Learning Optimization Models of Graphs

Rakshit Trivedi
College of Computing
Georgia Tech
rstrivedi@gatech.edu

Jiachen Yang
College of Computing
Georgia Tech
jiachen.yang@gatech.edu

Hongyuan Zha
College of Computing
Georgia Tech
zha@cc.gatech.edu

Abstract

We consider the problem of learning in graph structured domains and posit that an observed graph is the outcome of an underlying optimization mechanism (cause) that drives the formation of the edges in such domains. We propose a maximum entropy framework that unifies a novel structured policy network (forward model) with inverse reinforcement learning to discover this underlying mechanism in the form of latent reward function from the observed graph (inverse problem). The learned components can be effectively used to generate or infer desired structural and semantic properties of graph. We evaluate our method on the task of graph generation and demonstrate the generalizability of our approach across instance sizes and graphs that share the underlying formation mechanism of a given domain.

1 Introduction

Graphs are natural models of information in many problem domains, spanning recommendation systems, knowledge graphs, biological networks, social networks and many more. Learning over graph structured data has emerged as an important task that has proven to be challenging in general due to their non-Euclidean nature. This has led to massive efforts in building sophisticated machine learning approaches for learning in graph structured domains [1, 2]. Most approaches either focus on extracting features from the observed graph for use in downstream applications or learning a distribution over a family of graphs for generating new graphs [3]. In this work, we inspire from the field of Economics [4] and take a novel optimization perspective on learning over graphs. Specifically, we posit that there exists an underlying optimization mechanism in graph structured domains that drives the formation of the observed graph structure. A sound graph formation model can help determine how the networks come into existence, which can be of fundamental importance in a variety of applications where the network architecture often influences decision making and final outcomes [4, 5]. As graph formation models are unknown in many domains, there is a conspicuous need to build learning techniques that can discover an underlying optimization mechanism. To this end, we pose and address the following questions: (i) How can one effectively discover the latent optimization model that governed the formation of an observed graph and (ii) How can one leverage such a learned model to generate, learn or infer desired properties over graphs?

Recently, deep learning over graphs [2] have successfully addressed problem of encoding complex graph information in continuous vectors for downstream applications. Advances in deep reinforcement learning (RL) [6, 7] has enabled learning of optimal policies directly from raw state information in graphs with combinatorial search spaces [8, 9, 10]. However, these techniques require large sample size, cannot scale to large graphs, or require a known domain-specific reward function, which are significant restrictions for general applications [9, 8, 11, 12, 13, 10, 14].

Present Work— In this work, we propose **GraphOpt**, an efficient and scalable maximum entropy framework that unifies a novel structured policy network with inverse reinforcement learning to learn an optimization model of the observed graph. GraphOpt is based on the key observations that (i) graph formation is a sequential process, in which the structure at any intermediate time influences the creation of specific new links in the future; and (ii) this formation phenomenon can be modeled as the

Workshop on Perception as Generative Reasoning, NeurIPS 2019, Vancouver, Canada.

outcome of a sequential decision-making process that optimizes an underlying unknown objective function. Specifically, GraphOpt learns a structured policy network that creates edges in a sequential manner to construct a graph with minimal differences in graph properties from an observed graph. As the true graph formation objective function is unknown, the construction policy is trained using a latent reward function learned via inverse reinforcement learning (IRL) (solving the inverse problem). The structured policy network uses a graph neural network to capture the complex information of a partially constructed graph in a continuous vector representation. Unlike previous works on reinforcement learning over graphs, we propose a novel *continuous latent action space* that is induced directly from data and is independent of the size of graph, thereby ensuring that GraphOpt scales to large graphs. For efficient learning, we design a maximum entropy based training procedure that leverages Soft Actor Critic [15] and Guided Cost learning [16] to learn jointly the graph construction policy and a latent reward that induces the underlying graph formation objective.

