Titre: Title:	Learning-Accelerated Exact Mixed-Integer Second-Order Cone Programming for Unit Commitment
Auteur: Author:	Philippe Maisonneuve
Date:	2023
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Maisonneuve, P. (2023). Learning-Accelerated Exact Mixed-Integer Second-Order Cone Programming for Unit Commitment [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/55101/

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Learning-accelerated Exact Mixed-integer Second-order Cone Programming for Unit Commitment

PHILIPPE MAISONNEUVE
Département de génie électrique

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise* ès *sciences appliquées*
Génie énergétique

Août 2023

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Learning-accelerated Exact Mixed-integer Second-order Cone Programming for Unit Commitment

présenté par **Philippe MAISONNEUVE**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Keyhan SHESHYEKANI, président
Antoine LESAGE-LANDRY, membre et directeur de recherche
Quentin CAPPART, membre

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to those who supported me during the demanding but rewarding journey of my Master's degree.

I start by warmly thanking my research supervisor Antoine Lesage-Landry. His guidance and passion have been a beacon in the completion of this thesis. His optimism, kindness, and rational approach are qualities I admire and aspire to embody.

A huge thank you to Vincent Mai, whose mentorship has shaped my thinking and actions to become a better researcher. The generosity of IVADO in financially supporting this research also deserves special recognition.

My sincere thanks go to Jean-Luc Lupien and Marie-Christine Paré. Jean-Luc, your unwavering support helped me stay on course even during the most tumultuous times. Marie-Christine, sharing similar challenges and pains with you has been a comfort.

My colleagues, Julien Pallage, Loreley Sepho, Matthias Molénat, Abraham Kouamé N'Zi, Olivier Daoust, Olivier Bélanger, Victor Darleguy, Samuel Mendoza, Bouh Abdillahi, and Yu-Hsin Wu, have all contributed to making our work environment vibrant and stimulating. A special shout-out to Julien for his support during the toughest times and to Yu-Hsin for his infectious good humor, despite the mysterious disappearance of his snacks, for which I know he won't hold a grudge.

I owe a lot to my family – my grandparents and my mother, who, despite their unfamiliarity with my field of study, have always supported me with love and encouragement in my quest for knowledge.

Lastly, my friends Dylan Barat, Frederik Nolte, Jean-Sébastien Patenaude, and Milen Bozhilov deserve special thanks. You have been a family to me during times when my own could not be present.

In short, every person mentioned here has made a unique and valuable contribution to my journey. I cherish the opportunity to have been surrounded by such extraordinary and caring individuals. I am humbly grateful and proud of what we have accomplished together.

RÉSUMÉ

Dans le domaine des réseaux électriques, la gestion de l'engagement des unités – mieux connu sous son appellation anglaise "unit commitment" (UC) – est une problématique d'optimisation majeure. Ces enjeux sont fondamentaux puisqu'ils engagent des décisions cruciales en matière de planification des unités de production d'électricité pour répondre efficacement à la demande pour une période spécifique. Leur complexité, due à leur nature binaire et à la non-convexité des contraintes d'écoulement de puissance, exige leur modélisation en tant que programmes coniques mixtes entiers du second ordre (MISOCPs) et programmes quadratiques convexes mixtes entiers (MICQPs).

Les méthodes d'apprentissage machines offre un socle robuste pour développer des heuristiques sur mesure, destinées à résoudre ces problématiques d'optimisation du UC. En puisant dans des motifs et des données, elles offrent des solutions spécifiquement personalisé à chaque cas d'utilisation.

Dans ce mémoire, nous proposons une méthodologie visant à augmenter l'efficacité des solveurs d'optimisation pour UC à l'aide d'un graph Covonlutional neural network (GCNN). Notre approche tire avantage de la structure inhérente de ces problèmes mixtes entiers en les représentant sous forme de graphes k-partites variables-contraintes. Cette représentation nous donne l'opportunité de saisir les interactions subtiles entre les variables et les contraintes du problème.

Grâce à l'utilisation du GCNN, nous sommes en mesure d'extraire des informations précieuses de ces graphes. Le processus d'apprentissage, réalisé par imitation, s'appuie sur la règle d'expert du "strong branching" comme référence. Cette approche permet à notre modèle d'élaborer des stratégies de sélection de variables efficaces pour l'algorithme de séparation et évaluation, accélérant de manière significative l'ensemble du processus d'optimisation. Il est notable que notre méthode assure le maintien de l'optimalité de la solution, un aspect souvent négligé dans les approches reposant systématique sur l'apprentissage de bout en bout. La préservation de l'optimalité est essentielle dans le contexte de l'UC, où la planification optimale des unités de production d'électricité peut impacter fortement l'efficacité opérationnelle et la fiabilité des systèmes électriques. Un élément clé de notre approche est l'intégration d'informations spécifiques au problème dans le processus de sélection des variables, au cœur de l'optimisation de séparation et évaluation.

Ce mémoire démontre l'efficacité et la robustesse de notre approche basée sur l'apprentissage, en adressant deux formulations différentes configuratioons d'UC sur divers modèles de réseaux

électriques. Notre méthode offre une meilleure performance que le solveur `SCIP` sur lequel elle s'appuis, et ce, dans différents scénarios. Les résultats obtenus à la suite à des analyses numériques menées sur sept modèles de réseaux standards, témoignent de l'adaptabilité et de la rapidité de notre méthode, soulignant son potentiel pour réaliser des avancées significatives dans l'optimisation des réseaux électriques.

ABSTRACT

The complexity of Unit Commitment (UC) problems in power systems is underscored by their binary attributes and the non-convex nature of power flow constraints. This results in these problems being best represented as mixed-integer second order cone programs (MISOCPs) and mixed-integer convex quadratic programs (MICQPs). The considerable optimization challenge posed by UC problems is deeply rooted in their significant role in decision-making. Specifically, they necessitate critical scheduling of power generation units to efficiently fulfil demand within a predetermined time period.

Machine learning method provide a robust framework for creating customized heuristics designed to solve optimization problems like UC, by learning from patterns and data, hence offering solutions that are tailored specifically to individual use-cases.

In this Master thesis, we present a methodology that aims to improve the efficiency of UC optimization solvers using a GCNN. Our approach harnesses the structure of these mixed-integer problems, representing them as variable-constraint k-partite graphs. This representation allows us to capture the intricate relationships between variables and constraints in the problem.

Utilizing GCNN, we extract valuable insights from these graphs. The training process is conducted via imitation learning and uses the strong branching as expert rule. This approach enables our model to learn effective variable selection policies for the branch and bound algorithm, thus accelerating the overall optimization process. Notably, our method ensures the preservation of solution optimality, a feature that is often compromised in end-to-end learning approaches. This preservation of optimality is crucial in the context of UC problems, where the optimal scheduling of power generation units can have significant implications for the operational efficiency and reliability of power systems. A key aspect of our approach is the integration of problem-specific information into the variable selection process within the branch and bound optimization.

Our work demonstrates the effectiveness and robustness of our machine learning-based approach in addressing two different formulation of the UC problem across various power grid models. Our method consistently outperformed the `SCIP` solver it is based on, even under varied scenarios. These results, obtained from numerical analysis performed on seven standard grid models, illustrate the adaptability and speed of our approach, highlighting its potential for significant advancements in power system optimization.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABREVIATIONS

GCNN	Graph Convolutional Neural Networks
CNN	Convolutional Neural Network
LNS	Large Neighbourhood Search
MICQP	Mixed-integer Convex Quadratic Programing
MILP	Mixed integer Linear Programming
MIP	Mixed integer Programming
MISOCP	Mixed-integer Second-order Cone Programming
ML	Machine Learning
RL	Reinforcement Learning
`Gurobi`	A proprietary and highly competitive solver
`SCIP`	Open source Solver for Constraint Integer Programming
UC	Unit Commitment
UC-ED	Unit Commitment with Economic Dispatch constraints
UC-PF	Unit Commitment with Power Flow constraints
CPU	Central Processing Unit
GPU	Graphical Processing Unit

CHAPTER 1 INTRODUCTION

Branch and bound algorithms have found extensive use in mixed-integer programming algorithms (MIP) problem-solving due to their broad applicability and reliable performance. However, these algorithms traditionally depend on generalized heuristics, which may not always cater to the specific requirements of each unique problem scenario. The development of specialized heuristics for every individual case has often been viewed as resource-intensive.

Machine learning offers an interesting avenue for refining these heuristics. Research suggests that integrating machine learning within branch and bound algorithms can assist in the development of heuristics that are more suited to specific problem classes or individual instances. We remark that machine learning algorithms do not directly produce the optimal solutions but rather provide a more efficient search strategy to find them. This approach can potentially enhance the speed of traditional MIP algorithms while still preserving the optimality guarantee they offered.

Our study primarily targets the Unit Commitment (UC) problem, a critical optimization task in power systems. This problem's complexity arises from balancing between various factors such as power demand, generation costs, and operational constraints. It is of utmost importance to resolve UC problems quickly and accurately, as their solutions directly influence the efficiency and reliability of power systems. By achieving optimal solutions, we can reduce operating costs while ensuring a consistent power supply. This efficiency also equips us to swiftly respond to changes in power demand or unexpected events, thereby significantly enhancing grid stability and overall system performance.

1.1 Objective of this Master Thesis

This Master Thesis explore ways to improve the computational speed of solving UC problems in power systems. By integrating machine learning with branch and bound algorithms, we aim to develop tailored heuristics that can more effectively handle each problem instance. Our approach, which leverages graph convolutional neural network (GCNN) and imitation learning, seeks to accelerate the optimization process while maintaining solution optimality.

We apply our method to the UC problem, demonstrating its practical adaptability to complex, real-world problems in power engineering. Through numerical analysis performed on seven grid models, we aim to confirm the robustness and efficiency of our approach. In various scenarios, our method aims to outperform the traditional `SCIP` solver, showing potential

for significant advancements in power system optimization.

This work hope to contribute to the development of more advanced and efficient optimization techniques in the future, potentially allowing for the use of more precise, though computationally demanding, relaxations

1.2 Structure of the Master Thesis

The remainder of this Master's thesis is organized into four additional chapters. Chapter 2 provides a comprehensive review of the relevant literature, setting the context for the subsequent discussions. In Chapter 3, we present an article submitted for publication [1] that details our use of GCNNs to enhance the performance of exact MISOCP for UC. Following, Chapter 4 offers a general discussion on the implications and the impacts of the findings presented in the article. The thesis concludes with Chapter 5, where we provide concluding remarks and explore potential avenues for future research.

CHAPTER 2 LITERATURE REVIEW

In this literature review, we examine how the integration of advanced machine learning techniques can enhance the heuristics of branch and bound algorithms. Initially, we provide an in-depth explanation of the algorithm's operation, highlighting potential areas for improvement. In addition, we explore the current methodologies used to solve UC problems, offering insight into how they might be refined through the application of machine learning. Finally, we provide a succinct overview of the key technologies underpinning these developments. The goal of this review is to present a clear and comprehensive understanding of the state-of-the-art in mixed-integer optimization problem-solving, with a specific focus on the use of machine learning within the branch and bound framework.

2.1 The branch and bound Algorithm

A central method in combinatorial optimization is the branch and bound algorithm which offers a pragmatic approach to tackling problems with an extensive scope, often incorporating thousands of variables [2]. Direct enumeration of all possible solution combinations in such instances is computationally intractable. The branch and bound algorithm addresses this challenge by using a divide-and-conquer strategy to explore groups of potential solutions, thereby drastically reducing the computational burden.

The branch and bound algorithm works by dividing the problem space into smaller and smaller subsets (branching), and then evaluating the potential solutions in these subsets for their feasibility and optimality (bounding). In doing so, it can eliminate many candidate solutions without explicitly evaluating them, leading to efficient optimization. Here, we delve deeper into the mechanics of this algorithm and how it efficiently tackles the problem of variable selection.

