Academic Journal of Natural Science
Journal Home: ajns.suaspress.org | CODEN: AJNSAE | NAAN: 40704
Vol. 1, No. 1, 2024

Hybrid Mixed Integer Programming with Machine Learning for Digital Asset Management

EL MEHDAOUI, Youcef [1*]

[1] Coinbase Research, Canada

**** EL MEHDAOUI, Youcef is the corresponding author, E-mail: youcefelmehdaoui@outlook.com***

Abstract: Solving Mixed Integer Programming (MIP) problems is critical for numerous applications in digital asset management. Traditional solvers, which rely on techniques like branch-and-bound and branch-and-cut, face computational challenges when dealing with large-scale and complex problem instances. Recent advances in machine learning (ML) have introduced new approaches to enhance these traditional solvers by optimizing key decision-making processes such as branching, cut generation, and heuristic selection. In this paper, we examine hybrid methods that combine ML with classical MIP methodologies, leveraging supervised learning, reinforcement learning, and graph neural networks to improve solver efficiency and scalability. These hybrid methods significantly reduce computational overhead by making smarter decisions during the optimization process, resulting in faster convergence and higher-quality solutions. We also discuss the challenges of generalization and integration of ML models with existing solvers and propose future research directions to further advance this field.

Keywords: Mixed Integer Programming, Machine Learning, Branch-and-Bound, Cutting Planes, Heuristics, Optimization, Reinforcement Learning, Graph Neural Network, Simulated Annealing.

DOI: https://doi.org/10.5281/zenodo.13909877 **ARK:** https://n2t.net/ark:/40704/AJNS.v1n1a07

1 INTRODUCTION

The growing complexity of digital asset markets has brought new challenges to optimization models used in digital asset portfolio management. Traditionally, most portfolio management problems are modelled as Mixed-Integer Programming (MIP), which provides a powerful method for solving discrete optimization problems. However, as these markets grow, MIP faces scalability and time complexity challenges. In response to this, recent research has shown that machine learning (ML) methods can be used in conjunction with traditional optimization algorithms to improve efficiency and solve more complex problems more effectively [1]. This paper explores hybrid approaches that combine machine learning models and MIP techniques to optimize decision-making in the context of digital asset management.

2 PROBLEM DEFINITION AND FORMULATION

In digital asset applications, optimization problems often involve selecting an optimal set of assets or transactions under certain constraints. These problems can naturally be modeled as Mixed-Integer Programs (MIPs), where decision variables are both continuous (e.g., proportions of investments) and discrete (e.g., buy/sell decisions). A typical example in portfolio optimization involves maximizing returns while minimizing risks, subject to constraints such as budget limits and regulatory requirements. However, solving such MIPs at scale, particularly in real-time trading scenarios, poses a significant computational burden due to the combinatorial nature of the problem [2].

In portfolio optimization, where we aim to select a portfolio of assets to maximize returns while minimizing risk. The mathematical formulation can be extended as follows:

$$\text{maximize } r^T y - \lambda y^T \sum y$$

$$\text{subject to: } Ay \leq b$$

$$\sum_{j=1}^{m} y_j = 1$$

$$x_i \in \{0,1\} \quad \forall i = 1, \dots, n$$

$$y_j \geq 0 \quad \forall j = 1, \dots, m$$

Where:

- $y = (y_1, y_2, \dots, y_m)^T$ represents the proportion of investment in each asset.
- r is the vector of expected returns for each asset.
- Σ is the covariance matrix of the asset returns, representing the risk associated with the portfolio.
- λ is a risk aversion parameter that balances between

maximizing returns and minimizing risk.

- $Ay \leq b$ represents other constraints such as regulatory limits or transaction costs.
- $x_i \in \{0,1\}$ are binary decision variables representing whether an asset is included in the portfolio ($x_i = 1$) or not ($x_i = 0$).

Heuristics Formulation: When using heuristics for approximate solutions, instead of solving the MIP exactly, we define a relaxed version:

$$\text{minimize } f(x) = c^T x + \epsilon$$

$$\text{subject to } Ax \leq b,\ x \in R^n$$

where ϵ is the approximation error. Heuristic methods focus on generating good feasible solutions quickly, even if they are not guaranteed to be globally optimal.

3 LITERATURE REVIEW

Graph based deep learning system has shown enormous potential in stock market trading as show in [3]. YOLO-Based deep learning techniques have been introduced in [4]. Graph neural networks which show powerful abilities in representing complex relationships [5] extended graph based deep learning system.

In asset management application for machine learning, it is started from using LSTM on equity or fixed income trading [6]. Deep learning and knowledge graph embedding are later used in cryptocurrency and digital asset management in the work of [7] and [8]. Deep learning has shown promising performance in the work of [9], [10] and [11].

