The Application of Graph Neural Networks in Maze Problem

Xinyang Chen

Faculty of Engineering, University of New South Wales, Sydney, Australia

z5499873@ad.unsw.edu.au

Abstract. Graph Neural Network (GNN) shows significant applications in many fields these days, being a possible strong solution for Maze Problem, like finding paths and predicting distance. This paper attempts to apply Graph Neural Network with traditional algorithms as solutions. First, this paper built a generator to construct mazes as the data set for later experiments. Then, this paper proposes two methods. One is GNN-ShortestPath, that predicts distance to the target position by aggregating messages along the graph. The other one is GNN-Policy, which generates distribution of actions in steps. This paper performs experiments based on the methods and evaluates the results on Successful Rate and Optimal Rate. Finally, this paper analyses results and discusses limitation and possible improvements. Despite of many limitations in the experiments, this paper shows that application of Graph Neural Network in solving Maze Problem has great advantages and traditional algorithms can assist in the performance of Graph Neural Network.

Keywords: Graph Neural Network; Maze Problem; Shortest Path.

1. Introduction

Graph Neural Network (GNN) shows great advantage these years in graph's learning and processing, widely used in social network, knowledge graph and drug discovery [1]. However, application of GNN in path planning and graph searching still needs further exploration. As a typical problem querying reachable or shortest path, Maze Problem has comprehensive graph structure and great constructing convenience, making it an ideal environment for GNN evaluation in these areas. Traditional graph searching algorithms, including Breadth-first Search (BFS), Dijkstra and A*, can find optimal solutions, but require global graph information and explicit graph search, causing high cost and huge complexity in computation [2]. On the opposite, solutions based on GNN can learn from data distribution and thereby predict potential distance to target node, avoiding explicitly global searching. Effective implementation of GNN searching solutions to maze problem can greatly benefit the performance of agent investigation and robot path planning [3].

Currently, straightforward applications of GNN on shortest path learning are still facing some challenges. The first problem is reachable problem in maze generation. MSH Shah et.al compared different kinds of maze generation algorithms and found that imperfect generation of maze may cause disconnection between source and sink [4]. Low reachable rate will theoretically limit the Successful Rate(SR). Another major problem is the lack of completeness guarantee, mentioned by Xuebo et al. [5], that models based on GNN still needs further improvement. One possible solution is to implement better noise prediction and decoding strategy, instead of greedy algorithm or beam search with small batches, which may lead to dead end and decrease SR. Besides, research applying neural network to maze problem lacks entire extraction of the topological information in graphs. Arbaaz et al. proved that GNN can outperform traditional learning methods on graph-based planning, but the difficulty still exists [6].

Considering these challenges, this paper systematically performs experiments on two models to solve maze problem: 1) Distanced-based(GNN-SP): Learning distance data from nodes to target to predict and recover path by different encoders 2) Policy-based(GNN-Policy): Learning policy from each moves to generate shortest paths. This paper wants to show that GNN can play important roles in the improvement of traditional graph search algorithms.

2. Method

2.1. Maze PathFinding Problem Definition

This paper proposed an experiment on Maze Pathfinding Problem: Given a 2-dimension maze, it can be represented as a matrix with binary value, where value of 1 means viable element, and value of 0 means wall. The target of the problem is to find path between given source element and sink element and minimize the length of the path, with every element on the path has value of 1.
To apply GNN on Maze Pathfinding Problem, a maze will be regarded as a planar graph, where each viable element can be regarded as a node in the graph. Undirected edges exist between neighbours. The Maze Pathfinding Problem is equivalent to find the shortest path between source node and sink node.

2.2. Maze Designing

To systematically study Maze Pathfinding Problem based on GNN, a large-scale and diverse maze dataset is required. Therefore, this paper implemented two maze generators. Each generator produces maze as 2-dimension grid, where value of 1 means viable element and value of 0 means wall. Source and target elements are randomly sampled from viable elements.

Depth-First Search(DFS) Generator: DFS Generator is based on recursive backtracking, with a randomly selected grid as the start element [7]. The generator will mark the start element as viable element, and then randomly select unvisited neighbour elements by marking them as viable element too. This process will be repeated until a dead end appears, when the generator will recursively go back to the previous element and continue to explore unvisited elements. Mazes generated by DFS Generator always have a tree-like structure, making each path unique and the graph acyclic. Paths in these mazes are often long and narrow with fewer selections, being suitable for evaluation of GNN models in the aspect of long-distance deduction.

Prim Generator: What Prim Generator will do is constructing minimum spanning tree. The generator starts from a viable set with a randomly selected element, marking surrounding walls of the set as “border”. In each step, Prim Generator will randomly select a wall from “border” and mark it as viable so that viable set can extend in the graph. Mazes generated by Prim Generator always have more branching paths, especially compared with mazes generated by DFS Generator. Sometimes cycles occur in the mazes increasing the complexity of mazes. These mazes provide more possible solution paths to GNN, being suitable for evaluation of models’ planning ability.

