Nonogram-Solver: Graph Neural Network Approach

Yejin Cho
Department of Computer Science
University of Southern California
Los Angeles, USA
yejincho@usc.edu

***Abstract*—Nonogram puzzles, popular among a wide range of audiences, offer not only a source of entertainment but also enhance logical reasoning and cognitive abilities. Notably, the value or state of a particular node (pixel) is not solely determined by its own features, but also heavily influenced by the structure of its surrounding nodes. This characteristic fundamentally aligns with the inherent nature of Graph Neural Networks (GNNs), making it an apt approach for tackling Nonogram puzzles. In this study, we frame Nonogram puzzles as a node classification problem, an approach that leverages the inherent structure and characteristics of Nonogram graphs to predict and assign appropriate labels to the nodes. This exploration provides an understanding of the properties of Nonogram puzzles and aids in developing efficient solving strategies, thereby enhancing the overall solving experience.**

***Index Terms*—Nonogram, Graph Neural Network, Node Classification, Grid Puzzle**

I. Introduction

Nonogram, also known as Hanjie, Paint by Numbers, Picross, Griddlers, and Pic-a-Pix, is a puzzle game that involves revealing hidden patterns by deducing and filling in a grid-based board. The essence of Nonogram lies in its ability to represent pictures as graphs composed of nodes and their interconnections, known as edges. This graph representation captures the components of a Nonogram puzzle, where each node corresponds to a pixel and edges represent the relationships between adjacent pixels. Solving a Nonogram puzzle involves assigning colors to each node based on the puzzle rules. However, Nonogram puzzles come with complexities and difficulties that require logical thinking and problem-solving skills to overcome. This is implied by the fact that the nonogram problem is known to be NP-complete [1].

By treating Nonogram as a node classification problem, we aim to leverage the inherent structure and characteristics of Nonogram graphs to predict and assign appropriate labels to the nodes. This approach holds significant potential for various applications, and the outcomes of node classification in Nonogram can provide valuable insights and contribute to the development of advanced solving techniques.

In the following sections, we will delve deeper into the concepts and components of Nonogram puzzles, examine their unique data characteristics, and discuss the rationale behind treating Nonogram as a node classifcation problem. Furthermore, we will highlight the potential applications and expected outcomes of solving the node classifcation problem in Nonogram puzzles.

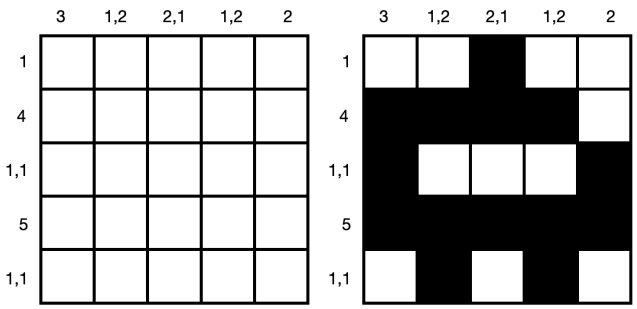

Fig. 1. Example of Nonogram

II. Explaining Nonogram

The objective of Nonogram puzzle is to determine the color or shading of each cell in a grid based on the numbers provided on the side of the grid. [2]

The puzzle is presented on a grid, usually rectangular. The grid has clues or numbers on its top and left side. These numbers indicate the number of consecutive colored cells in a given row or column. If there are multiple numbers in a clue, it means there are multiple groups of consecutive colored cells, and at least one blank cell must separate these groups. solution reveals a hidden picture, usually a pixelated representation of an object or scene.

A number in a row or column clue defines a sequence of colored cells. Different sequences in the same row or column must be separated by at least one blank cell. The order of the sequences in the clues must be maintained in the grid.

For example, consider a 5x5 grid with the row clues as [[3], [1, 2], [2, 1], [1, 2], [2]] and column clues as [[1], [4], [1, 1], [5], [1, 1]] . The solution would involve filling the grid in a manner that respects the provided clues, revealing a hidden picture.

III. Prior Research

A. Traditional Nonogram Solving Methods

Nonograms have attracted various algorithmic approaches due to their puzzle nature.

- **Logical Rules:** Authors of [3] introduced a strategy for resolving nonogram puzzles by employing specific logical rules and backtracking to efficiently determine undetermined cells.

- **Dynamic Programming:** Papers such as [2] and [4] propose methods based on dynamic programming.
- **Evolutionary Algorithms:** Some solvers, as referenced in [5] and [6], employ evolutionary algorithms. The solvers possess the capability to handle puzzles that have several potential solutions to provide only a single solution.

