SAPIENZA
UNIVERSITÀ DI ROMA

Deep Learning applications over Heterogeneous Networks: from Multimedia to Genes

Dipartimento di Ingegneria Informatica Automatica e Gestionale ANTONIO RUBERTI,
SAPIENZA – Università di Roma

Dottorato di Ricerca in Ingegneria Informatica – XXXIV Ciclo

Candidate

Jesús Fernando Cevallos Moreno
ID number 1843057

Thesis Advisor

Prof. Massimo Mecella

Co-Advisor

Prof. Aminael Sánchez R.

A thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science Engineering

April 2022

Jesús Fernando Cevallos Moreno. *Deep Learning applications over Heterogeneous Networks: from Multimedia to Genes.*
Ph.D. thesis. Sapienza – University of Rome

VERSION: 2022, July the 19th
WEBSITE: http://diag.uniroma1.it/users/jesus-fernando_cevallos-moreno
EMAIL: cevallos@diag.uniroma1.it

To my Heavenly Father,
to my lovely father, Marco E. Cevallos,
to Cielito,
and to all the people who pursue the union with the Divine Persons.

Acknowledgments

The candidate sincerely thanks the guidance of professors Massimo Mecella and Aminael Sánchez. The author would also like to thank all the other people who have supported him in developing this research. Particular thanks go to his sisters and brothers of the Id Institute of Christ the Redeemer and his blood relatives. These have dedicated immense effort to permit the candidate to take this research journey. Special thanks are given to his superiors and brothers Rev. Fr. Jesús Fernández, and Rev. Fr. Vicente de la Fuente, his lovely mother, María Moreno and his brothers and sisters, Mabel, Salomé, and Marco.

Special gratitude goes to the noblest and brilliant professor Rev. Fr. Luis Casasús Latorre, who has charitably helped the candidate focus and achieve his objectives both from a technical and a motivational side. Passionate researchers like Rebecca Sattler, and Dr. Raúl Caulier have also enabled the candidate to go beyond his limitations within a precious religious and academic brotherhood. Great thanks to them also. The clever dedication and respectful friendship of highly productive researchers like Dr. Lorenzo R. Celsi and Dr. Peyman Zarrineh merit particular gratitude from the author.

The author acknowledges ELIS Innovation Hub for supporting this research. Inside EIH, the availability and interest of highly professional and generous people such as Luigi de Costanzo and Marco O. Migliori deserve special gratitude. The author would like to thank also his brothers, the priests from the community of Sant'Antonio Abate at Varese, special thanks to the Rev. Fr. Luigi Panighetti. Finally, many thanks go to the Happiness Project crew and the engineers Edison Sánchez, Dr. Leonardo De Laurentiis, Federico Di Domenicantonio, Valerio Paduano and Federico Kieffer for their valuable advices and friendship.

Extended Abstract

Research context

Networks are ubiquitous in nature and technology. Many research fields in industry and academia model the environments they study as networks, in that composite realities like the Internet, virtual realities, medical services, and particle physics phenomena, among others, can be seen as a group of simpler entities that interact between them. Moreover, networks are often heterogeneous because the interacting entities can be further differentiated into classes or groups, and the interactions themselves can be classified into different types. Some of the most prominent examples of heterogeneous graph models are those used in bioinformatics and bio-medicine. Many studies in these fields investigate multiple entities like molecules, proteins, and DNA segments that interact in numerous ways to drive biological processes.

Apart from bioinformatics, social networks and computer networks are also heterogeneous networks in which lots of research efforts have concentrated in the last years. The proliferation of data collection techniques and the democratization of social media production and consuming services have augmented the volume and heterogeneity of publicly available data on the Internet. One proof of this phenomenon is the non-structured database paradigm that has evolved in the last years. Non-structured databases are a building block of the recent big-data pipelines built to orchestrate multiple heterogeneous data sources and extract added value from them.

However, not only the modes of collecting and storing heterogeneous information networks have witnessed a continuous evolution. Graph analysis and network science are two disciplines that have evolved in the last years to cope with the need to extract valuable knowledge from heterogeneous relational data. Using heuristic-based and meta-heuristic-based solutions for solving computationally expensive problems is one of the most common practices adopted by network scientists that deal with big-data formatted as heterogeneous graphs (het-graph).

The rocket-fast speed at which the processor has evolved in the last decades has also democratized computing resources. Consequentially, the neural network paradigm, which was mainly forgotten by researchers in the last decade of the last century, has risen with unprecedented popularity: multiple gigabytes of training data are downloaded for free in minutes, and home computer processors are able to optimize millions of parameters through gradient descent-based algorithms to produce constant-time task-specific neural modules that show unprecedented accuracy in many particular inductive tasks.

This democratization of computing resources and extensive labeled data implied the democratization of deep learning. Consequentially, a question was born in the mind of researchers: how to foster synergies between deep learning and graph analysis tools? But the answer was not trivial: the main problem associated with the data collected from (heterogeneous) network environments was the non-euclidean format of the relationships between nodes of a graph. Fortunately, research efforts based on signal theory, spectral analysis, and spatial message passing frameworks solved the challenge of accommodating graphs to neural networks.

Traditional graph analysis instruments gained enormous scalability as a consequence. Moreover, this recent deep-learning based gain had immediate positive implications in the research agenda of the last years. Modern navigation, social recommender, and weather forecasting systems are just some ubiquitous proofs of this. The first contextual factor in which our research places itself is this

growth in the applicability of deep learning-based heterogeneous graph analysis to academical and industrial research fields.

The synergy between deep-learning and heterogeneous graphs is, however, a two-way relationship: deep learning brings scalability and generalization power to graph analysis algorithms, but also het-graph models help deep learning practitioners design more efficient learning pipelines. More specifically, solution spaces are often prohibitively large when optimization tasks are designed in complex industrial-scale environments. Developing algorithms that efficiently explore candidate solutions to find the global optima becomes non-trivial in these environments, even for deep-learning-based algorithms. In these cases, the injection of graph-model-specific inductive biases during the architectural and procedural design of deep-learning pipelines has proven crucial to tackle the curse of dimensionality. Thus, in the present research, we were not only interested in how deep learning has empowered modern graph analysis, but we also pretended to place our sight in the opposite direction of this synergy: how het-graph models facilitate the design of inductive biases in deep learning architectures.

The dissertation studies the state-of-the-art techniques that combine deep learning with heterogeneous graph-modeled scenarios. Two main paradigms of collaboration have been identified. The first one consists of enhancing the scalability and representation power of graph algorithms through deep learning. The second is the augmented efficiency of solution-space exploration that heterogeneous graph modeled scenarios induce in the design of deep learning optimization pipelines. Moreover, this research identified two open research opportunities where the studied synergisms could be helpful to solve. The first one is the online optimization of service function chain deployment in virtualized content delivery networks for live-streaming. The second is the inference of developmental regulatory mechanisms between genes and cis-regulatory elements. The candidate demonstrated his proficiency in the research field by applying the synergisms identified in the first phase of the research to solve such open problems.

This research in a nutshell.

This research aimed to investigate the synergies between deep learning and heterogeneous graph-based scenario modeling. The candidate has thoroughly studied the state-of-the-art (SOTA) techniques that combine deep learning to heterogeneous graph (het-graph) modeled scenarios. Two main paradigms of collaboration have been identified:

1. Deep learning enhances the scalability and the representation power of graph algorithms and shallow machine learning approaches for graph analysis.

2. Het-graph modeled scenarios help design solution-space exploration biases for deep learning-based optimization algorithms.

Moreover, the candidate has chosen two important research fields from industry and academia to identify two open problems where the studied synergisms could be helpful. These open problems were:

1. The online optimization of service function chain deployment in virtualized content delivery networks for live-streaming,

2. The inference of developmental regulatory mechanisms between genes and cis-regulatory elements.

Finally, the candidate demonstrated his proficiency in the research field by applying the synergisms identified in the first phase of the research to solve these open problems.

Research Objectives and contributions

We now give a more detailed description of the overall research process, formatting the exposition with a series of research goals and contributions.

Studying the state-of-the-art

Network models in nature and technology are receiving increasing attention in industrial and academic research fields. Thousands of publications use network models, and various hundreds of them apply deep learning-based techniques to reach the research goals. Thus, the first goal that the candidate aimed to fulfill can be formulated as follows:

Goal 1

Acquire mature knowledge of state-of-the-art techniques that create synergies between deep learning and heterogeneous graph-based scenario modeling.

In the context of this research, the heterogeneous graph model was the first formalism to investigate. The candidate studied the commonly used notations and concepts to proceed with the methodological analysis of deep learning-based instruments applied in het-graph modeled problems. Moreover, the candidate successfully researched how modern artificial intelligence has been applied to learn characteristics from heterogeneous networks and produce insights in analyzing this kind of environment. Transductive learning or static optimization algorithms, dubbed shallow-ML approaches, were reviewed as a propaedeutic study. The candidate then investigated how researchers are using deep learning-based techniques for two tasks:

- Overcoming the curse of dimensionality often induced by heterogeneous network environments.
- Creating inductive learning modules with higher generalization capabilities with respect to shallow-ML-based approaches.

Other research branch that evidences collaborations between het-graph modeled scenarios and deep learning pipelines was identified and the candidate sought to answer the question of how deep learning practitioners benefit from het-graph modeling when implementing efficient optimization pipelines in network-related environments. The fruit of this preliminary SOTA research was the conclusion that two main research trends exist by which the previous tasks are addressed by industry and academia:

1. Heterogeneous graph representation learning (het-graph-RL). This paradigm creates inductive solutions to embed graph data into simpler euclidean latent spaces minimizing information loss. Het-graph-RL reveals a critical building block of deep-learning graph-analysis pipelines.
2. Heterogeneous graph-model-driven design of deep reinforcement learning (DRL) algorithms. Researchers are exploiting het-graphs to inject exploration biases in DRL pipelines. These biases have revealed crucial for the convergence of optimization modules in many heterogeneous network environments.

In summarizing, the first contribution of this research is:

Contribution 1.1

The main synergisms between deep learning and het-graph modeling were studied and two main collaboration paradigms were found:

1. **Deep learning for het-graph models:** Enhancing the scalability, expressiveness and generalization capacity of traditional machine learning for network environments.
2. **Het-graph models for deep-learning:** Using het-graph models to create efficient exploration biases of prohibitively large solution spaces in deep learning-based optimization tasks.

Moreover, the candidate identified two main good practices of the research community that concretize these synergisms:

1. Using deep neural networks to create inductive representation algorithms that encode high-dimensional heterogeneous information into simpler euclidean spaces.
2. Using graph modelization to induce efficient exploration of prohibitively large action spaces by deep reinforcement learning agents.

The candidate created a systematic synthesis of the extensive review performed for these two research branches. Moreover, remarkable application examples were also studied. (Refer to chapter 1 of this thesis).

Identifying open challenges in concrete research domains

A more vertical penetration of the methodologies research that tackled the first objective was focused in mastering the good practices when designing applications for concrete network-based research fields. We can summarize this second objective as:

Goal 2

Understand how the combination of deep learning with heterogeneous graph scenario modeling has tackled research challenges in at least two concrete research areas and identify open challenges in these contexts.

Two research fields were identified that are evolving and promise to develop exponentially in the following years: the video delivery industry and bioinformatics. We performed extensive research on the common problems that the video delivery industry has recently managed to solve with the help of het-graph models and deep learning. We find that the QoS and Cost optimization of virtualized content delivery network systems was a particularly active research area in this industry. Thus a first contribution is:

Contribution 2.1

The candidate has systematically reviewed the good practices for designing proactive AI-assisted service function chain (SFC) deployment for video delivery systems. An open challenge has been identified: the multi-objective online optimization of virtualized live-video delivery systems. Refer to section 2.1 of this thesis for the extended exposition of this contribution.

Besides networking, bioinformatics is another emergent research field that deals with heterogeneous networks. One of the most common examples of this practice has developed alongside deep-learning-based high-throughput sequencing analysis. More specifically, this research confirmed to us that graph models have been extensively used to analyze multi-omics data sources. At the same time, we observed unprecedented growth in the availability of temporal datasets that contain

high-throughput sequencing data at various time points. Interestingly, developmental traces of gene expression and cis-regulatory elements' activity were available in these data. Thus, with his team's help, consensus, and supervision, the candidate chose to face the un-solved challenge of inferring transcriptional gene regulatory networks from these temporal datasets. We can state that the second contribution to our goal number 2 is the following:

Contribution 2.2

An extensive investigation on deep learning-based tools for high-throughput sequencing analysis has been conducted. In particular, standard transcriptional gene regulatory network inference methodologies have been studied and categorized. Moreover, the candidate, with the guidance of his supervisor and co-supervisor, has identified a research opportunity considering the recent growth in the availability of temporal multi-omics datasets to shed light on regulatory mechanisms through deep learning based-inference on heterogeneous genetic networks in the field of developmental studies. Section 2.2 contains a detailed exposition of these results.

Putting in practice the acquired knowledge

Once that concrete open challenges have been individuated, the task we pursue is solving them. The good methodological practices learned in the research path bring ideas and instruments to approach such problems. However, the main contribution of the candidate was to go beyond. More specifically, we can declare the following research goal:

Goal 3

Demonstrate the maturity of the acquired knowledge by implementing solutions to the open challenges identified in the previous research phase:

1. Create a constant-time solution to intelligent online SFC deployment for live-streaming vCDNs.
2. Create an inductive instrument to shed light on developmental transcriptional regulatory mechanisms between cis-regulatory elements and genes.

The state-of-the-art techniques studied offered many hints to designing proper approaches for these open problems. Specifically, the candidate was allowed to use het-graph models and deep learning-based instruments to solve these problems. However, novel ideas on how to combine these approaches were necessary to design the proper solutions.

Service function chain deployment in live-streaming scenarios requires more efficient state-space representations than those presented by SOTA techniques developed for video-on-demand and general virtualized network functions. Apart from using the well-established Deep Reinforcement Learning framework to solve this network-related problem, we exploited the het-graph characteristics of our model to design a proper action space serialization, lightweight state-space representation, an attention mechanism over actions, and a suitable reward shaping. More specifically, we took into account the following facts in the design of the DRL algorithm:

- Different request characteristics might require different importance distributions among the nodes' features.
- Different relations among nodes like content ingestion and content streaming are important to take into account when deciding where to place the virtual network functions.

More importantly, we enabled the efficiency of the state-space exploration by designing a dense reward policy based on the live streaming SFC meta-path abstraction and the QoS constraints.

Finally, our experiment's results were enhanced by including techniques like experience replay, target networks, and action advantage learning. In summarizing, the first contribution to goal three can be expressed as follows:

Contribution 3.1

The candidate exploited the acquired knowledge to engineer the first solution for online multi-objective optimization of SFC deployment in live-streaming virtualized content delivery networks. With the supervision and contribution of team members, the candidate created efficient exploration biases and reward shaping based on the het-graph modelization of vCDN to achieve the convergence of a DRL-based agent to near-to-optimal SFC deployment policies. We refer the reader to chapter 3 for a thorough exposition of the proposed solution.

Remarkably, this contribution was the subject of an original research journal article [33]:

J. F. Cevallos M., R. Sattler, R.P. Caulier , L.R. Celsi, A. Sánchez R., and M. Mecella. 2021. *"Online Service Function Chain Deployment for Live-Streaming in Virtualized Content Delivery Networks: A Deep Reinforcement Learning Approach."* Future Internet 13 (11): 278.

The second challenge addressed by the candidate, with the supervision and collaboration of the research team, was focused on developmental studies. In particular, we notice the substantial lack of developmental studies that model the interaction of cis-regulatory elements (CREs) and genes as a heterogeneous network for inference of transcriptional regulatory mechanisms in the developmental bioinformatics literature. On the other hand, we noticed that a substantial part of the recently gained understanding of more general transcriptional regulatory mechanisms had used het-graph models and het-graph-RL techniques. Thus, we saw two main contextual factors that represented a research opportunity:

- Almost the totality of the recent literature on gene expression regulatory networks inference was based on supervised-learning approaches, while the novel CRE-gene temporal-omics datasets were substantially unlabeled.
- Transcriptional regulatory mechanisms between CREs and genes were mainly modeled as bipartite graphs. In these models, the set of genes and CREs composed the parts of the bipartite graph, and most of the recent literature focused on learning to generalize the pre-defined anchoring of elements between these parts.

With these preliminaries, we observed that even if het-graph models and deep learning were useful pathways to follow to reach our goal, the SOTA techniques needed additional instruments to overcome the difficulties posed by the particular problem faced. To synthesize our contributions, we can say that the proposed solution is innovative and efficient mainly because of three reasons:

1. We modeled the CRE and gene network as a heterogeneous graph containing various types of relationships among these elements, this modeling permitted us to introduce manifold learning techniques to regularize the euclidean distances between feature vectors and induce sparse-graph structures.
2. Learning the importance distribution of features relies on downstream task-specific feedback, i.e., supervised learning. We translated domain-specific knowledge into proper algorithmic rules to design an efficient het-graph-RL without the need for labeled training data (i.e., our solution learns to identify plausible regulatory mechanisms in a completely unsupervised fashion).

3. Given that the inference of regulatory mechanisms was made through unsupervised clustering (which is equivalent to meta-graph identification in het-graphs), we introduced proper regularization techniques to limit the manipulation of the global geometry of the original feature space.

We can succinctly express the second contribution to goal number three as follows:

Contribution 3.2

With the help and supervision of his team, the candidate designed and developed a novel algorithm for the identification of regulatory mechanisms between cis-regulatory elements and genes in the developmental stages of tissues. Our contribution was the first to co-cluster temporal gene expression profiles and temporal CRE activity markers to the best of the authors' knowledge. Our clusters were formed taking into account correlations from three points of view:

1. Correlations between the expression profile of genes,
2. Correlations between the activity profile of CREs,
3. Correlations between the gene expression profiles and CREs' activity profiles.

The heterogeneous network modeled was fed to a novel deep learning-based het-graph-RL algorithm based on manifold learning techniques. Moreover, we incorporated proper regularization mechanisms to limit the manipulation of the embedding space geometry. Lastly, we designed model-based feature combination rules to converge, in an unsupervised fashion, to clusters that were validated as significant by domain experts' criteria.

Remarkably, this contribution was the subject of an original research journal article [34]:

J. F. Cevallos M., P. Zarrineh, A. Sánchez R., and M. Mecella. 2022. "DeepReGraph Co-Clusters Temporal Gene Expression and Cis-Regulatory Elements through Heterogeneous Graph Representation Learning." F1000Research 11: 518

Side research activities and contributions

It is worth mentioning that this research has been financially supported by ELIS Innovation Hub (EIH), which is a non-profit company that also supported a series of joint-research projects (JRP) to create valuable synergies between academia and industry. In the context of recent JRP program editions, which took place contemporary to this research period, the candidate had the opportunity to deepen the SOTA artificial intelligence-based solutions from the industry and practice the acquired knowledge. We now enlist some remarkable works that contributed positively to the development of the candidate's technical knowledge and non-technical attitudes as a researcher:

- The **Smart MBI-L project**, supported by a noted energy company, aimed at developing novel solutions to the synchronization problem in a distributed ecosystem where mobile devices were used to monitor the health status of the span nodes of the Italian national high-voltage electric distribution network. The candidate contributed to the design and development of the correspondent solution while learning to differentiate among heterogeneous characteristics of these network elements and the main connections between them.
- The **People Analytics project**, was supported by the HR department of another worldwide energy company. This project aimed to exploit the proprietary unstructured short-text corpora

made of electronic employee profiles to extract value-added recommendation systems for HR decision-making-support systems. The candidate was responsible for leading the design and development of this solution while learning the theoretical connections between deep learning-based natural language processing techniques for word embedding creation and knowledge graphs.

- The **RPI-Prediction project** was supported by a transport infrastructure company. This project's objective was to create a deep-learning-based instrument to predict the temporal profile of **R**oad **P**erformance **I**ndicators as a function of multi-sensor networks like weather sensing stations and traffic sensors on the road. The candidate led the design and development of this solution while learning from domain experts the importance of network modelization in this field.
- Finally, the candidate also participated in the **CDN project**, a collaboration between EIH and a telecommunications company to assess the R&D opportunities in optimizing a media distribution service. The candidate designed and implemented a big-data stream-processing pipeline for the proprietary content delivery network of the telecommunications company. In doing this, the candidate learned the correlations among the heterogeneous characteristics of the media delivery requests and the powerful insights that deep-learning-based analysis instruments could give when applied to a heterogeneous graph model of a CDN.

Thesis outline

This thesis is structured as follows:

Chapter 1 introduces the concept of heterogeneous networks and heterogeneous graphs and then overviews SOTA deep-learning-based techniques applied to heterogeneous network models. These techniques were grouped into two main groups: heterogeneous graph representation learning (het-graphs-RL) and graph exploration biases in DRL.

Chapter 2 instead depicts two specialized applications of deep-learning-based solutions on het-graph modeled problems in the video delivery industry and bioinformatics field. The optimization of SFC Deployment on live-streaming scenarios and the co-clustering of temporal gene expression and CRE activity are identified as open research challenges in these fields.

Chapter 3 brings a thorough exposition of the solution to the SFC deployment optimization challenge proposed in the context of this research. The het-graph modelization, the action-state design, and the MDP model's reward shaping are explained in detail. The experiment description and the results are exposed and discussed at the end of this chapter.

Chapter 4 contains a detailed description of the developmental regulation inference instrument developed to respond to the second open challenge. A complete description of the het-graph model created and the het-graph-RL algorithm designed is given. The conducted experiments, the obtained results, and a discussion of the latter are presented at the end of this chapter.

Finally, chapter 5 depicts some concluding remarks on this job. A brief overview of the presented work is given alongside the exposition of novel application fields in which our proposed solutions could be helpful. Moreover, further research directions could empower the intuitions and instruments we have presented.

Abbreviations

The following abbreviations are used in this thesis:

ANN	Aritifical Neural Network
AP	Activator protein 1 Motif
BHLH	Basic Helix-Loop-Helix Motif
CDN	Content Delivery Network
CP	Content Provider
CRE	cis-Regulatory Elements
cCRE	candidate CRE
DT	Data-Transportation
DDPG	Deep Deterministic Policy Gradient
DNA	Deoxyribonucleic acid
DRL	Deep Reinforcement Learning
GAE	Graph Auto-encoder
GCN	Graph Convolutional Network
GCRL	Graph Convolutional Reinforcement Learning
GERM	Gene Expression Regulatory Mechanism
GNN	Graph Neural Network
GP-LLC	Greedy Policy of Lowest Latency and Lowest Cost algorithm
Graph-RL	Graph Representation Learning
GRL	Graph Reinforcement Learning
GRN	Transcriptional Gene Regulatory Network
GSEA	Gene set enrichment Analysis
het-graph	Heterogeneous graph
het-graph-RL	Heterogeneous graph representation learning
HPO	Hyper-parameter Optimization
ILP	Integer Linear Programming
ISP	Internet Service Provider
QoE	Quality of Experience
QoS	Quality of Service
P2P	Peer-to-Peer
MANO	Management and orchestration framework
MC	Markov chain
MDP	Markov decision process
MVNO	Mobile Virtual Network Operator
MEA	Motif Enrichment Analysis
MEF2	Myocyte enhancer factor 2 Motif
MEF2C	Myocyte enhancer factor 2C Motif
NFV	Network Function Virtualization
NFVI	Network Function Virtualization Infrastructure
mRNA	messenger RNA
NGS	Next-Generation Sequencing
OTT	Overt-The-Top Content
PPO	Proximal Policy Optimization
R-GCN	Relational Graph Convolutional Network
RL	Reinforcement Learning
RNA	Ribonucleic Acid
RNAP	RNA Polymerase Enzime
RNA-seq	RNA sequencing
RTT	Round-Trip-Time

SDN	Software Defined Networking
SFC	Service Function Chain
SOTA	State-of-the-art
TD-learning	Temporal difference-learning
TF	Transcription Factor
vCDN	virtualized-Content Delivery Network
VNF	Virtual Network Function
VNF-FGE	Virtual Network Function Forwarding Graph-Embedding
VNI	Virtualized Network Infrastructure
VNO	Virtual Network Orchestrator

Chapter 1

Introduction

Many industries like manufacturing [212], transportation [128, 116], networking [220], medicine [266], Decision-Making [212], Supply Chain Management[116] , Industry 4.0 [264], among others, use graph models to solve complex problems. Graph theory and network analysis have probably grown in the last years partly because of this usefulness [235, 127]. Moreover, graph models in industry and academia are often *heterogeneous* because they are composed of multiple types of nodes and edges. This dissertation focuses on the industrial and academic research efforts that create synergisms between heterogeneous graph modeled scenarios and deep learning techniques. To well-specify the scope of this research, this introductory chapter begins by providing an overview of heterogeneous graph models. It then reviews the main aspects by which deep learning has helped overcome the curse of dimensionality in some het-graph modeled problems and finally, it reviews how het-graph scenario modeling has helped design efficient exploration of high-dimensional solution spaces in deep learning-based optimization tasks.

1.1 What are Heterogeneous Networks?

Many industrial and academic research fields study networks, i.e., environments constituted by multiple entities with measurable relations between these elements. Graphs are the abstractions that provide the means to model network-based environment modeling. Nowadays graph modeling is widely used in Bioinformatics [300], Advertisement [297], Networking [261], Energy [45], Cybersecurity [43], among other industrial and academical research fields. A graph can be formally defined by a tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, ..., v_{|\mathcal{V}|}\}$ is the set of vertices or nodes and $\mathcal{E} = \{e_{i,j}\}_{i,j=1}^{|\mathcal{V}|}$ is the set of edges in the graph. An edge between the nodes v_i and v_j can be denoted by $(i, j) \in \mathcal{E}$.

Graphs are commonly represented through an *adjacency matrix* denoted by $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, where $A[i, j] = 1$ if $(i, j) \in \mathcal{E}$ and 0 otherwise. In *weighted graphs*, however, such an edge is not only a binary but a quantifiable relation, e.g. *similarity* between nodes or the inverse of any measure of *distance* between them. In such a case, the elements of the adjacency matrix will contain real numbers representing the weight of the links between nodes.

Moreover, many graphs include node-level features or attributes. One can represent these features using a real-valued matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}|\times m}$ where m is the dimension of the node-level feature space. Notice that every node in a graph might contain more than one feature vector. In that case, multiple feature matrices need to be used. Finally, the *neighborhood* of v, usually denoted with $\mathcal{N}(v)$, corresponds to the set of nodes that are connected to v by the edges of the graph. One can extend the notion of node neighborhood to include larger sets of nodes when considering the edges connecting the neighbors of a node, the neighbors of the neighbors, etc. We will speak of first, second, and n-order neighborhoods in such a case.[90]

Graphs or Networks?

Academia and Industry commonly use the term *network* to refer to real-world environments constituted by multiple similar elements that somehow interact between them -e.g., *social network*s- while the term *graph* is often used when referring to the related **model** or **data**[91]. In this dissertation, we use this distinction also.

We define a *graph model* as the model that represents a given *network* environment, i.e., an environment conceived as the composition of multiple similar elements and includes relations between those elements [25]. Unsurprisingly, heterogeneous graphs are also referred to as *heterogeneous information networks*.

A graph can be *homogeneous* if it has a unique type of nodes and a unique type of relationships defined between them, or *heterogeneous* if it introduces multiple edge and node types. Heterogeneous graphs can be denoted as the triplet $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{T})$, where the set of node and edge types, $\mathcal{T} = \{\mathcal{T}_\mathcal{E} \cup \mathcal{T}_\mathcal{V}\}$, has been included. If $\mathcal{T}_\mathcal{E}$ is the set of edge types and $\mathcal{T}_\mathcal{V}$ is the set of node types, then we have that, for a heterogeneous graph, $|\mathcal{T}_\mathcal{E}| + |\mathcal{T}_\mathcal{V}| > 2$. Note that nodes will have multiple neighborhoods in heterogeneous graphs, each one derived from a specific relation type and that each relation type adds specific *semantic information* to the graph. Note that the term *multidimensional network*, *multi-layer network*, or *multiplex network* is used in network theory context to refer to networks that are represented by graphs with different types of edges. [223, 44] In other words, a multi-layer network is a network that can be modeled by a graph where $|\mathcal{T}_\mathcal{E}| > 1$, without necessarily having $|\mathcal{T}_\mathcal{V}| > 1$.

In computer networking, the term *heterogeneous network* is used when referring to computer networks in which a wide variety of nodes exist according to their characteristics. The abbreviation *HetNet*, instead, is commonly used to refer to wireless networks in which various types of access points are available. This dissertation uses the term heterogeneous network in a more general context to refer to any network-like reality where multiple types of nodes and relations exist. In contrast, the term heterogeneous graph refers to the data model representing such an environment. Heterogeneous-Graph is abbreviated with het-graph.

Multiple environments in nature can be modeled as heterogeneous graphs. In other words, multiple networks in nature are heterogeneous in the types of nodes and relationships between nodes. A transportation network, for example, is constituted by various types of stations and routes between such stations where not every type of route connects stations of every type. For example, airways can connect only airports, while train lines and roads will not connect stations from different continents. Another example is the case of social networks, where various types of nodes exist: individuals, events, fan pages, institutions, and various connections among these elements exist: friendship, likes, posting, etc. Telecommunication networks are also constituted by multiple nodes like terminals, radio stations, high-frequency antennas, and various types of connections: last-mile connections, wireless fixed access connections, mobile radio links, satellite connections, etc.

Another network that can be modeled as an het-graph is the one that corresponds to the film industry. Nodes are actors, directors, and movies. Movies have various features like genre, year, company, rating, etc. Links or edges in this het-graph are easily defined. Some examples of the information that graph analysis could help extract are the mean ratings of actors by genre, the mean rating of companies by genre, the cluster's actors where the films in with they co-participate have similar ratings, etc. Some streaming companies, for example, have produced this kind of information from their proprietary big-data modeled as het-graph to create successful strategies to enter the film production market.[243, 239] More examples of heterogeneous networks are bibliographic networks, gene-disease association networks, and trading networks, among others.

Heterogeneous Networks and *Heterogeneous* Graphs.

In contrast to *homogeneous* graphs, where a unique type of nodes and edges exist, *heterogeneous* graphs introduce multiple edge and node types. If $\mathcal{T}_\mathcal{E}$ is the set of edge types and $\mathcal{T}_\mathcal{V}$ is the set of node types, then we have that, for a heterogeneous graph, $|\mathcal{T}_\mathcal{E}| + |\mathcal{T}_\mathcal{V}| > 2$. A heterogeneous graph will then be the model of a heterogeneous network environment.

Multidimensional networks, *multi-layer networks*, or *multiplex networks* are terms used in network theory to refer to networks that are modeled by graphs with different types of edges, without necessarily having different types of nodes. In other words, in a multi-layer graph we have that $|\mathcal{T}_\mathcal{E}| > 1$, and $|\mathcal{T}_\mathcal{V}| \geq 1$. Notice that heterogeneous graphs are a special case of multi-layer graphs.

We can extend the concept of neighborhood also to heterogeneous graphs: Giving a node $v \in \mathcal{V}$, and a specific edge-type $\tau \in \mathcal{T}_\mathcal{E}$, the neighborhood of v in τ is denoted as $\mathcal{N}_\tau(v)$ and corresponds to the set of nodes that are connected to v by τ-type edges. We also use $\mathcal{N}(v)$ to denote the set of nodes contained in every neighborhood of v:

$$\mathcal{N}(v) = \{\mathcal{N}_\tau(v), \forall \tau \in \mathcal{T}_\mathcal{E}\}$$

where, for practical reasons, we use the same notation for the neighborhood of a node in a homogeneous graph and the set of nodes in every neighborhood of a node in heterogeneous graphs.

In the case of heterogeneous graphs, edges are denoted not only by the nodes that are connected through it, but also by the type of relation τ, for example, $(i, \tau, j) \in \mathcal{E}$. Notice that if a graph that contains node-level features is heterogeneous, then each node type, $t \in \mathcal{T}_\mathcal{V}$, may have its own feature space. One can represent each node feature set by a different feature matrix $\mathbf{X_t} \in \mathbb{R}^{|\mathcal{V}_t| \times m_t}$, $\forall t \in \mathcal{T}_\mathcal{V}$, where m_t, is the dimension of the type-t node feature vector. Finally, note that one can represent a heterogeneous graph by multiple adjacency matrices, each matrix containing data for one type of relation between nodes: $\mathbf{A}_\tau \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$, $\forall \tau \in \mathcal{T}_\mathcal{E}$

Meta descriptors for heterogeneous networks

When considering het-graphs, we need to introduce some concepts that help schematize the information on a graph and help understand various ML algorithms designed to analyze het-graphs. We will now introduce such concepts with the help of a sample environment that can be straightforwardly modeled as a heterogeneous graph modeled environment: citation networks.

If we think of citation networks, node classes could be authors, papers, and institutions. Instead, links between nodes could be citations that connect two papers, authorships, defined between authors and papers, and co-authorships, which connect authors. Finally, affiliations could also be modeled as relations between authors and institutions. With proper graph analysis tools, multiple statistics could be extracted from a citation network. For example, author node centrality could help identify prominent authors, and paper clusters with high intra-citation density could help to identify various investigation arguments, etc. This information could be helpful whenever a new investigation needs to be carried out when venues are organized, etc. [144, 95]

Figure 1.1 contains other examples of relations that can be defined on an het-graph model of a citation network. This figure presents some little graphs where node types with circles and edge types with connections between these circles. This kind of abstraction over the instances of nodes can be seen as the graph's meta-data and is itself formatted as a graph. As a consequence, we can refer to the graphs presented in Figure 1.1 as meta-descriptors. These meta-descriptors are classified into three main classes, and these classes constitute the building blocks of a framework that helps to create complete meta-descriptions for heterogeneous graphs:

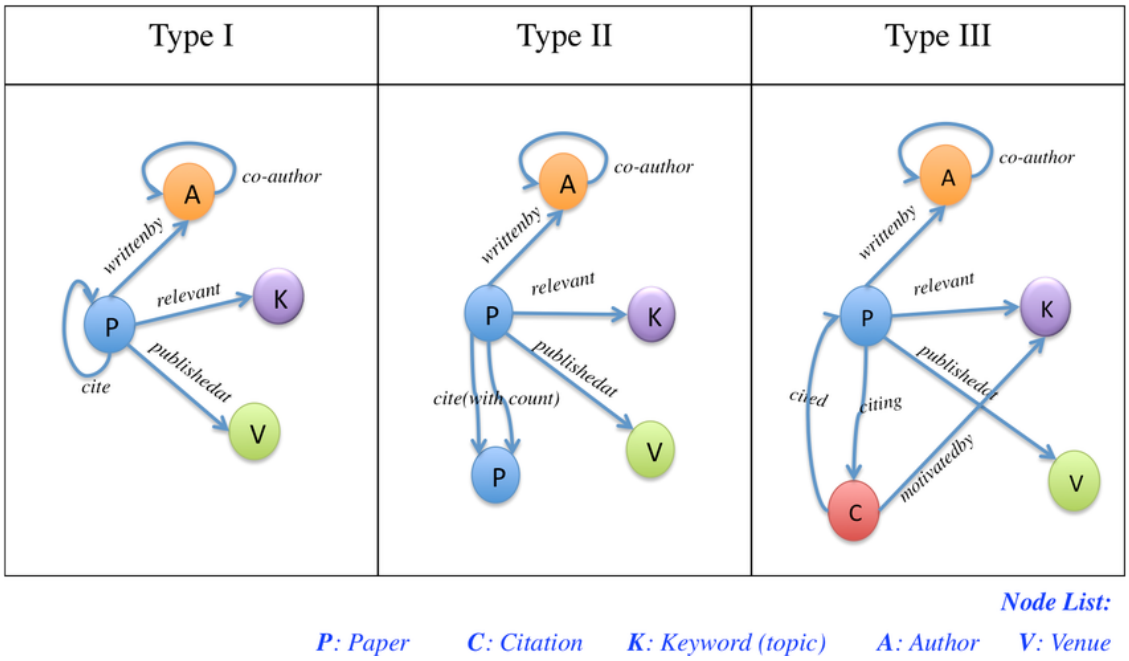

Figure 1.1. Citation networks can be modeled as heterogeneous graphs. Citation networks, in fact, may include multiple types of nodes (author, paper, venue, keyword, for example) and relationships among nodes or edges (authorship, co-autorship, relevancy, publication, among others). This figure represents three possible designs for het-graphs of citation networks in the form of meta-graphs with increasing heterogeneity. The figure is obtained from [144].

- **Network-schema:** The network schema is a meta-level graph or meta-data graph that gives us an overview of the relations between diverse types of nodes and edges in a het-graph. Specifically, if $\mathcal{G}$ is a het-graph, the network schema of $\mathcal{G}$ is another graph $\mathcal{M}_\mathcal{G}$ where each type of node in $\mathcal{G}$ is mapped to a node in $\mathcal{M}_\mathcal{G}$, and every type of edge in $\mathcal{G}$ is mapped to an edge in $\mathcal{M}_\mathcal{G}$. Consequentially, $\mathcal{M}_\mathcal{G}$ will be directed graph.