2.3. GNN-ShortestPath

GNN-ShortestPath(GNN-SP) aims to study distance field on a maze graph by GNN, to find shortest paths through message passing. In GNN-SP, features of nodes would be initialized with normalized index and targets’ indicator. The model is based on Graph Isomorphism Network (GIN), recursively aggregating information from neighbours by message passing and finally output predicted distance from the node to the target. The labels are generated as the shortest distance calculated by Breadth-First Search, and Smooth L1 regression loss is chosen as the loss function, encouraging to minimize predicted distance. Finding shortest path is implemented by encoders, gradually minimizing predicted distance until reaching the target. The encoders adopt different strategies like greedy search or random walk.

GNN-SP is fed on global shortest path distance as signals, so the output can reflect complete information of the graph rather than local searching strategy. Every node in GNN-SP model has predicted distance toward target element, making it convenient to explain experiment results. Moreover, GNN can avoid explicit search by BFS, decreasing high computation cost in large-scale mazes. Greedy encodes can optimize the solutions, while encodes of random walk can be more robust. Diversity and flexibility of GNN-SP are fundamental for evaluating the model on large maze dataset.

2.4. GNN-Policy

GNN-Policy no longer tries to learn global distance field like GNN-SP but generates probability distributions for each node's next action. The labels are calculated by BFS and supervised based on if the action correctly marches toward the target. The model is based on GIN and aggregates neighbours' information by message passing. Given a maze graph, the model processes features for every node and generate a logit representing possible action in the next step. The output probability is normalized by a softmax layer, and cross entropy function is selected as the loss function.

GNN-Policy is an end-to-end model directly producing actions' distribution, without any explicit distance labels or heuristic functions. Supervisor signals in GNN-Policy are generated from action labels in BFS, so that the model is imitating BFS's action decisions as a locally optimal choice. Lack of global view, GNN-Policy model is less stable than GNN-SP because of local errors' accumulation, causing more failures during searching. However, searching algorithms combined with GNN-Policy model might possibly reduce size of searching space and improve efficiency.

2.5. Implementation Details

Dataset used in this paper is the set of mazes generated by DFS and Prim algorithms. The training set consists of approximately 8,000 mazes. 1,000 of total mazes used for validation. Test sets include reachable, unreachable, and mixed subsets, with maze sizes of 31× 31. For generalization experiments, models were further evaluated on larger mazes of size 61× 61.

For GNN-ShortestPath (GNN-SP), the input feature dimension was set to 4 (including node coordinates and goal indicator), with hidden dimension 64 and 6 GNN layers. The model was trained using Smooth L1 regression loss, optimized with Adam (learning rate 3×10-4, weight decay 1×10-4. A batch size of 8 was used, with a maximum of 30 epochs and early stopping patience of 5. At inference, path decoding was performed via greedy or random-walk strategies.

For GNN-Policy, the model outputs a 4-dimensional action distribution corresponding to the four movement directions (up, right, down, left). Training labels were derived from BFS trajectories, and the model was trained with cross-entropy loss. Adam optimizer was used with a learning rate of 1×10-3. Weight decay was set to either 0.0 or 1×10-4 depending on the setting. The number of training epochs was 10 or 30, depending on the experiment. Additional hyperparameters included a visited penalty (0.2), goal bias (0.03), loop threshold (6), and optional entropy regularization coefficient (1×10-3).

2.6. Evaluation Metrics

To completely compare the performance of GNN-SP and GNN-Policy models, this paper adopts three evaluation metrics in different aspects. Successful Rate(SR) represents the proportion of finding correct solution path in limited steps over the maze dataset. This metrics reflect the basic ability of models in robustly solving maze problem. Optimal Rate(OR) measures the quality of found solution path by comparing the models' output with shortest paths found by BFS. It is defined as the ratio of lengths between solution path and shortest path. Number of Extended Nodes shows the size of searching space, as an efficient criterion to evaluate computational cost. By analysing the three metrics, models' accuracy and efficiency can be compared and discussed in various aspects.

3. Results and Discussion

Experiments results are shown in Table 1. GNN-ShortestPath and GNN-Policy with DFS achieve 100% successful rate, meaning that these two algorithms successfully solve the maze problem. However, simple application of GNN-Policy results low successful rate, showing that GNN-Policy method does not find ways to solve different maze problems. Both GNN-ShortestPath and GNN-Policy with DFS can achieve 1.000 optimal rate, which means that these two methods can find the shortest paths as solutions. Simple GNN-Policy does not always find the shortest path according to

its optimal rate. Besides, GNN-Policy method with DFS requires around 1,700 extended nodes on average, which is lower than plain DFS.

