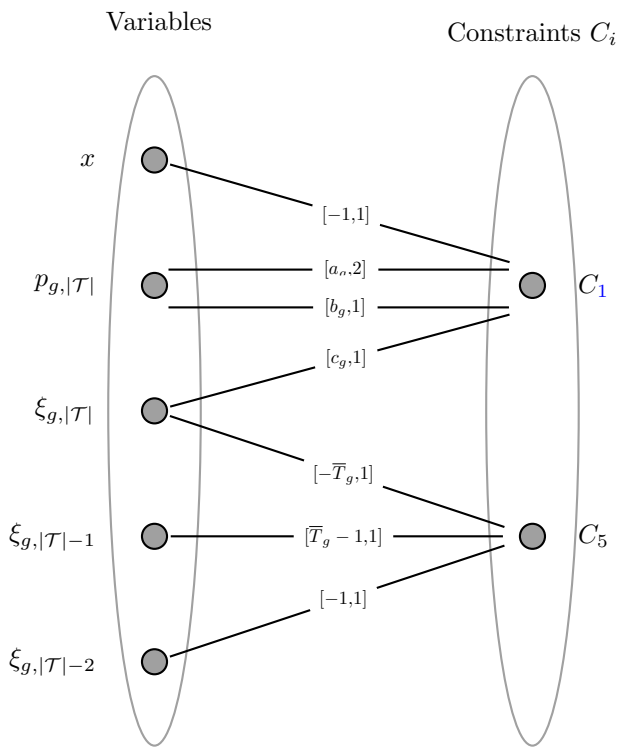

Figure 2: Bipartite graph representation of constraints (1) for $t = |\mathcal{T}|$ and (6) for $t = |\mathcal{T}| - 1$

3.5 Expert strategy

A multitude of branching strategies have been proposed, each possessing its unique set of attributes and potential drawbacks [23].

Strong branching excels in the exploration of fewer nodes, albeit at a high computational cost per node, making it an ideal candidate for imitation learning [5]. To circumvent its time-intensive nature, we employ a GCNN to speed up node evaluations, effectively offsetting the primary drawback of strong branching. However, the property of exploring fewer nodes, while nearly universal in linear problems, doesn't always hold true for non-linear ones as suggested by [5]'s results. Thus, we found it necessary to empirically validate the performance of strong branching in the context of UC instances to ensure its suitability for our approach.

In the case of UC, Tables 1 and 2 underscore the utility of full-strong branching as a robust branching strategy. As such, we chose full-strong branching as the primary algorithm to imitate in our approach, yielding considerable performance gains. Yet, it is essential to bear in mind the need for problem-specific empirical evaluations. When navigating non-linear problems, varying problem characteristics might warrant the adoption of alternative algorithms for achieving optimal performance.

3.6 Strong branching

We now turn our focus towards the strong branching method, highlighting its implementation and subsequent benefits and challenges within our framework. Proposed by [3], strong branching computes, at each node in the branch and bound tree, the optimal solution of the convex relaxation for all possible branchings on a set of candidate variables, and then selects the variable that yield the greatest

Table 1: Comparison of branching strategies for the MISOCP UC-PF (RTS-96 test case)

Branching Strategy	Average Time (s)	Average Node Count
relpscost	47.06	45094.1
vanillafullstrong	377.4	11179.2
inference	75.5	69109.1
pscost	44.7	30644.6
mostinf	223.8	186968.3
leastinf	37.4	25768.0
allfullstrong	**642.5**	**4660.3**
cloud	89.1	7256.8

Table 2: Comparison of branching strategies for the MICQP UD-ED (RTS-96 test case)

Branching Strategy	Average Time (s)	Average Node Count
relpscost	2.57	2477.8
vanillafullstrong	17.29	567.3
inference	2.23	2936.3
pscost	1.48	1695.4
mostinf	10.73	9990.2
leastinf	1.36	1154.6
allfullstrong	**26.63**	**235.9**
cloud	5.39	358.2

improvement in the dual bound. This implies that strong branching has the ability to perfectly predict the immediate impact of branching decisions, allowing for more efficient exploration of the search space.