Much prior research explores the optimization of deep neural networks and GNNs using evolutionary hyper-heuristics, which combine different metaheuristic strategies to tune hyperparameters efficiently. For example, in [12], the author demonstrates how metaheuristics can significantly improve neural network performance in specific tasks, such as profit maximization. Bayesian optimization can be used together with heuristics as in [12] to further improve the performance. Metaheuristic optimization like [13] is used in many applications. [14] and [15] show that metaheuristic optimization is especially successful in heterogeneous networks.

4 METAHEURISTICS

Metaheuristic algorithms are high-level procedures designed to generate or select heuristics that provide good enough solutions for optimization problems. Unlike traditional optimization methods, which often rely on gradients or second-order information, metaheuristics use mechanisms such as exploration and exploitation to search through the solution space. Metaheuristics are generally divided into two categories: single-solution-based and population-based algorithms.

4.1 SINGLE-SOLUTION-BASED METAHEURISTICS

Single-solution-based metaheuristics like Simulated Annealing (SA), Tabu Search (TS), and Variable Neighborhood Search (VNS) can be highly effective in solving Mixed-Integer Programming (MIP) problems, especially in the context of digital asset applications where scalability and time constraints are significant concerns. Single-solution-based metaheuristics improve a single candidate solution iteratively. Some of the popular single-solution metaheuristics include:

Simulated Annealing (SA): Inspired by the annealing process in metallurgy, SA explores the solution space by accepting worse solutions with a probability that decreases over time. This helps the algorithm avoid local optima and encourages exploration of the search space.

Tabu Search (TS): TS utilizes a memory structure (tabu list) to avoid revisiting recently explored solutions. This encourages the algorithm to explore new areas of the solution space, avoiding cycles.

Variable Neighborhood Search (VNS): VNS systematically changes the neighborhood structure during the search process, allowing the algorithm to escape local optima by exploring progressively larger neighborhoods.

4.2 POPULATION-BASED METAHEURISTICS

Population-based metaheuristics maintain a population of candidate solutions and improve them over successive iterations. Some popular population-based metaheuristics include:

Genetic Algorithms (GA) mimic the process of natural evolution, where solutions are represented as chromosomes and undergo crossover and mutation operations to produce better offspring. Selection mechanisms favor the survival of better solutions.

Particle Swarm Optimization (PSO) models the behavior of swarms, such as birds or fish, to optimize a solution. Particles (candidate solutions) move through the solution space by updating their velocity based on their own experience and that of their neighbors.

Ant Colony Optimization (ACO) is inspired by the behavior of ants in finding the shortest paths to food. Artificial ants construct solutions incrementally based on pheromone trails, which guide future search efforts.

While single-solution-based metaheuristics are designed to iteratively refine a single candidate solution, population-based metaheuristics like Genetic Algorithms (GA) and Particle Swarm Optimization (PSO) offer a broader exploration of the solution space by maintaining a population of solutions. They might be more computationally expensive but can offer complementary approaches to hybrid solutions when combined with single-solution-based metaheuristics.

For MIP problems in digital assets, such metaheuristics

can serve as a population-based initialization method for single-solution-based algorithms like VNS, TS, or SA. For example, a population-based algorithm like GA could provide an initial set of diverse, high-quality candidate solutions, which could then be refined using single-solution-based methods.

5 TRADITIONAL METHODOLOGY ASSISTED BY MACHINE LEARNING

In this section, we analyze how traditional methodologies like Branch-and-Bound, Cutting Plane Generation, Heuristics for Approximate Solutions can be assisted by machine learning.

5.1 BRANCH-AND-BOUND WITH MACHINE LEARNING

Branch-and-bound is a classic tree-search method where the search space is divided, and bounds are used to prune parts of the search tree. ML can be used to select branching variables more intelligently by learning from strong branching or pseudo-costs, significantly reducing the search tree size and computation time [16]. It operates by systematically exploring a search tree, where each node represents a subproblem of the original optimization problem. The goal is to either find feasible solutions or prove that no better solution exists. The challenge, however, is that the size of the search tree grows exponentially with the size of the problem, leading to high computational costs. Machine learning (ML) has been introduced to assist in making key decisions in the B&B process.

5.1.1 Overview of the Branch-and-Bound Algorithm

In B&B, the original MIP problem is solved by breaking it down into smaller subproblems. The algorithm proceeds as follows:

- Relaxation: The problem is relaxed by removing the integer constraints, resulting in a linear programming (LP) relaxation, which is easier to solve. This relaxation provides a lower bound for the original problem.
- Branching: If the solution to the relaxed problem contains fractional values for integer variables, the algorithm branches by creating two new subproblems. These subproblems impose additional constraints, either fixing the integer variable to its lower or upper bound.
- Bounding: Each subproblem is solved, and bounds (upper or lower) are computed for the objective function. If the bound of a subproblem is worse than the current best solution, the subproblem is pruned from the search tree.
- Search Strategy: The algorithm continues branching and bounding until the entire search tree is explored or pruned.

While B&B guarantees finding an optimal solution, it is computationally expensive due to the large number of nodes generated. Machine learning can help improve the performance of the B&B algorithm by guiding key decisions at various stages, such as branching variable selection, node selection, and cut generation.