Figure 2.1 present an iteration principle of the branch and bound algorithm. Here, every dot represents a potential solution to the problem at hand. Initially, the entire solution space is partitioned into two distinct regions through a process termed branching. On the right, we observe the blue region, where the algorithm solves the relaxation of the problem. The outcome of this relaxation provides an insight: any solution within this blue region is at best equal to, or perhaps worse than, the relaxed solution itself. This is the bounding step during wich potential solutions are gauged for their feasibility and optimality based on the relaxed solution. Contrastingly, on the left side, we have the red region where specific feasible points are solved. If any feasible solution derived from this red region surpasses the

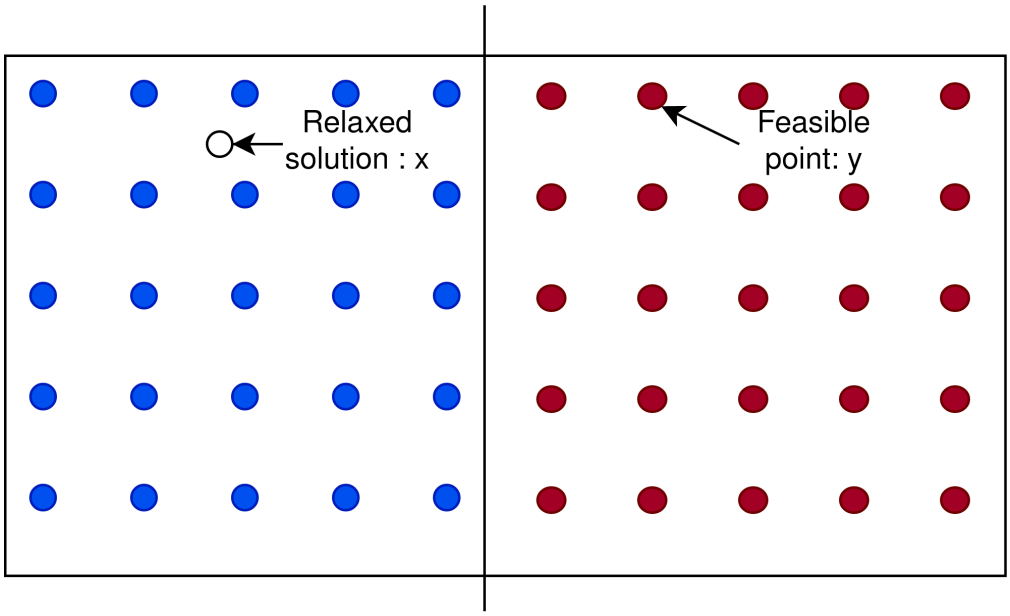

Figure 2.1 Branch and bound iteration exemple

quality of the relaxed solution from the blue region, it enables us to deduce that all solutions nested within the blue region are inherently suboptimal in comparison to those in the red region. This discernment allows the algorithm to eliminate several less promising candidate solutions, optimizing the search process.

The branch and bound process can be intricate, with each branching step potentially leading to a series of sub-problems that themselves can be branched further. Each of these sub-problems is bounded to determine its potential as a viable solution. This iterative branching and bounding continue until the algorithm either identifies the optimal solution or concludes that no such solution exists within the given constraints.

As we delve deeper into the branch and bound structure, it becomes evident that its strength lies in its ability to prune significant portions of the search space, thereby reducing the computational expense associated with seeking the optimal solution. Through a clever combination of branching strategies and bounding techniques, we can, therefore accelerate the optimization process.

The effectiveness of the branch and bound algorithm relies on the use of heuristics [3]. Heuristic strategies, which are the outcome of years of cumulative engineering effort, are honed empirically and have proven to be remarkably effective in guiding solvers towards solutions [2]. However, it is worth examining the limitations of these heuristics.

Foremost among these limitations is the probable sub-optimality of these heuristics. Their

development methodology, rooted largely in empirical observations, lacks a theoretical basis to guarantee that the resulting strategies are indeed optimal [4].

Another inherent weakness lies in their generality. Heuristics tend to take a one-size-fits-all approach, which, while aiding a broad applicability across diverse optimization problems, overlooks the distinctive structures of individual problems [5]. This absence of problem-specific customization can hinder the efficiency and the effectiveness of heuristics in specific optimization scenarios.

Machine learning emerges as a promising alternative to mitigate these limitations. It offers a novel perspective on heuristic development, which can lead to the discovery of new, potentially superior solving strategies. Furthermore, machine learning algorithms can be designed and trained to recognize the unique characteristics of individual problems, enabling the development of customized solving strategies. This tailored approach can capitalize on the inherent structure of the problems, offering a promising avenue to enhance the efficacy of solvers in optimization problems [6].

2.1.1 Type of relaxations

In the branch and bound algorithm, computing bounds necessitates periodic solutions to a continuous relaxation of the problem. When solving a mixed integer linear program (MILP), the obvious choice is a continuous linear problem relaxation, solved efficiently with the simplex algorithm.

Non-linear problems, however, pose a dilemma concerning the choice of relaxation. Choosing a linear relaxation ensures rapid resolution but offers less accurate bounds, potentially leading to more extensive and time-consuming exploration of the search space. Conversely, a relaxation that aligns more closely with the problem class, while yielding more precise bounds, can be computationally expensive and slow to evaluate.

The tradeoff between computational speed and bound accuracy leads to a crucial decision, affecting the efficiency of the overall problem-solving process. Machine learning can be instrumental in this context. By training machine learning models on various problem instances and their optimal relaxation choices, it may be possible to formulate decision rules or predictive models for relaxation selection. These machine learning guided decisions can optimize the trade-off between computation time and bound accuracy, thus boosting the efficiency of the branch and bound algorithm in non-linear problems.

In [7], a novel methodology is introduced, leveraging machine learning techniques to determine when it is advantageous to linearly relax the quadratic constraints of a problem instance.

This method represents a considerable advancement in dealing with non-linear optimization problems, particularly when deciding on the appropriate relaxation to be used in a branch and bound algorithm.

2.1.2 Primal heuristics

Primal heuristics are vital in solving MIPs, helping define the primal bound in the branch and bound algorithm. Primal bound are best feasible solution found so far in the branch and bound algorithm for mixed-integer optimization problems [2]. These techniques aim to find feasible solutions, with methods such as genetic algorithms [8], tabu search [9], and large neighborhood searches [10] among the common approaches. However, they can be computationally expensive, thus a key challenge is determining when to call these heuristics and which ones to use.

Reference [5] argues that the decision to execute primal heuristics should be contingent on a high likelihood of success. They propose a machine learning method to assess this probability, aiming to improve the efficiency and effectiveness of the heuristic call.

Building on this idea, the more recent study [11] explores the development of a learning-based schedule for primal heuristics. The approach determines both the ordering and duration for which each heuristic should be run, with the ultimate goal of generating a good primal solution as quickly as possible. The integration of machine learning in this context signifies a promising progression towards the optimization of heuristic ordering and subsequent problem-solving efficiency.

2.1.3 Cutting plane method

Alongside heuristics, cutting planes are another set of tools used to refine the search process in MIPs. These are additional constraints that can modify the result of a problem's relaxation without excluding feasible integer solutions. In the context of the branch and bound algorithm, cutting planes can be used to obtain tighter bounds on the optimal solution, which can potentially lead to the faster convergence of the algorithm. The implementation of these cuts shifts the procedure from a simple branch and bound to a more complex branch and cut algorithm [12].

Deciding which cuts to perform to optimize the algorithm involves the use of heuristics. In most solvers, this selection is determined by a linear combination of several factors, including the cut's parallelism to the objective and constraints, its efficacy, its support factors, among others [13]. Although this method is well-tuned and generally effective, it may not always be

the most efficient approach.

Machine learning, specifically neural networks, offer potential for enhancing the selection process. Reference [14] proposes the use of neural networks to predict the improvement in the objective brought about by a cut. Their study focuses on quadratic programming. Furthermore, [15] extends these ideas by implementing a RL strategy for cut selection. These studies suggest that machine learning approaches can provide valuable contributions to cut selection heuristics, potentially improving problem-solving efficiency.

2.1.4 Presolving Techniques

Presolving techniques serve a crucial role in the initial phase of solving an optimization problem. Often, valuable information regarding the solution can be inferred directly from the constraints. For instance, it may be feasible to assign fixed values to certain variables or simplify the constraints to reduce their numbers. Generally, applying presolving measures tends to accelerate the algorithms employed during the resolution phase. Reference [6] suggests employing machine learning to fine-tune the intensity of the presolving phase. However, today, no work offers a effective approach to this end.

2.1.5 Node Selection

The process of solving an MIP via the branch and bound technique requires the sequential addition of artificial constraints, thereby reducing the size of the solution's subspace. Consequently, during each algorithm iteration, a decision must be made regarding the subproblem formulation from which we continue the algorithm, with an additional constraint. This decision-making process is known as node selection. Efficient node selection is critical to the success of the algorithm. Currently, the best-estimate search [4] represents the state-of-the-art method for such decisions.

Among machine learning methods, [16] propose an imitation learning method, emulating a sub-optimal oracle derived from knowing the problem's final solution and exploring solely the nodes leading to it. Reference [17] adopts a comparable strategy, learning from the tree's final structure and presuming that the optimal exploration path is constructed by eliminating node choices leading to dead ends.

2.1.6 Variable Selection

Every iteration of the branch and bound algorithm allows for a cut to be made based on any non-fixed variable. Selecting the order of these cuts judiciously can dramatically hasten the

algorithm, and numerous methods have been proposed for this purpose. This task is more commonly referred to as branching [5], [4]. We note that some branching heuristics are more beneficial for certain classes of problems than others, and none have outperformed the others on all metrics conclusively.

A substantial part of the research on enhancing solvers through machine learning concentrates on variable selection. Numerous studies propose imitation learning approaches that mimic the behaviour of computation-heavy heuristics.

Pseudocost Branching The pseudocost branching technique provides a computationally efficient method. This approach maintains a history of the dual bound's improvement following each variable selection to estimate future choices' gains probabilistically. However, this method's limitation is that near the root of the branch and bound tree, where few decisions have been made, it becomes challenging to estimate each decision's gain due to the lack of representative data [18].

Strong Branching Strong branching is a branching method that evaluates all potential continuous relaxations following a virtual branch on each variable within a specific set [12]. The chosen branching variable is the one that induces the most substantial improvement of the dual bound. This method is undoubtedly efficient, given that it operates based on the exact impact that branching will have on the subsequent resolution step. However, it is significantly slow due to the need to solve numerous relaxed problems for each branching operation. When the relaxation is systematically evaluated for each variable, we refer to the method as full strong branching [19].

In terms of the number of nodes generated, strong branching stands as the most efficient algorithm. However, its execution speed does not reach the same level of efficiency [20]. It is worth noting that the efficacy of strong branching can be diminished when navigating the nodes of non-linear problems [20]. Although it has the potential to serve as a highly effective expert strategy in many cases, it does not guarantee universal applicability. The limitations of strong branching remind us that optimization strategies must be tailored to the nature and structure of the specific problem at hand.

Hybrid Branching Hybrid branching is the most prevalently used strategy in contemporary applications [21]. This method computes the branching variable by taking a weighted sum of the scores produced by other branching methods for each candidate variable. The variable that yields the highest aggregated score is subsequently selected as the branching

variable.

Other Branching Strategies As the field of optimization has matured, a myriad of branching methods have been developed. For instance, reliability branching [22] employs the pseudocost technique to prioritize variables, computing a partial strong branching on the subset of the most promising candidates.

Some methods, such as backdoor branching [23] and information-based branching [24], introduce innovative strategies that involve restarting the branch and bound process multiple times. This approach leverages the information acquired during previous iterations to enhance the efficiency of subsequent ones.

Other notable branching methods include inference branching, Non-chimerical Branching [25], active constraint branching [26], and cloud branching [27]. The specifics of these methods, their implementations, and their respective strengths and weaknesses can be found within their respective citations. Each branching strategy has unique characteristics, making them suitable for distinct classes of problems and diverse computational environments.

2.1.7 Large Neighbourhood Search

The branch and bound algorithm, a pivotal technique in combinatorial optimization, can be substantially improved through the application of a large neigbourhood search (LNS) algorithm [28]. By deliberately fixing a subset of variables in the problem, we create a reduced sub-problem that is computationally easier to solve, with the benefit that the solution to this sub-problem is guaranteed to be close to the solution of the original problem.

This concept of problem simplification through variable fixing is very powerful. It is akin to zeroing in on a specific area on a map to find a location, rather than aimlessly wandering the entire terrain. By iteratively fixing a subset of variables and optimizing the rest, we create a pathway of progressively refined solutions, where each solution is either an improvement or maintains the status quo.

Despite the demonstrated efficacy of LNS, it is not a universally implemented strategy in all optimization solvers. The selection of which variables to fix i.e., the neighbourhood to explore, can profoundly impact the success and efficiency of LNS. This selection process is generally heuristic-based, but recent research has highlighted the potential of machine learning approaches to enhance it.