Despite its apparent advantage, strong branching is computationally expensive, especially when the number of candidate variables is large like in the UC problem. It necessitates solving several convex relaxation of the problem at each node in the tree, each corresponding to a potential branching decision. The algorithm is particularly time-consuming when full-strong branching is used, i.e., when the relaxation is evaluated for every single variable.

3.7 Data collection strategy

We utilize the data collection mechanism introduced by [16]. Relying solely on strong branching in the data collection can be the source of various problems. Specifically, using only strong branching could result in a lack of variety in the explored states and can, as a result, introduce correlation between data points. This potential bias in the learning process would contravene the assumption that observations should be independent and identically distributed (i.i.d.) [8]. By diversifying the data collection process, we aim to satisfy this assumption and thereby improve the reliability of our results.

We use a strategy combining the advantages of strong branching with another branching technique, namely pseudocost branching. The process starts by exploring the decision tree using this secondary branching technique, which typically leads to less optimal but more diverse branching decisions. This exploration phase enriches the dataset by capturing a wide range of problem states, thus mitigating the risk of overfitting to a specific pattern of strong branching decisions.

Periodically, the strategy transitions randomly to strong branching. At this stage, the state of the branch and bound tree and the chosen variable for branching are recorded for each node where strong branching is applied. This information then serves as the training data for the machine learning model presented next. The exploration phase generates data that represents a wide range of possible problem states, enhancing the robustness of the model.

3.8 Imitation learning

Imitation Learning (IL) is a type of machine learning where an agent learns to perform tasks by mimicking expert behaviour [20]. This learning method is particularly useful in situations where the optimal behaviour is difficult to define explicitly, but examples of good behaviour are readily available.

In this work, the expert is the strong branching strategy. Our aim is to develop a machine learning model that can make branching decisions as effectively as strong branching, but with reduced computational time. The application of IL for MISOCP and MICQP is as follows.

3.9 Data collection

We first generate a dataset of branching decisions using the data generation strategy as described in Section 3.7. This dataset collects the states of the branch and bound tree at various nodes (features), and the branching decisions made by the full strong branching strategy at these nodes (labels).

3.10 Model training

Next, we train a GCNN on this dataset. The GCNN architecture is well suited to this task because it can effectively capture the complex, graph-structured nature of the MISOCP and MICQP UC.

3.11 Policy derivation

After training, the GCNN produces a probability distribution over the candidate branching variables for each node in the tree. We interpret this distribution as the GCNN's policy for branching decisions, with the variable with the highest probability being selected for branching.

3.12 Policy evaluation

We evaluate the effectiveness of our learned policy by comparing its performance to the standard `SCIP` solver on a separate test set of UC problem instances. We assess performances in terms of the computational time.

By employing IL, we aim to devise a method that can generalize the expert behaviour of strong branching to a wide range of non-linear problem instances, thus reducing the computational burden of the UC problem and enabling more efficient operations in power systems while remaining exact by using the method illustrated in **??**.

4 Results

We now present the numerical experiments conducted to validate our machine learning-based MISOCP and MICQP approaches for the UC problem.

4.1 Computational tools

In this research, the computational tools we used are central to our methodology and results. We leveraged `SCIP` [1] and `Ecole` [26], two prominent software in the field of combinatorial optimization and machine learning.

`SCIP` is an advanced solver for a range of optimization problems including mixed-integer non-linear programming. `SCIP` is open-source and well known for its flexibility and efficiency. This allows us to adapt it to our problem context with relative ease. It provides a robust base for our exploration of efficient solutions to the UC problem, and is the backbone of our computations.

`Ecole`, short for Extensible Combinatorial Optimization Learning Environments, provides an interface between `SCIP` and our machine learning model, acting as a bridge that connects the combinatorial optimization solver with the learning environment. `Ecole` transforms combinatorial optimization problems into Markov decision processes, enabling the integration of `SCIP` with powerful Python-based machine learning tools. This transformation, while abstract, allows us to employ advanced machine learning techniques to devise and refine heuristics for branch and bound variable selection tailored to the UC problem.

The experimental evaluations in this research were conducted on two computing platforms: the infrastructure provided by The Digital Research Alliance of Canada, and the server resources at GERAD.

The base instances for our numerical experiments were sourced from the MATPOWER case library [36] except for RTS-96 which is from [37].