- **Meta-path**: The meta-path is a more fine-grain entity that gives us information about a specific relationship pattern between multiple edges and nodes in the graph. These patterns are said to encode semantic information because they encode a particular domain-specific meaning. Meta-paths are denoted by a sequence of node-types connected by edge-types:

$$m = N_0 \xrightarrow{E_0} N_1 \xrightarrow{E_1} N_2 ... \xrightarrow{E_l} N_{l+1}$$

  where N_i denotes the i-type node, and E_j denotes the j-type edge. Notice that a specific node-type could occur more than once in a meta-path. For example, in panel I of Figure 1.1, we have the meta-path

$$m_{co-authorship} = A \xrightarrow{co-author} A$$

  representing the co-authorship relation between multiple authors. Notice that the unique node-type in this relation is author nodes (symbolized by the capital letter A) hand the unique edge type is the "co-author" edge.

- **Meta-graph:** We can visualize a larger semantic unit if we capture the interaction between multiple meta-paths. This is the purpose of the meta-graph abstraction. In other words, meta-graphs are graphs made by joining various meta-paths that share at least one node. If we refer to Figure 1.1, each one of the three panels represents a specific meta-graph for a citation network. Notice that a meta-graph may contain multiple nodes referring to the same node type, as in meta-paths.

Lastly, many heterogeneous graphs define constraints over node types and edge types. For example, in the citation graph example, the *publishedat* relationship type can only matter a paper and a venue node. The *co-author* relationship instead can only be defined between two authors.

1.2 Synergisms between deep learning and heterogeneous network modeled environments

In the previous section, we have briefly introduced the concept of heterogeneous networks and het-graph models. We have given some examples of how this modeling technique may be used in network-related contexts.

This section is focused in describing the synergies between deep learning and heterogeneous graph-based scenario modeling instead. The term "Deep learning" is commonly referred to the usage of Artificial Neural Networks with multiple hidden layers. Deep Learning is being used in many industrial fields such as Electrical Utility [169], automotive [146], construction [5], finance [241], Tourism [181], among others. We refer the reader to [130] for a comprehensive reading of this discipline and its applications.

Two main paradigms of collaboration between deep learning and het-graph environment modeling were found in this research:

1. Deep learning enhances the scalability, expressive power and the applicability domain of graph algorithms and shallow machine learning approaches for graph analysis.

2. Het-graph modeled scenarios help design solution-space exploration biases for deep learning-based optimization algorithms.

In the rest of this section we will describe in detail each one of these paradigms.

1.2.1 Deep Learning helps solving Het-graph modeled problems

The reason for the growth in the usage of graph models is that graph theory enables efficient ways of extracting added value from graph-formatted data [16]. However, when these data are high-dimensional, a set of phenomena grouped under the name of "curse of dimensionality" can lead to the impossibility of finding practical solutions within reasonable time and using reasonably limited computing resources. The reason for this exponential growth in algorithmic complexity is mainly the exponential growth in data sparsity as a function the number of dimensions of data. For this reason, in the last years, deep neural networks [131] have helped to overcome the curse of dimensionality for graph-formatted data in various ways [320]. This section will briefly describe how deep learning is helping graph analysis extract added value from high-dimensional heterogeneous graph-formatted data.

Deep learning has permitted the application of graph analysis tools to high-dimensional network-based environments.[52] Deep learning helps to find solutions for heterogeneous network-based problems like path-finding, network classification, graph generation, node clustering, among others, mainly through deep heterogeneous graph representation learning (Graph RL). In this section we will give a state-of-the-art of Deep Heterogeneous Graph RL. To better explain deep-learning-based techniques for representation learning of het-grahs, we firstly introduce the concept of graph representation learning (graph-RL) and why it is useful. We then overview how deep learning has enhanced graph-RL permitting it to scale to high-dimensional contexts and overview recent literature on deep het-graph-RL alongside recent remarkable applications.

What is Graph Representation Learning?

Traditional machine learning approaches for graphs extract node level statistics [182] or exploit graph spectral analysis [180] for node classification, link prediction, and clustering tasks. On the

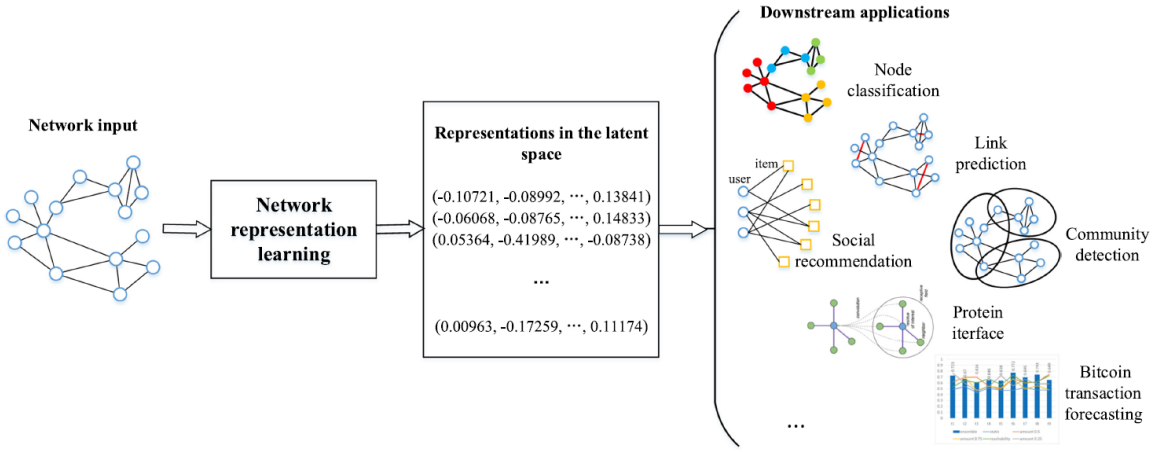

Figure 1.2. Graph or network representation learning aims to optimize node representations where topological information of the underlying graph are reflected. The use of the obtained representation has proven useful in many application fields like those in the right hand side of this figure. This figure was obtained from [327]

other hand, deep ANNs are well known by the function approximation capacity they have and the capacity of learning abstract and complex features from high-dimensional input. However, it is often impossible to feed high-scale graph-formatted data to deep ANN modules and extract added-value from these data if it is not encoded with a proper representation. To this end, Graph representation learning (Graph-RL) techniques are those that seek to embed the information of a graph on a reduced-dimension latent space or manifold minimizing the information loss. Note that the information encoded in a graph can be divided in topological and contextual. Topological information encodes the relations between nodes in the graph and contextual information encodes the information carried out by the nodes. These information sets can be rich and complex in graphs. For example, nodes and relations can be characterized by features, and relations induce various level of neighboring proximities between elements. Graph RL produces a latent representation of a graph where the topological and contextual characteristics in this latent space resemble the ones in the original graph.

Graph-RL has been widely studied by the machine learning research community in the last years.[291, 30, 80, 90, 236, 36, 301, 122] These techniques help to exploit the potentialities of Deep ANN to graph-like reasoning tasks. The main idea and the fruitful usefulness of Graph RL is summarized in figure 1.2. Graph representation learning has also been referred to as *Graph Embedding* in the research literature. One can say that graph embedding consists in finding a reduced dimensional representation of the nodes of a graph while preserving most of the semantic information of $\mathcal{G}$.[287, 91] In other words, in graph RL, we aim to find a reduced real-valued matrix: $\mathbf{Z} \in \mathbb{R}^{|\mathcal{V}| \times d}$, where d is the dimension of the new node-feature space and $d \ll m$, such that the information loss with respect to $\mathcal{G}$ is minimized and the existent relations (edges) between nodes in $\mathcal{G}$ can be inferred from the information present in $\mathbf{Z}$.[287, 236] Each row of $\mathbf{Z}$ is called an "embedding vector" and is a reduced-dimension representation of the whole set of original feature vectors of a given node.[287, 236]

The graph embedding process can be represented by an encoding function that takes in input a node $v \in \mathcal{V}$ and its neighborhood, $\mathcal{N}(v)$, and returns a new representation for that node:

$$\text{Enc} : (v, \mathcal{N}(v)) \rightarrow z \in \mathbb{R}^d \tag{1.1}$$

One of the most common methodologies for implementing (1.1) is the encoder-decoder paradigm [90]. This paradigm also models an auxiliary decoding function. Considering a homogeneous graph, i.e. a graph with a unique node feature space $\mathbf{X}$, the encoder function in (1.1) takes as input the

original feature vectors of $\mathbf{X}$, and outputs the corresponding embedding vectors, so it can be denoted with f, where:

$$f(\mathbf{x}_i, \{\mathbf{x}_j, \forall v_j \in \mathcal{N}(v_i)\}) = \mathbf{z}_i, \forall i \in |\mathcal{V}|$$

The decoding function, instead, takes as input a pair of node embeddings produced by the encoder, e.g. $\mathbf{z}_i$ and $\mathbf{z}_j$, and outputs a real number. Such a number is the prediction of the value of a pre-defined pair-wise relationship between $\mathbf{x}_i$ and $\mathbf{x}_j$:

$$\text{Dec} : (\mathbf{z}_i, \mathbf{z}_j) \rightarrow \hat{\nu}_{i,j} \in \mathbb{R} \tag{1.2}$$

For example, $\hat{\nu}_{i,j}$ could be the predicted cosine similarity between such vectors, or any other similarity/distance function.

The encoding and decoding functions should minimize a reconstruction loss of the form:

$$\mathcal{L} = \sum_{i,j \in |\mathcal{V}|} \Delta(\hat{\nu}_{i,j}, \nu_{i,j}) \tag{1.3}$$

where $\nu_{i,j}$ is the real value of the pair-wise relationship between $\mathbf{x}_i$ and $\mathbf{x}_j$ that (1.2) predicts, and Δ is a pre-defined discrepancy function that takes as input the predicted and real values for any pair of nodes. Once the encoding and decoding functions that minimize (1.3) have been found, the original feature vectors can be fed to f to obtain the definitive graph embedding $\mathbf{Z}$. This embedding consists of a group of points in a low-dimension embedding space.

The reconstruction loss in (1.3) is traditionally minimized after assigning initial values for the node embedding vectors and then modifying them until the correspondent optimal solution is found. Probabilistic approaches such as random-walks, or *shallow* ML algorithms like expectation-maximization can be used to solve the problem. Examples of such approaches are based on matrix factorization like laplacian eigenmaps [20], or dot-product base decoders [32, 193, 4] where the original pairwise similarity relationship is reconstructed through a dot-product of the node embeddings. DeepWalk [204] and node2Vec [82] are other examples of traditional embedding approaches based on random-walks. The decoder in this case minimizes a stochastic similarity measure between two nodes, which is the probability of traversing two nodes on a short random walk.

Deep Graph Representation Learning

The embedding approaches mentioned until know use traditional ML or optimization techniques and are referred to as *shallow* graph-RL techniques, in that they do not use deep ANNs. These algorithms are often transductive, in that optimization needs to be re-run whenever the graph changes, and the embeddings produces by these approaches might not be generalisable to new nodes in the graph that were not taken into account during the optimization phase. Moreover, shallow Graph-RL algorithms lack of parameter sharing and are thus might poorly scale in the number of edges to encode.

Deep auto-encoder architectures instead, [272] are used to perform dimensionality reduction because, different from shallow graph-RL techniques, they generalize the embedding and permit to encode data points unseen during the training phase. Among deep auto-encoders, graph auto-encoders use the encoder/decoder framework for inductive graph representation learning.[80] Deep graph auto-encoders are typically part of clustering, or link prediction pipelines on graphs whose node or edge features are originally high-dimensional. [167]

Deep GAE implement (1.1) mostly using graph neural networks (GNN). The latter architectures are the ones who derive from the ANN-based implementation of the *neural message passing framework*. Deep GAE based on GNNs typically take the original feature vectors and the adjacency matrix of $\mathcal{G}$ in input and produce the node embedding vectors in output. Deep GAEs based on GNNs are non-linear parametric functions, and after a pre-defined initialization schema, the parameters of these functions can be optimized to minimize (1.3) through gradient descent.

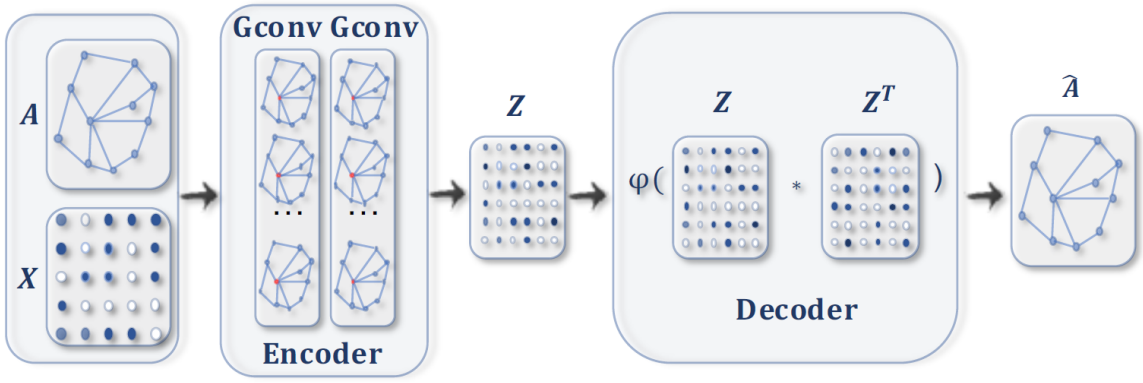

Figure 1.3. Schematic representation of Deep graph auto-encoders (GAE) taken from [282]. Deep GAEs are made stacking Graph convolution layers. Each one of the latter takes as input the node feature matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times m}$ and a pre-defined graph adjacency matrix denoted by $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$, and performs non-linear parametric operations with this information to produce an intermediate latent feature matrix which is given as input to the next graph convolution layer with the adjacency matrix. The final graph convolutional layer produces $\mathbf{Z}$ in output, which is the final embedding of the graph. A reconstruction technique then is used to produce $\hat{\mathbf{A}}$. In this example the dot-product among the features in $\mathbf{Z}$ is used. The parameters of the Deep GAE are optimized minimizing the discrepancy between the $\hat{\mathbf{A}}$ and $\mathbf{A}$.

In the neural message passing framework [74], for every node $v \in \mathcal{V}$, the GNN produces a node embedding $\mathbf{z}$ combining the feature vector of that node, $\mathbf{x}$, with an aggregation of the feature vectors of the nodes in the neighborhood set of v, $\mathcal{N}(v)$. As a consequence, the embedding vector $\mathbf{z}_i$ of a node $\mathbf{x}_i$ will be a differentiable function of $\mathbf{x}_i$ and $\mathbf{m}_i$, where the last is a message vector which is itself a differentiable function of the feature vectors of the nodes in $\mathcal{N}(\mathbf{x}_i)$. The first implementations of such models were proposed in [164, 226] and are implemented through neural networks whose learnable parameters can be represented by matrices. In matrix notation, such models can be represented as follows:

$$\mathbf{Z} = \sigma(\mathbf{W}_{self}\mathbf{X} + \mathbf{W}_{neigh}\mathbf{A}\mathbf{X}) \tag{1.4}$$

where σ is a non-linear operator like the *sigmoid* or the *ReLU* function, $\mathbf{A}$ is the adjacency matrix of $\mathcal{G}$ and the learnable parameter matrices $\mathbf{W}_{self}$ and $\mathbf{W}_{neigh}$ are responsible for combining the features of each node with the features of an aggregation over its neighborhood. Notice that the usage of the adjacency matrix implies summing of the nodes in the neighborhood of each node and thus is the simplest aggregation operator possible. Other aggregation operations are possible, like normalized sums[277], Laplacian based aggregations, among other types of set pooling approaches [309, 208, 89, 178].

One of the most straightforward models of GNN are the Graph Convolution Networks presented in [277]. Such a model implements the message passing framework through a unique learnable parameter matrix $\mathbf{W}$. Such a matrix can be seen as a parameter sharing framework between $\mathbf{W}_{self}$ and $\mathbf{W}_{neigh}$. The parameters of $\mathbf{W}$ are optimized to produce embeddings that combine -without distinction- the features of each node with a *symmetric-normalized* aggregation of the features of the nodes of it's neighborhood:

$$\mathbf{Z} = \sigma(\mathbf{W}\tilde{\mathbf{A}}\mathbf{X}) \tag{1.5}$$

Where $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times m}$ is the matrix of the original feature vectors of nodes, and $\tilde{\mathbf{A}} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ is the *symmetric-normalized* adjacency matrix of $\mathcal{G}$ with added *self-loops*:

$$\tilde{\mathbf{A}} = \hat{\mathbf{D}}^{-\frac{1}{2}}(\mathbf{A} + \mathbf{I})\hat{\mathbf{D}}^{-\frac{1}{2}} \tag{1.6}$$

where $\hat{\mathbf{D}}$ it the degree matrix of $\mathbf{A} + \mathbf{I}$. The intuition behind the multiplication with the self-loop adjacency matrix is that, for each node embedding $\mathbf{z}_i$ of $\mathbf{x}_i$, we want to combine the features of

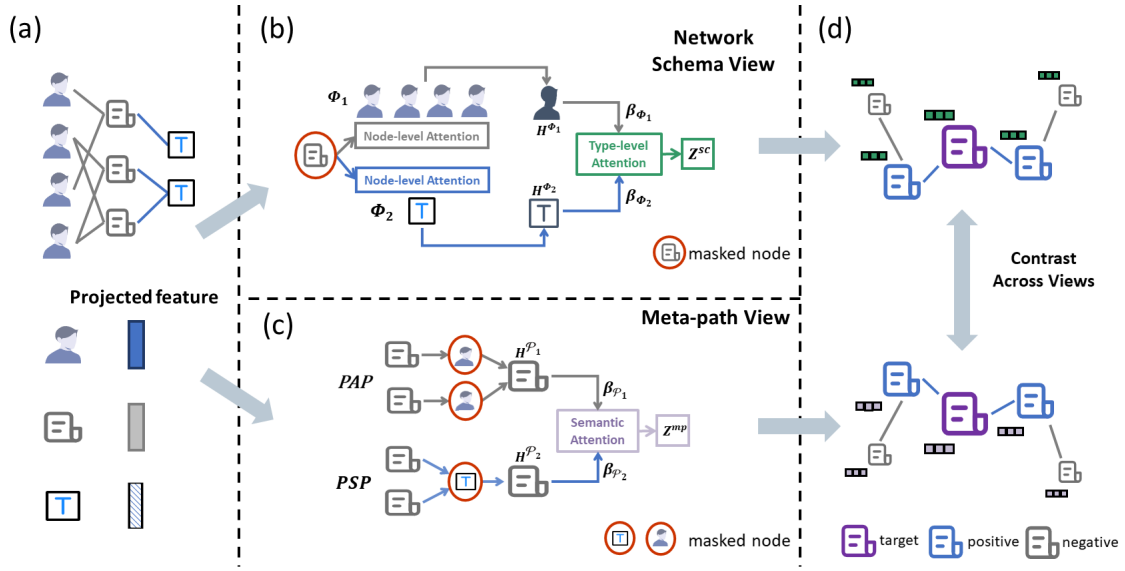

Figure 1.4. Given a heterogeneous network, a deep learning-based heterogeneous-graph representation learning (het-graph-RL) framework usually samples instances of nodes and edges from *network schemes* and/or *meta-paths* to train a deep neural network to produce the final node embeddings. Attention mechanisms are widely used in deep het-grap-rl techniques to learn to distinguish between various node and edge types and entities. This figure is a schematic representation of HeCo [271] a deep het-grap-rl algorithm that trains separate modules from network-schema-based embeddings and meta-path based embeddings and then fusions these embeddings utilizing contrastive learning.

every node in $\mathcal{N}(v_i)$ with the feature vector v itself, i.e. with $\mathbf{x}_i$, just like in the basic GNN model. The symmetric normalization operation has the objective of normalizing such an aggregation for high values of $|\mathcal{N}(\mathbf{x}_i)|$. Additional benefits related to numerical stability during the learning process are achieved thanks to the symmetric normalization of the adjacency matrix.

Stacking GNN layers, or iterating the forward pass of a GNN, is equivalent to take into account greater-order neighborhood information for composing the embedding of each node. One can stack various GCN layers to produce encoders that take into account higher order neighborhoods:

$$\mathbf{H}^k = \sigma(\mathbf{W}^{k-1}\tilde{\mathbf{A}}\mathbf{H}^{k-1}) \tag{1.7}$$

In this case, the layer k will receive as input the encoding vectors of the previous layer, $\mathbf{H}^{k-1}$, and the final encoding $\mathbf{Z}$ will correspond to the embedding matrix produced by the last layer $\mathbf{H}^{last}$. Notice also that $\mathbf{H^0} = \mathbf{X}$.

The self-loop graph neural networks are simpler with respect to the basic GNN model in that they have less parameters. However, this simplicity comes at the cost of augmenting the probability of the collapse of the learned embedding to a uniform *over-smoothed* representation. Over-smoothing means that the features of the nodes in the neighborhood of v, $\mathcal{N}(v)$, tend to dominate the formation of $\mathbf{z}_i$ with respect to the information coming from $\mathbf{x}_i$. Note that GNNs are prone to produce over-smoothed embeddings when more layers are stacked together. To reduce the over-smoothing effect, numerous solutions have been produced to induce a bias that regulates the influence of the neighborhood of the node when forming the embedding. Concatenations [89] and gated combinations [205] of the node and local-neighborhood embeddings are one example. But there exist other approaches like creating skip-connections between each layer's embeddings [139] or using combinations of the embeddings of various layers for producing the final embedding [40, 289].

Deep Graph-RL for Heterogeneous Networks

So far, we have talked about graph-RL design alternatives for homogeneous graphs, i.e., graphs in which a unique type of nodes and edges exist. On the other hand, novel challenges arise when the

graph model one wants to embed into a low-dimensional latent space or manifold is heterogeneous. Partly following Shi *et al.* [236], we summarize these challenges in three main points:

1. **Structural complexity**: homogeneous graphs have a unique node type and edge type. Consequently, neighborhoods are also uniform in type, and the unique source of complexity arises from the eventual need to encode high-order neighborhood relations in the embeddings. Instead, het-graphs have multiple node types and edge types, which complexifies the neighborhood relations even if we only consider the first-order neighborhood. In other words, a question to be answered is: *How can we encode nodes on a unique latent space in a way that we minimize particular node-type and edge-type information loss?*

2. **Attribute complexity**: We have said that graph-RL algorithms aim to minimize the topological information loss and the contextual information loss. In other words, apart of including the topological information that the edge distribution represents, a well-defined graph RL algorithm should include the information encoded by the node and edge features when embedding nodes in the corresponding latent space. We have explained how multiple algorithms tackle this problem for homogeneous graphs. However, when we need to embed a heterogeneous graph in a latent space, multiple node types might have different feature formats. For example, one node type could be featured by images while another node type might be featured by text. The second challenge is expressed by the following question: *How can we encode nodes with multiple types of features on the same common latent space?*

3. **Application dependability**: When performing graph embedding, we can assign diverse importance weights to different types of information in the graph: We could focus more on minimizing topological information loss with respect to contextual information, or vice-versa. Also, the importance of the neighboring relation and, if a node has multiple features, the importance distribution among the node features may not be uniform. It is worth noting that these preferences have more degrees of freedom in the case of het-graph-RL. The third challenge could be then expressed by the following question: *How important is it to preserve the information contained in each particular type of relation and each particular types of node of a het-graph when performing het-graph-RL?*

Note that, in het-graph-RL, our objective remains to find a single embedding feature space $\mathbf{Z}$ that encodes most of the information in $\mathcal{G}$, taking into account every node and edge type, eventually with some desired attention or importance distribution among such types of entities. The heterogeneous graph embedding process can be then represented also by (1.1), i.e., by an encoding function that takes in input a node $v \in \mathcal{V}$ and the nodes of its neighborhood set, $\mathcal{N}(v)$, and returns a new representation for that node.

RESCAL [187], DistMult [294], TransE [24] and TransH [275] are examples of shallow graph embedding algorithms specialized for heterogeneous graphs. Heterogeneous or multi-relational graph embedding algorithms introduce the relation type inside the reconstruction loss function and are used most frequently for link prediction tasks. Metapath2Vec [54], SERL [254], Multi-Net [14], and Metagraph2vec [317] are random-walk based variants designed for the embedding of heterogeneous graphs. The *Relational Graph Convolutional Network* (RGCN)[229] is instead one of the first GNN models designed for deep heterogeneous graph embedding. In such a model, an independent parameter matrix is assigned for each type of relation between the nodes of the graph, and the embedding for each node $v_i \in \mathcal{V}$ combines the aggregations produced by each neighborhood $\mathcal{N}_\tau(v_i), \forall \tau \in \mathcal{T}_\mathcal{E}$ where τ is the type of relation. Such a model is used in various deep het-graph-RL pipelines and implies a larger quantity of parameters. Various parameter sharing approaches have been proposed to simplify this approach, some of them making use of the *attention* mechanism [260, 153]. HNE [35] instead, is one of the first deep learning-based complete pipelines to implement het-graph-RL. Authors of HNE used GCN-based modules to capture complex relations between multiple types of nodes and edges. CARL [314] was one of the first works to include special-purpose

encoding modules for preserving information from unstructured semantic node features like text. Later, SNHE [315] incorporated the usage of gated recurrent units for learning semantic-preserving embeddings from text-features nodes in het-graphs. DMNE [186] is a similar framework that uses deep auto-encoders and conceives the het-graph in input as multiple edge-type-specific graphs. The HAHE [329] model instead incorporates the attention mechanism to distinguish between different meta-path types and between the neighboring nodes' types when creating the final embeddings. Similarly, HAN [270] uses attention modules to assign different importance to meta-paths and neighboring nodes but focuses on the concrete instances rather than the types of these. The Heterogeneous Graph Neural Network (HetGNN) [313] is instead one of the most used pipelines for het-graph-RL. In the HetGNN model, the neighboring nodes are separated into same-type sub-groups and aggregated. Then the attention mechanism is used to distinguish between these groups of neighboring nodes' representations rather than the neighbors themselves. Taking into account dynamic heterogeneous graphs instead, i.e., heterogeneous graphs that change over time, authors of DHNE [302] created an algorithm that trains the embedding neural networks with samples of random-walks made across multiple temporal states of the het-graph. The Graph Transformer Network (GTN) [308] exports the technique of the Spatial Transformer Networks [113] to learn meta-paths from a parametric combination of candidate edge-specific adjacency matrices and produces powerful inductive meta-path based embeddings. Edge-type specific adjacency matrices are also used by MV-ACM [323] which incorporates the usage of generative adversarial network (GAN) modules for training the embedding machinery. Contrastive learning instead was used by HeCo [271]. HeCo is a deep het-grap-rl algorithm that trains separate modules from network-schema-based and meta-path-based embeddings. It then fuses these embeddings utilizing contrastive learning. Figure 1.4 shows a schematic representation of this algorithm. Finally, one of the most recent publications in the field of het-graph-RL is R-HGNN [304]. This work presents a relation-aware embedding algorithm that uses graph convolutions over edge-specific sub-graphs and cross-relation message passing for obtaining the final node embeddings. A summary of these remarkable deep het-graph-RL algorithms is offered in Table 1.1.

Table 1.1. Some recent remarkable works in the Deep Heterogeneous Graph RL literature

Reference	Description	Main applications
HNE [35]	Convolutional neural network based, encoding of multiple node features	node classification, link prediction, clustering
HAN [270]	Attention-based weighting of meta-path and node neighbors' importance for embedding nodes	Node classification, clustering, and data visualization
HetGNN [313]	LSTM aggregation of neighbors' features attention-weighted neighbor type-aggregation	Link-prediction, recommendation classification & clustering
CARL [314]	Trained to preserve topological and unstructured (text) node-features (semantic information)	link prediction, document retrieval, node recommendation and relevance search
HAHE [329]	Attention-based weighting of meta-path type and instance importance for embedding nodes	Node classification Data Visualization
DHNE [302]	Embeddings for temporal dynamic graphs based on historic-graph random-walk & skip-gram	Node classification Data Visualization
DMNE [186]	Embeddings for nodes of multi-networks based on Deep Auto-encoders and random walks & skip-gram	Node classification Data Visualization
SNHE [315]	Deep semantic encoder with gated recurrent units	link prediction, node recommendation document retrieval and relevance search
HeCo [271]	Node-masking and combined-view contrastive learning,	Node classification, node clustering
GTN [308]	Concatenation of multiple soft-selected meta-path embeddings	Node classification
MV-ACM [323]	Generative adversarial networks for edge-type specific similarity reconstruction	Node classification, link prediction
R-HGNN [304]	Graph convolutions on edge-specific subgraphs and cross-relation message passing for node embeddings	Node classification, clustering, link prediction, data visualization

Deep het-graph-RL algorithms have been used in various fields. One example is recommender systems. Multiple data sources and the ubiquity of user profiling mechanisms offer recommender systems heterogeneous auxiliary data to infer new plausible users' preferences. For this reason, recommender systems have been modeled as heterogeneous networks. MCRec [100] uses the attention mechanism over meta-path-based representation learning for building a top-N recommender system. Given the proliferation of environments like e-commerce, social networks and Internet in general, highly heterogeneous information sources under which text analysis becomes challenging

have turn ubiquitous. HGAT [299] is a het-graph model based attention network that performs representation learning for short text classification such as tweets, queries, comments, etc. GNewsRec [102] and GNUD [103] created a deep-het-grap-rl mechanism for news recommendation. Other recent interesting applications of deep het-graph-RL include user cash-out detection [101], intent recommendation [65], share recommendation [117], friend recommendation [145], among others [55, 296, 287, 295].

1.2.2 Het-graph modeling helps solving deep learning-based optimization tasks

So far, we have described deep learning based het-graph-RL, which is one of the most important research trends in which deep learning in bringing scalability and opening the application range of graph analysis algorithms. However, the synergism between deep-learning and heterogeneous graphs is a two-way relationship. More specifically, also het-graph modeling is helping researchers to design more efficient deep learning pipelines in industrial-scale optimization problems and other complex research fields like genetics and particle physics.

Solution space dimensionality is often prohibitively large when optimization tasks are designed in industrial high-scale scenarions. Developing algorithms that efficiently explore candidate solutions to find the global optima becomes non-trivial in these environments, even for deep-learning-based algorithms. In these cases, the injection of inductive biases during the architectural and procedural design of deep-learning pipelines has proven crucial to tackle the curse of dimensionality. In machine learning, we can reduce the solution space by injecting such constraints *a priori* through multiple techniques. In this section we will describe some common examples of graph-model based inductive biases for deep learning based optimization tasks, as we believe they represent a second type of collaboration paradigm between graph modeling and deep learning. Inductive biases could be more or less explicit, for example if they are encoded as part of the loss function one tends to minimize, or if they are included in the optimization algorithm one uses. And could also be given in advance to very learning process, for example in the form of handcrafted feature engineering. Inductive biases are synonyms of regularities.

Deep learning based optimization tasks over high-dimensional solution spaces is most often achieved through the Deep Reinforcement Learning (DRL) paradigm. Great research effort has been made in the field of DRL to overcome the difficulties inherent to high-dimensional solution spaces. In this research, we have found that one clever way to enhance the effectiveness of the exploration of such solution spaces in a deep learning context is through the injection of proper *exploration biases* in DRL agents. Moreover, we have observed that het-graph models have concrete characteristics that could be exploited when designing exploration biases for DRL algorithms. After briefly introducing some important notation for DRL and describing in a high-level some common DRL algorithms, this section overviews some methods to inject exploration biases based on graph characteristics and gives some samples of how they have been exploited.

Deep Reinforcement Learning

Reinforcement Learning (RL) has its roots in optimal-control, trial-and-error learning in animal psychology, and temporal difference-learning (TD-learning). [250]. A discrete stochastic version of optimal control has been modeled by *Bellman* [21] in the so-called Markov Decision Processes (MDP). The term "Reinforcement Learning" has been used in the engineering literature since the 1960's [265]. However, with some rare exceptions, it was only from the earlies 1990's that the research community has concentrated particular attention to RL [249, 256, 276, 120, 78, 251]. RL is a framework that combines TD-learning with trial-and-error selective and associative mechanisms to learn to maximize a reward signal from a dynamic system. RL can be seen as a methodology to solve optimal control problems where the objective is to control the behavior of a dynamical system over time [22]. Moreover, Deep Reinforcement Learning (DRL) scales the validity of Reinforcement Learning-based solutions to high-dimensional and complex environments [12, 138, 69].

RL modelisation involves an agent that learns to act inside an environment to maximize a reward signal on a long-term fashion. Such interaction is modeled as a Markov decision process (MDP), with the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, where:

- The state space $\mathcal{S}$ contains all the possible environment states in which an agent can be at a particular moment,
- The action space $\mathcal{A}$ is the set of all the possible actions that an agent can take for interacting with the environment,
- When the agent is at a certain state and takes a certain action, the action could take the agent to another state of the environment, and such transitions are governed by a transition probability distribution $\mathcal{P}$,
- Finally, the RL agent receives a reward for each action taken. The environment delivers rewards following a reward policy $\mathcal{R}$ which is a function of the environment state and the actions taken.

RL agents have a degree of visibility over the state space, which embeds some information about the current environment conditions. However, state space samples most of the times do not contain full information about the environment conditions. Moreover, in typical real-world deployments of RL models, the agent has to learn to maximize the reward with partial knowledge of the transition probabilities and the reward policies of the environment. This condition has been called model-free reinforcement learning [31].

In the optimal control problems [22], the goodness of the current state and agent decisions with respect to the optimization objectives are evaluated through diverse established functions. One of these functions is the *discounted future reward*, which is the function that RL agents seek to maximize, and it is defined as:

$$G_\tau = R_{\tau+1} + \gamma R_{\tau+2} + ... = \sum_{k=0}^{\infty} \gamma^k R_{\tau+k+1} \tag{1.8}$$

where γ is a fixed parameter that takes into account the diminishing value of future action rewards with respect to immediate feedback and is known in literature as the *discount factor*. Notice that, as the dynamical systems' conditions variate in time, the functions that describe such systems depend on time. Such a dependency is indicated with the index τ in (1.8) and in the rest of this work.

Every RL agent is said to interact with the environment through an action policy, denoted as $\pi(s)$, that tells what actions to take depending on the current state s. Given a specific action policy $\pi(s)$, the *Action-value Function*, also called *Q-value function* indicates how much valuable it is to take a specific action a_τ being at state s_τ and following the policy $\pi(s)$ from the next state on:

$$Q_\pi(s, a) = E_\pi[G_\tau \| s_\tau = s, a_\tau = a] \tag{1.9}$$

from (1.9) we can derive the *recursive Bellman equation*:

$$Q_\pi(s_\tau, a_\tau) = R_{\tau+1} + \gamma Q_\pi(s_{\tau+1}, a_{\tau+1}) \tag{1.10}$$

notice that, if we denote the final state with s_{final}, then $Q_\pi(s_{final}, a) = R_a$.

The *state value function* $V_\pi(s)$ quantifies the expected return when being at a certain state s given that we are following the policy $\pi(s)$.

$$V_\pi(s) = E_\pi[G_\tau \| s_\tau = s] \tag{1.11}$$

Equation (1.11) is another important function introduced in the optimal control and MDP literature that quantifies the goodness of being at a certain state. Finally, the *action-advantage function* is

a useful abstraction that quantifies the advantage of some actions with respect to others under a same starting state.

$$A_\pi(s_\tau, a_\tau) = Q_\pi(s_\tau, a_\tau) - V_\pi(s:\tau) \tag{1.12}$$

The TD-learning mechanism uses (1.10) to approximate the Q-values for state-action pairs in the traditional *Q-learning algorithm* [276]. However, in large state or action spaces, it is not always feasible to use tabular methods to approximate the Q-values. When the environment is complex and the state and action spaces are high-scaled, approximators based on Deep Artificial Neural Networks (Deep ANN) are the unique alternative to model these value functions, as opposed to traditional tabular methods. Consequently, in the last decades, the application domain of Reinforcement Learning has been widely extended thanks to the usage of Deep ANNs. In sunmmary, Deep Reinforcement Learning exploits the powerful function approximation capacities of Deep ANNs to converge to optimal policies for high-scale MDPs.

The combination of Deep ANNs and RL has originated many DRL frameworks. Such frameworks can be differentiated between them by which of the value functions constitute the learning objective of the agent:

- The objective of *Policy-Based* DRL agents is to learn an *action policy*, denoted as $\pi(s)$, that maximizes (1.8).[232] The Deep Deterministic Policy Gradient based reinforcement learning algorithms has provided feasible solutions on high-dimensional action spaces, mostly by the use of the Deep Deterministic Policy Gradient (DDPG) algorithm [142]. However, as pointed out by Dulac-Arnold *et. al.* in [57], DDPG is not suitable for finding solutions over large discrete action spaces, several optimizations like nearest-neighbor approximate search through the action space, or the leverage of prior information about the actions can show to be critical to achieve the a satisfactorily performance.