We evaluate the practical usefulness of GraphOpt in the following tasks: (i) Exposing Structure Optimization Mechanism — the ability of the learned reward function and graph construction policy to construct graphs with similar properties as that of the observed graph and (ii) Generalization Capacity — the ability of the learned components to generalize across graphs of different sizes and characteristics that share the same underlying formation mechanism of a given domain.

2 Proposed Approach

2.1 Formation Mechanism as a Markov Decision Process

The graph formation mechanism is the central focus of our work and one aspect of our work can be considered as solving the inverse problem of learning (instead of hand-designing) the utility function described in [4] using data-driven approach. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{Y}, \mathcal{X})$ denote a graph, where $\mathcal{V}$ is the set of nodes, $\mathcal{E}$ is the set of edges, $\mathcal{Y}$ is the set of edge types and $\mathcal{X}$ is the set of node features. We define a Graph Formation Markov decision process (GF-MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, P)$ as follows:

State $s_t \in \mathcal{S}$. The state of the environment s_t at time t is the partially constructed graph $G_t = (\mathcal{V}, \mathcal{E}_t, \mathcal{Y}_t, \mathcal{X})$. Initial state $s_0 = G_0$ is a graph with all nodes but no edges i.e. $\mathcal{E}_t = \emptyset$. For ease of exposition and w.l.o.g., we let $s_t = (\mathcal{V}, \mathcal{E}_t)$ represent a state in this paper.

Action $a_t \in \mathcal{A}$. Each step in sequential process involves the creation of an edge between two nodes in $\mathcal{V}$. Node feature information, represented as real-valued vectors, plays a key role in determining the compatibility of two nodes for edge creation. To capture this insight, we propose a novel *continuous latent action space* for graph structured environments that leads to data-induced continuous action space (unlike large discrete space used in previous works [17]), thereby enabling support for large graphs. Specifically, an action $a_t \in \mathcal{A}$ is defined as a 2-tuple $(\mathbf{a}^{(1)}, \mathbf{a}^{(2)})$, where each component $\mathbf{a}^{(i)} \in \mathbb{R}^d$ is interpreted as a node feature representation.

Transition Dynamics. The transition function $P(s_{t+1}|s_t, a_t)$ is defined such that an action mapped to edge (v_i, v_j) chosen at a state $s_t = (\mathcal{V}, \mathcal{E}_t)$ produces a next state $s_{t+1} = (\mathcal{V}, \mathcal{E}_t \cup (v_i, v_j))$. As our goal is to construct $\mathcal{G}'$ with similar *properties* as the observed $\mathcal{G}$, all edges are allowed for selection.

Reward $\mathcal{R}$. A key objective of our work is to learn an underlying optimization objective , that drives the the formation of the observed graph. The GF-MDP perspective gives a concrete instantiation of the latent objective: It is exactly the expected return $\mathbb{E}_{\pi_\phi, P}[\sum_{t=0}^{T} \mathcal{R}(s_t)]$ for executing a policy π_ϕ with transition function P under a latent reward function $\mathcal{R}$ evaluated at every state. In contrast to existing RL frameworks for modeling graph structured data [9], which impose restrictive domain-specific forms on the reward function, we propose to learn $\mathcal{R}$ directly from the observed graph.

2.2 GraphOpt's Neural Policy Architecture

GraphOpt operates in a challenging non-Euclidean graph environment that we address using a novel graph neural network (GNN) based structured policy network. At time step t, the GNN encodes the graph state s_t into a low dimensional representation for the policy to compute a corresponding action a_t. We first describe the action selection procedure and then outline the state encoder architecture.

