GraphOpt: Learning Optimization Models of Graph Formation

Rakshit Trivedi [1] **Jiachen Yang** [1] **Hongyuan Zha** [1 2]

Abstract

Formation mechanisms are fundamental to the study of complex networks, but learning them from observations is challenging. In real-world domains, one often has access only to the final constructed graph, instead of the full construction process, and observed graphs exhibit complex structural properties. In this work, we propose GraphOpt, an end-to-end framework that jointly learns an implicit model of graph structure formation and discovers an *underlying optimization mechanism* in the form of a *latent* objective function. The learned objective can serve as an explanation for the observed graph properties, thereby lending itself to transfer across different graphs within a domain. GraphOpt poses link formation in graphs as a sequential decision-making process and solves it using maximum entropy inverse reinforcement learning algorithm. Further, it employs a novel continuous *latent action space* that aids scalability. Empirically, we demonstrate that GraphOpt discovers a latent objective transferable across graphs with different characteristics. GraphOpt also learns a robust stochastic policy that achieves competitive link prediction performance without being explicitly trained on this task and further enables construction of graphs with properties similar to those of the observed graph.

1. Introduction and Related Work

Learning generative mechanisms of graph-structured data is an important approach to build graph constructors for data augmentation (Chakrabarti & Faloutsos, 2006) and inference modules for downstream network analysis and prediction tasks. Such models have wide-ranging applications in several domains spanning recommendation systems (Zhang et al., 2017), biological networks (Singh & Lio, 2019), knowledge graphs (Xiao et al., 2016), social networks (Liu et al., 2016) and many more. Formation mechanisms play a fundamental role in driving the generative process of structures observed in the real-world (Barabási & Pósfai, 2016). Modeling these mechanisms is important as it could facilitate synthesis of novel graph structures for various subsequent studies such as analysing disease progression. Moreover, access to these mechanisms could help to build intelligent systems that *generalize* beyond the task of structure generation and support *transfer* to graphs beyond available observations. However, in real-world domains, such formation mechanisms are often unknown and learning them from data is challenging due to: limited or no access to the construction process (one often observes only the final graph); complex discrete nature of graph and rare availability of large collection of graph samples for learning.

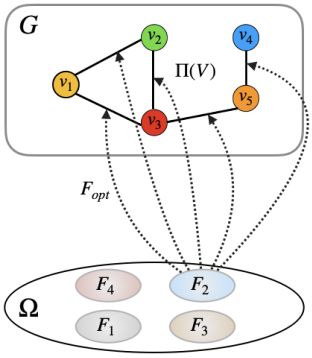


Figure 1. Ω is set of latent objective functions $\{\mathcal{F}_i\}$, any of which could lead to observed graph $\mathcal{G}$ when optimised. Our goal is to discover one such latent objective $\mathcal{F}_{\text{opt}}$ that could serve as an explanation of the observed graph properties, and optimise it to learn a graph construction procedure Π such that $\Pi(\mathcal{V})$, given node set $\mathcal{V}$, mimics the network patterns observed in $\mathcal{G}$. While $\mathcal{F}_{\text{opt}}$ may not match the unknown ground truth mechanism when one exists, it can produce an accurate Π and hence can be operationally equivalent to the true mechanism.

In this paper, we investigate the novel problem setting of discovering an underlying formation mechanism of the observed graph structure. Concretely, we propose **GraphOpt**, an end-to-end learning framework that jointly learns the forward model of graph construction and solves the inverse problem of discovering an *underlying optimization mechanism*, in the form of a *latent* objective function, that serves

[1]Georgia Institute of Technology, USA [2]Institute for Data and Decision Analytics, the Chinese University of Hong Kong, Shenzhen. Correspondence to: Rakshit Trivedi <rstrivedi@gatech.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

as an explanation for the existence of the observed graph structure. From a broader perspective of network science, GraphOpt naturally aligns more with the "optimization" viewpoint of graph formation (Newman, 2010; Papadopoulos et al., 2012; Barabási & Pósfai, 2016)—link formation is viewed as the outcome of an underlying optimization mechanism whereby decisions are based on current global state of the network. For instance, links in transportation networks appear as a result of optimizing some underlying cost function (Albert & Barabási, 2002). In contrast, most existing learning approaches for graphs are implicitly rooted in the "probabilistic" viewpoint, which models link formation in networks as random events that depend largely on local structural properties (Kumar et al., 2000; Vázquez et al., 2003). Figure 1 provides an overview of our proposition.

Formally, GraphOpt is realised as an efficient maximum entropy based reinforcement learning (Haarnoja et al., 2018; Ziebart et al., 2008) framework that models graph formation as a sequential decision-making process. It trains a novel structured policy network such that the learned stochastic policy constructs edges in a sequential manner to produce a graph with minimal deviation in a set of graph properties from an observed graph. This policy network uses a graph neural network to capture the complex information of a partially constructed graph into a continuous state representation. As the true graph formation objective function is unknown, the policy optimizes a latent reward function learned via inverse reinforcement learning (IRL) (Ziebart et al., 2008; Finn et al., 2016), which amounts to learning an optimization-based model of graph formation. Further, we propose a novel *continuous latent action space* that is independent of the size of graph, thereby allowing GraphOpt to learn over large graphs.

Traditional generative approaches include explicit probabilistic models that are carefully hand-designed to incorporate assumptions on structural properties (Robins et al., 2007; Leskovec et al., 2007; Airoldi et al., 2009; Leskovec et al., 2010; Erdos & Renyi, 1959). Such intuitive model specifications produce graphs that often exhibit disagreement with real-world graphs (Broido & Clauset, 2018; Dong et al., 2017). Recent advances in deep generative models for graphs (Li et al., 2018a; Simonovsky & Komodakis, 2018; You et al., 2018b; Kipf & Welling, 2016) address this issue by directly learning from data to be able to mimic the observed properties and produce realistic graphs. However, these techniques are either limited to learn over small graphs or require a large collection of graphs from the same distribution to achieve a desired fidelity, both of which pose great limitations on learning over real-world graphs. Further, the above techniques only facilitate graph generation but does not directly allow downstream inference tasks, which further limits their usability. A recently proposed deep generative model NetGAN (Bojchevski et al., 2018), resembles GraphOpt in being implicit model for graph construction, however, the two approaches are fundamentally different as NetGAN builds a probabilistic model of random walks over graphs and avoids learning an objective function which stands in contrast to our optimization-based framework.