- In *Value-Based* DRL frameworks instead, the agent learns to approximate $V(s)$ or $Q(s,a)$, using also a model of $A(s)$ as an attention mechanism to differentiate between similar valued states. One of the most famous value-based DRL algorithms is Deep Q-Learning [172], which performance has been enhanced by the research community with the introduction of various mechanisms such as target-networks [94], action-advantage based attention mechanisms [274], among others.

- Finally, *Actor-Critic* methods combine policy learning and value-based DRL[81]. This combination has enhanced the learning performance because the value functions act as a baseline that reduces the variance of the estimation of the gradients of the policy function. Despite Actor-Critic methods are not the only abstraction that merges policy-based and value-based reinforcement learning [179], they have received major attention by research community.

Notice that all the value functions described above are dependant of a given policy $\pi(s)$. The goodness of actions, states and state-action pairs are approximated *within* a certain policy. In other words, we are able to approximate values that depend on the future because a specific policy is supposed to govern the agent actions at least from the next step on. In fact, different policies imply different values for a defined state, action or state-action pair. We have said that RL agents learn to approximate a policy function $\pi^*(s)$ or a value function $Q_{\pi^*}(s,a)$ or $V_{\pi^*}(s)$ within a such a policy. However, another important dimension upon which DRL frameworks can be differentiated is centered precisely in the policy $\pi(s)$.[12, 250] *On-policy RL* agents learn to optimize $\pi(s)$ or a value function of it while interacting with the environment using the same policy $\pi(s)$. *Off-policy* RL agents instead learn to approximate a policy $\pi^*(s)$ or a function of it while acting with another policy $\pi(s)$. The latter approach permits to handle better the important trade-off between *exploration* of the action space and *exploitation* of the acquired knowledge about the optimal policy. Generally, off-policy RL algorithms are most widely used in practice, because they warrant better exploration of the action space and thus avoid to stuck the convergence of the policy on local optima. The

disadvantage of Off-policy DRL algorithms is that the variance of the estimates they learn augment notoriously, making convergence a slower process [250]. However, asynchronous gradient descent has proven effective for robustly applying on-policy DRL approaches to complex problems. The usage of multiple agents learning in parallel has a stabilizing effect on learning the optimal policies. The most prominent example of this idea is the Asynchronous Advantage Actor Critic (A3C) algorithm [170].

DRL algorithms have recently evolved to solve problems on high-dimensional action spaces through the usage of state-space discretization, Policy Learning, and sophisticated Value learning algorithms [274, 142, 57]. Examples of DRL industrial applications are Autonomous Driving Cars [123], Industry Automation [284, 203], Robotics [86], Cybersecurity [185], Power Systems [322], Communications and Networking [148], Economics [176], Healthcare [303], among others.

Reward Shaping in DRL

DRL algorithms learn optimal control policies through reward signals with none or incomplete environment knowledge. This often is a problem when the RL environment is characterized by sparse rewards, i.e. the agent has to take long series of actions to receive a reward signal from the environment. Sparse rewards in DRL environments could be affected by low training performances. In other words, the quantity of state-action-reward transitions that the agent needs to perform before converging to optimal policies could be prohibitively large [83].

Fortunately, *a priori* knowledge of the environment permits to create exploration biases to improve the performance of DRL training cycles. The main enabler for creating the exploration biases in DRL is model-based reward policy design, also referred to as *reward shaping*. Reward shaping is in fact an approach to solving problems in DRL modeled environments with sparse rewards [263, 104, 56, 306, 84, 10]. Given a particular optimization goal, reward shaping (RS) consists in introducing some "hints" to the agent injecting specialized rewards on the states of the Markov chain proportionally to the convenience of being at such state. These hint rewards are additional with respect to the baseline reward associated with the main goal. For example, in a robot soccer player environment, the main reward would be assigned when the robot scores a goal, but extra rewards could be associated with the inverse of the distance between the robot and the ball, or the inverse of the distance between the ball and the goal.

RS techniques are solutions to the temporal-credit assignment problem in DRL. When using reward shaping, the learning process is influenced by an exploration bias [13], which forces the exploration of the regions of the state space that result in better long-term rewards. In other words, reward shaping provides the agent hints to explore the action space with greater effectiveness with respect to a pure model-free approach. For this reason, reward shaping is a crucial enabler for the scalability of DRL algorithms to high-dimensional action spaces.

Potential-based reward shaping [50, 71] is a framework for RS that helps to avoid convergence to sub-optimal policies. Potential-based RS defines a potential function which assigns a convenience score to states taking into account the desired optimization goal. Whenever an agent takes an action, the potential of the final state is added to the baseline reward and the potential of the initial state is subtracted. Potential-based reward shaping produces equivalent results as a biased initialization of the state value or state-action value functions: it helps the agent effectively explore the action space without altering the optimal policy it must converge to.[85] De Moor *et al.* [48], for example, used a potential-based reward shaping for improving baseline DRL policies in a perishable inventory management context.

Numerous industrial fields are incorporating domain-specific RS to model-free DRL algorithms to increase the agents' training performance and converge to higher long-term rewards. For example, Zhang and Bailey [316] created an accurate RS schema for the Deep Deterministic Policy Gradient (DDPG) [142] and Proximal Policy Optimization (PPO) [231] algorithms to correct obstacle avoidance and navigation problems in robotic control. Shaik [233] created an enhanced reward function for a DRL algorithm for achieving human-like accuracy on complex tasks. Afshar *et al.* [2] used reward

shaping for efficient reserve price optimization instead.

RS is an approach that helps to model domain knowledge into reinforcement learning algorithms [85]. In cases where the domain model is formatted as a graph, whether heterogeneous or not, ML designers could tailor specialized reward signals considering the structural properties of a graph and the usage of graph algorithms. For example, consider optimal path-finding tasks in heterogeneous networks. Given a specific objective function to be maximized, RS could be straightforward if we consider domain-specific constraints and structural constraints of the optimal paths one needs to find. One example of this kind of graph model-based reward shaping comes from the field of knowledge graph (KG) reasoning. Xi *et al.* [268] proposed to use RS in a question answering task over a knowledge graph: agents' actions were to predict the next node in the path for constructing a reasoning path in the KG. Their work performed better than previous solutions [234, 47, 288] mainly because their RS was based on the model knowledge. Authors of these works, in fact, created a soft reward for target entities whose correctness cannot be determined, rather than just giving a reward if the paths reach the correct destinations. Authors in [268] also created an action dropout schema to enhance exploration.

In the rest of section 1.2.2 we mention three main approaches that can inject efficient exploration biases to DRL algorithms based on graph-model based reward shaping: Action-space serialization, action-space abstraction, and procedural biases based on graph algorithms. We finish this section mentioning an emerging technique for creating more architectural graph inductive biases in the DRL, dubbed graph reinforcement learning.

Graph-based action-space serialization

When dealing with DRL in sub-graph related objectives, the ideal behavior that agents are meant to learn can often be decomposed into a series of simpler actions focused on single edges or single nodes of the underlying graph model. For example, if we think of a community detection problem, we can conceive the process of sub-graph selection into a series of simple actions like node and edge selection. Also question-answering process through multi-hop knowledge graph reasoning could be conceived as a series of edge selections from a question node to an answer node. Generally in DRL, the decomposition of complex actions into series of sub-steps takes the name of action space serialization. Provided that a correct dense reward mechanism exists, action space serialization could improve parametric efficiency and thus help the convergence of DRL algorithms to optimal policies. In graph models, complex goals are usually composed of series of single actions like node and edge selection, edge creation, node creation, etc. We now give some examples of how this structural property of het-graph models has helped to design efficient action spaces in DRL contexts.

Serialization of action spaces is a common technique in network environments where the downstream task implies the selection of a sub-graph. Examples are community detection, virtual network embedding, among others. If we refer to virtual network embedding, many authors have chosen to model DRL agents where the action of embedding a virtual network to a substrate physical network is serialized into a sequential operation of selecting one-by-one the nodes of the network that are going to host each one of the virtual network components. [285, 118, 200, 201, 202] In other words, the actions of the reinforcement learner are the single physical-node assignation decisions for each VNF component of a virtual network. This kind of serialization has helped analysts to create agents that learn optimal sequential policies in network environments while facing problems one node-at-a-time. Notice that serialization of the state space could also permit to feed the RL agent with reduced local observations of the environment state, - e.g. the region representing only the neighborhood of the current node- and thus reduce the complexity of the underneath ANN-based policy or value function approximators. If many real-world environments modeled as graphs -think for example of diffusion processes, it is likely that the state that matters for an agent to take actions is relevant only at a somehow-local region of the current point.

Finally, we mention that action serialization is the exact opposite of the so-called temporal action abstraction which instead composes the baseline actions and their correspondent rewards to create

more coarse grain actions -dubbed as *options* in the RL context- and rewards and redefine a MDP that drives the training of the agent. Temporal action abstraction is useful when there are multiple similar valued actions that could create excessive reward sparseness and lead to inefficiencies in the DRL training process. In other words, when multiple similar fine-grain action paths could lead to a same sparse reward, a regular class of grouped actions could be conceived as an atomic action that receives immediate reward when executed. In these particular situations, temporal action abstraction schemes are modeled, and this technique could help the agent to generalize over actions and efficiently explore the solution space.

Temporal abstraction over actions in DRL is the basis of the so-called hierarchical RL [126] and could help to scale to high-dimensional action spaces or environments with sparse rewards. In other words, a reinforcement learner could be made to perform a long series of specific actions before reaching a unique final reward, or could be designed to perform an aggregation of such specific actions at once, receiving more dense rewards. It is worth nothing that, provided that rewards are consistent, the second case would require less training effort.

Think for example of a robot that needs to efficiently navigate across the rooms of a house. In this context, multiple steps could be aggregated into a single action that focuses reaching and traversing specific doors. These actions could be grouped as a unique action in a coarse (partially observable) MDP that would have its own atomic rewards for driving the training more efficiently. Network related applications of temporal action abstraction are perhaps more associated to homogeneous network environments, where single link or node related actions could be grouped without loss of significance with respect to a final objective.

Finally, we stress that both action serialization and action composition or abstraction are ways in which the reward policy is shaped to be more efficient in leading the agent to the desired solutions with respect to baseline rewards. Notice that action serialization is used when the creation of fine-grain specialized rewards is possible. These rewards could help learning the fine-grain actions sequences with respect to an objective, specially when each different action taken might influence decisively over the final reward entity. As an example, think of a resource allocation problem in a virtual conferencing infrastructure. Session resource allocation has consequences on the future state of the system resource availability and, if the goal is to maximize the service availability and minimize operational costs in a wide time-horizon, it might be important to allocate sessions in a proactive fashion with special attention to server consolidation. Allocating the wrong session in the wrong server instance could compromise service availability for a long period, and thus it could be misleading to group session allocation actions into atomic group allocations. Generally, if each single action could potentially change the exploration route and lead to very different -eventually sparse- rewards, then action composition is a very delicate and potentially erroneous design strategy. As a conclusion, we must say that there is always going to exists a bias-variance trade-off in the credit assignment problem that the design of inductive biases through action and reward modeling needs to be careful of.

Graph-based action-space abstraction

We now overview some examples of how graph-modeled scenarios have exploited graph-analysis and network theory to design efficient model-based reward shaping to enable action space abstraction.

Graph-model knowledge could help the experts to design the action space in a way that the agent focuses in coarse grain actions like meta-path selection rather than focusing on fine-grain actions like edge selection. For example, if the task is to identify a sub-graph, fine-grain actions like edge selection could be conceptually aggregated into larger abstract actions like meta-path selection. The meta-path would be made of various edges and possibly include various types of edges.

Concretely, given that the environment is modeled through a graph, composition of the action space could be easily implemented creating explicit programs that exploit graph-analysis to find optimal fine-grain actions taking as input simple constraints that need to be accomplished. Such high-level constraints would constitute the actions of the optimal policy that needs to be learnt by

the reinforcement learner. For example, authors of EAMCM [9] created a DRL agent that learns to create near-to-optimal VNE on substrate physical networks. In this case the heterogeneous network under analysis is made of physical nodes, virtual nodes, physical links and virtual links. The raw action space of a VNE agent would be high-dimensional because, given a chain of VNF modules, the agent needs to specify the physical nodes that are going to host each one of these and also the physical network paths that are going to implement the virtual links between each VNF in the chain. Virtual links could exploit none, one or several physical links, and searching the optimal physical path augments the action space exponentially.

Given that the topology of the underlying physical network is known, EAMCM [9] exploits graph analysis to compute the optimal routes between two physical nodes. For each VNF, the agent proposes an order in which nodes of the substrate network will be considered for the deploying the VNF. The first node that has available resources to host the VNF is the chosen one for the task. After iterating on the VNFs of the chain, the set of physical nodes upon which the VNF instances are to be deployed is fixed. Then, the action space for the routing decision task consists of a map of weights for each link in the substrate network. Given the previous fixed VNF placement, the chaining is performed using the Dijkstra's algorithm considering the weights of the links proposed by the agent. Authors use the term *proto-action* to refer to the output of their ANN based RL agent, because the real action, which constitutes the whole mapping of the VNF chain in the physical network, is computed using model-specific knowledge.

A very similar exploration bias in induced in the work of Yan *et al.* [292], where the agent that converges to optimal VNE policies chooses only the physical nodes that are going to host the VNF modules (serialization of the action space). The main exploration bias that they inject, however, is that, each time the agent selects a new physical node, the physical path that hosts the virtual link among the previous and the current VNF is computed through a hybrid search that combines searching for the shortest physical path and assigning the first available physical path. Thus, the RL agent does not care about physical link assignment.

It is worth mentioning the main drawback that this kind of model-based action space design could have. This scheme inherits the performance/precision balance of the underlying graph algorithm that resolves sub-tasks. In fact, even if the policy in [292] converges to robust VNE policies with respect to other state-of-the art techniques, it's authors cannot claim to converge to a global optimum policy. From this consideration it follows that also the scaling capacity of the sub-task solver is also an inherited property for the overall DRL agent.

Other examples of action abstraction come from the field of DRL-driven network routing algorithms. Works like [151, 38], model a simple action space where the agent is simply told to choose from a set of candidate end-to-end paths for routing. Given the ends to the request, the set of candidate paths is pre-computed with an external routing algorithm. Notice that action-space abstraction usually interferes with the design of the state-space, because DRL designers usually tend to create inductive biases for the agent to easily extract significant features from state-representations that permit the agent to choose the proper actions when observing states.

Graph-based procedural biases

Generally, well designed inductive biases help to improve convergence to optimal policies in high-dimensional environments, whether these biases are network-related or not. One last design strategy that its worth mentioning, even if it is not necessarily related to network domains, is a more direct way of creating exploration biases. In fact exploration biases can be designed in controlled environments like simulators, and imply *a priori* knowledge of the state transition function. We refer to the effects of this kind of modification of the transition function as *procedural bias*, in that it affects the results of the learning routine of the DRL agent on a more direct way with respect to the previous mentioned exploration biases. One procedural bias can be created forcing multiple action-state transitions under certain conditions. In other words, forcing the actuation of fixed fine-grain action policies in determinate state-space regions. Another example is resetting

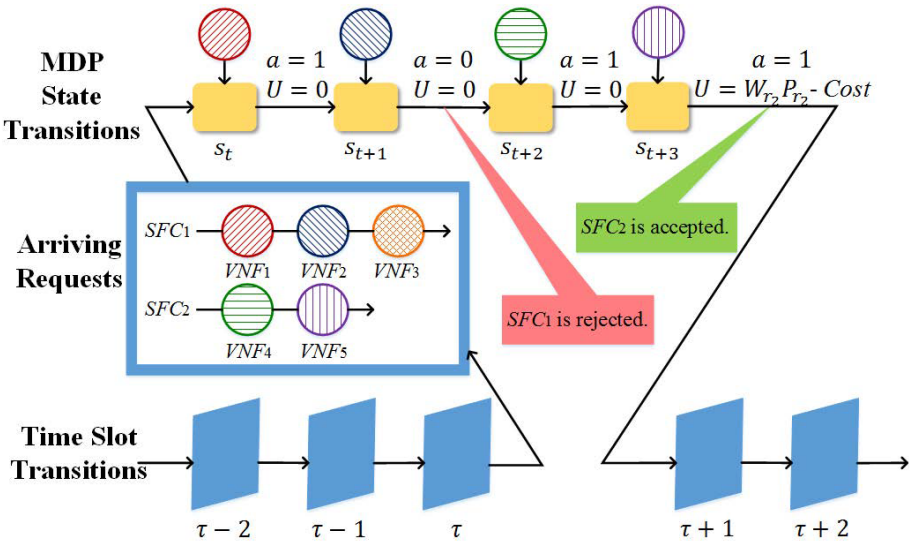

Figure 1.5. Service Function Chains (SFC) are composed of various virtual network functions (VNF) that need to be deployed onto a physical network. Before processing an SFC chain, in NFVDeep, the network state is backed up and resetted to the previous state whenever the reinforcement learning (RL) agent fails to deploy an SFC. When alterations of the normal transition function can be forced to happen, DRL designers could inject *exploration biases* to the reinforcement learner. We call the effect of this direct modification of the normal transition function a *procedural bias*, in that it directly affects the convergence results of the RL agent.

the environment to an initial state whenever the agent performs a specific action, in order to prevent the agent to consider specific actions. The resultant effect is that the reinforcement learner exploration is limited to a specific sub-region of the action space. Generally these procedural biases are case-specific and help the agent not to explore regions of the action space that could lead to converge to sub-optimal policies.

Authors of NFVDeep [285] created a DRL based algorithm for service function chaining (SFC) deployment. Each SFC is composed of a set of virtual network function (VNF) modules to be deployed to the substrate hosting network. Authors in [285] designed a *serialization and backtracking* algorithm that keeps track of the network state at the beginning of each process of SFC deployment. Figure 1.5 shows a schematic representation of this serialization and backtracking algorithm. Actions that assign a VNF module to the a substrate network are denoted by a, the obtained reward at the end of a complete SFC assignation routine is U. NFVDeep resets the network state conditions to the last tracked situation whenever a SFC fails to be deployed, as a result, no reward is given to the RL agent.

Thanks to this backtracking algorithm, NFVDeep is able to converge to a best-effort SFC deployment policy with a simple policy learning DRL schema (REINFORCE), without the need of any experience replay. The main drawback of these and other similar model-based approaches is the sub-optimality that could derive from limiting the exploration region of the state space. In the example mentioned above, one could argue that the agent is not able to plan the occupation of the network resources on a long-term basis, because the biased transitions do not permit the agent to experience the consequences of (and learn from) network overload conditions. Thus, we can say that the applicability of model-based procedural biases in DRL -not only in graph modeled environments- is limited by two main factors: learning environment control, and theoretical basis to restrict the exploration of the action space without loose of generality for optima.

Graph reinforcement learning

As we have already explained in the previous paragraph, in any RL framework, the agent takes actions based on observing the environment's conditions. In the case of DRL, the environment's

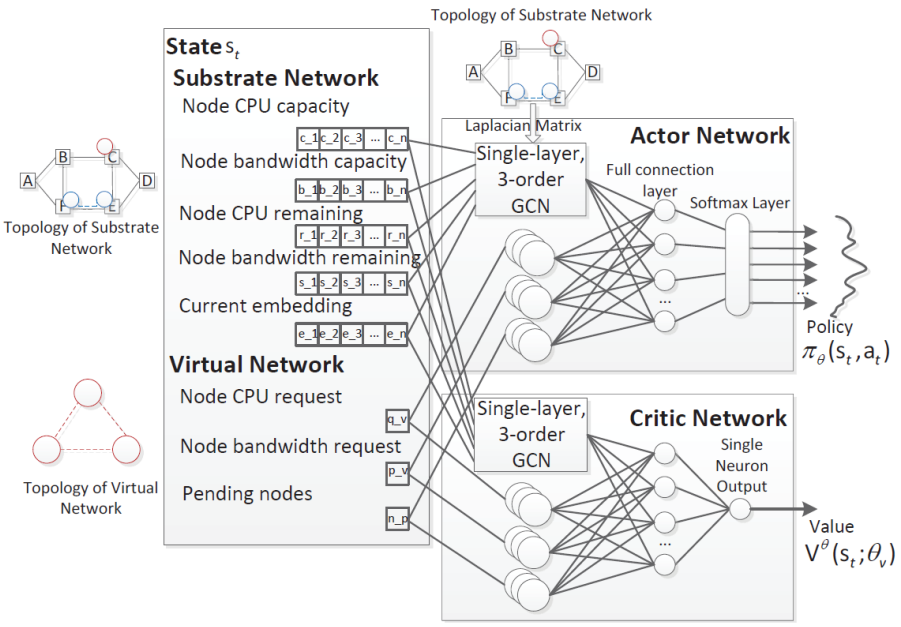

Figure 1.6. In Graph Reinforcement Learning, the architecture of function approximators contains modules that implement the neural message passing framework. By doing so, designers create inductive biases for the agent to learn the topological characteristics of a network environment. In this figure, which has been published in [292], we can see that a 3-layer GCN is used in both the actor and critic networks to extract features from a heterogeneous network made of physical nodes and links and virtual nodes and links. GCNs reduce the dimensionality of the original feature matrices exploiting the topological structure of the underlying networks. Graph Reinforcement Learning leverages scalability capacity to network-based problems.

conditions are made (partially) observable to the agent by any vector representation technique. Observations are then conceived as point samples of the so-called state-space, which is induced by the concrete state-representation technique.

In het-graph models, the environment state usually needs to represent a snapshot of the current network conditions -i.e. all the node and edge conditions- apart from other information that may be useful to solve specific downstream tasks. For example, suppose we need to find an optimal path for data transportation inside a computer network. In that case, we should be aware of both the characteristics of the links connecting computers and the characteristics of the data we intend to transmit.

Feature matrices can represent node and edge level features of a het-graph. The main artificial neural network architectures that adapt to the unstructured nature of graphs are graph recurrent networks (GRN), graph convolutional networks (GCN), graph adversarial networks (GAN), and deep graph auto-encoders (GAE). All these architectures are under the name of graph neural networks (GNN). Excellent surveys on these architectures exist. [282, 320, 326, 305, 225] Using GNNs creates inductive biases to help the reinforcement learner take into account the topological characteristics of a network-based environment when processing the state vectors.

The architectural inclusion of graph neural networks for DRL function approximators has been referred to as *Graph Reinforcement Learning* (GRL). Multiple examples of these techniques are available in literature [168]. For example, authors in [292] use a three-layered graph neural network to effectively process the global network state on an optimal virtual network embedding task. This task consists of mapping a virtual network onto a physical network. The physical network has various node and edge features that compose a set of constraints for the allocation scheme of the virtual network. A DRL agent needs to consider these network features; thus, they need to be included in the state representations for the agent to decide the best allocation actions. GCNs

have been used inside the architectural corpus of the value function approximators to create proper inductive biases for the agent to consider the network topology apart from the single node and edge features. Figure 1.6 schematizes the Actor-Critic framework proposed in [292].

Authors in [106] propose instead the usage of GCNs for intelligent traffic routing control on software-defined networking (SDN) sensor networks. The authors of this work model the state space samples as the reduced dimension vectors at the final layer of the GCN that ingests the adjacency matrix and the node feature vectors. Advances in deep learning-based molecular generation [106], knowledge graph reasoning [47, 7], traffic-signal control [298, 105], resource allocation in vehicle communication [88], train rescheduling [190], among others, have also been made possible through the usage of GRL. Finally, GRL has proven efficient in creating cooperative inductive biases in multi-agent reinforcement learning. [119]

Chapter 2

Specialized Applications and Open Challenges

The previous chapter exposed the state-of-the-art techniques that combine deep learning to het-graph modeled scenarios. We identified two main paradigms of collaboration, and two concrete practices that mostly implement these collaboration paradigms were thoroughly studied. Instead, this chapter focuses on a more vertical penetration of two concrete network-based research fields. More specifically, this chapter exposes how deep learning combined with heterogeneous graph modeling has tackled research challenges in two concrete research areas that are widely studied in the industrial and academic fields. These are the video content delivery networks (CDN) [198, 64] and the gene regulatory networks (GRN)s [238, 132, 216, 75]. Finally, we identify one research opportunity in the form of an open challenges for each one of these research fields.

2.1 Het-graph modeling in the Video-Delivery Industry

This section introduces the concept of video content delivery networks. It then focuses on the common problems that the video delivery industry has recently managed to solve with the help of het-graph modeling and deep learning. We specially focus on the QoS and Cost optimization problem in virtualized content delivery networks. Finally, we mention how the multi-objective online optimization of virtualized live-video delivery systems is an active area of research that could benefit from synergies between deep learning and het-graph modeling.

2.1.1 Video Content Delivery Networks

We use the term video-delivery industry to refer to the delivery of video-formatted content through the Internet. In the last years, the consumption of video by Internet users has grown. Video quality standards have increased and Live-Streaming is gaining importance with respect to Video-On-Demand. [41] Video traffic occupies more than three-quarters of total internet traffic nowadays, and the trend is to grow. [42] As a consequence, the video-delivery industry has positioned itself as an expanding industrial field with a lot of research effort dedicated to it. [64] Stakeholders in the Video-Delivery Industry are commonly classified into Clients, Content Providers (CP), Internet Service Providers (ISP) and Content Deliver Network (CDN) providers. Various commercial schemes have evolved in the last years among these actors to meet the user needs and maximize commercial fairness.[196, 278, 217, 244, 79]

CDNs are distributed systems that optimize the end-to-end delay of content requests over a network [29, 173], and are based on the redirection of requests and content-replication. [198, 174] The replication of content in Internet for optimizing delivery purposes is commonly referred to as *content-caching*. Well-designed CDN systems are distributed content-caching solutions that warrant good quality of service (QoS) and quality of experience (QoE) for users distributed in low,

medium or high scale. Content providers need Video CDNs to be efficient, scalable, and adaptive [27]. For this reason, cost and QoS optimization for video content delivery systems is an active research area. A lot of the research effort in this context is being placed on the optimization, and modeling of Content Delivery systems' performance.[49] CDN performance and cost optimization paradigms are multiple. The delivery system infrastructure, the data representation, the routing and caching algorithms, are optimized by industry following the needs of the users. [215, 174] CDN industry has focused in the last years in proactive optimized video delivery, caching, transcoding, recommendation, and has widely relied on graph models for solving these and other similar complex problems. [149, 135] Graph theory in fact, has helped to achieve important milestones in the design and evolution of such systems.[136]

2.1.2 A common network taxonomy of video CDNs

Traditional CDNs optimize the delay in video delivery using function-specific replication servers to store content in the network proximities of the end-users. Peer-to-Peer (P2P) networks instead, delegate content-caching to collaborative end-users, minimizing the deployment of function-specific nodes in the network. P2P-oriented CDNs depend on the collaboration of end users. Lastly, hybrid schemes that use both function-specific servers and client nodes to deploy CDN functionalities also exist and are reaching good performance and acceptation.[1] Whether P2P architectures are used or not, we can model CDNs as a network containing different types of nodes:

- Client: The receiver of the Video. The client node is the node responsible of displaying any video for the end-user. Normally it is positioned at the edge of the network and interacts directly with the CDN delegate servers.
- Origin servers: Are responsible for the generation of the video streams. Origin servers are located in the premises of CPs and interact directly with the CDN delegate servers. Notice that in the context of crowd-sourced Live-streaming (i.e. social video platforms), clients and content providers are the same.
- CDN delegate servers: Are function-specific modules located in the inter-connection path between the origin server and the client. These modules are commonly in charge of realizing delivery-oriented pre-processing steps like content redundancy, caching, redirection, transcoding, multicasting, load-balancing, among others. These function-specific modules also connect the clients and content providers to the ISP to scale the distribution of video. Notice that P2P or hybrid CDNs could deploy these function-specific modules in the client nodes.

Taking into account the above mentioned taxonomy, network-based models of CDNs can be more or less complex depending on the particular analysis and optimization needs. In many cases, nodes of a CDN network model will include various features like available functionalities, cache status, resource capacity, utilization, network availability, etc. Moreover, if we take into account the multiple network functions mentioned above, we notice that CDN delegate servers themselves can be differentiated between classes. On the other hand, the network connections between the nodes of a CDN are normally defined as the edges of the graph model. Notice that also the edges of CDNs might be differentiated in various classes and might contain different feature types. The plurality of node classes and features makes the CDN a heterogeneous network.

Edges of a CDN graph model might abstract physical network paths or might correspond to physical connections among specific network components. In both cases, the edges could be classified in various classes. For example, one could differentiate between inter-ISP and intra-ISP connections. Inter-ISP connections could additionally differentiate themselves between peer-to-peer connections and client-server connections. If we speak of client-server connections, one can distinguish connections between a CP and a CDN delegate node from connections between the end-user device and a CDN delegate node. Each connection type could have different features like costs, available bandwidth, transmission delay, etc. Some of these features might be reasonable to

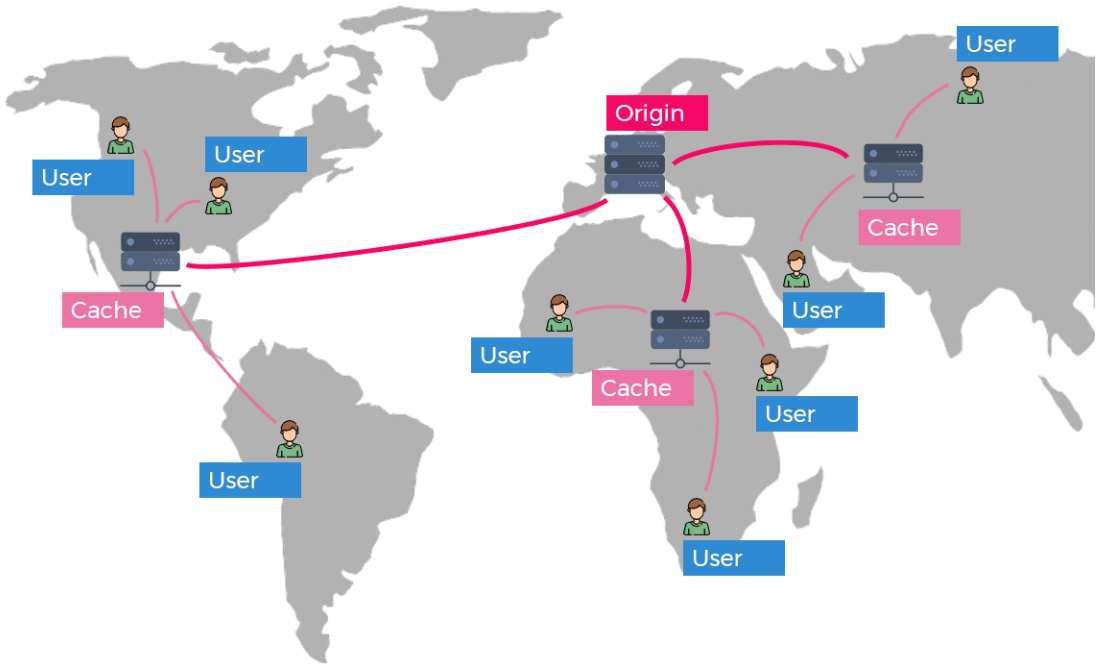

Figure 2.1. Content Delivery Networks (CDN) are distributed systems that optimize the end-to-end delay of content requests over a network, and are based on the redirection of requests and content-replication. The replication of content in Internet for delivery delay-optimizing purposes is commonly referred to as *content-caching*. Notice that a CDN can be straightforwardly modeled as a heterogeneous graph. This figure is obtained from [245].

include in certain types of edges, while others will not. For example, intra-ISP connections might not need to take into account data transportation costs, while inter-ISP connections might obviate instead the bandwidth capacity.

When modeling a CDN, we could also differentiate between video delivery through *Local* and *Wide* Area Networks. In the first case, the network dimensions will be smaller with respect to the second case. CDNs deployed over Wide Area Networks could use hierarchical caching schemes in which case different node classes could be modeled to represent different types of cache servers like metropolitan, regional, etc. *Video-on-Demand* (VoD) vs. *Live-Streaming*, constitutes another important factor to define a video CDN model. Live-Streaming contexts are less tolerant to session start-up delays and service function chain re-assignation. For this reason, a live-streaming CDN model should include node characteristics as the current utilization of the nodes, to permit the orchestrator module to better discern the server assignations. Clearly, also *Multicast* versus *Unicast* delivery has to be differentiated when modeling a CDN, and video formatting options [215] also count when modeling video CDNs, specially if one needs to keep track of network bandwidth usage, users Quality of Experience (QoE), etc.

2.1.3 Virtualized CDNs

Another important classification of CDNs is related with the deployment: Traditional CDN systems are deployed on dedicated hardware while virtualized Content Delivery Networks (vCDN) use Network Function Virtualization (NFV) [300], and Software-defined Networking (SDN) [157] to deploy their software components on network function virtualization infrastructures (NFVI) like virtual machines or containers. CDN network components are dynamically deployed in the form of Virtual Network Function (VNF) modules. These modules are deployed into a set of interconnected virtual machines which we can refer to as VNF hosting nodes. Notice that VNF hosting nodes compose the NFVI. Moreover, the VNF hosting nodes are commonly deployed on physical nodes of

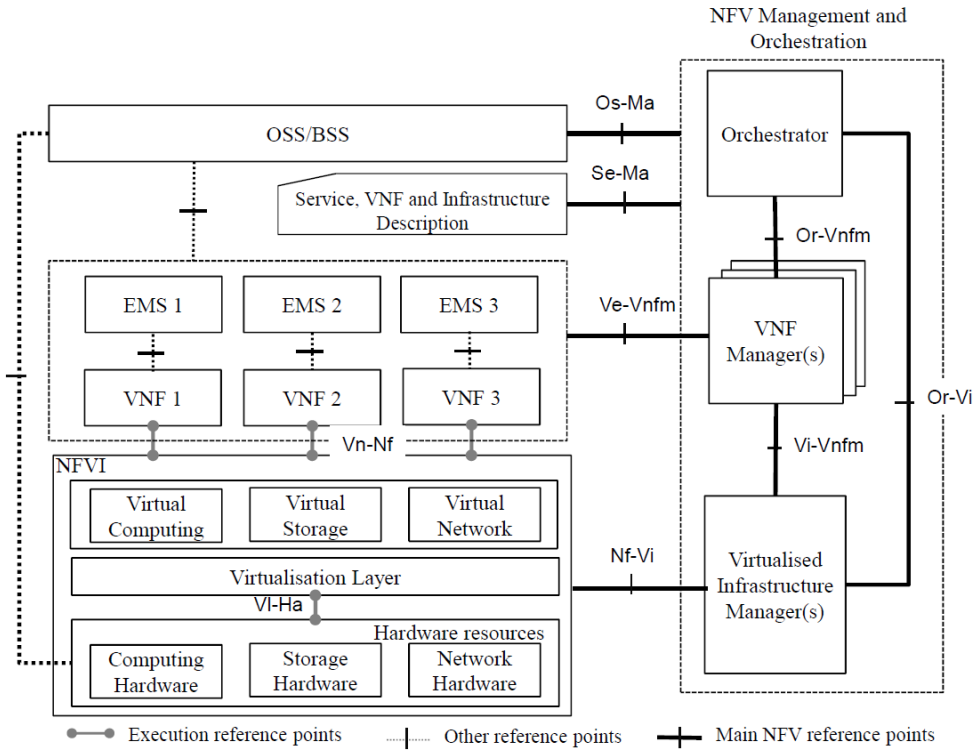

Figure 2.2. In this thesis we propose to model the virtualized content delivery networks (vCDN) following the network function virtualization management and orchestration (NFV-MANO) architectural framework for virtualized network systems published by the ETSI standard group specification. In this standardized architecture, the orchestrator module is in charge of deploying the service function chains, i.e. the allocation of the set of virtual network functions in the virtual network. The orchestrator coordinates and delegates the VNF instantiation and lifecycle to the VNF manager(s) module(s) and the virtualized infrastructure manager is in charge of dynamic resource allocation for the virtual infrastructure. This figure was obtained from [62]

the so-called *substrate network*. To model a vCDN it could be necessary to keep track of both the substrate network that hosts the NFVI, and the NFVI itself. Complex heterogeneous network-based models like these are often needed when optimizing costs or performance in vCDN contexts.

2.1.4 Cost and QoE/QoS optimization in virtualized CDNs

Virtualized Network systems are usually deployed as a set of composite chains of Virtual Network Functions (VNF), often called *service function chains* (SFC). Every incoming request to a virtualized network system will be mapped to a corresponding deployed SFC. The problem of deploying an SFC inside a VNF infrastructure has been called the *VNF Placement* or the *SFC Deployment* [285] problem. Many service requests can share the same SFC deployment scheme, or the SFC deployments can vary. Given two service requests that share the same requested chain of VNFs, the SFC deployment will vary when at least one pair of same-type VNFs are deployed on different physical locations for each request.

