Reinforcement Learning on Graph: A Survey

Mingshuo Nie[1], Dongming Chen[1]* and Dongqi Wang[1].
1. Software College, Northeastern University, Shenyang 110169, Liaoning, China.
* Corresponding author: chendm@mail.neu.edu.cn

Abstract

Graph mining tasks arise from many different application domains, ranging from social networks, transportation, E-commerce, etc., which have been receiving great attention from the theoretical and algorithm design communities in recent years, and there has been some pioneering work using the hotly researched Reinforcement Learning (RL) techniques to address graph data mining tasks. However, these graph mining algorithms and RL models are dispersed in different research areas, which makes it hard to compare different algorithms with each other. In this survey, we provide a comprehensive overview of RL models and graph mining and generalize these algorithms to Graph Reinforcement Learning (GRL) as a unified formulation. We further discuss the applications of GRL methods across various domains and summarize the method description, open-source codes, and benchmark datasets of GRL methods. Finally, we propose possible important directions and challenges to be solved in the future. This is the latest work on a comprehensive survey of GRL literature, and this work provides a global view for scholars as well as a learning resource for scholars outside the domain. In addition, we create an online open-source for both interested scholars who want to enter this rapidly developing domain and experts who would like to compare GRL methods.

Keywords: Graph reinforcement learning; Graph mining; Reinforcement learning; Graph neural networks

1 Introduction

The recent success of RL has solved challenges in different domains such as robotics [1], games [2], Natural Language Processing (NLP) [3], etc. RL addresses the problem of how agents should learn to take actions to maximize cumulative reward through interactions with the environment [4]. The rapid development of RL in cross-disciplinary domains has motivated scholars to explore novel RL models to address real-world applications, e.g., financial and economic [5, 6], biomedical [7, 8], and transportation [9-11]. On the other hand, many real world data can be represented with graphs, and data mining for graph structure has received extensive research, such as link prediction [12, 13], node classification [14], and graph classification [15]. Various strategies have been proposed by scholars to implement graph mining, including novel information aggregation functions [16], graph structure pooling [17], and neighborhood sampling methods [12]. With the increasing graph scale and the

continuous development of RL methods in recent years, scholars are focusing on combining graph mining tasks with RL, and there is increasing interest in addressing decision problems arising in graph mining tasks with powerful RL methods [14, 15, 18-20].

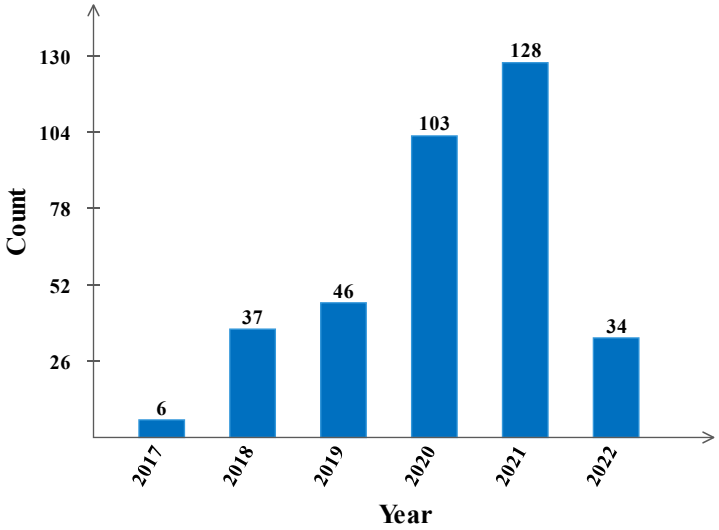

Figure 1. The trend of GRL (January 2017-March 2022).

The collaborative research on graph mining algorithms and RL models is gradually increasing, we show the trends of published papers on GRL ranging from January 2017 to March 2022 in Figure 1. We will continue to update more research articles in our published open-source repository.

The traditional methods and deep learning-based models for graph mining tasks have major differences in terms of model design and training approaches from RL-based methods, and scholars are facing many challenges in employing RL methods to analyze graph data. While machine learning models commonly used in computer vision and NLP effectively capture hidden patterns of Euclidean data (e.g., images and speech), the complexity of the graph data allows these models not to be adopted directly to the graph data. Secondly, the complexity of graph data makes RL methods difficult to accurately establish the environment or state space, and design action space or reward functions for specific tasks [21]. Specifically, graphs are massive, high-dimensional, discrete objects, and are challenging to work with in a modeling context. Thirdly, the goals of graph mining capture tasks vary among themselves, which requires RL methods to design effective architectures for specific objectives as well as to adapt to the graph structure. Fourthly, the emergence of large-scale graph data creates limitations for the RL methods to be employed. The historical data-driven RL methods need to continuously learn from the environment to update the parameters, and the increasing of the data scale causes inefficiencies in terms of RL methods. Finally, many interdisciplinary data in the real world can be modeled as network structures, and interdisciplinary data analysis integrating domain knowledge will drive models toward expansion and complexity, and this phenomenon also leads to significant difficulties in designing RL methods. We believe that the scalable and efficient RL methods to solve graph mining tasks are the key research problem.

To address the above challenges, scholars have been working extensively in the domain of RL and graph mining, and there are attempts in many fields [22, 23], ranging from solving tasks such as rumor detection [24] to interactive recommender systems [25] with GRL techniques. After the comprehensive survey, we found that existing methods that combine RL techniques with graph data

mining for co-learning are divided into two main categories: (1) Solving RL problems by exploiting graph structures [26-32], (2) Solving graph mining tasks with RL methods [15, 22, 33-35].

We define GRL as a series of solutions and measures that are proposed to address graph mining tasks based on key components such as nodes, links, and subgraphs in a graph, employing RL methods to explore the topology and attribute information of the network.

From this survey, we believe that GRL methods are in the early stage. The success of GRL in many domains is partially attributed to: (1) **Generalization and Adaptability.** RL methods enable learning specific objectives in networks with different topologies without considering the effect of network structure on prediction performance since RL methods allow for adaptive learning by representing graph mining tasks as the MDP with sequential decision characteristics. (2) **Data-driven and Efficient.** Existing graph data mining methods introduce abundant expert knowledge or require some rules to be developed manually [7, 18, 36], while RL methods allow fast learning without expert knowledge, and these methods achieve learning objectives more efficiently.

There are a limited number of comprehensive reviews available to summarize these works using RL methods to solve problems in graph data mining, thus, it is necessary to provide this systematic review of methods and frameworks for GRL. We provide an overview of the latest research status and trends in the domains discussed above and summarize the extensive literature for some specific real world problems to prove the importance of GRL. We aim to provide an intuitive understanding and high-level insight into the different methods so that scholars can choose the suitable direction to explore. To the best of our knowledge, this is the latest work to provide a comprehensive survey of various methods in GRL.

Our paper makes notable contributions summarized as follows:

- **Comprehensive review.** We provide a systematic and comprehensive overview of modern GRL techniques for graph data, and we categorize the GRL literature according to research domains and characteristics of RL methods that focus on solving data mining problems on graphs with RL.
- **Recent literature.** We provide a summary of recent work about GRL and an investigation of real world applications. By doing the survey, we hope to provide a useful resource and global perspective for the graph mining and RL community.
- **Future directions.** We discuss new directions in GRL and suggest six possible future research directions in terms of automated RL, Hierarchical RL, multi-agent RL, subgraph pattern mining, explainability, and evaluation metrics.
- **Abundant resources.** We collect abundant resources on GRL, including state-of-the-art models, benchmark data sets, and open-source codes. We create an open-source repository for GRL, which contains numerous papers on GRL in recent years, categorized by year of publication. In this open-source repository, we provide links to the papers and code (optional) to allow scholars to get a faster overview of the research progress and methods, and to provide more inspiration for research on GRL.

The rest of this survey is organized as follows. Section 2 provides a brief introduction to the graph data mining concepts discussed in this paper and a brief review of the key algorithms commonly used in GRL research. In Section 3, we review some current research on the topic of GRL and list the state, action, reward, termination, RL algorithms, and evaluation metrics from this research. In Section 4,

we discuss and propose some future research directions and challenges. Finally, we summarize the paper in Section 5.

2 Preliminaries

2.1 Graph Mining

Graphs are the natural abstraction of complex correlations found in numerous domains, different types of graphs have been the main topic in many scientific disciplines such as computer science, mathematics, engineering, sociology, and economics [37]. We summarize the common graph data mining problems in GRL research as follows, and the scholars try to solve these problems with RL-based methods. The abbreviations used in the paper are illustrated in Table B, and the notations used are illustrated in Table B.

2.1.1 Graph Neural Networks

Recently, Graph Neural Networks (GNNs) have been commonly employed to model and compute graph data to improve the ability to reason and predict relationships in graphs. These models can efficiently learn the relationships between nodes an links in graphs and establish rules for graph data, and have proved their powerful ability to model graph-structured data in different domains, ranging from social media to biological network analysis and network modeling and optimization [38, 39]. GNN learns the embedding of nodes and graphs by aggregating the features of target nodes and their neighboring nodes. The basic principle of the GNN model can be concluded as follows: Given a graph G with m nodes, the G can be represented by the adjacency matrix $A \in \{0,1\}^{m \times m}$ and the feature matrix $X \in R^{m \times d}$, where d indicates that each node corresponds to a d-dimensional feature vector, the information aggregation operation of GNN is defined as:

$$X_{i+1} = \sigma\left(D^{-\frac{1}{2}}\hat{A}D^{-\frac{1}{2}}X_i W_i\right) \tag{1}$$

where $\mathrm{X_i}$ denotes the input feature matrix of the i-th layer GCN. $\mathrm{X_0 = X}$ denotes the original feature vector of the input graph, and $\mathrm{X_i \in R^{m \times d_i}}$ is transformed into $\mathrm{X_{i+1} \in R^{m \times d_{i+1}}}$ by the message passing mechanism. $\mathrm{\widehat{A} = A + I}$ denotes that self-loops are added to the adjacency matrix of the input graph. D denotes the diagonal node degree matrix after the normalization operation is executed on $\mathrm{\widehat{A}}$. In addition, $\mathrm{W_i \in R^{d_i \times d_{i+1}}}$ is the learnable weight matrix and $\sigma(\cdot)$ denotes the nonlinear activation function. Motivated by Weisfeiler-Lehman (WL) graph isomorphism test, learning a GNN includes three main components [40, 41] : (1) **Initialization**: initialize the feature vectors of each node by the attribute of the vertices, (2) **Neighborhood detection**: determiner the local neighborhood for each node to further gather the information and (3) **Information aggregation**: update the node feature vectors by aggregating and compressing feature vectors of the detected neighbors.

Many different message passing strategies have been proposed to enable graph data mining by scholars, ranging from the novel information aggregation functions [16] to graph structure pooling [17]

and neighborhood sampling approaches [12]. Different GCNs models provide high-level variability in the design of neighborhood detection and information aggregation, and these various models define the different node neighborhood information and the aggregation compression methods to achieve graph learning. These methods commonly consider that the features of the target nodes are obtained by aggregating and combining the features of their neighbors.

2.1.2 Network Representation Learning

Models and algorithms for network representation learning are pervasive in our society, and impact human behavior via social networks, search engines, and recommender systems, to name a few. Network representation learning has been recently proposed as a new learning paradigm to embed network vertices into a low-dimensional vector space, by preserving network topology structure, vertex content, and other side information. The low-dimensional embeddings of nodes obtained via network representation learning methods enable the application of vector classification models or vector-based machine learning models for downstream tasks, e.g., link prediction, node classification, graph clustering, etc. Graph data mining methods based on network representation learning avoid the problem of missing information caused by the direct application of the original network structure. In addition, the deep network representation learning-based methods follow the message passing strategy and the graph embeddings obtained to preserve the proximity information in the low-dimensional vector space.

The network representation learning problem is defined as follows: Given a network $G = (V, E, X, Y)$, where E is the set of edges in the network, V is the set of nodes in the network, X is the attribute feature of the nodes, and Y denotes the label of the nodes, the network structure with node labels is commonly employed for node classification tasks. The network representation learning method allows the graph to be embedded based on the mapping function $\mathcal{F}: v \rightarrow e_v \in R^d$, where e is the embedding of the learned node v, the mapping function f is employed to map the network structure into the low-dimensional vector space and preserve the topological and attribute information of the graph, to ensure that nodes with similar structure and attributes have higher proximity in the vector space. Zhang et al. [42] propose that the learned node representations should satisfy the three conditions: (1) **Low-dimensional:** the dimension of learned node representations should be much smaller than the dimension of the original adjacency matrix representation, (2) **Informative**: the learned vertex representations should preserve vertex proximity reflected by network structure, vertex attributes and vertex labels, and (3) **Continuous:** the learned node representations should have continuous real values to support subsequent network analytic tasks and have smooth decision boundaries to ensure the robustness of these tasks.

In GRL, scholars have introduced RL into the design principles of network representation learning, and have formulated the network representation learning problem as the Markov Decision Process (MDP), they have also employed RL methods in multi-relational networks and heterogeneous information networks for constructing input data for network representation learning models or optimizing the selection scheme of neighbors.

2.1.3 Adversarial Attacks

Deep neural networks are very sensitive to adversarial attacks, which can significantly change the prediction results by slightly perturbing the input data [43]. On the basis of the targets and objectives chosen by the attackers, Sun et al. [44] categorize the network adversarial attack problem as (1) **Model-Objective attack.** The strategies of these attacks are attacking the specified model with attack methods to make the models become non-functional working in multiple scenarios, including evasion attacks and poisoning attacks. In addition, RL-based adversarial attack methods [19, 43, 45] have attracted much interest. (2) **Data-Objective attack.** Data-Objective attacks do not attack a specific model. Such attacks happen when the attacker only has access to the data but does not have enough information about the model, including statistical information and model poisoning [46, 47].

2.1.4 Knowledge Graphs

Knowledge graphs, which preserve rich human knowledge and facts, are commonly employed in downstream AI applications, and large-scale knowledge graphs such as DBpedia[48], Freebase[49], and Yago [50] are proved to be the basis of research for tasks such as recommender systems, dialogue generation, and Question and Answer (Q&A) tasks. The knowledge graph can be defined as $G = (h, r, t)$ to preserve the multi-relationship graph with a large a number of facts, where h denotes the head entity, t denotes the tail entity, and r denotes the relationship between h and t. The real-world knowledge graphs are usually incomplete, one needs to complete it by inferring the missing ones with the help of existing information, and scholars have proposed numerous methods to implement knowledge graph reasoning and completion. Knowledge graph embedding and multi-hop path reasoning are the main methods to tackle knowledge graph complementation. Knowledge graph completion methods are mainly divided into three categories [51]: (1) **Path ranking-based methods**[52]. such methods are to a path to connect two entities as a feature to predict the relationship between two entities. (2) **Representation learning-based methods** [53, 54]. Such methods aim to represent the semantic information of the research object as a dense low-dimensional real-value vector and reason via vector operations. (3) **RL-based methods** [3]. These methods define the knowledge graph reasoning task as MDP.