4.2 Numerical experiments

The results of our research illustrate the effectiveness and robustness of our machine learning-based approach in solving the unit commitment problem across a range of power grid models. We refer to our approach as GCNN & `SCIP` hereinafter. We tested our methods on a variety of networks and presents results averaged over 100 evaluations. Detailed results of these experiments based on computation speed are summarized in Tables 3 and 4 and illustrated in Figures 3 and 4.

Table 3: Performance of the MISOCP GCNN & `SCIP` method for UC-PF

Grid	Number of Generators/Lines	`SCIP` Average Time (s)	GCNN & `SCIP` Average Time (s)	Gain
RTS-96	99/73	659.5	593.4	10.0%
IEEE-118	54/118	1118.2	939.2	16.0 %
CASE300	69/300	1678.1	1406.2	16.2 %

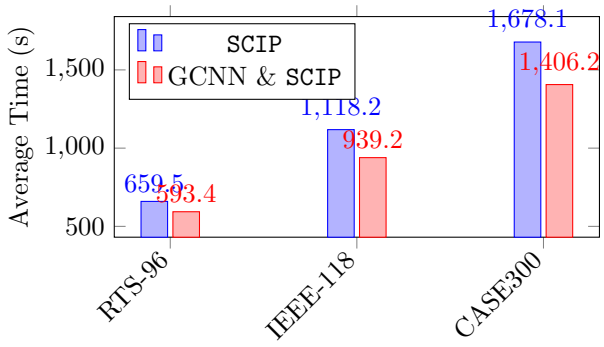

Figure 3: Bar chart representation of the performance of the MISOCP method for UC-PF

Table 4: Performance of the MICQP GCNN & `SCIP` method for UC-ED

Grid	Number of Generators	`SCIP` Average Time (s)	GCNN & `SCIP` Average Time (s)	Gain
CASE1888RTE	298	916.4	858.4	6.3%
CASE2848RTE	547	1375.8	1189.6	13.5%
CASE6470RTE	1330	3427.7	2465.8	28.1%
CASE6515RTE	1389	3007.8	2192.0	27.1%

Following the tabular data presented above, we include two graphical representations presented in Figures 5 and 6. These graphs capture the computational efficiency of each algorithm – `Gurobi` [18],

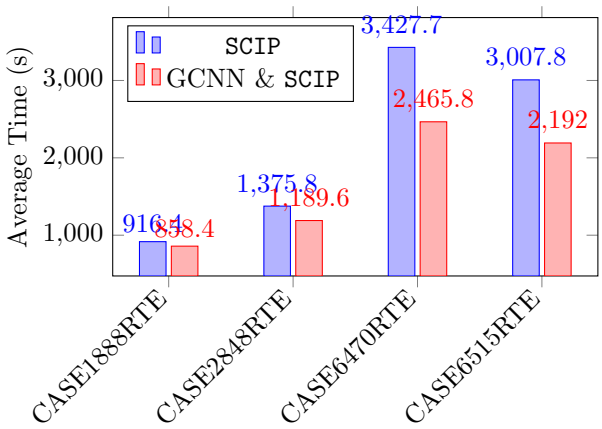

Figure 4: Bar chart representation of the performance of the MICQP method for UC-ED

`SCIP`, and our proposed GCNN & `SCIP` – over a collection of 100 UC instances generated randomly by adding normal random noise to load profiles.

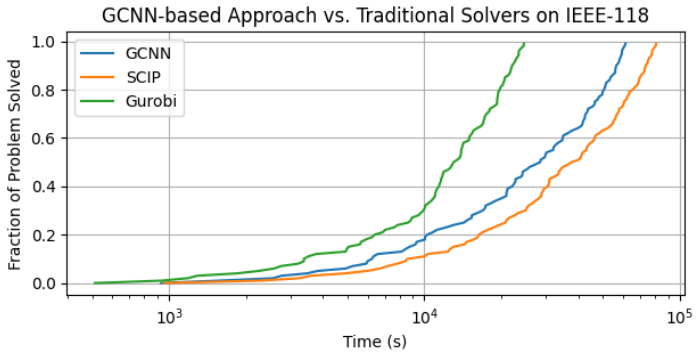

Figure 5: Computational Efficiency for the IEEE-118 test case on 100 MISOCP UC problem instances

