Solving the Minimum Vertex Cover Problem: A Physics-Inspired GNN Approach

Final Report – Undergraduate Thesis

Nour Fayadh

CS4490Z

Department of Computer Science

Western University

Apr 11, 2025

Prof. Roberto Solis-Oba, Dept. of Computer Science

Prof. Nazim Madhavji, Dept. of Computer Science.

**Glossary:**

* Graph Neural Networks (GNNs)
* Minimum Vertex Cover (MVC)
* Deep Graph Library (DGL)
* Combinatorial Optimization (CO)
* Maximum Independent Set (MIS)
* Maximum Cut (MaxCut)
* Quadratic Unconstrained Binary Optimization (QUBO)
* Graph Convolutional Network (GCN)

**Structured Abstract:**

* **Context and motivation**: The Minimum Vertex Cover (MVC) problem is an NP-hard problem that searches for the smallest subset of vertices covering all edges in the graph. It has applications in network security, bioinformatics, transport networks, etc. Finding exact solutions is computationally infeasible for large graphs, making approximation methods needed. Recent studies suggest graph neural networks (GNNs) can efficiently approximate solutions. This research aims to modify an existing physics-inspired GNN framework to solve the MVC problem efficiently.
* **Research Question:** How can physics-inspired Graph Neural Networks (GNNs) be leveraged to efficiently solve the Minimum Vertex Cover problem and how does its performance compare to traditional algorithms?
* **Principal ideas:** This research adapts a physics-inspired GNN to solve the MVC problem by leveraging node embeddings and recursive neighborhood aggregation to predict vertex covers. It incorporates problem-specific constraints and projection schemes to map soft assignments to binary solutions.
* **Research Methodology:** This research leverages a physics-inspired GNN for the MVC problem by modifying the loss function through changes to the QUBO matrix, effectively redefining the optimization landscape the model learns to minimize. Training is conducted on synthetic graphs generated using random models, with evaluation focusing on accuracy, runtime, and scalability. The implementation utilizes PyTorch Geometric, the Deep Graph Library (DGL), and potentially distributed computing for handling large-scale instances.
* **Anticipated type of results:** The anticipated result is a modified GNN framework that solves the MVC problem with competitive accuracy and better scalability than traditional techniques.
* **Anticipated Novelty:** The novelty lies in showing that the physic-inspired GNN framework can be adapted to solve many combinatorial problems, not just those explored in the paper.
* **Anticipated impact of results:** This research could develop a scalable solution for MVC with possible applications for other combinatorial optimization problems. On practice, it could design better solutions for real-world network and scheduling issues.
* **Limitations:** Limitations include reliance on high-quality datasets and challenges in scaling for large graphs.

**Table of Contents:**

1. Introduction
2. Background and Related Work
3. Research Objectives
4. Methodology
5. Results
6. Discussion
7. Conclusions
8. Future Work and Lessons Learnt
9. Acknowledgments
10. Appendix
11. References
12. **Introduction:**

The MVC problem is a foundational NP-hard problem in combinatorial optimization that involves identifying the smallest subset of vertices in an undirected graph such that every edge is incident to at least one vertex in the subset (Cormen et al., 2009). Due to its computational intractability, especially on large graphs, a range of solution approaches have been explored, including heuristics (Cai et al., 2013), exact methods (Singh et al., 2018), and, more recently, GNNs (Langedal et al., 2022; Li et al., 2018). While GNNs have been applied to the MVC problem in prior work, applying physics-inspired GNNs to MVC has not yet been studied extensively.

This thesis focuses on adapting a physics-inspired GNN framework originally proposed by Schuetz et al. (2022) for other NP-hard problems to solve the MVC problem. The goal is to evaluate the model’s ability to learn meaningful structural representations that can guide it toward competitive approximations, particularly when compared to well-established baselines like the 2-approximation algorithm and a heuristic by Kettani et al. (2013). The study targets scalability, accuracy, and feasibility as key evaluation criteria.

The key research questions addressed are:

* Can a physics-inspired GNN model be adapted to solve the MVC problem with competitive accuracy?
* How does its performance compare to classical algorithms across different graph sizes and densities?
* What are the practical and computational limitations of this approach?

This project demonstrates that the adapted GNN can achieve approximation ratios around 1.03 - 1.05 on both 3- and 5-degree regular graphs, with higher efficiency on medium graphs. Compared to classical methods, the GNN-based model balances approximation quality and flexibility, despite runtime trade-offs for larger graphs.

The novelty of this work lies in extending a physics-inspired GNN to solve the MVC problem, a problem not previously addressed in the original framework. By incorporating a QUBO-based formulation tailored to MVC and adding a post-processing repair step, the model bridges deep learning and classical combinatorial problem-solving.

The remainder of this report is organized as follows: Section 2 covers background and related work, Section 3 outlines the research objectives, Section 4 describes the methodology, Section 5 presents results, Section 6 discusses findings and limitations, Section 7 concludes the study, and Section 8 proposes future directions.

1. **Background and Related Work:**
2. **Scientific and Technical Developments:**

The MVC of an undirected graph is a subset of vertices, which ensures that for each edge in the graph, at least one of its endpoints is in the subset, with a condition on the subset to be minimal (Cormen et al., 2009). The MVC problem is NP-hard, which means it is computationally challenging to solve for large instances, and it was also one of the 21 problems proven NP-complete by Karp (Langedal et al., 2022). Given its computational complexity, approximation algorithms are needed to approximate solutions close to the optimal one. Studies has shown that it is not possible to approximate the solution to the MVC problem with an approximation ratio less than √2- ε (Lazzarinetti et al., 2024).

1. **Importance and Applications of MVC:**

The MVC problem is a fundamental combinatorial optimization (CO) problem with many real-world applications, including network security, scheduling, VLSI design, and industrial machine assignment (Cai et al., 2013). Various approaches have been developed to address the MVC problem, which can be categorized into heuristic and exact methods (Cai et al., 2013). Heuristic methods, such as greedy algorithms and local search, aim to provide good solutions efficiently, but they do not guarantee optimality (Cai et al., 2013). In contrast, exact methods, including branch-and-bound algorithms and Integer Linear Programming, guarantee an optimal solution but are computationally expensive and often impractical for large instances (Cai et al., 2013).