For example, both [29] and [30] suggests the use of RL strategies for neighbourhood selection in LNS. These approaches use RL to dynamically learn the most promising neighbourhoods

to explore, based on feedback from the search process. RL provides a method of learning from past successes and failures, which can significantly enhance the efficiency of the neighborhood selection process.

Reference [31] extend this concept by introducing a neighbourhood repair method based on a neural network. This approach can work in tandem with the RL-based selection strategies, providing a method to adjust or repair selected neighbourhoods, improving their quality, and subsequently the efficiency of the search process. It is akin to using machine learning to fine-tune the precision of a spotlight in our search strategy.

2.1.8 Neural Diving

Neural Diving [32], a recent innovation in primal heuristics, takes advantage of deep learning techniques to effectively navigate the combinatorial optimization landscape. Primal heuristics, aimed at rapidly finding good (though not necessarily optimal) solutions, have been identified as a crucial aspect of successful MIP solvers. The ability to find good solutions quickly can significantly reduce the time it takes for a solver to prove optimality, as these solutions provide a strong upper bound on the problem's objective function.

In Neural Diving, a trained deep neural network is used to generate multiple partial assignments of the integer variables in the given MIP. By doing so, it essentially breaks down the larger, more complex MIP into a series of smaller, more manageable sub-MIPs. These smaller sub-problems can then be efficiently solved using an off-the-shelf MIP solver (e.g., `SCIP`), effectively completing the assignments. Notably, these sub-MIPs can be solved in parallel if computational resources permit it, which can lead to a substantial decrease in overall solving time.

Training the neural network is a critical aspect of this method. The model is trained to prioritize feasible assignments that offer better objective values. To do this, it uses training data collected offline with a standard MIP solver. Unlike other methods, Neural Diving benefits from learning from all available feasible assignments, not only the optimal ones. This is a key advantage as collecting optimal assignments can be computationally expensive and may not always be feasible.

2.2 Machine Learning method for Variable Selection

Variable selection is indeed one of the key areas where machine learning, especially imitation learning, has been successfully applied to improve the efficiency of combinatorial optimization solvers. This makes sense given the nature of the variable selection problem, which essentially

revolves around making a choice among a set of alternatives based on an evaluation of their respective merits.

As mentioned, strong branching is often chosen as the expert algorithm in these scenarios, for it is known for its ability to effectively select variables for branching, despite being computationally intensive. Imitation learning models, on the other hand, are known for their computational speed and efficiency, making them an excellent complement to strong branching.

Imitation learning is a form of supervised learning, where the model learns by imitating an expert's decisions. The goal is to replicate the expert's performance while using significantly less computational resources. However, the performance of imitation learning models is intrinsically limited by the accuracy of the expert algorithm they are trying to imitate.

RL has also been proposed as an approach to improve variable selection, offering the potential for the model to learn more complex strategies beyond what can be achieved through mere imitation [33].

In the following section, we delve deeper into the methodology proposed by [4]. We explore its evolution, the different techniques that have been employed to increase its efficiency, and the various ways in which it has been applied to combinatorial optimization.

2.3 Graph Convolutional Neural Network

A GCNN is an extension of the traditional convolutional neural network (CNN) which adapts its operations to handle graph data. Contrary to the CNN that operates on regular matrix, a GCNN processes data structured as nodes and edges.

In a given graph, nodes have some initial features or characteristics. Consider a graph with a central node, two immediate neighbours, and their respective second-level neighbours. The relationships among these nodes are depicted below in Figure 2.2. During processing, the central node gathers information from its immediate neighbours. This flow is illustrated by arrows pointing from neighboring nodes towards the central node in Figure 2.3. The central node's feature representation becomes influenced by its direct neighbours. This flow ensures the network captures the structure and attributes of the neighbourhood. More advanced operations permit the intermediate node to also aggregate information from second-level neighbours. This information is obtained indirectly by the central node through its immediate neighbours. This richer aggregation provides the central node with a broader contextual understanding within the graph. As shown in Figure 2.4, a final feature aggregation occurs within the central node, encompassing all connected nodes characteristics.

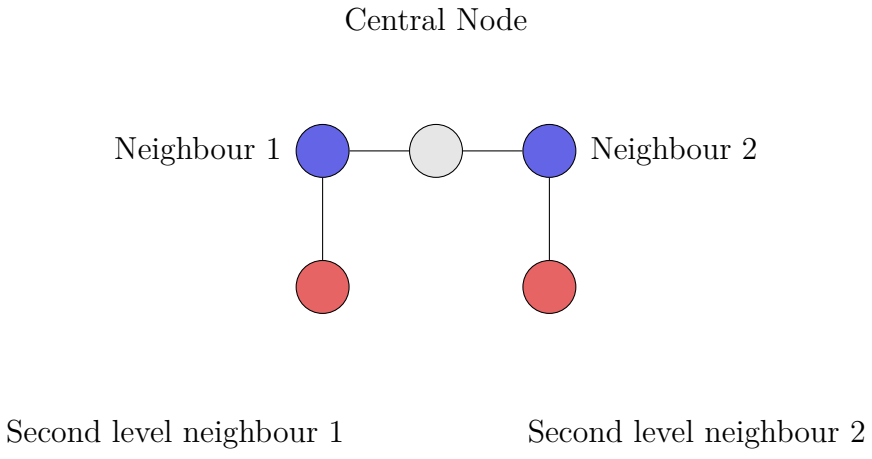

Figure 2.2 Initial configuration of nodes in a graph.

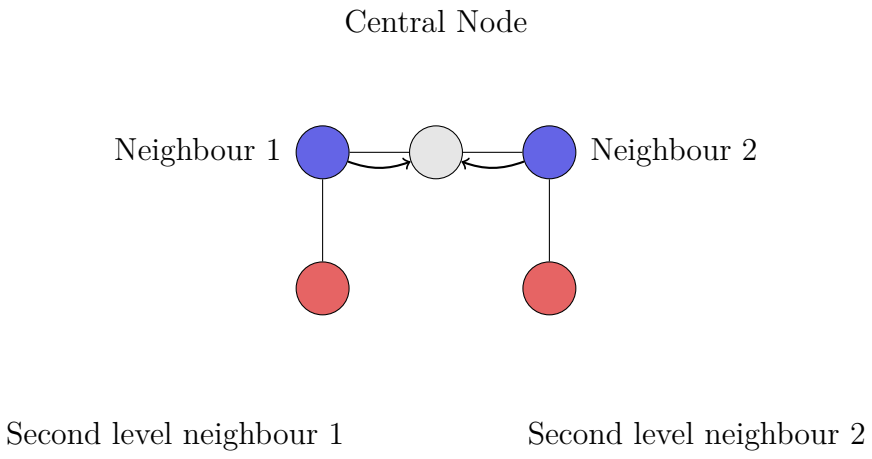

Figure 2.3 Information flow from immediate neighboring nodes to the central node.

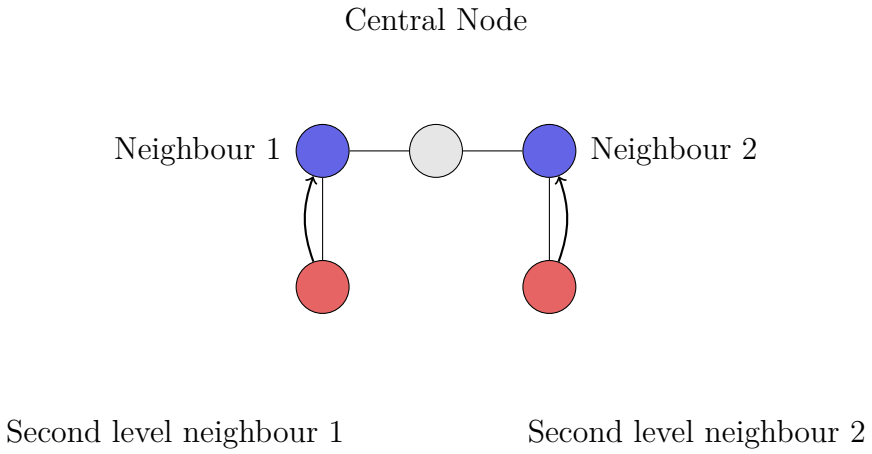

Figure 2.4 Aggregation of information from second-level neighbours.

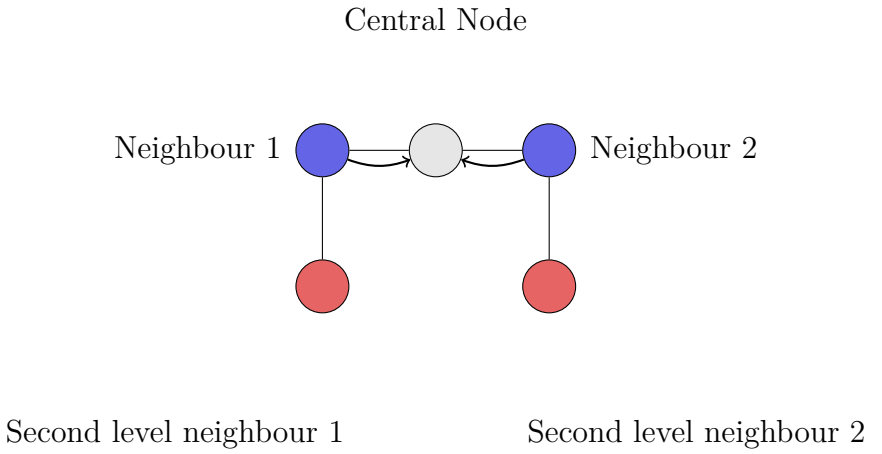

Figure 2.5 Final aggregation of features in the central node.

Following to the GCNN's feature aggregation, the central node's enhanced feature is introduced into a traditional neural network for additional processing, which can serve tasks like node classification or graph classification as illustrated in Figure 2.6.

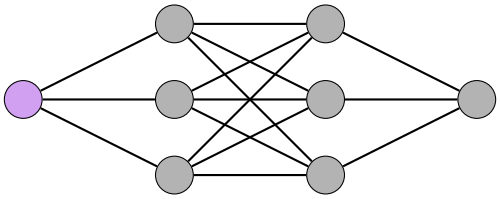

Figure 2.6 Neural network processing of the aggregated feature from the central node.

2.3.1 Representation

Applying machine learning to combinatorial optimization problems, like MIP, poses unique challenges. The representation of the problem must be compatible with the machine learning algorithm.

MIPs are defined by their size, which can vary drastically from instance to instance. This, combined with complex interactions between variables, means representing these problems for machine learning algorithms is no simple task. Furthermore, a MIP can be formulated in many different ways and the size of the decision space (i.e., the number of variables) can also fluctuate.

To make MIPs compatible with machine learning algorithms, the representation must be independent of the problem instance's size. Essentially, the key characteristics of the MIP need to be captured in a compact, fixed-size vector. Importantly, this representation must be invariant to certain changes such as reordering of constraints or variables.

One effective approach to handle variable-sized decision spaces is training a neural network to assign a branching score to each variable. The variable with the highest score is then selected for branching.

Reference [20] made a useful distinction between static and dynamic features in their approach. Static features depend solely on the problem's formulation and remain constant throughout the solution process, while dynamic features evolve as the algorithm explores the solution space.

However, these early representations relied heavily on handpicked features, a labor-intensive and non-generalizable approach. A more holistic representation using GCNNs was later proposed.

The use of GNNs to model MIPs was introduced by [34], outlining a structure for developing a rapid, albeit non-optimal, greedy heuristic. This approach was expanded upon by [35], who applied a GCNN to quickly solve Boolean satisfiability problems (SAT).

Building on these works, [4] demonstrated that GCNNs could be used not only to develop heuristics but to accelerate exact algorithms. In their approach, a MIP instance is represented as a bipartite graph, with two disjoint sets of nodes representing variables and constraints. The edges between these nodes bear weights reflecting their contribution to the associated constraint. A diverse set of features, mainly solver metrics, are associated with each node.

However, [36] questioned the efficiency gains of the GCNN approach, attributing a significant portion to the superior computational power of GPU hardware. They propose a hybrid architecture, combining a graph model in the initial stages of problem-solving with a traditional decision tree model for the remainder of the process. This approach, they argue, delivers superior performance on CPU-only machines.

