Challenges and Opportunities in Deep Reinforcement Learning with Graph Neural Networks: A Comprehensive review of Algorithms and Applications

Sai Munikoti, *Student Member, IEEE*, Deepesh Agarwal, Laya Das,
Mahantesh Halappanavar, *Senior Member, IEEE*, Balasubramaniam Natarajan, *Senior Member, IEEE*

Abstract—Deep reinforcement learning (DRL) has empowered a variety of artificial intelligence fields, including pattern recognition, robotics, recommendation-systems, and gaming. Similarly, graph neural networks (GNN) have also demonstrated their superior performance in supervised learning for graph-structured data. In recent times, the fusion of GNN with DRL for graph-structured environments has attracted a lot of attention. This paper provides a comprehensive review of these hybrid works. These works can be classified into two categories: (1) algorithmic enhancement, where DRL and GNN complement each other for better utility; (2) application-specific enhancement, where DRL and GNN support each other. This fusion effectively addresses various complex problems in engineering and life sciences. Based on the review, we further analyze the applicability and benefits of fusing these two domains, especially in terms of increasing generalizability and reducing computational complexity. Finally, the key challenges in integrating DRL and GNN, and potential future research directions are highlighted, which will be of interest to the broader machine learning community.

Index Terms—Deep Reinforcement Learning, Graph Neural Network, Survey, Hybrid DRL-GNN, Deep Learning.

I. INTRODUCTION

In the recent past, deep learning has witnessed an explosive growth in terms of development of novel architectures, algorithms and frameworks for addressing a wide range of challenging real-life problems ranging from computer vision to modeling to control. Among these developments, the use of deep neural networks (DNN) for solving sequential decision making problems within the reinforcement learning (RL) framework, resulting in deep reinforcement learning (DRL) is considered one of the state-of-the-art frameworks in artificial intelligence [1] (§II). This approach finds applications in combinatorial optimization [2], games [3], robotics [4], natural language processing [5], and computer vision [6]. The tremendous success of DRL in these applications can be credited to (1) the ability to tackle complex problems in a computationally efficient, scalable and flexible manner, which is otherwise numerically intractable [7]; (2) high computational efficiency allowing fast generation of high fidelity solutions that are crucial in highly dynamic environments with demand for real-time decisions [8]; (3) the ability to understand environment dynamics and produce near-optimal actions based solely on interactions with the environment, without the need for explicit prior knowledge of the underlying system [9], [10].

While DRL's effectiveness has most popularly been demonstrated in games, it is rapidly being adopted in various other real-life applications. Several of these applications involve environments exhibiting explicit structural relationships that can be represented as graphs. For example, a network of cities in the Travelling Salesman Problem (TSP) or an incomplete knowledge graph are inherently characterized by a graph-based arrangement of the different entities. Methods developed for handling data in the Euclidean space are not well-suited for such environments, that require special treatment in terms of encoding the nodes or aggregating the information from different agents. These aspects are systematically modelled with graph neural networks (GNN), detailed in §II. Incorporation of such structural relationship serves as an auxiliary input, and further improves the quality of solutions.

Recently, researchers have been exploring the advantages of fusing powerful GNN models with DRL to efficiently tackle such graph-structured applications. A thorough review of these hybrid works could be extremely beneficial in identifying challenges and determining future research directions. Furthermore, several review works related to DRL in general, are continuously being published [2], [5]–[15]. However, there are two major shortcomings in these reviews: (1) The majority of these surveys are conducted via the lens of a particular application domain. As a result, they are confined to specific approaches that ignore holistic perspectives across domains; (2) to the best of our knowledge, comprehensive reviews dedicated to the study of the combined potential of DRL and GNN do not exist in the current literature.

Contributions: This paper focuses on a systematic literature review of the fusion of DRL and GNN, and makes the following contributions:
- A rigorous review of articles that employ DRL and GNN spanning theoretical developments (§III-A) and multiple application domains (§III-B).
- A categorization of theoretical and application-specific

S. Munikoti, D. Agarwal and B. Natarajan are with Electrical and Computer Engineering, Kansas State University, Manhattan, KS-66506, USA, (e-mail: saimunikoti@ksu.edu, deepesh@ksu.edu, bala@ksu.edu)

M. Halappanavar is with Data Science and Machine Intelligence group, 99354, PNNL, Richland, USA. (e-mail: mahantesh.halappanavar@pnnl.gov)

L. Das is with Reliability and Risk Engineering Lab, ETH Zurich, 8092, Zurich, Switzerland. (e-mail: laydas@ethz.ch)

contributions of the integrated DRL-GNN efforts is developed (§III). To this end, various attributes are identified for classifying and analyzing existing works (§IV).

- The survey takes a holistic approach to reviewing the literature with special focus on critical aspects of algorithms such as computational efficiency, scalability, generalizability and applicability.
- Both DRL and GNN are still in early stages of development, as is the study of their fusion. Therefore, a thorough investigation of the associated challenges is performed and future research directions are identified (§V).

This review is limited to articles indexed in IEEE Xplore, Scopus and Google Scholar. Initially, the keywords "deep reinforcement" and "graph neural network" are used to select articles from databases. This search led to more than 100 papers from the year 2017 to 2022. The resulting list is filtered to identify articles that include both DRL and GNN which finally led to 40 papers. There are no relevant papers before 2017, indicating the relatively recent history and relevance of this trending research topic. Among the 40 articles, 22 come from conferences proceedings, 8 from journals, and the remaining 10 are preprint manuscripts.

The paper is organized as follows. In §II, we offer a brief methodological background of both DRL and GNN to equip readers to understand the fundamentals prior to looking at the fusion of those techniques. §III presents a comprehensive review of existing literature, including classification based on different novel attributes. In §IV, we discuss our findings in terms of the applicability and unique offerings of an approach involving GNNs and DRL. §V highlights key limitations in the existing literature as well as potential future directions for research. §VI concludes this study.

II. OVERVIEW OF DRL AND GNN

This section provides the foundations of two powerful learning paradigms namely DRL and GNN. We begin by describing RL and how it can be evolved to DRL by using deep learning. Then, we briefly explain the fundamentals of GNN algorithms. This section will equip the readers with the required background knowledge to follow the hybrid works on DRL and GNN (discussed in §III).

A. Deep Reinforcement Learning

Reinforcement learning (RL) is considered the third important branch of machine learning with supervised and unsupervised learning serving as the other two. RL is a sequential decision process where agents are trained to take optimal actions for different scenarios of an environment. The action transitions the environment to a new state and meanwhile the agent gets some reward that quantifies how good or bad the action was. To formulate the sequential decision process, RL employs a well known mathematical concept of Markov decision process (MDP). Typically, an MDP is defined by (X, A, p, R) where X is a finite state space, A is the action space for each state $x \in X$, p is the state transition probability from state x^t at time t to state x^{t+1} at time $t + 1$, and r is the immediate reward value obtained after an action $a \in A$ is performed. The agent's primary goal is to interact with its environment (take state as input) at each time step to find the optimal policy π^* (return action for the current state) in order to reach the goal while maximizing the cumulative rewards (expected return) over the entire time period. Agent takes the state s as input and returns an action a to be taken. At a particular time step t, the expected return R^t is the sum of rewards from the current time step onward till the last time step T. When taking an action, an agent must choose between taking the best action based on previous experiences (exploit) and gathering new experiences (exploration) in order to make better decisions in the future. A common approach to account for the trade-off is the greedy strategy, where the agent takes a random action with a probability ϵ.

In addition, there are real-life situations where agents lack sufficient knowledge about the environment for holistic learning. Therefore, *partially observed MDPs (POMDP)* is designed for these conditions. A POMDP is an MDP where the agent only possess a partial view of the state and its typical expression is similar to that of MDP $(X, a, \omega, T, p, O, r, b_o)$ with extra elements. The new elements include Ω that denotes observation, T signifies time, O represents observation probabilities, and b_o is the initial probability distribution of states.

RL is effective in scenarios, where state and action spaces are limited. However, these spaces are usually large and continuous in real-world applications, and traditional RL methods cannot find the optimal value functions or policy functions for all states in a computationally efficient way [16]. To mitigate the "curse of dimensionality", a deep neural network (DNN) is used as a function approximator and integrated with RL, resulting in the emergence of a new paradigm known as Deep reinforcement learning (DRL). There are several ways to classify existing DRL algorithms such as: model-free vs model-based, value vs policy based, and offline vs online learning. In the following, we will provide the fundamental concepts of these algorithms across different categories.

1) Value based DRL: Value-based methods aim to learn the value of the state or state-action pair and then select actions accordingly. The state-action value function $Q_\pi(s, a)$, expressed in Eq. (1), is the expected return starting from state s, taking action a, and thereafter following a policy π. *Deep Q learning* (DQN) is one of the widely use algorithm in this category [17]. Q-learning enables the agent to choose an action $a \in A$ with the highest Q-value available from state $s \in S$ based on a DNN model which maps discrete state-action space with Q values. DQN is updated every time step following the Bellman optimality equation as shown in Eq. (1), where R is the reward obtained and α is the learning rate which takes values between 0 and 1. DQN is an "off-policy" algorithm, where a target policy is used to take action at the current state X and a different behavior policy is used to select action at the next state.

$$Q_\pi(s,a) = E_\pi(R^t|s^t, a^t) = E_\pi(\sum_{k=0}^{\infty} \gamma^k r^{t+k}|s^t, a^t)$$
$$= (1-\alpha)Q(s,a) + \alpha \left[R + \gamma max_{a' in A} Q(s^{t+1}, a^{t+1}) \right]$$
(1)

A key feature of DQN training is the replay buffer, which stores trajectory information (s^t, a^t, r^t, s^{t+1}) during each step of the training. In DQN, DNN is trained using a minibatch

of a randomly selected sample (experiences) from replay buffer which offers various advantages in terms of sample efficiency, low variance and large learning scope. For each sample, the input (state) is passed through the current DNN to generate an output $\hat{Q}(s,a;\theta)$. The target Q value corresponds to Bellman optimality equation in (1), and is used to minimize the following loss function:

$$L(\theta) = E[R + \gamma max_{a' \in A} Q(s^{t+1}, a') - Q(s,a;\theta)] \quad (2)$$

DQN has many variations to improve its current design, including double DQN and dueling DQN. The max operator in the DQN update equation selects and evaluates an action using the same Q network. As a result, DQN significantly overestimates the value function. *Double DQN* addresses this problem by employing two distinct networks, one for action selection and the other for action evaluation. Similarly, *dueling Q* network approximates the Q function by decoupling the value function and the advantage function.