One important measure of Quality of Experience (QoE) in video delivery is the *session startup delay*, which is the time the end-user waits since the content is requested and the video starts to be displayed in the receiver's device. One important factor that influences the startup delay instead, is the round-trip-time (RTT) of the session request, which is the time between the content request is sent, and the response is received. Different SFC deployment schemes for a given a video delivery request might result in different RTTs depending on various factors, we now enlist the principal ones:

- VNF Cache-status: Notably, cache HIT and cache MISS events may result in very different request RTTs. Consequently, a realistic vCDN model should keep track of the caching memory status of every cache-VNF module for fine-grain RTT simulation.

- VNF resource utilization: An important factor that influences RTT computation is the request processing times. Such processing times will notably depend on the current VNF utilization. To model VNF utilization in a video-delivery context, major video streaming companies [280] recommend to consider not only the *content-delivery* tasks, but also the resource consumption associated with *content-ingestion* processes. In other words, any VNF must ingest a particular data stream before being able to deliver it through its own client connections, and such ingestion will incur non-negligible resource usage.

- VNF instantiation times: A realistic vCDN delay model must incorporate VNF instantiation times, as they may notably augment the starting delay of any video-streaming session.

- Heterogenity in VNF characteristics: Finally, both instantiation time and resource consumption may differ significantly depending on the specific characteristics of each VNF[49].

The QoS/QoE goodness of a particular SFC deployment policy is generally measured by the mean acceptance ratio (AR) of client requests, where the acceptance ratio is defined as the percentage of requests whose RTT is below a maximum threshold [285, 147, 107]. Notice that RTT is different from the total delay, which is the total propagation time of the data stream from the origin server to the end-user.

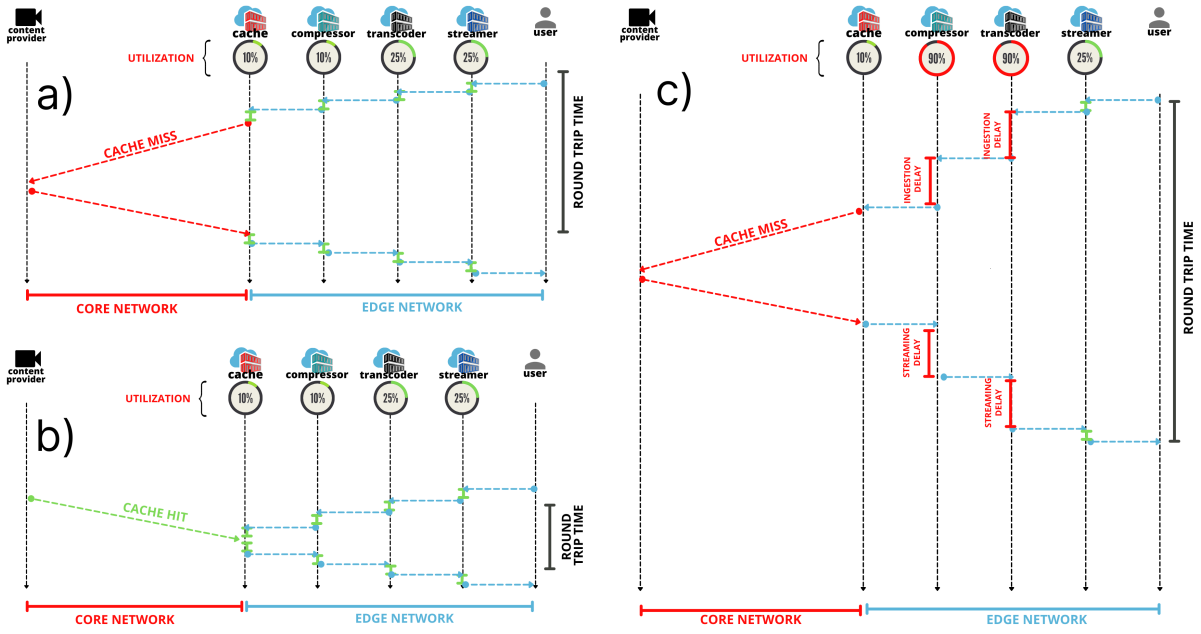

Figure 2.3. Some typical Round-Trip-Time augmenting factors in the case of video virtualized content delivery networks (vCDN). The cache VNF component acts a proxy for the origin servers located at the core of the network. Consequently, the total RTT changes drastically when MISS (a) or HIT (b) events occur. Moreover, low-utilization on the VNF chain components warrants faster processing times related to the ingestion and streaming tasks of each component. Higher utilization like the ones in (c) could also notably augment the overall RTT for a video delivery session.

Virtualized CDNs are commonly modeled following the NFV Management and orchestration (NFV-MANO) framework published by the ETSI standard group [62, 97, 253, 23]. Figure 2.2 shows

a schematic representation of this framework. In this standardized architecture, the orchestrator module is in charge of deploying the service function chains, i.e. the allocation of the set of virtual network functions in the virtual network. The orchestrator coordinates and delegates the VNF instantiation and lifecycle to the VNF manager module(s) and the virtualized infrastructure manager (VIM) is responsible for the dynamic scaling of the VNF modules' resource provision [97]. The resource scaling of the NFVI is commonly triggered when needed, for example, when traffic bursts occur. Such a resource scaling may influence the resource provision costs of the CDN, also called *hosting costs* [23].

In this thesis, we have focused in a particular emerging vCDN model, which is the cloud-hosted vCDN [1]. In this context, the cloud provider instantiates and controls the resource scaling of the network function virtualization infrastructure (NFVI), and offers to its clients a scalable NFVI in a pay-per-use basis. Routing between network hardware and between the virtual network is commonly managed also by the cloud provider in the case of cloud-hosted vCDN. However, a realistic model of a vCDN should always include several features for the nodes (which are the virtual machines that host the VNF modules) such as utilization, and VNF instantiation times. Consequently, these features need to be modeled if one seeks to create a realistic simulation environment that permits to study the optimization of QoE, QoS, and Costs in a vCDN (whether cloud-hosted or not). Finally, we argue that data-transportation (DT) costs may also vary as a function of the SFC deployment [253], for example if we consider a multi-cloud vCDN environment. Thus, DT costs may also be an important part of the operational costs of a vCDN.

A common design objective when designing vCDNs is the joint optimization of operational costs and QoE. Given the heterogeneous network structure of a vCDN and the complex features that affect both operational costs and QoE in these systems, the joint optimization objective mentioned is challenging to be solved. As explained previously, video delivery sessions in vCDNs are usually deployed as a composite chains of virtual network functions, what makes the problem of jointly optimizing costs and QoE in SFC Deployemt a complex problem to solve [257].

Table 2.1. Some recent remarkable works in the SFC Deployment Optimization Literature

Reference	Online	Approach	Main limitation
OPAC [110] / HPAC [111]	NO	Exact/Heuristic optimization for caching VNF placement	No composite SFC formation, Nor dynamic policy adaption
Yala *et al.* [290]	NO	Heuristic VNF resource allocation and placement	Polynomial time-complexity
Benkacem *et al.* [23]	YES	Linear Integer Programming (ILP) for optimal SFC network slicing	No instantiation-time modeling are (coarse-grain optimization)
Filelis-Papadopoulos *et al.* [68]	NO	Two-phase hybrid (heuristic/exact) cache placement optimization	Probabilistic modeling of cache-status
Yongzheng *et al.* [118]	YES	Randomized mixed integer solution for placement of wide-distributed SFCs	Polynomial time-complexity
Marotta *et al.* [156]	YES	Heuristic-based energy-efficient SFC deployment and routing	No ETSI-MANO [62] framework compliance
Khezri *et al.* [156]	YES	DRL-based reliability-aware VNF Placement	No RTT optimization considered
NFVDeep [285]	YES	DRL-based Optimization of SFC deployment	No utilization-dependent processing times modeled
DDQP [152]	YES	DRL-based reliability-aware optimization of SFC deployment	No VNF-instantiation time-penalty modeled
DDQN-VNFPA [200]	YES	DRL-based optimization of SFC deployment	Coarse-grain deployment policy adaption (15 min)
Santos *et al.* [224]	YES	DRL-based energy-efficient of SFC deployment	No round-trip-time optimization considered

Several exact optimization models, and heuristics have been proposed in recent years for solving

optimal SFC deployment. Ibn-Khedher *et al.* [110] defined a protocol for optimal VNF placement based on SDN traffic rules and an exact optimization algorithm. This work solves the optimal VNF placement, migration, and routing problem, modeling various system and network metrics as costs and user satisfaction. HPAC [111] is a heuristic proposed by the same authors to scale the solutions of [110] for bigger topologies based on the Gomory-Hu tree transformation of the network. Their call for future work includes the need for the dynamic triggering of adaptation like monitoring user demands, network loads, and other system parameters. Yala *et al.* [290] present a resource allocation and VNF placement optimization model for vCDN and a two-phase offline heuristic for solving it in polynomial time. This work models server availability through an empirical probabilistic model and optimizes this score alongside the VNF deployment costs. Their algorithm produces near-optimal solutions in minutes for a Network Function Virtualization Infrastructure (NFVI) deployed on a substrate network made of even 600 physical machines. They base the dimensioning criteria on extensive video streaming VNF QoE-aware benchmarking.

Authors in [23] keep track of resource utilization in the context of an optimization model for multi-cloud placement of VNF chains. Utilization statistics per node and network statistics per link are taken into account inside a simulation/optimization framework for VNF placement in vCDN in [68]. This offline algorithm can handle large-scale graph topologies being designed to run on a parallel-supercomputer environment. This work analyzes the effect of routing strategies on the results of the placement algorithm and performs better with a greedy max-bandwidth routing approach. The caching state of each cache-VNF is modeled with a probabilistic function in this work. Offline Optimization of Value Added Service (VAS) Chain deployment in vCDN is proposed in [51], where authors model an Integer Linear Programming (ILP) problem to optimize QoS and Provider Costs. This work models license costs for each VNF added in a new physical location. An online alternative is presented in [115], where authors model the cost of VNF instantiations when optimizing online VNF placement for vCDN. This model lacks to penalize the Roud Trip Time (RTT) of requests with the instantiation time of such VNFs. More scalable solutions for this problem are leveraged with heuristic-based approaches. For example, a randomized mixed integer solution is used to present an online VNF placement algorithm for geo-distributed VNF chain requests in [118]. This work optimizes different costs and the end-to-end delay providing near-to-optimal solutions in polynomial time.

In [155], Marotta *et al.* present a general-type VNF placement and SFC routing optimization model that minimizes power consumption taking into account that resource requirements are uncertain and fluctuate in virtual Evolved Packet Core scenarios. Such an algorithm is enhanced in a successive work [156] where authors improve the scalability of their solution by dividing the task into sub-problems and adopting various heuristics. Such an improvement permits solving high-scale VNF placement in less than a second, making such an algorithm suitable for online optimization. Remarkably, the congestion-induced delay has been modeled in this work. Ito *et al.* [112] instead, provide various models of the VNF placement problem where the objective is to warrant probabilistic failure recovery with minimum backup-required capacity. Authors in [112] model uncertainty in both failure events and virtual machine capacity.

Approaches based on Deep Reinforcement Learning have also been used to solve the SFC deployment problem. Other network-related problems like routing [213], and VNF forwarding graph embedding [210, 9, 292] have been solved with DRL techniques. Authors in [121] use the Deep Q-learning framework to implement a VNF placement algorithm which is aware of the server reliability. A policy learning algorithm is used for optimizing operational costs and network throughput through SFC Deployment in [285]. A fault-tolerant version of SFC Deployment is presented in [152], where authors use a Double Deep Q-network (DDQN) and propose different resource reservation schemes to balance the waste of resources and ensure service reliability. Authors in [200] assume to have accurate incoming traffic predictions in input and use a DDQN-based algorithm for choosing small-scale network sub-regions to optimize every 15 minutes. Such a work uses a threshold-based policy to optimize the number of fixed-dimensioned VNF instances to deploy or keep instantiated at each time interval. A Proximal Policy Optimization scheme is used in [224] to jointly minimize packet loss

and server energy consumption on a cellular network SFC deployment environment. The advantage of DRL approaches with respect to traditional optimization models is the constant time complexity reached after training. A well-designed DRL framework has the potential to achieve complex feature learning and near-to-optimal solutions even in unprecedented context situations [318]. Table 2.1 contains a summarized view of recent remarkable works focused in optimizing the SFC deployment or VNF placement problem from multiple points of view.

2.1.5 Open Challenge: Optimal SFC Deployment in Live-streaming vCDNs

In the previous paragraph we have explored common solutions to the SFC Deployment problem in virtualized CDNs. However, none of the previous mentioned works focused in live-video delivery. In this new context new challenging environment characteristics arise that make SFC Deployment a challenging task to all of the mentioned works. We now explain these characteristics, letting the description of the proposed solution to chapter 3 of this dissertation.

Online SFC Deployment implies taking fine-grained control decisions over the deployment scheme of SFCs when the system is running. For example, one could associate a particular SFC deployment for each incoming request to the system or keep the same SFC deployment for different requests but be able to change it whenever desired. In all cases, Online SFC Deployment implies that apadtation of the SFC policy can be done each time a new video delivery request comes into the system. *Offline* deployment instead takes one-time decisions over aggregations of incoming requests[73]. As we have mentioned, many works have proposed an offline optimization of SFC deployment for general-case vCDN scenarios. The effectiveness of offline optimization frameworks, however, relies on the estimation accuracy of the future environment's characteristics [67]. Such estimation may not be trivial in contexts where incoming traffic patterns may be unpredictable or when requests' characteristics might be heterogeneous, which is the case of Live-Streaming [318, 87, 285, 253]. In Live-Streaming scenarios in fact, the duration of streaming sessions is unknown to the system, and making predictions on the resource usage might be very difficult. Also, traffic patterns may turn unpredictable because of the skewness of the distribution of content popularity. Consequentially, if we seek to optimize SFC deployment for live-streaming vCDNs, we need to do it online.

Moreover, the characteristics of a live-streaming vCDN require to model an online optimization model taking into account:

- VNF-instantiation times,
- Content-delivery and content-ingestion resource usage,
- Utilization-dependant processing times,
- Fine-grained cache-status tracking,
- Operational costs composed of data-transportation costs and hosting costs,
- No *a priori* knowledge of session duration.

To the best of the authors knowledge, none of the above mentioned works took into account at the same time all these modeling factors. In chapter 3, we give a detailed exposition of the proposed solution for online optimizing SFC deployment in terms of QoE and operational costs in live-streaming vCDNs.

2.2 Het-graph modeling in Bioinformatics

Bioinformatics is another emergent research field that deals with heterogeneous networks. This section introduces gene regulatory networks and explains in more detail the networks that bioinformaticians analyze to infer transcriptional regulation of gene expression. It then describes how deep-learning-based high-throughput sequencing analysis is mostly done with the help of graph models. Finally,

we identify one research opportunity that, before this research, was not already addressed with het-graph modeling and deep learning: inferring transcriptional regulatory mechanisms between cis-regulatory elements and genes from temporal high-throughput sequencing datasets.

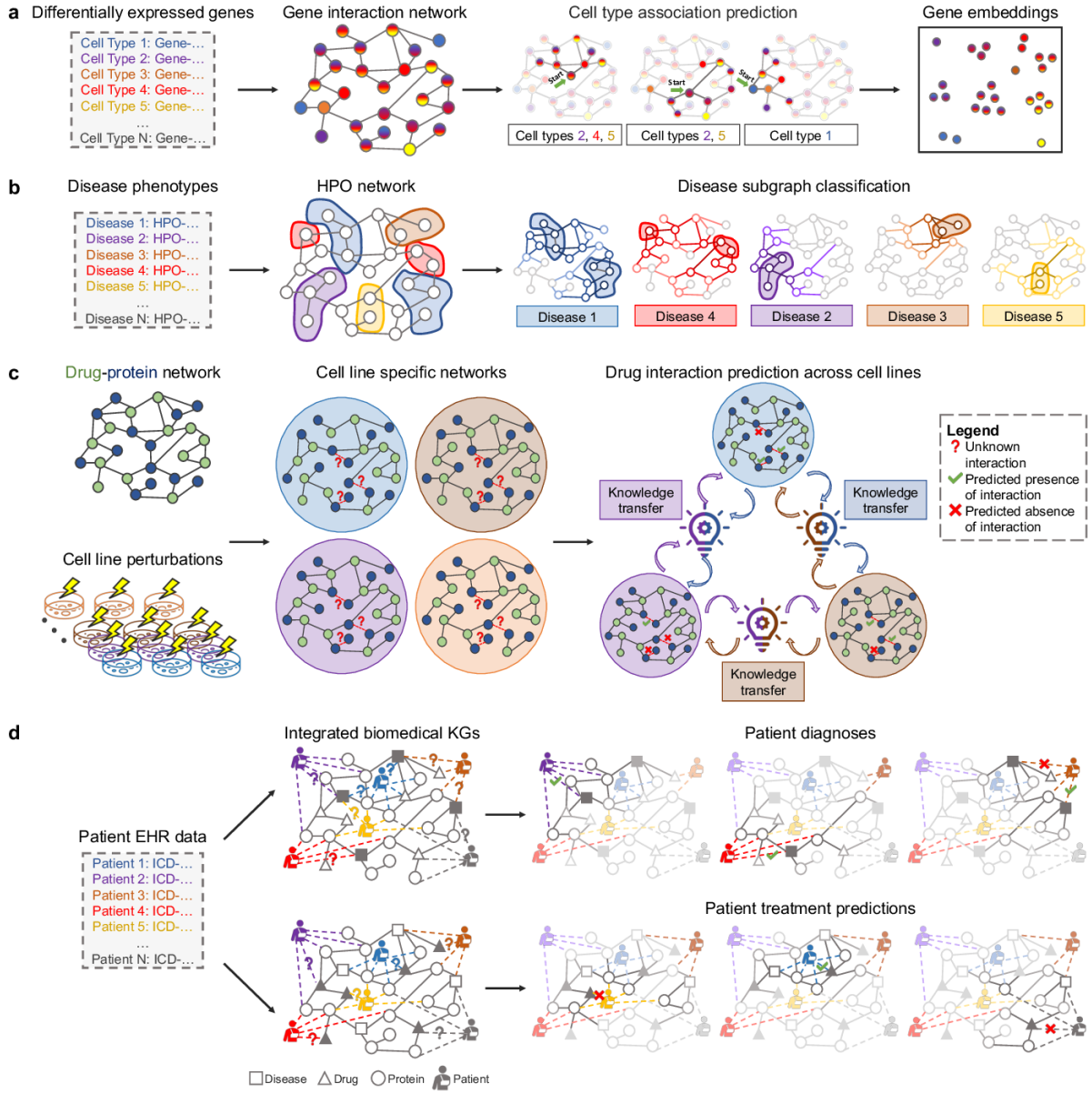

Figure 2.4. Some examples of graph representation learning in bioinformatics and biomedicine. (a) Incomplete gene-gene interaction networks. In these networks, nodes are genes, and their feature vectors are the expression values in multiple cells. The node embeddings could help predict gene expression values for every cell type in new genes. (b) A graph made of a portion of the Human Phenotype Ontology database. Researchers could use this graph to train a graph embedding algorithm that produces meaningful representations of diseases that could help investigate complex relations between their symptoms. (c) Node embeddings for a heterogeneous network made of cell line-specific drugs and proteins with multiple interaction types among these elements can lead to the discovery and prediction of unknown associations. (d) A biomedical knowledge graph. Nodes are patients, diseases, drugs, and proteins. Patient node embeddings could help to predict significant associations to diseases, treatments, etc. This figure was obtained from [166].

Bioinformatics focuses on managing and analyzing biological data through computer science, statistics, mathematics, and information engineering. [18] Bioinformatics has been receiving increasing attention from academia and industry in the last years. The Bioinformatics Market provides software tools for analyzing biological data, and the global bioinformatics market should reach

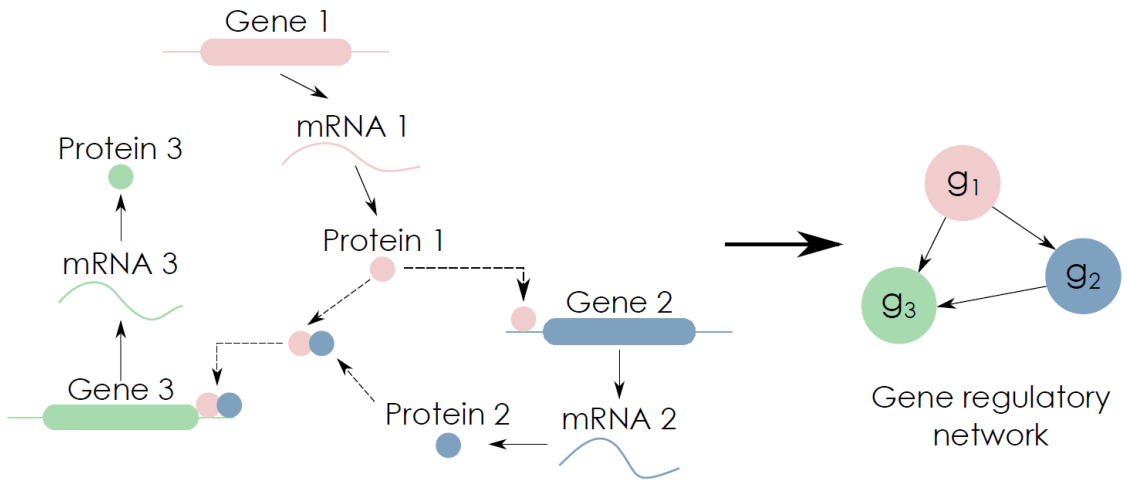

Figure 2.5. A widely used graph model of transcriptional gene regulatory networks (GRN). In this model, genes are related between them on a cause-and-effect basis. When some genes are expressed, they produce the proteins that serve as transcription factors for other genes to express themselves. Being that these cause-and-effect relationships could be more complex in reality, gene regulatory network models can be more complex and specialized, e. g. making use of het-graphs. This figure was obtained from [109].

USD 18.96 Billion by 2026. [214] On the other hand, many biological phenomena are modeled as graphs. [255] Examples are models of Gene Regulatory Networks [230, 222], Disease Infection Spread models,[177, 189, 63], models for Cellular Metabolic Networks, Protein Interaction Networks, among others [262, 70, 237, 199, 77]. Figure 2.4 shows some important problems in bioinformatics modeled as graphs and het-graphs, and the utility of performing graph-RL in these contexts. In this section, we will briefly review how het-graph models have been applied for gene regulatory networks, which consist of aggregations of molecules that interact with each other to drive the gene expression levels. [206, 219]

2.2.1 Transcriptional Gene Regulatory Networks

There are three primary gene expression regulation levels: transcriptional, post-transcriptional, and post-translational. Post-transcriptional and post-translational regulation of gene expression refers to the mechanisms that regulate the messenger RNA (mRNA) processing in the cell's cytoplasm. With transcriptional regulation instead, we refer to the biological mechanisms that influence the messenger RNA production at the cell nucleus. The messenger RNA is a copy of the genetic base-pair sequence produced by opening the DNA double-strand. This opening is done by the RNA polymerase (RNAP) enzyme, which concentration in the gene regions is regulated by specialized proteins called Transcription Factors (TF). We concentrate on transcriptional regulation, and we use the abbreviation GRN for transcriptional gene regulatory networks from now on. We refer the interested reader to [207] for a complete reference on gene expression regulation. Gene expression causes the cells of an organism to differentiate and acquire multiple functions. Gene regulatory interactions also govern the development of the structure of organisms' bodies, which is widely studied in evolutionary developmental biology. [109, 324]

2.2.2 A common network taxonomy for GRNs

In the last years, Next Generation Sequencing (NGS) technologies and, in particular, RNA sequencing (RNA-seq) [273] helped to measure the expression of genes on a genome-wide basis. As a consequence, multiple graph GRN models were recently proposed by academia. [222] A basic GRN model infers a particular cause-and-effect relationship between genes. A gene produces the transcription factor proteins that enable the expression of another gene. This expression regulatory mechanism can

also be a many-to-one, one-to-many, and many-to-many relationship in nature. In this basic model, GRNs are graphs where the genes are the nodes and the regulatory cause-and-effect relationships are the edges. The genes are associated with their gene expression level, a scalar quantity, or a vector containing various scalar measurements. When gene expression features consist of vectors, the scalar values of the vector usually correspond to different measurements, each one on a different cell sampled from a population. Thus, the values of the vector can be seen as the sampling values of a random variable, and researchers try to obtain knowledge about it's probability distribution characteristics. Moreover, different random variables could be found to be dependant between them, and the types of dependencies are multiple. In a probabilistic framework, these dependencies are mapped into edges in the simple GRN model depicted above usually indicating the presence of direct or indirect regulatory mechanisms. Thus, these kinds of GRN models are usually directed graphs. Given a graph model, not only probabilistic approaches can be set up to infer GRNs: regulatory relationships among genes can be given a hypothetical score by computing node-level statistics from available data. These methods are called data-driven models. Whether we use a probabilistic or a data-driven approach, GRN graphs usually turn to be weighted graphs. An example of these models is briefly schematized in Figure 2.5.

The most used model for GRN is a special kind of heterogeneous graph dubbed multi-omics network. In this model, even if a unique node type exists, multiple types of relationships among nodes may co-exist. Specifically, the previous simple GRN model is enriched by including more node-level features in this multi-layered model. A particular genomic measurement or experiment provides a new feaure type. These models create a graph where each *omics dataset defines a specific relationship type among the nodes. [39] A multi-omics network helps not only to infer causality in gene expression regulation between genes but also can help to detect similar functionalities between genes and gene ontology reconstruction. However, the transcription regulatory relationships between genes can be far more complex. Knowledge of all the gene expression mechanisms in the most simple organisms is still a challenge. The work of Huynh-Thu and Sanguinetti [109] mentions various modeling strategies that could help to model some punctual complexifications of these cause-and-effect relationships.

2.2.3 Cis-Regulatory Elements and GRNs inference

Transcription factors are responsible for the recruitment of the RNAP enzyme in non-coding DNA regions in the proximity of genes. These non-coding DNA regions are referred to as cis-regulatory elements (CRE). The CREs regulate the transcription of proximal genes, where proximity typically indicates an interval of $5e4$ base pairs. Additionally, multiple CREs can be responsible for the transcription regulation of one gene, and a unique set of CREs can regulate multiple genes. Finally, some structural modifications of the DNA, associated with long-term adaptation or cell-type differentiation, called epigenetic modifications, also regulate the transcription process. [6] Epigenetic modifications alter the functionality or accessibility of CREs. Consequently, the study of CREs and the epigenetic modifications associated produces a more detailed way of inferring gene regulatory networks. Associating CREs to the genes they regulate is also the key to deciphering temporal dynamic changes through development.

NGS experiments done over the same cell population are one of the most widely used methods to study the interaction between CREs and gene expression levels on a genome-wide basis. In single-cell RNA sequencing, each cell in the population is associated with a high-dimensional gene expression profile, where each gene has its expression score. On the other hand, ATAC sequencing characterizes the accessibility of non-coding regions for each cell in the population. Other CRE activity markers related for example to acetylation or methylation can also be obtained with NGS technologies. [184, 140]

To infer transcription regulatory associations between genes and CREs, researchers commonly study the covariance between two or more distributions of omics experiments among the same cell population: first, the distribution of gene expression profiles, and second the distributions of ATAC

or other CRE activity markers. [242, 143] For example, insights about developmental regulatory networks are typically gained by aligning single-cell RNA sequencing (scRNA-seq) and single-cell ATAC sequencing (scATAC-seq). [184, 140] Cell classification is another application of multi-view learning in various high-throughput experiments. [242, 143] The Seurat method [246], for example, makes different reduced dimension representations of single-cell gene expression and CREs data. This method uses canonical correlation analysis and graph representation learning to map multiple datasets inside a common latent space with reduced dimension and anchor cells across different modality datasets. Other methods rely on stronger prior knowledge about cell alignment to anchor multiple modalities of data. This is the case of the Multi-Omics Factor Analysis v2 (MOFA+), a statistical framework for the comprehensive and scalable integration of single-cell multi-modal data [11]. Multi-modal dataset alignment methods have also been developed based on state-of-the-art deep learning techniques. [307, 59] MAGAN [8], for example, successfully aligned scRNA-seq and scATAC-seq datasets using Generative Adversarial Networks (GAN). Nguyen *et al.* [184] and Xi [283] are comprehensive surveys on multi-modal or multi-view learning applied to multi-omics datasets.

Despite multi-network alignment, this thesis concentrates on heterogeneous graph models, also called multi-layer or multi-dimensional networks. An example of this kind of model is proposed by Jagtap *et al.* [114], who built a heterogeneous network made of genes and micro-RNAs to study regulatory relations among these elements. The node-level features, in this case, were made of high-dimensional vectors that represented gene and micro-RNA (miRNA) expression maps of hundreds of normal and tumor tissues. The relations that they modeled in this heterogeneous graph were the correlation between gene expression and miRNAs' expression patterns and other edge types that connected genes and miRNAs based on *a priori* biological knowledge, such as the positioning of the elements in the DNA chain. Their objective was to infer gene expression regulatory relations between miRNAs and genes. Heterogeneous data interaction was also modeled by Huang *et al.* [108] for predicting transcription factor interactions with their target genes. Wang [269] and Zhou [328] instead, modeled gene-disease interactions trough a het-graph model. Other recent problems solved through heterogeneous graph-based models include Gene Ontology Representation Learning [188], Gene prioritization for rare diseases [211] and drug repurposing [163].

2.2.4 Open Challenge: Co-clustering of temporal gene expression and CRE activity for GRN inference

Recently, a wide range of temporal high-throughput experiments has become available for both normal development, and disease-related investigations [76, 195, 311]. For example, the ENCODE consortium has provided comprehensive fetal developmental datasets for 12 tissues in mice from the E10.5 stage to the birth time P0 [76]. These datasets include both gene expression data and major transcriptional and epigenetic features. Some of the latter come from ATAC-seq experiments that measure chromatin accessibility, whole-genome shotgun bisulfite sequencing experiments measuring DNA methylation, and histone modifications [76]. Similar datasets, though in a lesser amount, exist to study aging (see a recent review by Pagiatakis *et al.* [195]). Cancer, instead, is among the diseases with the most considerable amount of available gene regulatory datasets (see a recent review by Zboril *et al.* [311]).

This recent growth in temporal datasets availability has awakened a question: can we use these temporal data for CRE-gene GRN inference? We took the answer to this question as a challenge in developmental studies. In particular, we seek to infer GRNs from temporal datasets containing gene expression and CRE activity markers in various phases of tissue-specific development. Many methods have been proposed to study genes and CREs separately through development in the last years. Recent literature in this field seeks to identify many-to-many regulation mechanisms through clustering each temporal dataset separately and then anchoring clusters using domain knowledge. Infinite Gaussian Process Mixture Model [158] and Convolutional Neural Networks [194] are among the most successful methods to find temporal clusters of either genes or CREs. Generally, these

kinds of high-throughput datasets are highly clusterable, and even conventional clustering methods like K-Means clustering can also generate satisfactory results when used for a single source dataset (*i.e.* only genes or only CREs). However, as CREs and genes are distinct entities on different genome coordinates, a concurrent study has been challenging. Specifically, combining or aligning the clusters across data sources is still a significant challenge.

However, the novel availability of high-throughput temporal datasets permits us to model a heterogeneous GRN graph where nodes are genes and CREs, both of them featured with vectors that contain measurements or experiments done in different cell developmental stages. In this novel model, genes would be featured by vectors containing gene expression time series, and the feature vectors of CRES would be the various activity markers' time series. We argue that CRE-gene temporal GRNs can be extracted from such a heterogeneous graph. The relationships defined among this new heterogeneous set of nodes should also be multiple: One edge type -defined between CREs and genes- should represent the regulation of gene expression. Another class of relationships -defined only between CREs- could map the similarities in the activity markers' profiles. Moreover, another relationship could map the gene expression pattern similarity, and would be defined only between genes. Lastly, an additional relation type could be created by looking at base-pair distances between CREs and genes. Considering this model, the co-clustering of CREs and genes by temporal data to find regulatory mechanisms would however be a challenging task due to two main reasons, apart of the computational issues:

1. Heterogeneous data integration: When studying the influence of CREs on temporal gene-expression patterns, the more CRE activity marker datasets we include in the model, the better results we can achieve. Consequentially the features of CREs and genes are often dimensionally different. Genes are characterized by one time series of gene-expression values, while CREs may have multiple time series as features, one for each activity marker dataset included in the experiment.

2. Non-graph formatted data: Taking into account the general baseline of biological domain knowledge, the unique relationship that connects genes and CREs on a network model is the base-pair distance in the genome. However, if we aim to infer many-to-many CRE-gene regulatory mechanisms, we should also include gene-gene and CRE-CRE associations based on the feature vectors. Usually, these associations are complete similarity metrics, like the inverse of the euclidean distance, an thus, need to be regularized to create a graph structure upon which machine learning models could learn to extract patterns. A correct static regularization mechanism could be non-trivial to implement. Instead, dynamic regularization models should be more appropriate, augmenting the complexity of the task.

Considering the previously stated characteristics for the task of co-clustering genes and CREs based on temporal features, we propose to face this challenge with the aid of deep learning techniques. In chapter 4 we expose in detail how we designed a heterogeneous-graph version of a state-of-the-art graph RL algorithm to face this challenge.

Chapter 3

Use Case 1: Online optimization of SFC Deployment on live-streaming virtualized CDN

As we have discussed in section 2.1, performance optimization is a key target in video delivery network systems. In this chapter, we focus on the particular case of Live-Video delivery, also referred to as *live-streaming*. In particular, this chapter exposes how we have addressed the first open challenge identified in the previous phase of this research. More specifically, we explain how we have engineered the first solution for online multi-objective optimization of SFC deployment in live-streaming virtualized content delivery networks using deep reinforcement learning.

This chapter starts with a high-level problem definition in section 3.1. Section 3.2 explains the modeling details of our proposed solution and the trace-driven simulations made with real-world data. In section 3.3 instead, we show the results of our simulation tests. Finally, in section 3.4 we discuss how and why our algorithm revealed substantial QoS/QoE performance improvements in terms of session acceptance ratio against the compared algorithms while keeping operational costs within proper bounds.

3.1 Problem definition

Consider a multi-cloud-hosted vCDN following the NFV-MANO framework [62, 97, 253, 23], and an elastic container-based virtualization of a CDN [97, 300]. We seek to *Online* optimize SFC deployment in a Live-Streaming vCDN, and we seek such optimization in terms of both QoS and operational costs. QoS is measured by the acceptance ratio as a function of the session startup delay. The operational costs, instead, are composed of hosting costs and data-transportation costs.

As explained in section 2.1, every time a user initiates a live-video session, the vCDN has to allocate the correspondent SFC into the NFV-I, taking into account the characteristics of the request in terms of desired bitrate, codec, and channel. We also fix a global constraint to limit the session startup delay that has to be respected in every SFC request. The optimal SFC deployment problem implies mapping every SFC request to the NFV-I maximizing the overall acceptance ratio of requests and minimizing at the same time the operational costs.

Operational costs depend on the resource provision of the system. The latter depends on the resource provision policy acted by the virtualized infrastructure manager and the SFC deployment policy performed by the orchestrator module. As explained in section 2.1, the virtualized infrastructure manager module is responsible for the resource scaling of VNF containers, and such resource scaling may influence the resource provision costs, also called *hosting costs*. On the other hand, the SFC deployment policy may also affect data transportation (DT) costs.

The model for the Online VNF-SFC deployment optimization on Live-Streaming vCDN must take into account, at the same time:

- VNF-instantiation times,
- Content-delivery and content-ingestion resource usage,
- Utilization-dependant processing times,
- Fine-grained cache-status tracking,
- Operational costs composed of data-transportation costs and hosting costs,
- No *a-priory* knowledge of session duration.

Moreover, the challenge is to jointly optimize QoS and operational costs with an Online DRL-based approach to learn robust SFC deployment policies.

Further, we must model bounded network resource availability to test network overload scenarios. We aimed to create and validate a *safe-exploration* framework that facilitates the assessment of market-entry conditions for new cloud-hosted Live-Streaming vCDN operators. The research described in this chapter can be seen as an upgrade proposal for the framework presented in [285], in that the optimization objective of SFC deployment that we pursue is the same: maximizing the QoS and minimizing the operational costs. The main difference concerning the work in [285] is that we have added necessary constraints to the network model that arise in the specific case of live-streaming delivery. However, adding these new constraints to the model makes it challenging for the DRL-based solution in [285] to learn good SFC deployment policies. So a new DRL framework to learn near-to-optimal policies in this new context is presented. In the rest of this chapter, we describe how we enhanced the algorithm used in [285]. We have added the usage of value-learning, experience replay, target networks, efficient state-space definition, and a reward shaping strategy that enhances the exploration to find a suitable DRL technique for the particular case of Live-Streaming in v-CDN scenarios.