B. Graph Neural Networks

Graph Neural Networks (GNNs) have emerged as compelling deep learning architectures tailored to understand and extract patterns from graph-structured data. Their design allows for sophisticated representation learning of nodes, edges, and entire graphs, and their effectiveness is manifested across numerous applications.

One of the pioneering architectures in this domain is the Graph Convolutional Network (GCN) [7]. The inception of GCNs dates back to the early 2010s, originating from the desire to adapt convolutional operations to non-Euclidean domains such as graphs. These initial endeavors hinged on spectral graph theory. A significant stride was made by Thomas Kipf and Max Welling in 2016, who introduced a more streamlined version of GCNs. By utilizing a first-order approximation of spectral graph convolutions, they endowed the architecture with computational efficiency and ease of implementation, catalyzing a surge in the adoption and evolution of GCNs.

Building on this momentum, other variants of GNNs have been proposed. The Graph Attention Network (GAT) [8], for instance, embeds an attention mechanism, allowing the model to dynamically assess the significance of neighboring nodes. This inclusion augments the network's flexibility, yielding richer node representations by emphasizing contextually relevant neighbors.

Another notable model is GraphSAGE([9]), which offers a novel method to fuse features from neighboring nodes, thereby creating enriched node representations. Unlike traditional GNNs that require the entire graph for training, GraphSAGE's design permits learning either from the full graph or a subset of nodes, enhancing its adaptability.

Comparing architectually, GCNs adopt a spectral approach to consolidate information from adjacent nodes. GATs weave in an attention mechanism, calibrating the relevance of each neighboring node during the aggregation process. GraphSAGE leverages diverse aggregation functions, enabling a robust amalgamation of information from neighboring entities.

C. Using GNNs to solve NP-hard puzzles

Using Graph Neural Networks (GNNs) to tackle NP-hard problems or grid-based puzzles like Sudoku, Kakuro and Nonogram is a compelling and recent research direction.

For solving Sudoku, solutions such as [10] and [11] were proposed. Especially, [12]'s approach exploits the graph-size invariance of GNNs by converting a multi-class node classification problem into a binary node classification problem.

Authors of [13] solved Kakuro Problems using Recurrent Relational Networks. They say that the learning process needed much more time to converge on problems of similar size compared to Sudoku.

IV. Experiments

In this section, we detail the methodology employed in constructing our models for the Nonogram puzzle. The models were primarily based on Graph Convolutional Networks (GCN) [7], Graph Attention Networks (GAT), and GraphSAGE architectures. In addition, we compare results from CNN, RNN models to prove that Graph-based models are fit to solve Nonograms.

A. Dataset

The dataset used for this research was sourced from WebPbn platform. We collected 9400+ Nonogram puzzles, each with varying grid size and complexity. Each puzzle in the dataset includes its clues, solution grid, and the resulting image. We only used black-and-white puzzles.

	Dataset
Number of Puzzles	9418
Number of Rectangle Puzzles	4155
Average Grid Size	27.36 x 26.04
Width Quartiles	20 | 23 | 35
Height Quartiles	20 | 20 | 33
Minimum-Sized Puzzle	1 x 1
Maximum-Sized Puzzle	99 x 99
Average Number of Clues	195
Average Clue Value	3.92
Density of Filled Cells	0.0003

TABLE I
DATASET STATISTICS.

The dataset was split into training, and testing sets with a ratio of 80%:20%. The training set was used for model training, the validation set for hyperparameter tuning, and the testing set for the final evaluation of the models' performance.

B. Creating Nodes and Edges

The inherent structure of nonograms lends itself naturally to the framework of a graph. Given the puzzle's matrix-like appearance, it is not a far stretch to envision it as a mesh or a grid. Each cell within the nonogram can be perceived as a node, and the adjacency relations between these cells can be represented by edges, facilitating the translation of the puzzle into a graph representation.

Every individual cell in the nonogram becomes a distinct node in the graph. These nodes retain information about their position within the grid and their current state (filled, empty, or undetermined). This nodal information plays a crucial role when applying graph-based algorithms, as the state of a node can be influenced by its neighboring nodes.

Edges are crafted based on the principle of cell adjacency. In its simplest form, each cell in the nonogram grid has potential connectivity to its immediate neighbors: top, bottom, left, and right (in a 4-connectivity scheme). This results in a regular grid structure.

However, we also explored the possibility of including diagonal connections, resulting in an 8-connectivity scheme. Interestingly, our experiments revealed that this approach led to a decrease in accuracy. The inclusion of diagonal connections might have introduced additional complexity or noise that hampered the performance of our model.