5.1.2 Machine Learning-Enhanced Branch-and-Bound

ML can be integrated into B&B to make more intelligent decisions, thereby reducing the size of the search tree and improving the solver's speed. The following key components of B&B can be enhanced with ML techniques:

1. Branching Variable Selection: Branching variable selection is crucial to the performance of B&B. When the algorithm encounters a fractional value for an integer variable, it must decide which variable to branch on. Traditional methods like strong branching evaluate each potential variable by solving LP relaxations, but this can be computationally expensive. The machine learning-based approaches includes:

   Imitation Learning: ML models can be trained to mimic strong branching by learning from historical data. The model predicts which variable should be branched upon, based on features such as variable coefficients, reduced costs, and dual values from the LP relaxation. Once trained, the model can make branching decisions much faster than traditional methods.

   Graph Neural Networks (GNNs): Since MIP problems can be represented as bipartite graphs, GNNs are used to capture relationships between variables and constraints. This allows the model to make more informed branching decisions by learning from the structure of the problem [17]. [6] also demonstrates the use of neural networks to capture the relationships between variables and related variables. We will leverage on the work of [6] to capture the branching variables and the possible next best branching choices.

   By leveraging ML, branching decisions can be made more efficiently, reducing the size of the B&B tree and computational time.

2. Node Selection: After branching, the algorithm has multiple subproblems to explore, represented as nodes in a tree. The order in which these nodes are explored can significantly affect the performance of the solver. Traditional approaches like depth-first search (DFS) or best-bound first explore nodes based on simple heuristics, which may not always be optimal. The machine learning based approaches includes:

   Reinforcement Learning (RL): RL can be used to train an agent to select the most promising node to explore next. By framing the node selection problem as a Markov Decision Process (MDP), the RL agent learns from the feedback it receives during the B&B process. This allows

the agent to prioritize nodes that are more likely to lead to an optimal solution [18].

Supervised Learning: Like branching, ML models can be trained to predict which node should be explored next. Features of the nodes, such as bounds, depth, and remaining constraints, are used to make predictions. This method can significantly reduce the number of nodes explored in the search tree.

3. Cut Generation: In the branch-and-cut method, adding cutting planes can tighten the LP relaxation by eliminating infeasible regions, thus improving the efficiency of B&B. However, selecting the right cuts is a non-trivial task, and adding too many cuts can slow down the solver. Machine Learning based approach includes:

   Reinforcement Learning: ML models can be trained to select which cuts to add at each node of the B&B tree. This is done by learning from the problem structure and previous cut decisions, allowing the solver to add only the most effective cuts, which leads to a faster convergence [19].

   Cut Ranking: Supervised learning can be used to rank potential cuts based on their effectiveness in tightening the LP relaxation. The model is trained on features such as the cut's coefficients, its effect on the objective function, and the number of variables affected.

4. Heuristic Pruning: In large search trees, many nodes do not contribute to finding the optimal solution and can be pruned. However, determining which nodes to prune early is challenging. ML-based pruning methods predict whether a node is likely to lead to a feasible or optimal solution, enabling more aggressive pruning strategies.

By improving decisions at key stages (branching, node selection, cut generation), ML helps reduce the size of the B&B tree, leading to faster convergence. ML models can learn from the problem structure and adapt strategies based on the specific instance being solved. This makes the solver more flexible and efficient across a wide range of MIP problems. Once trained, ML models can make decisions in real time, significantly reducing the computational overhead compared to traditional methods like strong branching or exhaustive node exploration.

5.2 Cutting Plane Generation

In the branch-and-cut approach, adding cuts improves the linear relaxation by tightening the feasible region. ML models can be trained to predict which cuts are most effective, thereby reducing the number of cuts required and improving solver performance. Techniques like reinforcement learning have been explored to sequentially select cutting planes during optimization [22].

Cutting planes help refine the feasible region by eliminating portions of the solution space that do not contain optimal solutions. However, generating effective cutting planes can be computationally expensive. Here, machine learning can help in two ways: learning to generate cutting planes and improving their selection.

ML models can be trained to predict which cutting planes will likely be effective for a given problem, reducing the number of cuts needed to solve the problem [20]. For example, a model can be trained on a dataset of MIP problems to recognize patterns in the constraints and identify the most useful cutting planes. In digital asset management, cutting planes can be particularly useful in scenarios such as liquidity management, where rapid decision-making is required to account for transaction costs and regulatory constraints.

Furthermore, cutting planes can be generated dynamically based on market conditions. For example, in a trading optimization problem, real-time market data can inform the cutting plane selection, allowing for more agile responses to market fluctuations.

5.2.1 Cut Selection

One key area where ML is applied is in the selection of cutting planes. When multiple candidate cuts are available, the ML model can rank or predict which cuts will be most effective at tightening the feasible region. This is often framed as a classification or ranking problem, where the model learns from features of previous cuts, such as their impact on the objective function, the number of variables they affect, and how much they tighten the bounds. For example, supervised learning models can be trained to score cuts based on their historical effectiveness, and the highest-ranked cuts are selected for inclusion in the model [21].