2.2 Reinforcement Learning Methods

RL has recently achieved huge success in a variety of applications [55]. RL automatically tackles sequential decision problems in real-world environments via goal-directed learning and decision making. Such methods achieve great success in many real match games [56, 57].

The RL problem is commonly formulated as an MDP, which is a sequential decision mathematical model in which actions affect not only the current short-term rewards but also the subsequent states and future rewards, and serves to simulate the random strategies and rewards of agents. In MDP, the prediction and control problem can be implemented by dynamic programming. The formal definition of MDP is the tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, p(s_0), \gamma\}$, where $\mathcal{S}$ denotes the set of all possible states, which is the

generalization of the environment, $\mathcal{A}$ denotes the set of actions that can be adopted in the states, which is the all possible actions of the agent, $\mathcal{R}: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathrm{R}$ denotes the reward function, which is the rewards that the environment returns to the agent after executing an action, $\mathcal{T}: \mathcal{S} \times \mathcal{A} \to \mathrm{p}(\mathcal{S})$ is identified by the state transition function, $\gamma \in [0,1]$ denotes the discount factor, which can be treated as a hyperparameter of the agent and serves to contribute to the faster availability of short-term rewards to the agent. The process of interaction between agent and environment is shown in Figure 2.

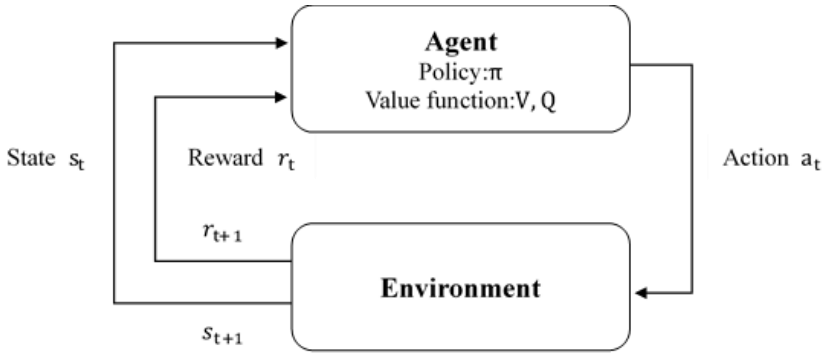

Figure 2. The interaction process between agent and environment. The interaction process between agent and environment is described as: The first state is sampled from the initial state distribution $\mathrm{p}(\mathrm{s}_0)$. At timestep t, the agent makes a calculated decision based on the state $\mathrm{s}_\mathrm{t} \in \mathcal{S}$ returned from the environment and chooses the corresponding action $\mathrm{a}_\mathrm{t} \in \mathcal{A}$, which will be applied to the environment, resulting in the changes in the environment. The agent obtains the state representation $\mathrm{s}_{\mathrm{t+1}} \backsim \mathcal{T}(\cdot\,|\mathrm{s}_\mathrm{t}, \mathrm{a}_\mathrm{t})$ and a reward $\mathrm{r}_\mathrm{t} = \mathcal{R}(\mathrm{s}_\mathrm{t}, \mathrm{a}_\mathrm{t}, \mathrm{s}_{\mathrm{t+1}})$ according to the changes in the environment. The agent acts in the environment according to a policy $\pi: \mathcal{S} \to \mathrm{p}(\mathcal{A})$. By repeatedly selecting actions and transitioning to a next state, we can sample a trace $\{\mathrm{s}_0, \mathrm{a}_0, \mathrm{r}_0, \mathrm{s}_1, \mathrm{a}_1, \mathrm{r}_1, \ldots\}$ through the environment.

Our goal is to find a policy π that maximizes our expected return $\mathrm{Q}(\mathrm{s}, \mathrm{a})$, the target policy is defined as Equation 2 [58]:

$$\pi^* = \arg\max_{\pi} Q(s,a) = \arg\max_{\pi} E_{\pi,\mathbb{T}}\left[\sum_{\mathrm{k}=0}^{\mathrm{K}} \gamma^{\mathrm{k}}\, \mathrm{r}_{\mathrm{t+k}} \middle| \mathrm{s}_\mathrm{t} = \mathrm{s}, \mathrm{a}_\mathrm{t} = \mathrm{a}\right] \tag{2}$$

RL algorithms are categorized into Model-based and Model-free algorithms according to the accessibility of the agent to the environment [59]. Model-based algorithms have to first construct the state representation that the model should predict, and there are potential problems between representation learning, model learning, and planning. In addition, model-based algorithms often suffer from the challenge that the agents cannot effectively model the real environment.

Deep Reinforcement Learning (DRL) is poised to revolutionize the field of artificial intelligence and represents a step toward building autonomous systems with a higher-level understanding of the visual world [60]. Deep learning enables RL to scale to decision-making problems that were previously intractable, i.e., settings with high-dimensional state and action spaces.

In GRL, it is common for scholars to employ model-free RL algorithms for graph data mining. This survey is aimed at solving graph mining tasks with RL methods. Therefore, we do not summarize all the RL methods, and instead, we focus on the current research results in the field of GRL. In this section, we present these RL methods for graph data mining.

2.2.1 Q-learning

Q-learning [61]is an off-policy learner and Temporal Difference (TD) learning method, which is an important result of early research on RL. The target policy in Q-learning updates the values directly on the Q-table to achieve the selection of the optimal policy, whereas, the behavior policy employs the ε-greedy policy for semi-random exploration of the environment. The learning goal of the action value function Q to be learned in Q-learning is that the optimal action value function q^* is learned by direct approximation, so that the Q-learning algorithm can be formulated as:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[R_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right] \quad (3)$$

where the equation denotes that state s_t explores the environment with a behavior policy based on the values in the Q-table at timestep t,, i.e., it performs action a, the reward R and a new state s_{t+1} is obtained based on the feedback from the environment. The equation is executed to update to obtain the latest the Q-table, it will continue to update the new state s_{t+1} after completing the above operation until the termination time t.

2.2.2 REINFORCE

The REINFORCE method [62] does not optimize directly on the policy space but learns the parameterized policy without the intermediate value estimation function, which uses the Monte Carlo method for learning the policy parameters with the estimated returns and the full trace. The method creates a neural network-based policy that takes states as inputs and generates probability distributions in the operation space as outputs. The goal of the policy is to maximize the expected reward. This discounted reward is defined as the sum of all rewards obtained by the agent in the future. The policy π is parameterized with a set of weights θ so that $\pi(s;\theta) \equiv \pi(s)$, which is the action probability distribution on the state, and REINFORCE is updated with the following method:

$$\Delta w_{i,j} = a_{i,j}(r - b_{i,j})\frac{\partial}{\partial w_{i,j}}\ln(g_i) \quad (4)$$

where $a_{i,j}$ is a non-negative learning factor, r denotes the discounted reward value, and $b_{i,j}$ is a representation function of the state for reducing the variance of the gradient estimate. Williams [62] points that $b_{i,j}$ could have a profound effect on the convergence speed of the algorithm. g_i is the probability density function for randomly generating unit activation-based actions.

2.2.3 Actor–critic

The Actor-Critic algorithm [63] combines the basic ideas of magnetic gradient and approximate dynamic programming, which employs a parameterized policy and a value function, and the value function to provide a better $\widehat{A}(s,a)$ estimate for the computation of the policy gradient. The Actor-Critic algorithm studies both policy and state value functions, combining the advantages of value function-based and policy gradient-based algorithms. In this method, the Actor denotes the policy function for learning the policy that can obtain as many rewards as possible, and the Critic denotes the

estimated value function for evaluating the estimated value of the current policy.

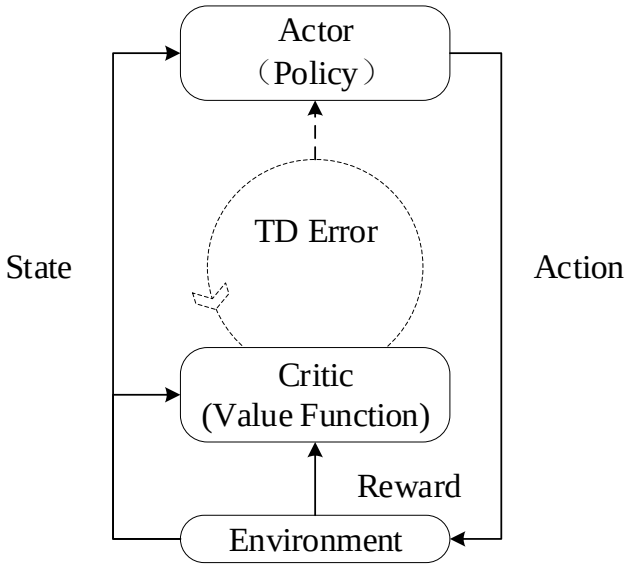

Figure 3. The framework of Actor-Critic algorithm. Actor receives a state from the environment and selects an action to perform. Meanwhile, Critic receives the current state and the state generated by the previous interaction, and calculates the TD error to update Critic and Actor.

The actor-critic architecture is widely employed as the basic framework for RL algorithms Asynchronous Advantage Actor-Critic (A3C) [64], Advantage Actor-Critic(A2C)[65], DPG [66], DDPG [67]. The A3C develops for single and distributed machine setups combines advantage updates with an actor-critic formulation that relies on asynchronous update policy and value function networks trained in parallel on several processing threads [60].

2.2.4 Deep Q-network

Q-table updates with the Q-learning algorithm in large-scale graph data suffer from the problem of high number of intermediate state values, leading to dimensional disasters. To address the above challenges, the main method proposed by scholars is the Deep Q-Network (DQN) [2] which combines value function approximation and neural networks through function approximation and state/action space reduction. DQN is widely exploited to learn policies by leveraging deep neural networks.

The scholars propose to employ Q-learning algorithms to represent states with graph embeddings according to the property of graph data in order to minimize the impact of non-Euclidean structures on the Q-table scale. Overall, DQN combines deep learning with Q-learning and approximates the action value function with deep neural networks, and obtains the trace $\{s_t, a_t, r_t, s_{t+1}\}$ from the interaction of the agent with the environment. This method can learn successful policies directly from high-dimensional sensory inputs using end-to-end RL [68]. The loss function $L(\theta_t)$ of this method is given by:

$$L(\theta_t) = E_{(s_t,a_t,r_t,s_{t+1})\sim D}\left[\left(r + \gamma \max_{a+1} Q_{\theta_{t-1}}(s_{t+1}, a_{t+1}) - Q_{\theta_t}(s_t, a_t)\right)^2\right] \quad (5)$$

DQN introduces two techniques to stabilize the training process: (1) a replay buffer to reuse past experiences. (2) a separate target network that is periodically updated. Since the success of DQN, a large number of improved algorithms have been proposed. Double Deep Q-Network (DDQN) [69]

reduces the risk of high estimation bias in Q-learning by decoupling selection and evaluation. the DDQN addresses the problem of not sufficiently considering the importance of different samples in the DQN algorithm by calculating the priority of each sample in the experience pool and increasing the probability of valuable training samples. Moreover, scholars have proposed more improved versions of DQN [70, 71] to address problems such as the lack of long-term memory capability of DQN.

3 Reinforcement Learning on Graph

Existing methods to solve graph data mining problems with RL methods focus on network representation learning, adversarial attacks, relational reasoning. In addition, many real-world applications study the GRL problem from different perspectives. We will provide a detailed overview of the research progress in GRL in this section.

3.1 Dataset & Open-source

3.1.1 Datasets

We provide a summary of the experimental datasets employed in these studies cited in this survey and a list of the statistical information of datasets, sources, tasks, and the methods employing these datasets, as shown in Tables 1 and 2. In addition, we summarize the experimental results of node classification on Cora, Citeseer, and Pubmed and the results of knowledge graph reasoning on NELL-995, WN18RR, and FB15K-237, where the classification accuracy is the evaluation metric for the node classification task and Hit@1, Hit@10, and MRR are the evaluation metrics for the knowledge graph reasoning, as shown in Table 3 and Table 4.

3.1.2 Open-source for reinforcement learning methods

To facilitate research into GRL, we develop an open-source repository. The papers in this repository are categorized by year and these papers cover the fields of solving GRL problem, such as node classification, graph classification, and model explainability, which are available in public[1]. We believe that this repository could provide a comprehensive and efficient summary of methods in the field of GRL. In addition, we summarize the open-source implementations of GRL methods reviewed in the survey. We provide the links of the codes of the models in Table C.

[1] https://github.com/neunms/Reinforcement-Learning-on-Graph-A-Survey

Table 1. Commonly used datasets for GRL.

Dataset	|V|	|E|	Source	Citation	Task
Yelp	45954	3846979	[72]	RioGNN [73], CARE-GNN [36], UNICORN [74]	Fraud Detection
Amazon	11944	4398392	[75]	RioGNN [73], CARE-GNN [36], SparRL [37]	Fraud Detection, Community detection
MIMIC-III	28522	337636545	[76]	RioGNN [73]	Diabetes Detection
Cora*	2708	5429	[77]	Policy-GNN[41], RL-S2V [19], GraphNAS [78], GPA [14], AGNN [79], GDPNet [80], NIPA [81]	Node Classification
Citeseer	3327	4732	[77]	Policy-GNN [41], RL-S2V [19], GraphNAS [78], SparRL [37], GPA [14], AGNN [79], GDPNet [80]	Node Classification
Pubmed	19717	44338	[77]	Policy-GNN[41],NIPA [81], RL-S2V[19], GraphNAS[78], GPA[14], AGNN[79], GDPNet[80]	Node Classification
Twitter*	81306	1768149	[82]	SparRL[37], AdRumor-RL[24]	Fraud Detection
Facebook	4,039	88,234	[82]	SparRL[37]	Fraud Detection
YouTube	4,890	20,787	[83]	SparRL[37]	Community detection
Email-Eu-Core	1,005	16,064	[84]	SparRL[37]	Community detection
NELL-995*	75492	237	[3]	RF[33],AttnPath[85], RLH[86], PAAR[87], Zheng et al.[88], ADRL[89], GRL[51], MARLPaR[90], ConvE[91], MINERVA[92], RLPath[93]	KG reasoning, Link Prediction, Fact Prediction
WN18RR	40493	11	[94]	RF[33], RLH[86], Zheng et al.[88], ADRL[89], GRL[51], MARLPaR[90], ConvE[91], MINERVA[92], KBGAN[95]	KG reasoning
FB15K-237*	14505	200	[96]	RF[33],AttnPath[85], RLH[86], PAAR[87], Zheng et al.[88], ADRL[89], GRL[51], ConvE[91], MINERVA[92], KBGAN[95], RLPath[93]	KG reasoning, Link Prediction, Fact Prediction

*These datasets are used at different scales, yet they share the same data sources. We marked the maximum number of nodes and edges.

Table 2. Commonly used datasets for GRL in Graph Classification.