2.2.1 Action Selection

We design a stochastic actor that takes as input state s_t and outputs a link formation action a_t. As link formation often depends on node features, whereby nodes with similar features have higher

probability of forming a link, we propose a novel *continuous latent action space* which induces action representations over a node feature space learned from data. Specifically, action a_t is a 2-tuple $(\mathbf{a}^{(1)}, \mathbf{a}^{(2)})$ whose components $\mathbf{a}^{(i)} \in \mathbb{R}^d$ are mapped to the node feature representations so as to select two nodes to construct an edge. Let $v \in \mathcal{V}$ denote a node and let $\mathbf{z}^v \in \mathbb{R}^d$ denote its embedding (learned in Section 2.2.2). Under a Gaussian policy π_ϕ, the next action is computed as follows:

$$[\boldsymbol{\mu}, \log(\boldsymbol{\sigma^2})] = \pi(s_t) = g_\phi(\text{Enc}_\omega(s_t)) \qquad \mathbf{a}^{(1)}, \mathbf{a}^{(2)} \sim \mathcal{N}(\boldsymbol{\mu}, \log(\boldsymbol{\sigma}^2)) \tag{1}$$

where g_ϕ is a standard two layer MLP with the policy parameters ϕ. $\text{Enc}_\omega(\cdot)$ is a state encoder for which we employ a graph neural network architecture with parameters ω (Section 2.2.2). Then we select two nodes to construct an edge using a similarity criterion: $v_i = \text{argmax}_{\mathbf{z}^v \,:\, \forall v \in V} \, \sigma\langle \mathbf{a}^{(i)}, \mathbf{z}^v \rangle$ for $i = 1, 2$, where $\langle \cdot, \cdot \rangle$ is a dot product and σ is the sigmoid function. As the mapping from continuous action vectors to nodes is external, GraphOpt is fully differentiable.

2.2.2 Structured State Encoder

During the graph formation process, the present structure of the graph may be a crucial factor that determines a new edge creation. To capture this, at each step t of the environment, we compute the state information as a function of graph structure parameterized by the graph neural network. Specifically, the state s_t is represented by a node embedding matrix $\mathbf{Z}_t \in \mathbb{R}^{n_t \times d}$. We use a p-step inductive message propagation architecture inspired from [1] to compute this matrix based on the graph constructed at any time step t, independent of the number of nodes.

2.3 Maximum Entropy Learning Procedure

Our approach comprises of three learning objectives to learn: (i) Graph construction policy π, (ii) Latent reward function $\mathcal{R}$, and (iii) State encoder network. Off-policy RL methods such as DDPG [18] often suffer from stability issues while on-policy methods such as PPO [19] suffer from poor sample efficiency. To this end, we adopt Soft-Actor-Critic (SAC) [20, 15], a max-ent variant of the actor-critic framework, and combine it with max-ent based Guided Cost Learning (GCL) objective [16] to facilitate joint learning of the graph construction policy network and the latent reward function for the graph environment. For GCL, the measured trajectories are collected in the form of permutations over the ordering of the edges in original graph. All permutations can be considered "expert" as each starts from same initial state ($\mathcal{E}_0 = \emptyset$) and end at same final state (original graph) while not containing any wrong edges in the intermediate states. During training, an epoch starts with initial state representation computed using state encoder when there is no edge between the nodes. At each step, the agent either chooses to create a new edge or repeat an already created edge. We train the policy, Q and representation networks after every few steps taken by the environment, while the reward network is trained after end of each epoch.

In contrast to generative adversarial formulations of inverse problem for imitation learning [21], which converge to an uninformative discriminator, maximum entropy inverse RL satisfies the key objective of our work by recovering a useful latent reward.

3 Experiments

We evaluate the effectiveness of our approach from the following standpoints: (i) Ability to expose the underlying structural optimization mechanism in the form of latent reward function and (ii) Generalization capacity across graph environments with varying degree of differences between them. Data specifications and metrics are made available in the Appendix Section A.1 and A.2.