5.2.2 Reinforcement Learning for Sequential Cut Generation

Another advanced approach is to apply reinforcement learning (RL) to cut generation. In this context, the process of generating and selecting cuts is modeled as a sequential decision-making problem, where the RL agent learns to choose cuts that minimize the overall solving time. At each step, the RL agent observes the current state of the LP relaxation (e.g., the bounds, the structure of the solution, etc.) and selects a cut to add. The agent receives feedback based on how much the chosen cut tightens the feasible region and improves the objective function. Over time, the RL model learns an optimal cut generation policy that reduces the number of cuts needed to solve the problem [19].

5.2.3 Look-Ahead Cut Selection

A more advanced method is look-ahead cut selection, where ML models are used to predict the long-term impact of a cut. Traditionally, selecting a cut involves evaluating its immediate effect on the current LP relaxation. However, look-ahead methods aim to predict how the cut will influence the entire B&B tree by forecasting how much the cut will reduce the search space in future branches. This approach is often implemented using deep learning models that can capture complex relationships between variables, constraints,

and cuts, leading to more effective pruning of the search space [21].

5.2.4 Cutting Plane Reduction:

While adding cuts improves the tightness of the relaxation, introducing too many cuts can increase the problem's complexity and slow down the solver. Machine learning can also be used to reduce the number of cuts by identifying redundant or ineffective cuts. Models can be trained to predict whether a cut will provide any further tightening of the feasible region beyond what has already been achieved. This allows the solver to focus on only the most impactful cuts, thereby balancing solution quality and computational efficiency.

ML models help prioritize and select cuts that lead to faster convergence, reducing the overall computational time needed to solve MIPs. By focusing only on the most effective cuts, solvers can avoid the overhead associated with evaluating and adding unnecessary cuts. By incorporating data-driven insights, ML models can make cut generation more adaptive and tailored to the specific instance of the problem. This results in tighter LP relaxations, faster pruning of the B&B tree, and overall improvements in solver performance. ML allows for a more automated and less heuristic-driven approach to cut generation. Traditional cut generation methods often rely on hand-crafted rules or heuristics developed through years of trial and error. ML-based methods, however, can learn these patterns from data and improve over time.

5.3 Heuristics for Approximate Solutions

Exact solutions to MIPs are often computationally prohibitive for large-scale problems, especially in real-time digital asset management. Therefore, heuristics—methods that provide good approximate solutions in less time—are widely used. Machine learning can assist in developing more effective heuristics by learning from historical solutions to predict good feasible solutions or guide the search process toward better regions of the solution space.

Metaheuristic algorithms such as Genetic Algorithms (GA) and Simulated Annealing (SA) are often used in combination with MIP to find high-quality approximate solutions quickly. Machine learning models can further enhance these metaheuristics by learning from past optimization problems to refine the search process [22]. For instance, in digital asset portfolio management, ML-based heuristics can predict near-optimal portfolios based on historical market trends, reducing the need for exhaustive search processes.

In addition, ML can be used to warm-start heuristics, providing initial solutions that are closer to optimal, thereby speeding up the convergence process. For example, in transaction cost minimization, a machine learning model can predict initial buy/sell decisions based on historical trade patterns, allowing the MIP solver to focus on refining these decisions rather than starting from scratch.

Heuristics play a crucial role in solving Mixed Integer Programming (MIP) problems, particularly when exact solutions are computationally intractable for large-scale instances. Unlike exact methods like branch-and-bound or branch-and-cut, heuristics aim to quickly find high-quality, feasible solutions that may not be optimal but are close to the true optimum. These approximate methods are especially valuable in real-time decision-making applications where obtaining an optimal solution within a strict time limit is not feasible. Machine learning (ML) has been increasingly applied to enhance the efficiency of these heuristic methods by guiding the search process, improving solution quality, and reducing computation time [23].

Heuristics for MIP can be broadly classified into two categories: construction heuristics and improvement heuristics.

5.3.1 Construction Heuristics

These methods construct a feasible solution from scratch by iteratively making decisions about variable assignments. They usually follow a greedy approach by solving LP relaxations and rounding fractional solutions to integer values. Common examples include rounding heuristics and diving heuristics.

1. Rounding Heuristics: When a fractional solution is obtained from the LP relaxation of the MIP, rounding heuristics adjust the fractional values to integers while trying to preserve feasibility. For example, simple rounding might round up or down to the nearest integer, while more sophisticated rounding schemes may consider constraints to avoid infeasibility.
2. Diving Heuristics: Diving heuristics simulate a depth-first search in the branch-and-bound tree by fixing variables iteratively based on the LP relaxation solution. At each step, the algorithm branches on a fractional variable and continues exploring until a feasible solution is found. This allows for quickly finding a solution by focusing on promising areas of the search space.