1. **Solution Approaches for MVC:**

Several heuristic approaches have been proposed for solving the MVC problem. Local search algorithms such as NuMVC utilize a two-stage exchange strategy and edge weighting with forgetting (Cai et al., 2013). Another heuristic method, FastVC, employs a fast construction procedure and the Best from Multiple Selections heuristic (Cai et al., 2015). Greedy algorithms, like the one introduced by Chen et al. (2016), integrate Dijkstra’s algorithm to aid in selecting vertices and constructing an approximate vertex cover. An additional heuristic approach would be the one proposed by Kettani et al. (2013), which selects a minimum-degree vertex, adds its neighborhood to the cover, and iteratively expands the solution while minimizing its size.

Exact methods, on the other hand, aim to guarantee optimal solutions regardless of computational cost. One such method is the algorithm introduced by Singh et al. (2018), which follows a straightforward approach by sorting vertices by their degree and iteratively eliminating vertices whose neighbors are all included in the remaining list, ensuring an optimal solution for MVC. Branch-and-bound techniques, such as the algorithm proposed by Langedal et al. (2022), use a simplified branch-and-reduce solver to compute exact solutions for small, connected components when reduction rules cannot be applied. Integer Linear Programming (ILP) is another exact method, which employs a pricing scheme and a density criterion to select sets and analyze their performance (Chandra Chekuri & Lewis Tseng, n.d.).

1. **Graph Neural Networks for MVC:**

GNNs have emerged as a promising approach for solving combinatorial optimization problems, including MVC. GNNs are deep learning models that learn representations of nodes and graphs by aggregating information from their neighbors. GNNs can be used for tasks like node classification, graph classification, and link prediction (Wu et al., 2019). In recent years, GNNs have been used to solve CO problems, including the MVC problem. For example, in the paper by Langedal et al. (2022), the algorithm uses a GNN model to predict which vertex should be included in the vertex cover during the initial vertex cover construction stage if the reduction rules are not applicable. Similarly, in the paper by Li et al. (2018), the authors solve the MVC problem by converting it to a Maximum Independent Set (MIS) problem, then training a graph convolutional network (GCN) to predict how likely each vertex is to belong to the MIS solution. Moreover, the G-DFL4CO framework presented in (Liu et al., 2024) solves the MVC problem by using a graph neural network to predict the connections in a graph and then uses an optimizer to find a minimal set of nodes that cover all the edges.

1. **Commercial Developments:**

Beyond theoretical studies, the MVC problem has found practical applications in various commercial domains, such as computational biochemistry, where resolving conflicts between sequences in a sample is essential (Applications of Graph Theory, n.d.). This can be addressed using a conflict graph, where vertices represent sequences and edges indicate conflicts. The goal is to minimize the number of sequences removed to eliminate all conflicts, which corresponds to solving the minimum vertex cover problem. This approach has been applied to a specific example of the SNP assembly problem (Applications of Graph Theory, n.d.).

Another notable commercial application of MVC is in network security, where protecting against virus attacks caused by the propagation of stealth worms is critical (Applications of Graph Theory, n.d.). A team of computer scientists, led by Eric Filiol at the Virology and Cryptology Lab, ESAT, and the French Navy, ESCANSIC, utilized the vertex cover problem to design optimal real-time defense strategies (Applications of Graph Theory, n.d.). By identifying a minimum vertex cover in a graph, where vertices represent servers and edges represent connections, they optimized defenses to effectively hinder worm propagation (Applications of Graph Theory, n.d.).

1. **Analysis and Research Gap:**

The literature on solving the MVC problem highlights various approaches, including heuristic algorithms, exact methods, and GNNs. While heuristics like NuMVC (Cai et al., 2013) perform well, they struggle with scalability due to input size constraints. GNNs have been applied to CO, but Langedal et al. (2022) found that simple greedy algorithms can sometimes outperform them, raising concerns about their effectiveness for MVC. Additionally, physics-inspired GNNs have been extensively studied for problems like Maximum Cut and Maximum Independent Set but remain unexplored for MVC (Schuetz et al., 2022).

Scalability is a major focus in GNN research, yet existing methods rely on graph sampling and clustering, which can discard critical structural information. This trade-off between efficiency and graph integrity remains an unresolved challenge. Furthermore, while heuristics and GNNs each offer advantages, their comparative performance across diverse graph instances has not been systematically analyzed.

These gaps indicate a need to adapt physics-inspired GNNs for MVC, develop scalable approaches that preserve graph integrity, and benchmark GNNs rigorously against traditional solvers. This thesis aims to address these challenges by investigating GNN-based models that balance scalability and completeness while systematically evaluating their effectiveness for MVC.

1. **Research Objectives:**
   * **O1:** Comprehensively understand the MVC problem and the traditional algorithms, including heuristic and exact methods.
   * **O2:** Design and implement a physics-inspired GNN framework modified to solve the MVC problem.
   * **O3:** Test the performance of the GNN and compare it with traditional approximation methods using benchmarks such as runtime efficiency, solution quality, and scalability.
   * **O4:** Analyze the effectiveness of the physics-inspired GNN in solving the MVC problem.
   * **O5:** Identify potential limitations and propose possible future research for GNN-based solutions to combinatorial optimization problems.
2. **Methodology**

This study explores the application of Graph Neural Networks (GNNs) to the Minimum Vertex Cover (MVC) problem in combinatorial optimization. The methodology aligns with several objectives, including, O1, O2, O3, O4, and O5.

The experimental datasets consist of randomly generated graphs created using a custom “generate\_graph” function. This study specifically focuses on 3-degree regular and 5-degree regular graphs to evaluate model performance under uniform degree constraints. Graphs are generated with varying sizes and densities to evaluate scalability. Each graph is represented using adjacency matrices and node feature vectors encoding structural attributes like degree and connectivity. Data preprocessing includes normalization and basic feature construction to ensure the model can effectively learn from graph structure.