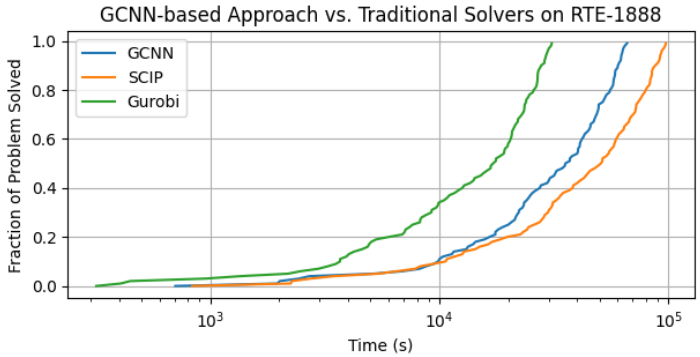

Figure 6: Computational Efficiency for the CASE1888RTE test case on 100 MICQP UC problem instances

Figures 5 and 6 underscore the potential of our GCNN-based approach in efficiently solving UC problems in various grid models, demonstrating how it is possible to use our method to improve solvers like `SCIP`.

We note that in all cases, `Gurobi`, a proprietary solver, outperformed our method. Given that `Gurobi` is known to be significantly faster than `SCIP`, which our approach builds upon due to its open-source nature, this was not unexpected. Despite this, our performance relative to `Gurobi` is encouraging and leads us to conjecture that improvements to `Gurobi` may be achieved if it were to implement a methodology similar to ours. By embedding contextual information to the MIP solver `Gurobi` may significantly enhance the efficiency and speed of solving complex optimization problems in power engineering, such as unit commitment, thereby optimizing power generation and reducing operational costs.

Next, we evaluate the robustness of our approach. We examine its effectiveness on grids with both additions and removal of lines. Specifically, we modify the grid by adjusting up to 10% of the lines either by moving their location or omitting them entirely. However, to ensure feasibility of the problem, if removing a line yield an infeasible problem, the line is re-added, leading to total modifications ranging between 0 and 10%. The detailed results of these experiments are summarized in Table 5 and Figure 7.

Table 5: Robustness of the GCNN & `SCIP` method under 10% line modifications

Grid	Modification Lines	`SCIP` Avg Time (s)	GCNN & `SCIP` Avg Time (s)	Gain
RTS-96	Moved	1014.1	895.6	11.7%
RTS-96	Cut	912.2	795.4	12.8%
IEEE-118	Moved	1091.1	1027.5	5.8%
IEEE-118	Cut	1042.3	937.0	10.1%
CASE300	Moved	1621.1	1396.5	13.9%
CASE300	Cut	1831.2	1658.1	9.4%

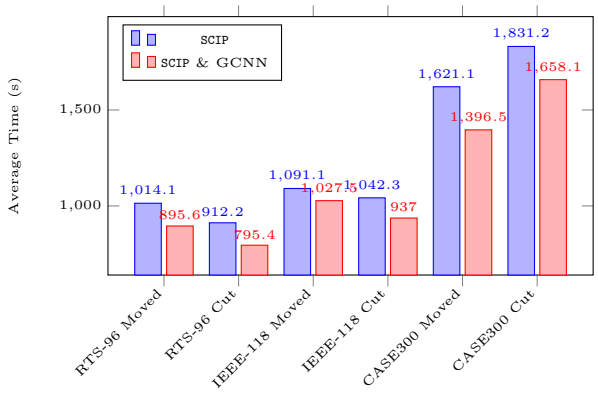

Figure 7: Bars chart representation of the performance of the MISOCP method for UC-PF after grid modification

We remark that the GCNN we employed is trained solely on the original, unmodified versions of the grid models, i.e., the original constraints set. The rationale behind these experiments was to simulate a realistic scenario where our model is confronted to unforeseen variations in the grid structure, akin to the dynamic and unpredictable conditions that often characterize real-world power systems.

Our work consistently outperformed the traditional solver `SCIP` under these varied scenarios, thus establishing robustness to alterations to the grid topology. These results illustrate the generalizability of our method.