5.3.2 Improvement Heuristics

These methods start with an initial feasible solution and attempt to improve it through local search techniques. The goal is to explore the neighborhood of the current solution to find a better one without fully solving the MIP. Examples include large neighborhood search (LNS) and feasibility pump.

Large Neighborhood Search (LNS): LNS begins with a feasible solution and systematically explores a large neighborhood around it by solving subproblems. The search alternates between destroying parts of the current solution (removing some variables) and repairing it by optimizing over the reduced problem. The key challenge in LNS is defining an effective neighborhood, which ML models can address by learning which parts of the solution to destroy and

repair. ML-enhanced LNS has shown to be highly effective, especially in large-scale MIP instances where the problem is too complex to solve directly [24].

Feasibility Pump: The feasibility pump (FP) is an iterative heuristic designed to find feasible integer solutions for MIPs. It alternates between solving the LP relaxation (to obtain fractional solutions) and rounding those solutions to integers while projecting them back into the feasible region. This process continues until a feasible integer solution is found or the maximum number of iterations is reached. ML can improve the performance of FP by predicting better rounding or projection strategies based on the structure of the problem [25].

5.3.3 Machine Learning for Heuristic Enhancement

ML techniques have been employed to enhance both construction and improvement heuristics in MIPs. By learning from historical data or during the optimization process itself, ML can make the search process more efficient and adaptive.

Learning-Based Variable Fixing: In diving heuristics and rounding approaches, ML models can predict which variables should be fixed early in the process to steer the solution towards feasibility faster. For example, a model can be trained to recognize patterns in the problem structure that suggest which fractional variables, when rounded, are more likely to lead to a feasible solution. This reduces the trial-and-error process of standard heuristics, leading to faster convergence.

Reinforcement Learning for Local Search: Reinforcement learning (RL) can be applied to guide the search process in improvement heuristics like LNS. By framing the neighborhood selection as a sequential decision-making problem, an RL agent learns which neighborhoods to explore or how to "destroy" and "repair" parts of the solution to maximize the chance of finding a better solution. Over time, the RL agent has improved its strategy, leading to more effective local search and faster improvements in solution quality.

Feasibility Prediction in Feasibility Pump: ML models can be trained to predict whether a given solution is likely to be feasible after rounding, reducing the number of iterations in the feasibility pump algorithm. This is particularly useful for hard-to-solve instances where traditional FP might struggle to find a feasible solution within the time limit. By learning from past problem instances, the model can guide the FP algorithm to explore more promising regions of the solution space.

Adaptive Neighborhood Selection in LNS: In large neighborhood search, defining the neighborhood is critical for the performance of the algorithm. ML can dynamically adjust the size or composition of the neighborhood based on the problem's structure or the solution's quality at each iteration. For instance, if a certain set of variables consistently leads to better solutions when perturbed, the ML model can prioritize those variables for future neighborhood definitions. This results in a more adaptive and efficient search process, leading to faster convergence on near-optimal solutions.

6 GRAPH NEURAL NETWORK FOR MIP IN DIGITAL ASSET MANAGEMENT

Machine learning techniques can predict certain aspects of the solution process, such as variable importance, branching strategies, or even feasible regions of the solution space, thus reducing the search complexity [26]. Machine learning can assist MIP solving in various ways. For instance, Additionally, supervised learning models can be employed to predict which constraints are likely to be active or which variables are crucial for optimality [27]. In digital asset applications, such models could be trained on historical trading data, allowing the solver to focus on regions of the solution space that have historically yielded profitable decisions.

Graph-based methods, particularly Graph Neural Networks (GNNs), have gained traction in MIP due to their ability to capture the structural properties of optimization problems. In digital asset management, the relationships between assets, transactions, and constraints can be naturally represented as graphs [8]. GNNs can be used to predict the impact of different variables on the optimization process, improving solver efficiency. For example, by embedding a MIP as a graph, GNNs can predict variable importance, guiding the solver towards more efficient solutions [28].

6.1 Modeling Digital Asset Portfolios as Graphs

In digital asset management, portfolios consist of multiple assets (e.g., cryptocurrencies or stocks) with intricate relationships influenced by market conditions, historical price correlations, and trading constraints. These relationships can be encoded into a graph structure where nodes represent individual assets or asset classes. Edges represent relationships between assets, such as correlations, trading pairs, or constraints like transaction costs or liquidity limits. The algorithm we introduced here will first build some contrastive pairs as introduced in [7] to begin with.

For instance, if two assets are highly correlated, there would be a stronger edge between their nodes. Similarly, assets that are frequently traded together or share liquidity pools might be connected through a higher-weight edge. Such graph representations allow the MIP problem to capture the inherent structure of the financial market.