2.3.2 Generalization

The degree to which machine learning can discover and exploit structure in combinatorial optimization problems depends on the level of specificity or generality in the training set. Training on a narrow range of problems allows the model to specialize and perform well on similar instances, while training on a diverse set of problems pushes the model to discover more general structures applicable across various contexts. However, the latter tends to be more challenging due to increased complexity and diversity of the problems.

We provide some concrete examples of these strategies below:

1. Reference [34] trained a model to develop an effective branching strategy at the initial nodes of a problem. The learned strategy is specific to each problem instance but can be reused for subsequent nodes within that instance.

2. References like [4] and [37] focused on problems within a particular family. By specializing their models on a specific type of problem, they were able to achieve high performance within that domain.

3. Reference [38] strived for a more general approach, training a model on a broad spectrum of MILPs. They did not rely explicitly on the problem structure, but rather used deep neural networks to represent the branch and bound search tree with a newly proposed feature.

4. Prior works like [39] and [40] aimed for ultimate generalization across all problem types. They trained models to determine which traditional heuristic should be used for a given instance based on its structure. Although this task can be simplified for specific problem classes, its general nature makes it an applicable strategy for a wide range of problems.

The trade-off between specificity and generality in these strategies reflects the inherent balance between achieving high performance on a specific type of problem and maintaining applicability across a broad range of problem types.

2.4 Data

Training a model to make effective branching decisions through imitation learning requires a dataset that maps system states to good variable decisions. This necessitates solving standard optimization problems on a large number of instances and recording the decisions made by the solver.

2.4.1 Instance Generation

Although it is possible to study a large number of states with a limited number of instances, these states are often highly similar. Therefore, to achieve a truly varied dataset, it is crucial to repeat the data collection process across a wide variety of instances unless the learned task is instance-specific.

Traditionally, the datasets used to evaluate optimization problem-solving methods aren't very large. Testing a few hundred problems is usually sufficient to compare different approaches and measure performance according to various metrics. However, developing a machine learning algorithm often necessitates massive datasets for training [34]. To be efficient, most machine learning algorithms for combinatorial optimization require training on several hundreds of thousands of instances [41].

In this context, data augmentation is often essential to create datasets of appropriate size. This typically involves the use of an instance generator. One limitation of this approach is that such generators aren't always available. Furthermore, the instances created by these generators often don't follow the same distribution as the actual problem. Therefore, caution is needed as the model may learn structures that aren't inherent to the actual problem but rather artifacts of the generator's imperfections [32]. Reference [20] propose a generator that combines the constraints and objectives of different instances.

2.4.2 Data Acquisition

The strategy for collecting training data is straightforward: solve a problem instance using the explore-then-strong-branch strategy and, when strong branching is used, record the state of the branch and bound tree along with the selected variable.

According to Prouvost et al. [41], to collect independent and identically distributed data for a model, it is necessary to go beyond the systematic use of strong branching when collecting data. This necessitates the use of what they term the explore-then-strong-branch strategy. Using only strong branching would result in correlated data and a lack of diversity in the explored system states. To avoid this, the authors propose using an alternative branching technique most of the time – in their case, pseudocost branching – to explore the decisions of strong branching in less ideal states.

Khalil et al. [34] directly train their algorithm to make good variable choices on each instance by learning from decisions made on the initial nodes. The model then uses this learned information to solve the remainder of the problem. In this case, as the model is instance-specific, it isn't necessary to generate more instances.

2.4.3 Metrics

It is difficult in the case of solver acceleration to determine which metrics to optimize. Indeed, it is desirable to have the fastest solver possible. It is also possible that one goal is to have a solver that does not require much memory although it is not the most common concern. However, it is not possible to evaluate a machine learning algorithm directly on these metrics, because in learning to branch methods ,the nature of the decision taken is local and its quality is difficult to evaluate from the global performance of the algorithm.

The challenge is therefore to find optimizable local metrics that have a positive influence on the global metrics that really interest us. The most used global metrics during the test phase are the computation time, the primal dual gap, and the size of the explored branch and bound tree [5]. It will rarely be the same ones on which the machine learning algorithms will be trained.

In [4], the authors train their algorithm on the accuracy it obtains on the task of imitating strong branching decisions. Comparing the different machine learning approaches is however not enough. In a second phase of evaluation they compare the different machine learning algorithms, according to the time required for computation, the number of nodes explored and the number of win (number problems solved the fastest by each algorithm). These new metrics represent the real objectives on which it is impossible to train directly.

2.4.4 Reinforcement Learning

RL techniques for variable selection offer certain benefits that are not available with imitation learning methods. Notably, their performance isn't limited by the efficiency of the heuristics they aim to imitate. Unfortunately, as discussed earlier, whether it is through heuristic methods or RL approaches, existing algorithms are often constrained by the local nature of the evaluation criteria they employ. For a RL method to outperform an imitation learning technique, a more global training metric is needed, which is challenging to set as explained in the previous section.

Éthève et al. [33] propose a method to train a RL model to optimize a global criterion – specifically, the size of the branch and bound subtree originating from the selected variable. it is indeed possible to trace the impact of each decision on the size of this tree. Regrettably, measuring the size of this tree necessitates solving the optimization problem multiple times for each instance, making the training process exceedingly slow.

Another advantage of RL learning techniques is their independence from the need to construct a data set for training – other than a vast set of instances.

However, RL strategies also encounter many issues. Because the length of the episodes is proportional to the model's performance, policies initialized randomly perform poorly, leading standard RL algorithms to be particularly slow during the initial stages of training. Collecting a large dataset for training can take a significant amount of time. Additionally, the decision space's size, unique to each instance, impedes the Markovian learning process [4].

At the time of the writing of this Master's thesis, few competitive methods employ RL learning.

2.5 Unit commitment

The UC problem represents a significant task in the operations of power systems. It entails the optimal scheduling of power generation units to serve the projected load demand at the lowest feasible cost while adhering to an array of operational constraints.

2.5.1 Formulations of the UC Problem

The UC problem can be understood through two distinct formulations, which have their unique considerations and implications.

2.5.1.1 Unit Commitment with Economic Dispatch

Our first formulation of the UC problem, known as unit commitment with economic dispatch, seeks to minimize the operational costs of power generation while meeting the aggregated demand. It achieves this by optimally dividing the load among online units, thereby streamlining the allocation process. This approach simplifies the problem by leaving out network constraints and presuming unrestricted power flow within the grid [42]. While this abstraction may appear to compromise on realism, it crucially aids in scalability. This ensures that the system can effectively manage larger and more complex power infrastructures.

2.5.1.2 Unit Commitment with Power Flow

Addressing the computational intensity due to the non-convex nature of the Unit Commitment with Optimal Power Flow (UC-PF) model, the problem is frequently resolved using Second-Order Cone (SOC) relaxation techniques. These techniques provide a greater level of realism compared to approximation methods, albeit with increased computational requirements [42, 43]. The UC-PF model is an enhancement of the UC model with an economic dispatch approach, embedding specific grid-based transmission constraints within the UC issue. These constraints include limitations on transmission lines and the magnitude of bus voltage [44]. Despite the increased computational load, this sophisticated formulation offers a more accurate representation of the power system's operations.

2.5.2 Decomposition Techniques for the UC Problem

The application of advanced methodologies to decompose the Unit Commitment (UC) problem into more manageable components has facilitated more efficient solutions within large-scale infrastructures. Notably, reference [45] implemented a Benders-style decomposition that employed a dynamically enhanced master problem. This application of Kron reduction served to condense the representation of non-contingency operating scenarios. Research conducted by reference [46] involved the application of Benders' decomposition to a second-order conic relaxation of UC-PF problem. This process, involving the sequential execution of continuous convex optimization problems, maintained power flow feasibility.

2.5.2.1 Machine Learning applications in the UC Problem

The field of machine learning has recently been employed to tackle the UC problem, demonstrating promise in both initial applications and more advanced uses.

Many papers try to solve this problem directly by using an end-to-end learning method [47, 48]. The research by Pineda and Morales [49] serves as an early example, showing how simple machine learning methods, based on past UC examples, can quickly create good solutions. Importantly, [49]'s team suggests that any improvements in the learning method should lead to a significant increase in the quality of the results. [50] introduced machine learning methods to harness data from previously resolved instances to enhance MIP solver efficiency by predicting redundant constraints, ideal initial feasible solutions, and probable optimal solution regions the researchers managed to reduce the problem size. The study further demonstrated the method's robustness against data shifts.

On the other hand, [51] re-expresses the UC problem into a form of decision-making process and use a sophisticated learning algorithm. These techniques have been shown to be more efficient than traditional methods, but they come with their own issues. For instance, they show similar levels of accuracy with less computation time, but their worst-case performance is uncertain [52]. Moreover, the size of the problems they've been tested on is much smaller than typical industry problems, suggesting potential issues with scalability [53]. Despite these potential hurdles, machine learning's ability to address the UC problem is a promising direction for future studies and applications.

2.6 Technologies

While machine learning can significantly enhance certain decision-making processes in the branch and bound algorithm, such as variable selection, other elements still rely heavily on heuristics to ensure rapid resolution. Fortunately, these heuristics already exist and have been meticulously refined over the years within solvers.

To develop a competitive solution method in the current landscape, it is crucial to leverage these established solvers. Novel methods, therefore, need to be integrated into these existing frameworks to yield effective results.

The evolution of machine learning-accelerated solvers is fundamentally intertwined with the progression of available technology. This section sheds light on a couple of pivotal technologies: `SCIP` and `Ecole`.

2.6.1 `SCIP`

`SCIP` (Solving Constrained Integer Programs) stands as one of the most proficient non-commercial solvers for integer programming. Its transparency concerning its solving methods sets it apart, facilitating the easy adaptation and integration of novel methods [54].

2.6.2 Ecole

Ecole, an acronym for Extensible Combinatorial Optimization Learning Environments, is an environment inspired by OpenAI's Gym [55]. It presents a range of control problems that occur in combinatorial optimization solvers as Markov Decision Processes (MDPs). SCIP is the integrated solver in Ecole [41].

Ecole bridges the gap between SCIP and the suite of conventional machine learning tools available in Python, enabling fast, efficient, and scalable operations [56] [57]. Its role is instrumental in streamlining the process of incorporating machine learning methodologies into combinatorial optimization problems.

CHAPTER 3 ARTICLE 1: LEARNING-ACCELERATED EXACT SECOND-ORDER CONE PROGRAMMING FOR UNIT COMMITMENT

Philippe Maisonneuve • Antoine Lesage-Landry

Submitted to: INFORMS Journal on Optimization [1]

Date of submission: August 2nd, 2023

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.

3.1 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 [42]. 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 [58].

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 [32], 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 [4], 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 [49]. 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.

3.1.1 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 [4] 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 3.1.2, we review the relevant literature. In Section 3.2, we present the UC formulation to be solved via the methodology detailed in Section 3.3. Section 3.4 provides the results of our numerical experiments. Lastly, we conclude in Section 3.5.

3.1.2 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 [42]. 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 [44]. This problem is typically solved using a convex relaxation, which offers a higher degree of realism in comparison to approximations [42] but also brings with it an increased computational burden. Particularly, our work leverages the SOC relaxation introduced in [43].

Reference [46] 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 [45] 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 [49] 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 [53]. 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 [47, 48]. [50] employed machine learning to improve MIP solver performance by predicting constraints and solution areas, demonstrating resilience to data shifts. Reference [51] 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 [59], 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 [20].

Recent work has evolved these techniques further by introducing graph neural networks (GNN) to represent MIPs [34]. 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 [36].

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 [34, 37], while others sought to generalize across all types of instances [15, 39, 40].

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 [5, 41]. Collecting the data involves solving numerous instances of the prob-

lem 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 [34].

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 [33, 60, 61].

3.2 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 [42] while the power flow formulation is adapted from [43].