Dataset	# Graphs	# Nodes (Avg.)	# Edges (Avg.)	Source	Citation	Task
MUTAG	188	17.93	19.79	[97]	SUGAR[15], SubgraphX[35], XGNN[18], GraphAug[98]	Graph Classification
PTC	344	14.29	14.69	[99]	SUGAR[15]	Graph Classification
PROTEINS	1113	39.06	72.82	[100]	SUGAR[15], GraphAug[98]	Graph Classification
D&D	1178	284.32	715.66	[101]	SUGAR[15]	Graph Classification
NCI1	4110	29.87	32.30	[102]	SUGAR[15], GraphAug[98]	Graph Classification
NCI109	4127	29.68	32.13	[102]	SUGAR[15], GraphAug[98]	Graph Classification
BBBP	2039	24.06	25.95	[103]	SubgraphX[35]	Graph Classification
GRAPH-SST2	70042	10.19	9.20	[104]	SubgraphX[35]	Graph Classification

Table 3. Reported experimental results for node classification on three frequently used datasets. Missing values ("-") in this table indicate that this method has no experimental results reported on the specific dataset.

Method	Cora	Citeseer	Pubmed
GCN[105]	81.50±0.0	70.30±0.0	79.00±0.0
GAT[106]	83.0 ±0.7	72.5 ±0.7	79.0 ±0.3
LGCN[107]	83.3 ± 0.5	73.0 ± 0.6	79.5 ± 0.2
DGI[108]	82.3 ±0.6	71.8 ±0.7	76.8 ±0.6
Policy-GNN[41]	91.9±1.4	89.7±2.1	92.1±2.2
GraphNAS[78]	-	73.1±0.9	79.6±0.4
AGNN[79]	83.6 ±0.3	73.8 ±0.7	79.7 ±0.4

Table 4. Reported experimental results for KG reasoning on three frequently used datasets. Missing values ("-") in this table indicate that this method has no experimental results reported on the specific dataset.

Method	Metrics	NELL-995	WN18RR	FB15K-237
TransE[53]	Hit@1	0.608	0.491	0.392
	Hit@10	0.793	0.626	0.614
	MRR	0.715	0.557	0.494
TransR[109]	Hit@1	0.631	0.526	0.405
	Hit@10	0.806	0.641	0.634
	MRR	0.727	0.581	0.532
ComplEX[110]	Hit@1	0.614	0.319	0.318
	Hit@10	0.815	0.462	0.542
	MRR	0.652	0.428	0.374
RF[33]	Hits@1	0.675	0.399	0.298
	Hit@10	-	-	-
	MRR	0.774	0.464	0.386
RLH[86]	Hit@1	0.692	0.453	0.342
	Hit@10	0.873	0.516	0.648
	MRR	0.723	0.481	0.460
Zheng et al.[88]	Hit@1	0.655	0.438	0.332
	Hit@10	0.847	0.539	0.591
	MRR	0.731	0.473	0.422
ADRL[89]	Hit@1	0.808	0.683	0.574
	Hit@10	0.975	0.723	0.796
	MRR	0.916	0.704	0.712
GRL[51]	Hit@10	0.886	0.935	0.912
	MRR	0.794	0.937	0.954
MARLPaR[90]	Hit@1	-	0.421	-

	Hit@10	-	0.520	-
ConvE[91]	Hit@1	0.672	0.402	0.313
	Hit@10	0.864	0.519	0.601
	MRR	0.747	0.438	0.410
MINERVA[92]	Hit@1	0.663	0.413	0.217
	Hit@10	0.831	0.513	0.456
	MRR	0.725	0.448	0.293
KBGAN[95]	Hit@1	-	-	-
	Hit@10	-	0.469	0.458
	MRR	-	0.215	0.278

3.2 Graph mining with reinforcement learning

3.2.1 Network representation learning

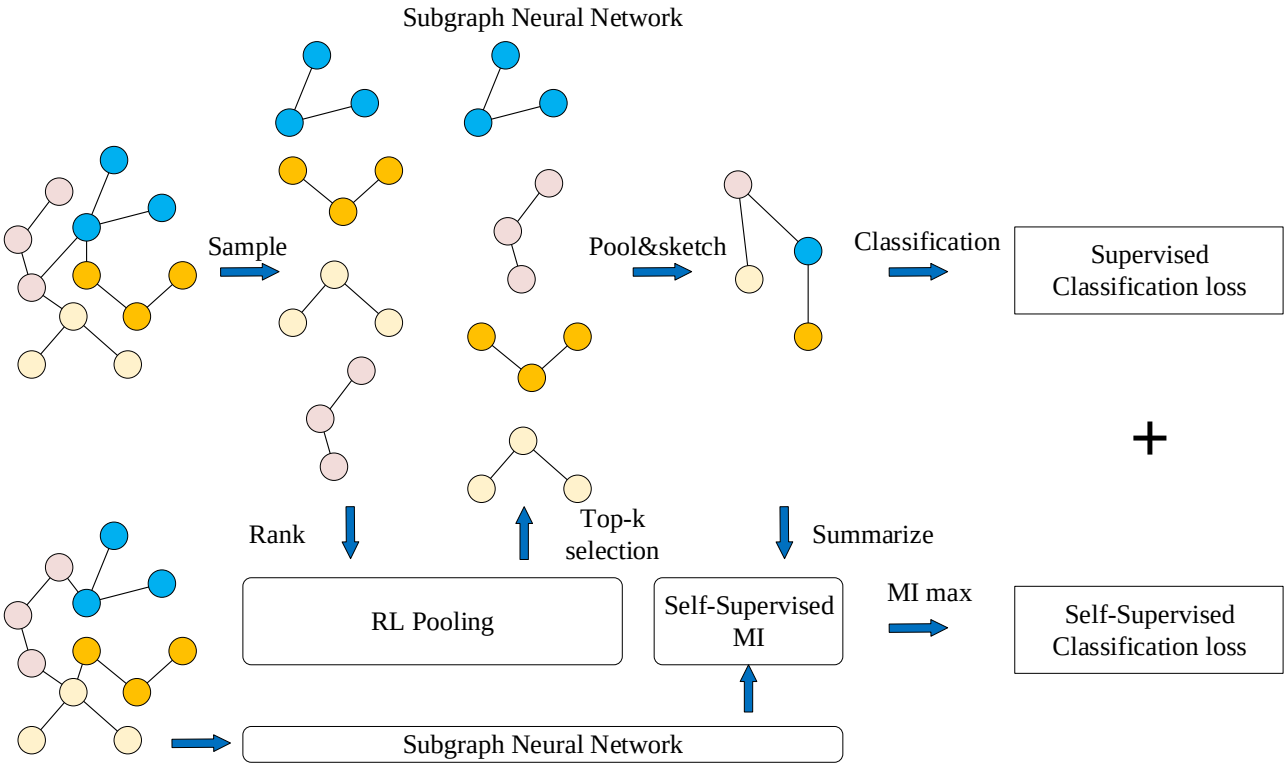

Figure 5. An illustration of the SUGAR architecture [15]. This method consists of Subgraph Neural Network, Reinforcement Pooling Module, and Self-Supervised Mutual Information Module. (Redrawn from [15])

Network representation learning is to learn a mapping that embeds the nodes of a graph as low-dimensional vectors, while encoding a variety of structural and semantic information. These methods aim to optimize the representations so that geometric relationships in the embedding space preserve the structure of the original graph [111, 112]. The vector representation obtained could effectively support extensive tasks such as node classification, node clustering, link prediction and graph classification [113, 114]. Existing network embedding methods commonly suffer from three challenges [15]: (1) **Low feature discrimination.** Fusing all features and relationships to obtain an overview network representation always brings the potential concern of over-smoothing, which leads to indistinguishable features of the graph. (2) **Demand for the prior knowledge.** Preservation of structural features in the form of similarity measures or motifs is always based on heuristics and requires substantial prior knowledge. (3) **Low explainability.** Many methods exploit substructures by step-by-step pooling, which loses much of the detailed structural information and leads to a lack of sufficient explainability for downstream tasks. To address the above challenges, SUGAR [15] retains structural information from the "node-subgraph-graph" levels by adaptively selecting significant subgraphs with Q-learning to represent the discriminative information of the graph, and this hierarchical learning approach provides powerful representations, generalization, and explainability.

GNNs receive increasing interest due to their effectiveness in network representation learning. Generally, the message passing architecture predefined by scholars is probably not applicable to all nodes in graphs, and the multi-hop message passing mechanism is poor for passing important information to other modules. Therefore, the novel and effective node sampling strategies and innovation of new message passing mechanisms for graph data start to attract more attention from scholars [16, 32, 115]. Batch sampling and importance sampling are commonly employed by scholars in studies for GNNs to obtain the local structure of graph data, however, such sampling methods may cause information loss. Lai et al. [41] defined a meta-policy module and a GNN module for learning relationships and network representations between node attributes and aggregation iterations. The defined MDP samples the number of hops of the current node and neighboring nodes iteratively with meta-policy and trains the GNN by aggregating the node information within a specified within the sampled hops. The proposed Policy-GNN algorithm solves the challenge of determining the aggregation range of nodes in large-scale networks with DQN algorithm [68]. Hong et al. [32] focused on the modular RL method and designed a transformer model without morphological information to efficiently encode relationships in graphs, thus solving the challenge of the multi-hop information transfer mechanism. In addition to the parameter sharing and Buffer mechanism for network representation learning in Policy-GNN that enable the improvement of model efficiency, Yan et al. [21] proposed a novel and efficient algorithm for automatic virtual network embedding that combines the Asynchronous Advantage Actor-Critic (A3C) [64] with the GCN, which employs the GCN to automatically extract spatial features in an irregular graph topology (i.e., the substrate network) in learning agent.

Existing representation learning methods based on GNNs and their variants depend on the aggregation of neighborhood information, which makes them sensitive to noise in the graph, scholars have improved the performance of network representation learning from the perspective of removing graph noisy data. GDPNet [80] employs two phases of signal neighborhood selection and representation learning to remove noisy neighborhoods as a MDP to optimize the neighborhood of each target node, and learns a policy with task-specific rewards received from the representation learning phase, allowing the model to perform graph denoising just with weak supervision from the task-specific reward signals. GAM [116], which also focuses on noisy graph data, enables to limit the learning attention of the model to small but informative parts of the graph with attention-guided walks, The graph attention model employs the Partially Observable Markov Decision Process (POMDP) method for training to selectively process informative portions of the graph.

The ability of GNN models for representation learning will change significantly with slight modifications of the architecture. The design principles of the architecture require substantial domain knowledge to guide, e.g., the manual setting of hidden dimensions, aggregation functions, and parameters of the target classifier. The GraphNAS algorithm proposed by Gao et al. [78] designs a search space covering sampling functions, aggregation functions and gated functions and searches the graph neural architectures with RL. The algorithm first uses a recurrent net-work to generate variable-length strings that de-scribe the architectures of GNNs. The AGNN approach [79], which has the same objective as GraphNAS, verifies the architecture in the predefined search space via small steps with RL-based controllers, decomposing the search space into the following six classes of actions: Hidden dimension, Attention function, Attention head, Aggregate function, Combine function, and Activation

function.

However, the above algorithms fail to consider heterogeneous neighborhoods in the aggregation of nodes [73], causing neglect or simplification of the diverse relationship of nodes and edges in real networks, which brings a loss of heterogeneity information. Heterogeneous information networks contain multiple types of nodes and relationship, and maintain abundant and complex interdependencies between nodes, and are widely observed in real networks and practical applications [117, 118]. Relational GNNs based on artificial meta-paths [119] or meta-graphs [120] rely on inherent entity relationships and require the support of substantial domain knowledge. Zhong et al. [121] solve the dependence on the handcrafted meta-paths via proposing a RL enhanced heterogeneous GNN model to design different meta-paths for nodes in heterogeneous information networks, and replacing the manual design of meta-paths with agent. This method discovers many meaningful meta-paths that are ignored by human knowledge.

In addition to improving the design solution of meta-paths, the adoption of novel neighborhood aggregation techniques enables to effectively improve the performance of heterogeneous network representation learning. Peng et al. [73] propose a task-driven GNN framework based on multi-relational graphs that learns multi-relational node representations with the semi-supervised method, which leverages relational sampling, message passing, metric learning, and RL to guide neighbor selection within and between different relations. These above works further illustrate the massive role of RL methods in multi-relational networks.

The graph data augmentations provide further improvements in terms of the performance of network representation learning, and scholars believe that invariance of the learning mechanism is critical to the data augmentations. The GraphAug [98] enables to avoid compromising the critical label-related information of the graph on the basis of the label-invariance of the computed graph using an automated augmentation model, thus producing label invariance at most time. They proposed a RL-based training method to maximize the estimated label-invariant probability.

The GPA algorithm [14] which is employed to improve the learning performance of graph representations studies how to efficiently label the nodes in GNNs thereby reducing the annotation cost of training GNNs using an active learning method.

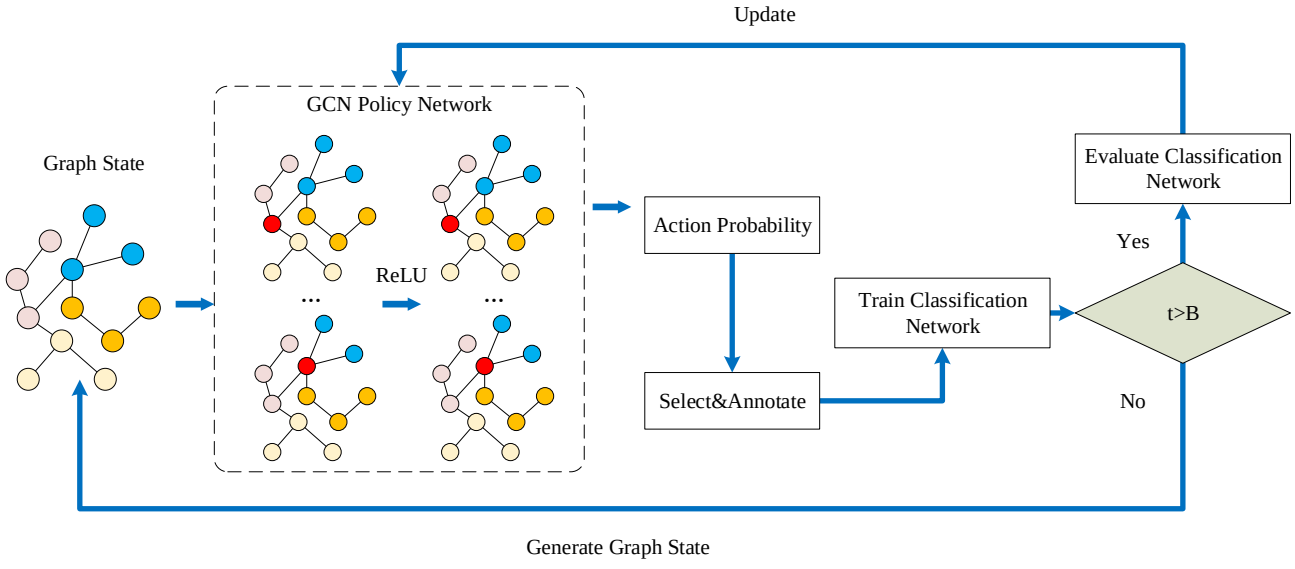

Figure 6. An illustration of the GPA architecture [14]. In GCN Policy Network, each column denotes a layer of GNN,

and the graphs in each column correspond to the feature aggregation on different nodes. (Redrawn from [14])

Table 5. GRL algorithms on network representation learning tasks

Algorithm	Action	State	Reward	Termination	RL algorithm	Metrics
SUGAR[15]	Add or minus a fixed value $\Delta k \in [0,1]$ from k.	The middle graph consists of the subgraphs selected in each training round	Define the corresponding reward value based on the classification accuracy of the previous round	The change of k among ten consecutive epochsis no more than Δk	Q-learning	Average accuracy, Standard deviation, Training process, Testing performance, and Result visualization
RL-HGNN[121]	All nodes involved in the meta-path	The set of avail-able relation types	The performance improvement on the specific task comparing with the history performances.	-	DQN	Micro-F1, Macro-F1, Time Consuming for Meta-path Design.
Policy-GNN[41]	The attribute of the current node.	The number of hops of the current node.	the performance improvement on the specific task com-paring with the last state.	-	DQN	Accuracy
RioGNN[73]	All actions when the relation r is at the depth d of the l-th layer.	The average node distance of each epoch	The similarity as the decisive factor within the reward function.	The RL will be terminated as long as the same action appears three times in a row at the current accuracy	MDP	Accuracy
GraphNAS[78]	Select the specified	Graph neural	Define the	To maximize the	Personalized	Micro-F1, Accuracy

	parameters for the graph neural network architecture	network architecture	corresponding reward value based on the special task accuracy	Expected accuracy of the generated architectures.	RL algorithm	
GPA[14]	The actions are defined on the basis of the action probabilities given by the policy network	State representation of nodes are defined on the basis of several commonly used heuristics criteria in active learning	the classification GNN's performance score on the validation set after convergence	Performance convergence of classifiers	MDP	Micro-F1 and Macro-F1
GDPNet[80]	Select neighbors	Embedding of information about the current node and its neighbors	Define the corresponding reward value based on the special task accuracy	-	MDP	Accuracy

Note: Missing values ("-") in this table indicate that the personalized termination conditions contribute to the stabilization of the training process.