2) Policy based DRL: These methods learn the policy directly unlike value based methods that learn the values first and then determine the optimal policy. Typically, a parametrized policy π_θ is chosen with parameters constantly updated by minimizing the expected return using a gradient based approach also known as *policy gradient theorem*. They are particularly suitable for very large action space (continuous problems) and learning stochastic policies. Next, we discuss three widely used policy based methods: (i) *REINFORCE*: where parameter updates at a given time step involves only the action taken from the current state – the update relies on estimated return by Monte-Carlo method using episode samples. Since it relies on expected return from the current time step, it works only for the episodic tasks [7]; (ii) *Trust region policy optimization (TRPO)*: where the key idea is to restrict too much change in policy at any one step by constraining the updates. This constraint is in the policy space rather than in the parameter space; and (iii) *Proximal policy optimization (PPO)*: that relies on clipped surrogate objective function to reduce the deviation between new policy and old policy. It is relatively simpler in implementation and empirically performs on par with TRPO [7]

3) Actor-critic DRL: Both value based and policy based algorithms have some limitations. While value based algorithms are not efficient for high dimensional action space, policy based algorithms have high variance in gradient estimates. To overcome these shortcomings, an actor-critic method has been proposed that combines the two approaches [18]. Fundamentally, the agent is trained with two estimators. First, is an actor function that controls the agent's behavior by learning the optimal policy, i.e., provides the best action a^t for any input state X^t. Second is a critic function that evaluates the action by computing the value function. Some of the popular variants of algorithms under this category are discussed next. *Advantage actor-critic (A2C)* consists of two DNNs – one for actor and one for critic [19]. Besides A2C, *asynchronous advantage actor-critic (A3C)* executes different agents in parallel on multiple instances of the environment instead of experience replay as in A2C. Although A3C is memory efficient, its updates are not optimal as different agents work with different version of model parameters. *Deep deterministic policy gradient (DDPG)* is an extension over deterministic policy gradient (DPG), which is designed for continuous action space [20]. DPG defines policy to be the function $\mu_\theta : X \to A$. Here, instead of getting the integral over actions as in stochastic policy, we only need to sum over the state space as action is deterministic. DDPG employs a parameterized actor function with a parameterized critic function that approximates the value function using samples. In this way, DDPG can tackle large variance in policy gradients of actor-only methods.

B. Graph Neural Network

Learning with graph-structured data, such as knowledge graphs, biological, and social networks have recently attracted a lot of research attention. There are numerous benefits of representing data as graphs, such as systematic modeling of relationships, simplified representation of complex problems, etc. It is however challenging to interpret and evaluate such graph-structured data by employing conventional DNN-based learning methods. The fundamental mathematical procedures like convolutions are difficult to implement on graphs owing to their uneven structure, irregular size of unordered nodes and dynamic neighborhood composition. Graph Neural Networks (GNN) address these shortcomings by extending DNN techniques to graph-structured data. GNN architectures can jointly model both structural information as well as node attributes. They provide significant performance improvement for graph-related downstream tasks like node classification, link prediction, community detection and graph classification [21]. Typically, GNN models consist of a message passing scheme that propagates the feature information of the nodes to its neighbors until a stable equilibrium is reached. Several GNN algorithms have been proposed to improve this message passing technique. We discuss some of the key approaches below.

1) GCN: Graph convolutional network [22] is the first effort that employs convolution operations (similar to CNN) on graph-structured data. The core idea behind any GNN is to generate unique Euclidean representation of nodes/links in the graph. Conventionally, spectral methods generate node representation vectors using eigen decomposition, but are computationally inefficient and are not generalizable. GCN overcomes these challenges with its powerful approximation, where the update equation of the node representation vector h_u at a particular layer l is given by:

$$h_u^{(l)} = g\left[\theta^{(l)} h_u^{(l-1)} A^*\right] = \left[\theta^{(l)} h_u^{(l-1)} D^{-\frac{1}{2}} A D^{\frac{1}{2}}\right], \quad (3)$$

where, A is Adjacency matrix and D is degree matrix. A^* is normalized in this way to scale the node features and ensures numerical stability at the same time. It is important to note that GCN relies on the entire graph (i.e., full adjacency matrix) for learning node representation which is inefficient as the number of neighbors of a node can vary from one to thousands or even more and cannot be generalized to graphs of different sizes.
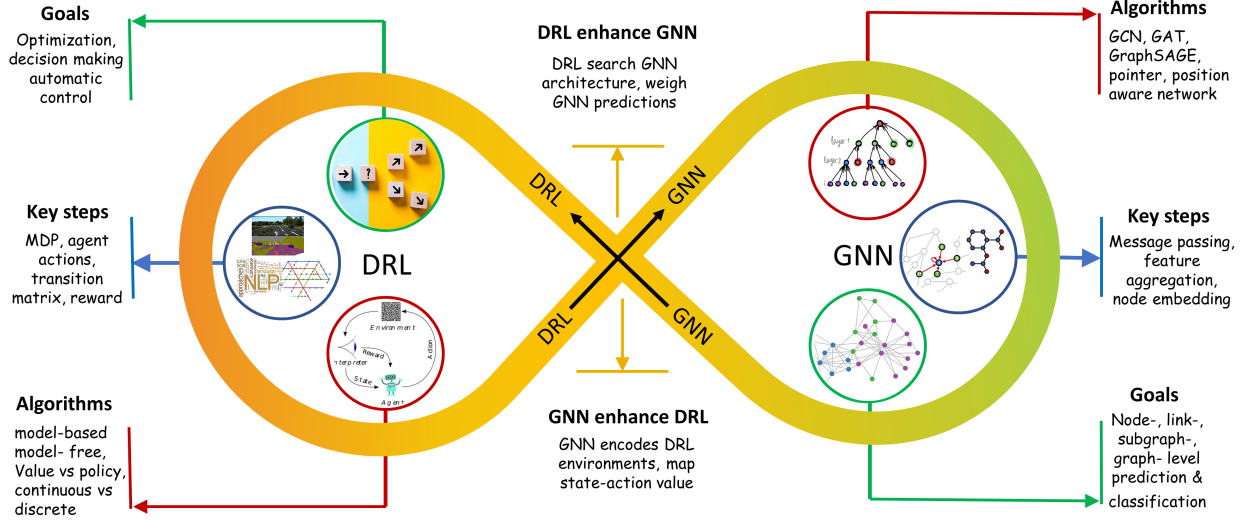
Goals
Optimization, decision making automatic control

DRL enhance GNN
DRL search GNN architecture, weigh GNN predictions

Algorithms
GCN, GAT, GraphSAGE, pointer, position aware network

Key steps
MDP, agent actions, transition matrix, reward

DRL

DRL

GNN

DRL

GNN

GNN

Key steps
Message passing, feature aggregation, node embedding

Algorithms
model-based model- free, Value vs policy, continuous vs discrete

GNN enhance DRL
GNN encodes DRL environments, map state-action value

Goals
Node-, link-, subgraph-, graph- level prediction & classification

Fig. 1. Fusion of Deep reinforcement learning with Graph neural network

2) GraphSAGE: It is an inductive node embedding approach that exploits node attributes to learn an embedding function [23]. It supports simultaneous learning of topological structure as well as distribution of node features within a confined neighborhood. The fundamental premise is to train a neural network that can recognize structural properties of node neighborhood, thereby indicating its local role in the graph along with global position. Initially, the algorithm samples node features in the local neighborhood of each node in the graph-structured data. This is followed by learning appropriate functional mappings to aggregate the information received by each node as it propagates through the GNN layers. This inductive learning approach is scalable across graphs of different sizes as well as subgraphs within a given graph. The operation performed at l^{th} node embedding layer is given by:

$$h_u^{(l)} = f^{(l)}\left(h_u^{(l-1)}, h_{N(u)}^{(l-1)}\right) = g\left[\theta_C^{(l)} h_u^{(l-1)} + \theta_A^{(l)} \tilde{A}\left(h_{N(u)}^{(l-1)}\right)\right]$$

where $\tilde{A}$ represents the aggregation operation; $g[\cdot]$ specifies the activation function; $h_u^{(l)}$ denotes the node embedding of node u at l^{th} layer; $N(u)$ describes the neighborhood of node u; θ_C and θ_A are the parameters of the combination and aggregation operation of GNN, respectively.

3) GAT: Graph attention network (GAT) assumes that contributions of neighboring nodes to the target node are neither predetermined like GCN nor identical like GraphSage. GAT adopts attention mechanisms to learn the relative weights between two connected nodes. The graph convolutional operation according to GAT is defined as follows:

$$h_u^{(l)} = g\left[\sum_{v \in N(u) \cup u} \alpha_{uv}^{(l)} \theta^{(l)} h_u^{(l-1)}\right]$$

$$a_{uv}^l = \text{softmax}\left(g\left[a^T\left(\theta^{(l)} h_u^{(l-1)} || \theta^l h_u^{(l-1)}\right)\right]\right),$$

(4)

where the attention weight α_{uv} quantifies the connection strength between node u and its neighbor v. The attention weight a is learned across all node-pairs using softmax function that ensures weights sum up to one over all neighbors of the node u. This mechanism selectively aggregates the neighborhood contributions and suppresses minor structural details.

III. CATEGORIZATION OF DRL+GNN METHODS

DRL and GNN have emerged as extremely powerful tools in modern deep learning. While DRL exploits the expressive power of DNNs to solve sequential decision making problems with RL, GNNs are novel architectures that are particularly suited to handle graph-structured data. We identify two broad categories of research articles that jointly make use of GNN and DRL, as shown in Fig. 2. The first category of articles makes algorithmic and methodological improvements in the application of DRL (or GNN) by making use of GNN (or DRL). On the other hand, the second category of articles make use of both DRL and GNN to solve practical problems in different application domains. The summary of surveyed DRL and GNN fused works is depicted in Table I and individual components of surveyed papers are outlined in Table II.

A. Algorithmic Developments

In this section, we discuss the articles that focus on developing novel formulations or algorithms to improve DRL or GNN. In these articles, either GNN is used to improve the formulation and performance of DRL, or DRL is used to improve the applicability of GNN.

*1) **DRL enhancing GNN**:* The articles that make use of DRL for improving GNN are used for diverse purposes including neural architecture search (NAS), improving the explainability of GNN predictions and designing adversarial examples for GNN.