The model design is adapted from the physics-inspired GNN approach proposed by Schuetz et al. (2022), originally framed for the Maximum Independent Set (MIS) and Maximum Cut problems using a Quadratic Unconstrained Binary Optimization (QUBO) formulation. In this study, the model architecture is preserved but the QUBO formulation is modified to solve the MVC problem, using the matrix representation defined in (Minimum Vertex Cover - OpenQAOA, n.d.). Specifically, the QUBO objective function for MVC is defined as:

C(x) = ∑ xᵢ + P ∑ (1 - xᵢ) (1 - xⱼ), where the first sum is over all vertices i ∈ V and the second is over all edges (i, j) ∈ E.

This cost function penalizes configurations where both endpoints of an edge are excluded from the cover while favoring smaller covers (Minimum Vertex Cover - OpenQAOA, n.d.). This matrix penalizes uncovered edges and rewards smaller vertex covers, effectively encoding the MVC objective.

The GNN architecture is a two-layer Graph Convolutional Network (GCN) with randomly initialized node embeddings. Each layer aggregates information from neighboring nodes, and the final output layer generates soft probabilities of nodes belonging to the vertex cover. These soft probabilities are converted into binary decisions via a sigmoid activation followed by rounding.

Despite binary output, the GNN does not always produce feasible solutions that cover all edges. To address this, a greedy post-processing algorithm is applied to add missing vertices that ensure all edges are covered. A subsequent minimization step prunes unnecessary nodes from the cover, improving solution quality while maintaining feasibility. This approach alongside the GNN design supports objective O2.

The loss function is a differentiable relaxation of the QUBO Hamiltonian:

L(θ) = Σᵢⱼ pᵢ(θ) · Qᵢⱼ · pⱼ(θ), where Q is the QUBO formulation defined above. Training is conducted in an unsupervised manner using PyTorch Geometric and Deep Graph Library (DGL), with the Adam optimizer, dropout for regularization, and early stopping for generalization. Multiple random seeds are used to prevent convergence to local optima.

The GNN's performance is benchmarked against multiple traditional methods (O3): the Kettani et al. (2013) heuristic, and a 2-Approximation Algorithm from CS4445 (Analysis of Algorithms II). To assess solution optimality, IBM CPLEX is used and provides exact solutions for comparison. Evaluation metrics include approximation ratio, runtime, and scalability.

Note on CPLEX Optimality: To ensure practical runtime on large graphs, a time limit was imposed on the CPLEX solver. While CPLEX guarantees a feasible solution within this time, the result may not be globally optimal. The solver outputs an MIP gap, which indicates how close the solution is to the optimal value (IBM CPLEX Optimizer for Z/OS, n.d.). In this analysis, we acknowledge that the time limit may result in slight deviations from optimality, but the gap remained within acceptable bounds, making the results suitable for performance comparison with heuristic and GNN-based approaches. All experiments are conducted on SHARCNET high-performance computing systems.

No human subjects are involved in this research. Ethical practices include responsible computing and open-source compliance. This study contributes to understanding the feasibility and potential of GNNs in solving classical optimization problems (O5).

1. **Results**
2. System Context and Architecture

This project investigates the design and performance of a Graph Neural Network (GNN)-based solver for the classical Minimum Vertex Cover (MVC) problem. The developed model is situated within the broader landscape of scalable graph-based optimization heuristics. It serves as an alternative to exact methods (e.g., ILP solvers) and classical heuristics, providing an approximate but efficient solution mechanism capable of operating on graphs ranging from 10 to 100,000 nodes.

The system is composed of three modular layers:

* Graph Processing Layer – Generates synthetic regular graphs of degree 3 and 5 using NetworkX.
* GNN Architecture – Encodes the graph into a latent space via message passing, followed by binary classification of nodes.
* Evaluation Pipeline – Compares the GNN’s solution to three established baselines:
  + ILP solver via IBM CPLEX (optimal solution)
  + A greedy heuristic algorithm (Kettani et al., 2013)
  + A 2-approximation algorithm (CS4445 curriculum)

My work focuses entirely on the GNN solver subsystem, encompassing:

* Model architecture design
* Loss function construction
* Post-processing for feasibility
* Comprehensive experimental benchmarking and analysis

The system is designed as a modular GNN-based solver that accepts raw graph inputs and internally converts them into QUBO formulations of the Minimum Vertex Cover problem. This design enables compatibility with standard graph representations while leveraging a physics-inspired optimization approach under the hood. (O2)

1. System Design and Technical Implementation
2. Key Requirements

* Provide a valid vertex cover on every input graph
* Achieve approximation ratios ≤ 2
* Scale to graphs with up to 100,000 nodes
* Outperform or match classical baselines

1. Design and Architecture

The architecture consists of the following layers:

* Input Encoding: Raw graphs are generated using NetworkX and converted to DGL graph objects. A QUBO matrix representing the MVC objective is computed for each graph and used exclusively in the loss function to guide learning
* Message Passing Layer: Two layers of GraphConv are applied with ReLU activations and dropout, enabling the model to aggregate neighborhood features.
* Classification Layer: A sigmoid activation produces soft node assignment values  [0,1], which are later binarized and post-processed into valid vertex covers.

A thresholding and repair post-processing step ensures that uncovered edges are handled, maintaining the validity of the cover.

1. Component Interfaces & Quality Attributes

* Portability: Implemented using Python, DGL, and PyTorch
* Performance: On 3-degree graphs, the GNN achieved an average approximation ratio of 1.049 with a runtime of 25.27 seconds on small graphs, and a ratio of 1.039 with 1237.15 seconds on medium graphs. On 5-degree graphs, it achieved ratios of 1.036 (small) and 1.034 (medium), with runtimes of 34.97 and 1799.09 seconds respectively.
* Scalability: The GNN solver can leverage GPU acceleration through DGL and PyTorch, but runtime increases significantly with graph size due to the complexity of QUBO transformation and inference over large node embeddings
* Reliability: Valid cover enforced via post-processing; results are reproducible when using fixed random seeds for graph generation and model initialization

1. Rationale for Design Choices

The GNN model follows a physics-inspired iterative aggregation paradigm based on the approach introduced by Schuetz et al. (2022), which captures structural patterns in graphs using QUBO based optimization. ReLU activations were used to enable non-linear learning and stabilize training. DGL was selected for its support of efficient graph batching and modular GNN construction.