We perform extensive experiments on graphs with varying properties to gauge the efficacy of GraphOpt on the following measures: (i) Can GraphOpt discover a useful and transferable latent objective for a given domain? (ii) How well does GraphOpt's construction policy generalize to downstream inference tasks? (iii) Can GraphOpt serve as an effective graph constructor useful to synthesize new graphs that exhibit structural patterns similar to the ones found in an observed graph? We comprehensively answer all three questions in the positive via experiments demonstrating effective transfer in the domain of citation graphs; competitive link prediction performance against dedicated baselines demonstrating compelling generalization properties; and consistently superior performance on graph construction experiments against strong baselines that learn from single input graph. We discuss more related works in Appendix B.

2. Proposed Approach: GraphOpt

We first elaborate on the optimization viewpoint of graph formation and how it motivates our modeling approach. Next, we formally define the problem we tackle and define the corresponding sequential decision-making process. Finally, we present architecture details of the GraphOpt framework.

2.1. Optimization Models of Graph Formation

A reasonable graph formation model can help to determine how networks come into existence, which can be fundamentally important in various applications where the network structure often influences decision making (Paula et al., 2018). While networks in certain domains (e.g. transportation (Albert & Barabási, 2002)) can be explained by an underlying optimization mechanism with a known functional form, most general networks often exhibit properties that have given rise to long-standing debates in the network science community on the true mechanisms underlying their emergence (Barabási, 2012). For instance, power laws observed in social and biological networks can be explained by the probabilistic model of preferential attachment (Barabási & Albert, 1999), but they can also be the result of an underlying optimization process in which nodes optimize between popularity and similarity when forming connections (Papadopoulos et al., 2012; D'souza et al., 2007; Fabrikant et al., 2002). For complex networks, the functional form of the objective being optimized in this process is often unknown or difficult to specify in closed form.

In this work, we rigorously investigate the optimization viewpoint and its implications on developing learning ap-

proaches for graphs. As the true underlying mechanisms are unknown, we design an algorithm that *discovers* a latent objective that is operationally equivalent to the true mechanism, in the sense that the discovered objective, when optimised, enables a suitable graph construction procedure to produce graphs with similar properties as the observed one. As both the construction procedure and the objective depend on the global information, our approach is naturally aligned with the optimization viewpoint of graph formation.

2.2. Problem Definition

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we propose a graph formation model in which the optimization of some latent objective function $\mathcal{F}_{\text{opt}}\colon \mathcal{G} \to \mathbb{R}$ drives the formation of edges in $\mathcal{E}$. $\mathcal{F}_{\text{opt}}$ may correspond to any domain-specific or generic graph property but is unknown in general. Our primary aim is to learn a graph construction procedure Π^* and discover a latent objective $\mathcal{F}_{\text{opt}}$, such that the optimization of $\mathcal{F}_{\text{opt}}$ by Π^* leads to the construction of a graph $\mathcal{G}' = \Pi^*(\mathcal{V})$ with structural properties similar to the observed graph $\mathcal{G}$, given node set $\mathcal{V}$ and initially empty edge set $\mathcal{E}_0$. Learning the construction procedure is formalized as:

$$\Pi^* = \underset{\Pi}{\operatorname{argmax}}\, \mathcal{F}_{\text{opt}}(\Pi(\mathcal{V})), \tag{1}$$

$\mathcal{F}_{\text{opt}}$ is often unknown as we only observe the final graph. To this end, we draw inspiration from inverse reinforcement learning (Ng & Russell, 2000) and formulate our objective as the following minmax optimization problem:

$$\begin{aligned} \Pi^* &= \underset{\Pi}{\operatorname{argmin}} \max_{\mathcal{F}} \left[\mathcal{F}(\mathcal{G}) - \mathcal{F}(\Pi(\mathcal{V}))\right] \\ \mathcal{F}_{\text{opt}} &= \underset{\mathcal{F}}{\operatorname{argmax}} \left[\mathcal{F}(\mathcal{G}) - \mathcal{F}(\Pi^*(\mathcal{V}))\right] \end{aligned} \tag{2}$$

where the goal is to learn jointly: (i) a latent objective $\mathcal{F}$, defined via a reward function, that assigns higher value to $\mathcal{G}$ than to all other graphs with different structural properties; (ii) a construction procedure Π, defined as the sequential execution of a policy, that constructs $\mathcal{G}'$ with minimal difference from $\mathcal{G}$ in structural properties measured by $\mathcal{F}$.

2.3. Graph Formation as a Markov Decision Process

The graph formation mechanism is the central focus of our work. Formation of real-world graphs in general is not confined to result in a connected graph. Therefore, we propose a mechanism for link formation without this constraint. 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 $G_t = (\mathcal{V}, \mathcal{E}_t, \mathcal{Y}, \mathcal{X})$. Initial state $s_0 = G_0$ is a graph with all nodes but no edges, i.e. $\mathcal{E}_t = \emptyset$. Node features $\mathcal{X}$ and edge types $\mathcal{Y}$ are optional. This definition is sufficient to describe the graph at any time t and allows for sequential construction of edges without enforcing connectivity. 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 procedure Π involves the creation of an edge between two nodes in $\mathcal{V}$. For any state s_t, information of nodes in $\mathcal{V}$, as encoded in their representations, is vital in determining the compatibility of two nodes for next edge creation. To capture this insight, we propose a novel *continuous latent action space*, whereby an action is mapped to the creation of an edge between two nodes with feature representation most similar to the action vector (Section 2.4.1). In contrast to previous RL approaches to modeling graph structured data that define a discrete action space (Bojja et al., 2018; Das et al., 2018; You et al., 2018a), our continuous latent action is independent of the size of graph, thereby facilitating scalable learning.

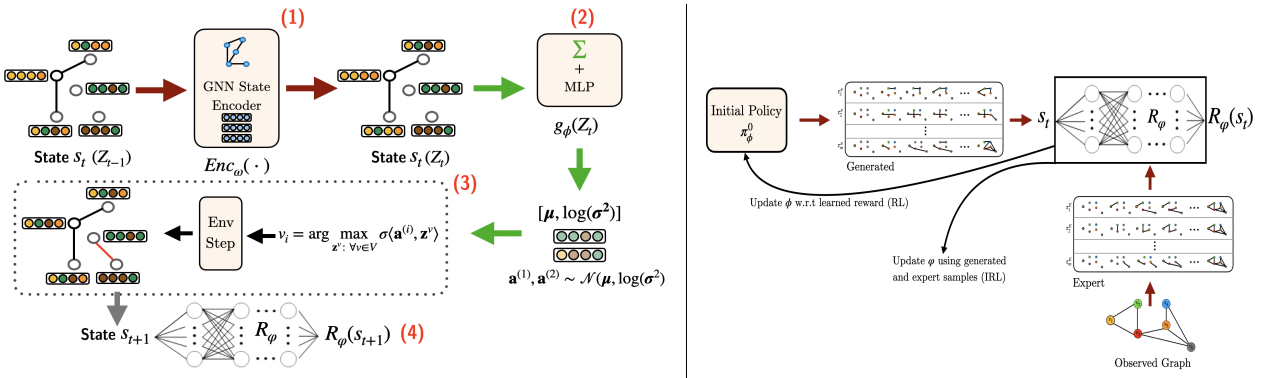
Transition Dynamics. The transition function $P(s_{t+1}|s_t, a_t)$ is defined such that an action mapped to edge (v_i, v_j) chosen at a state $s_t = (\mathcal{V}, \mathcal{E}_t)$ produces a next state $s_{t+1} = (\mathcal{V}, \mathcal{E}_t \cup (v_i, v_j))$. All edges are allowed for selection except when transition is a self-loop—i.e., both action components map to same node—which is rejected with no change in state.