Neural Architecture Search (NAS): refers to the process of automatically searching for an optimal architecture of a neural network (eg. number of layers, number of nodes in layer,
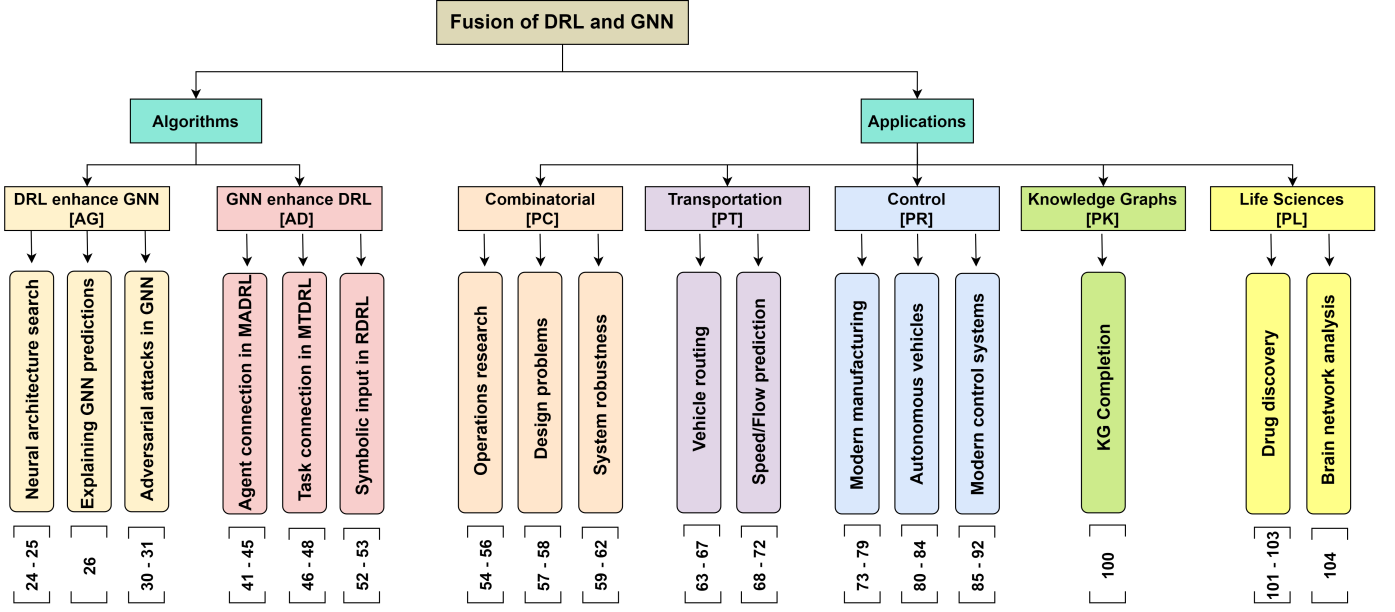
Fig. 2. Classification of hybrid DRL and GNN works

etc.) to solve a particular task. A DRL-based controller that makes use of exploration guided with conservative exploitation is used in [24] to perform an efficient search of different GNN architectures. The search space is made up of hidden dimension, attention head, attention, aggregation, combination, and activation functions. The authors introduce homogeneity of models as a method to perform a guided parameter sharing between offspring and ancestor architectures. The superiority of the proposed method is demonstrated with better performance on benchmark datasets compared to existing methods of architecture search [25].

Explaining GNN predictions: Generating explanations for DNN predictions is an important task in improving the transparency of ML models. Shan et al. [26] use DRL to improve existing methods of explaining GNN predictions. The problem of generating explanations for GNN predictions involves identifying the sub-graph that is most influential in generating a prediction. The authors devise a DRL-based iterative graph generator that starts with a seed node (the most important node for a prediction) and adds edges to generate the explanatory sub-graphs. The DRL model is rewarded with mutual information of predictions and the distribution of predictions based solely on the explanatory sub-graph to learn a sub-graph generation policy with policy gradient. The authors show that the proposed method achieves better explainability in terms of qualitative and quantitative similarity between the generated sub-graphs and the ground truth explanations.

Generating adversarial attacks for GNN: Recent studies [27]–[29] have shown that GNNs are vulnerable to adversarial attacks that perturb or poison the data used for training them. DRL has been used to learn strategies to make adversarial attacks on GNNs, which in turn can be used to devise defense strategies to such attacks. RLS2V [30] is one of the first frameworks that uses DRL to perform an attack aimed at evading detection during classification. Specifically, it employs a Q-learning and structure-to-vector based attack methodology that learns to modify the graph structure (adding or dropping existing edges) with only the prediction feedback (reduction in accuracy) of the target classifier. The authors in [31] consider a novel poisoning attack (NIPA) on graph data that injects fake nodes (e.g., fake accounts in a social networks) into the graph, and uses carefully crafted labels for the fake nodes together with links between them and other (fake as well as genuine) nodes in the graph to poison the graph data. NIPA frames the sequential addition of adversarial connections and the design of adversarial labels for the injected fake nodes as an MDP and solves it with deep Q-learning. To effectively cope with the large search space, NIPA adopts hierarchical Q-learning and GCN based encoding of the states into their low-dimensional latent representations to handle the non-linearity of the mapping between states and actions.

*2) **GNN enhancing DRL**:* This subsection discusses the papers related to the algorithmic improvement of DRL. Specifically, we focus on efforts wherein GNN has been used for relational DRL problems (RDRL) for effective modeling of the relationship among (1) different agents in a multi-agent deep reinforcement learning (MADRL) framework, and (2) different tasks in multi-task deep reinforcement learning (MTDRL) framework.

Modeling relationship among agents in MADRL: In MADRL, a group of agents cooperate or compete with each other to achieve a common goal. This framework has recently been used for a number of challenging tasks, including traffic light control, autonomous driving, and network packet delivery [32]–[34]. In such scenarios, the communication among agents offers additional information about the environment and state of other agents. Several methods have been proposed to learn this communication. The first body of work in capturing these relationships is related to attention-based approaches [35]–[38]. ATOC [39], DGN [40], and COMA-GAT [36] pro-

vide communication through the attention mechanism. Along these lines, G2ANet [41] employs hard attention to filter out irrelevant data and soft attention to focus on relevant information. DCG [42] employs coordination graphs, which uses message passing mechanism to coordinate the behaviors of agents. For each agent, these attention-based algorithms learn the significance distribution of other agents. The authors in GraphComm [43], explore both static and dynamic relations simultaneously among agents. Specifically, it leverages the relational graph module to incorporate static relationships via relational graph provided by prior system knowledge and uses proximity relational graph for dynamic relationships. Agents' Q values are learned in a CTDE manner via MLP and GRU network along with RGCN and GAT to exchange messages among agents for static and dynamic relations, respectively.

Similarly, in [44], Zhang et al. propose a Structural Relational Inference Actor-Critic (SRI-AC) framework for CTDE that can automatically infer the pairwise interaction between agents and learn a state representation. The model is used to predict which agents need to interact in advance, and then supply the most relevant agent observation information to the critic network. In particular, each agent has a critic, which leverages information from the combined action as well as appropriate observational data during training. Then a variational autoencoder (VAE) is used to infer the pairwise interaction, and state representation from observed data, followed by a critic network that employs GAT to integrate the knowledge from neighboring agents. In a similar vein, [45] presents a novel state categorization method for CTDE DRL. Basically, it separates the state into agents' own observations, allies' partial information, and opponents' information specific to the Starcraft game, and then leverages GAT to learn the correlation and relationship among the agents.

Modeling relationship among tasks in MTDRL: This framework provides an elegant way to exploit commonalities between multiple tasks in order to learn policies with improved returns, generalization, data efficiency, and robustness. One of the inherent assumptions in a majority of MTDRL works is compatible state-action spaces, i.e., same dimensions of states and actions across multiple tasks. However, this is violated in many practical applications like combinatorial optimization and robotics. This issue has been addressed by using GNNs that are capable of processing graphs of arbitrary size, thereby supporting MTDRL in incompatible state-action environments [46]. Since GNNs provide the flexibility to incorporate structural information, it enables the integration of additional domain knowledge, where states are characterized as labeled graphs. The use of GNN in MTDRL has been demonstrated in continuous control environments that exploit physical morphology of the RL agents for constructing input graphs [47], [48]. Here, limb features are encoded in the form of node labels and edges represent the physical connections between the corresponding limbs. In this way, the structure of the agents is explicitly modeled in the form of graphs. NerveNet [47] serves as the policy network which first propagates information over agent structure, followed by predicting actions for different parts of the agent. The authors in [48] formulate a single global policy that can be represented as a collection of modular neural networks called Shared Modular Policies (SMP), each of which is designated to handle tasks related to its corresponding actuator.

Relational symbolic input for RDRL: The fundamental premise of RDRL is to integrate DRL with relational learning or Inductive Logic Programming [49], wherein states, actions and policies are represented by a first-order/relational language [50]. The tasks in this space are characterized by variable state and action spaces. In these problems, it is difficult to find fixed-length representation that is required by a majority of the existing DRL methodologies. This issue can be handled using GNN by formulating relational problems in terms of graph-structured data. The mechanics of a relational domain are typically represented by Relational Dynamic Influence Diagram Language (RDDL) [51]. Garg et al. [52] propose SymNet for automated extraction of objects, interactions, and action templates from RDDL. Node embeddings are generated using GNN and action templates are applied over object tuples to create a probability distribution. This is followed by updating the model using policy gradient method. However, SymNet is computationally expensive and is applicable only when RDDL domain definition is available since pre-defined transition dynamics are required to construct the graphs. Symbolic Relational DRL (SR-SRL) [53] addresses these limitations by considering an enriched symbolic input comprising of objects and relations along with their features in the form of a graph. This does not require information about transition dynamics and generalizes over any number of object tuples.

B. Applications

The second broad category of articles exploit the versatility of DRL along with the flexible encoding capability of GNNs to address interesting challenges in different application domains. These domains span a wide spectrum, including combinatorial optimization, transportation, control, knowledge graphs and life sciences, which we briefly review next.