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 $i \in G$;

$S^{\mathrm{d}}_{i,t} \in \mathbb{C}$ 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 $ij \in \mathcal{L}$;

$\theta^{\mathbf{\Delta}}_{ij} \in \mathbb{R}$ 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 $t \in \mathcal{T}$;

$S^{\text{gen}}_{i,t} \in \mathbb{C}$ 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:

$$\sum_{t \in \mathcal{T}} \sum_{g \in \mathcal{G}} a_g p^2_{g,t} + b_g p_{g,t} + C_S \xi_{g,t} \leq x. \tag{3.1}$$

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:

$$\sum_{g \in \mathcal{G}} p_{g,t} = D_t \quad \forall t \in \mathcal{T} \tag{3.2}$$

$$\sum_{g \in \mathcal{G}} q_{g,t} = Q_t \quad \forall t \in \mathcal{T}. \tag{3.3}$$

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

$$\overline{S}_g \xi_{g,t} \leq S^{\text{gen}}_{g,t} \leq \underline{S}_g \xi_{g,t}, \quad \forall g \in \mathcal{G}, \forall t \in \mathcal{T}, \tag{3.4}$$

where,

$$\overline{S}^{\text{gen}}_{g,t} = p_{g,t} + \jmath q_{g,t}, \quad \forall g \in \mathcal{G}, \forall t \in \mathcal{T}. \tag{3.5}$$

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

by:

$$\sum_{k=t}^{\min(\overline{T}_g,|\mathcal{T}|)} \xi_{g,k} \geq \overline{T}_g(\xi_{g,t} - \xi_{g,t-1}), \quad \forall g \in \mathcal{G}, t \in \mathcal{T} \setminus \{1\} \tag{3.6}$$

$$\sum_{k=t}^{\min(\underline{T}_g,|\mathcal{T}|)} (1 - \xi_{g,k}) \geq \underline{T}_g(\xi_{g,t-1} - \xi_{g,t}), \quad \forall g \in \mathcal{G}, t \in \mathcal{T} \setminus \{1\}. \tag{3.7}$$

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.

$$p_{g,t} - p_{g,t-1} \leq \overline{r}_g, \quad \forall g \in \mathcal{G}, t \in \mathcal{T} \setminus \{1\} \tag{3.8}$$

$$p_{g,t-1} - p_{g,t} \leq \underline{r}_g, \quad \forall g \in \mathcal{G}, t \in \mathcal{T} \setminus \{1\}. \tag{3.9}$$

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:

$$\begin{aligned} &\min_{x, S^{\text{gen}}_{g,t}, \xi_{g,t}} \quad x \\ &\text{subject to (3.1)}-\text{(3.9)}, \\ &\qquad \xi_{g,t} \in \{0,1\} \quad \forall g \in \mathcal{G} \quad \forall t \in \mathcal{T}. \end{aligned} \tag{3.10}$$

Traditionally the UC-ED and the optimal power flow problems are 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 [46].

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:

$$W_{ij,t} = V_{i,t} V^*_{j,t}, \quad \forall i, j \in \mathcal{N}, \forall t \in \mathcal{T}. \tag{3.11}$$

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

$$(\underline{v}_i)^2 \leq W_{ii,t} \leq (\overline{v}_i)^2, \quad \forall i \in \mathcal{N}, \forall t \in \mathcal{T}. \tag{3.12}$$

In UC-PF, the constraints (3.2)−(3.3) are removed and the power balance at each node is ensured by:

$$S^{\text{gen}}_{i,t} - S^{\text{d}}_{i,t} = \sum_{ij \in \mathcal{L}} S_{ij,t}, \quad \forall i \in \mathcal{N}, \forall t \in \mathcal{T}, \tag{3.13}$$

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 (3.11), the power flowing in each line is modelled by:

$$S_{ij} = (W_{ii,t} - W_{ij,t})Y^*_{ij,t}, \quad ij \in \mathcal{L}, \tag{3.14}$$

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:

$$|S_{ij,t}|^2 \leq \left(\overline{S}_{ij}\right)^2, \quad \forall ij \in \mathcal{L}, \forall t \in \mathcal{T}. \tag{3.15}$$

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

$$\tan\left(-\theta^{\mathbf{\Delta}}_{ij}\right) \text{Re}(W_{ij,t}) \leq \text{Im}(W_{ij,t}) \leq \tan\left(\theta^{\mathbf{\Delta}}_{ij}\right) \text{Re}(W_{ij,t}), \quad \forall ij \in \mathcal{L}, \forall t \in \mathcal{T}. \tag{3.16}$$

Lastly, we use the SOC relaxation of [43] to obtain a mixed-integer convex UC-PF formulation and impose:

$$|W_{ij,t}|^2 \leq W_{ii,t} W_{jj,t} \ \forall ij \in \mathcal{L}, \forall t \in \mathcal{T}, \tag{3.17}$$

instead of (3.11).

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

$$\begin{aligned} \min_{x, S^{\text{gen}}_{g,t}, \xi_{g,t}} \quad & x \\ \text{subject to } & \text{(3.1), (3.4)−(3.9), (3.12)−(3.17)} \\ & \xi_{g,t} \in \{0,1\} \quad \forall g \in \mathcal{G} \quad \forall t \in \mathcal{T}. \end{aligned} \tag{3.18}$$

3.3 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 [4]'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.

3.3.1 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.

3.3.2 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 3.2 depicts a tripartite graph for constraints (3.1) and (3.17).

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

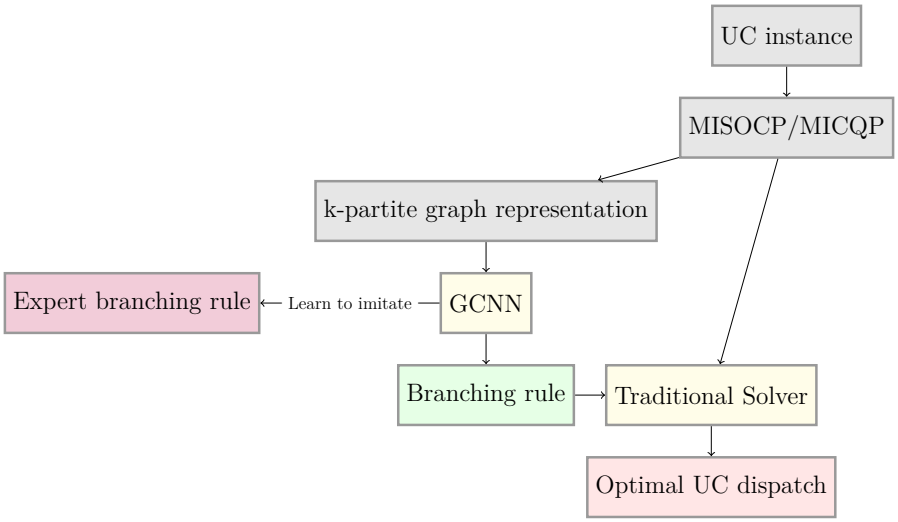

Figure 3.1 Flow chart representation of the presented methodology

accurately reflecting the structure of the MISOCPs.

3.3.3 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 3.3 provides an example of the bipartite graph for constraints (3.1) evaluated at $t = |\mathcal{T}|$ and (3.6) 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.

3.3.4 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

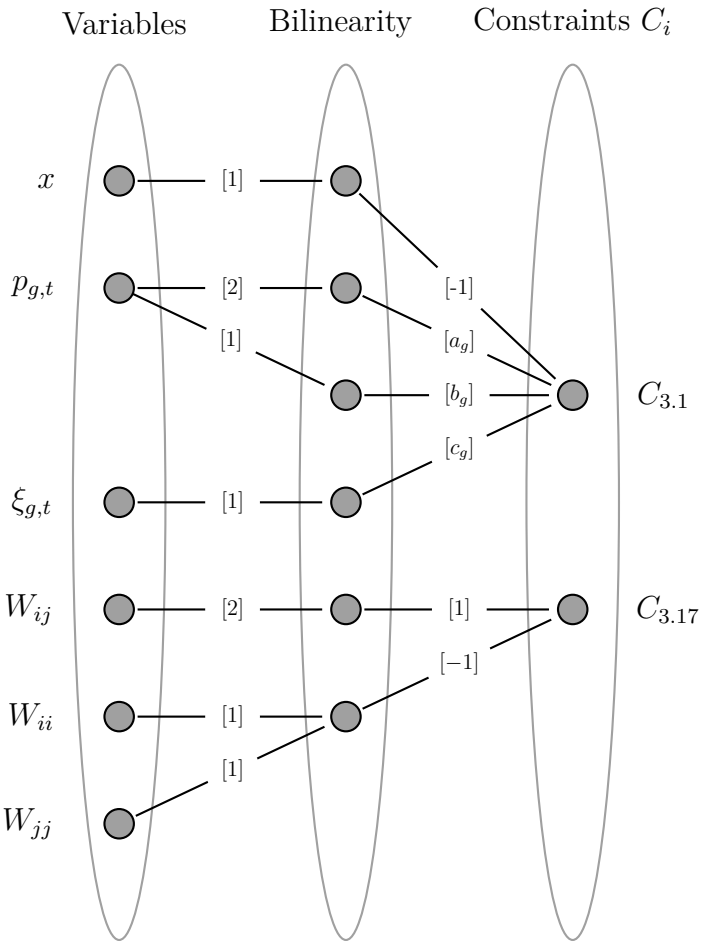

Figure 3.2 Tripartite graph representation of constraints (3.1) and (3.17)

local graph structures and of propagating information across the graph, enabling the effective handling of irregular and dynamic structures frequent in graph data [62].

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

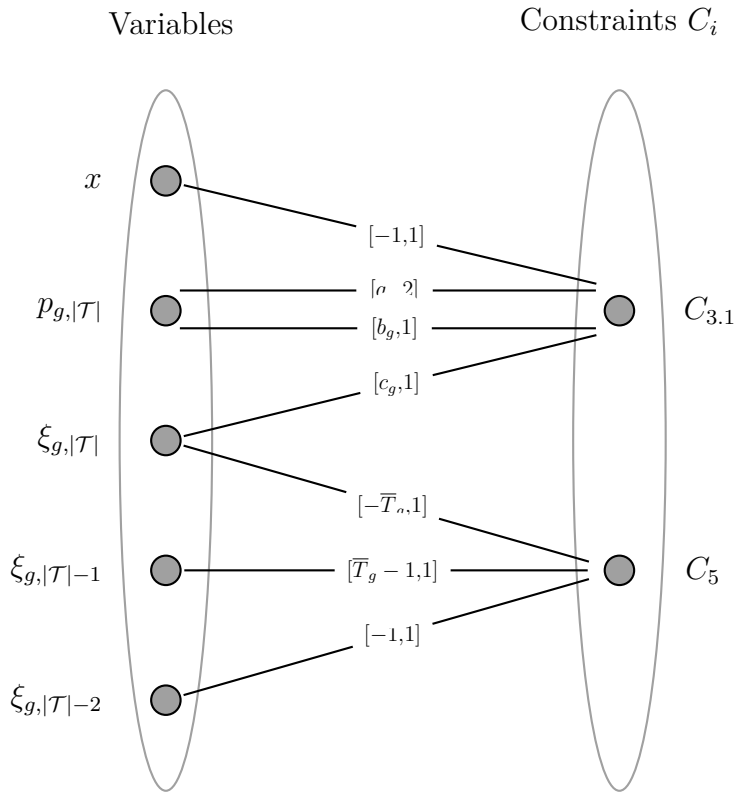

Figure 3.3 Bipartite graph representation of constraints (3.1) for $t = |\mathcal{T}|$ and (3.6) for $t = |\mathcal{T}| - 1$

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.

3.3.5 Expert Strategy

A multitude of branching strategies have been proposed, each possessing its unique set of attributes and potential drawbacks [63].

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 [64]. 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 [64]'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 3.1 and 3.2 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.

3.3.6 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 [65], 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 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.

3.3.7 Data collection strategy

We utilize the data collection mechanism introduced by [4]. 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.) [66]. By diversifying the data collection process, we aim to satisfy this assumption and thereby improve the reliability of our results.

Table 3.1 Comparison of branching strategies for the MISOCP UC-PF (RTS-96 test case)

Branching Strategy	Average Time (s)	Average Node Count
relpscost	51.7	49623.20
vanillafullstrong	331.8	11261.60
inference	65.3	58744.40
pscost	45.6	33949.30
mostinf	236.5	199840.30
leastinf	28.1	23121.30
allfullstrong	**582.7**	**4792.90**
cloud	86.2	7160.80

Table 3.2 Comparison of branching strategies for the MICQP UD-ED (RTS-96 test case)

Branching Strategy	Average Time (s)	Average Node Count
relpscost	2.57	2477.80
vanillafullstrong	17.29	567.30
inference	2.23	2936.30
pscost	1.48	1695.40
mostinf	10.73	9990.20
leastinf	1.36	1154.60
allfullstrong	**26.63**	**235.90**
cloud	5.39	358.20

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.

