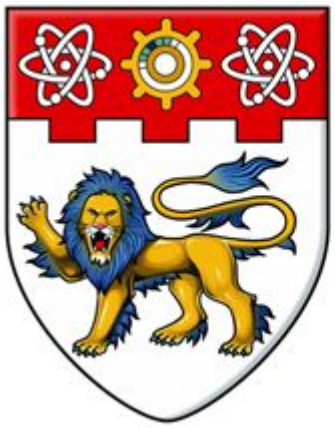


NANYANG TECHNOLOGICAL UNIVERSITY

Assignment 2 : Mid-Progress Report

CZ4071 – NETWORK SCIENCE
(Semester 2, AY 2019/2020)

Paper Reviewed in this report:
[1905.13732] End to end learning and optimization on graphs

Team 3:

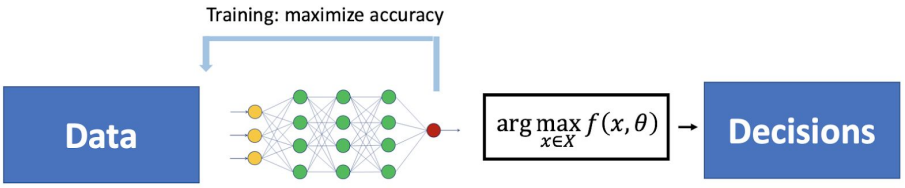
WANG ZIYAN	U1822682G
WILSON THURMAN TENG	U1820540H
CAI LINGZHI	U1622184H
LUO JINQI	U1722733J
TEO BOON SHUAN	U1821709J
LIU FANGBING	U1622143H

End to end learning and optimization on graphs

Problem Description

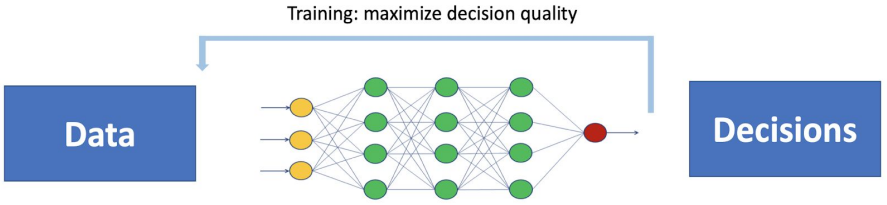
Graph Optimization (GOpt) problems have become more common following the abundance of open-source graph data in recent years. For example, Cluster Detection and Location Analysis. However, current real-world implementations of GOpt that leverage machine learning are insufficient as node attributes or linked edges data available are partially-observed in nature. Towards this goal, researchers propose solutions in two directions, which differ in the number of processing stages.

The first approach implements two stages (Fig. 1.1). In the first stage, a predictive model setting with standard loss will be trained. In the second stage, the posterior generated will then be fed to finish the optimization process. However, the problem with this approach is an oversimplified standardized loss function which leads to poor performance when dealing with complex optimization problems in particular.



(Fig. 1.1) Two-stage Approach

A preferable alternative would be the end-to-end approach (Image 1.2). In this approach, the solution of the specific optimization problem is entirely dependent on the parameters obtained from training deep learning models to develop a smaller-scale representation of the problem (e.g. Policy model of Reinforcement-Learning). However, training this representation is time-intensive and disregards additional prior knowledge which may lead to unsatisfactory performance. This problem is further exacerbated if data is lacking.



(Fig. 1.2) End-to-end Approach

In this paper, the author investigated the shortcomings of the aforementioned approaches and proposed a new solution, ClusterNet. ClusterNet has a trainable (differentiable) layer for the solver of a simpler problem that is embedded into the neural network. By obtaining the parameters of this layer in the training process, we can use the model to generate the mapping scheme of the Problem of Interest to a simplified version of it.

Specifically, the writer formalized the optimization problem as:

$$\max_{x \in \mathcal{X}} f(x, A)$$

Eqn. 1

where A denotes the adjacency matrix of the partially-observed graph *G=(V, E)* and *x* denotes the final decision of our targeted variable.

Traditionally, to solve this optimization problem, the completed *A* is obtained by the predictive mapping model learnt from the loss *L(A, A_train)*, where *A_train* is defined as the adjacency matrix with only the training edges. Afterwhich, the optimal *x* can be calculated using *Eqn.1*. However in this paper, the author proposed an end-to-end model that directly maps *A_train* to a feasible decision *x*. That is, the trained model directly solves the optimal *x* from the optimization function *f(x,A_train)*. Intuitively, we observe that *f(x,A_train)* represents the quality of the decision made from evaluating the training data, while the original loss *L(A, A_train)* only measures the predictive accuracy.

Practical Application of this paper

Intuitively, many GOpt problems can be interpreted as a variation of the cluster assignment problem. For instance, in community detection we can interpret the cluster assignment problem as assigning the nodes to communities. Or, in maxcut, we can use two clusters to assign nodes to either side of the cut. Another example is maximum coverage and related problems, where we attempt to select a set of K nodes which covers as many neighbouring nodes as possible. This problem can be approximated by clustering the nodes into K components and choosing nodes whose embedding is close to the center of each cluster. We do not claim that any of these problems is exactly reducible to K-means. Rather, the idea is that including K-means as a layer in the network provides a useful inductive bias.