*1) **Combinatorial optimization (CO)**:* Many CO problems are computationally expensive and require approximations and heuristics to solve in polynomial time. There has been an increasing interest in solving CO problems using machine learning techniques. In this regard, CO problems are often framed as an MDP where the optimal actions/solutions can be learnt with DRL. Further, the underlying environment is represented as graph that is processed using GNN. The articles addressing these challenges can further be divided into following sub-categories:

Solving CO problems in Operations Research: Manchanda et al. [54] use GNN to capture the structural information of CO problems and thus address the poor generalizability and scalability of existing DRL-based approaches. They learned a construction heuristic for a budget-constrained maximum vertex cover (MVC) problem by combining supervised learning and DRL. GCN is first utilized to find appropriate candidate nodes by learning the scoring function computed using the probabilistic greedy approach. Then, the candidate nodes are used in an algorithm similar to [55] to sequentially construct a solution. Since the degree of nodes in large graphs can be rather high, importance sampling based on computed score

is used to select the neighboring nodes while determining embeddings, thereby reducing the computational complexity. Extensive experiments on random/real-world graphs reveal that the proposed method marginally outperforms S2V-DQN and scales to much larger graph instances up to a hundred thousand nodes. Additionally, it is significantly more efficient in terms of the computation efficiency due to a relatively smaller number of learned parameters.

Another interesting application of DRL and GNN can be seen in solving diffusion processes in graphs, such as influence maximization, and epidemic test prioritization, among others. The goal is to identify a set of nodes on a temporally evolving graph such that the global objective of curbing spread or maximizing information spread is achieved. Various graph-theoretic algorithms have been developed to address this class of problems. However, they are inefficient when scaling to larger graphs. Further, the added difficulty is that the states are partially observed, for instance, we might not know the ground truth infection status for every node in the graph at any point in time. To address these challenges, [56] poses the problem of controlling a diffusive process on a temporally evolving graph as a partially-observed Markov decision process (POMDP). The problem of selecting a subset of nodes for dynamical intervention is formulated as a ranking problem, and an actor-critic proximal policy optimization is employed to solve it. Specifically, the architecture contains two separate GCN modules; one updates the node representation according to the dynamic process and the other is in charge of long-range information propagation. The results on various real-world networks, including COVID-19 contact tracing data, show the superior performance of this approach.

Solving design problems: Several design problems, especially electronic circuit design are CO problems, which can benefit from a DRL-GNN formulation. For instance, automatic transistor sizing is a challenging problem in circuit design due to the large design space, complex performance tradeoffs, and fast technological advancements. The authors in [57], present the GCN-RL Circuit Designer, which uses DRL to transfer knowledge between different technology nodes and topologies. GCN is employed to learn the circuit topology representation. The GCN-RL agent retrieves features of the topology graph with transistors as nodes and wires as links. Actor critic approach with continuous space DRL algorithm DDPG is used. The generalizability of DRL enables training on one technology node and then applying the trained agent to search the same circuit under different technology nodes. GCN extracts circuit features which enables transfer of knowledge between different topologies that share similar design principles, for example, between a two-stage and three-stage trans-impedance amplifier.

A similar problem is Logic synthesis for combinational circuits, in which the lowest equivalent representation for Boolean logic functions is sought. A widely used logic synthesis paradigm represents Boolean logic with standardized logic networks, such as and-inverter graphs (AIG), which iteratively conducts logic minimization operations on the graph. To this end, [58] poses this problem as an MDP and DRL is incorporated with GCN to explore the solution search space.

Specifically, this work leverages Monte Carlo policy gradient based RL algorithm, REINFORCE. Since circuits and AIG logic can be naturally modeled as graphs, they leverage GCN to extract the current state's features.

System robustness: Recently, [59] demonstrates a novel application of DRL with GNN, where the authors used DRL to search for optimal graph topology for a given graph objective. Essentially, the construction of a graph is framed as a sequential decision process of adding a fixed number of links to the current graph one at a time such that the robustness score of the final graph is the maximum among all feasible combinations of the given graph and edges. Particularly, the state represents the current graph, while the action corresponds to a new node that needs to be added. They are encoded using the S2V variant of GNN [60], and DQN constitutes the underlying DRL engine. Although this approach is more computationally efficient than conventional approaches (such as greedy, Fiedler vector, etc.), it requires an iterative algorithm at each step of the episode to compute the global score of the current graph (robustness in this case), which demands some computational effort. This can be avoided by using learning-based models to compute intermediate rewards, i.e., global graph scores [61], [62].

2) Transportation: Transportation problems that are handled with DRL and GNN can be broadly classified into two classes, namely routing and speed prediction.

Vehicle routing: One of the early efforts to apply GNN with DRL in vehicle routing problems (VRP) can be found in Traveling Salesperson Problem (TSP), where the objective is to find the shortest possible route that visits each node in the graph exactly once and returns to the source node [63]. Here, the state is denoted by a graph embedding vector that describes the tour of the nodes until time step t, whereas action is defined as selecting a node from the non-visited pool and the reward is the negative tour length. A GNN with attention mechanism is used as an encoder followed by a pointer network decoder. The described encoder–decoder network's parameters are updated using the REINFORCE algorithm with a critic baseline. The methodology proposed by [64] adopts a GNN representation to offer a general framework for model-free RL that adapts to different problem classes by altering the reward. This framework uses the edge-to-vertex line graph to model problems and then formulates them in a single-player game framework. The MDPs for TSP and VRP are the same as in [65]. Rather than employing a full-featured Neural MCTS, [64] represents a policy as a graph isomorphism network (GIN) encoder with an attention-based decoder, which is learned throughout the tree-search operation. Further, [66] proposes to learn the improvement heuristics (methods that start from an arbitrary policy and improve iteratively) for VRP in a hierarchical manner. The authors devised an intrinsic MDP that includes not only the present solution's features but also the running history. REINFORCE method is used to train the policy, which is parameterized by a GAT.

Another important problem of cooperative combinatorial optimization in TSP is related to optimization of the multiple TSPs (MTSP). [67] developed an architecture consisting of a shared GNN and distributed policy networks to learn a common policy representation to produce near-optimal solutions

for the MTSP. Specifically, Hu et al. use a two-stage approach, where REINFORCE is used to learn an allocation of agents to vertices, and a regular optimization method is used to solve the single-agent TSP associated with each agent.

Speed/flow prediction: The second class of transportation problems deals with the prediction of speed/flow in the road network commonly referred as traffic signal control (TSC). In recent years, TSC has been modeled as an MDP and researchers have adopted DRL to control the traffic signals [68]–[71]. The authors in [70] propose Inductive Heterogeneous Graph Multi-agent Actor-critic (IHG-MA) algorithm consisting of three steps: (i) sampling of heterogeneous nodes via fast random walk with restart approach, (ii) encoding heterogeneous features of nodes in each group using Bi-GRUs, and (iii) aggregating embeddings of groups using graph attention mechanism. Finally, the proposed MA framework employs the actor-critic approach on the obtained node embeddings to compute the Q-value and policy for each SDRL agent, and optimizes the whole algorithm to learn the transferable traffic signal policies across different networks and traffic conditions. Shang et al. [72] proposed to use a DQN agent to effectively combine the predictions of GCN and GAT, thus improving the overall space-time modeling capabilities and forecasting performance. The DQN provides weights to combine the GCN and GAT predictions, where the weights are adaptive to different network topology, weather conditions, and other relevant attributes of the traffic data.

*3) **Manufacturing and control**:* DRL has also been explored in modern manufacturing systems because of the increasing complexity and interdependency across processes and system-levels [73]–[75]. Recently, Huang et al. [76] proposed an integrated process system model based on GNN. Here, the manufacturing system is represented as a graph, where machines are treated as nodes and material flow between machines as links. GCN is used to encode machine nodes and obtain a node's latent representation that reflects both the local condition of the machine (i.e., parameters of neighboring machines) and the global status of the entire system. Each machine is modeled as a distributed agent, and MARL is trained to learn an independent adaptive control policy conditioned on the node's latent feature vector. The latent characteristics of the node, machine process parameters, and total yield with defects serve as the state, action, and reward of the underlying MDP, respectively. Specifically, C-COMA [36] has been deployed by employing the Advantage Actor Critic (A2C) framework in a distributed setting and is easily compatible with GNN.

In manufacturing, job shop scheduling problem (JSSP) is also an important problem that aims to determine the optimal sequential assignments of machines to multiple jobs consisting of series of operations while preserving the problem constraints. Park et al. [77] propose a framework to construct the scheduling policy for JSSPs using GNN and DRL. They formulate the scheduling of a JSSP as a semidefinite programming problem (SDP) in a computationally efficient way by representing the state of a JSSP as a disjunctive graph [78], where nodes represent operations, conjunctive edges represent precedence/succeeding constraints between two nodes, and disjunctive edges represent machine-sharing constraints between two operations. Then, they employ a GNN to learn node embeddings that summarize the spatial structure of the JSSP and derive a scheduling policy that maps the embedded node features to scheduling action. Proximal policy optimization (PPO) algorithm, a variant of policy-based RL, is used to train GNN-based state representation module and the parameterized decision-making policy jointly [79].

Another key application of DRL is in the control of connected autonomous vehicles [80]–[83]. However, DRL-based controllers in most existing literature address only a single or fixed number of agents with both fixed-size observation and action spaces. This is because of the highly combinatorial and volatile nature of CAV networks with dynamically changing number of agents (vehicles) and the fast-growing joint action space associated with multi-agent driving tasks, which pose difficultly in achieving cooperative control. Recently, Chen et al. [84] presented a DRL based algorithm that combines GCN with DQN to achieve efficient information fusion from multiple resources. A centralized multi-agent controller is then built upon the fused information to make collaborative lane changing decisions for a dynamic number of CAVs within the CAV network.

Efficient allocation of communication resources in wireless networks that are commonly used in modern control systems to exchange data across a vast number of plants, sensors, and actuators is also addressed with DRL [85], [86]. These DRL techniques, however do not scale well with the network size. To overcome this issue, Lima et al. [87] employ a GNN to parameterize the resource allocation function. In particular, Gama et al. [88] use random edge GNNs since the underlying communication graph is randomly distributed, and then coupled it with REINFORCE since the action space is continuous.

Another interesting application can be found in multi agent formation control. Although a number of algorithms can achieve formation control effectively, they ignore the structure feature of the graph formed by agents [35], [89], [90]. Wang et al. [91] proposed a model named MAFCOA building on the framework of GAT. In particular, the model can be divided into two parts including formation control and obstacles avoidance. The first part uses GAT and focuses on cooperation among agents, while the second part focuses on obstacle avoidance with multi-LSTM models. The Multi-LSTM allows the agents to take obstacles into consideration in the order of distance and avoid arbitrary number of obstacles [92]. Moreover, in order to scale to more agents, the parameters are shared to train all the agents in a decentralized framework. Actor and critic approach is used with MADDPG to learn the optimal control policy for multiple agents.