C. Embedding

In order to prepare our Nonogram data for GNN models, we fixed the embedding size for both the row and column hints to a uniform length of 40. Given the variable lengths of hints in different puzzles, padding was necessary to achieve this uniformity. Specifically, we employed left-zero-padding: for a hint sequence shorter than 40, zeros were prepended to the sequence until the desired length was reached. For instance, a hint [3, 4, 5] would be transformed into [0, 0, ..., 0, 3, 4, 5], where the total length is 40.

After this preprocessing step, for each puzzle, we concatenated the row and column features, resulting in a combined feature vector. This concatenated vector was then used as the input for our model.

D. GNN-based Models Performance

We considered three primary graph-based architectures: GCN, GAT, and GraphSAGE. Each model's primary objective is to deduce the correct pattern from the provided clues and fill in the grid's cells accordingly.

	GCN	GAT	GraphSAGE
Number of Layers	3 layers	3 layers	2 layers
Dropout Rate	50%	50%	50%
Hidden Channels	16	32	16
Number of Heads (GAT only)	-	1	-
Training Epochs	30	30	30
Accuracy(Test)	64.38%	66.74%	71.26%

TABLE II
OVERVIEW OF MODEL ARCHITECTURES, HYPERPARAMETERS, AND PERFORMANCE.

E. Comparison with Non-GNN Models

Graph Neural Networks (GNNs) have demonstrated their prowess in handling structured data like Nonogram puzzles. To ascertain their effectiveness, it is essential to juxtapose them with traditional non-GNN models. In this section, we compare our proposed GNN models with some prevalent machine learning and deep learning models.

- **Data Preprocessing: Padding** To feed the Nonogram puzzles into the CNN and RNN models, which necessitate fixed input dimensions, we applied padding. Each puzzle was adjusted to match the size of the largest Nonogram in our dataset by appending additional empty cells (or rows/columns). This ensured a consistent input shape across all samples.
- **CNN (Convolutional Neural Network) [14]:** Our CNN architecture is designed with three convolutional layers, each followed by a ReLU activation function. The first two layers serve to capture the spatial patterns in the nonogram data with a kernel size of 3, while maintaining the size using padding. The third convolutional layer reduces the channel dimension to 1 and employs a sigmoid activation to bound the output values between 0 and 1.
- **RNN (Recurrent Neural Network) [15]:** Our RNN model treats the nonogram grid as sequences. Each row or column of the nonogram is treated as a sequence with the RNN iterating over it. The architecture employs a single RNN layer followed by a fully connected layer that decodes the RNN's final output to produce the desired shape. Dropout is also incorporated to prevent overfitting. To structure the data for the RNN, each input is reshaped and permuted to have a shape compatible with RNN input requirements. However, given the maximum size of our nonograms being 99x99, this architecture proved to be rather unsuitable. The large sequence length posed challenges in capturing dependencies and increased the model's time and space complexity, leading to suboptimal results.

	GraphSAGE	CNN	RNN
Number of Layers	2	5	1
Hidden Channels	16	64	512
Dropout Rate	50%	50%	50%
Accuracy	71.26%	11.59%	11.23%

TABLE III
COMPARISON OF DIFFERENT MODEL ARCHITECTURES AND THEIR PERFORMANCE ON NONOGRAM DATA, PRESENTED IN A TRANSPOSED FORMAT.

F. Impact of Size Variance on CNN Performance

One of the remarkable observations from our experiments was the notably low accuracy (around 10%) that the CNN model exhibited. To investigate the root cause of this poor performance, we hypothesized that excessive padding, which is necessary to handle nonograms of various sizes, might be a critical factor.

To test this hypothesis, we conducted a controlled experiment using a dataset of fixed size nonograms. Specifically, we isolated the 20x20 size nonograms from our dataset, resulting in 1989 training instances and 507 test instances. This subset of data allows the CNN model to operate without the need for extensive padding, providing a cleaner environment to observe its inherent capabilities.

With this constrained dataset, we retrained our CNN model under the same configuration as used in our previous experiments. The result was significantly improved, with a test accuracy of 60.92

This experiment elucidates a vital point: the extensive padding required to process nonograms of varying sizes is

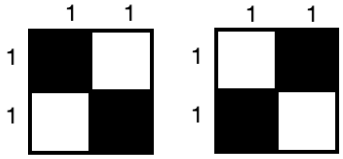

Fig. 2. Example of Non-Unique Solution

substantially detrimental to the CNN model's performance. When operating on a consistent and smaller input size, where the impact of padding is minimized, the CNN model demonstrated a considerable improvement in accuracy.