3.3.8 Imitation learning

Imitation Learning (IL) is a type of machine learning where an agent learns to perform tasks by mimicking expert behaviour [59]. 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.

3.3.9 Data Collection

We first generate a dataset of branching decisions using the data generation strategy as described in Section 3.3.7. 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).

3.3.10 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.

3.3.11 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.

3.3.12 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 compu-

tational burden of the UC problem and enabling more efficient operations in power systems while remaining exact by using the method illustrated in 3.1.

3.4 Results

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

3.4.1 Computational tools

In this research, the computational tools we used are central to our methodology and results. We leveraged `SCIP` [54] and `Ecole` [41], 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 [67] except for RTS-96 which is from [68].

3.4.2 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 3.3 and 3.4 and illustrated in Figures 3.4 and 3.5.

Table 3.3 Performance of the MISOCP GCNN & `SCIP` method for UC-PF

Grid	Number of Generators	Number of Lines	`SCIP` Average Time (s)	GCNN & `SCIP` Average Time (s)	Gain
RTS-96	99	120	867.8	783.3	9.7%
IEEE-118	54	186	1003.7	859.2	14.4 %
CASE300	70	411	1757.9	1442.6	17.9 %

Table 3.4 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	297	924.6	693.4	25.0%
CASE2848RTE	547	1487.5	1087.9	26.8%
CASE6470RTE	1330	3362.7	2212.4	34.2%
CASE6515RTE	1389	3213.8	2113.6	34.0%

Following the tabular data presented above, we include two graphical representations presented in Figures 3.6 and 3.7. These graphs capture the computational efficiency of each algorithm – `Gurobi` [69], `SCIP`, and our proposed GCNN & `SCIP` – over a collection of 100 UC instances generated randomly by adding normal random noise to load profiles.

Figures 3.6 and 3.7 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.

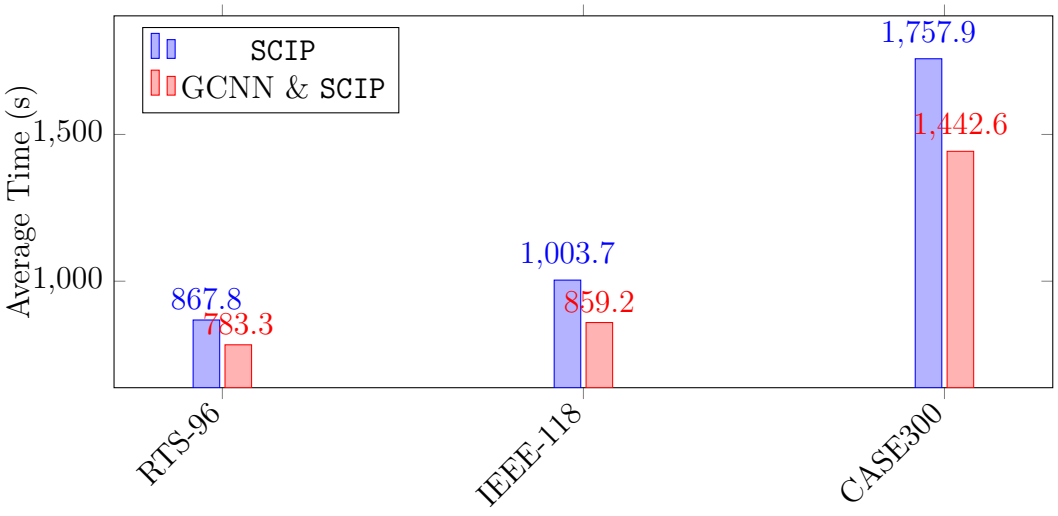

Figure 3.4 Bar chart representation of the performance of the MISOCP method for UC-PF

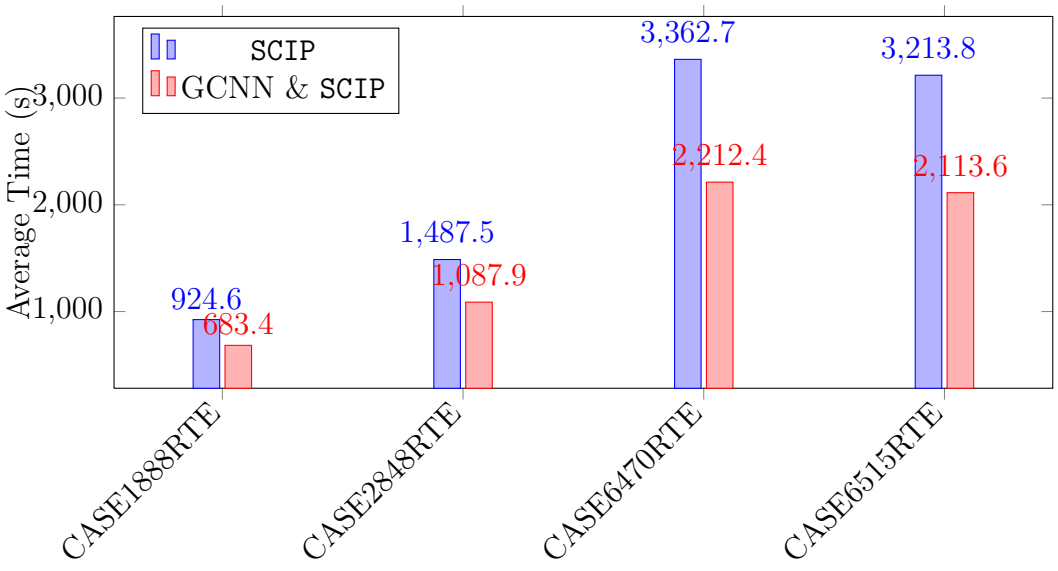

Figure 3.5 Bar chart representation of the performance of the MICQP method for UC-ED

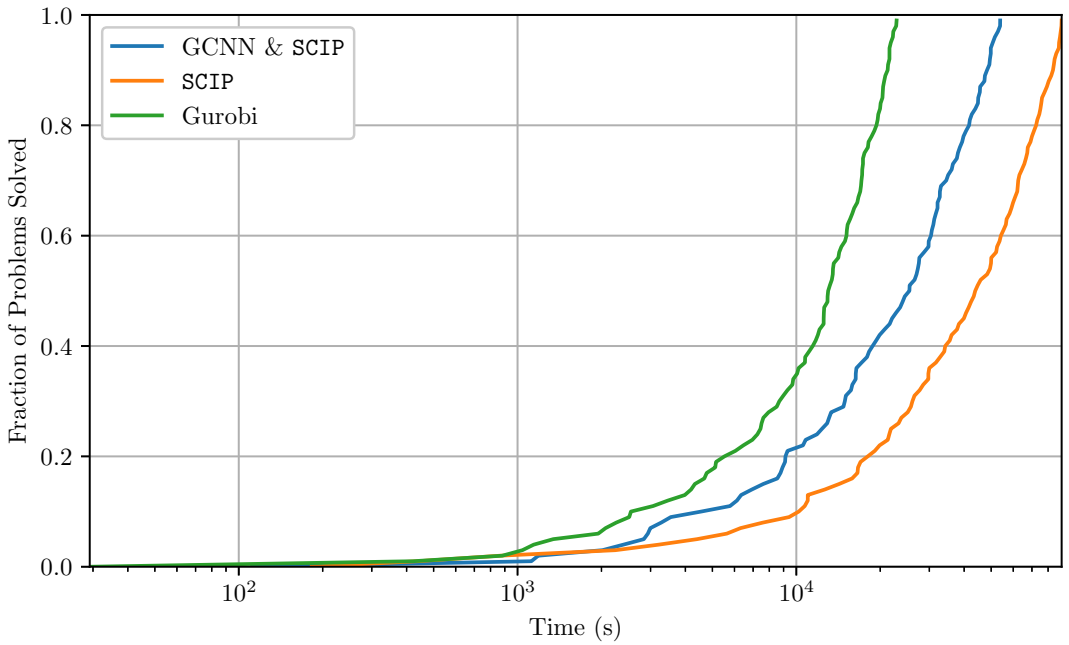

Figure 3.6 Computational Efficiency for the IEEE-118 test case on 100 MISOCP UC problem instances

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 3.5 and Figure 3.8.

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.

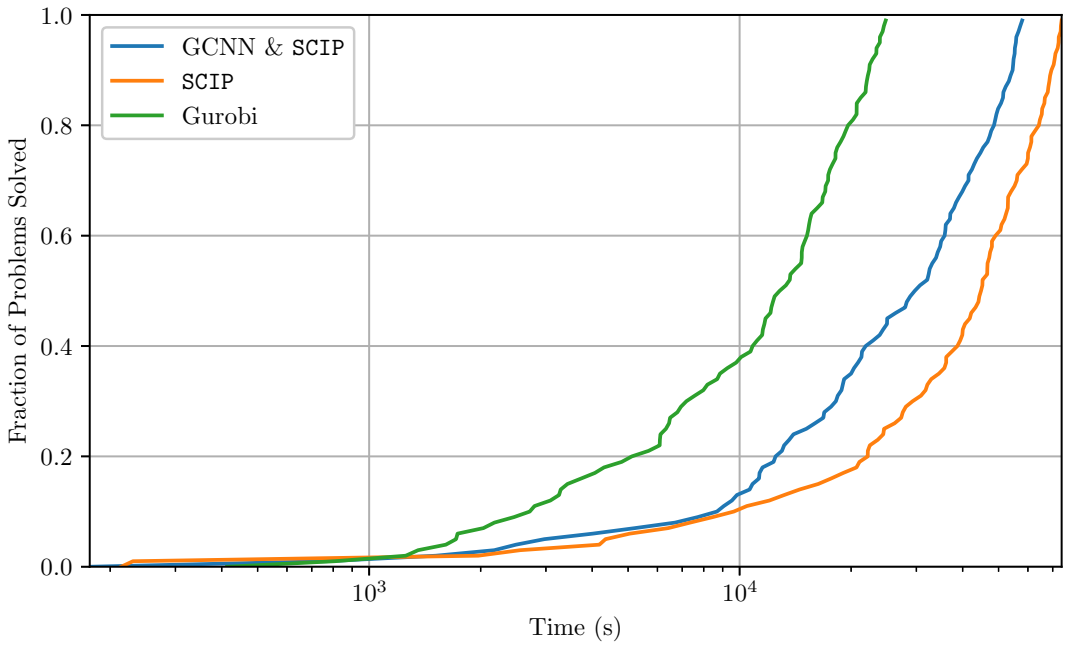

Figure 3.7 Computational Efficiency for the CASE1888RTE test case on 100 MICQP UC problem instances

3.5 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 [4], 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. Ad-

Table 3.5 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	820.2	736.6	10.2%
RTS-96	Cut	776.4	712.1	8.2%
IEEE-118	Moved	1034.2	913.8	11.7%
IEEE-118	Cut	984.9	847.2	14.0%
CASE300	Moved	1620.5	1326.4	18.1%
CASE300	Cut	1702.7	1422.2	16.4%

ditionally, 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.

3.6 Biographies

Philippe Maisonneuve

received the B.Eng. degree in Software Engineering from Polytechnique Montreal, QC, Canada, in 2022. He is currently working toward his Master's degree, focusing on the utilization of machine learning in energy systems and the integration of renewable energy into power grids. This manuscript is the outcome of his Master's research project aiming at bridging his undergraduate degree in Software Engineering with his passion for energy systems and renewable integration.