*4) **Knowledge graph completion**:* Knowledge Graphs (KG) are increasingly being used to represent heterogeneous graph-structured data in a wide variety of applications including recommendation systems [93], social networks [94], question-answering systems [95], smart manufacturing [96], information extraction [97], semantic parsing [98] and named entity disambiguation [99]. One of the key problems in real-world knowledge bases is that they are notoriously incomplete, i.e., a lot of relationships are missing. KG Completion (KGC)

is a knowledge base completion process that aims to fill-in the incomplete real-world knowledge bases by inferring missing entries with the help of existing ones. The entities and corresponding relations are represented by means of triplets consisting of head nodes (h), relations (r), and tail nodes (t). The problem of KGC entails prediction of missing tails for given pairs of head nodes and relations. Traditional RL-based methods do not consider generation of new subgraphs within existing knowledge graphs, e.g., new or missing target entities. Moreover, the domain-specific rules are not incorporated, which could be utilized to learn state transition processes when KGC is posed as a Markov process. Finally, the issue of reward sparsity leads to large variance of sampling methods and low learning efficiency. In order to overcome these limitations, [100] proposed a GAN-based DRL framework (GRL). The authors divide the problem into two scenarios: when the target entity can be located within limited time steps, and when the target entity cannot be found from the original KG while there are still time steps to go, in which case a new sub-graph is formed. KGC being defined as an MDP, explores the rules that can be introduced in both the state transition process and rewards to better guide the walking path under the optimization of GAN. LSTM is employed as a generator of GAN, which not only records previous trajectories (of states, actions, etc.) but also generates new sub-graphs and trains policy networks with GAN. Furthermore, to better generate new sub-graphs, a GCN is used to embed the KG into low-dimensional vectors and parameterize the message passing process at each layer. In addition, GRL also applies domain-specific rules and utilizes DDPG to optimize rewards and adversarial loss.

5) Life sciences: Along with engineering applications, recent advancements in ML have also demonstrated the potential to revolutionize various life sciences applications such as drug discovery [101]–[103] and brain network analysis [104]. To this end, [101] proposes a new method for designing antiviral candidates coupling DRL to a deep generative model. Specifically, the authors use the actor critic approach, in which a scaffold-based generative model is leveraged as the actor model to build valid 3D compounds. For the critic model, parallel GNNs are used as a binding probability predictor to determine whether the generated molecule actively binds with a target protein [102]. The results demonstrated that the model could produce molecules with higher druglikeness, synthetic accessibility, water solubility, and hyrdophilicity than current baselines. Do et al. [103] proposed a Graph Transformation Policy Network (GTPN) that combines the strengths of DRL and GNN to learn reactions directly from data with minimal chemical knowledge. Their model has three key components: a GNN, a node pair prediction network, and a policy network. The GNN is responsible for obtaining the atom's representation, the node pair prediction network is responsible for computing the most possible reaction atom pairs, and the policy network is responsible for determining the optimal sequence of bond changes that transforms the reactants into products. Additionally, the model's step-by-step creation of product molecules allows it to exhibit intermediate molecules, greatly improving its interpretability.

IV. DISCUSSION AND LESSONS LEARNED

Supported by an extensive review, we observe that the use of GNNs in a DRL framework is becoming increasingly popular from an algorithmic development perspective and applications of machine learning to complex problems. In this section, we present our perspectives in terms of applicability and advantages of fusing these learning frameworks.

A. Advantages of fusing DRL and GNN

As discussed before, GNN and DRL are fused in two different fronts, i.e., algorithmic enhancement where methodologies are enhancing each other and application where algorithms are supporting each other. This fusion has several advantages that can be summarized as follows:
(**1**) On moving from single agent to multi-agent or from single task to multi-task scenarios in DRL, the complexity of problem drastically increases. Therefore, various new approaches are continuously being proposed to improve the model performance. However, there is always a scope to incorporate auxiliary information for further improvement. Since MADRL/MTDRL involves multiple agents, incorporating the relational information among these agents in the core model with a GNN architecture can improve its performance. *Since GNNs are inherently designed to capture topological/attributed relationships, they are powerful models that allow capturing the multi-agent and multi-task relationships relative to other models*;
(**2**) GNN, like other DNN models require further improvements in terms of automatic setting generation, improving model explainability and enhancing robustness against adversarial attacks. These tasks can easily be handled via DRL due to inherent sequential nature. *DRL is well-suited compared to traditional optimization based approaches for these tasks since it offers a computationally lightweight framework to tackle large problem spaces in a scalable and generic way*;
(**3**) The performance of DRL in applications involving graph-structured environment such as knowledge graphs and transportation networks depend on encoder to a large extent. Therefore, GNNs are used to represent trajectory information in such environments and also act as a function approximator. *GNNs are very effective in representing/encoding graphs compared to other techniques such as graph signal processing or spectral graph theoretic approaches. Further, they are flexible and generic enough to work for different graph families and sizes.*

B. Problem-specific applicability of DRL and GNN methods

A fusion of GNN and DRL has found itself a set of niche problems that span diverse applications, while sharing common features. These common features are: (**1**) Sequential decision making setting of the problem, wherein learning occurs via interactions with the environment in a closed loop manner; (**2**) The learning agent exploits its acquired knowledge at any time, while also striking a balance between exploring multiple options for a potentially better solution; (**3**) The learning is aimed at achieving long term goals and avoid making myopic decisions; (**4**) The underlying system is most efficiently represented as a graph, thereby making GNNs the natural choice for representing such systems. A widely

TABLE I

SUMMARY OF SURVEYED DRL AND GNN FUSED WORKS. AD:ALGORITHMS-DRL ENHANCING GNN, AG:ALGORITHMS-GNN ENHANCING DRL, PC:APPLICATIONS IN COMBINATORIAL OPTIMIZATION, PT:APPLICATIONS IN TRANSPORTATION, PR:APPLICATIONS IN CONTROL, PK:APPLICATIONS IN KNOWLEDGE GRAPH, PL:APPLICATIONS IN LIFE SCIENCE

Reference	Category	Dynamic	Scalable	Generalizable across envs	Multiagent	Source code	Publication venue-year
Algorithms							
Zhou et al. [24]	AD		■	■			arxiv 2019
Gao et al. [25]	AD		■	■		■	IJCAI 2020
Shan et al. [26]	AD	■		■			NeurIPS 2021
Dai et al. [30]	AD		■	■		■	ICML 2018
Sun et al. [31]	AD		■	■			WWW 2020
Liu et al. [41]	AG	■	■	■	■		AAAI 2020
Bohmer et al. [42]	AG	■	■	■	■	■	ICML 2020
Shen et al. [43]	AG	■	■		■		ICASSP 2021
Zhang et al. [44]	AG	■	■		■		Elsevier 2021
Yun et al. [45]	AG	■	■	■	■		IEEE SMC-2021
Wang et al. [47]	AG	■	■	■	■	■	ICLR-2018
Huang et al. [48]	AG		■	■	■	■	ICML-2020
Garg et al. [52]	AG	■	■	■	■	■	ICML-2020
Janisch et al. [53]	AG	■	■	■		■	arxiv 2021
Applications							
Manchanda et al. [54]	PC	■	■	■		■	NeurIPS-2020
Khalil et al. [55]	PC		■	■		■	NeurIPS-2017
Meirom et al. [56]	PC	■	■	■			ICML-2021
Wang et al. [57]	PC		■	■			DAC-2020
Darvariu et al. [59]	PC		■	■		■	PRSA-2021
Drori et al. [64]	PT		■	■			ICMLA-2020
Lu et al. [66]	PT		■			■	ICLR-2020
Hu et al. [67]	PT		■	■			Elsevier-2020
Devailly et al. [69]	PT	■	■	■		■	IEEE-2021
Shang et al. [72]	PT	■	■	■			Elsevier-2022
Lima et al. [86]	PR	■	■	■			IFAC-2020
Wang et al. [91]	PR	■	■	■			IFAC-2020
Huang et al. [76]	PR	■					CIRP-2021
Park et al. [77]	PR	■	■	■			T&F-2021
Chen et al. [84]	PR	■			■		Wiley-2021
Wang et al. [100]	PK	■	■	■			Elsevier-2020
McNaughton et al. [101]	PL	■	■	■			ICLR-2022
Do et al. [103]	PL		■	■			KDD-2019

studied example of such a problem is the traveling salesperson problem, where the process of finding the optimal route is a sequential process of identifying nodes that lead to minimum total distance travelled. Furthermore, the underlying problem possesses a graph structure with nodes being destinations and links representing connection between them.

The majority of applications in literature involve static systems, so that a single GNN module can serve as both a function approximator and an encoder for the environment. However, depending on the nature of the problem, an appropriate GNN algorithm must be chosen for the best performance. Environments involving large graphs should rely on GraphSAGE [23] rather than GCN [22], as GraphSAGE is a sub-graph-based inductive learning approach that is scalable to larger networks. Similarly, applications where the position of a node with respect to the entire graph is vital, position-aware GNNs (PGNN) [105] are preferred. PGNN explicitly make use of anchor nodes along with neighboring sub-graph to improve the effectiveness of node embeddings. Furthermore, the expressive power of most GNNs is upper-bounded by the 1-Weisfeiler-Lehman (1-WL) graph isomorphism test, i.e., they cannot differentiate between different d-regular graphs.

Therefore, it is recommended to explore identity-aware GNN [106] for complex graph-structured environments, which inductively considers nodes' identities during message passing.