There are plenty of real-world business applications for clustering a graph and finding the central nodes. For example, Pupu Mall, a start-up company in China, runs its online supermarket with a 30-minute delivery guarantee to anywhere in the city. The company arranges its warehouses around 5 to 10 kilometers apart, and hires delivery men at these warehouses to cover a particular geographical segment of the city. Before expanding its services to a new city, the company models the traffic network of the city into a graph. The nodes represent the buildings, while the distance between any two nodes represent the time required for the delivery men to travel between the buildings. Considering the limited funding that start-ups possess, it is crucial for Pupu mall to be able to experiment with layouts of where to place their initial batch of warehouses to minimise the facility's distance to all other nodes in the graph.

Main Contribution of this paper

First, the author presents a general framework for graph learning and optimization integration. The framework utilizes a simpler problem of optimization in continuous space as a substitute for the more complicated, discrete problem. Unlike the end-to-end approach, this framework is able to simplify the optimization component instead of updating the entire algorithm.

Second, the author introduced the idea of simplifying graph optimization problems into specific clustering problems, in which the clustering layer is differentiable. This allows the solution to be applied in deep learning systems. In this paper, the author demonstrated this approach to interpret the cluster assignments as a solution to the discrete problem. Specifically, this idea is helpful for two classes of optimization problems:

1) Graph partitioning (e.g., community detection or maxcut)
2) Subset selection amongst K nodes (e.g., facility location, influence maximization, immunization, etc)

Third, the author shows experimental improvements over both two-stage baselines as well as alternate end-to-end approaches over a range of open-source research datasets. The author compares ClusterNet with other methods from two aspects, namely Results on Single Graphs (RSG) and Generalizing Across Graphs (GAG). The RSG explores how the model performs on combined link prediction and corresponding optimization problems. (Fig 3.1) shows the objective value obtained by each method on the full graph for community detection, with (Fig 3.2) showing the results of facility location. The GAG experiment investigates the ability of ClusterNet to generalise its learning across similar optimization problems from different domains. (Fig 3.3) shows the results of ClusterNet's performance compared to other baseline models for generalized strategy learning. We can see from the author's experiment that ClusterNet outperforms most of the baseline methods on learning+optimization problems. (Tables are obtained from the original paper)

	Learning + optimization				
	cora	cite.	prot.	adol	fb
ClusterNet	**0.54**	**0.55**	**0.29**	**0.49**	**0.30**
GCN-e2e	0.16	0.02	0.13	0.12	0.13
Train-CNM	0.20	0.42	0.09	0.01	0.14
Train-Newman	0.09	0.15	0.15	0.15	0.08
Train-SC	0.03	0.02	0.03	0.23	0.19
GCN-2stage-CNM	0.17	0.21	0.18	0.28	0.13
GCN-2stage-Newman	0.00	0.00	0.00	0.14	0.02
GCN-2stage-SC	0.14	0.16	0.04	0.31	0.25

	Learning + optimization				
	cora	cite.	prot.	adol	fb
ClusterNet	**10**	**14**	**6**	**6**	**4**
GCN-e2e	12	15	8	**6**	5
Train-greedy	14	16	8	8	6
Train-gonzalez	12	17	8	**6**	6
GCN-2Stage-greedy	14	17	8	7	6
GCN-2Stage-gonzalez	13	17	8	**6**	6

Fig 3.1 & Fig 3.2

	Community detection					Facility location				
	synthetic		pubmed			synthetic		pubmed	
No finetune	Avg.	%	Avg.	%	No finetune	Avg.	%	Avg.	%
ClusterNet	**0.57**	**26/30**	**0.30**	**7/8**	ClusterNet	**7.90**	**25/30**	7.88	3/8
GCN-e2e	0.26	0/30	0.01	0/8	GCN-e2e	8.63	11/30	8.62	1/8
Train-CNM	0.14	0/30	0.16	1/8	Train-greedy	14.00	0/30	9.50	1/8
Train-Newman	0.24	0/30	0.17	0/8	Train-gonzalez	10.30	2/30	9.38	1/8
Train-SC	0.16	0/30	0.04	0/8	2Stage-greedy	9.60	3/30	10.00	0/8
2Stage-CNM	0.51	0/30	0.24	0/8	2Stage-gonz.	10.00	2/30	**6.88**	**5/8**
2Stage-Newman	0.01	0/30	0.01	0/8	ClstrNet-1train	7.93	12/30	7.88	2/8
2Stage-SC	0.52	4/30	0.15	0/8					
ClstrNet-1train	0.55	0/30	0.25	0/8					
Finetune					Finetune				
ClstrNet-ft	0.60	20/30	0.40	2/8	ClstrNet-ft	8.08	12/30	8.01	3/8
ClstrNet-ft-only	0.60	10/30	0.42	6/8	ClstrNet-ft-only	7.84	16/30	7.76	4/8

Fig 3.3