Reward $\mathcal{R}$. The GF-MDP perspective gives a concrete instantiation of the key component of our model – an underlying (latent) optimization objective $\mathcal{F}_{\text{opt}}$ in (2): $\mathcal{F}_{\text{opt}}$ is exactly the expected return $\mathbb{E}_{\pi_\phi, P}[\sum_{t=0}^{T} \mathcal{R}(s_t)]$ for executing a policy $\pi_\phi(a_t|s_t)$ with transition function $P(s_{t+1}|s_t, a_t)$ under a latent reward function $\mathcal{R}$ evaluated at every state. In contrast to existing RL frameworks for modeling graph structured data (Lin et al., 2018; You et al., 2018a), which use specific forms on the reward function, we propose to learn $\mathcal{R}$ directly from the observed graph.

Optimization over $\mathcal{F}$ in (2) is the search for a reward function $\mathcal{R}$ that assigns greater value to the observed graph than all other generated graphs, which reflects the assumption of optimality of the observed graph. The optimization over Π in (2) is an optimization over π_ϕ to maximize the expected return of $\mathcal{R}$ over the formation process, which serves to construct a graph with minimal difference in structural properties, as measured by the reward, from the observed graph.

2.4. GraphOpt's Neural Policy Architecture

As GraphOpt operates in a graph-structured environment, we build a graph neural network (GNN) based structured policy network to effectively utilize the structural information. At time step t, GNN encodes graph state s_t into low dimensional representation for the policy to compute a cor-



(a) GraphOpt Neural Policy Architecture

(b) GraphOpt Learning Loop

Figure 2. Overview of GraphOpt Framework. (a) A GNN encoder maps a graph state s_t into a representation Z_t (1), which is aggregated and passed through an MLP (2), and interpreted as the mean and standard deviation of a Gaussian policy. A latent continuous action $(\mathbf{a}^{(1)}, \mathbf{a}^{(2)})$ is sampled and mapped to two nodes with most similar embeddings (3). States are evaluated by reward function R_φ (4). (b) GraphOpt interleaves policy improvement using the current reward function and reward updates using generated and expert trajectories.

responding action a_t. We first describe the action selection procedure followed by the state encoder architecture. Figure 2 (a) provides an overview of the policy network.

2.4.1. Stochastic Action Selection

We design a stochastic policy that takes as input state s_t and outputs a link formation action a_t. As outlined in GF-MDP, we introduce a novel *continuous latent action space* which induces action over node representations 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 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 2.4.2). Under a Gaussian policy π_ϕ, the next action is computed as follows:

$$\begin{aligned}[\boldsymbol{\mu}, \log(\boldsymbol{\sigma}^2)] = \pi(s_t) = g_\phi(\text{Enc}_\omega(s_t)) \\ \mathbf{a}^{(1)}, \mathbf{a}^{(2)} \sim \mathcal{N}(\boldsymbol{\mu}, \log(\boldsymbol{\sigma}^2))\end{aligned} \tag{3}$$

where g_ϕ is a two layer MLP with the policy parameters ϕ. $\text{Enc}_\omega(\cdot)$ is a state encoder that computes low-dimensional representations of graph states. For an effective encoding of state information, we employ a GNN architecture with parameters ω (Section 2.4.2). Then we select two nodes to construct an edge using a similarity criterion:

$$v_i = \underset{\mathbf{z}^v:\,\forall v\in V}{\text{argmax}}\ \sigma\langle \mathbf{a}^{(i)}, \mathbf{z}^v\rangle \quad \text{for } i = 1, 2 \tag{4}$$

where $\langle\cdot,\cdot\rangle$ is a dot product and σ is the sigmoid function. As the mapping from continuous $\mathbf{a}^{(i)}$'s to node indices is external to policy network, GraphOpt is fully differentiable.

2.4.2. 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. Structural information of graphs are often encoded into low-dimensional representations and input to the policy network (Wang et al., 2018). To achieve this, each state s_t is represented by a node embedding matrix $\mathbf{Z}_t \in \mathbb{R}^{n\times d}$ (where $n = |\mathcal{V}|$), computed using a GNN (Scarselli et al., 2009) via a p-step message propagation architecture. At initial state s_0 when $\mathcal{E} = \emptyset$, $\mathbf{Z}_0$ is initialized with node features. After adding an edge at time t, we perform p iterations of message passing across the node set to obtain $\mathbf{Z}_{t+1}$. For each iteration p, we update representation of each node as per the following equations:

Aggregate messages from the neighborhood of v:

$$\mathbf{m}_v^p \leftarrow AGG(M(\mathbf{H}_u^{p-1})),\ \ \forall u\colon \mathbf{A}_t(u, v) = 1$$

and then compute representation update for v using:

$$\mathbf{H}_v^p \leftarrow U(\mathbf{H}_v^{p-1}, \mathbf{m}_v^p)$$

where $\mathbf{A}_t$ is the adjacency matrix. We use max pooling as AGG aggregation function due to its better empirical performance. Both the message function M and the update function U are MLP. At the end of each training episode, we reset $\mathbf{Z}_t$ to initial state when resetting the environment.

3. Maximum Entropy Learning Procedure

GraphOpt contains three modules: a graph construction policy π, a latent reward function $\mathcal{R}$, and a state encoder network[1]. As the policy and latent reward are learned simultaneously in a graph structured environment, we require both stability and efficiency, which are difficult to satisfy simultaneously by off-policy methods such as DDPG (Lillicrap

[1]The state encoder network is trained by back-propagating the policy gradients to GNN parameters in an end-to-end manner.