3.2.2 Adversarial Attacks

Following recent research showing that GNNs are trained based on node attributes and link structures in the graph, the attacker can attack the GNN by modifying the graph data used for training, thus affecting the performance of node classification [19, 122]. The present research on adversarial attacks on GNNs focuses on modifying the connectivity between existing nodes, and the attackers inject hostile nodes with fake links into the original graph to degrade the performance of GNN in classification of existing nodes.

RL-S2V [19] learns how to modify the graph structure by sequentially adding or removing links to the graph only using the feedback from downstream tasks. Q-learning algorithm is used as the implementation solution of a RL module for sequential modification of network structure. The algorithm trains the original graph structure with structure2vec to obtain the representation of each node, and then parameterizes each node with a graph neural network to obtain the Q-value, and the candidate links that need to modify are obtained with Q-learning to achieve a black-box attack.

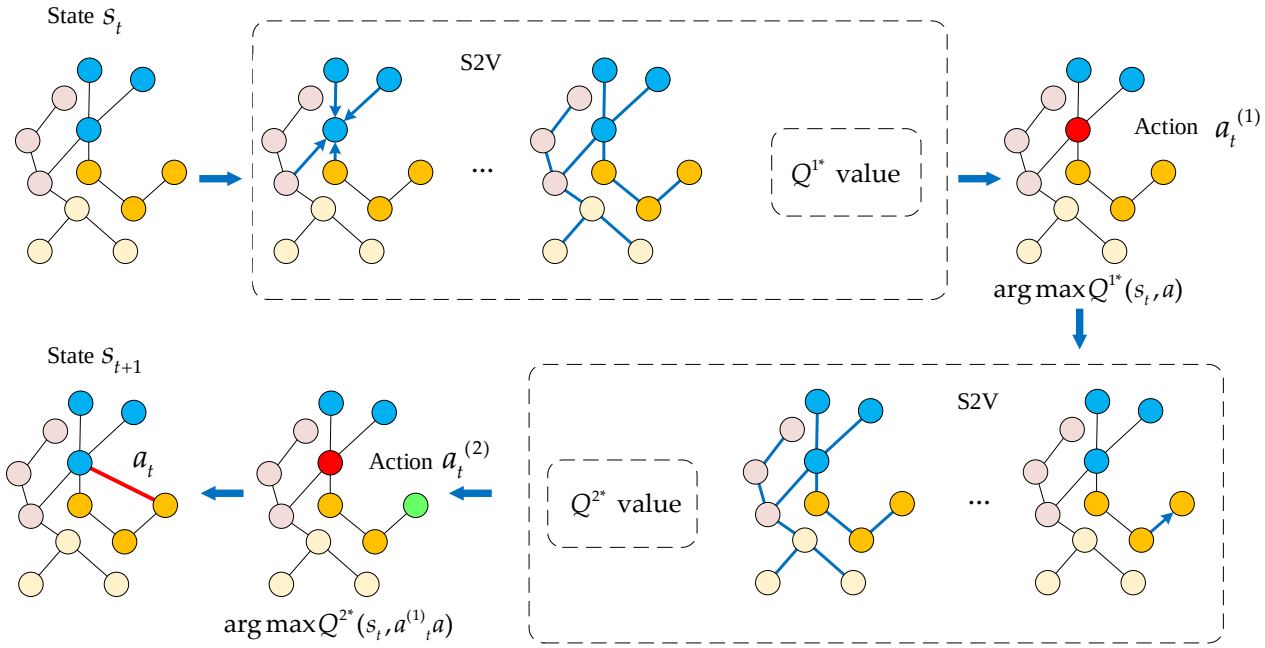

Figure 7. An illustration of the RL-S2V architecture [19]. The algorithm performs network representation learning via S2V for calculating Q-value, adding a single edge a_t is decomposed into two decision steps $a^{(1)}$ and $a^{(2)}$, withQ^{1*} and Q^{2*}. (Redrawn from [19])

However, such attack policies that require frequent modifications to the network structure need a high level of complexity to avoid the attack being caught. Sun et al. [81] present a method (Node Injection Poisoning Attack, NIPA) to poison the graphs to in-crease the node misclassification rate of GNNs without changing the link structure between the existing nodes in the graph. Instead of manipulating the links between existing nodes, the NIPA method injects fake nodes into the graph. This algorithm represents the adversarial connections and adversarial labels of the injected false nodes as MDPs and designs an appropriate reward function to guide the RL agent to reduce the node classification performance of GNNs. The graph rewiring method proposed by Ma et al. [45] also avoids

adding or removing edges resulting in significant changes to the graph structure.

Adversarial attack researches on graph neural networks also includes fraud entity detection tasks, such as opinion fraud and financial fraud. However, previous work has paid little attention to the camouflage behavior of fraudsters, which could hamper the performance of GNN-based fraud detectors during the aggregation process. These methods either fail to fit the fraud detection problems or break the end-to-end learning fashion of GNNs [36]. Dou et al. [36] propose a label-aware similarity metric and a RL-based similarity-aware neighbor selector. They formulate the RL process as a Bernoulli Multi-armed Bandit (BMAB) between the neighbor selector and the GNN with similarity metric to achieve the selection threshold for adaptively finding the best neighbor when the GNN is training, and formulate the relational-aware neighborhood aggregation method to obtain the final central node representation. Lyu et al. [24] consider that the explainability of the model is as important as its effectiveness, and their proposed AdRumor-RT model enables the generation of interpretable and effective evasive attack against a GCN-based rumor detector, where the AC algorithm is employed to implement black-box attacks.

Table 6. GRL algorithms on adversarial attacks tasks

Method	Action	State	Reward	Termination	RL algorithm	Metrics
NIPA[81]	Add the adversarial edges within the injected nodes between the injected nodes and the clean node and designing the adversarial labels of the injected nodes	The intermediate poisoned graph and labels information of the injected nodes	Guiding reward based on classifier success rate	The agent adds budget number of edges	DQN	Accuracy, key statistics of the poisoned graphs, Average Degrees of Injected Nodes, and Sparsity of the Origin Graph.
RL-S2V[19]	Add or delete edges in the graph	The modified graph	The prediction confidence of the target classifier	The agent modifies the specified number of links	Q-learning	Accuracy
CARE-GNN[36]	Plus or minus a fixed small value	The selection range of neighbor nodes	The average distance differences between two consecutive epochs.	The RL converges in the recent ten epochs and indicates an optimal threshold until the convergence of GNN.	BMAB	ROC-AUC and Recall

ReWatt[45]	The action space consists of all the valid rewiring operations	The state space of the environment consists of all the intermediate graphs generated after all the possible rewiring operations.	The reward function is developed based on the performance of the classifier	The attack process will stop either when the number of actions reaches the budget K or the attacker successfully changed the label of the slightly modified graph.	MDP	Attacking performance

3.2.3 Relational Reasoning

RL methods have achieved excellent results in terms of causal discovery from observed data, but searching the Directed Acyclic Graphs (DAGs) space or discovering the implied conditions usually has a high complexity. Discovering and understanding causal mechanisms could be defined as searching for DAGs that minimize the defined score functions. The agent in RL with random policies could automatically define the search policy based on the learn the uncertain information of the policy, which can be updated rapidly with the reward signal. Therefore, Zhu et al. [123] propose to leverage RL to find the underlying DAG from the graph space without the need of smooth score functions. This algorithm employs Actor-Critic as the search algorithm and outputs the graph that achieves the best reward among all the graphs generated during training. However, the problem of poor computational scores emerges in this method, and the action space composed of directed graphs is commonly difficult to be explored completely. Wang et al. [124] propose a causal discovery with Ordering-based Reinforcement Learning (CORL) method via incorporating RL methods into the ordering-based paradigm. The method describes the ordering search problem as a multi-step MDP and implements the ordering generation process with encoder-decoder structures, and finally optimizes the proposed model with RL based on the reward mechanism designed for each ordering. The generated ordering is then processed using variable selection to obtain the final causal graph.

Task-oriented Spoken Dialogue System (SDS) is a system that can continuously interact with a human to accomplish a predefined task. Chen et al. [125] design an alternative but complementary method to innovate the structure of neural networks incorporating the DQN algorithm in order to make them more adaptable to dialogue policy. The scholars have proposed some natural question generation models to provide more training data for the Q&A tasks in order to further improve the performance of the task. Chen et al. [126] focuses on the natural question generation and propose a RL-based Graph-to-Sequence (Graph2Seq) model, and the algorithm employs the self-critical sequence training (SCST) algorithm [127] to directly optimize the evaluation metric. Explicitly obtaining the preferences for recommended items and attributes of user through interactive conversations is the goal of Conversational Recommender Systems (CRS). Deng et al. [74] leverage a graph structure to integrate recommendation and conversation components as a whole. This algorithm exploits a dynamic weighted graph to model the changing interrelationships between users, items and attributes during the conversation, and considers a graph-based MDP environment for simultaneously processing the relationships.

With the development of Artificial Intelligence (AI), the knowledge graph has become the data infrastructure for many downstream real-world applications and has attained a variety of applications in dialogue systems and knowledge reasoning [33, 89-91]. Lin et al. [91] propose a policy-based agent to extend its reasoning paths sequentially through a RL approach on the multi-hop reasoning task. The MINERVA algorithm [92] leverages RL method to train an end-to-end model for the practical task of answering questions on multi-hop knowledge graph. However, these methods focusing on fixed multi-hop or single-hop reasoning consume substantial computational resources, and the incompleteness of hand-collected data affects the performance of the reasoning. Liu et al. [33] build an end-to-end dynamic knowledge graph reasoning framework with dynamic rewards, and this method integrates the

embedding of actions and search histories as a policy network into a feedforward neural network for dynamic path reasoning. Sun et al. [128] propose a temporal path-based RL model to solve the temporal knowledge graph reasoning task and design a temporal-based reward function to guide the learning of the model. AttnPath algorithm [85] is a path-based framework that solves the lack of memory modules and over-reliance on pre-training of algorithms in knowledge graph reasoning, and this algorithm combines LSTM and GAT to design a RL mechanism capable of forcing the agent to move. RLPath [93] considers both relation choosing and entity choosing in the relational path search.

To address the problem of incomplete knowledge graphs, scholars commonly employ multi-hop reasoning to infer the missing knowledge. Motivated by the hierarchical structure commonly employed by humans when dealing with fuzzy scenarios in cognition, Wan et al. [86] suggest a hierarchical RL method to simulate human thinking patterns, where the whole reasoning process is decomposed into two steps of RL policies. In addition, the policy gradient approach [129] and the REINFORCE [62] algorithm are employed to select two policies for encoding historical information and learning a structured action space. DeepPath [3] for hierarchical reasoning of knowledge graphs based on RL requires manual rules to process for obtaining more adequate hierarchical relationships. On the other hand, the PAAR algorithm [87] employs a multi-hop reasoning model based on hyperbolic knowledge graph embedding and RL for the hierarchical reasoning. Hierarchical information of the knowledge graph plays an important role for downstream tasks, and another RL method [88] employing hierarchical information is to reason about the knowledge graph from a methodological perspective.

Recommender systems are critical to various online applications such as E-commerce websites and social media platforms through providing the better item or information to users [130]. The Interactive Recommender System (IRS) receives substantial attention as its flexible recommendation policy and optimal long-term user experience, and scholars have introduced DRL models such as DQN [131] and DDPG [67] into the IRS for decision-making and long-term planning in dynamic environments. The KGQR algorithm [25] enables to incorporate graph learning and sequential decision problems in interactive recommender systems as a whole, to select candidate objects and learn user preferences from user feedback with prior knowledge, and improves the performance of RL through aggregating the semantic correlations between entities in the knowledge graph. It is capable of obtaining better recommendation performance with fewer user-item interactions.

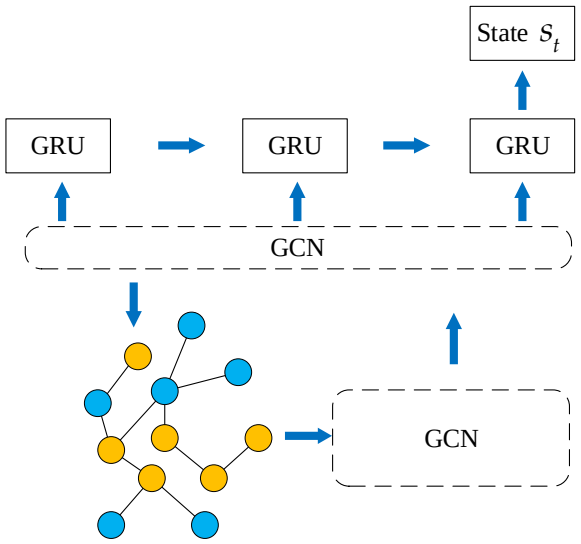

Figure 8. An illustration of the KGQR architecture [25]. The knowledge-enhanced state representation module preserves user preferences with recurrent neural network and graph neural network. (Redrawn from [25], and we skip the candidate selection module and Q-value network in KGQR.)