5 Conclusion

Our work presents advancements in machine learning for power system optimization, specifically in solving the UC problem. Our method extends on the techniques of [16], demonstrating effective application to mixed-integer non-linear programs, namely MISOCPs and MICQPs. Our novel representation

incorporates additional edge features into the graph, capturing non-linear characteristics commonly overlooked in current methodologies. By employing a k-partite graph representation and a GCNN, we are able to derive significant insights from these graphs, which in turn facilitates the learning of effective variable selection policies for the branch and bound algorithm, consequently expediting the overall optimization process. We apply our method to the unit commitment problem, highlighting its practical adaptability to complex, real-world problems in power engineering. Our numerical analysis, performed on seven standard datasets, confirms the robustness and efficiency of our approach. In various scenarios, our method consistently outperforms the traditional `SCIP` solver, showing potential for significant advancements in power system optimization.

Potential future research avenues could revolve around the innovative use of GCNNs and k-partite graphs to learn other types of heuristics, expanding their current applicability. Additionally, the introduction of dynamic features, particularly temporal ones, could endow the model with a broader context, thereby enhancing its predictive accuracy and efficacy. Moreover, exposing the model to more diverse datasets during the training phase could foster a more robust and versatile model, honing its capability to generalize across various scenarios and problem types. The exploration of these strategies has the potential to contribute substantially to the MIP in power systems and beyond. By doing so, we could foster the development of robust, flexible MIP techniques that remain efficient even as the problem topology and parameters change over time.

References

[1] Tobias Achterberg. Scip: solving constraint integer programs. Mathematical Programming Computation, 1(1):1–41, 2009.

[2] Alejandro Marcos Alvarez, Quentin Louveaux, and Louis Wehenkel. A machine learning-based approximation of strong branching. INFORMS Journal on Computing, 29(1):185–195, 2017.

[3] D. Applegate, R. Bixby, V. Chvatal, and B. Cook. Finding cuts in the TSP (a preliminary report). Technical report, 1995.

[4] Maria-Florina Balcan, Travis Dick, Tuomas Sandholm, and Ellen Vitercik. Learning to branch. In International conference on machine learning, pages 344–353. PMLR, 2018.

[5] Pietro Belotti, Jon Lee, Leo Liberti, François Margot, and Andreas Wächter. Branching and bounds tighteningtechniques for non-convex MINLP. Optimization Methods and Software, 24(4-5):597–634, 2009.

[6] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. European Journal of Operational Research, 290(2):405–421, 2021.

[7] Anya Castillo, Carl Laird, César A. Silva-Monroy, Jean-Paul Watson, and Richard P. O'Neill. The unit commitment problem with ac optimal power flow constraints. IEEE Transactions on Power Systems, 31(6):4853–4866, 2016.

[8] Aaron Clauset. A brief primer on probability distributions. In Santa Fe Institute, 2011.

[9] Carleton Coffrin, Hassan L. Hijazi, and Pascal Van Hentenryck. The QC relaxation: A theoretical and computational study on optimal power flow. IEEE Transactions on Power Systems, 31(4):3008–3018, 2016.

[10] Gonzalo E. Constante-Flores and Antonio J. Conejo. Security-constrained unit commitment: A decomposition approach embodying Kron reduction. European Journal of Operational Research, 2023.

[11] Gonzalo E. Constante-Flores, Antonio J. Conejo, and Feng Qiu. AC network-constrained unit commitment via relaxation and decomposition. IEEE Transactions on Power Systems, 37(3):2187–2196, 2022.

[12] Patrick de Mars and Aidan O'Sullivan. Applying reinforcement learning and tree search to the unit commitment problem. Applied Energy, 302:117519, 2021.

[13] Patrick de Mars and Aidan O'Sullivan. Reinforcement learning and A* search for the unit commitment problem. Energy and AI, 9:100179, 2022.

[14] Giovanni Di Liberto, Serdar Kadioglu, Kevin Leo, and Yuri Malitsky. Dash: Dynamic approach for switching heuristics. European Journal of Operational Research, 248(3):943–953, 2016.

[15] Marc Etheve, Zacharie Alès, Côme Bissuel, Olivier Juan, and Safia Kedad-Sidhoum. Reinforcement learning for variable selection in a branch and bound algorithm. In International Conference on Integration

of Constraint Programming, Artificial Intelligence, and Operations Research, pages 176–185. Springer, 2020.