Algorithm 1 GraphOpt Algorithm

1: **procedure** GRAPHOPT
2: **Input:** Empty trajectories list $\mathcal{T}_{gen}$, replay buffer B
3: node representation matrix $\mathbf{Z}_0$, parameters
4: $\psi, \phi, \theta, \omega, \varphi$.
5: **for** each epoch **do**
6: Reset adj. matrix $\mathbf{A}_0 = \mathbf{0}$
7: *# Using state encoder,*
8: Reset state to $s_0 = \text{Enc}_\omega(\mathbf{Z}_0, \mathbf{A}_0)$
9: Initialize trajectory $\tau = \{s_0\}$
10: **for** each environment step **do**
11: $(v_1, v_2) \leftarrow a_t \sim \pi_\phi(a_t|s_t)$ using Eq 3, 4
12: Update $\mathbf{A}_{t+1} \leftarrow \mathbf{A}_t[v_1, v_2] = 1$.
13: Update $s_{t+1} = \text{Enc}_\omega(s_t, A_{t+1})$
14: Compute $r_t = R_\varphi(s_{t+1})$
15: $B \leftarrow B \cup \{(s_t, a_t, r_t, s_{t+1}, A_{t+1})\}$
16: Update trajectory $\tau \leftarrow \text{concat}(\tau, s_{t+1})$
17: Train_Policy $(B, \psi, \phi, \theta, \bar{\psi}, \omega)$ *# Alg 2*
18: **If** each edge in G_t is repeated k times or
19: max_path_length reached **then**
20: reset episode, store τ in $\mathcal{T}_{gen}$,
21: and start new trajectory $\tau = \{s_0\}$
22: **end for**
23: Collect trajectories $\mathcal{T}_{meas}$ (expert)
24: Use $\mathcal{T}_{meas}$ and $\mathcal{T}_{gen}$ to update reward
25: estimator R_φ
26: **end for**
27: **end procedure**

et al., 2015) and on-policy methods such as PPO (Schulman et al., 2017). To this end, we adopt Soft-Actor-Critic (SAC) (Haarnoja et al., 2018), a maximum entropy variant of the actor-critic framework (Konda & Tsitsiklis, 2000), and combine it with maximum entropy based Inverse Optimal Control (IOC) objective (Finn et al., 2016). We build a unified training pipeline that optimizes following objectives:

(a) **Soft Q-function.** SAC trains a function $Q_\theta(s_t, a_t)$ on off-policy experiences by minimizing the Bellman residual

$$J_Q(\theta) = \mathbb{E}_{(s_t,a_t)\sim B}\big[\big(Q_\theta(s_t, a_t) - \hat{Q}(s_t, a_t)\big)^2/2\big]$$

where $\hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma\mathbb{E}_{s_{t+1}\sim p}[V_{\bar{\psi}}(s_{t+1})]$. Value function $V_{\bar{\psi}}$ is implicitly defined by parameters of $Q_\theta(s, a)$(Haarnoja et al., 2018, Equation 3).

(b) **Policy Network.** The policy network $\pi_\phi(a_t|s_t)$ is trained using the following objective function:

$$J_\pi(\phi) = \mathbb{E}_{s_t\sim B, \epsilon_t\sim\mathcal{N}}[\alpha \log \pi_\phi(a_t|s_t) - Q_\theta(s_t, a_t)].$$

Following (Haarnoja et al., 2018), we also use a reparameterization trick with a neural network transformation as: $a_t = f_\phi(\epsilon_t; s_t)$ that results in low variance estimator.

(c) **Reward function.** $R_\varphi(s_t)$ is learned using the inverse optimal control objective: $J_R(\varphi) := -\frac{1}{N}\sum_{\tau_i\in\mathcal{T}_{\text{meas}}} \mathcal{R}_\varphi(\tau_i) + \log\big(\frac{1}{M}\sum_{\tau_j\in\mathcal{T}_{\text{gen}}} z_j \exp(\mathcal{R}_\varphi(\tau_j))\big)$ where $\mathcal{T}_{gen}$ is the set of link formation trajectories obtained from the learned policy and $\mathcal{T}_{meas}$ is the set of link formation trajectories obtained from the observed graph. These measured trajectories are collected by accumulating edges from different permutations over the ordering of edges in the original graph. All permutations can be considered "expert" trajectories as each starts from same initial state ($\mathcal{E}_0 = \emptyset$) and contains only true edges seen in the observed graph.

Algorithm 1 outlines the complete set of steps that are used for end-to-end training of GraphOpt. An epoch starts with initial state representation computed using state encoder when there is no edge between the nodes (line 4-7). For every step in the environment, the policy either creates a new edge or repeats an existing edge (line 8). It receives a reward based on current reward function and the state of the environment is updated along with replay buffer and current trajectory information (line 9-13). We train the policy network, Q-network, and state encoder after every few steps taken by the environment (line 15). If all existing edges have repeated k times, the episode ends (line 16-19), the environment is reset and new trajectory τ is initialized. The reward network is trained after end of each epoch (line 21-23). Appendix A details the gradient updates. In contrast to generative adversarial approaches to (2) for imitation learning (Ho & Ermon, 2016), which converge to an uninformative discriminator, and in contrast to behavioral cloning (Torabi et al., 2018), which does not provide an explanatory mechanism, maximum entropy IRL satisfies the key objective of our work by recovering a useful latent reward. Figure 2 (b) provides an overview of our algorithm.

4. Experiments

In this section, we aim to answer the following questions to evaluate the efficacy of our approach:

(i) Can GraphOpt discover a useful latent objective that is operationally equivalent to some underlying mechanism of a graph-structured domain, and thereby *transferable* to an unseen graph in that domain? The success and necessity of transfer is shown by how well the objective discovered on a source graph facilitates construction when optimised by a new policy on a target graph, in contrast to directly running (i.e., without fine-tuning) the trained policy on the target.

(ii) How well does GraphOpt's construction policy, learned by optimizing the discovered objective, *generalize* to downstream inference tasks? Here, we turn to the classical problem of link prediction in graphs, which requires the construction policy to generalize to predict hidden links with high accuracy, and—more crucially—perform a link prediction task for which it was not explicitly trained. We further

stress test GraphOpt's generalization capacity by deploying a trained policy on unseen target graph environments of different size and characteristics, without fine-tuning.