Table 7. GRL algorithms on relational reasoning tasks

Algorithm	Action	State	Reward	Termination	RL algorithm	Metrics
AttnPath[85]	The agent choosing a relation path to step forward.	The state space is consisted of the embedding part, the LSTM part and the graph attention part	The reward is a feedback to the agent according to whether the action is valid, and whether a series of actions can lead to ground truth tail entities in a specified number of times.	-	REINFORCE	Mean Selection Rate (MSR) and Mean Replacement Rate (MRR)
UNICORN[74]	The actions can be selected from the candidate item set to recommend items or from the candidate attribute set to ask attributes.	All the given information for conversational recommendation, including the previous conversation history and the full graph	The reward is defined as a weighted sum of five specific reward functions	-	DQN	Success rate at the turn t (SR@t), Average turn (AT), Two-level hierarchical version of normalized discounted cumulative gain (hNDCG@(T, K)
TITer[128]	the set of optional actions is sampled from the set of outgoing edges.	The state space is represented by a special quintuple.	The dynamic reward function is defined based on the search results	-	MDP	Mean Reciprocal Rank (MRR), Hits@1/3/10
RLH[86]	the set of outgoing edges of the current entity	The tuple of entities and the relationships between them	The reward for each step is defined based on whether the agent	-	REINFORCE	Mean average precision (MAP), Mean reciprocal rank

			reaches the target entity.			(MRR)
CORL[124]	The action space consisting of all the variables at each decision step,	The state space is defined as the embedding of the input data pre-processed with the encoder module.	The reward function consisting of episodic and dense rewards	-	MDP	True Positive Rate (TPR), Structural Hamming Distance (SHD)
Zheng et al.[88]	The action space consisting of all available actions that the agent can take.	The state space consisting of the current entity, start entity, and query relation	The reward is defined as the results of the scoring function of the knowledge representation model	-	REINFORCE	Mean Replacement Rate (MRR), Hits@1/10
IMUP[132]	The actions are defined as the visit event.	The state is to describe the environment composed by users and the spatial knowledge graph.	The reward is defined as the weighted sum of three parts about Points of Information (POI).	-	DQN	Precision on Category (Prec_Cat), Recall on Category (Rec_Cat), Average Similarity (Avg_Sim), Average Distance (Avg_Dist)
ADRL[89]	The set of outgoing edges	The state space consisting of the source entity, the query relationship and the entity that is accessed.	The rewards that depend on the value function.	-	A3C	Mean Replacement Rate (MRR) and Hits@1//10

GRL[51]	The agent can select its neighboring relational path to another neighboring entity at each state.	The vector representation of entities and relationships obtained with GNN	The Adversarial rewards are defined by employing Wasserstein-GAN	The agent reaches the target entity	DDPG	Mean Replacement Rate (MRR), Hits@1//10, Mean average precision (MAP), and Mean Absolute Error (MAE)

Note: Missing values ("-") in this table indicate that the personalized termination conditions contribute to the stabilization of the training process.

3.3 Real world Applications with Reinforcement Learning on Graph

The research on GRL is booming in the past few years, and GRL methods play an important role in solving various real-world applications and attract substantial attention from scholars. We summarize abundant real-world applications in transportation network optimization, recommendation systems in E-commerce networks, drug structure prediction and molecular structure generation in medical research, and network diffusion models for controlling the COVID-19 virus to further prove the hot development in the field of GRL.

3.3.1 Explainability

In exploring GNNs, scholars often treat them as black boxes, and this assumption lacks explanations that are readily understood by humans, which prevents their use in critical applications involving medicine, privacy, and security [104, 133]. A great number of scholars working on deciphering the explainability of such black-box model of GNNs commonly focus on explaining the explainability of the node, edge, or node feature-level in the graph data, but ignore the GNN model and the frequent subgraph. The interpretation at the subgraph-level enables more intuitive and effective description of the GNN [134-136]. Yuan et al. [104] categorize existing methods for explainability of GNNs into instance-level and model-level methods to advance the understanding of GNNs. In the study of GNN explainability based on subgraph level, Yuan et al. [35] propose to employ the Monte Carlo tree search method [137] to explore the critical subgraphs and thus explain the prediction problem of GNNs from subgraph-level. RioGNN [73] learns the difference in importance between different relations to obtain discriminative node embeddings, and this measure of differentiating the vector representation of nodes can enhance the quality of node embeddings while improving the explainability of the model and achieving the explainability of multi-relational GNNs from the perspective of the individual importance of different relations. Lyu et al. [24] design subgraph and node-level features to support the understanding of attack policies and detector vulnerabilities.

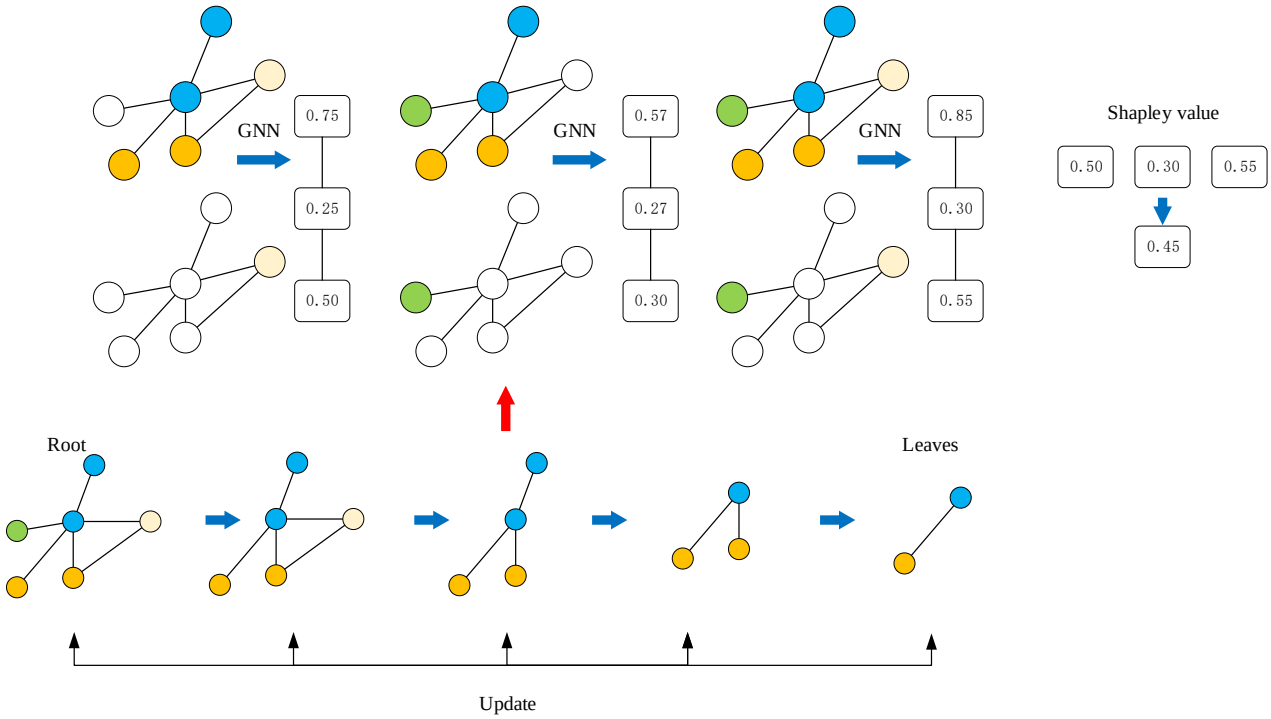

Figure 9. An illustration of the SubgraphX architecture [35]. The algorithm selects a specified path from the search tree in root-to-leaf order corresponding to one iteration of MCTS. For each node, the importance of the subgraph is evaluated by calculating the Shapley value through Monte Carlo sampling. (Redrawn from [35])

In addition, the scholars believe that explaining GNNs from the model level could enhance human trust for certain application domains. XGNN method [18] explains GNNs by formulating graph generation as RL, and the generated graph structures enable verification, understanding, and improvement of trained GNN models.

3.3.2 City Services

With the rapid development of modern cities, the rapid growth of the urban population has caused many problems, such as traffic congestion and inefficient communication [138]. The analysis of problems arising in transportation networks [139, 140] and communication networks [20, 26] with complex network methods has attracted the attention of numerous scholars. Modeling road networks and communication networks as graph data retain richer information since their structured property. The studies of transportation networks and communication networks could assist the relevant government institutions or telecommunication service providers to better optimize traffic roads or communication links, thus alleviating the traffic congestion problems and providing better communication services to citizens.

Traffic flow prediction focus on building some prediction models to estimate the traffic flow of specific roads based on the statistical information of traffic flow within or between these regions with traffic data provided by relevant government institutions or organizations. Peng et al. [22] address the problem of incompleteness and non-temporality of most traffic flow data by modeling existing traffic flow graphs with GCPN [10] and extracting temporal features with LSTM [141]. The IG-RL algorithm [142] for dynamic transportation networks focuses on the traffic signal control problem, which uses

GCN to learn the entities as node embeddings and the Q-learning algorithm is employed to train the model. This method enables the representation and utilization of traffic demand and road network structure in an appropriate way regardless of the number of entities and their location in the road network.

In addition to the control of traffic signals, Electronic Toll Collection (ETC) system serves an important role in alleviating the urban traffic congestion problem. Qiu et al. [9] employ a graph convolutional network framework to represent the value function and policy function in the Dynamic Electronic Toll Collection (DETC) problem and solve the DETC with collaborative Multi-Agent RL (MARL) algorithm. Moreover, the algorithm for optimizing on-demand ride-sharing services [143] and the lane change decision algorithm for connected autonomous vehicles [144] could also improve the operational efficiency of traffic flow in cities. Moreover, the algorithm for optimizing on-demand ride services [143] and the lane-change decisions algorithm for connected autonomous vehicle [144] could also improve the operational efficiency of traffic flow in cities.

The scholars suggest a graph convolution model of multi-intelligence collaboration [27] obtain efficient collaboration policies for improving the routing in packet switching networks. Almasan et al. [145] propose to integrate GNNs with the DQN. The proposed DRL+GNN architecture enables learning and generalize over arbitrary network topologies, and is evaluated in SDN-based optical transport network scenario. For Wireless Local Area Networks (WLANs), Nakashima et al. [146] propose a DRL-based channel allocation scheme. This algorithm employs a graph convolutional layer to extract features of channel vectors with topological information and learns DDQN [69] for channel allocation policy formulation.

3.3.3 Epidemic Control

In the fields of epidemic containment, product marketing, and fake news detection, scholars have employed a limited number of interventions to control the observed dynamic processes partially on the graph. Meirom et al. [23] propose a RL method controlling a partially-observed dynamic process on a graph by a limited number of interventions, which is successfully applied to curb the spread of an epidemic. In another epidemic control application, the RAI algorithm [147] leverage social relationships between mobile devices in the Social Internet of Things (SIoT) to assist in controlling the spread of the virus by allocating limited protection resources to influential individuals through early identification of suspected COVID-19 cases. To date, there has been little research on the use of GRL methods for network dynamics, and the existing methods have focused only on problems such as epidemic disease control. We believe that the network dynamics model based on GRL deserves further research and investigation.

3.3.4 Combination optimization

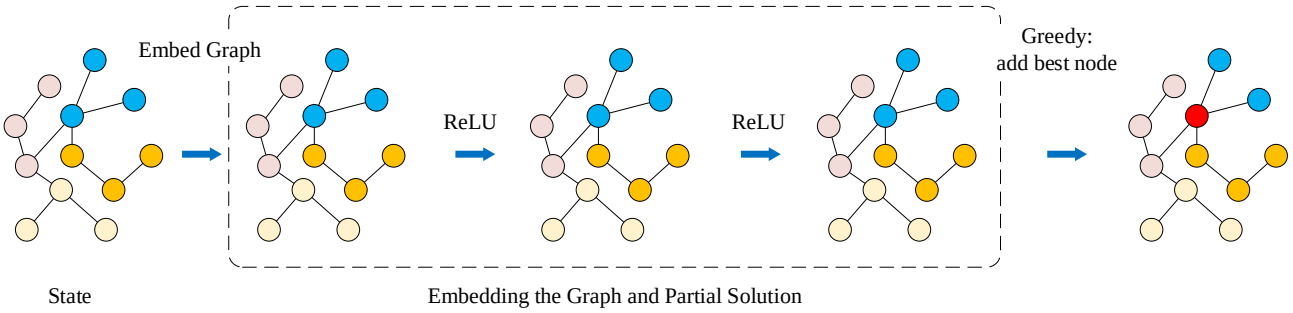

Figure 10. An illustration of the S2V-DQN architecture [148]. This framework illustrates an instance of the S2V-DQN framework for Minimum Vertex Cove (MVC). MVC is implemented through graph embedding and iterative adding nodes. (Redrawn from [148])

Combinatorial Optimization is an interdisciplinary problem, which spans optimization, operational research, discrete mathematics, and computer science, covering abundant classical algorithms such as constraint satisfaction problems, integer programming, graph algorithms etc., with many critical real-world applications [149]. Most real-world combinatorial optimization problems could be represented by graphs. Dai et al. [148] learn a policy for building solutions incrementally with Q-learning. in which the agent incrementally constructs efficient solutions based on the graph structure through adding nodes, and this greedy algorithm is a popular pattern for approximating algorithms and heuristics for combinatorial optimization. Toyer et al. [150] propose an Action Schema Network (AS-Net) for learning generalized policies for probabilistic planning problems. The AS-Net can employ a weight sharing scheme through simulating the relational structure of the planning problem, allowing the network to be applied to any problem in a given planning domain.

The dynamic version of many graphs data mining is a critical object of study for solving traffic, social, and telecommunication networks where the structure of networks in the real world evolves over time. The GTA-RL algorithm [34] is a graph-based heuristic learning algorithm for dynamic combinatorial optimization problems, which addresses the dynamics of the NP-hard problem by proposing a temporal feature encoder capable of embedding instances and a decoder capable of focusing on the embedded features to find instances of a given combinatorial problem. DeepOpt [151] employs a DRL method to solve the problem of placement policies for multiple resource types in a Virtual Network Function (VNF). The algorithm proposed by Silver et al. [152] constructs the shortest path query method in spatio-temporal graphs based on static and dynamic subgraphs, and employs a Proximal Policy Optimization (PPO) algorithm [153] to train an agent to optimize the state-to-action policy to solve the path planning problem in real-world tasks.

3.3.5 Medicine

GRL techniques are commonly used in Clinical Decision Support (CDS) applications, Medicine Combination Prediction (MCP), and chemical reaction product prediction. Wang et al. [7] propose a graph convolution RL model for MCP to solve medicine correlation and sequential problems. The method represents the MCP problem as an order-free MDP and designs a Deep Q-Learning (DQL)

mechanism to learn correlative and adverse interactions between medicines. Graph Transformation Policy Network (GTPN) proposed by Do et al. [8] addresses the problem of chemical reaction product prediction. They represented the entity system consisting of input reactants and reagent molecules as a labeled graph with multiple connected components by using a graph neural network, and the process of generating product molecules from reactant molecules is formulated as a series of graph transformations processes. In addition, GTPN uses A2C to find the optimal sequence of bond changes that transforms the reactants into products.