3.2 Materials and Methods

3.2.1 Problem modelisation

We now rigorously model our SFC Deployment optimization problem. First of all, the vCDN is modeled as a het-graph. We then formulate a high-level optimization statement. Successively, our optimization problem's decision variables, penalty terms, and feasibility constraints are described. Finally, we formally define the optimization objective.

Network elements and parameters:

We model three-node categories or classes in the network infrastructure of a vCDN. The *content provider (CP) nodes*, denoted as N_{CP}, produce live-video streams that are routed through the SFC to reach the end-users. The VNF *hosting nodes*, N_H, are the cloud-hosted virtual machines that instantiate and interconnect container VNFs through each other to form the SFCs. Lastly, we consider nodes representing geographic clusters of clients, N_{UC}. Geographic client clusters are created in such a way that every client in the same geographic cluster is considered to have the same data-propagation delays with respect to the hosting nodes in N_H. Client cluster nodes will be referred to as *client nodes* from now on. Notice that different hosting nodes may be deployed on different cloud providers. We denote the set of all nodes of the vCDN substrate network as:

$$N = N_{CP} \cup N_H \cup N_{UC}$$

Notice that, having defined three distinct node types, we have modeled our live-streaming vCDN as a heterogeneous graph. We assume that each live-streaming session request r is always mapped to a VNF chain containing a Streamer, a Compressor, a Transcoder, and a Cache module[19, 97].

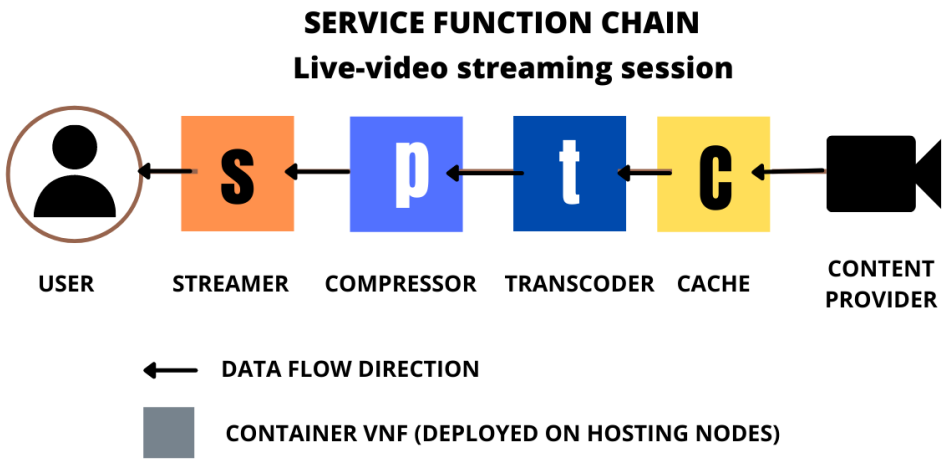

Figure 3.1. The assumed service function chain (SFC) composition for every Live-Video Streaming session request. We assume that every incoming session needs for a streamer, a compressor a transcoder and a cache virtual network function (VNF) modules. We assume container based virtualization of a virtualized Content Delivery Network (vCDN). Notice that a service function chain is a meta-path of the heterogeneous graph model of the vCDN. This fixed structural constraint for live-streaming VNF SFCs helps the shaping of dense reward policies to inject efficient exploration biases of the prohibitively large solution space.

In a live-streaming vCDN context, the caching module acts as a proxy that ingests video chunks from a Content Provider, stores them on memory, and sends them to the clients towards the rest of the SFC modules. Caching modules prevent origin server overloads, and improve session startup times which is a measure of QoE in the context of live-streaming. Compressors, instead, may help to vary video quality when requested. On the other hand, transcoding functionalities are necessary whenever the requested video codec is different from the original one. Finally, the streamer acts as a multiplexer for the end-users [97]. The order in which the VNF chain is composed is schematized in Figure 3.1. Notice that each service function chain is modeled as a meta-path of the heterogeneous graph that starts on a client node, then traverses four hosting nodes, and finishes on a content provider (CP) node. This fixed structural constraint for live-streaming VNF SFCs is crucial to injecting effective exploration biases into the proposed solution to the optimization problem based on deep reinforcement learning (DRL).

We will denote the set of VNF types considered in our model as K:

$$K = \{streamer, compressor, transcoder, cache\}$$

Any k-type VNF instantiated at a hosting node i will be denoted as $f_i^k, \forall k \in K, \forall i \in N_H$. We assume that every hosting node is able to instantiate a maximum of one k-type VNF. Note that, at any time, there might be multiple SFCs whose k-type module is assigned to a single hosting node i.

We define fixed-length time windows denoted as $t, \forall t \in \mathbb{N}$ which we call simulation time-steps following [285]. At each t, the VIM releases resources for timed-out sessions and processes the incoming session requests denoted as $R_t = \{r_1, r_2, ..., r_{|R_t|}\}$. It should be stressed that every r will request for an SFC composed of all the VNF types in K. We will denote the k-type VNF requested by r as $\hat{f}_r^k, \forall k \in K, r \in R_t$. Key notations for our vCDN SFC Deployment Problem are listed in table 3.1.

We now enlist all the network elements and parameters that are part of the proposed optimization problem:

- $N = N_{CP} \cup N_H \cup N_{UC}$ as the set of all **nodes** of the CDN network,
- $K = \{streamer, transcoder, compressor, cache\}$ is the set of **VNF types** considered in our model. Notice that K is the set of node classes in our HetNet-based model.

Table 3.1. Key notations for our vCDN SFC Deployment Problem

Notation	Meaning
N_{CP}	The set of content-provider nodes.
N_H	The set of VNF hosting nodes.
N_{UC}	The set of user clusters.
K	The set of VNF types (cache, compressor, transcoder, and streamer).
t	A fixed time-step in our simulation
R_t	The set of incoming SFC requests during time-step t
T	The max tolerable RTT for SFC any Live-Streaming request
b_r	The maximum scaled bitrate of r
p_r	The mean scaled session payload of r
$\hat{f}_r^k$	The k-type VNF requested by r
l_r	The *channel* or content provider requested by r
s_r	The *session workload* of r
$\gamma_{res,i}$	The unit cost of res resource in hosting node i
$d_{i,j}$	The data propagation delay between nodes i and j
f_i^k	The k-type VNF container instance in node i
$a_n^{t,k}$	1 if f_n^k is instantiated at the beginning of t
$x_{r,i}^k$	1 if $\hat{f}_r^k$ is assigned to node i, 0 otherwise
$z_{i,j,k}^r$	1 if the link between i and j is used to reach $\hat{f}_r^k$, 0 otherwise
$y_{l,i,k}$	1 if f_i^k is currently ingesting channel l, 0 otherwise
$c_{res,k,i}^t$	The res resource provision in f_i^k during t
$\sigma_{k,r,res}$	The client res resource demand of r in any k-type VNF instance.
$v_{k,res}$	The res resource demand for content ingestion in any k-type VNF instance.
$u_{res,k,i}$	The current res resource usage of f_i^k
$\mu_{res,k,i}$	The current res resource utilization of f_i^k
$\rho_{res,i,k}^r$	The processing time contribution of res resource in f_i^k for $\hat{f}_r^k$
ϕ_p	The normalization exponent for mean payload in the *session workload* formula
ϕ_b	The normalization exponent for maximum bitrate in the *session workload* formula
$o_{i,j}$	The unitary data-transportation cost for the link between nodes i and j.
v_r	1 if the SFC assigned to r respects the maximum tolerable RTT, 0 otherwise
$\rho_{res,k}^*$	The processing-time contribution of res in any k-type VNF assuming optimal utilization conditions.

- $L(N \times N)$, is the set of **links** between nodes in N, so that $(i,j) \in L, \forall i,j \in N$. L coincides with the set of edges in our heterogeneous graph model.
- $\mathbf{R} = \{CPU, Bandwidth, and Memory\}$, is the set of **resource types** for every VNF container,
- The **resource cost** matrix $\Gamma \in \mathbb{M}(|\mathbf{R}|, |N_H|)$, where $\gamma_{res,i}$ is the per-unit resource cost of resource res at node i,
- $D \in \mathbb{M}(|N|, |N|)$, is the **link delay** matrix so that variable $d_{i,j} \in \mathbb{R}^+$ represents the data propagation delay between the nodes i and j. We assume $d_{i,j} = 0$ for $i = j$. Notice that D is symmetric, and represents one feature space for the edges of our heterogeneous graph.
- $O \in \mathbb{M}(|N|, |N|)$, is the **data-transportation costs** matrix, where $o_{i,j} \in \mathbb{R}^+$ is the unitary data-transportation cost for the link that connects i and j. We assume $o_{i,j} = 0$ for $i = j$. Notice

that O is symmetric, and represents another feature space for the edges of our heterogeneous graph.

- $I_k, \forall k \in K$ are the parameters indicating the **instantiation times** of each k-type VNF.
- $R_t = \{r_1, r_2, ..., r_{|R_t|}\}$ is the set of incoming **session requests** set during time-step t. Every request r is characterized by $b_r \in [0, 1]$, the scaled *maximum* **bitrate**, $p_r \in [0, 1]$, the mean scaled **payload**, l_r, the **content provider** requested, and u_r, the **user cluster** from which r was originated,
- T is a fixed parameter indicating the maximum RTT threshold value for the incoming Live-Streaming requests.
- $P^* \in \mathbb{M}(|\mathbf{R}|, K)$ is the **optimal processing times** matrix, where $\rho^*_{res,k}$ is the processing-time contribution of the res resource in any k-type VNF assuming optimal utilization conditions,
- $\Psi \in \mathbb{M}(|\mathbf{R}|, K)$ is the **time degradation** matrix, where $\psi_{res,k}$ is the parameter representing the *degradation base* for the res resource in any k-type VNF,
- $c^t_{res,k,i}, \forall res \in \mathbf{R}, \forall k \in K, \forall i \in N_H$ indicates the res **resource capacity** of f^k_i VNF during t.

Optimization statement

Given a Live-Streaming vCDN constituted by the parameters enlisted in paragraph 3.2.1, we must decide the SFC deployment scheme for each incoming session request r, considering the penalties in the resulting RTT caused by the eventual instantiation of VNF containers, cache MISS events, and over-utilization of network resources. We must also consider that the entity of the vCDN operational costs is derived from our SFC deployments. We must deploy SFCs for every request to fine-grain maximize the resulting QoS and minimize Operational Costs.

Decision Variables

We propose a discrete optimization problem: For every incoming request $r \in R_t$, the decision variables in our optimization problem are the binary variables $x^k_{r,i}, \forall i \in N_H, \forall k \in K$ that equal 1 if $\hat{f}^k_r$ is assigned to f^k_i, and 0 otherwise.

Penalty Terms and Feasibility Constraints

We model two penalty terms and two feasibility constraints for our optimization problem. The first penalty term is the **Quality of Service penalty term** and is modeled as follows. The acceptance ratio during time-step t is computed as:

$$\chi^t_Q = \frac{\sum_{r \in R_t} \upsilon_r}{|R_t|} \tag{3.1}$$

where the binary variable υ_r indicates if the SFC assigned to r respects or not its maximum tolerable RTT, denoted by T_r:

$$\upsilon_r = \begin{cases} 1, & \text{if } RTT_r <= T_r \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

Notice that RTT_r is the round-trip-time of r and is computed as:

$$RTT_r = \sum_{i,j \in N} \sum_{k \in K} z^r_{i,j,k} \cdot d_{i,j} + \sum_{i \in N_H} \sum_{k \in K} x^k_{r,i} \cdot \{(1 - a^{t,k}_i) \cdot I_k + \rho^t_{i,k}\} \tag{3.3}$$

where:

- The binary variable $z^r_{i,j,k}$ equals 1 if the link between nodes i and j is used to reach $\hat{f}^k_r$, and 0 otherwise,
- The parameter $d_{i,j} \in \mathbb{R}^+$ represents the data-propagation delay between the nodes i and j. (We assume a unique path between every node i and j),
- The binary variable $x^k_{r,i}$ equals 1 if $\hat{f}^k_r$ is assigned to f^k_i, and 0 otherwise,
- The binary variable $a^{t,k}_i$ equals 1 if f^k_i is instantiated at the beginning of t, and 0 otherwise,
- I_k is a parameter indicating the instantiation time of any k-type VNF.
- $\rho^r_{i,k}$ is the processing time of r in f^k_i.

Notice that, by modeling RTT_r with (3.3), we include data-propagation delays, processing time delays, and VNF instantiation times when needed: We will include the delay to the content provider in such RTT only in the case of a cache MISS. In other words, if i is a CP node, then $z^r_{i,j,cache}$ will be 1 only if f^{cache}_j was not ingesting content from i at the time of receiving the assignation of $\hat{f}^{cache}_r$. On the other hand, whenever $\hat{f}^k_r$ is assigned to f^k_i, but f^k_i is not instantiated at the beginning of t, then the VIM will instantiate f^k_i, but adequate delay penalties are added to RTT_r, as shown in (3.3). Notice that we approximate the VNF instantiation states in the following manner: Any VNF that is not instantiated during t and receives a VNF request will start its own instantiation and will be supposed to finish such process at the beginning of $t+1$. From that moment on, unless the VNF has been turned off in the meantime because all its managed sessions are timed out, the VNF is considered ready to manage new incoming requests without any further instantiation time penalty.

Recall that we model three resource types for each VNF: CPU, Bandwidth, and Memory. We model the processing time of any request r in f^k_i as the sum of the processing times related to each of these three resources:

$$\rho^r_{i,k} = \rho^r_{cpu,i,k} + \rho^r_{mem,i,k} + \rho^r_{bw,i,k} \tag{3.4}$$

where $\rho^r_{res,i,k}, \forall res \in \{cpu, mem, bw\}$ are each of the resource processing time contributions for r in f^k_i, and each of such contributions is computed as:

$$\rho_{res,i,k} = \begin{cases} \rho^*_{res,k} \cdot \psi^{\mu_{res,k,i}-1}_{res,k}, & \text{if } \frac{\mu_{res,k,i}}{\alpha_{res,k}} > 1 \\ \rho^*_{res,k} & \text{otherwise} \end{cases} \tag{3.5}$$

where:

- the parameter $\rho^*_{res,k}$ is the processing-time contribution of res in any k-type VNF assuming optimal utilization conditions,
- $\psi_{res,k}$ is a parameter representing the *degradation base* for resource res in any k-type VNF,
- the variable $\mu_{res,k,i}$ is the res resource's utilization in f^k_i at the moment when $\hat{f}^k_r$ is assigned.
- $\alpha_{res,k}$ is the optimal utilization of resource res for every k -type VNF.

Note that (3.5) models utilization-dependent processing times. The resource utilization in any f^k_i, denoted as $\mu_{res,k,i}, \forall res \in \{cpu, mem, bw\}$ is computed as:

$$\mu_{res,k,i} = \begin{cases} \frac{u_{res,k,i}}{c^t_{res,k,i}}, & \text{if } c^t_{res,k,i} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.6}$$

where $u_{res,k,i}$ is the instantaneous res resource usage in f^k_i, and $c^t_{res,k,i}$ is the res resource capacity of f^k_i during t. The value of $c^t_{res,k,i}$ is fixed during an entire time-step t and depends on any dynamic

resource provisioning algorithm acted by the VIM. In this work we assume a bounded *greedy* resource provisioning policy as specified in paragraph 3.2.4. On the other hand, if we denote with $\tilde{R}_t$ the subset of R_t that contains the requests that have already been accepted at the current moment, we can compute $u_{res,k,i}$ as:

$$u_{res,k,i} = \hat{u}^t_{res,k,i} + \sum_{r \in \tilde{R}_t} x_{k,r,i} \cdot \sigma_{k,r,res} + \sum_{l \in N_{CP}} y^k_{l,i} \cdot \upsilon_{k,res} \tag{3.7}$$

where:

- The variable $\hat{u}^t_{res,k,i}$ indicates the res resource demand in f^k_i at the beginning of time-step t,
- The binary variable $x_{k,r,i}$ was already defined and it indicates if $\hat{f}^k_r$ is assigned to f^k_i,
- $\sigma_{k,r,res}$ is the res resource demand faced by any k-type VNF when serving r, and we call it the *client resource-demand*,
- The binary variable $y^k_{l,i}$ is 1 if f^k_i is currently ingesting content from content provider l, and 0 otherwise,
- The parameter $\upsilon_{k,res}$ models the res resource demand faced by any k-type VNF when ingesting content from any content provider.

Notice that, modeling resource usage with (3.7), we take into account not only the resource demand associated with the content transmission, but we also model the resource usage related to each content ingestion task the VNF is currently executing.

The res resource demand that any k-type VNF faces when serving a session request r is computed as:

$$\sigma_{k,r,res} = \sigma_{max,k,res} \cdot s_r \tag{3.8}$$

where $\sigma_{max,k,res}$ is a fixed parameter that indicates the maximum possible res resource consumption implied while serving any session request incoming to any k-type VNF. The variable $s_r \in [0,1]$ instead, is indicating the *session workload* of r, which depends on the specific characteristics of r. In particular, the *session workload* will depend on the *maximum bitrate* and the *mean payload per time-step* of r, denoted as b_r and p_r, respectively:

$$s_r = (p_r)^{\phi_p} \cdot (b_r)^{\phi_b} \tag{3.9}$$

In (3.9), the parameters $\phi_p, \phi_b \in [0,1]$ do not depend on r and are fixed normalization exponents that balance the contribution of b_r and p_r in s_r.

Recall that the binary variable v_r indicates if the SFC assigned to r respects or not its maximum tolerable RTT. Notice that we can approximate the total throughput served by the vCDN during t as:

$$\chi^t_T = \chi^t_Q \cdot \sum_{r \in R_t} s_r \tag{3.10}$$

The second penalty term is related to the **Operational Costs**, which is constituted by both the hosting costs and the Data-transportation costs.

We can compute the *Hosting Costs* for our vCDN during t as:

$$\chi^t_H = \chi^{t-1}_H - \tilde{\chi}^t_H + \sum_{i \in N_H} \sum_{k \in K} \sum_{res \in \mathbf{R}} \gamma_{res,i} \cdot c^t_{res,k,i} \tag{3.11}$$

where

- χ^{t-1}_H are the total Hosting Costs at the end of time-step $t-1$,

- $\tilde{\chi}_H^t$ are the hosting costs related to the timed-out sessions at the beginning of time-step t,
- $\mathbf{R}$ is the set of VNF resource types,
- $\gamma_{res,i}$ is the per-unit resource cost of resource res at node i.

Recall that $c^t_{res,k,i}$ is the res resource capacity at f_i^k during t. Notice that different nodes may have different per-unit resource costs as they may be instantiated in different cloud providers. Thus, modeling the hosting costs using (3.11), we have considered a possible multi-cloud vCDN deployment. Notice also that, using (3.11), we keep track of the current total hosting costs for our vCDN assuming that timed-out session resources are released at the end of each time-step.

We now model the Data-Transportation Costs. In our vCDN model, each hosting node instantiates a maximum of one VNF of each type. Consequently, all the SFCs that exploit the same link for transferring the same content between the same pair of VNFs will exploit a unique connection. Therefore, to realistically assess DT costs, we create the notion of session DT *amortized*-cost:

$$d^r_{cost} = \sum_{i,j \in N_H} \sum_{k \in K} \frac{p_r \cdot z^r_{i,j,k} \cdot o_{i,j}}{|R_r^{(i,j,k)}|} \tag{3.12}$$

where $o_{i,j}$ is a parameter indicating the unitary DT cost for the link between i and j, and $R_r^{(i,j,k)}$ is the set of SFCs that are using the link between i and j to transmit to f_j^k the content related to the same CP requested by r. Notice that DT costs for r are proportional to the mean payload p_r. Recall that $z^r_{i,j,k}$ indicates if the link between i and j is used to reach $\hat{f}_r^k$. According to (3.12), we compute the session DT cost for any session request r in the following manner: For each link on our vCDN, we first compute the whole DT cost among such a link (the numerator of 3.12). We then compute the number of concurrent sessions that are using such a link for transferring the same content requested by r. Lastly, we compute the ratio between these quantities and sum such ratios for every hop in the SFC of r to obtain the whole session amortized DT cost. The total amortized DT costs during t are then computed as:

$$\chi_D^t = \chi_D^{t-1} - \tilde{\chi}_D^t + \sum_{r \in R_t} v_r \cdot d^r_{cost} \tag{3.13}$$

where

- χ_D^{t-1} are the total DT costs at the end of the $t-1$ time-step,
- $\tilde{\chi}_D^t$ are the total DT costs regarding the timed-out sessions at the beginning of time-step t,
- d^r_{cost} is the session DT cost for r computed with (3.12). Recall that v_r indicates if r was accepted or not based on its resultant RTT.

On the other hand, the first constraint is the **VNF assignation constraint**: For any live-streaming request r, every k-type VNF request $\hat{f}_r^k$ must be assigned to one and only one node in N_H. We express such a constraint follows:

$$\sum_{i \in N_H} x^k_{r,i} = 1, \forall r \in R_t, \forall k \in K, \tag{3.14}$$

Finally, the second constraint is the **minimum service** constraint. For any time-step t, the acceptance ratio must be greater or equals than 0.5. We express such a constraint as:

$$\chi_Q^t \geq 0.5, \forall t \in \mathbb{N} \tag{3.15}$$

One could optimize operational costs by discarding a significant percentage of the incoming requests instead of serving them. The fewer requests are served, the less the resource consumption entity and the hosting costs will be. Also, data transfer costs are reduced when less traffic is generated due to the rejection of live-streaming requests. The constraint in (3.15) is created to avoid such naive solutions to our optimization problem.

Optimization Objective

We model an online multi-objective SFC deployment optimization: At each simulation time-step t, we measure the accomplishment of three objectives:

- Our first goal is to maximize the network throughput as defined in (3.10), and we express such objective as $\max(\chi_T^t)$,
- Our second goal is to minimize the hosting costs as defined in (3.11), and we express such objective as $\min(\chi_H^t)$,
- Out third goal is to minimize the DT costs as defined in (3.13), and such objective can be expressed as $\min(\chi_D^t)$.

We tackle such a multi-objective optimization goal with a weighted-sum that leads to a single objective function:

$$\max(w_T \cdot \chi_T^t - w_D \cdot \chi_D^t - w_H \cdot \chi_H^t) \tag{3.16}$$

where w_T, w_H, and w_D are parametric weights for the network throughput, hosting costs, and data transfer costs, respectively. We measure the objective function at each optimization time-step t. However, decisions are taken by our algorithm at each single VNF assignation step, because we seek to create an online optimization algorithm. Our problem can be seen as a best path-identification problem inside a heterogeneous network-based model containing complex constraints. To face the challenge of optimizing QoS and reducing Costs in the long run through single VNF assignation decisions, we have chosen to create a DRL agent with carefully designed inductive biases. This proposed DRL algorithm enhances the effectiveness of the agent exploration of the solution space, as we will describe in the next section.

3.2.2 Proposed Solution: Deep Reinforcement Learning with Enhanced-Exploration Biases

As we have already said in chapter 1, any RL framework is composed of an optimization objective, a reward policy, an environment, and an agent. In RL scenarios, a Markov decision process (MDP) is modeled, where the environment conditions are the nodes of a Markov chain (MC) and are generally referred to as *state-space* samples. The agent iteratively observes the state-space and chooses actions from the *action-space* to interact with the environment. Each action is corresponded by a *reward* signal and leads to the *transition* to another node in the Markov Chain, i.e. to another point in the *state-space*. Reward signals are generated by an *action-reward* mechanism that drives learning towards optimal action *policies* under dynamic environment conditions.

We propose a DRL-based framework to solve our Online SFC Deployment problem. To do that, we first need to embed our optimization problem in a MDP. We then need to create an *action reward mechanism* that drives the agent to learn optimal SFC Deployment policies, and finally, we need to specify the DRL algorithm we will use for solving the problem. The transition between states of the MDP will be indexed by $\tau, \forall \tau \in \mathbb{N}$ in the rest of this chapter.

Network model-based State-Space Design

Following [285, 118, 200, 201, 202], to reduce the prohibitively large entity of the SFC deployment action space, we propose to serialize the SFC Deployment process into single-VNF assignation actions. In other words, our agent interacts with the environment each time a particular VNF request, $\hat{f}_r^k$, of a particular SFC r, has to be assigned to a particular VNF instance, f_i^k of some hosting node i in the vCDN. Consequently, the actions of our agent, denoted by a_τ are the single-VNF assignation decisions for each VNF request of a SFC.

Before taking any action, however, the agent observes the environment's conditions. In practical terms, a state-space vector sample is given in input to the agent to obtain a VNF assignation action in output. Any DRL agent takes into account the optimization objective and the pre-defined maximization time horizon to learn the best policies. With this aim, DRL agents use deep ANN modules to learn to abstract the important features of the environment from state-space vectors. For this reason, it is important, and in some cases necessary, to reduce the most the abstraction effort that the agent is meant to do to learn these features. In this work, we use some knowledge of the het-graph model of the vCDN to design proper state-space vector representations that make it easier to the agent to learn the essential features of the state space. The design of the state-space representation is one of the main contributions of our work. Remarkably, at each transition, we propose to embed the environment conditions onto a vector s_τ that contains a snapshot of some essential network-state conditions and the current incoming request conditions. In particular, s_τ will be formed by the concatenation of three vectors:

$$s_\tau = (\hat{R}, \hat{I}, \hat{U}) \tag{3.17}$$

where:

- $\hat{R} = (l_r, u_r, s_r, \hat{f}_r^-)$ is called the request vector, and contains information about the VNF request under assignment. In particular, $\hat{R}$ codifies the requested CP, l_r, the client cluster from which the request was originated, u_r, the request's session workload, s_r, and the number of pending VNFs to complete the deployment of the current SFC, $\hat{f}_r^-$. Notice that $\hat{f}_r^- \in \{0, 1, 2, 3, 4\}$ and that $\hat{f}_r^-$ goes each time from 4 to 0 as our agent performs the assignation actions regarding r. Note that the first three components of $\hat{R}$ are invariable for the whole set of VNF requests regarding the SFC of r. Notice also that, using a *one-hot* encoding for l_r, u_r and $\hat{f}_r^-$, the dimension of $\hat{R}$ is $|\hat{R}| = |N_{UC}| + |N_{CP}| + |K| + 1$,
- $\hat{I} = (\hat{i}_1, \hat{i}_2, ..., \hat{i}_{|N_H|})$ is a binary vector where $\hat{i}_j$ equals 1 if f_j^k is ingesting the CP requested by r and 0 otherwise,
- $\hat{U} = (\hat{u}_1, \hat{u}_2, ..., \hat{u}_{|N_H|})$, where $\hat{u}_i$ is the utilization value of the resource that is currently the most utilized resource in f_k^i.

Notice that the action space serialization couples with the state space design. When the agent is asked to decide the hosting node that is meant to serve a current k-type VNF request, a VNF-specific map of the network status is given in input. This map hides unnecessary characteristics of the vCDN. More specifically, $\hat{I}$ contains information about the ingesting state of the k-type VNFs only, without considering other types of VNFs. Moreover, the ingesting state information is also simplified, because we focus only in the specific content provider requested in $\hat{R}$. We were able to design these light representations of the action and state spaces given the model-based information that we can hypothesize to have when the vCDN implementation accomplishes with the NFV-MANO framework. We will explain this idea in the next paragraph.

Dense Reward-Shaping Scheme

When designing the action-reward scheme, we take extreme care in giving the right entity to the penalty of resource over-utilization, as it seriously affects QoS. We also include a cost-related penalty to our reward function to jointly optimize QoS and Operational Costs. Recall that the actions taken by our agent are the single-VNF assignation decisions for each VNF request of a SFC. At each iteration τ, our agent observes the state information s_τ, takes an action a_τ, and receives a correspondent reward $r(s_\tau, a_\tau)$ computed as:

$$r(s_\tau, a_\tau) = w_Q \cdot r_{QoS}(s_\tau, a_\tau) - \nu \cdot (w_D \cdot \chi_D^t + w_H \cdot \chi_H^t) \tag{3.18}$$

where:

- The parameters w_Q, w_D, and w_H are weights for the reward contributions relating to the QoS, the DT costs, and the Hosting Costs, respectively,
- $r_{QoS}(s_\tau, a_\tau)$ is the reward contribution that is related to the QoS optimization objective. This reward has been carefully designed to inject an exploration bias in the agent to explore the action space effectively, as we will describe.
- ν is a binary variable that equals 1 if action a corresponds to the last assignation step of the last session request arrived in the current simulation time-step t, and 0 otherwise.

Recall that χ_D^t and χ_H^t are the total DT costs and hosting costs of our vCDN at the end of the simulation time-step t. Using (3.18), we subtract a penalty proportional to the current whole hosting and DT costs in the vCDN only at the last transition of each simulator time-step, i.e., when we assign the last VNF of the last SFC in R_t. Such a sparse cost penalty was also proposed in [285].

When modeling the QoS-related contribution of the reward instead, we propose the usage of an *inner delay-penalty function*, denoted as $d(t)$.

$$D : t \to \mathbb{R}^+$$

In practice, $d(t)$ will be continuous and non-increasing. We design $d(t)$ in such a way that $d(t) < 0, \forall t > T$. Recall that T is a fixed parameter indicating the maximum RTT threshold value for the incoming Live-Streaming requests. We specify the inner delay-penalty function used in our simulations in paragraph 3.2.4

Whenever our agent performs an assignation action a, for a VNF request $\hat{f}_r^k$ in r, we compute the generated contribution to the RTT of r. In particular, we compute the processing time of r in the assigned VNF, eventual instantiation times, and the transmission delay to the chosen node. We sum such RTT contribution at each assignation step to form the current partial RTT, which we denote as t_r^a. The QoS-related part of the reward assigned to a is then computed as:

$$r_{QoS}(a) = \begin{cases} d(t_r^a) \cdot 2^{-\hat{f}_r^-}, & \text{if } \hat{f}_r^- > 0 \text{ and } d(t_r^a) > 0 \\ d(t_r^a) \cdot 2^{\hat{f}_r^-}, & \text{if } \hat{f}_r^- > 0 \text{ and } d(t_r^a) < 0 \\ 1, & \text{if } \hat{f}_r^- = 0 \text{ and } d(t_r^a) > 0 \\ 0, & \text{if } \hat{f}_r^- = 0 \text{ and } d(t_r^a) < 0 \end{cases} \tag{3.19}$$

If we look at the first line of (3.19), we realize that a positive reward is given for every assignment that results in a non-prohibitive partial RTT. Moreover, **such a positive reward is inversely proportional to $\hat{f}_r^-$ (the number of pending assignations for the complete deployment of the SFC of r).** Notice that, since t_r^a is cumulative, we give larger rewards to the latter assignation actions of an SFC, as it is more difficult to avoid surpassing the RTT limit at the end of the SFC deployment with respect to the beginning.

The second line in (3.19) shows that a negative reward is given to the agent whenever t_r^a exceeds T. **Further, such a negative reward worsens proportionally to the prematureness of the assignation action that caused t_r^a to surpass T.** Such a worsening makes sense because bad assignation actions are easier to occur at the end of the SFC assignation process with respect to the beginning.

Finally, the third and fourth lines in (3.19) correspond to the case when we the agent performs the last assignation action of an SFC. The third line indicates that the QoS related reward is equal to 1 whenever a complete SFC request r is deployed, i.e., when every $\hat{f}_r^k$ in the SFC of r has been assigned without exceeding the RTT limit T, and the last line tells us that the reward will be 0 whenever the last assignation action incurs in a non-acceptable RTT for r.

This reward scheme is the main contribution of our work. According to the MDP embedding proposed 3.2.2, the majority of actions taken by our agent are given a non-zero reward. **Such a dense reward shaping improves the training convergence to optimal policies.**

Notice also that, in contrast to [285], **our reward mechanism penalizes bad assignments in contrast to ignoring them. We also claim that such an inclusion enhances the agent's exploration in the action space** and reduces the possibility of converging to local optima.

DRL Algorithm

Our DRL agent is based on the Double Deep Q Learning algorithm [94] with the incorporation of the dueling architectures [274]. We now briefly describe the theoretical basis of this algorithm.

To overcome the traditional Q-learning algorithm's limitations, Mnih *et al.* [171] proposed the usage of a Deep Artificial Neural Network (ANN) approximators of the Q-value function. To evict convergence to local-optima, they proposed to use an ϵ-greedy policy where actions are sampled from the ANN with probability $1-\epsilon$ and from a random distribution with probability ϵ, where ϵ decays slowly at each MDP transition during training. They also used the *Experience Replay* (ER) mechanism: a data structure $\mathcal{D}$ keeps $(s_\tau, a_\tau, r_\tau, s_{\tau+1})$ transitions for sampling uncorrelated training data and improve learning stability. ER mitigates the high correlation presented in sequences of observations during online learning. Moreover, authors in [172] implemented two neural network approximators for (1.9), the Q-network and the Target Q-network, indicated by $Q(s,a,\theta)$ and $Q(s,a,\theta^-)$, respectively. In [172], the target network is updated only periodically to reduce the variance of the target values and further stabilize learning with respect to [171]. Authors in [172] use stochastic gradient descent to minimize the following loss function:

$$\mathcal{L}(\theta) = E_{(s_\tau,a_\tau,r_\tau,s_{\tau+1})\sim U(\mathcal{D})}\{[r + \gamma \max_a Q(s_{\tau+1}, a, \theta^-) - Q(s_\tau, a_\tau; \theta)]^2\} \tag{3.20}$$

where minimization of (3.20) is done with respect to the parameters of $Q(s,a,\theta)$. Van Hasselt *et al.* [94] applied the concepts of Double Q-Learning [93] on large-scale function approximators. They replaced the target value in (3.20) with a more sophisticated target value:

$$\mathcal{L}(\theta) = E_{(s_\tau,a_\tau,r_\tau,s_{\tau+1})\sim U(\mathcal{D})}\{[r_\tau + \gamma Q(s_{\tau+1}, \operatorname*{argmax}_a Q(s_{\tau+1}, a; \theta), \theta^-) - Q(s_\tau, a_\tau; \theta)]^2\} \tag{3.21}$$

Doing such a replacement, authors in [94] avoided over-estimations of the Q-values which characterized (3.20). This technique is called Double Deep Q-Learning (DDQN), and it also helps to decorrelate the noise introduced by θ, from the noise of θ^-. Notice that θ are the parameters that approximate the function used to choose the best actions, while θ^- are the parameters of the approximator used to evaluate the choices. Such a differentiation in the *learning* and *acting* policies is also called *off-policy* learning.

Instead, Wang *et al.* [274] proposed a change in the architecture of the ANN approximator of the Q-function: they used a decomposition of the action value function in the sum of two other functions: the *action-advantage function* and the *state-value function*:

$$Q_\pi(s,a) = V_\pi(s) + A_\pi(a) \tag{3.22}$$

Authors in [274] proposed a two-stream architecture for an ANN approximator, where one stream approximated A_π and the other approximated V_π. They integrate such contributions at the final layer of the ANN Q_π using:

$$Q(s,a;\theta_1,\theta_2,\theta_3) = V(s;\theta_1,\theta_3) + (A(s,a;\theta_1,\theta_2) - \frac{1}{|\mathcal{A}|}\sum_{a'} A(s,a';\theta_1,\theta_2)) \tag{3.23}$$

where θ_1 are the parameters of the first layers of the ANN approximator, while θ_2 and θ_3 are the parameters encoding the action-advantage and the state-value heads, respectively. This architectural innovation works as an attention mechanism for states where actions have more relevance with respect to other states and is known as *Dueling DQN*. Dueling architectures have the ability to generalize learning in the presence of many similar-valued actions.

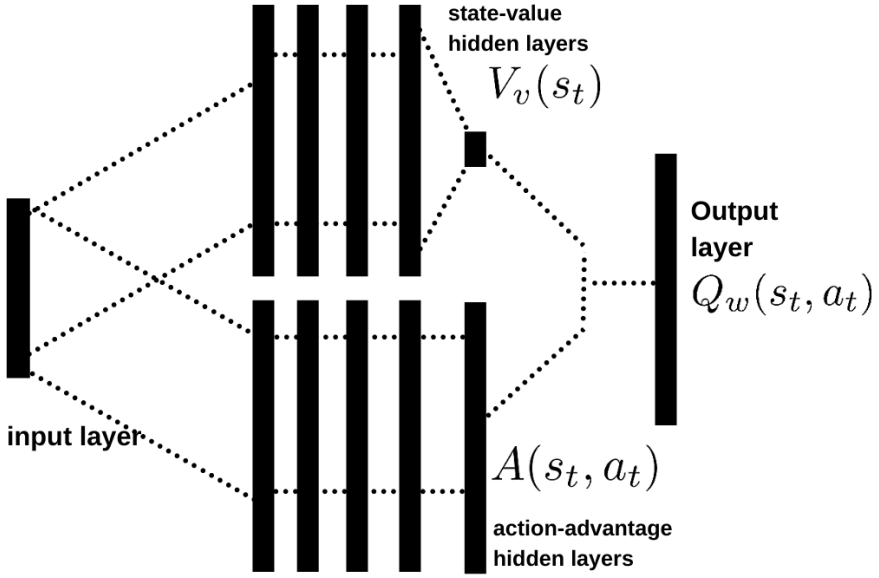

Figure 3.2. Dueling-architectured DDQN topology for our SFC Deployment agent: A two-stream deep neural network. One stream approximates the state-value function, and the other approximates the action advantage function. These values are combined to get the state-action value estimation in the output layer. The inputs are instead the embedding of the state space samples.

