553.467/667 Project Proposal

Andrew King (aking65), Jack Armand (jarmand1),
Vipul Bhat (vbhat1), Allison Rosen (arosen46)

March 31, 2020

1 Abstract

In this project, we will attempt to improve the generalization of the techniques used in the paper "Learning combinatorial optimization algorithms over graphs" by Hanjun Dai et al [1]. One of our strategies will be to adopt a supervised learning approach in which a neural network is trained to solve problems on small, pre-solved instances of a combinatorial graph problem instead of the reinforcement learning paradigm studied by Dai. Our hope is that training on these pre-solved solutions will allow our network to develop a heuristic that better generalizes to larger problem instances, and reduces computing time. The combinatorial problem we are specifically interested in is the traveling salesman problem because of the large number of solved instances available for training. Another way in which we wish to improve on Dai's work is to use a variety of graph encodings as inputs to our network in the hopes that we can represent a more general set of graph features relevant to the TSP.

2 Introduction

Many combinatorial graph problems, such as the Traveling Salesman Problem, have been popular areas of research in recent years due to their presence in NP-hard problems. An important step in reducing the time necessary to solve these combinatorial problems is developing a good heuristic. Although a heuristic does not typically yield the optimal solution, a good heuristic will place a worst-bound on it. The paper "Learning combinatorial optimization algorithms over graphs" by Hanjun Dai et al utilizes a reinforcement learning approach to develop a heuristic for the Traveling Salesman Problem [1]. The downside to this strategy is that the problem needs to be learned each time, which is both time consuming and poorly generalizable. We propose utilizing a neural network that will be trained on smaller, pre-solved instances of the Traveling Salesman Problem. We intend to apply the principles of deep learning to try to devise a better heuristic than the model developed by Dai. These pre-solved instances should also allow our network to generalize to other larger instances of the problem. As a result, if successful, this approach will be able to quickly yield a good heuristic for any instance of the Traveling Salesman without a need for significant computation.

3 Planned Research

The two areas we will focus on are the development a loss function so that a good heuristic can be learned from labeled training data and encoding our graph inputs in a way that retains the salient information about them. We will likely need to create a custom loss function to use in training the network. On a complete weighted graph with n nodes, the number of edges in a tour grows proportional to n, whereas the total number of edges grows proportional to n^2, so the number of correct edges in a tour can become very sparse in the total number of edges as n increases. We wish to develop a loss function so that our network does not learn the trivial solution of assigning a low

probability of being included in the optimal tour to every edge, which we might expect to happen if we use a squared error or cross entropy loss.

A fixed length vector input will be required in order to use the same network on TSPs of varying sizes. Thus we will need to find an effective way to encode TSP input into a fixed length vector while still capturing the information necessary to build a working model. For this, we have a few proposed modifications to Dai's strategy. Firstly, we believe it may be possible to have better selection criteria to determine the encoded nodes fed into our network. For example, Dai et al. chose to encode the 25 nearest neighbors to a given node, however it was not as effective as a farthest insertion heuristic. Could our network learn a hybrid insertion if we encode the nearest and furthest neighbors to any node? Secondly, we can look into different iterative encoding procedures altogether to see if there is one that retains more relevant information to the TSP. Thirdly, many papers have been written about graph autoencoders for any graph on a specified number of nodes. We can attempt to pre-train an autoencoder for the number of nodes in a TSP problem and feed the result into our network.

4 Data Set and Computing Platform

We will train our network on small instances of randomly generated complete graphs, whose edges are weighted by the euclidean distance between each pair of nodes. We will take the correct solution to be the output of GUROBI's MIP solver applied to these instances. The code used in our analysis will be publicly available in repository on github located at `https://github.com/agking10/Deep-Learning-Discrete-Optimization`. We are using the PyTorch python package to create and train our networks.

References

[1] Learning combinatorial optimization algorithms over graphs, H. Dai, E. B. Khalil, Y. Zhang, B. Dilkina, L. Song. 2017. Appeared in NIPS 2017.