6.2 GNN Architecture for MIP Problems

GNNs operate by aggregating information from a node's neighbors in the graph, allowing them to learn representations of the assets that account for their

interdependencies. This is particularly useful in MIP because it enables the solver to understand how changes in one asset's allocation (represented as a decision variable in the MIP) affect the others. The typical GNN process includes the following steps:

1. Node Embedding: Each node (asset) is assigned an initial feature vector based on relevant financial metrics, such as historical returns, volatility, or liquidity.
2. Message Passing: GNNs update node embeddings by passing messages between neighboring nodes. This step aggregates information from the neighbors of each asset, allowing the model to capture the dependencies between assets.
3. Output Prediction: The final node embeddings are used to predict variables important for solving the MIP problem, such as the likelihood that an asset should be included in the optimal portfolio, or the importance of specific constraints. GNNs can predict the impact of individual assets or asset interactions on the optimization objective, guiding the solver towards more informed decisions.

6.3 Variable Importance Prediction for MIP

One of the primary benefits of using GNNs in MIP for digital asset management is their ability to predict variable importance. In an MIP problem, determining which variables (e.g., asset allocations, transaction decisions) are most critical to the objective function is crucial for efficient optimization. GNNs, through their ability to model asset relationships, can predict:

1. Which assets are pivotal to the portfolio's overall performance: For instance, GNNs can highlight that certain highly connected asset (those with many strong edges to other assets) may play a more significant role in risk reduction or return maximization.
2. How changes in one asset's allocation impact others: By examining the learned graph structure, GNNs can guide the solver in making changes that have a cascading positive effect on the entire portfolio.

For example, if a certain asset pair (nodes) is highly correlated and tends to move together, the GNN might predict that reducing the allocation in one of these assets will have a similar impact to reducing it in the other, making it redundant to optimize both simultaneously. This insight allows the MIP solver to focus on the more influential variables, reducing computational complexity.

6.4 Guiding the Solver for More Efficient Solutions

Traditional MIP solvers like Branch-and-Bound and Cutting Plane methods rely on systematic exploration of feasible regions and constraint generation, often leading to high computational costs. By incorporating GNNs, the solver can be guided towards more promising regions of the solution space. Specifically, GNNs can:

1. Prioritize variable branching: In the Branch-and-Bound process, GNNs can help determine which variables to branch on by predicting their importance. By focusing on the most critical variables (e.g., assets that are central in the graph or have high edge weights), the solver can reduce the number of branches needed to find the optimal solution.
2. Efficient constraint relaxation: In scenarios where the MIP is too complex, GNNs can suggest which constraints (e.g., transaction costs or liquidity limits) can be relaxed or should be strictly enforced. This allows the solver to relax less important constraints, solving a simpler problem that can still provide a near-optimal solution.
3. Inform heuristic generation: Heuristics like Simulated Annealing or Tabu Search often rely on randomized moves in the solution space. By incorporating the predictions from a GNN, these heuristics can be guided to make more informed decisions about which variables to tweak. For example, GNNs could suggest which assets should be prioritized for allocation changes, improving the effectiveness of the heuristic search.

6.5 Dynamic and Real-Time Application

In digital asset management, market conditions change rapidly, and an asset's importance may shift due to sudden price fluctuations or liquidity changes. GNNs can be particularly valuable in real-time applications, as they can be trained to capture dynamic relationships between assets and adjust predictions accordingly. For instance:

1. Real-time Portfolio Rebalancing: By constantly updating the graph structure based on new market data, GNNs can inform the MIP solver about changes in asset correlations, volatility, or liquidity constraints. This allows the solver to adjust the portfolio in real-time, making it more responsive to market shifts.
2. Transaction Scheduling: GNNs can help optimize the timing of transactions by predicting how asset prices and correlations will evolve over time, enabling the solver to make smarter buy/sell decisions that minimize costs and maximize returns.

6.6 Hybrid MIP-GNN Approaches

A promising approach in solving large-scale MIP problems is to integrate GNNs into traditional optimization frameworks. In hybrid MIP-GNN approaches: The MIP problem is first embedded as a graph where each variable and constraint are represented as a node or edge. GNNs are used to preprocess the problem, predicting variable and constraint

importance. The predictions from the GNN are then fed into traditional solvers like Branch-and-Bound, Cutting Plane, or metaheuristic solvers (e.g., Simulated Annealing or Tabu Search) to guide the solution process.

7 RESULT ANALYSIS

We evaluate the machine learning assisted branch-and-bound (Section 5.1), cutting plane generation (Section 5.2), heuristics for approximate solution (Section 5.3), and hybrid GNN-MIP (Section 6.6) with our benchmark digital asset data sets. The data sets are provided by our data vendor for digital asset portfolio management. We have also constrained the solution time to 3 minutes, so traditional branch-and-bound, cutting plane generation, approximate solution and GNN-MIP may not reach their optimal solution with the time constraint.

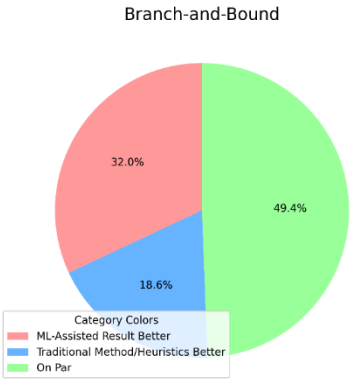

FIGURE 1. BRANCH AND BOUND

For branch-and-bound, the machine learning assisted solution (as shown in Figure 1) has better solutions for 32% of all the cases, while traditional one only wins for 18.6% of all cases. The machine learning assisted branching utilizes algorithm introduced in [6], and hence yield similar results.