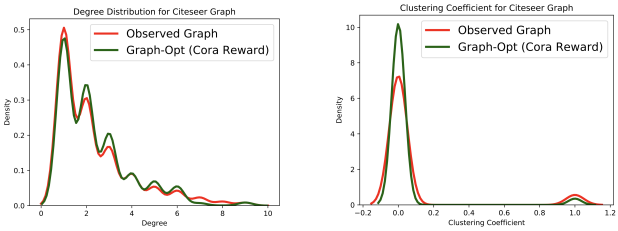
(iii) Can GraphOpt serve as an effective graph constructor to synthesize new graphs that exhibit structural patterns similar to those in an observed graph? We assess the performance of the learned construction policy by deploying it on the full set of training nodes and generating new graphs, analogous to standard practice in reinforcement learning (Sutton & Barto, 2018). GraphOpt's stochastic policy avoids copying the observed graph while preserving the statistical properties.

To the best of our knowledge, no single baseline can do all three tasks. Hence we compare GraphOpt with task-specific baselines for (ii) and (iii) and follow standard procedure in the IRL literature to report performance for (i).

Training. All experiments begin by using the observed graph to learn the construction policy and latent reward function via Algorithm 1. A key advantage of using SAC as the base RL algorithm is that it largely eliminates the need for per-task hyperparameter tuning. To encourage creation of new edges during training, we terminate an episode when the number of repeated creations of *each* existing edge reaches a threshold k, which signifies that the policy has lost the ability to explore further. We provide more details on other training configurations in Appendix C.4.

4.1. Discovering Transferable Latent Objective

For this experiment, we use two citation graphs: Cora-ML as a source environment and Citeseer as a target environment. We train on Cora-ML to discover a latent reward function, which is then transfered to train a new policy network from scratch on CiteSeer. While training on Citeseer, the reward function remains fixed and is not further trained.



(a) Degree distribution (b) Clus Coeff distribution

Figure 3. Degree and Clus. Coeff. distribution of graph constructed using the policy learned on CiteSeer, while optimizing the objective transferred from training on Cora-ML dataset.

Table 1. Transfer Performance Comparison

	Triangle Count	Clustering Coeff.	Max Degree
Cora-ML (train)	4890	0.241	168
CiteSeer (observed)	3501	0.1414	99
CiteSeer (reward)	2847.66 ± 57.13	0.098 ± 0.0010	80.66 ± 1.527
CiteSeer (direct)	2234 ± 58.96	0.084 ± 0.004	70.66 ± 2.081

After training, we input CiteSeer's node set with an empty edge set to the model and run the evaluation policy to construct edges. We collect 3 graphs and report mean and standard deviation for graph based statistics representing network patterns of the generated graphs. Table 1 and Figure 3 demonstrates that optimising the transferred objective, GraphOpt learns an effective policy to construct edge topologies that results in similar network patterns as observed in Citeseer. In contrast, the poor performance of the Cora-trained policy when directly deployed on CiteSeer without training with the transferred objective ("CiteSeer (direct)" in Table 1 and Figure 4b) shows that the discovered objective is important for transfer. This experiment demonstrates GraphOpt's effectiveness in discovering a useful latent objective that serves as an explanation for the formation of observed network patterns across citation graphs, thereby lending itself to transfer across graphs within this domain.

4.2. Policy Generalization to Prediction Task

We first discuss the task of link prediction, which demonstrates GraphOpt's ability to learn a construction policy that generalizes to unseen task for which it was not explicitly trained. We then show the performance of the policy to generalize to new graphs of different sizes and characteristics.

4.2.1. Link Prediction

Setup. We conduct link prediction experiments on a variety of graphs from both non-relational and relational domains[2]. We compare our performance with both explicit baselines that employ a dedicated hand-designed link prediction objective and implicit models that generalize to the link prediction task without using explicit link prediction objectives during training; GraphOpt falls under the latter category. For all experiments, we follow the protocol in (Zhang & Chen, 2018) by randomly removing 10% of edges to form a held-out test set and randomly sampling the same number of nonexistent links to form negative test samples. Training is then performed on the remaining graph. For relational graphs, we include edge type as an extra feature in the message passing scheme in the state encoder.

After training, we provide the observed graph as initial state and run the policy to assess how well it can predict hidden edges. For non-relational baselines, we label each edge from the test set as 1 if the policy created it and 0 otherwise, and compare with true labels to report AUC (Area under curve) and AP (average precision). For relational baselines, for a test triple $((e_s, r, e_o))$, we collect all the edges that are created for tuple (e_s, r) and rank them in order of creation to report MRR and HITS@10—this signifies policy's preference to create one test edge over another. We also perform link prediction experiments to assess the usefulness of node representations learned by our state encoder.

[2]Table 7 and 8 in Appendix C.1 provide dataset details

Table 2. Link Prediction on non-relational data: (*) is used to signify better performer amongst GraphOpt and method with implicit objective. Bold numbers are best two performers overall.

	Cora-ML		Political Blogs		E. Coli	
	AUC	**AP**	**AUC**	**AP**	**AUC**	**AP**
VGAE	94.70	96.10	92.60	93.44	93.22	93.10
Node2Vec	91.12	91.78	87.22	85.51	79.99	74.32
NetGAN	94.20*	95.22*	**95.51***	90.00	93.17	94.50
SEAL	**97.21**	**97.99**	95.32	**96.10**	**97.12**	**97.50**
GraphOpt-Policy	93.50	94.87	92.21	92.33*	94.43*	95*
GraphOpt-Embed	**96.21**	**96.66**	**95.50**	**95.32**	**97.20**	**97.44**

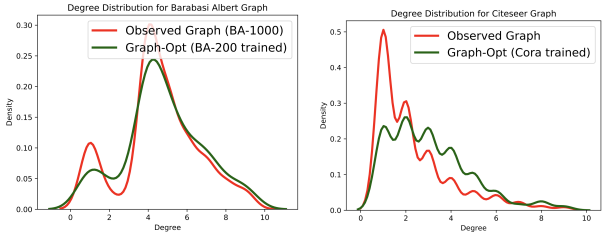
Table 3. Link Prediction performance on relational data: (*) is used to signify better performer amongst GraphOpt and RL method with +1/-1 reward. Bold numbers indicate best two performers overall.