For our SFC Deployment problem, we propose the usage of the DDQN algorithm [94] where the ANN approximator of the Q-value function uses the dueling mechanism as in [274]. Each layer of our Q-value function approximator is a fully connected layer. Consequently, it can be classified as a multilayer Perceptron (MLP) even if it has a two-stream architecture. Even if we approximate $A_\pi(a)$ and $V_\pi(s)$ with two streams, the final output layer of our ANN approximates the Q-value for each action using the combination of $A_\pi(a)$ and $V_\pi(s)$ according to (3.23). The input neurons receive the state-space vectors s_τ specified in paragraph 3.2.2. Figure 3.2 schematizes the proposed topology for our ANN. The parameters of our model are detailed instead in table 3.2.

Table 3.2. Deep ANN Assigner topology Parameters

Parameter	Value
Action-advantage hidden layers	2
State-value hidden layers	2
hidden layers dimension	128
Input layer dimension	$2 \cdot \lvert N_H \rvert + (\lvert N_{UC} \rvert + \lvert N_{CP} \rvert + \lvert K \rvert + 1)$
Output layer dimension	$\lvert N_H \rvert$
Activation function between hidden layers	ReLU

We index the training episodes with $e \in [0, 1, ..., M]$, where M is a fixed training hyper-parameter. We assume that an episode ends when all the requests of a fixed number of simulation time-steps N_{ep} have been processed. Notice that each simulation time-step t may have a different number of incoming requests, $|R_t|$, and that every incoming request r will be mapped to an SFC of length $|K|$, which coincides with the number of MDP transitions on each SFC deployment process. Consequently, the number of transitions in an episode e will be then given by

$$N^e = \sum_{t \in [t_0^e, t_f^e]} |K| \cdot |R_t| \tag{3.24}$$

where $t_0^e = t \cdot N_{ep}$ and $t_f^e = t \cdot (N_{ep} + 1)$ are the initial and final simulation timesteps of episode e,

respectively. (Recall that $t \in \mathbb{N}$)

To improve training performance and avoid convergence to local optima, we use the ϵ-greedy mechanism. We introduce a high number of randomly chosen actions at the beginning of our training phase and progressively diminish the probability of taking such random actions. Such randomness should help to reduce the bias in the initialization of the ANN approximator parameters. In order to gradually lower the number of random moves as our agent learns the optimal policy, our ϵ-greedy policy is characterized by an exponentially decaying ϵ as:

$$\epsilon(\tau) = \epsilon_{final} + (\epsilon_0 - \epsilon_{final}) \cdot e^{\frac{-\tau}{\epsilon_{decay}}}, \forall \tau \in \mathbb{N}^+ \tag{3.25}$$

where we define ϵ_0, ϵ_{final}, and ϵ_{decay} as fixed hyper-parameters such that

$$\epsilon_{decay} >> 1 \geq \epsilon_0 >> \epsilon_{final}$$

Notice that $\epsilon(0) = \epsilon_0$ and

$$\lim_{\tau \to +\infty} \epsilon(\tau) = \epsilon_{final}$$

We call our algorithm Enhanced-Exploration Dense-Reward Duelling DDQN (*E2-D4QN*) SFC Deployment. Algorithm 1 describes the training procedure of our E2-D4QN DRL agent. We call *learning network* the ANN approximator used to choose actions. In lines 1 to 3, we initialize the replay memory, the parameters of the first layers (θ_1), the action-advantage head (θ_2), and the state-value head (θ_3) of the ANN approximator. We then initialize the target network with the same parameter values of the learning network. We train our agent for M epochs, each of which will contain N_e MDP transitions. In lines 6-10 we set an *ending episode signal* τ_{end}. We need such a signal because, when the final state of an episode has been reached, the loss should be computed with respect to the pure reward of the last action taken, by definition of $Q(s, a)$. At each training iteration, our agent observes the environment conditions, takes an action using the ϵ-greedy mechanism, obtains a correspondent reward, and transits to another state (lines 11-14). Our agent stores the transition in the replay buffer and then randomly samples a batch of stored transitions to run the stochastic gradient descent on the loss function in (3.21) (lines 14-25). Notice that the target network will only be updated with the parameter values of the learning value each U iterations to increase training stability, where U is a fixed hyper-parameter. The complete list of the training hyper-parameters used for training is enlisted in paragraph 3.2.4.

3.2.3 Experiment Specifications

Network Topology

We used a real-world dataset to construct a trace-driven simulation for our experiment. We consider the topology of the proprietary CDN of an Italian Video Delivery operator in our experiments. Such an operator delivers Live video from content providers distributed around the globe to clients located in the Italian territory. This operator's network consists of 41 CP nodes, 16 hosting nodes, and 4 client cluster nodes. The hosting nodes and the client clusters are distributed in the Italian territory, while CP nodes are distributed worldwide. Each client cluster emits approximately 1×10^4 Live-Video requests per minute. The operator gave us access to the access log files concerning service from July 25th to July 29th, 2017.

Simulation Parameters

We took data from the first four days for training our SFC Deployment agent and used the last day's trace for evaluation purposes. Given a fixed simulation time-step interval of 15 seconds and a fixed number of $N^e = 80$ time-steps per episode, we trained our agent for 576 episodes, which correspond to 2 runs of the 4 -day training trace. At any moment, the vCDN conditions are composed by

Algorithm 1 E2-D4QN

Initialize $\mathcal{D}$
Initialize θ_1,θ_2, and θ_3 randomly
Initialize θ_1^-,θ_2^-, and θ_3^- with the values of θ_1,θ_2, and θ_3, respectively
for *episode* $e \in \{1, 2, ..., M\}$ **do**
  while $\tau \leq N^e$ **do**
    if $\tau = N^e$ **then**
      $\tau_{end} \leftarrow True$
    else
      $\tau_{end} \leftarrow False$
    end if
    Observe state s_τ from simulator.
    Update ϵ using (3.25).
    Sample a random assignation a_t action with probability ϵ
    or $a_\tau \leftarrow \underset{a}{\text{argmax}} Q(s_\tau, a; \Theta)$ with probability $1 - \epsilon$.
    Obtain the reward r_τ using (3.18),
    and the next state $s_{\tau+1}$ from the environment.
    Store transition tuple $(s_\tau, a_\tau, r_\tau, s_{\tau+1}, \tau_{end})$ in $\mathcal{D}$.
    Sample a batch of transition tuples $\mathcal{T}$ from $\mathcal{D}$.
    for all $(s_j, a_j, r_j, s_{j+1}, \tau_{end}) \in \mathcal{T}$ **do**
      if $\tau_{end} = True$ **then**
        $y_j \leftarrow r_j$
      else
        $y_j \leftarrow r + \gamma Q(s_{j+1}, \underset{a}{\text{argmax}} Q(s_{j+1}, a; \theta), \theta^-)$
      end if
      Compute the temporal difference error $\mathcal{L}(\theta)$ using (3.21).
      Compute the loss gradient $\nabla\mathcal{L}(\theta)$.
      $\Theta \leftarrow \Theta - l_r \cdot \nabla\mathcal{L}(\theta)$
      Update $\Theta^- \leftarrow \Theta$ only every U steps.
    end for
  end while
end for

the VNF instantiation states, the caching VNF memory states, the container resource provision, utilization, etc. Notice that the N^e and the discount-factor hyper- parameters' values determine the time horizon over which the cumulative discounted reward is maximized. Given this particular parameter setting, the optimization time horizon is 20 minutes. This interval corresponds to the time-length of the simulation time-step interval -i.e. 15 seconds- multiplied by N^e.

In the test phase of every algorithm, we should fix the initial network conditions to a proper value that reduces the evaluation bias. Setting the initial network conditions of any algorithm to be those encountered at the end of its training cycle might bias its evaluation. We want to evaluate every agent's capacity to recover the steady state from general environment conditions. Such an evaluation needs initial conditions to be different with respect to the steady-state achieved during training. In every experiment we did, we set the initial vCDN conditions as those registered at the end of the fourth day when considering a greedy SFC deployment policy. We fix the QoS, Hosting costs, and DT-cost weight parameters in (3.16) to 0.6, 0.3, and 0.1, respectively. In the context of this research, we did not have access to any information related to the data-transmission delays. Thus, for our experimentation, we have randomly generated fixed data-transmission delays considering the following realistic assumptions. We assume that the delay between a content provider and a hosting node is generally bigger concerning the delay between any two hosting nodes. We also

assumed that the delay between two hosting nodes is usually bigger than between hosting and client-cluster nodes. Consequently, in our experiment, delays between CP nodes and hosting nodes were generated uniformly in the interval 120 - 700 [ms], delays between hosting nodes, from the interval 20 - 250 [ms], the delays between hosting nodes and client clusters were randomly sampled from the interval 8 - 80 [ms]. Also, the unitary data-transportation costs were randomly generated for resembling a multi-cloud deployment scenario. For links between CP nodes and hosting nodes, we assume that unitary DT costs range between 0.088 and 0.1 USD per GB[1]. For links between hosting nodes, the unit DT costs were randomly generated between 0.08 and 0.004 USD per GB, while DT Cost between hosting nodes and client cluster nodes is assumed null.

The rest of the simulation parameters are given in paragraph 3.2.4.

Simulation Environment

The training and evaluation procedures for our experiment were made on a *Google Colab-Pro* hardware-accelerated Environment equipped with a Tesla P100-PCIE-16GB GPU, an Intel(R) Xeon(R) CPU @ 2.30GHz processor with two threads, and 13 GB of main memory. The source code for our vCDN simulator and our DRL framework's training cycles was made in python v. 3.6.9. We used torch library v. 1.7.0+cu101 (PyTorch) as a deep learning framework. The whole code is available online on our public repository[2].

Compared state-of-art Algorithms

We compare our algorithm with the NFVDeep framework presented in [285]. We have created three progressive enhancements of the NFVDeep algorithm for an exhaustive comparison with E2-D4QN. NFVDeep is a policy gradient DRL framework for maximizing network throughput and minimizing operational costs on general-case SFC deployment. Xiao *et al.* design a *backtracking* method: if a resource shortage or exceeded latency event occurs during SFC deployment, the controller ignores the request, and no reward is given to the agent. Consequently, sparse rewards characterize NFVDeep. The first algorithm we compare with is a reproduction of NFVDeep on our particular Live-Streaming vCDN Environment. The second algorithm introduces our dense-reward scheme on the NFVDeep framework, and we call it NFVDeep-Dense. The third method is an adaptation of NFVDeep that introduces our dueling DDQN framework but keeps the same reward policy as the original algorithm in [285], and we call it NFVDeep-D3QN. The fourth algorithm is called NFVDeep-Dense-D3QN, and it adds our dense reward policies to NFVDeep-D3QN. Notice that the difference between NFVDeep-Dense-D3QN and our E2-D4QN algorithm is that the latter does not use the backtracking mechanism: In contrast to any of the compared algorithms, we permit our agent to do wrong VNF assignations and to learn from its mistakes to escape from local optima.

Finally, we also compare our proposed algorithm with a greedy-policy lowest-latency and lowest-cost (GP-LLC) assignation agent, based on the work presented in [248]. GP-LLC is an extension of the algorithm in [248], that includes server-utilization, channel-ingestion state, and resource-costs awareness in the decisions of a greedy policy. For each incoming VNF request, GP-LLC will assign a hosting node. This greedy policy will try not to overload nodes with assignation actions and always choose the best available actions in terms of QoS. Moreover, given a set of candidate nodes respecting such a greedy QoS-preserving criterion, the LLC criterion will tend to optimize hosting costs. Paragraph 3.2.4 describes in detail the GP-LLC SFC Deployment algorithm.

[1] https://cloud.google.com/cdn/pricing
[2] https://github.com/QwertyJacob/e2d4qn_vcdn_sfc_deployment

3.2.4 Further modelisation details

Resource Provisioning Algorithm

In this paper we assume that the VIM component is acting a *greedy* resource provisioning algorithm, i.e. the resource provision on f_i^k for the next time-step will be computed as:

$$c_{res,k,i}^{t+1} = \min(c_{res,k,i}^{t} \cdot \frac{\mu_{res,k,i}^{t}}{\hat{\mu}_{res,k,i}}, c_{res,k,i}^{max}), \forall res \in \{cpu, bw, mem\} \tag{3.26}$$

where the parameter $c_{res,k,i}^{max}$ is the maximum *res* resource capacity available for f_i^k, and $\hat{\mu}_{res,k,i}$ is a parameter indicating a fixed *desired utilization* of f_i^k after the adaptation takes place and before receiving further session requests. Recall that $\mu_{res,k,i}^{t}$ is the current *res* resource utilization in f_i^k. Resource adaptation procedure is triggered periodically each T_a time-steps, where T_a is a fixed parameter. On the other hand, each time that any f_i^k is instantiated, the VIM allocates a fixed minimum resource capacity for each resource in such VNF instance, denoted as $c_{res,k,i}^{min}$.

Inner delay-penalty function

The core of our QoS related reward is the delay-penalty function, which has some properties specified in paragraph 3.2.2. The function that we used on our experiments is the following:

$$d(t) = \frac{1}{t} + e^{-t} + 2e^{\frac{-t}{100}} + e^{\frac{t}{500}} - 1 \tag{3.27}$$

Notice that the domanin of $d(t)$ will be the RTT of any SFC deployment and the co-domain will be the segment $[-1, 1]$. Notice also that:

$$\lim_{t \to +\infty} d(t) = -1 \text{ and } \lim_{t \to t_{min}} d(t) \approx 1$$

Such a bounded co-domain helps to stabilize and improve the learning performance of our agent. Notice, however that it is worth noting that similar functions could be easily designed for other values of T.

Simulation Parameters

The whole list of our simulation parameters is presented in table 3.3. Every simulation has used such parameters unless other values are explicitly specified.

Table 3.3. List of simulation parameters

Parameter	Description	Value
γ_{CPU}	CPU Unit Resource Costs (URC) (for each cloud provider)	$(0.19, 0.6, 0.05)$
γ_{MEM}	Memory URC	$(0.48, 1.2, 0.1)$
γ_{BW}	Bandwidth URC	$(0.9, 2.5, 0.25)$
c_{max}	Maximum resource provision parameter (assumed equal for all the resource types)	20
c_{min}	Minimum resource provision parameter (assumed equal for all the resource types)	5
ϕ_p	Payload workload exponent	0.2
ϕ_b	Bit-rate workload exponent	0.1
ρ^*_{cpu}	Optimal CPU Processing Time (baseline of over-usage degradation)	$5 \cdot 10^{-3}$
ρ^*_{mem}	Optimal memory PT	$1 \cdot 10^{-3}$
ρ^*_{bw}	Optimal bandwidth PT	$5 \cdot 10^{-2}$
ψ_{cpu}	CPU exponential degradation base	100
ψ_{mem}	Memory deg. b.	100
ψ_{bw}	Bandwidth deg. b.	100
I_{ch}	cache VNF Instantiation Time Penalization in ms (ITP)	10000
I_{st}	streamer VNF ITP	8000
I_{co}	compressor VNF ITP	7000
I_{tr}	transcoder VNF ITP	11000
T_a	Time-steps per greedy resource adaptation	20
$\hat{\mu}_{res,k,n}$	Desired resulting utilization after adaptation	0.4
α_{res}	Optimal resourse res utilization (assumed equal for every resource type)	0.75

Training Hyper-parameters

A complete list of the hyper-parameters' values used in the training cycles is specified in table 3.4. Every training procedure has used such values unless other values are explicitly specified.

Table 3.4. List of hyper-parameters' values for our training cycles.

Hyper-parameter	Value
Discount factor (γ)	0.99
Learning rate	$1.5 \cdot 10^{-4}$
Time-steps per episode	80
Initial ϵ-greedy action probability	0.9
Final ϵ-greedy action probability	0.0
ϵ-greedy decay steps	$2 \cdot 10^5$
Replay memory size	$1 \cdot 10^5$
Optimization batch size	64
Target-network update frequency	5000

GP-LLC Algorithm specification

Algorithm 2 GP-LLC VNF Assignation procedure

1: **for** $f_r^k \in r$ **do**
2: Get the non-overloaded hosting nodes set $\hat{N}_H$
3: Get the still-scalable hosting nodes set $\tilde{N}_H$
4: Get the set of hosting nodes that currently have a f^k ingesting l_r on N_l^k
5: **if** $|\hat{N}_H| > 0$ **then**
6: **if** $|\hat{N}_H \cap \tilde{N}_H| > 0$ **then**
7: use the LLC criterion to chose f_r^k from $\hat{N}_H \cap \tilde{N}_H$
8: **else**
9: use the LLC criterion to chose f_r^k from $\hat{N}_H$
10: **end if**
11: **else**
12: **if** $|\tilde{N}_H| > 0$ **then**
13: use the LLC criterion to chose f_r^k from $\tilde{N}_H$
14: **else**
15: choose a random node f_r^k from $|\hat{N}_H|$
16: **end if**
17: **end if**
18: **end for**

In this paper, we have compared our E2-D4QN agent with a greedy policy lowest-latency and lowest-cost (GP-LLC) SFC deployment agent. Algorithm 2 describes the behavior of the GP-LLC agent. Note that the lowest-latency and lowest-cost (LLC) criterion can be seen as a procedure that, given a set of candidate hosting nodes, N_H^c chooses the correct hosting node to deploy the current VNF request $\hat{f}_r^k$ of a SFC request r. Such a procedure is at the core of the GP-LLC algorithm, while the outer part of the algorithm is responsible for choosing the hosting nodes that form the candidate set according to a QoS maximization criterion. The LLC criterion woks as follows. Given a set of candidate hosting nodes and a VNF request, the LLC criterion will divide the candidate nodes in subsets considering the cloud provider they come from. It will then chose the hosting node corresponding to the route that will generate less transportation delay, i.e. the fastest route, from the cheapest cloud-provider candidate node subset.

The outer part of the algorithm acts instead as follows. Every time that a VNF request $\hat{f}_r^k$ needs to be processed, the GP-LLC agent monitors the network conditions. The agent identifies the hosting nodes that are currently not in overload conditions, $\hat{N}_H$, the ones that currently have a resource provision that is less than the maximum for all the resource types $\tilde{N}_H$, and the hosting nodes that currently have a f_k ingesting the content from the same content provider requested by $\hat{f}_r^k$, N_l^k (lines 2-4). Notice that $\tilde{N}_H$ is the set of nodes whose resource provision can still be augmented by the VIM. Notice also that choosing a node from N_l^k to assign $\hat{f}_r^k$, implies not to incur in a Cache MISS event and consequently warrants the acceptance of r. If $\hat{N}_H \cap \tilde{N}_H$ is not an empty set, the agent assigns $\hat{f}_r^k$ to a node in such a set following the LLC criterion. However, if $\hat{N}_H \cap \tilde{N}_H$ is an empty set, then if at least $\hat{N}_H$ is not empty, then a node from $\hat{N}_H$ will be chosen using the LLC criterion. If on the other hand, $\hat{N}_H$ is empty, then a node from $\tilde{N}_H$ will be chosen with the LLC criterion. Finally, if both $\hat{N}_H$ and $\tilde{N}_H$ are empty sets, then a random hosting node will be chosen for hosting $\hat{f}_r^k$ (lines 5-16). Choosing a random node in the last case instead of using the LLC criterion from the whole hosting node set will prevent bias in the assignation policy to cheap nodes with fast routes. Making such a random choice will then result in an increment in the overall load balance among the hosting nodes.

Notice that, whenever possible, GP-LLC will evict overloading nodes with assignation actions and will always choose the best actions in terms of QoS. Moreover, given a set of candidate nodes

respecting such a greedy QoS-preserving criterion, the inner LLC criterion will tend to optimize hosting costs and data-transmission delays. Notice also that GP-LLC does not take into account explicitly the data-transportation costs for VNF SFC deployment. However, because it pays attention to the ingestion states of the VNFs, it indirectly contributes to the optimization of the session amortized DT-costs, and thus, the whole DT-costs of the vCDN.

3.3 Results

Various performance metrics for all the algorithms mentioned in paragraph 3.2.3 are presented in figure 3.3. Recall that the measurements in such a figure are taken during the 1-day evaluation trace as mentioned in paragraph 3.2.3. Notice that, given the time-step duration and number of time-steps per episode specified in paragraph 3.2.3, one-day trace consists of 72 episodes, starting at 00:00:00 hours at finishing at 23:59:59 of the 29th July 2017.

3.3.1 Mean scaled Network throughput per episode

The network throughput for each simulation time-step was computed using (3.10) and the mean values for each episode were scaled and plotted in figure 3.3 (a). Also the scaled incoming traffic amount is plotted in such a figure. In the first twenty episodes of the trace, which correspond to the period from 0:00 to 6:00, the incoming traffic goes from intense to moderate. Incoming traffic has minor oscillations with respect to the antecedent descent from episode 20 to episode 60, and it starts to grow again from the sixtieth episode on, which corresponds to the period from 18:00 to the end of the trace.

The initial ten episodes are characterized for a comparable throughput between GP-LLC, E2-D4QN, and NFVDeep-Dense-D3QN. We can see, however, from the 20th episode on, the throughput of policy-based NFVDeep variants is lowered. From episode 15, however, which corresponds to the period from 05:00 to the end of the day, the throughput of our proposed algorithm is superior with respect to every other algorithm.

3.3.2 Mean Acceptance Ratio per episode

The AR for each simulation time-step was computed using (3.1) and the mean values for each episode are plotted in figure 3.3 (b). At the beginning of the test, corresponding to the first five episodes, E2-D4QN has a superior AR performance. From episodes 5 to 15, only E2-D4QN and NFVDeep-Dense-D3QN keep growing in the acceptance ratio. The unique algorithm that holds a satisfactory acceptance ratio during the rest of the day is E2-D4QN instead. It should be stressed that the AR cannot be one in our experiments because the parameter configuration described in paragraph 3.2.3 resembles overloaded network conditions on purpose.

3.3.3 Mean rewards per episode

Figure 3.3 (c) shows the mean rewards per episode. We plot the rewards obtained at every $|K|$ assignation steps. Notice that such a selection corresponds to the non-null rewards in the sparse-reward models.

During the first 15 episodes, -00:00 to 05:00- both GP-LLC and E2-D4QN increment their mean rewards starting from a worse performance with respect to the NFVDeep algorithms. This is explained because the *Enhanced-Exploration* mechanism of E2-D4QN and GP-LLC is the unique that includes negative rewards in the reward assignation policy. From episode 15 to 20, E2-D4QN reaches the rewards obtained by NFVDeep-Dense-D3QN, and from episode 20 on, E2-D4QN has a better performance with respect to the NFVDeep variants, most of all due to the lowering of operational costs for this algorithm. Finally, with the exception of the last 5 episodes, only E2-D4QN

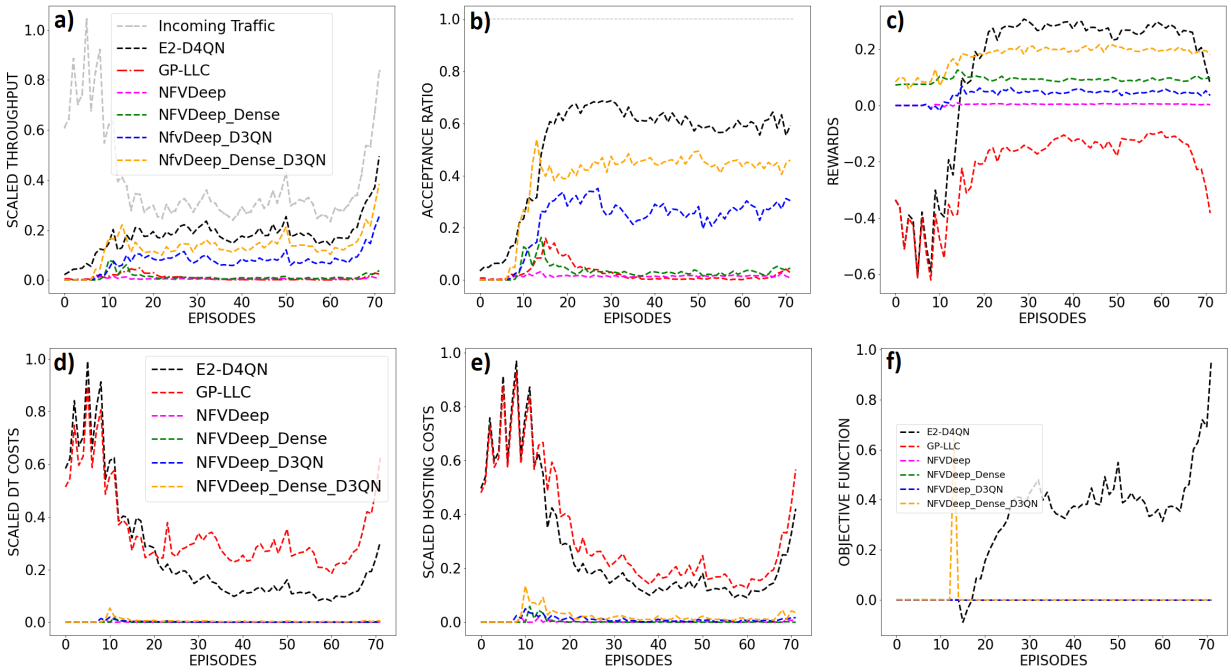

Figure 3.3. Basic evaluation metrics of *E2-D4QN*, *GP-LLC*, *NFVDeep* and three variants of the latter, presented in paragraph 3.2.3. (a) Scaled mean network throughput per episode. (b) Mean Acceptance Ratio per episode. (c) Mean rewards per episode. (d) Mean scaled total Data-Transportation Costs per episode (e) Mean scaled total hosting costs per episode. (f) Mean scaled optimization objective per episode.

dominates the rest of the trace in the mean reward metric, with the exception of the last five episodes.

3.3.4 Total scaled Data-Transportation costs per episode

The total DT costs per time-step as defined in (3.13) were computed and the mean values per episode were scaled and plotted in figure 3.3 (d). During the whole evaluation period, both E2-D4QN and GP-LLC incur in higher DT costs with respect to every other algorithm. This phenomenon is explained by the Enhanced Exploration mechanism, which permits E2-D4QN and GP-LLC, to accept requests even when the resulting RTT is over the acceptable threshold. Notice, however, that E2-D4QN and GP-LLC progressively reduce DT costs due to common path creation for similar SFCs. From the twentieth episode on, however, only E2-D4QN minimizes such costs while maintaining an acceptance ratio greater than 0.5.

3.3.5 Total Scaled Hosting Costs per episode

A similar explanation can be given for the total Hosting Cost behavior. Such a cost was computed for each time-step using (3.11) and the mean values per episode were scaled and plotted in figure 3.3 (e). Given the adaptive resource provisioning algorithm described in appendix 3.2.4, we can argue that, in general, hosting cost is high for E2-D4QN and GP-LLC because their throughput is high. The hosting costs burst that characterizes the first twenty episodes, however, can be explained by the network initialization state during experiments: Every algorithm is evaluated from an uncommon network state with respect to the steady state reached during training. The algorithms that are equipped with the Enhanced-Exploration mechanism tend to worse such a performance drop at the beginning of the testing trace because of the unconstrained nature of such mechanism. It is the Enhanced-Exploration however, that drives our proposed agent to learn robust policies that permit

to maximize the network acceptance ratio.

3.3.6 Optimization objective

The optimization objective as defined in (3.16) was computed at each time-step, and the mean values per episode were scaled and plotted in figure 3.3 (f). Recall that (3.16) is invalid whenever the acceptance ratio is above the minimum threshold of 0.5 as mentioned in paragraph 3.2.1. For this reason, in figure 3.3 (f) we set the optimization value to zero whenever the minimum service constraint was not met. Notice that no algorithm can achieve the minimum acceptance ratio during the first ten episodes of the test. This behavior can be explained by the greedy initialization with which every test has been carried out: The initial network state for every algorithm is very different from the states observed during training. From the tenth episode on, however, E2-D4QN is the only algorithm to achieve a satisfactory acceptance ratio, and thus, the optimization objective function has a non-zero value.

3.4 Discussion

Trace-driven simulations have revealed that our approach shows adaptability to the complexity of the particular context of Live-Streaming in vCDN with respect to the state-of-art algorithms designed for general-case SFC deployment. In particular, our simulation test revealed decisive QoS performance improvements in terms of acceptance ratio with respect to any other *backtracking* algorithm. We assess the algorithm's performance in a bounded-resource scenario aiming to build a safe-exploration strategy that enables the market entry of new vCDN players. Our experiments have shown that the proposed algorithm, E2-D4QN is the only one to adapt to such conditions, maintaining an acceptance ratio above the general case state-of-art techniques while keeping a satisfactory balance between network throughput and operational costs.

Based on the results in the previous section, we now argue the main reasons that make E2-D4QN the most suitable algorithm for online optimization of SFC Deployment on a Live-video delivery vCDN scenario. **We argue that the main reason for our proposed algorithm's advantage is the injected exploration bias, which is the result of the combination of the state-space design with a dense-reward shaping on a dueling DDQN framework. This exploration bias results in efficient discovery of long-term-convenient actions in contrast to actions convenient only in a short-horizon. Moreover, we claim that the fact that we removed the backtracking algorithm implemented in NFVDeep reduces the strength of an excessive exploration bias of the solution space and permits to discover solutions that perform better through allowing a wider range of serial SFC deployment strategies.**

3.4.1 Environment complexity adaptation

As explained in paragraph 3.2.3, we have compared our E2-D4QN agent with the NFVDeep algorithm presented in [285], with three progressive enhancements to such algorithm, and with an extension of the algorithm presented in [248], which we called GP-LLC. Authors in [285] assumed utilization-independent processing times. A consequence of this assumption is the possibility of computing the remaining latency space with respect to the RTT threshold before each assignment. In our work, instead, we argue that realistic processing times should be modeled as utilization-dependent. Moreover, Xiao *et al.* did not model ingestion-related utilization nor VNF instantiation time penalties. Relaxing the environment with these assumptions simplifies the environment and helps on-policy DRL schemes like the one in [285] to converge to suitable solutions. Lastly, in NFVDeep, prior knowledge of each SFC session's duration is also assumed. This feature helps the agent to learn to accept longer sessions to increase the throughput. Unfortunately, it is not realistic to assume session duration knowledge when modeling Live-Streaming in vCDN context. Our model is agnostic

to this feature and maximizes the overall approximated throughput when optimizing the acceptance ratio. Our work shows that the NFVDeep algorithm cannot reach a good AR on SFC Deployment optimization without assuming all the aforementioned relaxations.

3.4.2 State Value, Advantage Value and Action value Learning

In this work, we propose the usage of the dueling-DDQN framework for implementing a DRL agent that optimizes SFC Deployment. Such a framework is meant to learn approximators for the state value function, $V(s)$, the action advantage function, $A(a)$, and the action-value function, $Q(s, a)$. Learning such functions helps to differentiate between relevant actions in the presence of many similar-valued actions. This is the main reason why NFVDeep-D3QN improves AR with respect to NFVDeep: Learning the action advantage function, helps to identify convenient long-term actions from a set of similar valued actions. For example, suppose that, to maximize QoS, we adopt a round-robin load-balancing SFC deployment strategy. In that case, the SFC routes to content providers won't tend to divide the hosting nodes into non-overlapping clusters. This will provoke more resource usage in the long run: almost every node will ingest the content of almost every content provider. As generally content-ingestion resource usage is much heavier with respect to content-serving, this strategy will accentuate the resource leakage on the vCDN in the long run, provoking bad QoS performance. Our E2-D4QN learns to polarize the SFC routes in order to minimize content ingestion resource usage during the training phase. Such a biased policy performs in the best way possible with respect to the compared algorithms taking into account the whole evaluation period.

3.4.3 Dense Reward Shaping

Our agent converges to efficient policies by carefully designing a reward schema as the one presented in paragraph 3.2.2. Our algorithm assigns a specific reward at each MDP transition considering the optimality of VNF assignments in terms of QoS. This dense-reward schema enhances the agent's convergence. In fact, in our experiments, we have also noticed that the dense-reward algorithms improve the results of their sparse-reward counterparts. In other words, we see in figure 3.3 (b) that NFVDeep-Dense performs slightly better than NFVDeep, and NFVDeep-Dense-D3QN performs better than NFVDeep-D3QN. This improvement exists because dense rewards provide valuable feedback at each assignation step of the SFC, improving convergence of the DRL agents to shorter RTTs. On the other hand, we have also observed that even if cost-related penalties are sparsely subtracted in our experiments, the proposed DRL agent learns to optimize SFC deployment not only with respect to QoS but also taking into account the operational costs.

Finally, if we look at all the plots and analyze them together, we will notice that the yellow algorithm, namely NFVDeep-Dense-D3QN, is not performing very bad in terms of approximated throughput, and operational costs. This fact supports our claim that the enhancements proposed to the REINFORCE algorithm in the form of inductive biases were very useful, moreover, this supports our claim that we need to find a trade-off when creating inductive biases, because it demonstrates that the "backtracking" mechanism leads to a sub-optimal solution, as we will now discuss.

3.4.4 Enhanced Exploration

Notice that, even if it has the enhanced-exploration, GP-LLC does not learn inherent network traffic dynamics, making it impossible to differentiate convenient long-term actions from greedy actions. For example, GP-LLC won't adopt long-term resource consolidation policies, in which each channel is ingested by a defined subset of nodes to amortize the overall ingestion resource consumption. This fact supports the claim that the advantage of our proposed algorithm is not solely a merit of the enhanced exploration, which can be seen as a relaxation of a strong exploration bias on the solution space, but it obeys the effective biases created trough the reward shaping and the state space design. Moreover, one could argue that the main reason that keeps NFVDeep above

a satisfactory performance is the leakage of the enhanced exploration mechanism, but authors of NFVDeep have claim that the backtracking algorithm is what helps their agent to improve the training results. This fact could then be a negative argument that supports the previous mentioned hypothesis. Adding the enhanced exploration enriches the experience replay buffer with non-optimal SFC deployments during training, this may be the main reason why it helps preventing the agent from getting stuck at local optima, which instead we think is the main problem with NVFDeep under our environment conditions.

In practice, in all the backtracking algorithms, at each VNF assignment, the candidate nodes are filtered based on their utilization availability. Only non-overloaded nodes are available candidates. If the agent chooses an overloaded node, the action is ignored, and no reward is given. Our model instead performs assignment decisions without being constrained by current node utilization to enhance the exploration of the action space. We argue that the well-designed state-space codifies important features that drive learning towards robust VNF placements:

1. The request vector codifies useful information about the requests that help our agent extract the incoming traffic patterns.

2. The ingestion vector helps our agent to optimize ingestion-related resource demand by concentrating attention on the VNF instances that do ingest the content requested at the moment of assignation.

3. The maximum utilization vector gives our agent resource utilization awareness, making it able to converge to assignation policies that optimize processing times and preserve QoS.

Note that our agent learns optimal SFC Deployment policies without knowing the actual bounds of the resource provision. It learns to *recognize* the maximum resource provisioning for the VNFs and also learns to evict assignations to non-initialized VNFs thanks to our carefully designed state-space representation and reward assignation scheme. It should be stressed that all the compared algorithms where fed with the same state-space representations, however they were unable to abstract the link between these features and good policies with their underlying reward shaping and transition function.

3.4.5 Work limitations and future research directions

We have based the experiments in this paper on a real-world dataset concerning a particular video delivery operator. In this case, the hosting nodes of the corresponding proprietary CDN are deployed in the Italian territory. However, such a medium-scale deployment is not the unique possible configuration for a CDN. Consequently, as future work, we plan to obtain or generate data concerning high-scale topologies to assess the scalability of our algorithm to such scenarios.

Further, this paper presents the assessment of the performance of various DRL-based algorithms. However, the authors of this work had access to real-world dataset limited to a five-day trace. Consequently, the algorithms presented in this work were trained on a four-day trace, while the evaluation period consisted of a single day. Future research directions include assessing our agent's training and evaluation performance on data concerning more extended periods of time. We mention also that more extensive experiments could be done with respect to those presented in this chapter. For example, we could asses ablation studies trying to characterize the inner delay function that composes our reward shaping strategy, or test other value-based DRL algorithms. Last, but not least, we plan to implement the simulated framework on a real NFV-MANO vCDN architecture to prove our claims, an important feature of a real-world evaluation will be that we will avoid comparing between approximated metrics and have real-time exact measurements.