3.1 Exposing Structure Optimization Mechanism

Our method jointly learns a stochastic policy to construct a graph and the latent reward function, using measured trajectories extracted from an observed graph. Consequently, our learned policy is capable of constructing edges given a set of nodes such that the constructed graph resembles the observed graph with respect to various graph properties (the learned reward function can be interpreted as composite function of such properties that are optimized during policy training). Hence, the effectiveness of GraphOpt for generating graphs with similar properties as the observed graph serves as a natural measure to evaluate its performance and we compare GraphOpt against representative state-of-art baselines for generative modeling of graph structured data.

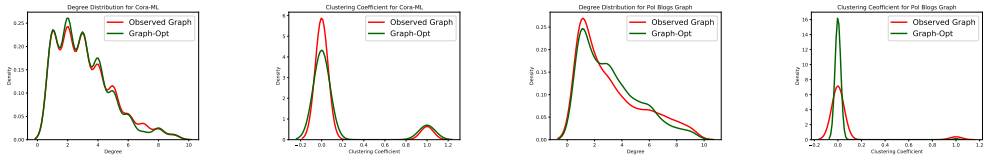
Setup. For these experiments, we use both synthetic and real world graphs that span different domains, characteristics and sizes. We use Barabasi-Albert (synthetic), Political Blogs [22] and Cora-ML [23]

(Table 2 in Appendix). For comparison, we use the following baselines: NetGan [24], Degree-Corrected Stochastic Block Model [25], Block two-level Erdos-Renyi random graph model [26] and Variational Graph AutoEncoder (VGAE) [27]. For all methods, we generate 5 graphs for evaluation and report average and deviation over various graph based statistics (Table 1). Greater similarity to the statistics of original graph (first row) is better. We also report the difference between statistics of constructed graph from that of original graph in Appendix A.4 for further interpretability.

Results. The major observation from the results in Table 1 is that GraphOpt, in the absence of a known objective function, is able to construct graphs with similar properties as the observed graph. Surprisingly, GraphOpt is able to consistently compare or exceed dedicated baselines in many cases. Specifically, Figure 1 (a-d) demonstrate the ability of Graphopt to capture intrinsic properties of graph structure which serves as an evidence of learning a useful objective function that was optimized by the construction policy. Further, many baseline methods work well (except VGAE which is purely node embedding approach) on specific metric as they are designed to optimize for a particular property, while the key objective of GraphOpt is to discover such an optimization function which provides flexibility of adapting for various properties which boosts its performance.

Table 1: Graph Based statistics for BA, Political Blogs and Cora-ML (Full results in Appendix A.4)

	Barabasi Albert			**Political Blogs**			**CORA-ML**		
Model	**Triangle Count**	**Clustering Coeff.**	**Max Degree**	**Triangle Count**	**Clustering Coeff.**	**Max Degree**	**Traingle Count**	**Clustering Coeff.**	**Max Degree**
Observed Graph	504	0.1471	33	303129	0.319	351	4890	0.2406	168
DC-SBM	269.33 ± 33.171	0.059 ± 0.057	23.66 ± 2.51	143152 ± 27723.46	0.026 ± 0.003	207.67 ± 6.65	1410 ± 74.90	0.07 ± 0.048	171 ± 16.52
BTER	262 ± 45.902	0.098 ± 0.098	22 ± 0	165299.33 ± 21975.26	0.14 ± 0.043	197 ± 2.64	2931 ± 54.61	0.04 ± 0.004	195.67 ± 24.33
VGAE	146.66 ± 45.092	0.008 ± 0.0080	30.33 ± 0.57	4354.33 ± 1290.35	0.002 ± 0.001	196.33 ± 3.21	21.33 ± 11.50	0.016 ± 0.005	9.33 ± 3.05
NetGan	344.33 ± 31.62	0.063 ± 0.054	33 ± 1.738	168913.66 ± 25054.51	0.19 ± 0.023	215 ± 13	1751 ± 105.27	0.18 ± 0.015	161 ± 4
GraphOpt	472.33 ± 20.40	0.11 ± 0.11	31.33 ± 1.52	197862 ± 11474.69	0.25 ± 0.029	221.66 ± 9.50	3938.33 ± 49.21	0.205 ± 0.013	169 ± 5.56