Along with static graphs, there are certain applications that involve dynamic graph-structured environments. For instance, in a connected autonomous vehicle network, the number of vehicles dynamically changes. Under this scenario, an appropriate strategy would be to use LSTMs fused with GNNs for capturing graph evolution as well as DRL trajectories. At any instant, the spatial information of the environment can be gathered via GNN and fed to an LSTM cell state for learning long-range spatio-temporal dependency. Furthermore, separate GNNs can be used to encode topological changes and long-range dependencies individually. In addition to the type of environment, the problem of interest can have a single learning agent or multiple agents. In a multi-agent application without any interaction between agents, traditional MADRL algorithms are most suited. However, in certain scenarios, the agents might interact with each other in search of a better solution. These interactions can further be predefined or might appear as an agent interacts with the environment. GNN models can be used to capture such relationships and

TABLE II
DRL AND GNN COMPONENTS OF THE SURVEYED PAPERS

Ref.	Category	State	Action	Reward	DRL	GNN	Remarks
Algorithms							
[25]	AD	Activation function in GNN layers	Changes in activations, aggregation, hidden units, # heads	Validation accuracy	PPO	GCN, GAT, LGCN	LSTM encodes graph architecture & DRL for optimizing accuracy
[26]	AD	Node feature and current subgraph	validation performance + controller entropy	Prediction loss	GraphSAGE	GAT	Identify most influential subgraph to interpret node prediction
[31]	AD	Intermediate poisoned graph	Add adversarial edge & change labels of injected nodes	Node classification accuracy	DQN	GCN	novel non-target specific node injection poisoning attack on graphs
[45]	AG	characteristics of agents; partial information of other agents and enemies	cooperative actions of individual agents	joint-action value function	DRQN	GAT	Novel MADRL algorithm to control multiple agents in RTS games.
[48]	AG	positions, velocity, rotations	position with lower and upper bound	actuator response	DDPG	Message passing	shared modular policy that is generalize to variants (several planar agents with different skeletal structures) not seen during training
[53]	AG	objects and their features, relations, global context	object labels, set of parameters, preconditions	goal specifications	A2C	GAT	Generic framework for solving relational domains
Applications							
[54]	PC	candidate nodes in solution set	Adding a node to solution set	marginal gain of action function	DQN	GCN	Scalable constrained learning of combinatorial algorithms
[55]	PC	nodes in solution set	Adding a node to solution set	marginal gain in cost function	DQN	structure2vec	First learning framework for graph combinatorial
[59]	PC	Graph and edge stub(node)	Adding a node to solution set	gain in graph robustness score	DQN	structure2vec	Learning graph construction using DRL and GNN
[63]	PT	Partial set of visited cities	next city to visit	Tour length	REINFORCE	GAT	Learning clever heuristics
[69]	PT	Current Connectivity and demand	Switch to next phase or continue with current	number of vehicles stopped on a lane	Double Duelling DQN	GCN	Inductive learning applicable to various road networks & traffic distribution
[72]	PT	GNN parameters	Modify GNN parameters	Target function error	DQN	GCN & GAT	Ensemble model weighing GCN and GAT predictions
[91]	PR	Agents local observations	Unit movement in X & Y directions	Distance from target point	REINFORCE	REGNN	Leverages graph attention & LSTM for cooperative agent behavior
[76]	PR	node (machine) latent features	machine control settings	difference between step wise yield & defect	C-COMA	GCN	Distributed adaptive control for multistage systems
[77]	PR	Snapshot of job shop at specific transition	Loading/skipping scheduling	negative of number of waiting jobs	PPO	GCN	Generic framework for Job shop scheduling incorporating spatial and temporal structure
[83]	PK	Combination of entity and relation space	Selecting neighboring relational path to another entity	Closeness of new state to target entity	DDPG	GCN	Leverages GAN and LSTM to capture graph dynamics for knowledge graph completion
[101]	PK	Combination of entity and relation space	Selecting neighboring relational path to another entity	Closeness of new state to target entity	A2C	GCN	knowledge graph completion with domain specific rules
[101]	PL	Intermediate molecule	Add new atom to partial molecule	binding probability and affinity	A2C	GAT	include 3D structure of both protein target and generated compounds for drug design.
[103]	PL	Intermediate compound	add new bond between an atom pair	+/- 1 whether generated product matches groundtruth	A2C	GAT	Uses GNN to represent reactant/reagent molecules, and DRL to find an optimal sequence of bond changes for product transformation

provide auxiliary information to the agent in order to further improve model performance. This also applies to multi-task situations where different tasks are correlated or structurally related.

GNNs encompasses various tuning parameters in their architecture. Thus, neural architecture search in them are very effective. DRL algorithms such as DQN are an appropriate choice for searching since the search methodology is generic and applicable across different architectures. In fact, any applications involving search operations in GNN such as adversarial attack can be tackled very effectively with DRL. The use of multimodal data has led to heterogeneous graph-structured data in various applications, including knowledge graphs and recommender systems. Conventional GNNs are not designed to handle heterogeneity. Therefore, it is recommended to employ customized GNNs like relational GCN [107], heterogeneous GAT [108] and HetSANN [109] for encoding. Fundamentally, all these works demand separate aggregation and combination functions (model parameters) for each node/link type, i.e., node attributes or link relationships, so that a powerful node representation can be learned.

V. CHALLENGES AND FUTURE RESEARCH OPPORTUNITIES

The articles surveyed in this work reveal the wide applicability and importance of fusing GNN and DRL (summarized in §IV). This section identifies the challenges that lie ahead for widespread adoption, and suggests future directions to unlock the full potential of a combined GNN-DRL framework.

A. Generalization across problems

Enhancing generalizability of DRL algorithms is one of the key research areas, especially since deep neural networks are notorious for overfitting to the training environment. Although few methodologies are proposed in the recent literature, it is still an open issue that is also applicable to graph-structured environments and data. One possible research direction would be in developing *graph meta reinforcement learning framework* where agents can quickly adapt to new tasks or environment with fewer samples [110]. Specifically, this can be achieved by providing context variables that vary with applications. For example, in case of CO, unseen problems can be smaller instances of the same problem, problem instances with different distributions, or even the ones from the other types of CO problems. Although certain generalization efforts can be seen recently, there is more to be done. Another way to enhance generalizability of DRL algorithms is by exposing agents to several graph environments that can be created with training graphs via graph augmentation techniques. One potential idea would be in leveraging generative adversarial networks (GAN) for graph augmentation by generating synthetic examples via perturbing input graphs in terms of adding/removing nodes and links or modifying node/link attributes. This enables DRL agent to adapt by learning invariant features across varied and noisy environments. We believe that although this task is challenging, it is extremely important and a promising direction for research in DRL.

B. Explainability of models

A substantial amount of explainable Artificial Intelligence (XAI) literature is emerging on feature relevance techniques to explain the predictions of a deep neural network (DNN) or explaining models that consume image source data. However, it is unclear as to how XAI techniques can aid in understanding models beyond classification tasks, such as DRL. An improved interpretability and explainability of DRL (XRL) models could help shed light on the underlying mechanisms in situations where it is essential to justify and explain the agent's behavior, which is still contemplated as a black box. The recent efforts on XRL are problem specific and cannot be generalized to real-world RL tasks [111].

The concepts based on representation learning, like hierarchical RL, self-attention and Hindsight Experience Replay have been highlighted as a few encouraging approaches to improve explainability in DRL models [111], and can be considered as future lines of research. Furthermore, DRL is also employed for explaining node/link predictions in GNN by identifying the most influential sub-graph [26]. The use of GNN to improve the prediction of DRL can also be investigated as a part of future research in this space.

C. Seamless transition from simulated to real environment

Most of the prevailing GNN-DRL methods are developed based on synthetic graph-structured datasets and simulated platforms. Real-life scenarios are far more intricate compared to simulated platforms although various synthetic graphs are being continuously developed to mimic real-world networks. Thus, DRL needs rigorous validation before being deployed confidently in real life applications. This validation is specifically more crucial for applications like connected autonomous vehicle, manufacturing process, etc., where safety is of utmost importance. There are some attempts to transfer the trained DRL agent from a simulator to an actual test bed [112] in general DRL, but there is a significant gap for graph-structured environments. Therefore, seamless transition from a simulated setting to real-world scenarios in a cautious, protected and productive approach is an important future research direction.

D. Tackling constrained problems

Optimization problems for real-life applications are mostly bounded by a variety of constraints in terms of finance, time, resources, etc. Most of the existing DRL work deal with constraints through penalties in rewards, which is appropriate if the constraints are soft, i.e., they can be violated at some cost. However, hard constraints must be strictly met and imposing a penalty cannot eliminate them, and thus is not a perfect approach. An alternative way to deal with hard constraints is by masking the constraints while designing the training environment, to keep the exploration space away from constraint violation, as considered in autonomous driving [113], [114]. These constraints become further complex when the underlying environment is graph-structured, such as a transportation network. Elegant approaches for handling hard constraints currently do not exist in literature. Therefore, further research is needed to explore the rich literature of constrained dynamical systems and other strategies for dealing with hard constraints.

E. Robustness against data/environment

From the perspective of practitioners, it is critical that solutions developed by GNN and DRL are robust to changes in data and the environment. Robustness signifies the sensitivity of predictions to the input. Primarily, this study is carried out for generality/transferability purposes or to safeguard models from adversarial attacks. In this regard, substantial work has been accomplished for standard DNN, but there have been few attempts in DRL, particularly for graph-structured environments. The sensitivity analysis can suggest suitable modifications in terms of environment design, model specification, training process, and data fidelity, among others. Therefore, there are a lot of miles to cover in this space before reaching a secured and robust framework. One possibility is to utilize GAN to improve DRL robustness, as they have been shown to be very effective in the supervised learning case.

F. Dynamic/heterogeneous graph environment

A majority of the existing GNN models perform prediction and inferencing over homogeneous graphs. However, a large number of real-world applications, like critical infrastructure networks, recommendation systems, and social networks, involve learning on heterogeneous graphs. Heterogeneous graph-structured data can represent numerous types of entities (nodes) and relations (edges) within a common graphical framework. It is difficult to handle such diverse graphs using existing GNN models. Consequently, developing new models and algorithms that are capable of learning with heterogeneous graphs would be highly beneficial in real-world systems like

cybersecurity [115], [116], text analysis [117], [118] and recommendation engines [119], [120]. Integrating the use of DRL techniques to achieve this goal can be considered as one of the possible future directions of research. Furthermore, the existing GNN methodologies assume graph-structured data to be static, i.e., the possibility of addition and/or removal of nodes/edges is disregarded. However, many practical applications like social networks encompass dynamic spatial relations that continuously evolve over time. Although spatial-temporal GNNs (STGNNs) possess the ability to partially handle dynamic graphs [121], further work is required to integrate a thorough understanding about executing downstream tasks like node classification, link prediction, community detection, and graph classification in dynamic graphs.

VI. SUMMARY AND CONCLUSIONS

This paper presents a systematic survey of literature focusing on works fusing DRL and GNN approaches. Although several reviews related to DRL have been published in recent years, most of these studies are limited to a particular application domain. This study, for the first time, presents a methodical review spanning diverse range of application domains. We have reviewed papers from the perspectives of both fundamental algorithmic enhancements as well as application-specific developments. From an algorithmic perspective, either a GNN is exploited to strengthen the formulation and improve performance of DRL, or DRL is employed to expand the applicability of GNN. Recent works blending the usage of both DRL and GNN across multiple applications (broadly classified into combinatorial optimization, transportation, manufacturing and control, knowledge graphs and life sciences) have been thoroughly investigated and discussed. We also highlight the key advantages of fusing DRL and GNN methodologies and outline the applicability of each of those components. Furthermore, we identify the challenges involved in effective integration of DRL and GNN, and propose some potential future research directions in this area.