1. System Implementation and Testing

Frameworks and Tools

* DGL/PyTorch: GNN training and evaluation
* NetworkX: Graph generation and manipulation
* IBM CPLEX: Optimal ILP baseline
* Pandas: Storing results in Excel format

New Techniques Introduced

* Node-wise binary classification using soft outputs projected to hard decisions
* Custom loss function approximating MVC objective
* Repair strategy to ensure full edge coverage post-inference

Testing Approach (O3)

* Unit Testing: Verified that soft label generation and post-processing produce valid vertex covers
* Pipeline Testing: Ran the full GNN → Repair → Evaluation pipeline on synthetic graphs to ensure correctness and performance
* Validation: Quantitative comparison with CPLEX for optimality check

1. Empirical Study and Results (O3, O4)

Two families of graphs were evaluated:

* 3-Degree Regular Graphs
* 5-Degree Regular Graphs

Each graph family includes:

* Small graphs: 10–1,000 nodes
* Medium graphs: 1,001–100,000 nodes

Graphs larger than 100,000 nodes were excluded due to memory and compute limitations encountered on the SHARCNET compute nodes, which affected both the GNN (due to large QUBO matrix sizes) and ILP solvers.

Each configuration (GNN, heuristic, approximation, and ILP) was tested on 73 distinct graph instances for each degree category (3-degree and 5-degree), covering graph sizes from 10 to 100,000 nodes. For 3-degree graphs, the GNN achieved a mean approximation ratio of 1.049 on small graphs and 1.039 on medium graphs. For 5-degree graphs, it attained a mean approximation ratio of 1.036 on small graphs and 1.034 on medium graphs. In both cases, the GNN maintained competitive accuracy while runtime increased substantially with graph size, averaging 25.27 seconds on small and 1237.15 seconds on medium 3-degree graphs, and 34.97 seconds on small and 1799.09 seconds on medium 5-degree graphs. These averages were computed across all 73 instances in each category (see Section: Results, 3-Degree and 5-Degree Graph Summary Tables).

In addition to total runtime, this study also records the breakdown between training time and post-processing time for the GNN solver. For 3-degree small graphs, training time averaged 25.44 seconds, while post-processing took 0.07 seconds. For 3-degree medium graphs, training time averaged 324.07 seconds, while post-processing took 614.57 seconds. For 5-degree small graphs, training averaged 33.96 seconds, and post-processing 0.13 seconds. For 5-degree medium graphs, training averaged 279.55 seconds, and post-processing 1073.08 seconds. These separate measurements help isolate the computational overhead introduced by the post-repair strategy and clarify the GNN pipeline’s internal cost distribution.

Overall, these results demonstrate that the GNN achieves competitive accuracy and practical scalability, especially when compared to exact methods, even though runtime becomes increasingly sensitive to post-processing costs on larger graphs.

Note: A time limit was applied to the CPLEX solver to ensure practical runtime. The results include MIP gaps where exact solutions could not be reached within the limit.

**3-Degree Graphs, Summary Table (Averaged across 73 instances)**

|  |  |  |  |
| --- | --- | --- | --- |
| Graph Size | Method | Avg. Approx. Ratio/MIP gap for ILP | Avg. Runtime (s) |
| Small Graphs (10–1,000 nodes): | ILP | 0.03211536 | 123.7315161 |
| Heuristic | 1.021136594 | 0.069479279 |
| Approximate | 1.298757607 | 0.032710782 |
| GNN | 1.049496946 | 25.26963423 |
| GNN Training | Look at GNN Avg. Approx. Ratio | 25.44281053 |
| Post-Processing | Look at GNN Avg. Approx. Ratio | 0.06796332 |
| Medium Graphs (1,001–100,000 nodes): | ILP | 0.09679848 | 792.1533439 |
| Heuristic | 1.000323811 | 820.9894271 |
| Approximate | 1.281852242 | 285.3860113 |
| GNN | 1.039218761 | 1237.1526 |
| GNN Training | Look at GNN Avg. Approx. Ratio | 324.07204 |
| Post-Processing | Look at GNN Avg. Approx. Ratio | 614.5732438 |