(a) Degree distribution (b) Clus. Coeff. distribution (c) Degree distribution (d) Clus. Coeff. distribution

Figure 1: (**a-b**) Original vs. GraphOpt: Cora-ML (**c-d**) Original vs. GraphOpt: Pol.Blogs

3.2 Generalization Capacity

One of the key outcomes of our approach is its intrinsic ability to generalize—the learned reward function is trained to capture the underlying formation mechanism of a particular domain and the policy is trained to construct graphs for this domain. We performed three generalization experiments: **Generalization over size of instances:** Train on a BA graph of 200 nodes, take an original BA graph of 1000 nodes and use that to evaluate the new graph of 1000 nodes generated using previously learned policy and reward. **Generalization across graph - full model:** Train on the Cora-ML dataset and then evaluate the full model (learned policy, reward function and representation network) on a CiteSeer graph to perform construction. **Generalization across graph - Latent Reward function:** Train GraphOpt on the Cora-ML dataset and then instantiate a new model but transfer the learned reward function to this model while the other components are untrained. We then train this new model on the CiteSeer dataset with fixed reward function. The learned framework is evaluated as before. The results are provided in Appendix A.3.

4 Conclusion

We presented a novel perspective on learning over graphs by posing it as an inverse problem of learning the underlying cause of the observed graph. Our forward model is based on reinforcement learning framework and uses a graph neural network based structured policy network to capture intricate structural dependencies. Differently from a large part of literature on reinforcement learning over graphs that optimize highly curated task-specific objective, our approach enables the recovery of underlying task-agnostic latent objective function that is generalizable across different size and characteristics of graphs that share common underlying formation mechanism. We demonstrate that the recovered function allows to learn policies that provides quantitatively superior performance on graph generation task compared to state-of-art task-specific baselines. Extending our optimization perspective to other learning tasks over structured data pose interesting future direction.

References

[1] William L. Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv:1709.05584*, 2017.

[2] Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep learning on graphs: A survey. *arXiv:1812.04202*, 2018.

[3] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[4] Aureo De Paula, Seth Richards-Shubik, and Elie Tamer. Identifying preferences in networks with bounded degree. *Econometrica*, 2018.

[5] Albert-László Barabási. Network science: Luck or reason. *Nature*, 489(7417):507, 2012.

[6] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[7] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[8] Wenhan Xiong, Thien Hoang, and William Yang Wang. Deeppath: A reinforcement learning method for knowledge graph reasoning. *arXiv preprint arXiv:1707.06690*, 2017.

[9] Jiaxuan You, Bowen Liu, Rex Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal directed molecule generation. In *NIPS*, 2018.

[10] Zhuwen Li, Qifeng Chen, and Vladlen Koltun. Combinatorial optimization with graph convolutional networks and guided tree search. In *NeurIPS*, 2018.

[11] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. 2018.

[12] Yelong Shen, Jianshu Chen, Po-Sen Huang, Yuqing Guo, and Jianfeng Gao. M-walk: Learning to walk over graphs using monte carlo tree search. In *NeurIPS*, 2018.

[13] Hanjun Dai, Elias B. Khalil, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *NIPS*, 2017.

[14] Shaileshh Venkatakrishnan Bojja, Mohammad Alizadeh, and Pramod Viswanath. Graph2seq: Scalable learning dynamics for graphs. *arXiv:1802.04948v3*, 2018.

[15] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. *arxiv:1812.05905*, 2018.

[16] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning*, pages 49–58, 2016.

[17] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018.

[18] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[19] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv:1707.06347*, 2017.

[20] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 2018.

[21] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573, 2016.

[22] Lada A. Adamic and Natalie Glance. The political blogosphere and the 2004 u.s. election: Divided they blog. In *Proceedings of the 3rd International Workshop on Link Discovery*, 2005.