REFERENCES

[1] "Mit technology review," https://www.technologyreview.com/10-breakthrough-technologies/, 2017.

[2] N. Mazyavkina, S. Sviridov, S. Ivanov, and E. Burnaev, "Reinforcement learning for combinatorial optimization: A survey," *Computers & Operations Research*, vol. 134, p. 105400, 2021.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[4] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *International conference on machine learning*. PMLR, 2016, pp. 49–58.

[5] J. Luketina, N. Nardelli, G. Farquhar, J. Foerster, J. Andreas, E. Grefenstette, S. Whiteson, and T. Rocktäschel, "A survey of reinforcement learning informed by natural language," *arXiv preprint arXiv:1906.03926*, 2019.

[6] A. Bernstein and E. Burnaev, "Reinforcement learning in computer vision," in *Tenth International Conference on Machine Vision (ICMV 2017)*, vol. 10696. SPIE, 2018, pp. 458–464.

[7] N. P. Farazi, B. Zou, T. Ahamed, and L. Barua, "Deep reinforcement learning in transportation research: A review," *Transportation research interdisciplinary perspectives*, vol. 11, p. 100425, 2021.

[8] S. K. Zhou, H. N. Le, K. Luu, H. V. Nguyen, and N. Ayache, "Deep reinforcement learning in medical imaging: A literature review," *Medical image analysis*, vol. 73, p. 102193, 2021.

[9] M. S. Frikha, S. M. Gammar, A. Lahmadi, and L. Andrey, "Reinforcement and deep reinforcement learning for wireless internet of things: A survey," *Computer Communications*, vol. 178, pp. 98–113, 2021.

[10] R. N. Boute, J. Gijsbrechts, W. van Jaarsveld, and N. Vanvuchelen, "Deep reinforcement learning for inventory control: A roadmap," *European Journal of Operational Research*, 2021.

[11] A. Perera and P. Kamalaruban, "Applications of reinforcement learning in energy systems," *Renewable and Sustainable Energy Reviews*, vol. 137, p. 110618, 2021.

[12] F. Obite, A. D. Usman, and E. Okafor, "An overview of deep reinforcement learning for spectrum sensing in cognitive radio networks," *Digital Signal Processing*, vol. 113, p. 103014, 2021.

[13] M. Glavic, "(deep) reinforcement learning for electric power system control and related problems: A short review and perspectives," *Annual Reviews in Control*, vol. 48, pp. 22–35, 2019.

[14] M. Botvinick, J. X. Wang, W. Dabney, K. J. Miller, and Z. Kurth-Nelson, "Deep reinforcement learning and its neuroscientific implications," *Neuron*, vol. 107, no. 4, pp. 603–616, 2020.

[15] A. Alomari, N. Idris, A. Q. M. Sabri, and I. Alsmadi, "Deep reinforcement and transfer learning for abstractive text summarization: A review," *Computer Speech & Language*, vol. 71, p. 101276, 2022.

[16] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017.

[17] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[18] I. Grondman, L. Busoniu, G. A. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1291–1307, 2012.

[19] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

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

[21] K. Madhawa and T. Murata, "Active learning for node classification: an evaluation," *Entropy*, vol. 22, no. 10, p. 1164, 2020.

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

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

[24] K. Zhou, Q. Song, X. Huang, and X. Hu, "Auto-gnn: Neural architecture search of graph neural networks," *arXiv preprint arXiv:1909.03184*, 2019.

[25] Y. Gao, H. Yang, P. Zhang, C. Zhou, and Y. Hu, "Graph neural architecture search." in *IJCAI*, vol. 20, 2020, pp. 1403–1409.

[26] C. Shan, Y. Shen, Y. Zhang, X. Li, and D. Li, "Reinforcement learning enhanced explainer for graph neural networks," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[27] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial attacks on neural networks for graph data," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2847–2856.

[28] X. Tang, Y. Li, Y. Sun, H. Yao, P. Mitra, and S. Wang, "Transferring robustness for graph neural network against poisoning attacks," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 600–608.

[29] H. Wu, C. Wang, Y. Tyshetskiy, A. Docherty, K. Lu, and L. Zhu, "Adversarial examples on graph data: Deep insights into attack and defense," *arXiv preprint arXiv:1903.01610*, 2019.

[30] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song, "Adversarial attack on graph structured data," in *International conference on machine learning*. PMLR, 2018, pp. 1115–1124.

[31] Y. Sun, S. Wang, X. Tang, T.-Y. Hsieh, and V. Honavar, "Non-target-specific node injection attacks on graph neural networks: A hierarchical reinforcement learning approach," in *Proc. WWW*, vol. 3, 2020.

[32] J. Jin and X. Ma, "Hierarchical multi-agent control of traffic lights based on collective learning," *Engineering applications of artificial intelligence*, vol. 68, pp. 236–248, 2018.

[33] H. Mao, Z. Gong, Z. Zhang, Z. Xiao, and Y. Ni, "Learning multi-agent communication under limited-bandwidth restriction for internet packet routing," *arXiv preprint arXiv:1903.05561*, 2019.

[34] H. Mao, Z. Zhang, Z. Xiao, and Z. Gong, "Modelling the dynamic joint policy of teammates with attention multi-agent ddpg," *arXiv preprint arXiv:1811.07029*, 2018.

[35] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," in *International Conference on Machine Learning.* PMLR, 2019, pp. 2961–2970.

[36] J. Su, S. Adams, and P. A. Beling, "Counterfactual multi-agent reinforcement learning with graph convolution communication," *arXiv preprint arXiv:2004.00470*, 2020.

[37] Y. Wang, D. Shi, C. Xue, H. Jiang, G. Wang, and P. Gong, "Ahac: Actor hierarchical attention critic for multi-agent reinforcement learning," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC).* IEEE, 2020, pp. 3013–3020.

[38] H. Mao, Z. Zhang, Z. Xiao, Z. Gong, and Y. Ni, "Learning multi-agent communication with double attentional deep reinforcement learning," *Autonomous Agents and Multi-Agent Systems*, vol. 34, no. 1, pp. 1–34, 2020.

[39] J. Jiang and Z. Lu, "Learning attentional communication for multi-agent cooperation," *Advances in neural information processing systems*, vol. 31, 2018.

[40] J. Jiang, C. Dun, T. Huang, and Z. Lu, "Graph convolutional reinforcement learning," *arXiv preprint arXiv:1810.09202*, 2018.

[41] Y. Liu, W. Wang, Y. Hu, J. Hao, X. Chen, and Y. Gao, "Multi-agent game abstraction via graph attention neural network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, 2020, pp. 7211–7218.

[42] W. Böhmer, V. Kurin, and S. Whiteson, "Deep coordination graphs," in *International Conference on Machine Learning.* PMLR, 2020, pp. 980–991.

[43] S. Shen, Y. Fu, H. Su, H. Pan, P. Qiao, Y. Dou, and C. Wang, "Graphcomm: A graph neural network based method for multi-agent reinforcement learning," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2021, pp. 3510–3514.

[44] X. Zhang, Y. Liu, X. Xu, Q. Huang, H. Mao, and A. Carie, "Structural relational inference actor-critic for multi-agent reinforcement learning," *Neurocomputing*, vol. 459, pp. 383–394, 2021.

[45] W. J. Yun, S. Yi, and J. Kim, "Multi-agent deep reinforcement learning using attentive graph neural architectures for real-time strategy games," in *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC).* IEEE, 2021, pp. 2967–2972.

[46] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner *et al.*, "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.

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

[48] W. Huang, I. Mordatch, and D. Pathak, "One policy to control them all: Shared modular policies for agent-agnostic control," in *International Conference on Machine Learning.* PMLR, 2020, pp. 4455–4464.

[49] S. Muggleton and L. De Raedt, "Inductive logic programming: Theory and methods," *The Journal of Logic Programming*, vol. 19, pp. 629–679, 1994.

[50] V. Zambaldi, D. Raposo, A. Santoro, V. Bapst, Y. Li, I. Babuschkin, K. Tuyls, D. Reichert, T. Lillicrap, E. Lockhart *et al.*, "Relational deep reinforcement learning," *arXiv preprint arXiv:1806.01830*, 2018.

[51] S. Sanner *et al.*, "Relational dynamic influence diagram language (rddl): Language description," *Unpublished ms. Australian National University*, vol. 32, p. 27, 2010.

[52] S. Garg, A. Bajpai *et al.*, "Symbolic network: generalized neural policies for relational mdps," in *International Conference on Machine Learning.* PMLR, 2020, pp. 3397–3407.

[53] J. Janisch, T. Pevný, and V. Lisý, "Symbolic relational deep reinforcement learning based on graph neural networks," *arXiv preprint arXiv:2009.12462*, 2020.

[54] S. Manchanda, A. Mittal, A. Dhawan, S. Medya, S. Ranu, and A. Singh, "Gcomb: Learning budget-constrained combinatorial algorithms over billion-sized graphs," *Advances in Neural Information Processing Systems*, vol. 33, pp. 20 000–20 011, 2020.

[55] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[56] 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.* PMLR, 2021, pp. 7565–7577.

[57] H. Wang, K. Wang, J. Yang, L. Shen, N. Sun, H.-S. Lee, and S. Han, "Gcn-rl circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning," in *2020 57th ACM/IEEE Design Automation Conference (DAC).* IEEE, 2020, pp. 1–6.

[58] K. Zhu, M. Liu, H. Chen, Z. Zhao, and D. Z. Pan, "Exploring logic optimizations with reinforcement learning and graph convolutional network," in *2020 ACM/IEEE 2nd Workshop on Machine Learning for CAD (MLCAD).* IEEE, 2020, pp. 145–150.

[59] V.-A. Darvariu, S. Hailes, and M. Musolesi, "Goal-directed graph construction using reinforcement learning," *Proceedings of the Royal Society A*, vol. 477, no. 2254, p. 20210168, 2021.

[60] H. Dai, B. Dai, and L. Song, "Discriminative embeddings of latent variable models for structured data," in *International conference on machine learning.* PMLR, 2016, pp. 2702–2711.

[61] S. Munikoti, L. Das, and B. Natarajan, "Scalable graph neural network-based framework for identifying critical nodes and links in complex networks," *Neurocomputing*, vol. 468, pp. 211–221, 2022.