Figure 1: 3-degree ILP performance

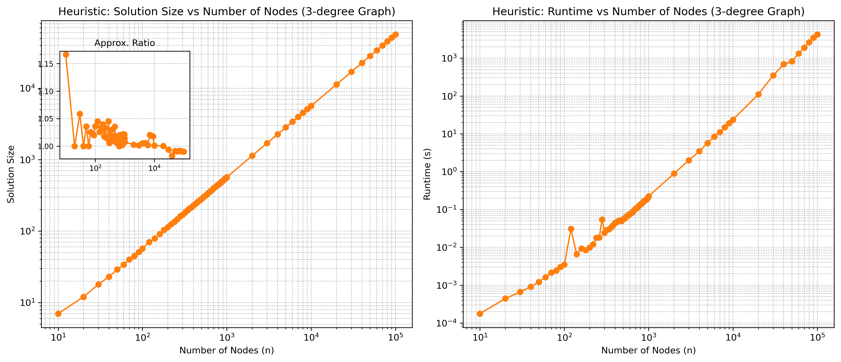
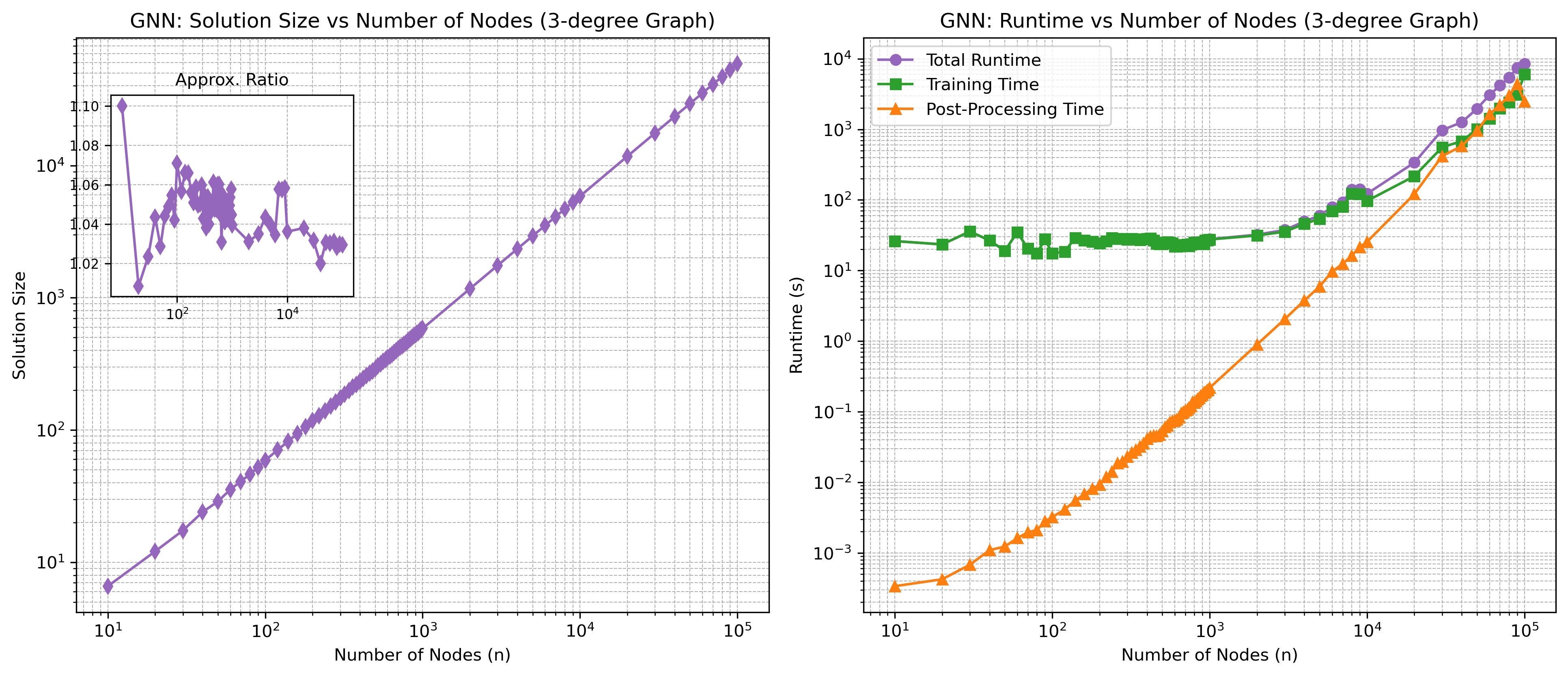


Figure 2: 3-degree Heuristic performance

Figure 3: 3-degree Approximate performance



Figure 4: 3-degree GNN performance



**5-Degree Graphs, Summary Table (Averaged across 73 instances)**

|  |  |  |  |
| --- | --- | --- | --- |
| Graph Size | Method | Avg. Approx. Ratio/MIP gap for ILP | Avg. Runtime (s) |
| Small Graphs (10–1,000 nodes): | ILP | 0.08735034 | 123.1735965 |
| Heuristic | 1.015303786 | 0.090561607 |
| Approximate | 1.275839205 | 0.040503011 |
| GNN | 1.036379805 | 34.97263579 |
| GNN Training | Look at GNN Avg. Approx. Ratio | 33.960472 |
| Post-Processing | Look at GNN Avg. Approx. Ratio | 0.125761249 |
| Medium Graphs (1,001–100,000 nodes): | ILP | 0.191191586 | 703.6414442 |
| Heuristic | 1.006587513 | 864.5926435 |
| Approximate | 1.269497872 | 239.4322917 |
| GNN | 1.034066114 | 1799.094824 |
| GNN Training | Look at GNN Avg. Approx. Ratio | 279.549775 |
| Post-Processing | Look at GNN Avg. Approx. Ratio | 1073.076534 |