Chapter 4

Use Case 2: GRN inference through temporal co-clustering of Gene Expression and CRE activity

This chapter presents a solution to the open challenge identified in section 2.2.4, i.e., the problem of co-clustering genes and cis-regulatory elements characterized by developmental temporal features to infer gene expression regulatory relationships. Gene expression data, as well as data from three CRE activity markers from a publicly available dataset of mouse fetal heart tissue, were used for concept proofing of the proposed solution. In this study we used open chromatin accessibility from ATAC-seq experiments, as well as H3K27ac and H3K27me3 histone marks as CREs activity markers. However, this method can be executed with other sets of markers. We modeled all data sources as a heterogeneous graph containing genes and CREs and adapted a state-of-the-art representation learning algorithm to produce a low-dimensional and easy-to-cluster embedding of the heterogeneous graph of genes and CREs. Deep graph auto-encoders and an adaptive-sparsity generative model are the algorithmic core of our solution.

Taking into account *a priori* model knowledge, we have designed proper combination rules for the heterogeneous gene expression and CRE activity data and we have created a deep het-graph-RL algorithm where the objective function permitted to identify well-known gene expression regulatory mechanisms by clustering. Our algorithm sheds light on developmental regulatory mechanisms in mouse fetal-heart tissue. Function enrichment analysis proves that the genes in the co-clusters are involved in distinct biological processes. The enriched transcription factor binding sites in CREs prioritize the candidate transcript factors which drive the temporal changes in gene expression.

4.1 Problem Definition

The main novelty of this work is the introduction of a model for regulatory networks discovery which is based on the concept of heterogeneous graphs [90]. We modeled a graph containing two classes of nodes: genes and their potential cis-regulatory elements, now on termed as candidate-CREs (cCRE), and multiple types of relations or edges between nodes. One such relation type is the similarity between temporal gene expression profiles. Another relation type is the similarity between temporal cCREs activity patterns. Finally, we introduced a third edge type: the base-pair distance between genes and cCREs in the genome. Having defined such relations, we propose the usage of graph representation learning (graph-RL) to group genes and cCREs as a function of the probability of the existence of gene regulation mechanisms between them.

To this end, we extended a state-of-the-art grap-RL algorithm presented in [137] to make it capable of learning representations of heterogeneous graphs and applied it to the ENCODE heart dataset in [76]. We used open chromatin regions from ATAC-seq as cCREs. Besides ATAC-seq, we used H3K27ac and H3K27me3 histone marks, well-known markers of enhancer activity and polycomb

repression, respectively. Our framework produced a relation type between genes and cCREs which is aware of both same-class pattern similarities and base-pair proximities between elements of different classes. Moreover, through the usage of the adaptive neighbors model presented in [137] we produced a "clustering-friendly" embedding where we could straightforwardly perform clustering. The resulting clusters contained both genes and cCREs and tended to identify possible regulatory mechanisms. We demonstrated this claim by evaluating such clusters' semantic power with external expert criteria. We concluded that the proposed methodology is able to learn suitable combinations of multiple entity relations to form a unique relation under a graph that resembles a gene regulatory network. Our framework is called DeepReGraph, because it is a *Deep* learning-based algorithm for identification of gene expression *Re*gulatory mechanisms through heterogeneous *Graph* RL. Our full code, the pre-processed datasets used, and the values of the parameters and hyper-parameters used in our experiment are online available at our public repository[1].

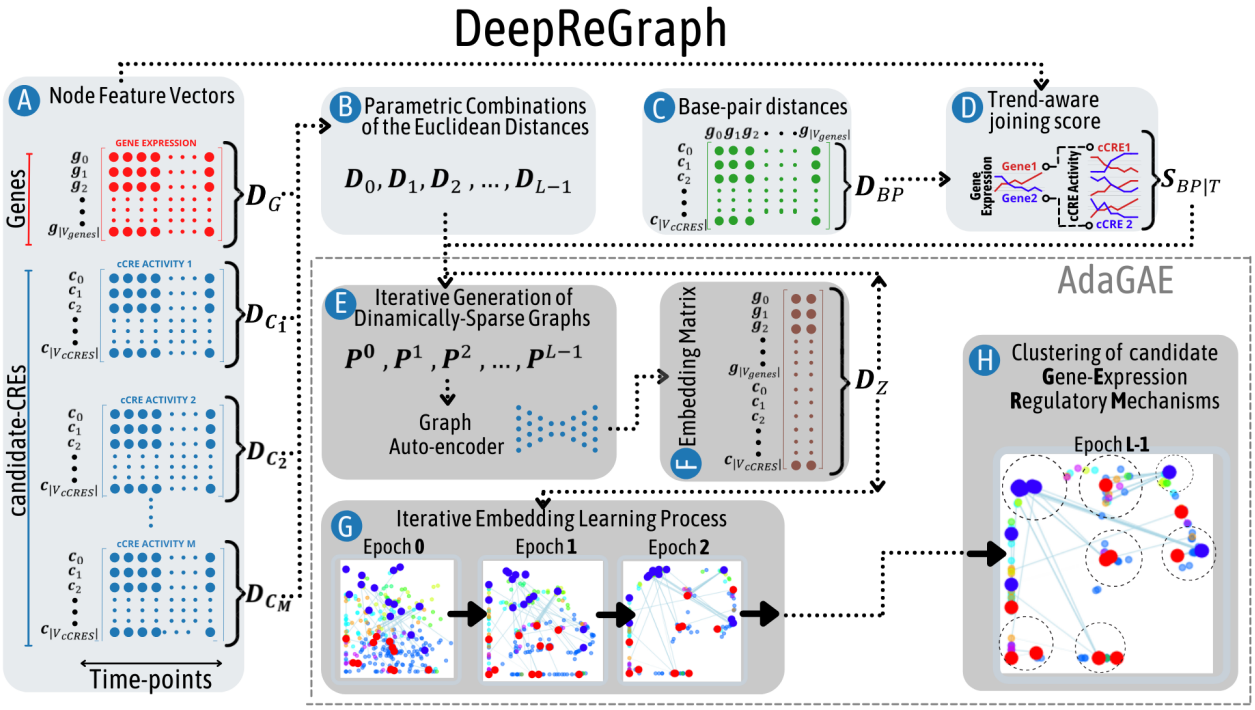

Figure 4.1. Schematic representation of DeepReGraph. Our framework combines multiple temporal enhancer activity markers with temporal gene expression data and base-pair distances to build a heterogeneous graph of genes and cis-regulatory elements (CRE). The proposed framework then applies an adapted version of AdaGAE [137] to find a low-dimensional, easy-to-cluster embedding representation of data through an iterative optimisation process. In the final embedding produced by our method, the spatial distribution of genes and cCREs resembles candidate gene-expression regulatory Mechanisms.

Recall that heterogeneous graphs consist of the triplet $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{T})$, where the set of node and edge types, $\mathcal{T}$, has been included. $\mathcal{T}_\mathcal{E}$ is the set of edge types and $\mathcal{T}_\mathcal{V}$ is the set of node types. In this thesis we argue that the work in [137] can be used for heterogeneous graph representation learning if all the relation-wise similarity measures $S_1, S_2, ...S_{|\mathcal{T}_\mathcal{E}|}$ are some-how combined to form a unique similarity measure S. We can then define a generative probability distribution family P as a function of S and drive manifold learning considering the family of distributions P as the set of learning objective functions. By designing a manifold learning in such a way, the pairwise similarity relationship in Z will represent a combination of multiple types of similarities in the original feature space. Moreover, if we use the adaptive neighbors framework for such a heterogeneous manifold

[1] Our source code, the data and a interactive notebook are available at: https://github.com/QwertyJacob/DeepReGraph.

learning, we can induce the formation of clusters were each cluster is formed taking into account multi-semantic information.

4.2 Materials and Methods

To demonstrate DeepReGraph's ability to cluster genes and cCREs simultaneously, we applied this method to a mouse heart fetal developmental dataset from the ENCODE project [76]. Gene expression and cCRE activities were time-series formatted. We used three well-characterized epigenetic markers: ATAC-seq, H3K27ac, and H3K27me3 measurements as features of cCRE temporal activities. We aimed to create a low-dimensional and clustered representation of the whole dataset, where clusters represent candidate gene-expression regulatory mechanisms (GERM). In other words, we aimed to create clusters containing both genes and cCREs, with the working hypothesis that cCRES on a cluster are plausibly regulating the expression of the genes in the same cluster. Moreover, genes on the same GERM cluster should have similar gene expression profiles, and cCREs should have similar activity profiles. Lastly, the base-pair distances between genes and cCREs on the same cluster should be relatively small with respect to distances between elements in different clusters. We validated heterogeneous clusters of genes and cCRES to study how gene regulatory networks (GRN) drive changes in gene expression during mouse heart fetal development. To validate gene expression clusters, we performed enrichment analysis of biological process gene ontology terms using the ClusterProfiler bioconductor package [281]. To validate the cCRE clusters, we looked for the Enriched Homer Motifs [96].

4.2.1 Data pre-processing

Data pre-processing was done in two stages. Firstly, gene expression and cCRE datasets were normalized across temporal data. The second stage included filtering-out temporal profiles where the correlation across replicates or the variance across time was below some pre-defined thresholds. We explain the pre-processing stages in greater detail below.

We firstly downloaded the gene expression values, called peaks, and bam files of ATAC-seq, H3K27ac, and H3K27me3 from the ENCODE database[76] (mice heart fetal tissue). We used logFPKM values of gene expression throughout this paper. To detect cCRE regions, we re-centered ATAC-seq called peaks across time-points and replicates using the Bioconductor DiffBind package [218]. To determine chromatin accessibility in each cCRE region, we took regions of 500 nucleotides long (250 nucleotides on each side from the center of cCREs). We recorded binding intensities for each replicate of each time point separately by using the Bioconductor Rsubread package [141]. We counted H3K27ac and H3K27me3 marks for cCREs in the same manner, but we used regions of 3000 nucleotides long instead (1500 nucleotides each side from the center of cCREs). We took the median value of non-peak regions in each experiment separately to remove background counts from real intensity counts. Then, we subtracted these median values from the intensities of each experiment separately. We used logRPKM values of denoised counts across replicates and time-points. Removing the background noise caused the distribution of each experiment to resemble a Gaussian distribution.

At the end of this process, we had two replicas of gene expression time-series profiles for a set of genes and two replicas of cCRE activity time-series for each activity marker over a set of cCREs. We then proceeded to filter out invalid genes and cCREs based on two criteria. The first group of filters we applied to this data were proposed in [61]. For each gene, we measured the correlation of the gene expression profiles of both replicates and discarded every gene whose replicate had a negative correlation value. We did the same with the cCREs for each activity marker profile. If only one activity marker had a negative correlation between replicates, the cCRE was discarded. We also discarded every element (gene or cCRE) where two consecutive time probes had an absolute ratio greater than 2.

We also discarded low-expressed genes and low activity-characterized cCREs [99]. We computed the mean gene expression value for each gene and discarded every gene whose mean expression value

was under the 0.8 percentile of such mean value distribution. We did the same with cCREs: we computed the mean activity value for each activity marker and discarded every cCRE where the mean value was below a minimum percentile threshold. Such thresholds were 0.8 for ATAC-seq and 0.7 for H3K27ac and H3K27me3. Lastly, for gene expression and each activity marker dataset, we created a 3-rd degree polynomial regression to estimate the variance of each time series as a function of its mean value. We discarded all the elements where the actual variance was below the predicted variance. Our pre-processed and cleaned data consisted of 607 genes and 5239 cCREs from the mouse heart dataset. This dataset is available online on our public repository. Our objective was to shed light on regulatory mechanisms among genes and cCREs. Having cleaned our dataset, modeling it as a heterogeneous graph is the strategy we used to combine all the data and learn a new relation type between elements that gives information about gene expression regulation. We explain the graph modelling in the next paragraph.

4.2.2 Heterogeneous Graph Modeling

Many industrial data are represented as a graph, i.e., as multiple entities with quantifiable relationships between them. Nowadays, graph modelisation is widely used in bioinformatics [300]. One strategy to extract added-value from graphs is the use of graph-RL algorithms. Graph-RL has been widely studied by the research community over the past few years [90, 236, 36, 301]. Graph-RL algorithms seek to map the information of a graph on a reduced-dimension latent space where the geometrical distances between elements in such space resemble the relations in the original graph. In other words, graph-RL, also referred to as "graph embedding", consists in finding a reduced dimensional representation of the nodes of a graph while conserving most of the semantic information of such a graph. [287, 91].

A graph can be homogeneous if it has a unique type of node and a unique type of relation defined between them, or heterogeneous, if it introduces multiple edge and node types. We argue that regulatory networks during embryonic development can also benefit from being modeled as a heterogeneous graph. Reduced-dimension representations of such a graph could contain multiple types of information that, if well combined, could semantically express regulatory mechanisms of gene expression.

We defined a heterogeneous graph and described it as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{T})$, where $\mathcal{V}$ is the set of nodes, $\mathcal{E}$ is the set of edges and $\mathcal{T} = \mathcal{T}_{\mathcal{E}} \cup \mathcal{T}_{\mathcal{V}}$ is the set containing all the node types and edge types. The set of node types consists of genes and cCREs, and is denoted by $\mathcal{T}_{\mathcal{V}} = \{genes, ccres\}$. Each gene $g_i, \forall i \in |\mathcal{V}_{genes}|$ in the dataset is characterized by a gene expression time-series profile that will be encoded in a vector denoted by $\mathbf{g}_i$. Note that the length of $\mathbf{g}_i$ coincides with the number of time-points considered in the gene expression data. The vectors $\mathbf{g}_i, \forall i \in |\mathbf{V}_{genes}|$ can be concatenated as rows to form the feature matrix represented in red in panel A of Figure 4.1.

Every candidate cis-regulatory element, instead, will be denoted as $c_j, \forall j \in |\mathcal{V}_{genes}|$ and will be characterized by M activity time-series profiles, that will be encoded in corresponding vectors denoted as $\mathbf{c}_j^1, \mathbf{c}_j^2, ..., \mathbf{c}_j^M$. The time-points under which gene expression and cCRE activity values are available are the same. In this work, $M = 3$. Specifically, we considered the ATAC-seq, H3K27ac, and H3K27me3 activity time-series profiles for each cCRE, but one could include more cCRE activity markers in the set of features. Note that the vectors $\mathbf{c}_j^m, \forall j \in |\mathbf{V}_{cCREs}|$ can be concatenated as rows to form the matrices represented in blue in panel A of Figure 4.1.

Each node of our graph was associated to one or multiple feature vectors that can be seen as points on a multi-dimensional feature space. The set of feature-spaces defined for the nodes in $\mathcal{G}$ is denoted as $R = \{G, C_1, C_2, ...C_M\}$ where G is the gene expression feature space and $C_1, C_2, ...C_M$ are the feature spaces of the M cCRE activity data. Each one of these feature spaces will be referred to as *original feature spaces*, and the vectors $\mathbf{g_i}, \mathbf{c_j^1}, \mathbf{c_j^2}, ..., \mathbf{c_j^M}$ as original feature vectors from now on. Note that such vectors correspond to the rows of the matrices in panel A of Figure 4.1. It is well-known that the probability of the existence of a regulatory mechanism between a gene and a cis-regulatory element is inversely proportional to the base-pair distance between them [325].

For this reason, in our work, we used another data source, called the Link Matrix, that gives us information about the base-pair distance between genes and cCREs. We represented such a matrix in panel C of Figure 4.1. Interaction between cCREs and genes is generally possible in the range of 1 Mega base-pair. [175] Consequently, we considered only distances lower than 10^6 base-pairs. For practical purposes, we scaled the base-pair distance values in the interval $[0, 1]$ to define a scaled base-pair distance function, D^{BP}:

$$\begin{aligned} D^{BP} : (g_i, c_j) \rightarrow d^{BP}_{i,j} \in [0, 1], \\ \forall\ g_i \in \mathcal{V}_{genes}, c_j \in \mathcal{V}_{cCREs} \end{aligned}$$

Given the scaled distance function, we created a base-pair proximity relationship S_{BP} using a parametric transformation:

$$S_{BP}(i, j) = \frac{1}{(d^{BP}_{i,j}/\beta_c)^{\beta_d} + 1},\ \forall i, j \in |\mathcal{V}| \tag{4.1}$$

where β_c and β_d are fixed hyper-parameters. Notice that the domain and co-domain of S_{BP} is defined in the interval $[0, 1]$. The gene-to-cCREs relationships defined by S_{BP} are sparse in the sense that, for each gene, only a few cCREs will have a non-zero proximity value for it.

The interaction between cCREs and the genes they regulate is complex. Some cCRE markers are correlated with gene expression and some others are anti-correlated. [53] This correlation or anti-correlation is governed by the mechanism that a specific marker affects the gene expression. For example, chromatin should be opened prior to the gene expression. Therefore, chromatin accessibility is commonly directly correlated with gene expression. H3K27ac is also directly correlated with gene expression, as this epigenetic modification happens in active transcription regions. On the other hand, H3K27me3 shows epigenetic modifications that happen at polycomb repressed regions, and are consequently inversely correlated with the expression of the corresponding regulated genes. We needed to capture the relevance of these and other temporal covariances between gene expression and the correspondent regulatory elements' activity markers.

To this end, we created a joining score, J_T, as a function of the temporal slopes of gene expression and cCRE activity time-series:

$$\begin{aligned} J_T(g_i, c_j) =& |\gamma^g_i + \frac{\sum_{m \in M} \omega_m \gamma^m_j}{\sum_{m \in M} \omega_m}|,\ \text{where:} \\ \gamma^g_i =& \text{sgn}(\frac{\partial \mathbf{g}_i}{\partial t}),\ \ \gamma^m_j = \text{sgn}(\frac{\partial \mathbf{c}^m_j}{\partial t}) \\ & \forall\ g_i \in \mathcal{V}_{genes}, c_j \in \mathcal{V}_{cCREs} \end{aligned} \tag{4.2}$$

where the temporal slope of gene expression vector $\mathbf{g}_i$ is denoted by $\frac{\partial \mathbf{g}_i}{\partial t}$, the slope of cCRE activity vectors is $\frac{\partial \mathbf{c}^m_j}{\partial t}, \forall m \in \{0, 1, ..., M-1\}$, $\text{sgn}(\cdot)$ is the signum function, and $\omega_m \in [-1, 1]$ is a fixed parametric weight for the $\mathbf{c}^m$ trend slope.

We computed a trend-aware score multiplying (4.1) and (4.2) for each gene-cCRE pair in our graph:

$$\begin{aligned} S_{BP|T}(g_i, c_j) = S_{BP}(i, j) \cdot J_T(g_i, c_j), \\ \forall\ g_i \in \mathcal{V}_{genes}, c_j \in \mathcal{V}_{cCREs} \end{aligned} \tag{4.3}$$

We represented the computation of Equation (4.3) in panel D of Figure 4.1. Having defined the trend-aware score, we could differentiate between any pair of genes, g_j and g_k, when these were equally proximal to any cCRE c_i but had gene expression vectors with different temporal trends. In this case, the original base-pair proximity score in (4.1) assigned the same value to the association (g_j, c_i) and (g_k, c_i). Notice that the trend-aware score in (4.2), instead, distinguishes

the most plausible association of c_i with respect to the alternatives g_j and g_k as a function of the temporal slopes of the gene expressions elements. In our experiment, we found the best results setting $\omega_{ATAC} = 1, \omega_{H3K27ac} = 0$ and $\omega_{H3k27me3} = 0$ in Equation (4.2).

The set of edge types in the graph is denoted as $\mathcal{T}_\mathcal{E} = \{S_{BP|T}, S_G, S_{C_1}, S_{C_2}, ..., S_{C_M}\}$. As explained before, $S_{BP|T}$ indicates the trend-aware base-pair proximity relationship explained above; S_G stands for the gene expression profile similarity between genes, and S_{C_1}, S_{C_2}, ..., S_{C_M} are the relationships that express the different cCRE activity time-series similarities. We aimed to combine all the edge types in $\mathcal{T}_\mathcal{E}$ to find a unique multi-semantic edge type that resembled gene expression regulation mechanisms. The process of combining edge types is represented in panel E of Figure 1 and is explained later in this section. To perform the combination of all edge types in $\mathcal{T}_\mathcal{E}$, we modeled the data as a heterogeneous graph and designed a graph-RL algorithm capable of extracting such a dependency between elements using prior biological knowledge and the data available. We explain the graph-RL algorithm we used to fulfil our objective in the next paragraph.

4.2.3 Graph Representation Learning

We used a graph-RL algorithm to find a unique feature space for the nodes of $\mathcal{G}$ that comprises most of the information encoded by the feature spaces contained in R. In our experiment, we adapt the encoder/decoder paradigm to a heterogeneous graph context. Our main objective was for the the spatial distribution of these points in the embedding space to reflect plausible gene-expression regulation mechanisms. To this end, we proposed to find a parametric encoding function, and in particular, we chose to implement (1.1) using a deep graph auto-encoder as explained in the next paragraph.

4.2.4 Deep Graph Auto-encoders

In this work, we proposed the use of the basic GNN model presented in [90]. In a homogeneous graph context, the basic GNN takes as input the node feature matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times m}$ and a pre-defined graph adjacency matrix denoted by $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$, and performs non-linear parametric operations with this information to produce $\mathbf{Z}$. In matrix notation, the operations made by the basic GNN model can be represented as follows:

$$\mathbf{Z} = \sigma(\mathbf{W}_{self}\mathbf{X} + \mathbf{W}_{neigh}\tilde{\mathbf{A}}\mathbf{X}) \tag{4.4}$$

where σ is a non-linear operator, the learnable parameter matrices $\mathbf{W}_{self}$ and $\mathbf{W}_{neigh}$ are responsible for combining the features of each node with the features of an aggregation over its neighborhood, and $\tilde{\mathbf{A}}$ is the symmetric-normalized adjacency matrix of $\mathcal{G}$:

$$\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{A})\mathbf{D}^{-\frac{1}{2}}$$

where $\mathbf{D}$ is the degree matrix of $\mathbf{A}$. In our experiment, σ was the rectified linear unit (ReLU) [3]. Notice that the embedding vector z of a node feature vector x is a differentiable function of x.

We defined a parametric "architecture" for the encoding function using (4.4), but still needed to define $\mathbf{A}$ and $\mathbf{X}$, to drive the optimization of the parameters of f to minimize (1.3). The initialization of these matrices can follow multiple strategies. For example, in the neural message passing framework [74], given a homogeneous graph context, one generally constructs $\mathbf{X}$ stacking the original feature vectors as the rows of such a matrix. However, it is not the only valid strategy. In our heterogeneous graph context, we decided to initialize $\mathbf{X}$ as the identity matrix because we had multiple original feature matrices. We were confident to use the identity matrix to initialize $\mathbf{X}$ because we carefully initialized the adjacency matrix $\mathbf{A}$ combining the information of every relationship type in our graph, as we will explain in the next paragraph. This implies that the current published expert model performs a transductive embedding of the underlying training entities, but we can change this initialization strategy to converge to inductive embedding of new gene expression and cCRE activity profiles.

Finally, the similarity measure that the decoder is meant to predict needs to be specified to implement (1.2). In this paper, we proposed the use of the decoding similarity function proposed by [137] to converge into a clustering-friendly embedding, i.e., an embedding with dense and easily identifiable clusters. In the next paragraph, we give a brief explanation of the algorithm in [137], which performs graph-RL on a single-feature space. For a more detailed exposition of the work of Xuelong *et al.*, the reader is referred to the original paper [137].

4.2.5 Adaptive-Sparsity Graph Generative Model

In this work, we proposed the use of AdaGAE [137], which is a state-of-the-art method for RL and clustering through graph modelling of a dataset. The whole AdaGAE algorithm is wrapped in a gray rectangle in Figure 4.1. This algorithm is defined in a homogeneous data context. In other words, given a unique original feature matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times m}$, the objective was to produce a low-dimensional embedding matrix $\mathbf{Z} \in \mathbb{R}^{|\mathcal{V}| \times d}$ where $d \ll m$. Authors in [137] used a deep GAE to embed high-dimensional datasets in a graph-like, low-dimensional format. Xuelong *et al.* generated a sparse graph as a function of the Euclidean distances between the original feature vectors using a k-nearest neighbors [125] (k-NN) model.

The main novelty of AdaGAE is the adaptiveness in the sparsity parameter k. The graph auto-encoder is, in fact, iteratively optimized with respect to various objective functions like (1.3), and each objective function is constructed using a different sparsity parameter. With a custom decoding function, such a dynamic model leads to a sparse graph. In other words, they produce a clustering-friendly embedding of the original data.

In other words, at each iteration, $l \in \{0, 1, 2..., L-1\}$, a sparse and weighted graph is generated. In this graph, the nodes correspond to the samples of the original dataset. Instead, the weights of the edges are inversely proportional to the Euclidean distances between the original feature vectors. Specifically, at iteration l, a graph is generated in which the weight of the link between nodes v_i and v_j is denoted as $p^l_{i,j}$ and is a function of a sparsity parameter k^l:

$$p^l_{i,j} = \left(\frac{d_{i,k^l+1} - d_{i,j}}{\sum_{j=1}^{k^l}(d_{i,k^l+1} - d_{i,j})}\right)_+ , \ \forall i, j \in |\mathcal{V}| \tag{4.5}$$

where $(\cdot)_+ = \max(., 0)$ is the rectifier operator, $d_{i,j}$ denotes the Euclidean distance between the feature vectors of v_i and v_j, and d_{i,k^l+1} denotes the distance between the feature vectors of v_i and its $(k^l + 1)$-th nearest neighbour.

Notice that, in (4.5), the parameter k^l might have a different value at each iteration. Such a parameter induces the sparsity of the generated graph: for each node v_i, only up to k^l elements will have a non-zero weighted link with that node. Thus, at each iteration l, AdaGAE generates a k^l-sparse graph. Notice also that the function (4.5) is a discrete probability density function (PDF):

$$\mathbf{P}^l_i : (v_j) \rightarrow p_{i,j} \in [0, 1]$$
$$\sum_{j \in |\mathcal{V}|} p^l_{i,j} = 1$$

where a different PDF $\mathbf{P}^l_i$ is defined for each single node in the dataset. Thus, at each iteration l, AdaGAE generates a probability distribution family $\mathbf{P}^l$:

$$\mathbf{P}^l = \{\mathbf{P}^l_0, \mathbf{P}^l_1, ..., \mathbf{P}^l_{|\mathcal{V}|}\}$$

where each PDF in $\mathbf{P}^l$ contains information about the weights of the links between one node and the rest of the nodes in the graph. Authors in [137] proposed to re-initialize the adjacency matrix of the auto-encoder at each iteration using the family $\mathbf{P}^l$. In other words, at each iteration l, AdaGAE stacks the different distributions $\mathbf{P}^l_i, \forall i \in |\mathcal{V}|$ to form $\mathbf{A}$.

The embedding matrix $\mathbf{Z}$ produced by AdaGAE was iteratively optimized. We denoted with $\mathbf{Z}^l$ the embedding matrix after iteration l. The embedding vector of node v_i after iteration l, instead, coincides with the i-th row of $\mathbf{Z}^l$ and is denoted as $\mathbf{z_i^l}$. At each iteration, AdaGAE proposes to implement the decoding function in (1.2) as a function of the embeddings of the previous iteration:

$$\mathbf{Q}^l(i,j) = q_{i,j}^l = \frac{\exp -\hat{d}_{i,j}^{l-1}}{\sum_{j=1}^{|\mathcal{V}|} \exp -\hat{d}_{i,j}^{l-1}} \tag{4.6}$$

where $\hat{d}_{i,j}^{l-1}$ denotes the euclidean distance between the embedding vectors $\mathbf{z}_i^{l-1}$ and $\mathbf{z}_j^{l-1}$. Notice that this encoding function is also a point-wise probability distribution function:

$$\begin{aligned} &\mathbf{Q}_i^l : (v_j) \rightarrow q_{i,j} \in [0,1] \\ &\sum_{j \in |\mathcal{V}|} q_{i,j}^l = 1 \end{aligned}$$

and that we can group every PDF $\mathbf{Q}_i^l, \forall i \in |\mathcal{V}|$ in a PDF family $\mathbf{Q}^l$ as we did with $\mathbf{P}^l$.

At each iteration l, AdaGAE proposes to use the cross-entropy loss of $\mathbf{Q^l}$ with respect to $\mathbf{P^l}$ to implement (1.3). Explicitly, the reconstruction loss that AdaGAE seeks to minimize is the following:

$$\begin{aligned} &\mathcal{L}^l = \sum_{i \in |\mathcal{V}|} \mathrm{H}(\mathbf{P}_i^l, \mathbf{Q}_i^l) \\ &\text{where:} \\ &\mathrm{H}(\mathbf{P}_i^l, \mathbf{Q}_i^l) = -\sum_{j \in |\mathcal{V}|} p_{i,j}^l \cdot \log q_{i,j}^l \end{aligned} \tag{4.7}$$

Xuelong *et al.* relied on the graph convolutional network (GCN) model [282] to implement the encoder of their deep GAE. However, in this work, we did not used the self-loop aggregation mechanism of the GCN to avoid oversmoothing of the embeddings. We represented the iterative computation of (4.5) and (4.6) and the iterative optimization of (4.7) in panel E of Figure 4.1.

Notice that the main dynamic criterion that causes the objective functions to change is the sparsity parameter k^l. By doing so, AdaGAE considers high-order neighborhoods of each node to construct the final embedding matrix. In other words, AdaGAE is said to "exploit the high-level information" present in the data.

Different embeddings for various iterations are plotted in panel G of Figure 4.1. In those plots, one can see the embedding of a small set of genes and cCREs extracted from the original mouse heart dataset. Large points represent genes, while cCREs are the small points. Point colors, instead, correspond to single-modality or homogeneous cluster assignments, i.e., cluster assignments of genes and cCREs, taking into account their corresponding feature spaces separately. Note that the embedding turns more cluster-friendly at each iteration. In other words, the density and separation of clusters are increased at each iteration, and clusters can be caught visually in the final embedding in panel H of Figure 4.1. In AdaGAE, the sparsity parameter initialization, k_0, the increment of this parameter from iteration to iteration, δk, and the number of iterations, L, regulate the number of clusters in the final embedding. In our experiment, we tested various parameter configurations and found eight significant GERM clusters with $L = 11$, $k_0 = 350$, $\delta k = 25$ given our mentioned data-set.

Notice that the original work of Xuelong *et al.* defined (4.5) as a function of the Euclidean distances over a unique feature space. We extended the AdaGAE framework to deal with multiple node feature spaces. In particular, at each iteration l, we combined the Euclidean distances of multiple node feature spaces - the gene expression and the cCRE activity time-series - to generate a unique distance function that is given to (4.5) to generate $\mathbf{P}^l$. In the next paragraph, we explain how we combined the feature spaces to adapt AdaGAE to heterogeneous graph-RL.

4.2.6 Extending AdaGAE to Heterogeneous Networks

As explained previously, we modeled the whole set of gene expression and cCRE-activity data as a heterogeneous graph. We proposed to extend the work in [137] to heterogeneous graph-RL. We ran the same process described in the previous paragraph, with some differences. The first difference was the construction of the reference distance function to feed to (4.5). At each iteration l, we combined the Euclidean distances defined in each one of the original feature spaces to form a unique distance function $\mathbf{D}^l$ that generated the sparse distribution family $\mathbf{P}^l$.

Recall that $R = \{G, C_1, C_2, ..., C_M\}$ is the set of feature spaces defined, where G is the gene-expression feature space and $C_1, C_2, ..., C_M$ are the cCRE activity feature spaces, for example ATAC-seq, H3k27ac, aH3k27me3, etc. At each iteration l, for each feature space $r \in R$, we computed the Euclidean distances between the original feature vectors in r, scaled these distances into the interval $[0, 1]$, and denoted them with $d^r_{i,j}$, $\forall i, j \in |\mathcal{V}|$. Notice that the Euclidean distances between the original feature vectors were only defined between same-class elements: The Euclidean distances in the gene-expression feature space G, were defined between genes, and the distances in $C_1, C_2, ..., C_M$, only between cCREs. Consequently, we set the distance between different-class elements to the maximum value, i.e., 1, for each feature space. Finally, we create a custom linear combination of these distance functions to form a unique distance function $\hat{\mathbf{D}}^l$:

$$
\begin{aligned}
&\hat{\mathbf{D}}^l : (v_i, v_j) \rightarrow \hat{d}^l_{i,j} \in \mathbb{R}, \forall i, j \in |\mathcal{V}| \\
&where: \\
&\hat{d}^l_{i,j} = \begin{cases} 1 - \alpha^l_G \cdot (1 - d^G_{i,j}), & \text{if } v_i, v_j \in \mathcal{V}_{genes} \\ \frac{\sum_{m=0}^{M} 1 - \alpha^l_{C_m} \cdot (1 - d^{C_m}_{i,j})}{M}, & \text{if } v_i, v_j \in \mathcal{V}_{cCREs} \\ 1, & \text{otherwise} \end{cases}
\end{aligned} \tag{4.8}
$$

where $\mathcal{V}_{genes}$ is the set of genes and $\mathcal{V}_{cCREs}$ is the set of cCREs, $d^G_{i,j}$ is the Euclidean distance between the expression profile of gene i and gene j, $d^{C_m}_{i,j}$ is the Euclidean distance between the C_m activity profiles of cCREs i and j, and α^l_G and $\alpha^l_{C_m}$ are fixed parameters that indicate the importance weights of the distances of gene-expression and cCRE-activity at iteration l, respectively.

At the beginning of the l-th iteration, the Euclidean distances between the embedding vectors of $\mathbf{Z^{l-1}}$ are computed and denoted by $\mathbf{D}^{l-1}_Z$. We combined such information with $\hat{\mathbf{D}}^l$ defined in (4.8), to generate the unified distance function, $\mathbf{D}^l$ as follows:

$$
\mathbf{D}^l = \frac{\hat{\mathbf{D}}^l + \hat{\mathbf{D}}^{l-1}_Z}{2} \tag{4.9}
$$

where $\hat{\mathbf{D}}^{l-1}_Z$ is a *weighted version* of $\mathbf{D}^{l-1}_Z$:

$$
\hat{\mathbf{D}}^l_Z = 1 - \alpha^l_Z \cdot (1 - \mathbf{D}^{l-1}_Z) \tag{4.10}
$$

In (4.10), α^l_Z is the fixed parametric importance weight of the Euclidean distances in $\mathbf{D}^{l-1}_Z$. Notice that, in (4.9), $\hat{\mathbf{D}}^{l-1}_Z$ allows to exploit the high-level information in the data and $\hat{\mathbf{D}}^l$ helps to reduce the loss of information with respect to the original feature spaces at each iteration. In other words, in our extension of the AdaGAE setting, the sparsity is not the only thing that changes from iteration to iteration: also the parametric combinations of the original feature matrices change. As we do not want to loss the information encoded in the manifolds that learn from each one of these iterations, we create a sort of offline recursive manifold learning, induced by (4.9). This setting can be conceived also as an offline parametric skip connection mechanism that permits our model to learn abstract features taking into account various parametric combinations of the original feature spaces, progressively manipulating the spatial distribution of the points in the manifold and inducing a progressive cohesion and separation of GERM clusters.

At each iteration $l, \forall l \in [0, L-1]$, we computed $\mathbf{D}^l$ using (4.9). This computation is represented in panel E of Figure 4.1. We then used $\mathbf{D}^l$ and the corresponding sparsity parameter k^l to compute a connectivity distribution family $\hat{\mathbf{P}}^l$ using (4.5). Notice that this distribution family was generated using only the same-class relation types in $\mathcal{T}_\mathcal{E} - \{S_{BP|T}\}$. In other words, we only used the node feature spaces in R to create $\hat{\mathbf{P}}^l$. We needed to add the trend-aware base-pair proximity relationship $S_{BP|T}$ defined in (4.3) to compute the definitive distribution family $\mathbf{P}^l$ that takes into account all information we have:

$$\mathbf{P}^l = \hat{\mathbf{P}}^l + \omega^l_{BP} S_{BP|T} \tag{4.11}$$

where ω^l_{BP} is a parametric importance weight of the base-pair proximity information $S_{BP|T}$ during iteration l.

After computing $\mathbf{P}^l$, we computed Q^l using (4.6), and optimized the embedding matrix $\mathbf{Z}$, minimizing a reconstruction loss of the form (1.3). AdaGAE proposed the use of (4.7) to implement the reconstruction loss. In our work, however, we propose to extend such a loss function to better separate elements that are distant in the original feature spaces, as explained in the next paragraph.