	Kinship		FB15K-237		WN18RR	
	MRR	**H@10**	**MRR**	**H@10**	**MRR**	**H@10**
ConvE	**87.1**	**98.1**	**43.5**	**62.2**	**44.9**	**54**
NeuralLP	61.9	91.2	22.7	34.8	46.3	65.7
Reward Shaping	**87.8**	**98.2**	**40.7**	56.4	47.2	54.2
Minerva	72.0	92.4*	29.3	45.6	44.8*	51.3
GraphOpt-Policy	82.2*	92.33	33.12*	53.22*	44.2	53.6*
GraphOpt-Embed	84	96.12	39.66	**58.51**	**47.3**	**58.43**

Performance. Tables 2 and 3 show that GraphOpt's link prediction performance surpasses implicit baselines on most datasets. It is highly competitive with NetGAN (non-relational),which uses a GAN based objective, and shows significant improvement over RL based Minerva (relational), which uses LSTM encoder for state representation and a fixed reward value of +1/-1 for each step. Superior prediction performance demonstrates that GraphOpt learns a model with strong generalization capacity. GraphOpt's success in this aspect can be attributed to the combination of a stochastic policy that encourages exploration during training and the ability of the GNN to encode state representations that generalize to inference time. As a tradeoff for its greater generality, GraphOpt does not have the luxury of domain-specific architectures or objectives; hence its competitive but often slightly worse performance against state-of-the-art dedicated link prediction baselines is not surprising. However, GraphOpt's comparable performance demonstrates its greater potential for domains where an objective function is not known *a priori* and hand-designing an objective is difficult—this opens up exciting avenues of research for improvement. Surprisingly, the embedding based prediction shows strong performance, often surpassing baselines on various datasets (last row in Table 2 and Table 3). This demonstrates that GraphOpt learns a representation network that can be independently leveraged to perform various downstream tasks. As the encoder is trained in the same computational graph as the policy, through optimising the discovered latent objective, this further supports the usefulness of the discovered objective.

Table 4. 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
Cora (train)	4890	0.241	168
CiteSeer (observed)	3501	0.1414	99
Citeseer (eval)	2234 ± 58.96	0.084 ± 0.004	70.66 ± 2.081



(a) BA-200 trained policy (b) Cora trained policy

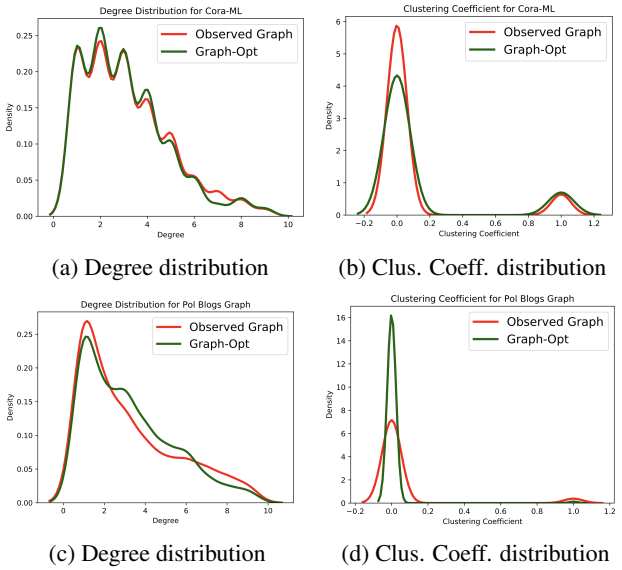
Figure 4. Policy Transfer across different size (Barabasi-Albert graph) and different graph (Cora-ML→Citeseer).

4.2.2. Evaluating Policy Performance on Unseen Environments

We now focus on the performance of the construction policy when trained on a source graph and deployed on an unseen target graph without further training[3]. Specifically, we investigate two aspects of this direct policy transfer:

(i) Transfer from small to large graph from same distribution. We train GraphOpt on a source BA graph of 200 nodes. Then we input the node set of a target BA graph with 1000 nodes to the learned policy and run it without further training to generate 3 graphs. Figure 4a and Table 4 (top 3 rows) demonstrate that generated graphs exhibit similar properties to the BA graph of 1000 nodes. This suggests that the learned construction policy can be used to generate synthetic graphs of larger size than the training graph, while preserving the underlying structural properties. *(ii) Direct transfer from source to target graph in the same domain.* We return to the earlier experiment on citation graphs, but this time transferring the trained policy from Cora-ML to run on the CiteSeer node set, without using the learned objective. As expected, Table 4 (bottom 3 rows) and Figure 4b show that the policy mostly fails to construct similar patterns as observed in the original Citeseer, but it does approximate some patterns (e.g. max degree) well. This supports the necessity of using GraphOpt's discovered reward for transfer and suggests room for further investigation.

[3]For these experiments, we are only interested in evaluating the learned policy on a target environment; hence the learned reward is not used on the target graph.



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

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

Figure 5. (**a-b**) Original vs. GraphOpt: Cora-ML (**c-d**) Original vs. GraphOpt: Pol.Blogs

4.3. Synthesizing Graphs via Learned Generative Mechanism

Setup. We evaluate GraphOpt's ability to synthesize new graphs after learning to construct from an observed graph. We use both synthetic and real world graphs that span different domains, characteristics and sizes (Table 6 in Appendix C.1). All graphs are undirected. For training, we use the complete observed graph. For evaluation, we provide the node set of the input graph and empty edge set and run the trained policy to construct edges. GraphOpt learns from a single large graph and hence we compare with strong baselines with similar setting (details in Appendix C.2). For all methods, we generate 3 graphs for evaluation and report mean and standard deviation of percentage error of graph based statistics between observed and generated graphs (Table 5). For GraphOpt, we run the evaluation policy up to either the original termination condition or a multiple of actual number of edges in observed graph, whichever is earlier. We follow reported stopping criteria for baselines.

Performance. Table 5 demonstrates that GraphOpt learns a construction policy that effectively captures structural patterns in the observed network and constructs graphs with similar properties to the observed graph. GraphOpt registers consistent and significantly superior performance across all datasets and against all baselines, which can perform well on some but not all metrics as they model specific statistics (except NetGAN). BTER recovers clustering coefficient statistics well but struggles on others; DC-SBM recovers Max Degree better than others. Further, Figure 5 demonstrates the ability of Graphopt to capture intrinsic properties of graph structure. Our stochastic policy ensures that generated graphs are not merely copies of the observed graph, which is further substantiated by link prediction experiments in Section 5.2. These performance characteristics are also visible for NetGAN, which performs well in general across all metrics and datasets. However, our superior performance can be attributed to the following differences: (i) Our construction policy optimises a useful latent objective, whereas NetGAN's generator imitates the given graph by optimizing against an eventually uninformative discriminator; (ii) Our use of GNN captures better structural properties to provide rich state information to the policy network, in contrast to LSTM based path processing in NetGAN; (iii) GraphOpt allows construction of disconnected components, often found in most real-world graphs such as in these datasets. Given these properties, we envision the use of GraphOpt as a graph constructor that ingests real-world graphs and generates synthetic versions to enrich graph repositories (Liu et al., 2020) with large-scale benchmark test sets.

We provide more details on baselines/metrics used for experiments in Appendix C.2 and Appendix C.5 respectively.