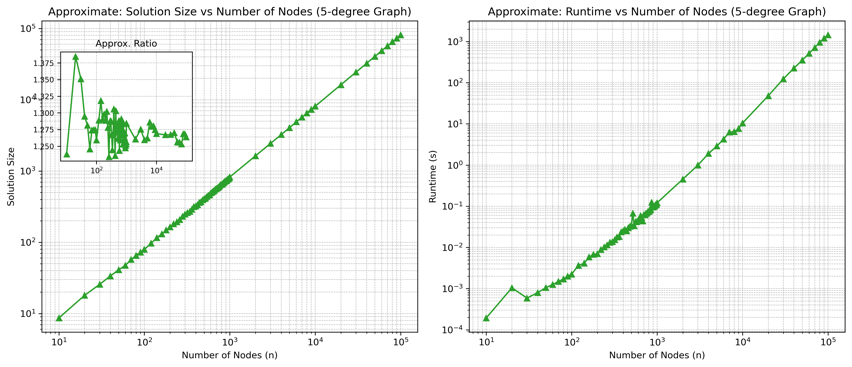
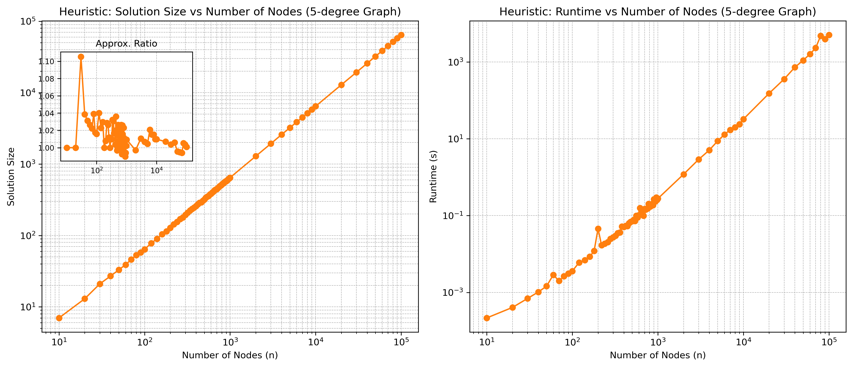
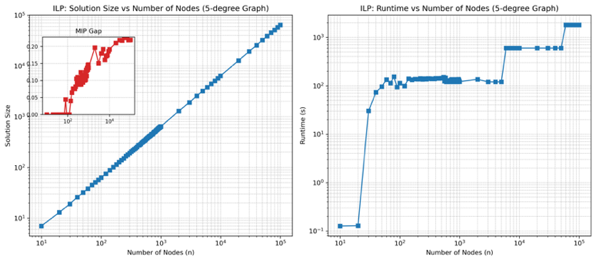
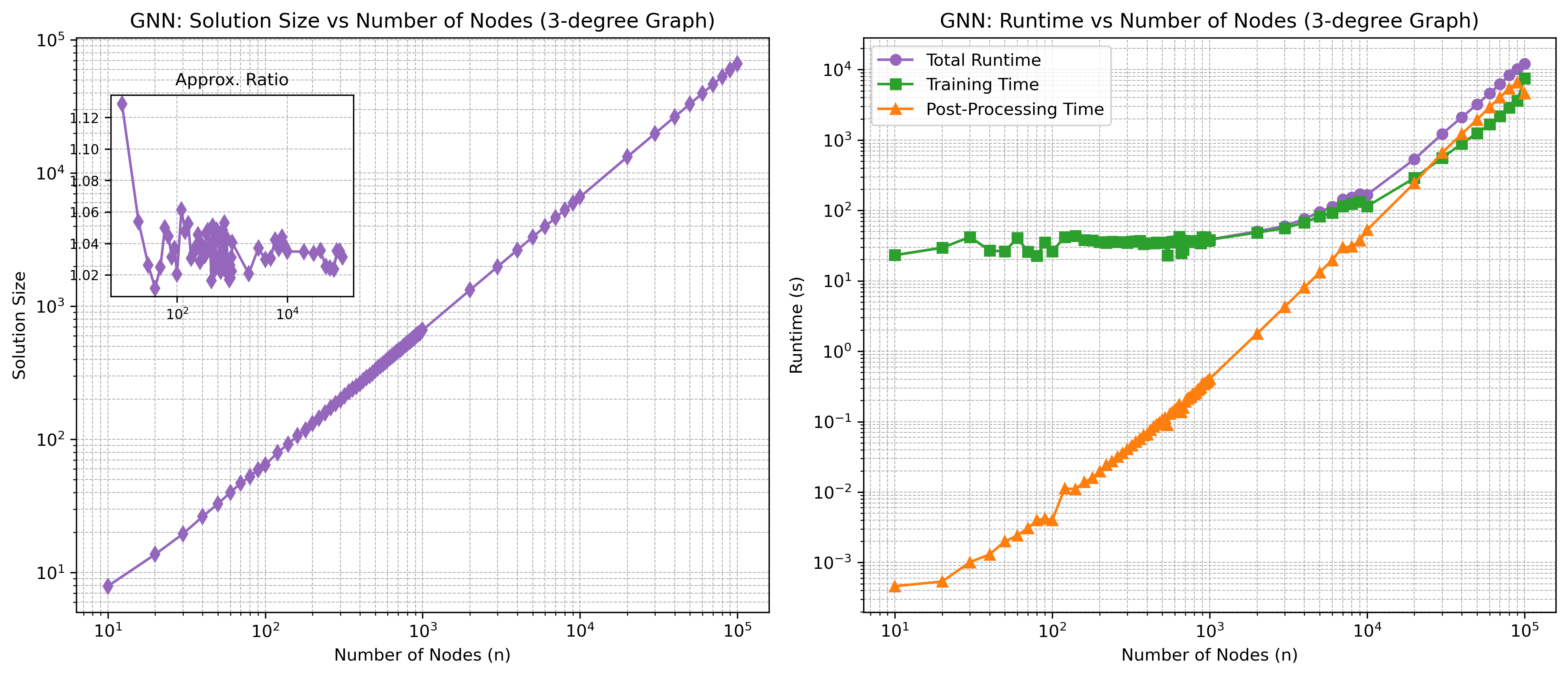


Figure 5: 5-degree ILP performance

Figure 6: 5-degree Heuristic performance

Figure 7: 5-degree Approximate performance

Figure 8: 5-degree GNN performance



Observation: While runtime typically increases with graph size, an exception was observed at 100,000 nodes where post-processing time decreased relative to the 90,000 nodes case. This unexpected dip may reflect improved initial GNN predictions requiring less repair, or differences in system performance during SHARCNET execution. The anomaly is clearly visible in both the 3-degree GNN performance and 5-degree GNN performance plots and is discussed further in Section 6(a).

1. Novelty and Comparative Insight (O4)

The GNN-based approach shows strong solution quality, with runtimes that are competitive for small graphs and significantly higher for medium-sized graphs when compared to ILP. The key advantages include:

* Learns structural patterns from training data, unlike fixed-ratio approximation methods which use deterministic rules
* Maintains feasibility across varied inputs through a dedicated repair step
* The architecture is flexible and, with appropriate adaptation, could potentially be extended to other graph optimization problems

The novelty of this work lies in demonstrating that the physics-inspired GNN framework introduced by Schuetz et al. (2022), which was originally applied to problems like MaxCut and MIS, can be effectively adapted to solve the Minimum Vertex Cover problem with competitive accuracy. This adaptation expands the demonstrated applicability of the framework to a broader class of NP-hard problems.