[16] Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. Exact combinatorial optimization with graph convolutional neural networks. Advances in Neural Information Processing Systems, 32, 2019.

[17] Prateek Gupta, Maxime Gasse, Elias Khalil, Pawan Mudigonda, Andrea Lodi, and Yoshua Bengio. Hybrid models for learning to branch. Advances in neural information processing systems, 33:18087–18097, 2020.

[18] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023.

[19] Christoph Hansknecht, Imke Joormann, and Sebastian Stiller. Cuts, primal heuristics, and learning to branch for the time-dependent traveling salesman problem. arXiv preprint arXiv:1805.01415, 2018.

[20] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. ACM Computing Surveys (CSUR), 50(2):1–35, 2017.

[21] Elias Khalil, Pierre Le Bodic, Le Song, George Nemhauser, and Bistra Dilkina. Learning to branch in mixed integer programming. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 30, 2016.

[22] Elias B Khalil, Bistra Dilkina, George L Nemhauser, Shabbir Ahmed, and Yufen Shao. Learning to run heuristics in tree search. In IJCAI, pages 659–666, 2017.

[23] David R. Morrison, Sheldon H. Jacobson, Jason J. Sauppe, and Edward C. Sewell. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. Discrete Optimization, 19:79–102, 2016.

[24] Vinod Nair, Sergey Bartunov, Felix Gimeno, Ingrid von Glehn, Pawel Lichocki, Ivan Lobov, Brendan O'Donoghue, Nicolas Sonnerat, Christian Tjandraatmadja, Pengming Wang, et al. Solving mixed integer programs using neural networks. arXiv preprint arXiv:2012.13349, 2020.

[25] S. Pineda and J.M. Morales. Is learning for the unit commitment problem a low-hanging fruit? Electric Power Systems Research, 207:107851, 2022.

[26] Antoine Prouvost, Justin Dumouchelle, Lara Scavuzzo, Maxime Gasse, Didier Chételat, and Andrea Lodi. Ecole: A gym-like library for machine learning in combinatorial optimization solvers. arXiv preprint arXiv:2011.06069, 2020.

[27] Jingtao Qin, Nanpeng Yu, and Yuanqi Gao. Solving unit commitment problems with multi-step deep reinforcement learning. In 2021 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), pages 140–145, 2021.

[28] Qingyu Qu, Xijun Li, Yunfan Zhou, Jia Zeng, Mingxuan Yuan, Jie Wang, Jinhu Lv, Kexin Liu, and Kun Mao. An improved reinforcement learning algorithm for learning to branch. arXiv preprint arXiv:2201.06213, 2022.

[29] T. Sawa and K. Furukawa. Unit commitment using quadratic programming and unit decommitment. In 2012 IEEE Power and Energy Society General Meeting, pages 1–6, 2012.

[30] Haoran Sun, Wenbo Chen, Hui Li, and Le Song. Improving learning to branch via reinforcement learning. In Learning Meets Combinatorial Algorithms at NeurIPS2020, 2020.

[31] Yunhao Tang, Shipra Agrawal, and Yuri Faenza. Reinforcement learning for integer programming: Learning to cut. In Proceedings of the 37th International Conference on Machine Learning, volume 119 of Proceedings of Machine Learning Research, pages 9367–9376. PMLR, 13–18 Jul 2020.

[32] Joshua Adam Taylor. Convex Optimization of Power Systems. Cambridge University Press, 2015.

[33] Pascal Van Hentenryck. Machine learning for optimal power flows. Tutorials in Operations Research: Emerging Optimization Methods and Modeling Techniques with Applications, pages 62–82, 2021.

[34] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems, 32(1):4–24, 2020.

[35] Álinson 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, 2021.

[36] Ray Daniel Zimmerman, Carlos Edmundo Murillo-Sánchez, and Robert John Thomas. Matpower: Steady-state operations, planning, and analysis tools for power systems research and education. IEEE Transactions on power systems, 26(1):12–19, 2010.

[37] Ray Daniel Zimmerman, Carlos Edmundo Murillo-Sánchez, and Robert John Thomas. Matpower: Steady-state operations, planning, and analysis tools for power systems research and education. IEEE Transactions on Power Systems, 26(1):12–19, 2010.