Results show that GNN-ShortestPath can achieve high performance as traditional graph search algorithms without explicit node extension. High successful rate and optimal rate are caused by supervision of BFS. Supervised by BFS, GNN can learn from distance labels and propagate messages until the shortest path is found. Another reason is that training set and validation set come from the same generator designed in this experiment. Reachable and stable mazes make it easy for models to learn and perform.

Simple application of GNN-Policy method does not perform well, showing that predictions based on local information are unstable and difficult to find the optimal solution. Several reasons exist for the low performance of this method. First, previous errors are difficult to be corrected, significantly enhancing the failing opportunity. Next, complex mazes can make every local action similar to the model, increasing the probability to get wrong direction. Cycle in mazes may also decrease the successful rate.

However, GNN-Policy can be used to help increase the performance of traditional graph search algorithms. This experiment combines GNN-Policy and DFS, considering the prediction of GNN-Policy as the traversing order. Results show that this method can match the successful rate and optimal rate of GNN-ShortestPath method. Moreover, compared with single application of DFS algorithm, combination of GNN-Policy and DFS can reduce the number of extended nodes, making it a possible optimization for maze problem solving.

Although high successful rate and optimal rate are recorded in this experiment, many limitations exist and are hoped to be solved in the futural experiments. First, models require the supervision of traditional graph search algorithms like BFS and DFS [8-10], which will limit the performance and restrict application scenarios. More comparison between proposed models should be performed with the results from traditional algorithms. Reinforcement learning might be a possible solution for this limitation. Second, more parameters and evaluation metrics can be applied to the models. Analysis based on successful rate and optimal rate is not enough. Other metrics, like time cost should be considered to measure the performance of the models. Various maze data sets and generators can also be applied to this experiment to test if the models can be generalized. Third, large complex mazes can contain disturbing options or cycles, which can lower the precision and performance of the models. Despite of these limitations, this experiment still shows that application of GNN is possible optimisation for maze problem.

Table 1. Experiment Results on Three Different Models

Method	SR(%)	OR	Average Nodes
GNN-SP	100.0	1.000	-
GNN-Policy	6.0	1.630	-
GNN-Policy+DFS	100.0	1.000	~1700

4. Conclusion

This paper adopts GNN methods to solve maze path finding problem. Two different models are established. One is GNN-ShortestPath, which learns global distribution based on BFS. Another is GNN-Policy, which learns local information and makes predictions of local action. The experiment is performed based on maze dataset generated by DFS and Prim algorithm. Results show that GNN-ShortestPath can achieve high performance with effective node expansion, while GNN-Policy might fail because of prediction errors. Despite of limitations in the experiment, this paper shows the advantages of GNN in solving maze problem and how traditional graph algorithms can be combined with GNN methods. In the future, GNN is going to enhance traditional algorithms to solve maze problems in many ways.

References

[1] Wu Z, Pan S, Chen F, Long G, Zhang C, Yu PS. A comprehensive survey on graph neural networks. IEEE Trans Neural Netw Learn Syst. 2021;32(1):4–24. doi:10.1109/TNNLS.2020.2978386.

[2] Kaur NKS. A review of various maze solving algorithms based on graph theory. IJSRD. 2019;6(12):431–4.

[3] Ji X, Li H, Pan Z, Gao X, Tu C. Decentralized, unlabeled multi-agent navigation in obstacle-rich environments using graph neural networks. In: Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE; 2021. p. 8936–43.

[4] Shah MSH, Mohite MJM, Musale AG, Borade JL. Survey paper on maze generation algorithms for puzzle solving games. Int J Sci Eng Res. 2017;8(2):1064–7.

[5] Ji X, Li H, Pan Z, Gao X, Tu C. Decentralized, unlabeled multi-agent navigation in obstacle-rich environments using graph neural networks. In: Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE; 2021. p. 8936–43.

[6] Khan A, Ribeiro A, Kumar V, Francis AG. Graph neural networks for motion planning. arXiv preprint arXiv:2006.06248. 2020.

[7] Coremen TH, Leiserson CE, Rivest RL, Stein C. Introduction to Algorithms. 3rd ed. Prentice Hall of India; 2009. p. 587–748.

[8] Wang Q. Comparative analysis of maze solving algorithms based on OCaml. In: AIP Conference Proceedings. AIP Publishing LLC; 2024. Vol. 3194, No. 1, p. 040001.

[9] Biju A, Gayathri MP, Jayapandian N, Chris A, Thaleeparambil NR. Efficient pathfinding in a maze to overcome challenges in robotics and AI using breadth-first search. In: Proceedings of the 2025 IEEE 14th International Conference on Communication Systems and Network Technologies (CSNT). IEEE; 2025. p. 727–32.

[10] Mahmud MS, Sarker U, Islam MM, Sarwar H. A greedy approach in path selection for DFS based maze-map discovery algorithm for an autonomous robot. In: Proceedings of the 2012 15th International Conference on Computer and Information Technology (ICCIT). IEEE; 2012. p. 546–50.