In the pharmaceutical industry, design and discovery aids for de novo drugs serve an important research value, and many scholars have conducted studies on chemical molecule generation and optimization with RL methods [154, 155]. These methods that use RL [10] [156] [116] to pre-design the molecular structure of drugs can significantly save working time, money, and labor costs in chemical laboratories and provide an efficient aid for the design and discovery of new drugs.

Table 8. GRL algorithms on real world applications

Algorithm	Action	State	Reward	Termination	RL algorithm	Metrics
XGNN[18]	The action is defined to add an edge to the current graph by determining the starting node and the ending node of the edge.	The partially generated graph	The reward consisting of the guidance from the trained GNNs and validated graph rules	-	MDP	Accuracy and Metrics for explanation
Marl-eGCN[9]	The toll set by the transit authority on special road that has an ETC gantry.	The number of vehicles that travel to the special zone on road	The number of vehicles which arrive at destinations	-	MDP	Traffic throughput
S2V-DQN[148]	A node of original graph that is not part of the current state	A sequence of actions (nodes) on a graph	The change in the cost function after taking an action and transitioning to a new state	Corresponds to tagging the node that is selected as the last action with feature equal 1.	Q-learning	Approximation ratio, Generalization ability, and Time-approximation trade-off
GTPN[8]	The action consisting of three consecutive sub-actions	The intermediate graph	The Reward is determined based on whether the model is successful in predicting	-	A2C	Coverage@k, Recall@k, and precision@K
IG-RL[142]	The action is defined as whether to switch to the next phase or prolong the current phase.	Current connectivity and demand in the network	The reward for a given agent is defined as the negative sum of local queues lengths	An episode either ends as soon as all trips are completed or after a fixed amount of time.	Q-learning	Trips Duration and Total Delay Evolution

GCQ[144]	The customized discrete action space ensures that does not prevent vehicle collisions during the lane change process.	The state space consisting of nodes feature, adjacency matrix, and a mask.	The reward function consisting of intention and speed rewards, collision and lane-changing penalties	The algorithm boundary is controlled by specifying the total number of connected self-driving vehicles entering the roadway.	DQN	Episode reward and Total simulation steps per episode
Peng et al. [22]	The increase and decrease of the traffic flow between any two stations.	The intermediate generated graph	The reward is defined as a sum of structural rewards and adversarial rewards.	-	PPO	Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE)
DeepOpt[151]	The action value can be denoted with a binary list indicating the selection status of each network node.	The state information including the input of the node attributes and edge attributes.	The reward function consisting of custom penalty terms	-	MDP	The Reject Ratio of Service Function Chain Requests, influence of Topology Change, Time Consumption
Vulcan[157]	An action selects a vertex on the graph which is not included in the partial solution and connected to the partial solution.	The state can be represented by the embedded vertex vector through graph embedding technology, including partial solution and other vertices on a graph.	The reward consisting of the cost function and the remaining path weight.	All terminals are added to the partial solution.	DQN	Gain

A3C+GCN[21]	An action is a valid embedding process that allocates virtual network requests onto a subset of substrate network components.	The state is the real-time representation of special network status.	Reward shaping	-	A3C	Acceptance Ratio, Long-Term Average Revenue and Running Time
GCN-RL[158]	The continuous action space is defined to determine the transistor sizes	The state consisting of transistor index, component type and the selected model feature vector for the component.	The reward is defined as a weighted sum of the normalized performance metrics	-	DDPG	FoM and the performance of Voltage Amplifier and Voltage Amplifier
GMETAEXP[28]	Each action is a test input to the program, which can be text (sequences of characters) or images (2D array of characters).	The state contains the full interaction history in the episode	Customized step-by-step rewards	Once the agent has visited all nodes or met a custom training termination condition	MDP	Coverage performance and generalization performance
Note: Missing values (“-”) in this table indicate that the personalized termination conditions contribute to the stabilization of the training process.						

4 Open Research Directions

Although several GRL methods have been explored and proposed by scholars at present, we believe that this field is still in the initial stage and has much exploration space and research value, and we suggest some future research directions that might be of interest to scholars in this field.

Automatic RL. Modeling of the environment, selection of RL algorithms and hyperparameter design of the model in GRL methods commonly require detailed expert exploration. The different definitions of state and action spaces, batch sizes and frequency of batch updates, and the number of timestep will result in completely different experimental performances. Automatic RL provides a framework for automatically making appropriate decisions about the settings of the RL process before starting to learn, enabling a non-expert way to perform, and improving the range of applications of RL. We believe that applying this automatic learning method to graph mining would significantly improve the efficiency of GRL methods and provide more efficient generation solutions for the optimal combination of graph mining algorithms and RL methods. In our survey, there are no scholars who have applied automatic RL to graph data mining tasks.

Hierarchical RL. Hierarchical RL approaches in algorithm design allow top-level policies to focus on higher-level goals, while sub-policies are responsible for fine-grained control [159, 160] [161]. The scholars have proposed the implementation of the top-level policy based on manual formulation [162] and automatic policy formulation [163] to choose between sub-policies. Hierarchical policies for GRL allow models to design top-level and sub-level structures to enhance model explainability and robustness, and either using a hierarchical data processing policy or a hierarchical algorithm design policy can make the algorithms more understandable. We believe that hierarchical GRL is a worthy field for exploration.

Multi-agent RL. Single agents usually ignore the interests of other agents in the process of interacting with the environment. Individual agents often consider other agents in the environment as part of the environment and will fail to adapt to the instable environment during the learning process as the interaction of other agents with the environment. Multi-agent RL considers the communication between multiple agents, allowing them to collaborate learning effectively. Existing multi-agent RL methods have proposed several policies such as bidirectional channels [164], sequential transfer [165] and all-to-all channels [166] to achieve interaction between agents. From existing studies, scholars are trying to learn dynamic multi-agent environments and provide an abstract representation of the interactions between agents to adapt to the dynamic evolution. The collaborative policies in multi-agent RL could help scholars to solve graph mining problems with parallel and partitioned policies.

Subgraph Patterns Mining. In the process of network local structure analysis, many scholars have explored and discovered many novel graph data mining methods that exploit the local structures of the graphs with different strategies [167] [41] [15]. In addition, they have employed batch sampling and importance sampling to obtain the local structure for network representation learning. Graph local structure selection provides natural advantages for GRL. The sequential construction of graph local structures can be represented as MDPs and guided by the performance of downstream classification and prediction tasks. These adaptive neighbor selection methods enable the performance of GNNs to

be improved and the importance of different relationships in messages passing to be analyzed in heterogeneous information networks.

Explainability for GRL. There is a limited number of scholars working on the explainability for graph neural networks, which is important for improving the performance of graph neural network models and assisting in understanding the intrinsic meaning and potential mechanisms of graph neural network models [168] [169]. We believe that the sequential decision-making of RL is suitable for the studies of explainability of graph structures.

Evaluation metrics for GRL. It is crucial to define the reward function in RL. A reasonable reward function could enable RL algorithms to converge faster and better to obtain the expected goal. However, existing GRL methods commonly employ the performance of downstream classifiers or predictors for designing reward functions. These methods allow GRL methods to be evaluated based on the performance of downstream tasks, but the execution of GRL methods includes multiple steps, and evaluating the results of each action using downstream tasks would cause substantial resource consumption. Although many scholars have proposed novel methods to reduce this consumption or to design more efficient reward functions to guide the training process of RL, the problem of resource consumption still remains, and we believe that designing reward functions that rely on actions or providing a unified GRL evaluation framework can effectively improve the performance of the algorithm. This is an urgent problem to be solved.

5 Conclusion

Network science has developed into an interdisciplinary field spanning physics, economics and biology, and computer science, with many critical real-world applications, and enables the modeling and analysis of the interaction of entities in complex systems [170]. In this survey, we conduct a comprehensive overview of the state-of-the-art methods in GRL, which suggest a variety of solution strategies for different graph data mining tasks, in which RL methods are combined to solve existing challenges better. Although GRL methods have been applied in many fields, the in-depth combination of more efficient RL with excellent graph neural network models is expected to provide better generalization and explainability and improve the ability to tackle sample complexity. We hope that this survey will help scholars to understand the key concepts of GRL and drive the field forward in the future. We believe that GRL methods could reveal valuable information in the growing volume of graph-structured data and enable scholars and engineers to analyze and exploit graph-structured data with more clarity.

Acknowledgment

This research was supported by the Key Technologies Research and Development Program of Liaoning Province in China (No. 2021JH1/10400079).

Appendix

A. Abbreviations

We summarize the full descriptions and abbreviations used in this review to facilitate searching by scholars in Table A.

Table A. A list of abbreviations.

Abbreviation	Full description
GRL	Graph Reinforcement Learning
NRL	Network Representation Learning.
NLP	Natural Language Processing
GNN	Graph neural network
RL	Reinforcement learning
DRL	Deep reinforcement learning
MDP	Markov Decision Process
MCTS	Monte Carlo Tree Search
BMAB	Bernoulli Multi-armed Bandit
QL	Q-learning
DQN	Deep Q-learning Network
DDQN	Double DQN
CDQN	Cascaded DQN
AC	Actor-Critic
A2C	Advantage Actor-Critic
A3C	Asynchronous Advantage Actor-Critic
DDPG	Deep Deterministic Policy Gradient
PPO	proximal policy optimization
POMDP	Partially Observable Markov Decision Process
DAG	Directed Acyclic Graph
GAT	Graph attention network
SCST	self-critical sequence training
ETC	Electronic Toll Collection
DETC	Dynamic Electronic Toll Collection
JSSP	job shop scheduling problem
CDS	Clinical Decision Support
TSP	traveling salesman problem
MI	mutual information
MVC	Minimum Vertex Cove

B. Notations

We provide the commonly used notations and descriptions in different domains in Table B

Table B. Commonly used notations.

Notation	Description	Domain
$\|\cdot\|$	The length of a set.	GNN
G	A graph.	GNN
$A \in \{0,1\}^{m\times m}$	The graph adjacency matrix.	GNN
$X_i \in R^{m\times d_i}$	The feature matrix of a graph in layer i.	GNN
E	The set of edges in a graph. $e \in E$.	GNN
V	The set of nodes in a graph. $v \in V$.	GNN
D	The degree matrix of A. $D_{ii} = \sum_{j=1}^{n} A_{ij}$.	GNN
n	The number of nodes, $n = \|V\|$.	GNN
m	The number of edges, $m = \|E\|$.	GNN
d	The dimension of a node feature vector.	GNN
$W_i \in R^{d_i\times d_{i+1}}$	Learnable model parameters.	GNN
$\sigma(\cdot)$	The sigmoid activation function.	GNN
$\mathcal{F}: v \to e_v \in R^d$	Mapping functions in graph embedding.	GNN
(h, r, t)	h denotes the head entity, t denotes the tail entity, r denotes the relationship between h and t.	KG
$s_t \in \mathcal{S}$	The set of all possible states	RL
$a_t \in \mathcal{A}$	The set of actions that are available in the state.	RL
$r_t \in \mathcal{R}$	The reward given to the agent by the environment after the agent performs an action.	RL
$\mathcal{T}$	The state transfer function is defined by the environment	RL
γ	Discount Factor	RL
$\pi: \mathcal{S} \to p(\mathcal{A})$	Policy for agent to perform in the environment	RL
$a_{i,j}$	Non-negative learning factor	REINFORCE
$b_{i,j}$	Representation function of the state for reducing the variance of the gradient estimate.	REINFORCE
g_i	The probability density function for randomly generating unit activation-based actions.	REINFORCE
$L(\theta_i)$	Loss function	DQN

C. Open-source Implementations

Here we summarize the open-source implementations of GRL in this survey in Table C.

Table C. A Summary of Open-source Implementations

Model	Year	Framework	Link	Source
AGILE	2022	PyTorch	https://github.com/clvrai/agile	[30]
GTA-RL	2022	PyTorch	https://github.com/udeshmg/GTA-RL	[34]
SubgraphX	2021	PyTorch	https://github.com/divelab/DIG	[35]
SUGAR	2021	Tensorflow	https://github.com/RingBDStack/SUGAR	[15]
CORL	2021	PyTorch	https://github.com/huawei-noah/trustworthyAI/tree/master/gcastle	[124]

RioGNN	2021	PyTorch	https://github.com/safe-graph/RioGNN	[73]
IG-RL	2021	PyTorch	https://github.com/FXDevailly/IG-RL	[142]
TITer	2021	Python	https://github.com/JHL-HUST/TITer/	[128]
SparRL	2021	PyTorch	https://github.com/rwickman/SparRL-PyTorch	[37]
PAAR	2021	PyTorch	https://github.com/seukgcode/PAAR	[87]
Policy-GNN	2020	PyTorch	https://github.com/lhenry15/Policy-GNN	[41]
CARE-GNN	2020	PyTorch	https://github.com/YingtongDou/CARE-GNN	[36]
RL-BIC	2020	Tensorflow	https://github.com/huawei-noah/trustworthyAI/tree/master/Causal_Structure_Learning/Causal_Discovery_RL	[123]
RL-based Graph2Seq	2020	PyTorch	https://github.com/hugochan/RL-based-Graph2Seq-for-NQG	[126]
DGN	2020	Tensorflow	https://github.com/PKU-AI-Edge/DGN/	[27]
DeepGraphMolGen	2020	PyTorch	https://github.com/dbkgroup/prop_gen	[154]
KG-A2C	2020	PyTorch	https://github.com/rajammanabrolu/KG-A2C	[171]
GPA	2020	PyTorch	https://github.com/ShengdingHu/GraphPolicyNetworkActiveLearning	[14]
GAEA	2020	Tensorflow	https://github.com/salesforce/GAEA	[172]
CompNet	2019	PyTorch	https://github.com/WOW5678/CompNet	[7]
GRPI	2019	Python	https://github.com/LASP-UCL/Graph-RL	[111]
DRL+GNN	2019	PyTorch	https://github.com/knowledgedefinednetworking/DRL-GNN	[145]
GPN	2019	PyTorch	https://github.com/qiang-ma/graph-pointer-network	[161]
PGPR	2019	PyTorch	https://github.com/orcax/PGPR	[173]
DGN	2018	PyTorch	https://github.com/PKU-AI-Edge/DGN	[27]
GCPN	2018	Python	https://github.com/bowenliu16/rl_graph_generation	[10]
KG-DQN	2018	PyTorch	https://github.com/rajammanabrolu/KG-DQN	[174]
ASNets	2018	Tensorflow	https://github.com/qxcv/asnets	[150]
S2V-DQN	2017	C+Python	https://github.com/Hanjun-Dai/graph_comb_opt	[148]
DeepPath	2017	Tensorflow	https://github.com/xwhan/DeepPath	[3]
MINERVA	2017	Tensorflow	https://github.com/shehzaadzd/MINERVA	[92]
KBGAN	2017	PyTorch	https://github.com/cai-lw/KBGAN	[95]

References

[1] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International journal of robotics research,* vol. 37, no. 4-5, pp. 421-436, 2018.

[2] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature,* vol. 518,

no. 7540, pp. 529-533, 2015/02/01 2015.

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

[4] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *International conference on machine learning*, 2016: PMLR, pp. 1329-1338.

