University of Bergen
Department of Informatics

Solving Maximum Weighted Matching problem using Graph Neural Networks

Author: Nikita Zaicev
Supervisor: Fredrik Manne

UNIVERSITETET I BERGEN
Det matematisk-naturvitenskapelige fakultet

December, 2023

Abstract

In this work we tried to train a Graph Neural Network (GNN) to solve a popular problem of Maximum Weighted Matching.

Acknowledgements

I want to thank Fredrikk Manne, Kenneth Langedal and Johannes Langguth for helping me with this work.

Nikita Zaicev
Saturday 30th December, 2023

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

This chapter is dedicated to the general introduction of the problem at hand and previous work and research that is relevant for this project.

Machine Learning (ML) and Neural Networks have shown to be extremely potent and versatile in solving vast variety of problems across different fields. One research field that has been popular and challenging in the last few years is Combinatorial Optimization (CO). CO includes problems such as Maximum Indepndent Set (MIS) and Maximum Weighted Matching (MWM). Such problems can be viewed in the context of graphs. GNN is a subclass of Neural Networks designed specifically for solving problems related to graphs, but there are many problems that have not yet been solved efficiently with GNNs. This work attemps to find out whether a GNN can be worth using for solving MWM

1.1 Background

Many researches have been done related to GNNs in the past years and some of them have shown mixed results. Lorenzo Brusca and Lars C. P. M. Quaedvlieg et al. [1] showed a self-training GNN for MIS

1.2 Project structure

The rest of this document has the following structure:

1. Approach
2. Training
3. Results
4. Conclusion

1.2.1 Listings

You can do listings, like in Listing 1.1

Listing 1.1: Look at this cool listing. Find the rest in Appendix A.1

```
$ java -jar myAwesomeCode.jar
```

You can also do language highlighting for instance with Golang: And in line 6 of Listing 1.2 you can see that we can ref to lines in listings.

Listing 1.2: Hello world in Golang

```
package main

import "fmt"

func main() {
    fmt.Println("hello world")
}
```

1.2.2 Figures

Example of a centred figure

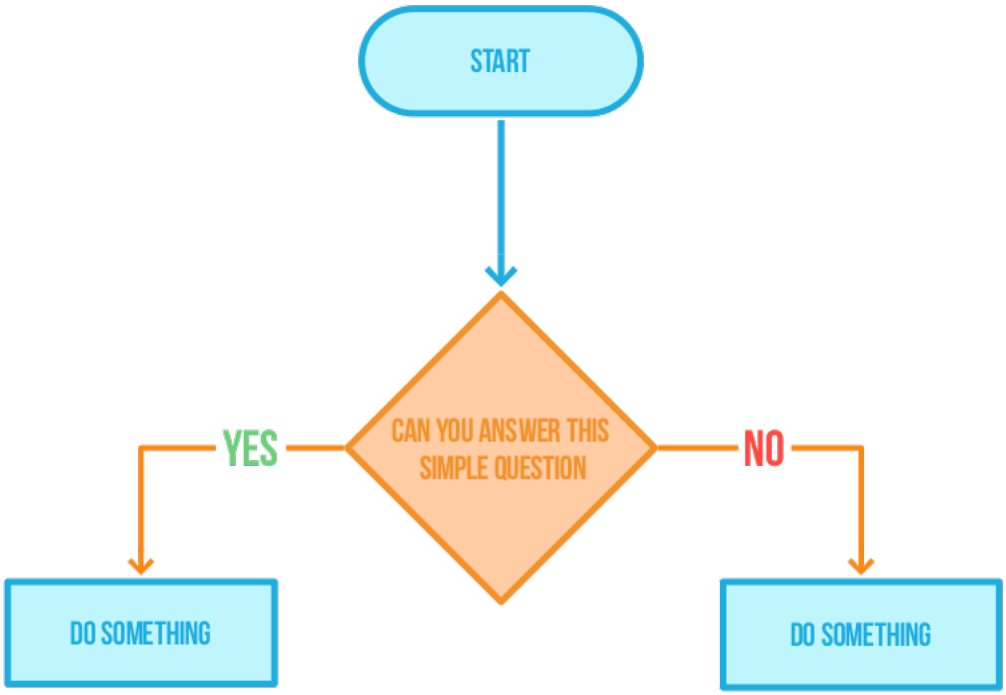

Figure 1.1: Caption for flowchart

Credit: Acme company makes everything `https://acme.com/`

1.2.3 Tables

We can also do tables. Protip: use `https://www.tablesgenerator.com/` for generating tables.