[23] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 2000.

[24] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. NetGAN: Generating graphs via random walks. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[25] Brian Karrer and M.E.J. Newman. Stochastic blockmodels and community structure in networks. *Physical Review E*, 2010.

[26] C. Seshadhri, Tamara G. Kolda, and Ali Pinar. Community structure and scale-free collections of erdos-renyi graphs. *Physical Review E*, 2012.

[27] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[28] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lisa Getoor, Brian Galligher, and Eliassi-Rad. Collective classification in network data. *Ai Magazine*, 2008.

A Datasets, Metrics and further Experiment Results

A.1 Datasets

Exposing Structure Optimization and Generalization Experiments. Table 2 provide statistics and reference to the dataset used for generative experiments.

Table 2: Dataset Statistics for Construction Experiments

Graph	Nodes	Edges	Density	Avg. Degree	Source
Barabasi-Albert (BA)	100	384	0.0384	7.68	Synthetic
Political Blogs	1224	19090	0.0127	27.316	[22]
CORA-ML	2810	5429	0.001	3.898	[23]
CiteSeer	3327	4732	0.00042	2.811	[28]

A.2 Metrics

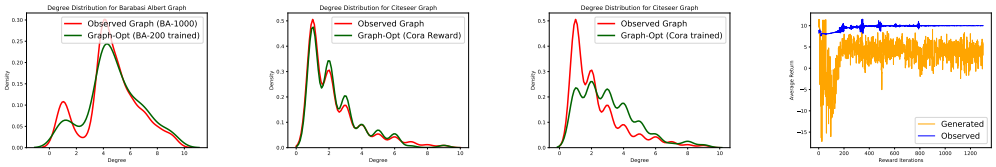
We provide several results on our constructed graphs and below we discuss some information about the metrics reported in the tables and figures:

- Graph Construction Experiments We chose the following statistics that cover various aspects of graph structure
  - Triangle Count: Number of triangles in the graph
  - Clustering Coefficient: measure of the degree to which nodes in a graph tend to cluster together.
  - Longest Connected Component (LCC): Size of the largest connected component.
  - Assortativity: Pearson correlation of degrees of connected nodes, $A = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$ where the (x_i, y_i) pairs are the degrees of connected nodes.
  - Max Degree: Maximum degree of all nodes in the graph.

In addition to the above quantitative metrics, we also report visual comparison between distributions of degree and clustering coefficient for graphs that are constructed by GraphOpt and the original observed graph. For the IRL reward curve we display plot of Average Returns vs the number of iterations which is standard empirical measure of evaluating convergence in reinforcement learning.

A.3 Generalization Experiments

One of the key outcomes of our approach is its intrinsic ability to generalize—the learned reward function is trained to capture the underlying formation mechanism of a particular domain and the policy is trained to construct graphs for this domain. Hence, it is important to evaluate how GraphOpt generalizes to variations in graphs within the same domain. We performed three generalization experiments: **Generalization over size of instances:** Train on a BA graph of 200 nodes, take an original BA graph of 1000 nodes and use that to evaluate the new graph of 1000 nodes generated using previously learned policy and reward. **Generalization across graph - full model:** Train on the Cora-ML dataset and then evaluate the full model (learned policy, reward function and representation network) on a CiteSeer graph to perform construction. **Generalization across graph - Latent Reward function:** Train GraphOpt on the Cora-ML dataset and then instantiate a new model but transfer the learned reward function to this model while the other components are untrained. We then train this new model on the CiteSeer dataset with fixed reward function. The learned framework is evaluated as before. Table 3 and Figure 2 (a-c) demonstrates GraphOpt's versatile capacity to generalize across instances within the same domain. The results demonstrate both the inductive ability of our framework and the usefulness of the learned objective towards generalization. We also demonstrate the convergence of our method for Cora Dataset in Figure 2 (d). It depicts both the expert reward curve and generated reward curve. The goal in IRL is for the policy generated curve to approach the expert curve at convergence.