Compared to prior work, this project successfully:

* Applies the physics-inspired GNN approach to vertex cover for the first time
* Benchmarks on a wide range of graph sizes, including very small graphs (10 nodes)
* Incorporates a QUBO-based label generation strategy tailored for MVC
* Integrates a practical repair strategy post-classification to guarantee solution validity
* Demonstrates robustness across multiple degree regimes (3 and 5)

1. **Discussion**
2. Threats to the Validity of the Results (O5)

Several factors could have influenced the quality and interpretation of the results presented. One key limitation stems from the use of IBM CPLEX as the baseline for optimal solutions. Due to the computational demands of solving ILP formulations on medium-sized graphs, a runtime cap was imposed. In some instances, this led to CPLEX returning feasible solutions with non-zero MIP gaps. While these gaps were generally small, they introduce uncertainty when using CPLEX as a gold standard, particularly in approximation ratio comparisons.

Additionally, although the GNN model consistently produced valid covers after post-processing, there were occasional instances where the soft predictions (prior to repair) did not form a complete vertex cover. The repair step, while effective, adds extra computation and introduces a dependency on heuristic corrections that may vary by graph structure. This highlights a possible vulnerability to structural variation across graph instances.

A further limitation arises from the runtime imbalance between GNN training and post-processing. While post-processing times were minimal for small graphs (e.g., ~0.07 s for 3-degree and ~0.13 s for 5-degree), they became a bottleneck for medium graphs. For example, post-processing on 3-degree medium graphs averaged 614.57 seconds, almost twice the training time of 324.07 seconds, while on 5-degree medium graphs, it reached 1073.08 seconds compared to 279.55 seconds for training. This trend suggests that as graph size increases, the computational cost of enforcing feasibility grows rapidly, which may hinder real-time deployment or large-scale scalability.

Interestingly, while post-processing time generally increased with graph size, an exception was observed at 100,000 nodes, where it decreased relative to the 90,000-node case. This may be due to the GNN producing a more complete initial solution, which reduces the repair burden, or variability in system load across SHARCNET runs. While not typical, this highlights how post-processing cost can be influenced by both solution quality and environmental factors.

Furthermore, graphs were synthetically generated using regular degree structures (3-degree and 5-degree), which may not fully represent real-world graph diversity. This controlled environment was necessary for benchmarking, but it limits generalizability.

1. Implications of the Research Results (O4)

The results demonstrate that physics-inspired GNNs can be adapted beyond their original domain, which were MIS and MaxCut, to solve the Minimum Vertex Cover problem with competitive performance. This opens the door to applying similar GNN-based optimization techniques to a wider range of classical NP-hard problems. From a research perspective, this adaptation extends the relevance of QUBO-based GNN training pipelines. It illustrates the potential of integrating structured problem formulations into neural optimization architectures, reinforcing the connection between combinatorial structure and learning-driven solvers. In practice, the approach has the potential to benefit domains requiring near-optimal solutions in reasonable time on large-scale graphs, such as network design, scheduling, and security modeling. However, runtime and memory bottlenecks must be addressed for deployment in production environments.

1. Limitations of the Results (O5)

The most prominent limitation lies in the scalability of the QUBO construction and GNN inference pipeline. While the model performed well on graphs up to 100,000 nodes, it failed to scale beyond that range due to memory constraints on SHARCNET compute nodes. This limits the immediate applicability of the method to even larger graphs found in industrial settings. Another limitation involves generalization: the model was trained and tested exclusively on synthetic regular graphs. Performance on sparse, irregular, or noisy real-world graphs remains untested. The effectiveness of the model's learned representation in less structured scenarios is still an open question.

1. Generalisability of the Results

The current results are generalizable to regular synthetic graphs of degrees 3 and 5, across sizes up to 100,000 nodes. They offer a strong proof-of-concept that physics-inspired GNNs can model MVC structures. However, their generalization to irregular, weighted, or real-world graphs is not guaranteed.

1. **Conclusions**

This thesis addressed the challenge of solving the Minimum Vertex Cover (MVC) problem using a physics-inspired GNN approach. Building on the framework introduced by Schuetz et al. (2022), the model was adapted to represent the MVC objective through a QUBO formulation, enabling it to learn feasible and high-quality vertex covers across various graph instances.

The main contributions of this research are:

* A modified GNN architecture and training procedure tailored to the MVC problem
* A comprehensive comparison with classical solvers, including CPLEX, a heuristic, and a 2-approximation algorithm
* The demonstration of competitive approximation ratios (∼1.03–1.05) and the practical integration of a repair strategy for solution feasibility

These results validate the adaptability of physics-inspired GNNs for solving MVC and suggest that this class of models can generalize to a broader family of combinatorial problems. Although runtime performance degrades on very large graphs and generalizability to real-world graphs remains uncertain, the approach presents a promising direction for hybrid learning optimization frameworks.

In conclusion, this work contributes both a novel application and a strong empirical foundation for extending physics-inspired GNNs to new problem domains, reaffirming their relevance in scalable combinatorial optimization.

1. **Future Work and Lessons Learnt**
2. Future Work (O5)

While this research demonstrates the applicability of physics-inspired GNNs to the Minimum Vertex Cover (MVC) problem, several directions exist for future exploration:

1. Extension to Irregular and Real-World Graphs: This study was conducted exclusively on synthetic regular graphs (3-degree and 5-degree). Future work should evaluate the model's robustness and performance on sparse, irregular, weighted, or real-world graphs, which often contain noise and structural diversity.
2. Improved Scalability via Efficient QUBO Handling: A key bottleneck identified was the memory consumption of QUBO matrix generation and inference on large graphs. Optimizing this stage or adopting sparse QUBO representations could reduce memory usage and enable training and inference on larger graph instances.
3. Architecture Exploration: This work preserves the architecture proposed by Schuetz et al. (2022). Future research could explore deeper GNN architectures, different message-passing schemes, or hybrid models that combine physics-inspired loss terms with supervised fine-tuning.
4. Integration with Real-Time Decision-Making Systems: The model could be embedded within real-time decision pipelines for domains such as scheduling, cybersecurity, or infrastructure planning. This would require not only performance improvements but also increased interpretability and reliability.
5. Distributed Training and Inference: Leveraging distributed GNN frameworks and cloud-based computing platforms may allow scaling beyond the current 100,000-node graph limit, making the model viable for industrial-scale applications.
6. Lessons Learnt
7. Post-Processing is Crucial for Practical Feasibility: Despite the theoretical formulation, the raw GNN output frequently did not yield valid vertex covers. The use of a repair step was essential to enforce feasibility. This highlights a novel insight: validity in combinatorial problems may require architecture-independent constraints during post-processing.
8. QUBO Encoding Directly Affects Learnability: The structure of the QUBO matrix strongly influenced convergence and model behavior. Small changes in penalty weight or graph topology could shift the optimization landscape significantly. This underscores that QUBO design is not just a pre-processing step, but a tunable hyperparameter space in itself.
9. Heuristic Baselines May Mask Invalid Outputs: Some classical heuristics produced solutions with extremely low approximation ratios, but further inspection revealed that some of these were infeasible. This reinforces that solution quality must be assessed not only by ratio but by strict constraint satisfaction, especially in benchmarking studies.