[5] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep direct reinforcement learning for financial signal representation and trading," *IEEE transactions on neural networks and learning systems,* vol. 28, no. 3, pp. 653-664, 2016.

[6] J. B. Heaton, N. G. Polson, and J. H. Witte, "Deep learning for finance: deep portfolios," *Applied Stochastic Models in Business and Industry,* vol. 33, no. 1, pp. 3-12, 2017.

[7] S. Wang, P. Ren, Z. Chen, Z. Ren, J. Ma, and M. de Rijke, "Order-free medicine combination prediction with graph convolutional reinforcement learning," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 1623-1632.

[8] K. Do, T. Tran, and S. Venkatesh, "Graph transformation policy network for chemical reaction prediction," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 750-760.

[9] W. Qiu, H. Chen, and B. An, "Dynamic Electronic Toll Collection via Multi-Agent Deep Reinforcement Learning with Edge-Based Graph Convolutional Networks," in *IJCAI*, 2019, pp. 4568-4574.

[10] J. You, B. Liu, Z. Ying, V. Pande, and J. Leskovec, "Graph convolutional policy network for goal-directed molecular graph generation," *Advances in neural information processing systems,* vol. 31, 2018.

[11] M. Obara, T. Kashiyama, and Y. Sekimoto, "Deep Reinforcement Learning Approach for Train Rescheduling Utilizing Graph Theory," in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 4525-4533.

[12] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," *Advances in neural information processing systems,* vol. 31, 2018.

[13] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *International conference on machine learning*, 2016: PMLR, pp. 2071-2080.

[14] S. Hu *et al.*, "Graph policy network for transferable active learning on graphs," *Advances in Neural Information Processing Systems,* vol. 33, pp. 10174-10185, 2020.

[15] Q. Sun *et al.*, "Sugar: Subgraph neural network with reinforcement pooling and self-supervised mutual information mechanism," in *Proceedings of the Web Conference 2021*, 2021, pp. 2081-2091.

[16] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems,* vol. 30, 2017.

[17] H. Yuan and S. Ji, "Structpool: Structured graph pooling via conditional random fields," in *Proceedings of the 8th International Conference on Learning Representations*, 2020.

[18] H. Yuan, J. Tang, X. Hu, and S. Ji, "XGNN: Towards Model-Level Explanations of Graph Neural Networks," presented at the Proceedings of the 26th ACM SIGKDD International Conference

on Knowledge Discovery & Data Mining, Virtual Event, CA, USA, 2020. [Online]. Available: https://doi.org/10.1145/3394486.3403085.

[19] H. Dai *et al.*, "Adversarial attack on graph structured data," in *International conference on machine learning*, 2018: PMLR, pp. 1115-1124.

[20] P. T. A. Quang, Y. Hadjadj-Aoul, and A. Outtagarts, "A deep reinforcement learning approach for VNF forwarding graph embedding," *IEEE Transactions on Network and Service Management,* vol. 16, no. 4, pp. 1318-1331, 2019.

[21] Z. Yan, J. Ge, Y. Wu, L. Li, and T. Li, "Automatic Virtual Network Embedding: A Deep Reinforcement Learning Approach With Graph Convolutional Networks," *IEEE Journal on Selected Areas in Communications,* vol. 38, no. 6, pp. 1040-1057, 2020.

[22] H. Peng *et al.*, "Dynamic graph convolutional network for long-term traffic flow prediction with reinforcement learning," *Information Sciences,* vol. 578, pp. 401-416, 2021.

[23] E. Meirom, H. Maron, S. Mannor, and G. Chechik, "Controlling graph dynamics with reinforcement learning and graph neural networks," in *International Conference on Machine Learning*, 2021: PMLR, pp. 7565-7577.

[24] Y. Lyu, X. Yang, J. Liu, S. Xie, and X. Zhang, "Interpretable and Effective Reinforcement Learning for Attacking against Graph-based Rumor Detection," *arXiv preprint arXiv:2201.05819,* 2022.

[25] S. Zhou *et al.*, "Interactive Recommender System via Knowledge Graph-enhanced Reinforcement Learning," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*: Association for Computing Machinery, 2020, pp. 179–188.

[26] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Basar, "Fully decentralized multi-agent reinforcement learning with networked agents," in *International Conference on Machine Learning*, 2018: PMLR, pp. 5872-5881.

[27] J. Jiang, C. Dun, T. Huang, and Z. Lu, "Graph Convolutional Reinforcement Learning," in *International Conference on Learning Representations*, 2019.

[28] H. Dai, Y. Li, C. Wang, R. Singh, P.-S. Huang, and P. Kohli, "Learning transferable graph exploration," *Advances in Neural Information Processing Systems,* vol. 32, 2019.

[29] J. Jin, S. Zhou, W. Zhang, T. He, Y. Yu, and R. Fakoor, "Graph-Enhanced Exploration for Goal-oriented Reinforcement Learning," in *International Conference on Learning Representations*, 2021.

[30] A. Jain, N. Kosaka, K.-M. Kim, and J. J. Lim, "Know Your Action Set: Learning Action Relations for Reinforcement Learning," in *International Conference on Learning Representations*, 2021.

[31] Z. Zhang, Z. Yang, H. Liu, P. Tokekar, and F. Huang, "Reinforcement Learning under a Multi-agent Predictive State Representation Model: Method and Theory," in *International Conference on Learning Representations*, 2021.

[32] S. Hong, D. Yoon, and K.-E. Kim, "Structure-Aware Transformer Policy for Inhomogeneous Multi-Task Reinforcement Learning," in *International Conference on Learning Representations*, 2021.

[33] H. Liu, S. Zhou, C. Chen, T. Gao, J. Xu, and M. Shu, "Dynamic knowledge graph

reasoning based on deep reinforcement learning," *Knowledge-Based Systems,* vol. 241, p. 108235, 2022/04/06/ 2022.

[34] U. Gunarathna, R. Borovica-Gajic, S. Karunasekara, and E. Tanin, "Solving Dynamic Graph Problems with Multi-Attention Deep Reinforcement Learning," *arXiv preprint arXiv:2201.04895,* 2022.

[35] H. Yuan, H. Yu, J. Wang, K. Li, and S. Ji, "On explainability of graph neural networks via subgraph explorations," in *International Conference on Machine Learning*, 2021: PMLR, pp. 12241-12252.

[36] Y. Dou, Z. Liu, L. Sun, Y. Deng, H. Peng, and P. S. Yu, "Enhancing Graph Neural Network-based Fraud Detectors against Camouflaged Fraudsters," *arXiv: Social and Information Networks,* 2020.

[37] R. Wickman, X. Zhang, and W. Li, "SparRL: Graph Sparsification via Deep Reinforcement Learning," *arXiv preprint arXiv:2112.01565,* 2021.

[38] K. Rusek, J. Suárez-Varela, A. Mestres, P. Barlet-Ros, and A. Cabellos-Aparicio, "Unveiling the potential of Graph Neural Networks for network modeling and optimization in SDN," in *Symposium on SDN Research*, 2019.

[39] J. Suárez-Varela *et al.*, "Challenging the generalization capabilities of Graph Neural Networks for network modeling," in *Proceedings of the ACM SIGCOMM 2019 Conference Posters and Demos*, 2019, pp. 114-115.

[40] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-Lehman Graph Kernels," *The Journal of Machine Learning Research,* vol. 12, pp. 2539-2561, 2011.

[41] K.-H. Lai, D. Zha, K. Zhou, and X. Hu, "Policy-gnn: Aggregation optimization for graph neural networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 461-471.

[42] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Network Representation Learning: A Survey," *IEEE Transactions on Big Data,* vol. 6, no. 1, pp. 3-28, 2020.

[43] M. Sun *et al.*, "Data poisoning attack against unsupervised node embedding methods," *arXiv preprint arXiv:1810.12881,* 2018.

[44] L. Sun *et al.*, "Adversarial attack and defense on graph data: A survey," *arXiv preprint arXiv:1812.10528,* 2018.

[45] Y. Ma, S. Wang, T. Derr, L. Wu, and J. Tang, "Attacking graph convolutional networks via rewiring," *arXiv preprint arXiv:1906.03750,* 2019.

[46] Z. Xi, R. Pang, S. Ji, and T. Wang, "Graph backdoor," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 1523-1540.

[47] Z. Zhang, J. Jia, B. Wang, and N. Z. Gong, "Backdoor attacks to graph neural networks," in *Proceedings of the 26th ACM Symposium on Access Control Models and Technologies*, 2021, pp. 15-26.

[48] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "Dbpedia: A nucleus for a web of open data," in *The semantic web*: Springer, 2007, pp. 722-735.

[49] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *Proceedings of the 2008 ACM SIGMOD*

international conference on Management of data, 2008, pp. 1247-1250.

[50] F. Mahdisoltani, J. Biega, and F. Suchanek, "Yago3: A knowledge base from multilingual wikipedias," in *7th biennial conference on innovative data systems research*, 2014: CIDR Conference.

[51] Q. Wang, Y. Ji, Y. Hao, and J. Cao, "GRL: Knowledge graph completion with GAN-based reinforcement learning," *Knowledge-Based Systems,* vol. 209, p. 106421, 2020/12/17/ 2020.

[52] M. Gardner, P. Talukdar, B. Kisiel, and T. Mitchell, "Improving learning and inference in a large knowledge-base using latent syntactic cues," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013, pp. 833-838.

[53] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," *Advances in neural information processing systems,* vol. 26, 2013.

[54] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2014, vol. 28, no. 1.

[55] T. Wang, R. Liao, J. Ba, and S. Fidler, "Nervenet: Learning structured policy with graph neural networks," in *International conference on learning representations*, 2018.

[56] J. Schrittwieser *et al.*, "Mastering Atari, Go, chess and shogi by planning with a learned model," *Nature,* vol. 588, no. 7839, pp. 604-609, 2020/12/01 2020.

[57] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *nature,* vol. 529, no. 7587, pp. 484-489, 2016.

[58] T. M. Moerland, J. Broekens, and C. M. Jonker, "Model-based reinforcement learning: A survey," *arXiv preprint arXiv:2006.16712,* 2020.

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

[60] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine,* vol. 34, no. 6, pp. 26-38, 2017.

[61] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning,* vol. 8, no. 3, pp. 279-292, 1992.

[62] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning,* vol. 8, no. 3, pp. 229-256, 1992.

[63] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," *Advances in neural information processing systems,* vol. 12, 1999.

[64] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016: PMLR, pp. 1928-1937.

[65] J. X. Wang *et al.*, "Learning to reinforcement learn," *arXiv preprint arXiv:1611.05763,* 2016.

[66] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International conference on machine learning*, 2014: PMLR, pp. 387-395.

[67] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971,* 2015.

[68] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *nature,* vol. 518, no. 7540, pp. 529-533, 2015.

[69] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, 2016, vol. 30, no. 1.

[70] M. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable mdps," in *2015 aaai fall symposium series*, 2015.

[71] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *International conference on machine learning*, 2016: PMLR, pp. 1995-2003.

[72] A. Mukherjee, V. Venkataraman, B. Liu, and N. Glance, "What yelp fake review filter might be doing?," in *Proceedings of the International AAAI Conference on Web and Social Media*, 2013, vol. 7, no. 1.

[73] H. Peng, R. Zhang, Y. Dou, R. Yang, J. Zhang, and P. S. Yu, "Reinforced neighborhood selection guided multi-relational graph neural networks," *ACM Transactions on Information Systems (TOIS),* vol. 40, no. 4, pp. 1-46, 2021.

[74] Y. Deng, Y. Li, F. Sun, B. Ding, and W. Lam, "Unified conversational recommendation policy learning via graph-based reinforcement learning," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 1431-1441.

[75] J. J. McAuley and J. Leskovec, "From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews," in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 897-908.

[76] A. L. Goldberger *et al.*, "PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals," *circulation,* vol. 101, no. 23, pp. e215-e220, 2000.

[77] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine,* vol. 29, no. 3, pp. 93-93, 2008.

[78] Y. Gao, H. Yang, P. Zhang, C. Zhou, and Y. Hu, "GraphNAS: Graph Neural Architecture Search with Reinforcement Learning," *arXiv: Learning,* 2019.

[79] K. Zhou, Q. Song, X. Huang, and X. Hu, "Auto-GNN: Neural Architecture Search of Graph Neural Networks," *arXiv: Learning,* 2019.

[80] L. Wang *et al.*, "Learning Robust Representations with Graph Denoising Policy Network," in *2019 IEEE International Conference on Data Mining (ICDM)*, 2019, pp. 1378-1383.

[81] Y. Sun, S. Wang, X. Tang, T.-Y. Hsieh, and V. Honavar, "Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach," in *Proceedings of the Web Conference 2020*, 2020, pp. 673-683.

[82] J. Leskovec and J. Mcauley, "Learning to discover social circles in ego networks," *Advances in neural information processing systems,* vol. 25, 2012.

[83] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, "Measurement and analysis of online social networks," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, 2007, pp. 29-42.

[84] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM transactions on Knowledge Discovery from Data (TKDD),* vol. 1, no. 1, pp. 2-es, 2007.

[85] H. Wang, S. Li, R. Pan, and M. Mao, "Incorporating graph attention mechanism into

knowledge graph reasoning based on deep reinforcement learning," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 2623-2631.

[86] G. Wan, S. Pan, C. Gong, C. Zhou, and G. Haffari, "Reasoning like human: Hierarchical reinforcement learning for knowledge graph reasoning," in *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, 2021, pp. 1926-1932.

[87] X. Zhou, P. Wang, Q. Luo, and Z. Pan, "Multi-hop Knowledge Graph Reasoning Based on Hyperbolic Knowledge Graph Embedding and Reinforcement Learning," in *The 10th International Joint Conference on Knowledge Graphs*, 2021, pp. 1-9.

[88] M. Zheng, Y. Zhou, and Q. Cui, "Hierarchical Policy Network with Multi-agent for Knowledge Graph Reasoning Based on Reinforcement Learning," in *International Conference on Knowledge Science, Engineering and Management*, 2021: Springer, pp. 445-457.

[89] Q. Wang, Y. Hao, and J. Cao, "ADRL: An attention-based deep reinforcement learning framework for knowledge graph reasoning," *Knowledge-Based Systems,* vol. 197, p. 105910, 2020/06/07/ 2020.

[90] Z. Li, X. Jin, S. Guan, Y. Wang, and X. Cheng, "Path Reasoning over Knowledge Graph: A Multi-agent and Reinforcement Learning Based Method," in *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, 2018, pp. 929-936.

[91] X. V. Lin, R. Socher, and C. Xiong, "Multi-hop knowledge graph reasoning with reward shaping," *arXiv preprint arXiv:1808.10568,* 2018.

[92] R. Das *et al.*, "Go for a Walk and Arrive at the Answer: Reasoning Over Paths in Knowledge Bases using Reinforcement Learning," in *International Conference on Learning Representations*, 2018.

[93] L. Chen, J. Cui, X. Tang, Y. Qian, Y. Li, and Y. Zhang, "RLPath: a knowledge graph link prediction method using reinforcement learning based attentive relation path searching and representation learning," *Applied Intelligence,* vol. 52, no. 4, pp. 4715-4726, 2022/03/01 2022.