5. Conclusion

In this work, we investigate a novel setting for learning over graphs that is motivated by the optimization perspective of graph formation in network science. Our novel optimization-based learning framework, GraphOpt, models graph formation as a sequential decision-making process, learns a forward model of graph construction, and discovers a latent objective that is operationally equivalent to some underlying mechanism that could explain the formation of edges in observed graph. GraphOpt employs a novel combination of structured policy network, continuous latent action space and inverse reinforcement learning. Empirically, GraphOpt discovers a latent objective and a robust stochastic policy that transfer across graphs with different characteristics, exhibit competitive generalization for link prediction task and enable construction of graphs with similar properties as that of the observed graph. We believe that our investigation of the optimization-based perspective on network formation stemming from the wider debate in network science literature and its implications for building sophisticated models to learn effectively from graph-structured observations, coupled with the versatility of our proposed approach, would benefit the graph learning community and open exciting avenues for future research.

Acknowledgements

We thank the anonymous reviewers for critical feedback that helped us to improve the clarity and precision of our presentation. This work was supported by NSF IIS 1717916. Part of the work done by Hongyuan Zha is supported by Shenzhen Institute of Artificial Intelligence and Robotics for Society, and Shenzhen Research Institute of Big Data.

Table 5. Percent deviation of graph statistics for generated graph from observed one (lower is better). First row displays the actual statistics of the observed graph. Results for more graphs and more metrics for generated graphs are available in Appendix D.1.

	Barabasi Albert			Political Blogs			CORA-ML		
Model	**Triangle Cnt.**	**Clust. Coeff.**	**Max Degree**	**Triangle Cnt.**	**Clust. Coeff.**	**Max Degree**	**Traingle Cnt.**	**Clust. Coeff.**	**Max Degree**
Observed Graph	504	0.1471	33	303129	0.319	351	4890	0.2406	168
DC-SBM	46.56 ± 6.58	59.44 ± 7.11	28.29 ± 7.63	52.78 ± 9.15	91.73 ± 1.18	40.86 ± 1.89	71.17 ± 1.53	68.25 ± 20.16	6.94 ± 5.40
BTER	48.02 ± 9.11	33.20 ± 1.28	33.33 ± 0	45.47 ± 7.25	54.17 ± 13.57	43.87 ± 0.75	40.06 ± 1.17	81.66 ± 1.74	16.47 ± 14.49
VGAE	70.89 ± 8.95	94.40 ± 0.81	8.08 ± 1.75	98.56 ± 0.44	99.32 ± 0.55	44.06 ± 0.92	99.56 ± 0.24	93.10 ± 2.11	94.44 ± 1.82
NetGAN	31.68 ± 6.28	40.69 ± 4.27	$\mathbf{4.04 \pm 1.74}$	44.28 ± 8.27	37.55 ± 7.2	$\mathbf{38.75 \pm 3.70}$	64.19 ± 2.15	41.12 ± 18.82	4.17 ± 2.38
GraphOpt	$\mathbf{6.28 \pm 4.05}$	$\mathbf{25.52 \pm 8.25}$	$\mathbf{5.05 \pm 4.63}$	$\mathbf{34.73 \pm 3.79}$	$\mathbf{20.34 \pm 9.1}$	$\mathbf{36.85 \pm 2.71}$	$\mathbf{19.46 \pm 1.01}$	$\mathbf{14.63 \pm 5.78}$	$\mathbf{2.58 \pm 1.24}$

References

Abbeel, P. and Ng, A. Y. Apprenticeship learning via inverse reinforcement learning. In *ICML*, 2004.

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

Airoldi, E. M., Blei, D. M., Fienberg, S. E., and Xing, E. P. Mixed membership stochastic blockmodels. In *Advances in Neural Information Processing Systems 21*. 2009.

Albert, R. and Barabási, A.-L. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 2002.

Barabási, A.-L. Network science: Luck or reason. *Nature*, 2012.

Barabási, A.-L. and Albert, R. Emergence of scaling in random networks. *science*, 1999.

Barabási, A.-L. and Pósfai, M. *Network science*. Cambridge University Press, 2016.

Bojchevski, A., Shchur, O., Zügner, D., and Günnemann, S. Netgan: Generating graphs via random walks. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Bojja, S. V., Alizadeh, M., and Viswanath, P. Graph2seq: Scalable learning dynamics for graphs. *arXiv:1802.04948v3*, 2018.

Broido, A. D. and Clauset, A. Scale-free networks are rare. In *Nature Communications*, 2018.

Cao, N. D. and Kipf, T. Molgan: An implicit generative model for small molecular graphs. *arXiv:1805.11973*, 2018.

Chakrabarti, D. and Faloutsos, C. Graph mining: Laws, generators, and algorithms. *ACM Comput. Surv.*, 2006.

Dai, H., Khalil, E. B., Zhang, Y., Dilkina, B., and Song, L. Learning combinatorial optimization algorithms over graphs. In *NIPS*, 2017.

Das, R., Dhuliawala, S., Zaheer, M., Vilnis, L., Durugkar, I., Krishnamurthy, A., Smola, A., and McCallum, A. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. 2018.

Dettmers, T., Minervini, P., Stenetorp, P., and Riedel, S. Convolutional 2d knowledge graph embeddings. In *AAAI*, 2018.

Dong, Y., Johnson, R. A., Xu, J., and Chawla, N. V. Structural diversity and homophily: A study across more than one hundred big networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017.

D'souza, R. M., Borgs, C., Chayes, J. T., Berger, N., and Kleinberg, R. D. Emergence of tempered preferential attachment from optimization. *Proceedings of the National Academy of Sciences*, 2007.

Erdos, P. and Renyi, A. On random graphs i. *Publicationes Mathematicae (Debrecen)*, 1959.

Fabrikant, A., Koutsoupias, E., and Papadimitriou, C. H. Heuristically optimized trade-offs: A new paradigm for power laws in the internet. In *International Colloquium on Automata, Languages, and Programming*, pp. 110–122. Springer, 2002.

Finn, C., Levine, S., and Abbeel, P. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning*, pp. 49–58, 2016.

Fox, R., Pakman, A., and Tishby, N. Taming the noise in reinforcement learning via soft updates. In *UAI*, 2016.

Geses, G. A., Biswas, R., Alam, M., and Sack, H. A survey on knowledge graph embeddings with literals: Which model links better literal-ly? *arXiv:1910.12507*, 2019.

Grover, A. and Leskovec, J. Node2vec: Scalable feature learning for networks. In *KDD*, 2016.

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. Reinforcement learning with deep energy-based policies. In *ICML*, 2017.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., and Levine, S. Soft actor-critic algorithms and applications. *arxiv:1812.05905*, 2018.