Antoine Lesage-Landry

is an Assistant Professor in the Department of Electrical Engineering at Polytechnique Montreal, QC, Canada. He received the B.Eng. degree in Engineering Physics from Polytechnique Montreal, in 2015, and the Ph.D. degree in Electrical Engineering from the University of Toronto, ON, Canada, in 2019. From 2019 to 2020, he was a Postdoctoral Scholar in the Energy & Resources Group at the University of California, Berkeley, CA, USA. His research

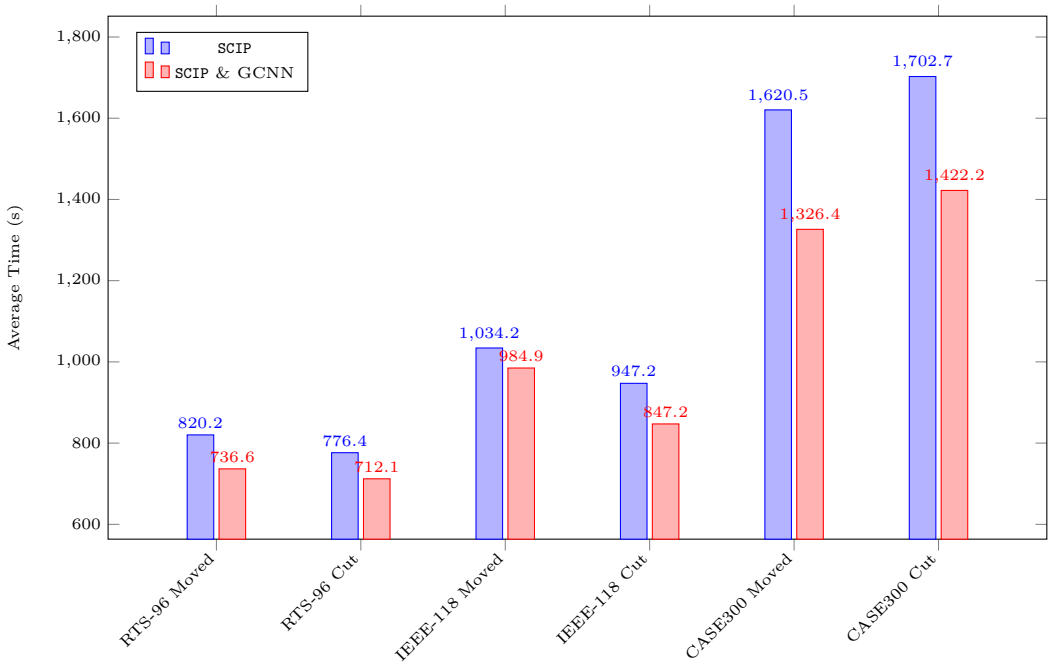

Figure 3.8 Bars chart representation of the performance of the MISOCP method for UC-PF after grid modification

interests include optimization & machine learning, and their application to renewable power systems.

Acknowledgements

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

CHAPTER 4 GENERAL DISCUSSION

Our study aimed at addressing the optimization challenges in UC problems within power systems, primarily driven by their binary nature and non-convex power flow constraints. We introduced a novel approach using a GCNN, leveraging a unique variable-constraint k-partite graph representation of MISOCs and MICQPs. This method allows for a thorough understanding of the complex relationships between the variables and constraints of these problem.

Our model, trained through imitation learning under the strong branching expert rule, learned effective variable selection policies. Unlike many learning approaches, our method prioritizes the preservation of solution optimality, an essential feature for UC problems, ensuring optimal scheduling of power generation units.

This approach was validated on several standard grid testcasesand consistently outperformed the traditional solver `SCIP`, even when grid modifications were introduced. These results underscore our methodology's effectiveness and robustness, providing clear evidence of GCNN's potential to enhance the efficiency of UC optimization solvers.

Our work demonstrates the efficacy of integrating machine learning techniques, like GCNNs, into traditional optimization frameworks, which opens up new avenues for solving complex optimization problems.

The ability of our method to handle grid changes emphasizes its suitability for real-world applications, particularly given the dynamic and unpredictable conditions of real-world power systems.

Lastly, the performance comparison with proprietary solver `Gurobi` suggests potential avenues for enhancing solver efficiency. While `Gurobi` currently outperforms our method, the comparative performance is promising. Implementing similar techniques into `Gurobi`'s framework might yield substantial improvements, increasing the efficiency and speed of solving complex optimization problems in power engineering, with potential benefits for optimizing generation and reducing operational costs.

CHAPTER 5 CONCLUSION AND RECOMMENDATIONS

In this master's thesis, we propose a novel methodology using a GCNN to enhance the UC mixed-integer optimization solvers. We represent the complex UC optimization process, crucial in power system operations, as a MISOCP. The key steps in our methodology are creating a k-partite graph from the MIP's constraints and then training our GCNN model on this graph using imitation learning. We propose an approach based on tri-partite graph to modelize MISOCP which is then specialize as a bipartite graph for an alternative MICQP formulation. This process allows us to derive efficient variable selection policies for the branch and bound algorithm, significantly speeding up the optimization process.

Crucially, unlike other methods, our approach preserves solution optimality and effectively integrates problem-specific information into the optimization process. This strategy bridges the gap between traditional optimization algorithms and machine learning techniques, offering a robust and efficient framework for tackling complex UC optimization problems.

Our research has effectively demonstrated the robustness and efficacy of our machine learning-based MIP approach in addressing two distinct formulations of the UC problem across a variety of power grid models. The results, derived from numerical analyses conducted on seven standard grid models, underscore our method's adaptability and consistency, thereby highlighting its potential for significant advancements in power system optimization.

This work has successfully demonstrated the transformative potential of integrating machine learning into optimization solvers to develop tailored heuristic. We specifically leveraged the power of GCCNs to generate a sophisticated understanding of the inherent intricacies of MIPs.

Our methodology extends the application of techniques developed for mixed-integer linear optimization problems to non-linear problems such as MISOCPs and MICQPs. It consistently outperforms the `SCIP` solver it is based upon, even under altered scenarios. However, it is crucial to note that these performances are problem-specific. Unlike MILP, where full strong branching consistently explores fewer nodes of the branch and bound tree than other branching strategies, there is no universal expert policy for non-linear programs as suggested by the results of [64].

The process of training a model to make effective branching decisions through imitation learning necessitates a dataset that maps system states to optimal variable decisions. This requires solving standard optimization problems on a large number of instances and recording

the decisions made by the solver. In our case, the generation of these instances was achieved using Gaussian noise applied to limited load data. These generated loads may not accurately reflect real scenarios, and an organization with access to more extensive historical data might yield different results.

Moreover, while our approach has demonstrated robustness under some grid modifications, it is important to consider that under substantial changes or unusual situations, the model may be out of distribution and not perform as expected.

Finally, it is crucial to acknowledge its inherent limitations. Our method leans heavily on both graphical processing unit (GPU) and central processing unit (CPU) computing capabilities, whereas the traditional solver SCIP primarily depends on the CPU. This difference in hardware utilization makes it challenging to accurately measure the relative performance improvement. Specifically, in a scenario where our experiment run on a computer with an excellent CPU but lacks a suitable GPU, the outcomes may not be as favourable as observed in our study. It is worth noting that our calculations were performed on the Digital Research Alliance of Canada's cluster, which offered access to high-performance CPUs and GPUs. Hence, our method's practical utility might be constrained in environments where similar computing resources are not readily available, marking a potential limitation to its wide-scale applicability. We can speculate that for a critical problem like UC, systems operators, would have access to similar infrastructure.

Therefore, while our work presents promising advancements in power system optimization, these considerations underline the necessity for further research and development in this field.

Future Work

Our work has demonstrated substantial potential for addressing the UC problem using a machine learning-based optimization approach. Going forward, there is room for exploration at the intersection of machine learning and combinatorial optimization, especially in enhancing computational speed across diverse power systems MIPs under varying network reconfiguration scenarios. The degree of machine learning's efficacy in discovering and exploiting structure in combinatorial optimization problems is hinged on the specificity or generality of the training set. Models trained on a specific range of problems excel at similar instances, while those trained on diverse problems tend to identify more broadly applicable structures across different contexts.

Take, for instance, [5]; they trained a model to develop an effective branching strategy at the initial nodes of a problem, crafting a strategy that was both instance-specific yet reusable for

subsequent nodes. Conversely, references like [4] and [37] honed their models to specialize in a certain type of problem, ensuring high performance within that domain.

Reference [38] pursued a more general approach by training a model on a wide range of MILPs, employing deep neural networks to represent the branch and bound search tree with a novel feature. In earlier studies, [39] and [40] aimed for ultimate generalization across all problem types by training models to select a traditional heuristic based on an instance's structure.

We may speculate on the existence of a trade-off between specificity and generality in these strategies, reflecting the need to balance high performance on a specific problem type and wide applicability across various problems. This balance is particularly relevant when considering network reconfiguration in power systems. By developing a method capable of accelerating the solution phase for multiple grid configurations, we can improve the resiliability of our approach, potentially leading to a more robust and versatile optimization technique for power system optimization.

An intriguing future direction involves enhancing the graph representation through the integration of additional, dynamic features. Currently, our representation of the UC problem as a variable-constraint k-partite graph has been effective in capturing mixed-integer problems' inherent structure. Still, there is room to enrich this model.

By introducing more features into the graph, we could describe more complex variable-constraint relationships. Features could include edges representing different variable interactions or nodes encapsulating the temporal dynamics of power generation units. Such features could provide a more nuanced understanding of the problem, enabling our model to adapt better to changes in problem parameters.

Moreover, these enhancements could also capture more complex constraints and dynamics. In the context of power systems, we could integrate features representing the stochastic nature of renewable energy sources, thereby enhancing our model's ability to handle the inherent non-convexity and uncertainty in these problems.

By augmenting our graph representation, we could potentially develop more robust and adaptive MIP solvers that can cope with changes in problem topology and parameters. This flexibility is particularly significant in the context of power systems, which are subject to continuous fluctuations due to demand variability, equipment failures, and the integration of renewable energy sources.

For other applications, the potential of GCNNs in optimization is expansive and largely untapped. The structure of MIPs, when represented as graphs, offers a rich source of in-

formation for developing tailored heuristics. GCNNs trained on various problem instances and their optimal solutions could formulate decision rules or predictive models for different aspects of the optimization process, such as relaxation selection, heuristic calls, cut selection, presolving techniques, and node selection.

For example, in the branch and bound algorithm, the choice of relaxation affects the efficiency of the overall problem-solving process. GCNNs could optimize this balance, enhancing the algorithm's efficiency in non-linear problems. Similarly, in the context of cutting planes, GCNNs could be used to improve the selection process and enhance problem-solving efficiency.

Node selection and presolving techniques are other areas where GCNNs could have a significant impact. Machine learning could be used to fine-tune the presolving phase, accelerating the resolution phase algorithms. In node selection, efficient decision-making is key to algorithmic success. GCNNs could be trained to make these decisions, potentially enhancing the best-estimate search's efficiency.

In the broader landscape of power system optimization, the significance of alternative objectives for the UC problem cannot be overstated. While the current research primarily focused on economic dispatch, the underlying methodologies and advancements offer a robust foundation for exploring these alternate avenues. For instance, the imminent environmental challenges and global mandates have rendered the emission minimization objective a priority. By modifying our optimization paradigm, emission factors for each unit can be seamlessly incorporated. This can be further bolstered by the predictive capabilities of GCNNs, which can forecast and consequently curtail emissions based on diverse parameters like historical patterns and weather data. Concurrently, the global energy landscape is witnessing an exponential surge in renewable sources, necessitating modifications to the UC, fostering greater renewable integration. The adaptability of our GCNN models makes them prime candidates for discerning the inherent variability and predictability of renewables like wind and solar, leading to their optimized inclusion. On a similar note, the cardinal importance of power system reliability can be catered to by extending our framework to spotlight units that elevate system resilience. Here again, the role of GCNNs is pivotal, enabling them to detect patterns or anomalies that could potentially compromise reliability, and hence preemptively addressing them. Collectively, these alternate objectives, when integrated into our research's fabric, pave the way for a more comprehensive, efficient, and resilient power system optimization approach.

The essence of this research, when viewed through the lens of alternative UC objectives, presents a realm of multi-faceted optimization possibilities. The crux lies in the harmonious integration of these diverse objectives to achieve a well-rounded unit commitment solution.

The groundwork laid by our research can be expanded upon to devise a multi-objective framework. Such a framework would entail the judicious balance of objectives by embedding variable weights, ensuring that considerations spanning cost, emissions, renewable integrations, and reliability are harmoniously addressed. This equilibrium can be dynamically maintained with the aid of GCNNs, as they're equipped to adjust these weights in response to real-time data fluctuations and evolving system conditions. Furthermore, GCNNs, with their vast predictive capabilities, stand to play a transformative role in this expanded UC landscape. By training them on a rich mosaic of datasets, encompassing diverse facets like costs, emissions, and renewable outputs, these networks can offer insights, leading to more effective decision-making across the entire spectrum of UC objectives. In essence, while the expansion to alternative objectives augments the complexity of the problem, the synergistic approach anchored by the foundational work on UC, coupled with the potential of GCNNs, promises a holistic and agile solution to the multifarious challenges of power system optimization.