[62] ——, "Bayesian graph neural network for fast identification of critical nodes in uncertain complex networks," in *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC).* IEEE, 2021, pp. 3245–3251.

[63] M. Deudon, P. Cournut, A. Lacoste, Y. Adulyasak, and L.-M. Rousseau, "Learning heuristics for the tsp by policy gradient," in *International conference on the integration of constraint programming, artificial intelligence, and operations research.* Springer, 2018, pp. 170–181.

[64] I. Drori, A. Kharkar, W. R. Sickinger, B. Kates, Q. Ma, S. Ge, E. Dolev, B. Dietrich, D. P. Williamson, and M. Udell, "Learning to solve combinatorial optimization problems on real-world graphs in linear time," in *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA).* IEEE, 2020, pp. 19–24.

[65] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," *arXiv preprint arXiv:1611.09940*, 2016.

[66] H. Lu, X. Zhang, and S. Yang, "A learning-based iterative method for solving vehicle routing problems," in *International conference on learning representations*, 2019.

[67] Y. Hu, Y. Yao, and W. S. Lee, "A reinforcement learning approach for optimizing multiple traveling salesman problems over graphs," *Knowledge-Based Systems*, vol. 204, p. 106244, 2020.

[68] S. Yang, B. Yang, H.-S. Wong, and Z. Kang, "Cooperative traffic signal control using multi-step return and off-policy asynchronous advantage actor-critic graph algorithm," *Knowledge-Based Systems*, vol. 183, p. 104855, 2019.

[69] 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.

[70] S. Yang, B. Yang, Z. Kang, and L. Deng, "Ihg-ma: Inductive heterogeneous graph multi-agent reinforcement learning for multi-intersection traffic signal control," *Neural networks*, vol. 139, pp. 265–277, 2021.

[71] J. Yoon, K. Ahn, J. Park, and H. Yeo, "Transferable traffic signal control: Reinforcement learning with graph centric state representation," *Transportation Research Part C: Emerging Technologies*, vol. 130, p. 103321, 2021.

[72] 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.

[73] Q. Xiao, C. Li, Y. Tang, and L. Li, "Meta-reinforcement learning of machining parameters for energy-efficient process control of flexible turning operations," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 1, pp. 5–18, 2019.

[74] J. Dornheim, N. Link, and P. Gumbsch, "Model-free adaptive optimal control of episodic fixed-horizon manufacturing processes using reinforcement learning," *International Journal of Control, Automation and Systems*, vol. 18, no. 6, pp. 1593–1604, 2020.

[75] J. Huang, Q. Chang, and J. Arinez, "Deep reinforcement learning based preventive maintenance policy for serial production lines," *Expert Systems with Applications*, vol. 160, p. 113701, 2020.

[76] J. Huang, J. Zhang, Q. Chang, and R. X. Gao, "Integrated process-system modelling and control through graph neural network and reinforcement learning," *CIRP Annals*, vol. 70, no. 1, pp. 377–380, 2021.

[77] J. Park, J. Chun, S. H. Kim, Y. Kim, and J. Park, "Learning to schedule job-shop problems: representation and policy learning using graph neural network and reinforcement learning," *International Journal of Production Research*, vol. 59, no. 11, pp. 3360–3377, 2021.

[78] T. Yamada, "Studies on metaheuristics for jobshop and flowshop scheduling problems," 2003.

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

[80] S. Chen, Y. Leng, and S. Labi, "A deep learning algorithm for simulating autonomous driving considering prior knowledge and temporal information," *Computer-Aided Civil and Infrastructure Engineering*, vol. 35, no. 4, pp. 305–321, 2020.

[81] D. M. Saxena, S. Bae, A. Nakhaei, K. Fujimura, and M. Likhachev, "Driving in dense traffic with model-free reinforcement learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 5385–5392.

[82] R. Du, S. Chen, Y. Li, J. Dong, P. Y. J. Ha, and S. Labi, "A cooperative control framework for cav lane change in a mixed traffic environment," *arXiv preprint arXiv:2010.05439*, 2020.

[83] Y. Wang, S. Hou, and X. Wang, "Reinforcement learning-based bird-view automated vehicle control to avoid crossing traffic," *Computer-Aided Civil and Infrastructure Engineering*, vol. 36, no. 7, pp. 890–901, 2021.

[84] 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.

[85] D. Baumann, J.-J. Zhu, G. Martius, and S. Trimpe, "Deep reinforcement learning for event-triggered control," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 943–950.

[86] V. Lima, M. Eisen, K. Gatsis, and A. Ribeiro, "Resource allocation in wireless control systems via deep policy gradient," in *2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE, 2020, pp. 1–5.

[87] ——, "Resource allocation in large-scale wireless control systems with graph neural networks," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 2634–2641, 2020.

[88] F. Gama, A. G. Marques, G. Leus, and A. Ribeiro, "Convolutional neural network architectures for signals supported on graphs," *IEEE Transactions on Signal Processing*, vol. 67, no. 4, pp. 1034–1049, 2019.

[89] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in neural information processing systems*, vol. 30, 2017.

[90] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[91] H. Wang, T. Qiu, Z. Liu, Z. Pu, and J. Yi, "Multi-agent formation control with obstacles avoidance under restricted communication through graph reinforcement learning," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 8150–8156, 2020.

[92] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3052–3059.

[93] X. Wang, D. Wang, C. Xu, X. He, Y. Cao, and T.-S. Chua, "Explainable reasoning over knowledge graphs for recommendation," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 5329–5336.

[94] Q. He, J. Yang, and B. Shi, "Constructing knowledge graph for social networks in a deep and holistic way," in *Companion Proceedings of the Web Conference 2020*, 2020, pp. 307–308.

[95] Y. Zhang, H. Dai, Z. Kozareva, A. J. Smola, and L. Song, "Variational reasoning for question answering with knowledge graph," in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[96] P. Zheng, L. Xia, C. Li, X. Li, and B. Liu, "Towards self-x cognitive manufacturing network: An industrial knowledge graph-based multi-agent reinforcement learning approach," *Journal of Manufacturing Systems*, vol. 61, pp. 16–26, 2021.

[97] H. Fei, Y. Ren, Y. Zhang, D. Ji, and X. Liang, "Enriching contextualized language model from knowledge graph for biomedical information extraction," *Briefings in bioinformatics*, vol. 22, no. 3, p. bbaa110, 2021.

[98] L. Nizzoli, M. Avvenuti, M. Tesconi, and S. Cresci, "Geo-semantic-parsing: Ai-powered geoparsing by traversing semantic knowledge graphs," *Decision Support Systems*, vol. 136, p. 113346, 2020.

[99] I. O. Mulang', K. Singh, C. Prabhu, A. Nadgeri, J. Hoffart, and J. Lehmann, "Evaluating the impact of knowledge graph context on entity disambiguation models," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 2157–2160.

[100] 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.

[101] A. D. McNaughton, C. Knutson, M. Bontha, J. A. Pope, and N. Kumar, "De novo design of protein target specific scaffold-based inhibitors via reinforcement learning," in *ICLR2022 Machine Learning for Drug Discovery*, 2022.

[102] C. Knutson, M. Bontha, J. A. Bilbrey, and N. Kumar, "Decoding the protein-ligand interactions using parallel graph neural networks," *arXiv preprint arXiv:2111.15144*, 2021.

[103] 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.

[104] X. Zhao, J. Wu, H. Peng, A. Beheshti, J. Monaghan, D. McAlpine, H. Hernandez-Perez, M. Dras, Q. Dai, Y. Li *et al.*, "Deep reinforcement learning guided graph neural networks for brain network analysis," *arXiv preprint arXiv:2203.10093*, 2022.

[105] J. You, R. Ying, and J. Leskovec, "Position-aware graph neural networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 7134–7143.

[106] J. You, J. Gomes-Selman, R. Ying, and J. Leskovec, "Identity-aware graph neural networks," *arXiv preprint arXiv:2101.10320*, 2021.

[107] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. v. d. Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European semantic web conference*. Springer, 2018, pp. 593–607.

[108] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *The world wide web conference*, 2019, pp. 2022–2032.

[109] H. Hong, H. Guo, Y. Lin, X. Yang, Z. Li, and J. Ye, "An attention-based graph neural network for heterogeneous structural learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 4132–4139.

[110] S. Munikoti, B. Natarajan, and M. Halappanavar, "Gramer: Graph meta reinforcement learning for multi-objective influence maximization," *arXiv preprint arXiv:2205.14834*, 2022.

[111] A. Heuillet, F. Couthouis, and N. Díaz-Rodríguez, "Explainability in deep reinforcement learning," *Knowledge-Based Systems*, vol. 214, p. 106685, 2021.

[112] B. Chalaki, L. E. Beaver, B. Remer, K. Jang, E. Vinitsky, A. M. Bayen, and A. A. Malikopoulos, "Zero-shot autonomous vehicle policy transfer: From simulation to real-world via adversarial learning," in *2020 IEEE 16th International Conference on Control & Automation (ICCA)*. IEEE, 2020, pp. 35–40.

[113] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takác, "Reinforcement learning for solving the vehicle routing problem," *Advances in neural information processing systems*, vol. 31, 2018.

[114] Y. Liu, A. Halev, and X. Liu, "Policy learning with constraints in model-free reinforcement learning: A survey," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, 2021.

[115] B. Hu, Z. Zhang, C. Shi, J. Zhou, X. Li, and Y. Qi, "Cash-out user detection based on attributed heterogeneous information network with a hierarchical attention mechanism," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 946–953.

[116] S. Hou, Y. Ye, Y. Song, and M. Abdulhayoglu, "Hindroid: An intelligent android malware detection system based on structured heterogeneous information network," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 1507–1515.

[117] H. Linmei, T. Yang, C. Shi, H. Ji, and X. Li, "Heterogeneous graph attention networks for semi-supervised short text classification," 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. 4821–4830.

[118] L. Hu, S. Xu, C. Li, C. Yang, C. Shi, N. Duan, X. Xie, and M. Zhou, "Graph neural news recommendation with unsupervised preference disentanglement," in *Proceedings of the 58th annual meeting of the association for computational linguistics*, 2020, pp. 4255–4264.

[119] C. Shi, B. Hu, W. X. Zhao, and S. Y. Philip, "Heterogeneous information network embedding for recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 2, pp. 357–370, 2018.

[120] B. Hu, C. Shi, W. X. Zhao, and P. S. Yu, "Leveraging meta-path based context for top-n recommendation with a neural co-attention model,"

in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1531–1540.

[121] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.