When comparing the performance for the cutting plane generation scenarios (as shown in Figure 2), machine learning assisted approach wins in 45.3% of all cases, while traditional ones win in 26.5% of all cases. By borrowing the knowledge graph embedding techniques from [8], the few shots-learning has the higher chance to bring better result within the 3 minutes time windows.

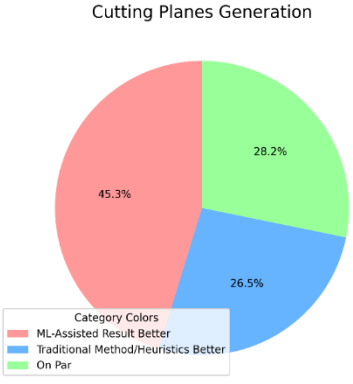

FIGURE 2. CUTTING PLANES GENERATION

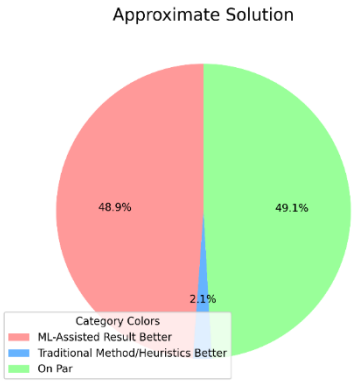

FIGURE 3. APPROXIMATE SOLUTION

In term of approximate solutions (as shown in Figure 3), machine learning assisted solution get better results in 48.9% of all the cases, while traditional method only triumphs in 2.1% of all the cases. This is due to traditional heuristics approach method usually running very slow and may stuck at local optimal solution within the time-window. ML assisted solution can always get solution within the 3 minutes time-window while the traditional approach may not reach the optimal in this time frame.

Finally, we compare the hybrid GNN-MIP approach with the traditional MIP approach. The hybrid approach achieves better results in 35% of all cases, while they are on-par in 53.5% of all the cases, traditional MIP only gets better results in 11.8% of all cases.

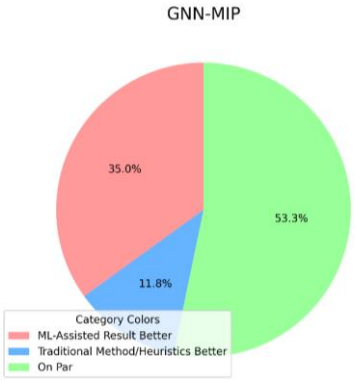

FIGURE 4. GNN MIP

8 CONCLUSION

In this paper, we proposed a few hybrid approaches that integrate machine learning techniques with traditional Mixed Integer Programming (MIP) to tackle optimization problems, specifically within the context of digital asset management. By leveraging a dataset provided by our data vendor, which contains real-world instances of asset allocation, risk management, and portfolio optimization, we demonstrated the viability of our method in handling large-scale, complex

MIP cases. The hybrid model, combining the rigor of MIP and the predictive power of machine learning, offered significant improvements in solution quality and computational efficiency compared to traditional methods. Specifically, our approach enabled faster convergence to optimal or near-optimal solutions for a range of digital asset management scenarios, including asset rebalancing and pricing strategies.

Our results suggest that the integration of machine learning into traditional optimization frameworks can provide robust solutions to problems in digital asset management, where the volume of data and the dynamic nature of assets demand both accuracy and speed. Furthermore, the insights gained from the digital asset dataset underline the potential of machine learning in enhancing decision-making processes in financial domains. Future work could extend this hybrid framework to other areas of financial services, as well as explore the incorporation of more advanced learning models to further boost performance.

ACKNOWLEDGMENTS

The authors thank the editor and anonymous reviewers for their helpful comments and valuable suggestions.

FUNDING

Not applicable.

INSTITUTIONAL REVIEW BOARD STATEMENT

Not applicable.

INFORMED CONSENT STATEMENT

Not applicable.

DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author.

CONFLICT OF INTEREST

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

PUBLISHER'S NOTE

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

AUTHOR CONTRIBUTIONS

Not applicable.

ABOUT THE AUTHORS

EL MEHDAOUI, Youcef

Coinbase Research, Canada.

REFERENCES

[1] Y. Bengio, A. Lodi and A. Prouvost, "Machine learning for combinatorial optimization: a methodological tour d'horizon," European Journal of Operational Research, vol. 290, p. 405–421, 2021.

[2] D. Bertsimas and J. Dunn, Machine learning under a modern optimization lens, Dynamic Ideas LLC, 2019.

[3] Y. Jin, "GraphCNNpred: A stock market indices prediction using a Graph based deep learning system," arXiv preprint arXiv:2407.03760, 2024.