Table 1.1: Caption of table

Title1	Title2	Title3
data1	data2	data3

1.2.4 Git

Git is fun, use it!

Chapter 2

Approach

This chapter focuses on describing which approaches were chosen for the task at hand and why.

To reiterate the current problem is to find out if GNN can compete at solving the MWM problem compared to other options such as greedy algorithm.

Before properly describing the choices made for approaching this problem, it is worth going through basic concepts of GNNs and neural networks in general. It is common to think of neural networks as of a black box where you give some data to this box and it gives back an answer. In this case the data is a graph consisting of vertices connected by edges that have weights and the expected answer should be pairs of vertices that were matched together. This is however not as straight forward as the more common examples of classification where each data item needs to be classified.

2.1 Line graph approach

Line graph approach was the first attempt at using a simple GNN, which later turned out to be too time consuming for larger graphs to be worth further expriements. It did however give some usefull insight as well as a showed to be a proof of concept. In the context of graphs line graph is a complement of the original graph that turns each edge to a vertex and connects the vertices is they shared a vertex in the original graph.

Examaple of a graph and its line graph convertion

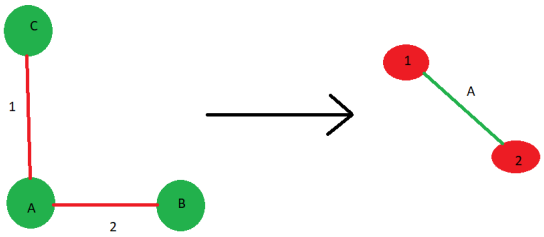

Figure 2.1: Original graph (left) and its line graph (right)

Credit: my paint skills

2.2 Edge classification approach

The usual thing to do is for each pair of vertices that are conneced by an edge, ask the model if this pair should be a part of the matching. In other words model is classifying an edge.

2.3 Result Validation

How good a solving method for MWM is can be evaluated by:

1. Time - how long an algorithm took to produce an answer.
2. Correctness - is the answer correct. In case of MWM the total weight aquired would be the measurement of how correct the solution is. It is unlikely that GNN can find an optimal solution for more complex problems so it is reasonable to look at how close GNN comes to the optimal solution.
3. Memory - how much memory is needed. However in this project there is less of a focus on memory

Chapter 3

Graph Neural Network

ML has become a powerfull tool for solving a wide variety of problems. One such field that has been particulary challenging is CO.

3.1 Difference from normal neural network

Why it works better for graphs

Chapter 4

Training

ML has become a powerfull tool for solving a wide variety of problems. One such field that has been particulary challenging is CO.

4.1 Data

What data was used for training

Chapter 5

Results

ML has become a powerfull tool for solving a wide variety of problems. One such field that has been particulary challenging is CO.

5.1 Performance on unseen data

How well the final model solves the matching problem

Chapter 6

Conclusion

ML has become a powerfull tool for solving a wide variety of problems. One such field that has been particulary challenging is CO.

6.1 Future work

Results show that this approach

Glossary

Git Git is a Version Control System (VCS) for tracking changes in computer files and coordinating work on those files among multiple people.

List of Acronyms and Abbreviations

CO Combinatorial Optimization.

GNN Graph Neural Network.

MIS Maximum Indepndent Set.

ML Machine Learning.

MWM Maximum Weighted Matching.

VCS Version Control System.

Bibliography

[1] Lorenzo Brusca, Lars C. P. M. Quaedvlieg, Stratis Skoulakis, Grigorios G Chrysos, and Volkan Cevher. Maximum independent set: Self-training through dynamic programming, 2023.

Appendix A

Generated code from Protocol buffers

Listing A.1: Source code of something

```
1 System.out.println("Hello Mars");
```