[94] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2d knowledge graph embeddings," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018, vol. 32, no. 1.

[95] L. Cai and W. Y. Wang, "Kbgan: Adversarial learning for knowledge graph embeddings," *arXiv preprint arXiv:1711.04071,* 2017.

[96] K. Toutanova, D. Chen, P. Pantel, H. Poon, P. Choudhury, and M. Gamon, "Representing text for joint embedding of text and knowledge bases," in *Proceedings of the 2015 conference on empirical methods in natural language processing*, 2015, pp. 1499-1509.

[97] A. K. Debnath, R. L. Lopez de Compadre, G. Debnath, A. J. Shusterman, and C. Hansch, "Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity," *Journal of medicinal chemistry,* vol. 34, no. 2, pp. 786-797, 1991.

[98] Y. Luo *et al.*, "Automated Data Augmentations for Graph Classification," *arXiv preprint arXiv:2202.13248,* 2022.

[99] H. Toivonen, A. Srinivasan, R. D. King, S. Kramer, and C. Helma, "Statistical evaluation

of the predictive toxicology challenge 2000–2001," *Bioinformatics,* vol. 19, no. 10, pp. 1183-1193, 2003.

[100] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. Vishwanathan, A. J. Smola, and H.-P. Kriegel, "Protein function prediction via graph kernels," *Bioinformatics,* vol. 21, no. suppl_1, pp. i47-i56, 2005.

[101] P. D. Dobson and A. J. Doig, "Distinguishing enzyme structures from non-enzymes without alignments," *Journal of molecular biology,* vol. 330, no. 4, pp. 771-783, 2003.

[102] N. Wale, I. A. Watson, and G. Karypis, "Comparison of descriptor spaces for chemical compound retrieval and classification," *Knowledge and Information Systems,* vol. 14, no. 3, pp. 347-375, 2008.

[103] Z. Wu *et al.*, "MoleculeNet: a benchmark for molecular machine learning," *Chemical science,* vol. 9, no. 2, pp. 513-530, 2018.

[104] H. Yuan, H. Yu, S. Gui, and S. Ji, "Explainability in graph neural networks: A taxonomic survey," *arXiv preprint arXiv:2012.15445,* 2020.

[105] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907,* 2016.

[106] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903,* 2017.

[107] H. Gao, Z. Wang, and S. Ji, "Large-Scale Learnable Graph Convolutional Networks," presented at the Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, United Kingdom, 2018. [Online]. Available: https://doi.org/10.1145/3219819.3219947.

[108] P. Velickovic, W. Fedus, and W. L. Hamilton, "Deep Graph Infomax."

[109] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[110] W.-t. Yih, K. Toutanova, J. C. Platt, and C. Meek, "Learning discriminative projections for text similarity measures," in *Proceedings of the fifteenth conference on computational natural language learning*, 2011, pp. 247-256.

[111] S. Madjiheurem and L. Toni, "Representation Learning on Graphs: A Reinforcement Learning Application," in *International Conference on Artificial Intelligence and Statistics*, 2019.

[112] D. Chen, M. Nie, H. Zhang, Z. Wang, and D. Wang, "Network Embedding Algorithm Taking in Variational Graph AutoEncoder," *Mathematics,* vol. 10, no. 3, 2022.

[113] B. Li and D. Pi, "Network representation learning: a systematic literature review," *Neural Computing and Applications,* vol. 32, no. 21, pp. 16647-16679, 2020/11/01 2020.

[114] X. Huang, D. Chen, T. Ren, and D. Wang, "A survey of community detection methods in multilayer networks," *Data Mining and Knowledge Discovery,* vol. 35, no. 1, pp. 1-45, 2021.

[115] J. Chen, T. Ma, and C. Xiao, "FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling," in *International Conference on Learning Representations*, 2018.

[116] J. B. Lee, R. Rossi, and X. Kong, "Graph Classification using Structural Attention," presented at the Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, United Kingdom, 2018. [Online]. Available: https://doi.org/10.1145/3219819.3219980.

[117] J. Jin *et al.*, "An Efficient Neighborhood-based Interaction Model for Recommendation on Heterogeneous Graph," in *Knowledge Discovery and Data Mining*, 2020.

[118] D. Seyler, P. Chandar, and M. Davis, "An Information Retrieval Framework for Contextual Suggestion Based on Heterogeneous Information Network Embeddings," in *International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2018.

[119] X. Wang *et al.*, "Heterogeneous graph attention network," in *The world wide web conference*, 2019, pp. 2022-2032.

[120] C. Yang, Y. Feng, P. Li, Y. Shi, and J. Han, "Meta-graph based hin spectral embedding: Methods, analyses, and insights," in *2018 IEEE International Conference on Data Mining (ICDM)*, 2018: IEEE, pp. 657-666.

[121] Z. Zhong, C.-T. Li, and J. Pang, "Reinforcement Learning Enhanced Heterogeneous Graph Neural Network," *arXiv: Learning,* 2020.

[122] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial Attacks on Neural Networks for Graph Data," *arXiv: Machine Learning,* 2018.

[123] S. Zhu, I. Ng, and Z. Chen, "Causal Discovery with Reinforcement Learning," in *International Conference on Learning Representations*, 2019.

[124] X. Wang *et al.*, "Ordering-based causal discovery with reinforcement learning," *arXiv preprint arXiv:2105.06631,* 2021.

[125] L. Chen, B. Tan, S. Long, and K. Yu, "Structured dialogue policy with graph neural networks," in *Proceedings of the 27th International Conference on Computational Linguistics*, 2018, pp. 1257-1268.

[126] Y. Chen, L. Wu, and M. J. Zaki, "Reinforcement Learning Based Graph-to-Sequence Model for Natural Question Generation," in *International Conference on Learning Representations*, 2019.

[127] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel, "Self-critical sequence training for image captioning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7008-7024.

[128] H. Sun, J. Zhong, Y. Ma, Z. Han, and K. He, "TimeTraveler: Reinforcement Learning for Temporal Knowledge Graph Forecasting," *arXiv preprint arXiv:2109.04101,* 2021.

[129] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems,* vol. 12, 1999.

[130] W. Song, Z. Duan, Z. Yang, H. Zhu, M. Zhang, and J. Tang, "Ekar: An Explainable Method for Knowledge Aware Recommendation," *arXiv e-prints,* p. arXiv: 1906.09506, 2019.

[131] X. Zhao, L. Zhang, Z. Ding, L. Xia, J. Tang, and D. Yin, "Recommendations with negative feedback via pairwise deep reinforcement learning," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1040-1048.

[132] P. Wang, K. Liu, L. Jiang, X. Li, and Y. Fu, "Incremental mobile user profiling: Reinforcement learning with spatial knowledge graph for modeling event streams," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 853-861.

[133] M. Yousefi, N. Mtetwa, Y. Zhang, and H. Tianfield, "A Reinforcement Learning

Approach for Attack Graph Analysis," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, 2018, pp. 212-217.

[134] R. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "GNNExplainer: generating explanations for graph neural networks," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*: Curran Associates Inc., 2019, p. Article 829.

[135] D. Luo *et al.*, "Parameterized explainer for graph neural network," *Advances in neural information processing systems,* vol. 33, pp. 19620-19631, 2020.

[136] M. Vu and M. T. Thai, "Pgm-explainer: Probabilistic graphical model explanations for graph neural networks," *Advances in neural information processing systems,* vol. 33, pp. 12225-12235, 2020.

[137] D. Silver *et al.*, "Mastering the game of Go without human knowledge," *Nature,* vol. 550, no. 7676, pp. 354-359, 2017/10/01 2017.

[138] P. Shang, X. Liu, C. Yu, G. Yan, Q. Xiang, and X. Mi, "A new ensemble deep graph reinforcement learning network for spatio-temporal traffic volume forecasting in a freeway network," *Digital Signal Processing,* vol. 123, p. 103419, 2022/04/30/ 2022.

[139] Y. Wang, Y. Tong, C. Long, P. Xu, K. Xu, and W. Lv, "Adaptive Dynamic Bipartite Graph Matching: A Reinforcement Learning Approach," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, 2019, pp. 1478-1489.

[140] T. Nishi, K. Otaki, K. Hayakawa, and T. Yoshimura, "Traffic Signal Control Based on Reinforcement Learning with Graph Convolutional Neural Nets," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 877-883.

[141] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation,* vol. 9, no. 8, pp. 1735-1780, 1997.

[142] F.-X. Devailly, D. Larocque, and L. Charlin, "IG-RL: Inductive graph reinforcement learning for massive-scale traffic signal control," *IEEE Transactions on Intelligent Transportation Systems,* 2021.

[143] Z. Yu and M. Hu, "Deep Reinforcement Learning With Graph Representation for Vehicle Repositioning," *IEEE Transactions on Intelligent Transportation Systems,* 2021.

[144] S. Chen, J. Dong, P. Ha, Y. Li, and S. Labi, "Graph neural network and reinforcement learning for multi‐agent cooperative control of connected autonomous vehicles," *Computer‐Aided Civil and Infrastructure Engineering,* vol. 36, no. 7, pp. 838-857, 2021.

[145] P. Almasan, J. Suárez-Varela, A. Badia-Sampera, K. Rusek, P. Barlet-Ros, and A. Cabellos-Aparicio, "Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case," *arXiv preprint arXiv:1910.07421,* 2019.

[146] K. Nakashima, S. Kamiya, K. Ohtsu, K. Yamamoto, T. Nishio, and M. Morikura, "Deep Reinforcement Learning-Based Channel Allocation for Wireless LANs With Graph Convolutional Networks," *IEEE Access,* vol. 8, pp. 31823-31834, 2020.

[147] B. Wang, Y. Sun, T. Q. Duong, L. D. Nguyen, and L. Hanzo, "Risk-Aware Identification of Highly Suspected COVID-19 Cases in Social IoT: A Joint Graph Theory and Reinforcement Learning Approach," *IEEE Access,* vol. 8, pp. 115655-115661, 2020.

[148] H. Dai, E. B. Khalil, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial

optimization algorithms over graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 6351-6361.

[149] J. Yan, S. Yang, and E. Hancock, "Learning for graph matching and related combinatorial optimization problems," presented at the Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, Yokohama, Yokohama, Japan, 2021.

[150] S. Toyer, F. Trevizan, S. Thiébaux, and L. Xie, "Action schema networks: Generalised policies with deep learning," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[151] P. Sun, J. Lan, J. Li, Z. Guo, and Y. Hu, "Combining deep reinforcement learning with graph neural networks for optimal VNF placement," *IEEE Communications Letters,* vol. 25, no. 1, pp. 176-180, 2020.

[152] S. H. Silva, A. Alaeddini, and P. Najafirad, "Temporal Graph Traversals Using Reinforcement Learning With Proximal Policy Optimization," *IEEE Access,* vol. 8, pp. 63910-63922, 2020.

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

[154] Y. Khemchandani *et al.*, "DeepGraphMolGen, a multi-objective, computational strategy for generating molecules with desirable properties: a graph convolution and reinforcement learning approach," *Journal of cheminformatics,* vol. 12, no. 1, pp. 1-17, 2020.

[155] W. Jin, R. Barzilay, and T. Jaakkola, "Multi-objective molecule generation using interpretable substructures," in *International conference on machine learning*, 2020: PMLR, pp. 4849-4859.

[156] N. De Cao and T. Kipf, "MolGAN: An implicit generative model for small molecular graphs," *arXiv preprint arXiv:1805.11973,* 2018.

[157] H. Du, Z. Yan, Q. Xiang, and Q. Zhan, "Vulcan: Solving the Steiner Tree Problem with Graph Neural Networks and Deep Reinforcement Learning," *arXiv preprint arXiv:2111.10810,* 2021.

[158] H. Wang *et al.*, "GCN-RL Circuit Designer: Transferable Transistor Sizing with Graph Neural Networks and Reinforcement Learning," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 2020, pp. 1-6.

[159] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine, "Diversity is all you need: Learning skills without a reward function," *arXiv preprint arXiv:1802.06070,* 2018.

[160] O. Nachum, S. S. Gu, H. Lee, and S. Levine, "Data-efficient hierarchical reinforcement learning," *Advances in neural information processing systems,* vol. 31, 2018.

[161] Q. Ma, S. Ge, D. He, D. Thaker, and I. Drori, "Combinatorial optimization by graph pointer networks and hierarchical reinforcement learning," *arXiv preprint arXiv:1911.04936,* 2019.

[162] C. Tessler, S. Givony, T. Zahavy, D. Mankowitz, and S. Mannor, "A deep hierarchical approach to lifelong learning in minecraft," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017, vol. 31, no. 1.

[163] A. S. Vezhnevets *et al.*, "Feudal networks for hierarchical reinforcement learning," in *International Conference on Machine Learning*, 2017: PMLR, pp. 3540-3549.

[164] P. Peng *et al.*, "Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games," *arXiv preprint arXiv:1703.10069,* 2017.

[165] J. Foerster, I. A. Assael, N. De Freitas, and S. Whiteson, "Learning to communicate with

deep multi-agent reinforcement learning," *Advances in neural information processing systems,* vol. 29, 2016.

[166] S. Sukhbaatar and R. Fergus, "Learning multiagent communication with backpropagation," *Advances in neural information processing systems,* vol. 29, 2016.

[167] D. Chen, M. Nie, J. Yan, D. Wang, and Q. Gan, "Network Representation Learning Algorithm Based on Complete Subgraph Folding," *Mathematics,* vol. 10, no. 4, p. 581, 2022.

[168] H. Van Seijen, M. Fatemi, J. Romoff, R. Laroche, T. Barnes, and J. Tsang, "Hybrid reward architecture for reinforcement learning," *Advances in Neural Information Processing Systems,* vol. 30, 2017.

[169] H. Kawano, "Hierarchical sub-task decomposition for reinforcement learning of multi-robot delivery mission," in *2013 IEEE International Conference on Robotics and Automation*, 2013: IEEE, pp. 828-835.

[170] C. Seshadhri, A. Sharma, A. Stolman, and A. Goel, "The impossibility of low-rank representations for triangle-rich complex networks," in *The National Academy of Sciences*, 2020, vol. 117, no. 11, pp. 5631-5637.

[171] P. Ammanabrolu and M. Hausknecht, "Graph constrained reinforcement learning for natural language action spaces," *arXiv preprint arXiv:2001.08837,* 2020.

[172] G. S. Ramachandran, I. Brugere, L. R. Varshney, and C. Xiong, "GAEA: Graph Augmentation for Equitable Access via Reinforcement Learning," in *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, 2021, pp. 884-894.

[173] Y. Xian, Z. Fu, S. Muthukrishnan, G. d. Melo, and Y. Zhang, "Reinforcement Knowledge Graph Reasoning for Explainable Recommendation," presented at the Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, Paris, France, 2019. [Online]. Available: https://doi.org/10.1145/3331184.3331203.

[174] P. Ammanabrolu and M. Riedl, "Playing Text-Adventure Games with Graph-Based Deep Reinforcement Learning," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 3557-3565.