Hamilton, W. L., Ying, R., and Leskovec, J. Representation learning on graphs: Methods and applications. *arXiv:1709.05584*, 2017.

Ho, J. and Ermon, S. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pp. 4565–4573, 2016.

Karrer, B. and Newman, M. Stochastic blockmodels and community structure in networks. *Physical Review E*, 2010.

Kipf, T. N. and Welling, M. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

Kok, S. and Domingos, P. Statistical predicate invention. In *ICML*, 2007.

Konda, V. R. and Tsitsiklis, J. N. Actor-critic algorithms. In *Advances in neural information processing systems*, pp. 1008–1014, 2000.

Kumar, R., Raghavan, P., Rajagopalan, S., Sivakumar, D., Tompkins, A., and Upfal, E. The web as a graph. In *Proceedings of the nineteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 1–10, 2000.

Leskovec, J., Kleinberg, J., and Faloutsos, C. Graph evolution: Densification and shrinking diameters. *ACM Trans. Knowl. Discov. Data*, 2007.

Leskovec, J., Chakrabarti, D., Kleinberg, J., Faloutsos, C., and Ghahramani, Z. Kronecker graphs: An approach to modeling networks. *J. Mach. Learn. Res.*, 2010.

Levine, S. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arxiv:1805.00909*, 2018.

Li, Y., Vinyals, O., Dyer, C., Pascanu, R., and Battaglia, P. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018a.

Li, Z., Chen, Q., and Koltun, V. Combinatorial optimization with graph convolutional networks and guided tree search. In *NeurIPS*, 2018b.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Lin, X. V., Socher, R., and Xiong, C. Multi-hop knowledge graph reasoning with reward shaping. *arxiv:1808.10568*, 2018.

Liu, B., Hu, W., Zitnik, M., and Leskovec, J. Open graph benchmark. *To appear*, 2020.

Liu, L., Cheung, W. K., Li, X., and Liao, L. Aligning users across social networks using network embedding. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 2016.

McCallum, A. K., Nigam, K., Rennie, J., and Seymore, K. Automating the construction of internet portals with machine learning. *Information Retrieval*, 2000.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529, 2015.

Newman, M. *Networks: An Introduction*. OUP Oxford, 2010.

Ng, A. Y. and Russell, S. Algorithms for inverse reinforcement learning. In *in Proc. 17th International Conf. on Machine Learning*, pp. 663–670. Morgan Kaufmann, 2000.

Papadopoulos, F., Kitsak, M., Serrano, M. Á., Boguná, M., and Krioukov, D. Popularity versus similarity in growing networks. *Nature*, 2012.

Paula, A. D., Richards-Shubik, S., and Tamer, E. Identifying preferences in networks with bounded degree. *Econometrica*, 2018.

Rawlik, K., Toussaint, M., and Vijayakumar, S. On stochastic optimal control and reinforcement learning by approximate inference (extended abstract). In *IJCAI*, 2013.

Robins, G., Pattison, P., Kalish, Y., and Lusher, D. An introduction to exponential random graph (p*) models for social networks. *Social Networks*, 2007.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Transactions on Neural Networks*, 2009.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv:1707.06347*, 2017.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad. Collective classification in network data. *Ai Magazine*, 2008.

Seshadhri, C., Kolda, T. G., and Pinar, A. Community structure and scale-free collections of erdos-renyi graphs. *Physical Review E*, 2012.

Shen, Y., Chen, J., Huang, P.-S., Guo, Y., and Gao, J. M-walk: Learning to walk over graphs using monte carlo tree search. In *NeurIPS*, 2018.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D. Mastering the game of go without human knowledge. *Nature*, 2017.

Simonovsky, M. and Komodakis, N. Graphvae: Towards generation of small graphs using variational autoencoders. *arxiv:1802.03480*, 2018.

Singh, V. and Lio, P. Towards probabilistic generative models harnessing graph neural networks for disease-gene prediction. *arxiv:1907.05628*, 2019.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pp. 1057–1063, 2000.

Torabi, F., Warnell, G., and Stone, P. Behavioral cloning from observation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 4950–4957, 2018.

Toussaint, M. Robot trajectory optimization using approximate inference. In *ICML*, 2009.

Toutanova, K., Chen, D., Pantel, P., Poon, H., Choudhury, P., and Gamon, M. Representing text for joint embedding of text and knowledge bases. In *EMNLP*, 2015.

Trouillon, T., Welbl, J., Riedel, S., Gaussier, E., and Bouchard, G. Complex embeddings for simple link prediction. In *ICML*, 2016.

Vázquez, A., Flammini, A., Maritan, A., and Vespignani, A. Modeling of protein interaction networks. *Complexus*, 1 (1):38–44, 2003.

Wang, H., Wang, J., Wang, J., Zhao, M., Zhang, W., Zhang, F., Xie, X., and Guo, M. Graphgan: Graph representation learning with generative adversarial nets. *1711.08267*, 2017.

Wang, T., Liao, R., Ba, J., and Fidler, S. Nervenet: Learning structured policy via graph neural networks. In *ICLR*, 2018.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 1992.

Xiao, H., Huang, M., and Zhu, X. Transg : A generative model for knowledge graph embedding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistic*, 2016.

Xiong, W., Hoang, T., and Wang, W. Y. Deeppath: A reinforcement learning method for knowledge graph reasoning. *arXiv preprint arXiv:1707.06690*, 2017.

Yang, F., Yang, Z., and Cohen, W. W. Differentiable learning of logical rules for knowledge base completion. *arxiv:1702.08367*, 2017.

You, J., Liu, B., Ying, R., Pande, V., and Leskovec, J. Graph convolutional policy network for goal directed molecule generation. In *NIPS*, 2018a.

You, J., Ying, R., Ren, X., Hamilton, W. L., and Leskovec, J. Graphrnn: Generative realistic graphs with deep auto-regressive mdoels. *arXiv preprint arXiv:1802.08773*, 2018b.

Zhang, M. and Chen, Y. Link prediction based on graph neural networks. *arxiv:1802.09691*, 2018.

Zhang, S., Yao, L., and Sun, A. Deep learning based recommender system: A survey and new perspectives. *arXiv:1707.07435*, 2017.

Zhang, Z., Cui, P., and Zhu, W. Deep learning on graphs: A survey. *arXiv:1812.04202*, 2018.

Ziebart, B. D. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. PhD thesis, Machine Learning Department, Carnegie Mellon University, 2010.

Ziebart, B. D., Maas, A., Bagnell, J., and Dey, A. K. Maximum entropy inverse reinforcement learning. In *AAAI*, 2008.