Therefore, while the CNN model exhibits potential for solving nonograms, it is notably challenged when tasked with handling puzzles of diverse sizes. This experiment emphasizes the significant role that padding plays in the performance of convolutional neural networks for this task, and suggests a need for further research into architectures more robust to input size variations.

	Diverse-sized	**20x20**
# of Training Data	7534	1989
# of Test Data	1884	507
Test Accuracy	11.59%	60.92%

TABLE IV
COMPARISON OF CNN MODEL ACCURACY UNDER DIFFERENT INPUT SIZES

V. LIMITATIONS

While the proposed models demonstrate promising results on the Nonogram dataset, they come with their inherent limitations:

- **Non-Unique Solutions**: There may be non-unique solutions to Nonogram, such as proposed example Fig. 2.
- **2-Dimensional Constraints**: Our models primarily focus on 2-dimensional Nonogram puzzles. Nonograms can also exist in a 3-dimensional space, where the challenges and intricacies of the puzzle significantly evolve. The current architectures might not seamlessly generalize to 3D Nonograms without modifications or may require an entirely different approach.
- **Complexity and Scalability**: As the complexity of the Nonogram increases, especially with the growth of grid size, the computational burden on the model intensifies. Further optimization techniques may be necessary for larger and more intricate puzzles.
- **Specificity of Features**: The current models leverage generic graph-based features to predict the puzzle outcomes. However, specific features tailored to the intricacies of the Nonogram puzzles might further enhance the performance.

It is essential to keep these limitations in mind when considering the deployment or further adaptation of the proposed methodologies. Future research could focus on developing architectures more attuned to 3-dimensional Nonograms and incorporating more problem-specific features.

VI. CONCLUSION

In this research, we delved into the unique problem of solving Nonogram puzzles utilizing Graph Neural Networks (GNNs). Our exploration revealed the intricacies and challenges associated with this domain, particularly in harnessing the structured information embedded within the puzzles.

Our findings indicated that GNNs, with their capability to capture spatial relationships, inherently lend themselves to Nonogram puzzles more effectively than traditional non-GNN models. Through our experiments, the GNN models, especially the GCN, GAT, and GraphSAGE, consistently outperformed their counterparts, affirming the advantages of adopting a graph-based approach for such problems.

However, the research also unveiled certain limitations. While 2-dimensional GNNs showcased promising results, it is plausible that 3-dimensional architectures might further enhance performance, given the spatial nature of Nonogram puzzles. This provides a direction for future research.

Furthermore, the comparison with non-GNN models underscored the flexibility and scalability of GNNs, especially in handling Nonograms of varying grid sizes without the necessity of resizing or padding, a significant challenge for models like CNNs.

In sum, this research underscores the potential of GNNs in the realm of puzzle-solving and offers a fresh perspective on approaching structured problems. As future work, exploring more sophisticated GNN architectures, augmenting training strategies, and delving into 3D GNNs could further push the boundaries in Nonogram puzzle solving.

ACKNOWLEDGMENT

REFERENCES

[1] N. Ueda and T. Nagao, "Np-completeness results for nonogram via parsimonious reductions," *preprint*, 1996.
[2] K. J. Batenburg and W. A. Kosters, "Solving nonograms by combining relaxations," *Pattern Recognition*, vol. 42, no. 8, pp. 1672–1683, 2009.
[3] C.-H. Yu, H.-L. Lee, and L.-H. Chen, "An efficient algorithm for solving nonograms," *Applied Intelligence*, vol. 35, pp. 18–31, 2011.
[4] I.-C. Wu, D.-J. Sun, L.-P. Chen, K.-Y. Chen, C.-H. Kuo, H.-H. Kang, and H.-H. Lin, "An efficient approach to solving nonograms," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 5, no. 3, pp. 251–264, 2013.
[5] K. J. Batenburg and W. A. Kosters, "A discrete tomography approach to japanese puzzles," in *Proceedings of the 16th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC)*, 2004, pp. 243–250.
[6] W. Wiggers and W. van Bergen, "A comparison of a genetic algorithm and a depth first search algorithm applied to japanese nonograms," in *Twente student conference on IT*. Citeseer, 2004.
[7] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.
[8] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.
[9] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[10] R. Palm, U. Paquet, and O. Winther, "Recurrent relational networks," *Advances in neural information processing systems*, vol. 31, 2018.
[11] P.-W. Wang, P. Donti, B. Wilder, and Z. Kolter, "Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6545–6554.
[12] Y. Nandwani, V. Jain, P. Singla *et al.*, "Neural models for output-space invariance in combinatorial problems," *arXiv preprint arXiv:2202.03229*, 2022.
[13] W. Daniec, "Solving kakuro problems using recurrent relational networks."
[14] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[15] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.