Considering starting and shutdown costs into the UC optimization model may also offers a more realistic representation of operational challenges in power systems. Starting a generating unit not only involves fuel costs but also incurs wear-and-tear, thermal stress on machinery, and potentially reduces the unit's lifespan. By accounting for these starting and shutdown costs, the optimization algorithm can make more informed decisions about when and how frequently to start or stop a unit, leading to more economically efficient and technically sound operational schedules. It also enhances the financial predictability for plant operators, allowing for better budgeting and maintenance planning. This granularity in modelling can contribute significantly to both the accuracy and the practical applicability of UC solutions.

In conclusion, the key directions outlined for future work, such as enhancing machine learning model specificity and generality, improving graph representation, and broadening the application of GCNNs, hold potential for advancing power system optimization. They may allow for improved computational speed, a deeper understanding of problem structures, and the development of fast and reliable mixed-integer optimization solvers. The exploration of these strategies is in its early stages and presents significant challenges. Nevertheless, these potential advancements may contribute to the development of more efficient and adaptable power system optimization techniques. We look forward to the progress that future research in these areas may bring.

REFERENCES

[1] P. Maisonneuve and A. Lesage-Landry, "Learning-accelerated exact second-order cone programming for unit commitment," *INFORMS Journal on Optimization*, 2023, submitted.

[2] S. Boyd and J. Mattingley, "Branch and bound methods," *Notes for EE364b, Stanford University*, vol. 2006, p. 07, 2007.

[3] E. L. Lawler and D. E. Wood, "Branch-and-bound methods: A survey," *Operations research*, vol. 14, no. 4, pp. 699–719, 1966.

[4] M. Gasse, D. Chételat, N. Ferroni, L. Charlin, and A. Lodi, "Exact combinatorial optimization with graph convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[5] E. B. Khalil, B. Dilkina, G. L. Nemhauser, S. Ahmed, and Y. Shao, "Learning to run heuristics in tree search." in *IJCAI*, 2017, pp. 659–666.

[6] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: a methodological tour d'horizon," *European Journal of Operational Research*, vol. 290, no. 2, pp. 405–421, 2021.

[7] P. Bonami, A. Lodi, and G. Zarpellon, "Learning a classification of mixed-integer quadratic programming problems," in *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Springer, 2018, pp. 595–604.

[8] C. Cotta and J. M. Troya, "Embedding branch and bound within evolutionary algorithms," *Applied Intelligence*, vol. 18, no. 2, pp. 137–153, 2003.

[9] F. Glover, "Tabu search: A tutorial," *Interfaces*, vol. 20, no. 4, pp. 74–94, 1990.

[10] T. Berthold, S. Heinz, M. Pfetsch, and S. Vigerske, "Large neighborhood search beyond mip," pp. 51–60, 01 2011.

[11] A. Chmiela, E. Khalil, A. Gleixner, A. Lodi, and S. Pokutta, "Learning to schedule heuristics in branch and bound," in *Advances in Neural Information Processing Systems*, vol. 34. Curran Associates, Inc., 2021, pp. 24 235–24 246.

[12] J. E. Mitchell, "Branch-and-cut algorithms for combinatorial optimization problems," *Handbook of applied optimization*, vol. 1, no. 1, pp. 65–77, 2002.

[13] A. Schrijver, "On cutting planes," *Combinatorics*, vol. 79, pp. 291–296, 1980.

[14] R. Baltean-Lugojan, P. Bonami, R. Misener, and A. Tramontani, "Selecting cutting planes for quadratic semidefinite outer-approximation via trained neural networks," *Optimization-Online*, 2018.

[15] Y. Tang, S. Agrawal, and Y. Faenza, "Reinforcement learning for integer programming: Learning to cut," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 13–18 Jul 2020, pp. 9367–9376.

[16] H. He, H. Daume III, and J. M. Eisner, "Learning to search in branch and bound algorithms," in *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[17] J. Song, R. Lanka, A. Zhao, Y. Yue, and M. Ono, "Learning to search via self-imitation," *CoRR*, vol. abs/1804.00846, 2018.

[18] M. Bénichou, J.-M. Gauthier, P. Girodet, G. Hentges, G. Ribière, and O. Vincent, "Experiments in mixed-integer linear programming," *Mathematical Programming*, vol. 1, no. 1, pp. 76–94, 1971.

[19] D. Applegate, R. Bixby, V. Chvátal, and W. Cook, *Finding cuts in the TSP (A preliminary report)*. Citeseer, 1995, vol. 95.

[20] A. M. Alvarez, Q. Louveaux, and L. Wehenkel, "A machine learning-based approximation of strong branching," *INFORMS Journal on Computing*, vol. 29, no. 1, pp. 185–195, 2017.

[21] T. Achterberg and T. Berthold, "Hybrid branching," in *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Springer, 2009, pp. 309–311.

[22] T. Achterberg, T. Koch, and A. Martin, "Branching rules revisited," *Operations Research Letters*, vol. 33, no. 1, pp. 42–54, 2005.

[23] M. Fischetti and M. Monaci, "Backdoor branching," *INFORMS Journal on Computing*, vol. 25, no. 4, pp. 693–700, 2013.

[24] F. K. Karzan, G. L. Nemhauser, and M. W. Savelsbergh, "Information-based branching schemes for binary linear mixed integer problems," *Mathematical Programming Computation*, vol. 1, no. 4, pp. 249–293, 2009.

[25] M. Fischetti and M. Monaci, "Branching on nonchimerical fractionalities," *Operations Research Letters*, vol. 40, no. 3, pp. 159–164, 2012.

[26] J. Patel and J. W. Chinneck, "Active-constraint variable ordering for faster feasibility of mixed integer linear programs," *Mathematical Programming*, vol. 110, no. 3, pp. 445–474, 2007.

[27] T. Berthold and D. Salvagnin, "Cloud branching," in *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Springer, 2013, pp. 28–43.

[28] D. Pisinger and S. Ropke, "Large neighborhood search," *Handbook of metaheuristics*, pp. 99–127, 2019.

[29] ravichandra addanki, V. Nair, and M. Alizadeh, "Neural large neighborhood search," in *Learning Meets Combinatorial Algorithms at NeurIPS2020*, 2020.

[30] J. Song, Y. Yue, B. Dilkina *et al.*, "A general large neighborhood search framework for solving integer linear programs," *Advances in Neural Information Processing Systems*, vol. 33, pp. 20 012–20 023, 2020.

[31] A. Hottung and K. Tierney, "Neural large neighborhood search for the capacitated vehicle routing problem," *arXiv preprint arXiv:1911.09539*, 2019.

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

[33] M. Etheve, Z. Alès, C. Bissuel, O. Juan, and S. 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*. Springer, 2020, pp. 176–185.

[34] E. Khalil, P. Le Bodic, L. Song, G. Nemhauser, and B. Dilkina, "Learning to branch in mixed integer programming," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[35] D. Selsam, M. Lamm, B. Bünz, P. Liang, L. de Moura, and D. L. Dill, "Learning a sat solver from single-bit supervision," *arXiv preprint arXiv:1802.03685*, 2018.

[36] P. Gupta, M. Gasse, E. Khalil, P. Mudigonda, A. Lodi, and Y. Bengio, "Hybrid models for learning to branch," *Advances in neural information processing systems*, vol. 33, pp. 18 087–18 097, 2020.

[37] C. Hansknecht, I. Joormann, and S. Stiller, "Cuts, primal heuristics, and learning to branch for the time-dependent traveling salesman problem," *arXiv preprint arXiv:1805.01415*, 2018.

[38] G. Zarpellon, J. Jo, A. Lodi, and Y. Bengio, "Parameterizing branch-and-bound search trees to learn branching policies," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 5, 2021, pp. 3931–3939.

[39] G. Di Liberto, S. Kadioglu, K. Leo, and Y. Malitsky, "Dash: Dynamic approach for switching heuristics," *European Journal of Operational Research*, vol. 248, no. 3, pp. 943–953, 2016.

[40] M.-F. Balcan, T. Dick, T. Sandholm, and E. Vitercik, "Learning to branch," in *International conference on machine learning*. PMLR, 2018, pp. 344–353.

[41] A. Prouvost, J. Dumouchelle, L. Scavuzzo, M. Gasse, D. Chételat, and A. Lodi, "Ecole: A gym-like library for machine learning in combinatorial optimization solvers," *arXiv preprint arXiv:2011.06069*, 2020.

[42] J. A. Taylor, *Convex Optimization of Power Systems*. Cambridge University Press, 2015.

[43] C. Coffrin, H. L. Hijazi, and P. Van Hentenryck, "The QC relaxation: A theoretical and computational study on optimal power flow," *IEEE Transactions on Power Systems*, vol. 31, no. 4, pp. 3008–3018, 2016.

[44] A. Castillo, C. Laird, C. A. Silva-Monroy, J.-P. Watson, and R. P. O'Neill, "The unit commitment problem with ac optimal power flow constraints," *IEEE Transactions on Power Systems*, vol. 31, no. 6, pp. 4853–4866, 2016.

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

[46] G. E. Constante-Flores, A. J. Conejo, and F. Qiu, "AC network-constrained unit commitment via relaxation and decomposition," *IEEE Transactions on Power Systems*, vol. 37, no. 3, pp. 2187–2196, 2022.

[47] P. de Mars and A. O'Sullivan, "Applying reinforcement learning and tree search to the unit commitment problem," *Applied Energy*, vol. 302, p. 117519, 2021.

[48] P. de Mars and A. O'Sullivan, "Reinforcement learning and A* search for the unit commitment problem," *Energy and AI*, vol. 9, p. 100179, 2022.

[49] S. Pineda and J. Morales, "Is learning for the unit commitment problem a low-hanging fruit?" *Electric Power Systems Research*, vol. 207, p. 107851, 2022.

[50] A. S. Xavier, F. Qiu, and S. Ahmed, "Learning to solve large-scale security-constrained unit commitment problems," *INFORMS Journal on Computing*, vol. 33, no. 2, pp. 739–756, 2021. [Online]. Available: https://doi.org/10.1287/ijoc.2020.0976

[51] J. Qin, N. Yu, and Y. 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)*, 2021, pp. 140–145.

[52] A. Venzke, G. Qu, S. Low, and S. Chatzivasileiadis, "Learning optimal power flow: Worst-case guarantees for neural networks," in *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, 2020, pp. 1–7.

[53] P. Van Hentenryck, "Machine learning for optimal power flows," *Tutorials in Operations Research: Emerging Optimization Methods and Modeling Techniques with Applications*, pp. 62–82, 2021.

[54] T. Achterberg, "Scip: solving constraint integer programs," *Mathematical Programming Computation*, vol. 1, no. 1, pp. 1–41, 2009.

[55] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[56] J. Kotary, F. Fioretto, P. V. Hentenryck, and B. Wilder, "End-to-end constrained optimization learning: A survey," *CoRR*, vol. abs/2103.16378, 2021.

[57] M. Glavic, R. Fonteneau, and D. Ernst, "Reinforcement learning for electric power system decision and control: Past considerations and perspectives," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 6918–6927, 2017, 20th IFAC World Congress.

[58] T. Sawa and K. Furukawa, "Unit commitment using quadratic programming and unit decommitment," in *2012 IEEE Power and Energy Society General Meeting*, 2012, pp. 1–6.

[59] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–35, 2017.

[60] Q. Qu, X. Li, Y. Zhou, J. Zeng, M. Yuan, J. Wang, J. Lv, K. Liu, and K. Mao, "An improved reinforcement learning algorithm for learning to branch," *arXiv preprint arXiv:2201.06213*, 2022.

[61] H. Sun, W. Chen, H. Li, and L. Song, "Improving learning to branch via reinforcement learning," in *Learning Meets Combinatorial Algorithms at NeurIPS2020*, 2020.

[62] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2020.

[63] D. R. Morrison, S. H. Jacobson, J. J. Sauppe, and E. C. Sewell, "Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning," *Discrete Optimization*, vol. 19, pp. 79–102, 2016.

[64] P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter, "Branching and bounds tighteningtechniques for non-convex MINLP," *Optimization Methods and Software*, vol. 24, no. 4-5, pp. 597–634, 2009.

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

[66] A. Clauset, "A brief primer on probability distributions," in *Santa Fe Institute*, 2011.

[67] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "Matpower: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Transactions on power systems*, vol. 26, no. 1, pp. 12–19, 2010.

[68] ——, "Matpower: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Transactions on Power Systems*, vol. 26, no. 1, pp. 12–19, 2010.

[69] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2023. [Online]. Available: https://www.gurobi.com