(a) Degree distribution (b) Degree distribution (c) Degree distribution (d) IRL Reward Curve

Figure 2: (**a**) BA-100 (train) vs. BA-1000 (eval) (**b**) CiteSeer-original vs. CiteSeer (trained using Cora-Trained reward) **(c)** CiteSeer-original vs. CiteSeer (eval using Cora-Trained full model) **(d)** IRL reward curve

Table 3: Generalization Performance Comparison

	Triangle Count	**Clustering Coeff.**	**Max Degree**
BA-200 (train)	780	0.12	43
BA-1000 (observed)	1632	0.0407	115
BA-1000 (eval)	1470.66 ± 25.71	0.036 ± 0.0044	119.33 ± 2.081
CiteSeer Dataset	3501	0.1414	99
Cora (train)	4890	0.241	168
Citeseer (reward)	2847.66 ± 57.13	0.098 ± 0.0010	80.66 ± 1.527
Citeseer (eval)	2234 ± 58.96	0.084 ± 0.004	70.66 ± 2.081

A.4 Experiments on Exposing Structure Optimization Mechanism

In this section, we provide earlier construction results (Table 1) with two more statistics (Largest connected component and Assortativity) and different perspective. For interpretability, we report average absolute difference (in parantheses) of various graph based statistics between original and generated graph (Table 4,5,6). Smaller difference to the statistics of original graph (first row) is better.

Table 4: Graph Statistics for BA Graph

Model	**Triangle Count**	**Clustering Coeff.**	**LCC**	**Assortativity**	**Max Degree**
Observed Graph	504	0.147	100	-0.096	33
DC-SBM	269 (235)	0.060 (0.087)	73 (27)	-0.008 (0.088)	24 (9)
BTER	262 (242)	0.098 (0.049)	65 (35)	-0.013 (0.083)	22 (11)
VGAE	147 (357)	0.008 (0.139)	91 (9)	-0.007 (0.089)	30 (3)
NetGan	344 (160)	0.064 (0.083)	99 (1)	-0.036 (0.06)	33 (0)
GraphOpt	472 (32)	0.110 (0.037)	94 (6)	-0.088 (0.008)	31 (2)

Table 5: Graph Statistics for Political Blogs Graph

Model	**Triangle Count**	**Clustering Coeff.**	**LCC**	**Assortativity**	**Max Degree**
Observed Graph	303129	0.320	1222	-0.221	351
DC-SBM	143152 (159977)	0.026 (0.294)	425 (797)	-0.025 (0.196)	208 (143)
BTER	165299 (137830)	0.147 (0.173)	544 (678)	-0.020 (0.201)	197 (154)
VGAE	4354 (298775)	0.002 (0.318)	709 (513)	-0.005 (0.216)	196 (155)
NetGan	168914 (134215)	0.120 (0.2)	860 (362)	-0.166 (0.055)	215 (136)
GraphOpt	197862 (105267)	0.255 (0.065)	925 (297)	-0.173 (0.048)	222 (129)

Table 6: Graph Statistics for CORA-ML Graph

Model	**Triangle Count**	**Clustering Coeff.**	**LCC**	**Assortativity**	**Max Degree**
Observed Graph	4890	0.241	2485	-0.066	168
DC-SBM	1410 (3480)	0.076 (0.165)	2513 (55)	-0.049 (0.017)	171 (3)
BTER	2931 (1959)	0.044 (0.197)	2357 (101)	0.007 (0.073)	196 (28)
VGAE	21 (4869)	0.017 (0.224)	2485 (27)	-0.002 (0.064)	9.3 (158.7)
NetGan	1751 (3139)	0.018 (0.061)	2472 (14)	-0.068 (0.002)	161 (7)
GraphOpt	3938 (952)	0.205 (0.036)	2333 (125)	-0.059 (0.007)	1