These lessons offer new insights into applying GNNs to structured optimization and inform future researchers on critical design and evaluation considerations for graph-based learning systems.

1. **Acknowledgments**

I would like to express my heartfelt gratitude to my thesis supervisor, Dr. Solis-Oba, for his continuous guidance, encouragement, and insightful feedback throughout this project. His support played a pivotal role in shaping the direction and depth of my research.

I would also like to thank Dr. Madhavji, the coordinator of the thesis course, for his organization, clear expectations, and the helpful check-ins that kept me on track during the term.

Special thanks to the IBM Academic Initiative for providing access to tools and resources that supported the development and evaluation of this work. I am also grateful to SHARCNET and the Digital Research Alliance of Canada for the computational resources that made the large-scale experiments in this thesis possible.

I would also like to acknowledge the developers and contributors of the open-source libraries used in this project, whose work made this research possible.

Finally, I am deeply thankful to my parents and best friends for being there through it all—their love, encouragement, and belief in me kept me going even during the most challenging moments.

1. **References**

* Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms, third edition*. <http://portal.acm.org/citation.cfm?id=1614191>
* Schuetz, M. J. A., Brubaker, J. K., & Katzgraber, H. G. (2022). Combinatorial optimization with physics-inspired graph neural networks. *Nature Machine Intelligence*, *4*(4), 367–377. <https://doi.org/10.1038/s42256-022-00468-6>
* Liu, Y., Zhou, C., Zhang, P., Pan, S., Li, Z., & Chen, H. (2024). *Decision-focused graph neural networks for combinatorial optimization*. *arXiv preprint arXiv:2406.03647*. <https://doi.org/10.48550/arXiv.2406.03647>
* Chen, J., Kou, L., & Cui, X. (2016). An approximation algorithm for the minimum vertex cover problem. *Procedia Engineering*, *137*, 180–185. <https://doi.org/10.1016/j.proeng.2016.01.248>
* Li, Z., Chen, Q., & Koltun, V. (2018). Combinatorial optimization with graph convolutional networks and guided tree search. *Neural Information Processing Systems*, *31*, 537–546. <http://papers.nips.cc/paper/7335-combinatorial-optimization-with-graph-convolutional-networks-and-guided-tree-search.pdf>
* Langedal, K., University of Bergen, Langguth, J., Simula Research Laboratory, Manne, F., University of Bergen, Schroeder, D. T., & Simula Research Laboratory. (2022). Efficient minimum weight vertex cover heuristics using graph neural networks. *20th International Symposium on Experimental Algorithms (SEA 2022)*, 12:1-12:17. <https://doi.org/10.4230/LIPIcs.SEA.2022.12>
* Singh, S., Singh, G., & Kushwah, N. (2018). Optimal algorithm for solving vertex cover problem in polynomial time. *International Journal of Scientific & Engineering Research*, *9*(8), 1476. <http://www.ijser.org>
* Cai, S., Su, K., Luo, C., & Sattar, A. (2013). NUMVC: an efficient local search algorithm for minimum vertex cover. *Journal of Artificial Intelligence Research*, *46*, 687–716. <https://doi.org/10.1613/jair.3907>
* Cai, S., State Key Laboratory of Computer Science, & Institute of Software, Chinese Academy of Sciences. (2015). Balance between Complexity and Quality: Local Search for Minimum Vertex Cover in Massive Graphs. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)*.
* Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Yu, P. S., Centre for Artificial Intelligence, FEIT, University of Technology Sydney, NSW 2007, Australia, Faculty of Information Technology, Monash University, Clayton, VIC 3800, Australia, & Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607-7053, USA. (2019). A comprehensive survey on graph neural networks. In *JOURNAL OF LATEX CLASS FILES: Vol. XX* (Issue XX, pp. 1–2).
* Lazzarinetti, G., Dondi, R., Manzoni, S., & Zoppis, I. (2024). An Attention-Based method for the minimum vertex cover problem on complex networks. *Algorithms*, *17*(2), 72. <https://doi.org/10.3390/a17020072>
* Chandra Chekuri & Lewis Tseng. (n.d.). *Approximation algorithms*.
* *Applications of graph Theory*. (n.d.). <http://www.dharwadker.org/pirzada/applications/>
* Glover, F., Kochenberger, G., & Du, Y. (2019). Quantum Bridge Analytics I: a tutorial on formulating and using QUBO models. *4OR*, *17*(4), 335–371. <https://doi.org/10.1007/s10288-019-00424-y>
* *Minimum Vertex Cover - OpenQAOA*. (n.d.). <https://openqaoa.entropicalabs.com/problems/minimum-vertex-cover/>
* Kettani, O., Ramdani, F., & Tadili, B. (2013). A heuristic approach for the vertex cover problem. *International Journal of Computer Applications*, *82*(4), 9–11. <https://doi.org/10.5120/14102-2126>
* *IBM CPLEX Optimizer for z/OS*. (n.d.). <https://www.ibm.com/docs/en/cofz/12.9.0?topic=parameters-relative-mip-gap-tolerance&utm_source=chatgpt.com>