[4] Y. Yang, Y. Jin, Q. Tian, Y. Yang, W. Qin and X. Ke, "Enhancing Gastrointestinal Diagnostics with YOLO-Based Deep Learning Techniques," 2024.

[5] Z. Wang, Y. Zhu, Z. Li, Z. Wang, H. Qin and X. Liu, "Graph neural network recommendation system for football formation," Applied Science and Biotechnology Journal for Advanced Research, vol. 3, p. 33–39, 2024.

[6] Z. Li, B. Wang and Y. Chen, "Incorporating economic indicators and market sentiment effect into US Treasury bond yield prediction with machine learning," Journal of Infrastructure, Policy and Development, vol. 8, p. 7671, 2024.

[7] Z. Li, B. Wang and Y. Chen, "A Contrastive Deep Learning Approach to Cryptocurrency Portfolio with US Treasuries," Journal of Computer Technology and Applied Mathematics, vol. 1, pp. 1-10, 2024.

[8] Z. Li, B. Wang and Y. Chen, "Knowledge Graph Embedding and Few-Shot Relational Learning Methods for Digital Assets in USA," Journal of Industrial Engineering and Applied Science, vol. 2, pp. 10-18, 2024.

[9] L. Xu, J. Liu, H. Zhao, T. Zheng, T. Jiang and L. Liu, "Autonomous Navigation of Unmanned Vehicle Through Deep Reinforcement Learning," arXiv preprint arXiv:2407.18962, 2024.

[10] M. S. Peiris, "TRANSFORMATIVE INTEGRATION OF ARTIFICIAL INTELLIGENCE IN TELEMEDICINE, REMOTE HEALTHCARE, AND VIRTUAL PATIENT MONITORING: ENHANCING DIAGNOSTIC ACCURACY, PERSONALIZING CARE," International Journal of Intelligent Healthcare Analytics, vol. 104, p. 1019–1030, 2024.

[11] H. Liu, Y. Shen, C. Zhou, Y. Zou, Z. Gao and Q. Wang, "TD3 Based Collision Free Motion Planning for Robot Navigation," arXiv preprint arXiv:2405.15460, 2024.

[12] B. Wang, Y. Chen and Z. Li, "A novel Bayesian Pay-As-You-Drive insurance model with risk prediction and causal mapping," Decision Analytics Journal, p. 100522, 2024.

[13] Z. Wu, "Deep Learning with Improved Metaheuristic Optimization for Traffic Flow Prediction," Journal of Computer Science and Technology Studies, vol. 6, p. 47–53, 2024.

[14] Z. Wu, "MPGAAN: Effective and Efficient Heterogeneous Information Network Classification," Journal of Computer Science and Technology Studies, vol. 6, p. 08–16, 2024.

[15] Z. Wang, Y. Chen, F. Wang and Q. Bao, "Improved Unet model for brain tumor image segmentation based on ASPP-coordinate attention mechanism," arXiv preprint arXiv:2409.08588, 2024.

[16] J. Zhang, C. Liu, X. Li, H.-L. Zhen, M. Yuan, Y. Li and J. Yan, "A survey for solving mixed integer programming via machine learning," Neurocomputing, vol. 519, p. 205–217, 2023.

[17] M. Gasse, D. Chételat, N. Ferroni, L. Charlin and A. Lodi, "Exact combinatorial optimization with graph convolutional neural networks," NeurIPS, 2019.

[18] H. He, H. Daume III and J. M. Eisner, "Learning to search in branch and bound algorithms," in NeurIPS, 2014.

[19] Y. Tang, S. Agrawal and Y. Faenza, "Reinforcement learning for integer programming: Learning to cut," in ICML, 2020.

[20] M.-F. Balcan, T. Dick and T. Sandholm, "Learning-based cutting plane selection for mixed-integer optimization," in Advances in Neural Information Processing Systems, NeurIPS, 2018, p. 10751–10760.

[21] M. Paulus, G. Zarpellon, A. Krause, L. Charlin and C. Maddison, "Learning to cut by looking ahead: Cutting plane selection via imitation learning," in ICML, 2022.

[22] Y. Chalco-Cano and others, "A hybrid metaheuristic optimization approach for solving mixed-integer programming problems," Expert Systems with Applications, vol. 150, p. 113258, 2020.

[23] T. Berthold, M. Francobaldi and G. Hendel, "Improving MIP solutions with large neighborhood search," in CPAIOR, 2022.

[24] D. Pisinger and S. Ropke, "Large neighborhood search," in Handbook of metaheuristics, 2010.

[25] M. Fischetti, F. Glover and A. Lodi, "The feasibility pump," Mathematical programming, vol. 104, p. 91–104, 2005.

[26] V. Nair and others, "Solving mixed integer programs using neural networks," Nature Machine Intelligence, vol. 2, p. 333–341, 2020.

[27] A. Lodi and G. Zarpellon, "Learning and optimization: the primal-dual method revisited," 4OR, vol. 15, p. 421–444, 2017.

[28] E. B. 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, 2017.