4.2.7 Regularization of the Loss Function

The minimization of (4.7) has the same effect of an attractive force that tends to collapse points in the embedding space, reducing the distances between z_i and z_j proportionally to $p^l_{i,j}$. We can observe such an effect analyzing the derivative of (4.7) with respect to $\mathbf{Q}l$:

$$\frac{\partial \mathcal{L}^l}{\partial \mathbf{Q}^l} = -\frac{\mathcal{K}}{\mathbf{Q}^l} \leq 0 \tag{4.12}$$

where $\mathcal{K}$ is a constant term that derives from the current value of $\mathbf{P}^l_i$, which do not depends on $\mathbf{Q}^l_i$. Looking at (4.12) we realize that, in order to minimize $\mathcal{L}^l$, we should maximize $\mathbf{Q}^l_i$, because of the inverted sign of the gradient. Maximizing $q^l_{i,j}$, however, requires -by definition- minimizing $\hat{d}^l_{i,j}$, which is equivalent to push the elements closer to each other in the space. Moreover, we notice that the absolute value of the gradient is proportional to the distances in the manifold, which implies that, given a fixed distance in the original feature space, the attractive force is proportional to the distance of the points in the embedding space. The farther they are from each other, the strongest the force that pushes them close to each other.

More specifically, the loss in (4.7) turn to effectively penalize the distance divergences in the original local environment of v_i:

$$\lim_{d_{i,j} \to 0} \mathbf{P}_i = k, \text{ s.t. } 0 \ll k \leq 1$$
$$\text{and} \lim_{\hat{d}_{i,j} \to \infty} \mathbf{Q}_i = 0^-$$
$$\text{thus} \lim_{\hat{d}_{i,j} \to \infty, d_{i,j} \to 0} \mathcal{L} = +\infty$$

Notice that we have removed the dependence of l in the formulas for ease of reading, and it is naive that these relations hold in every iteration $l \in [0, L-1]$. Recall that the family of distributions $\mathbf{P}$ induces a sparse graph where each node is connected to a maximum of k neighbours. Unfortunately, even in such local context, the loss in (4.7) lacks to penalize the divergences in the peripheries of every node v_i:

$$\lim_{d_{i,j} \to d_{i,k+1}} \mathbf{P}_i = 0,$$
$$\text{and} \lim_{\hat{d}_{i,j} \to 0} \mathbf{Q}_i = 1$$
$$\text{thus} \lim_{\hat{d}_{i,j} \to 0, d_{i,j} \to d_{i,k+1}} \mathcal{L} = 0$$

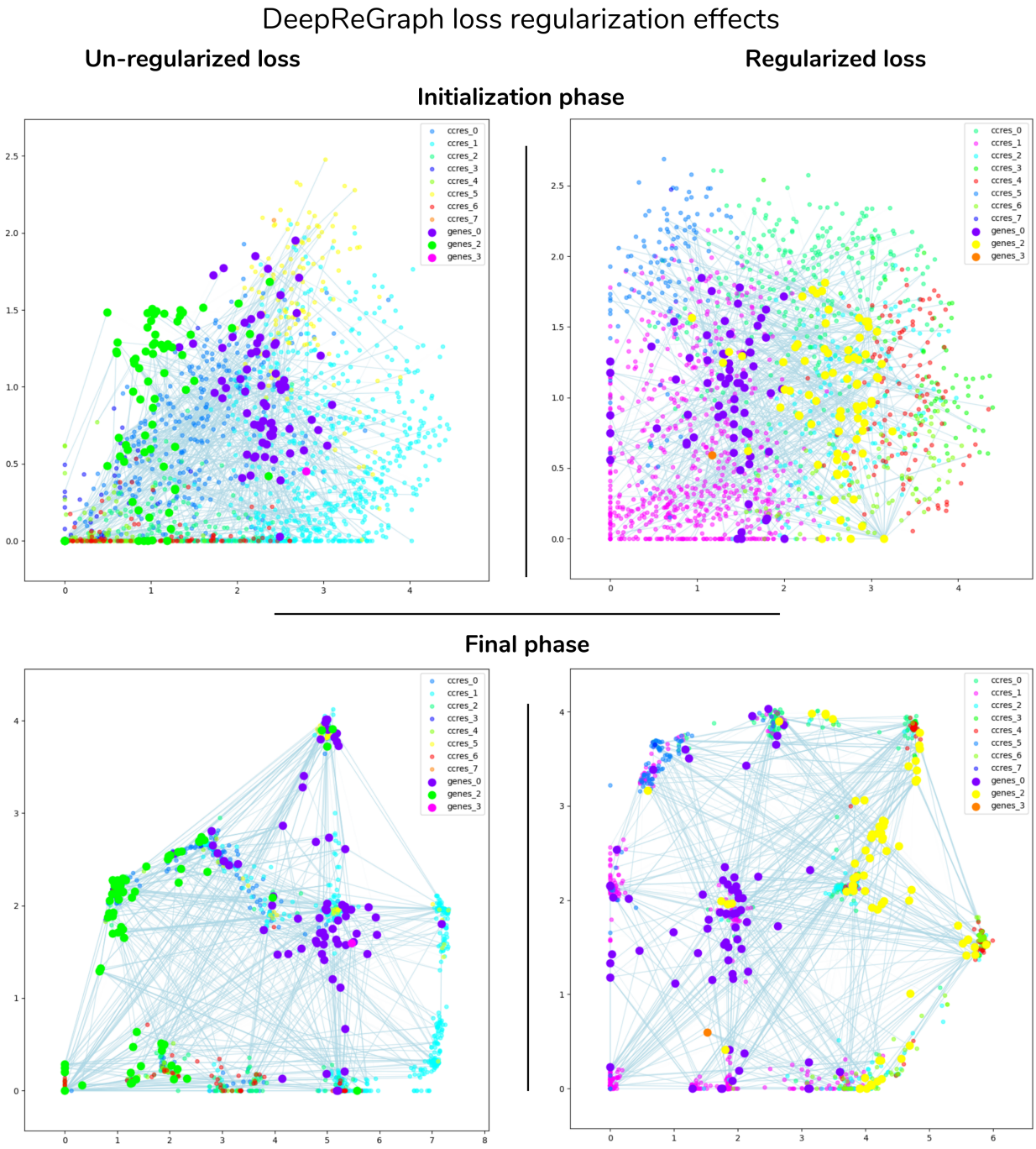

Figure 4.2. The $\mathbf{Z^1}$ and $\mathbf{Z^L}$ manifolds learnt by two runs of our algorithm (given that $\mathbf{Z}$ is bi-dimensional). The left-hand side shows the embeddings derived from the usage of the *simple* cross-entropy in (4.7), while the right-hand side shows the embeddings produced by the binary cross-entropy loss in (4.13). The $\mathbf{Z^1}$ embeddings occupy the upper row and the $\mathbf{Z^L}$ embeddings are below for both loss configurations. In the right-hand side, the regularization term was used only for the first iteration that produces $\mathbf{Z^1}$. The subsequent iterations optimize only (4.7). In this figure, node colors represent the homogeneous clusters of genes and cCREs, and the edges map the trend-aware distance score in (4.3).

Moreover, if we consider the global environment of v_i, we have that, by definition of $\mathbf{P}_i$, (4.7) leads to errors in the manipulation of the position of the most distant neighbors of such node:

$$d_{i,j} > d_{i,k+1} \implies \mathbf{P}_i = 0, \implies \mathcal{L} = 0, \ \forall \hat{d}_{i,j} \in \{0, \infty\}$$

A similar analysis for the lack of awareness of the global spatial distribution of information can

be found in [192], regarding the differences between two well-known manifold learning algorithms: UMAP [160] and tSNE [258]. Inspired by [160], in this work, we added a regularization term that acts as a repulsive force. To create this force, we take into account the distributions defined by $(1-\mathbf{Q}_i^l)$ and $(1-\mathbf{P}_i^l)$, and add to (4.7) the cross-entropy of the $(1-\mathbf{Q}_i^l)$ with respect to $(1-\mathbf{P}_i^l)$. At the end, our new loss function is defined as:

$$\mathcal{L}_{BCE}^l = \mathcal{L}^l + \hat{\mathcal{L}}^l = \sum_{i\in|\mathcal{V}|}\sum_{j\in|\mathcal{V}|}[\psi_A^l \cdot \mathrm{H}(\mathbf{P}^l,\mathbf{Q}^l) + \psi_R^l \cdot \mathrm{H}(1-\mathbf{P}^l, 1-\mathbf{Q}^l)] \tag{4.13}$$

where ψ_A^l and ψ_R^l are fixed parametric importance weights of the attractive and repulsive term during iteration l, respectively. Notice that the regularization term is denoted by $\hat{\mathcal{L}}^l$.

The repulsive force induced by $\hat{\mathcal{L}}^l$ pushes $\hat{d}_{i,j}^l$ in the manifold to increase proportionally to the inverse of $p_{i,j}^l$. In other words, we pushed elements away from each other proportionally to their distance $\mathbf{D}^l$ in (4.9). Again, we can observe such an effect analyzing the derivative of $\hat{\mathcal{L}}^l$ with respect to $\mathbf{Q}_i^l$:

$$\frac{\partial\hat{\mathcal{L}}^l}{\partial\mathbf{Q}^l} = \frac{1-\mathbf{P}^l}{1-\mathbf{Q}^l} \geq 0 \tag{4.14}$$

where we can see that, to minimize $\hat{\mathcal{L}}^l$, we should also minimize $\mathbf{Q}^l$, which means augmenting the distances in the manifold $\hat{D}$. Moreover, we notice that, fixing $\mathbf{P}^l$, the absolute value of this derivative is inversely proportional to such distances in the manifold. In other words, given a fixed distance in the original feature space, the closer we are in the embedding space, the biggest the error in $\hat{\mathcal{L}}^l$:

$$\lim_{d_{i,j}\to d_{i,k+1}} \mathbf{P}_i = 0,$$
$$\text{and} \lim_{\hat{d}_{i,j}\to 0} \mathbf{Q}_i = 1$$
$$\text{thus} \lim_{\hat{d}_{i,j}\to 0, d_{i,j}\to d_{i,k+1}} \hat{\mathcal{L}} = +\infty$$

And this penalisation works not only in the peripheries of the k-sparse domain, but also in the global context:

$$d_{i,j} > d_{i,k+1} \implies \mathbf{P}_i = 0, \implies \hat{\mathcal{L}} = -\log(1-\mathbf{Q}_i)$$
$$\text{and} \lim_{\hat{d}_{i,j}\to 0} \mathbf{Q}_i = 1$$
$$\text{thus} \lim_{\hat{d}_{i,j}\to 0, d_{i,j}> d_{i,k+1}} \hat{\mathcal{L}} = +\infty$$

Notice that the union of the attractive and repulsive force is equivalent to the minimization of the binary cross-entropy of $\mathbf{Q}^l$ with respect to $\mathbf{P}^l$ as defined in [160]. The usage of the binary cross-entropy in the reconstruction loss can be thought as a contrastive learning setting.

Figure 4.2 shows the $\mathbf{Z^1}$ and $\mathbf{Z^L}$ manifolds learnt by two runs of our algorithm (given that $\mathbf{Z}$ is bi-dimensional). The left-hand side shows the embeddings derived from the usage of the *simple* cross-entropy in (4.7), while the right-hand side shows the embeddings produced by the binary cross-entropy loss in (4.13). The $\mathbf{Z^1}$ embeddings occupy the upper row and the $\mathbf{Z^L}$ embeddings are below for both loss configurations.

One can see that the main disadvantage of not using the contrastive regularization is the formation of some clusters that are less heterogeneous, in the sense that they tend to group mostly same class elements. This means we have no gains in GERM inferences and it resembles two separate homogeneous clustering tasks, the clustering of genes from one side and the clustering of cCREs from other side. We argue that the need for a regularization is mostly a consequence of the sparseness of the trend-aware distance score in (4.3). This feature space represents the unique relation between different-class elements in our graph, and should drive the positioning of elements of different classes together in the embedding space. However, due to its inherent sparsity, the trend-aware distance score is not enough to produce such a mixing effect in the manifold. The

repulsive force term instead, has a side-effect of mixing different-class elements in the manifold, and this effect helps initializing a manifold $\mathbf{Z^1}$ that is more prone to converge to a $\mathbf{Z^L}$ that resembles heterogeneous GERM clusters. In other words, We observe that a side effect of separating dissimilar same-class elements in the original space is to mix different-class elements in the embedding space. This mixing effect is and effective initialization strategy for the manifold that then is able to perform additional slight deformations in the embedding space to find the GERM clusters.

We iteratively optimized the parameters of our GNN-based encoder. At each iteration, we minimized (4.13) with a different set of parameters. After running the optimization for a predefined number of iterations, we converged to a clustering-friendly embedding where the clusters reflected plausible gene regulatory networks. We then ran k-means clustering on the embedding and analyzed the clusters. This clusterization is represented in panel H of Figure 4.1.

4.3 Results

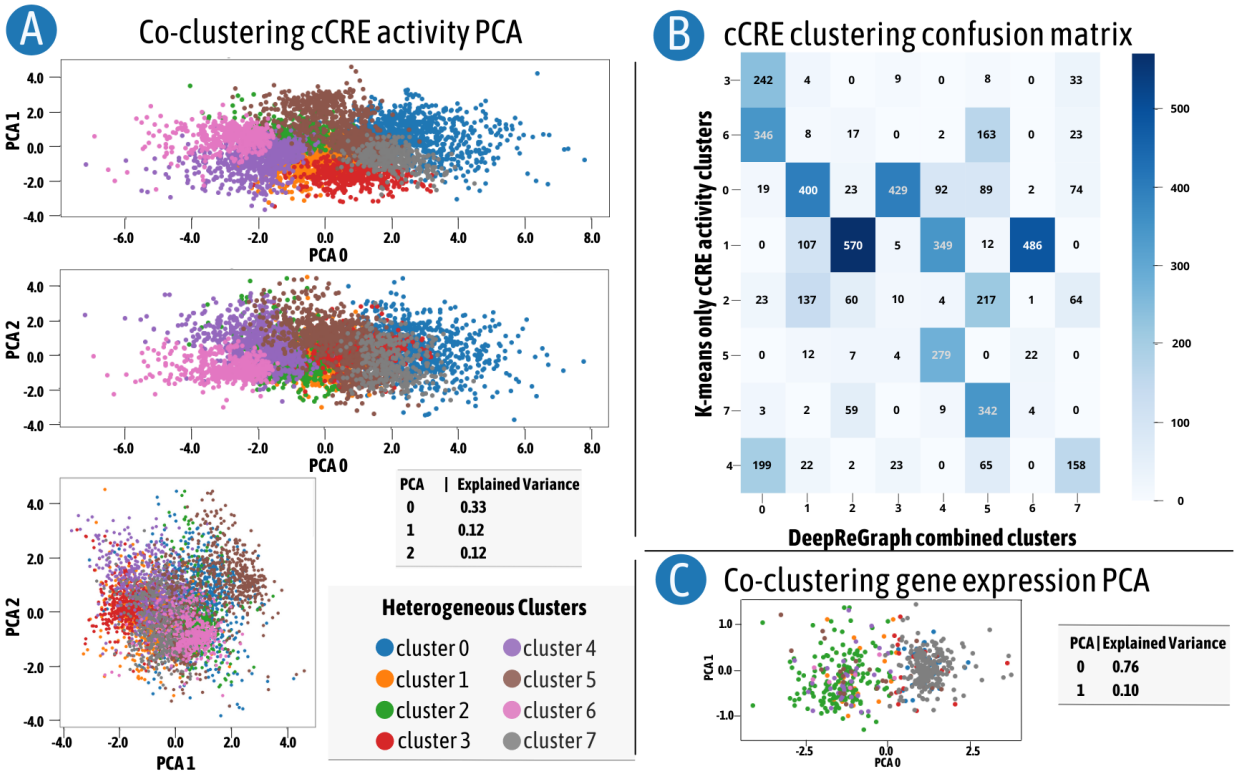

Figure 4.3. Intersections between single-modality k-means clusters and co-clusters induced by DeepReGraph. A) Principal component analysis (PCA) reduced dimension of candidate cis-regulatory elements (cCRE) profiles colored by the correspondent DeepReGraph cluster. B) Intersection between only-cCRE agglomerative clustering and cCREs extracted from DeepReGraph heterogeneous clusters. C) PCA reduced dimension of gene expression profiles colored by the corresponding DeepReGraph cluster.

DeepReGraph performs the co-clustering of genes and cCREs together. Consequentially, resulting clusters are heterogeneous in the class of elements they might contain. DeepReGraph helped us identify eight co-clusters when applied to developmental fetal mouse heart datasets. We assessed the quality of gene expression and cCREs clusters from a computational point of view. We also analyzed these clusters from a biological perspective as described below.

4.3.1 DeepReGraph generated high-quality co-clusters

We employed a principle component analysis (PCA) to visualize the cCRE clusters on dimension reduced plots as shown in panel A of Figure 4.3. This figure highlights a clear separation of clusters

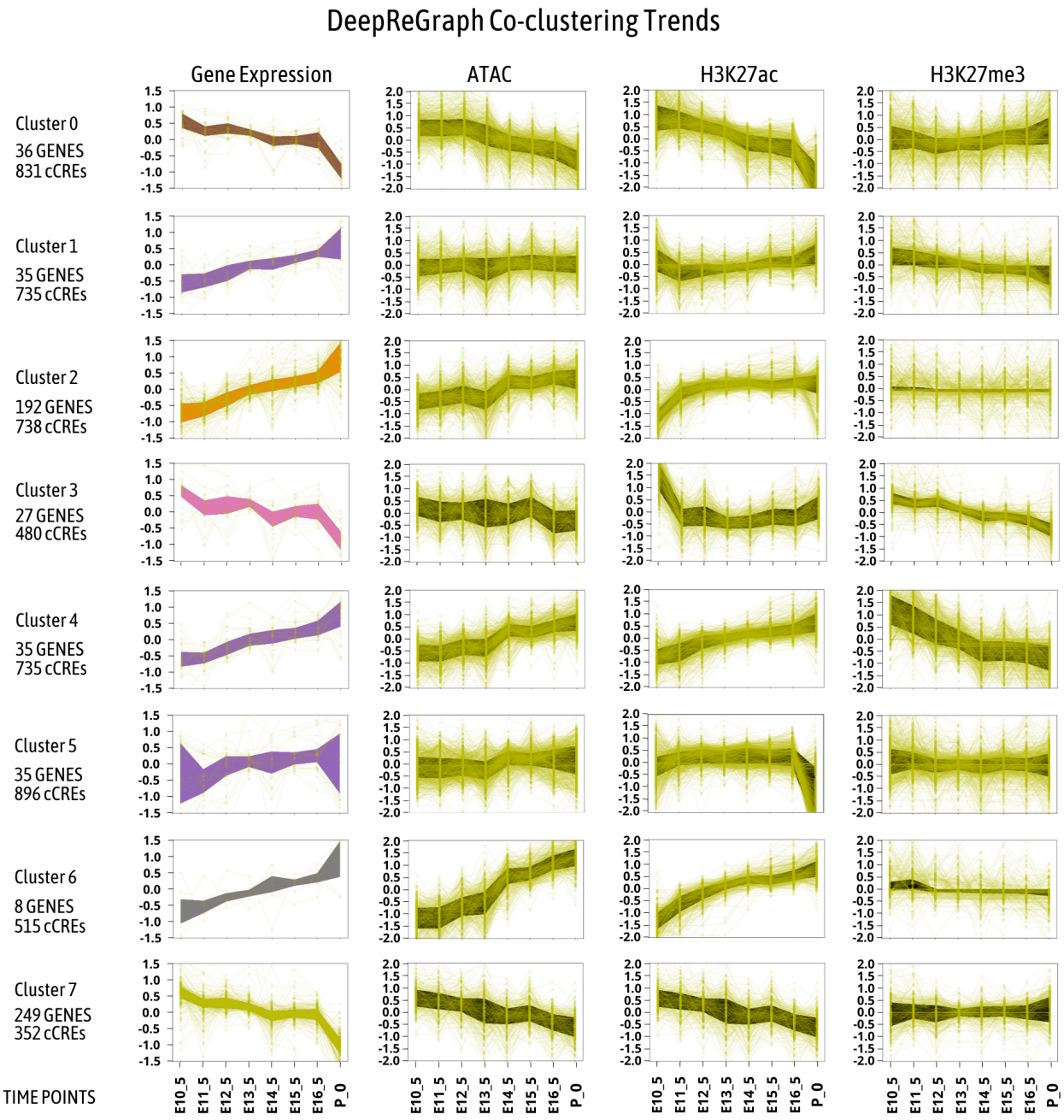

Figure 4.4. Co-clusters identified by DeepReGraph and the relative profile trends. The first column of plots contains gene expression time profiles, and the rest of the columns contain enhancer activity time profiles. The space between the first and third quartile for each plot was colored to better show the trend.

on $PC0$, $PC1$, and $PC2$ which explain the 57% of variability in the cCRE datasets. We also performed k-means clustering using $k = 8$. Panel B of Figure 4.3 shows a confusion matrix that compares cCRE clusters from DeepReGraph with the uni-modal clustering produced with k-means clustering. It is clear from this confusion matrix that the result of the multi-modal clustering of DeepReGraph was highly similar to the uni-modal clustering one. This similarity indicates that DeepReGraph clustering does take into account the same-class pattern similarities when defining clusters. We also visualized the PCA plot of gene expression data in panel C of Figure 4.3. PC0 by itself explains 76% of the variability in gene expression. Basically, gene expression has two distinct patterns throughout mouse fetal heart tissue development: genes with increasing expression and genes with decreasing expression. Interestingly, these two major patterns have been divided into

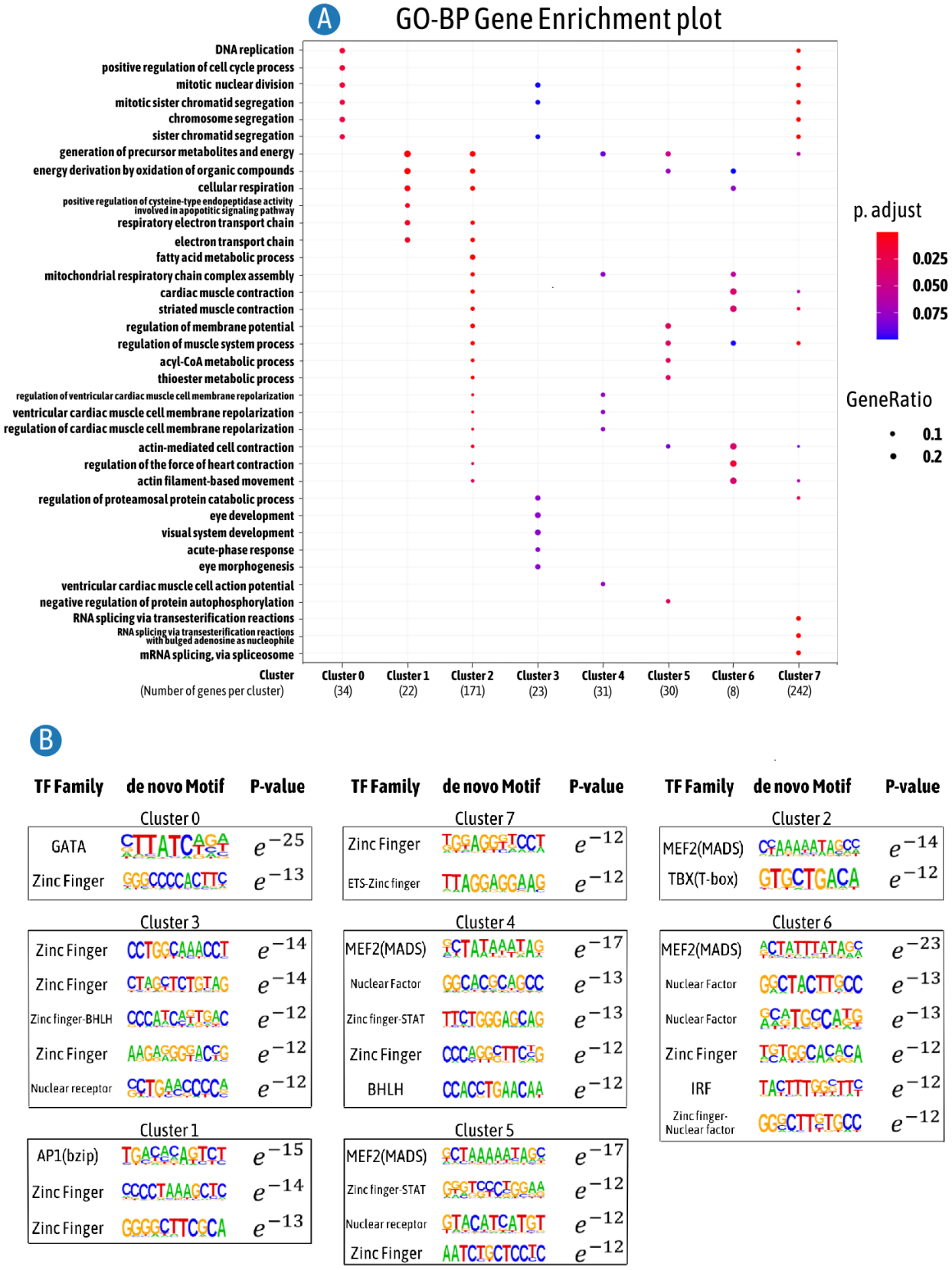

Figure 4.5. A) Enriched function of gene expression extracted from DeepReGraph clusters. B) Enriched transcription factor binding site motifs of cCRE for the same clusters.

sub-patterns based on the cCRE clusters that plausibly drive their regulation.

The clustered patterns produced by DeepReGraph are presented in Figure 4.4. In this figure, the y-axis contains the mean-centered values for gene expression in the left-most column, while mean-centered CRE features are in the last three columns. The mean value reduction process mentioned was done as follows: given a time-series vector $\mathbf{x} = [x_0, x_1, ..., x_t]$, with a mean value $\hat{x} = \frac{\sum_{i=0}^{t} x_i}{|\mathbf{x}|}$, the mean-reduced vector is $\tilde{\mathbf{x}} = [x_0 - \hat{x}, x_1 - \hat{x}, ..., x_t - \hat{x}]$. The x-axes of the plots instead correspond to the considered time-points of mouse fetal heart development. Notice that each cluster contains a set of genes and cCREs. The area between the first and third quartile is colored for each trend plot, to help visualize the trend of each cluster.

4.3.2 DeepReGraph revealed the regulatory signature of mouse fetal heart development

We investigated the enriched function of gene expression clusters and enriched transcription factor binding site motifs of cCRE clusters to decipher the general signature of mouse fetal heart development. Figure 4.5 summarizes these enrichment analysis. It clearly shows that all gene expression clusters have clear functional annotation and all cCRE clusters entail clear transcription factor binding site motifs.

In general, expression of genes related to cell proliferation functions decrease during mouse fetal development, while expression of genes related to heart functions increase. [279] However, if we look at the enriched terms for the detected gene expression clusters in panel A of Figure 4.5, and the pattern of gene expression in Figure 4.4, we can observe that the story is not so simple. Two of the largest gene expression clusters are cluster2 (192 genes) which represents the heart functional genes (enriched for heart contraction function), and cluster7 (249 genes) which represents the cell proliferation genes. The smaller gene expression clusters have more specific enriched functions. Considering the other gene expression clusters down-regulated during development, cluster0 was more enriched for DNA replication, while cluster3 was enriched for non-heart developmental processes. Similarly, the smaller gene expression clusters up-regulated during development gained specific functional enrichment: cluster1 was enriched in metabolic processes to generate energy for the heart to function. Cluster4 was enriched for ventricular cardiac muscle cell membrane repolarization, and cluster6 was enriched for heart contraction. Cluster5 contained genes enriched for regulation of muscle system process. Here, co-clustering of gene expression and cCREs enabled us to derive smaller and more specific clusters. Otherwise, as it is clear from Figure 4.3 panel C, the smaller clusters of gene expression with more focused functions could not be deduced from gene expression alone.

Multi-modal clustering can also describe how cCREs can drive gene expression during development. First, smaller gene expression clusters gained more cCREs per genes, as Figure 4.4 shows. Secondly, cCREs with different epigenetic patterns have been linked to similar pattern of gene expression, as can be seen also in Figure 4.4. For example, in the major cluster with up-regulated genes during development, i.e. cluster2, the trends of ATAC-seq and H3K27ac increased as expected. Similarly, for cluster7 which entails a large set of down-regulated genes throughout development, the trend of ATAC-seq and H3K27ac decreased. However, the H3K27me3 pattern for both cluster2 and cluster7 showed no changes in Figure 4.4. We can observe polycomb removal events in cluster1, cluster3, and cluster4 as H3K27me3 levels decreased in these clusters.

Multi-modal clustering can further clarify how gene expression changes through development, as cCRE clusters exhibited clearly enriched motifs in panel B of Figure 4.5. These enriched motifs can prioritize the candidate transcription factors which drive the development. For example, the Myocyte enhancer factor 2 (MEF2) motif was enriched in the clusters related to up-regulated genes. The only exception was cluster1, for which the Activator protein 1 (AP1) motif was enriched. However, we can assume that MEF2 binding transcription factor, and more probably the Myocyte enhancer factor 2C (MEF2C), was the major transcription factor which caused the increase in gene

expression values. Other binding site motifs were been enriched for the clusters entailing temporally up-regulated genes in Figure 4.5 panel B. Interestingly, these motifs were highly cluster-specific. A similar dynamic was seen for clusters with temporally decreased gene expression. Although zinc finger motifs were enriched in these clusters, the GATA motif was only enriched in cluster0 and the basic helix-loop-helix (BHLH), nuclear receptor motifs are only enriched in cluster3.

4.4 Discussion

This study introduced a novel method called DeepReGraph to perform multi-modal clustering of gene expression and cCREs. DeepReGraph allows a cluster-friendly embedding, where clusters contain genes and CREs and tend to identify gene expression regulation mechanisms. Interestingly, the results of multi-modal clusters derived by DeepReGraph for cCRES were highly similar to the uni-modal clustering using k-means. However, DeepReGraph generated gene expression clusters that could not be derived by using gene expression data alone. Such a result might be expected if we consider cCRE and gene expression changes in a "cause and effect" manner. cCREs are part of the regulatory network and are among the driving causes of alternation in gene expression. Therefore, we can expect cCRE uni-modal clustering to be similar to co-clustering gene expression and cCREs together. However, gene expression is controlled by cCREs. In mouse fetal heart development, we have shown that similar gene expression (similar effect) can be divided into different clusters based on the controlling cCREs. This result shows the added values of the multi-modal clustering method to understand the signature of development.

Developmental regulatory networks can be straightforwardly modeled as heterogeneous graphs. The main reason for such a claim is that they are made of two distinct classes of elements (genes and CREs) whose interaction tends to be highly correlated with multiple features like gene expression, base-pair distance, and cCRE activity. Modeling such regulatory networks as heterogeneous graphs is key to using graph-RL algorithms to converge to low-dimensional embeddings for such systems. The spatial distribution of nodes in the embedding might resemble complex relationships between nodes.

We undertook the challenge of converging to an embedding where gene expression regulation mechanisms are easily identifiable. To do so, we created our own heterogeneous graph-RL algorithm by carefully designing an extension of AdaGAE [137]. First, we designed a dynamic combination schema of multiple node feature spaces to create a unique node feature space. This unification was a necessary step to adapt AdaGAE to a heterogeneous graph scenario. We also created a repulsive force by extending the loss function in [137]. This repulsive force has proven essential to separate elements with different gene expression or activity trends. Third, we introduce a trend-aware regularization of the base-pair distance relationship between nodes. This regularization proved essential to produce more compact clusters. The resulting schema is responsible for producing a clustering-friendly embedding space that sheds light on regulatory mechanisms.

In this work, we extended the algorithm presented in [137] to produce an algorithm capable of embedding a heterogeneous graph into a low-dimensional, easy-to-cluster embedding. Other approaches to heterogeneous graph embedding that we could further investigate exist [236]; for example, the relational graph convolutional networks (RGCN) [229]. Such a model implies a greater number of parameters, and various parameter-sharing approaches have been proposed, some of them making use of the attention mechanism [153, 240]. We could further investigate attention-based prioritization of nodes and relationships for learning embeddings [329]. We have however to mention that most of the het-graph-rl algorithms focus on facing complexities like heterogeneity in the types of features, appart of scalability issues by leveraging batch gradient descent on GNNs. These particular problems are not present in our current parameter setting, so we could also study the applicability of other graph-rl algorithms for homogeneous graphs. For doing this we should design hand-crafted feature engineering to feed the models with a unified feature space. Comparing with other graph-RL algorithms in general could be useful to investigate on the results of adding, for

example:

1. Other types of neighborhood message aggregation. For example neighbour set pooling [309] or Janossy pooling [178],

2. Leverage of neighbour-wise attention mechanisms [260].

3. Architectural settings that resemble online skip connections [205] and recurrent gated embedding updates [139] -notice that we have used these mechanisms in an offline fashion.

If we consider the production of a unified embedding of heterogeneous data as a first step, we could conceive other offline clustering algorithms. The clustering-friendly embedding we present resembles a differentiable version of agglomerative clustering. However, other algorithms like the ones in [293, 221] use a differential expectation-maximization schema, where a distance-to-centroid loss is minimized to reach final embedding with compact clusters. Consequently, the use of different deep clustering approaches should be further investigated.

Regulatory networks and gene regulation are dynamic processes. Therefore, temporal datasets can potentially describe them better than static ones. However, initial efforts to associate regulatory networks and chromatin states to the gene expression were based on limited data. ChromHMM [60] is the most used method to assign states of the chromatin to genes. However, with the advent of large temporal datasets like ENCODE, which we used in this paper, it is possible to move beyond a static view of regulatory networks and gene expression. This study used chromatin accessibility, H3K27ac, and H3K27me3, three well-known epigenetic markers with a well-characterized effect on gene expression. This framework can be further expanded to other epigenetic markers. Such an expansion could have two main advantages. The first advantage is the potential improvement of clustering quality. The second consists in better deciphering the combinatorial trends of epigenetic changes and their effects on gene expression dynamics.

4.5 Main contribution of Deep Learning

We can think of the mathematical rules that we have leveraged to compute the various combinations of the original feature spaces and their regularization in terms of sparsity, as a compound deterministic algorithm. At the core of the algorithm we have a mathematical box that receives in input a pair of elements: a gene and a cCRE. The output of this inner-algorithm could be seen as the probability of existence of a regulatory mechanism between these candidates. The outer algorithm instead rates the significance of various candidate associations that share one element (the gene or the cCRE). This algorithm provides a set of signals that traverse the gene-CRE graph and the spectra of these signals potentially give us information about certain gene regulatory mechanisms among the analyzed elements. Having in mind this solution setting, we can say we leveraged deep graph neural networks with the above-mentioned mathematical rules acting as inductive biases for these. Such a setting has helped us in three ways:

- First, it has helped us to augment the scalability of our algorithm to bigger graphs, in terms of computational and memory costs. We perform the embedding and the message passing in parallel, drastically reducing the number of computations we need to perform.

- Second, we augment the flexibility or smoothness of the signals that the rigid feature preprocessing gives in input. This is because we use parameter-sharing architectures that mimic the convolution operation on the graph. Moreover, the parameter of our graph convolutional filters are optimized using a loss function that takes into account the whole data.

- The main advantage of deep learning however is the inductive representation learning. Inductive learning is a consequence of the parametric setting, and it permits us to confidently apply the het-graph-RL machinery to new data assuming it comes from the same underlying

distributions. As explained in paragraph 4.2.4, with respect to the parameter setting we used in the present experiment, we could enable the learning of inductive GERM inference machinery simply changing the initialization of our deep ANN architecture with, for example, a simple concatenation of the original feature vectors.

4.6 Work Limitations

We now present two main limitations of the presented work. The first one is inherited by the absence of data, which took us to adopt a linear similarity kernel rather than more powerful probabilistic approaches. The second limitation derives from the biological inductive biases we have injected into the RL algorithm. We argue that these inductive biases may reduce the information we could potentially find in the current dataset.

4.6.1 Limitations of the euclidean distance kernel

One of the weaknesses of the current algorithm is that it is oriented to extract gene regulatory relationships using low-scale data. The data pre-processing depicted in paragraph 4.2.1 produced one scalar gene expression value per time-point per gene and one scalar activity value per cCRE marker per time-point. In other words, we have used short time-series data. We claim that more powerful GRN inferential methods could be addressed if we had the availability to use experiments that contain high-scale sets of samples for each time-point. Powerful non-linear dependence finding frameworks that rely on information theory kernels [28], auto-regressive methods [165], or probabilistic frameworks [227] have been applied in past to discover gene-gene regulation interactions on medium or high-scale single-cell gene expression data. Taking into account various temporal high-scale single-cell experiments, we could leverage similar methodologies to learn node representations. The mentioned kernels could be at the heart of a powerful inductive Deep GAE that could plausibly discover other types of gene regulatory associations.

Moreover, we have discovered many-to-many relations that could explain a *one-hop* regulatory relationship between various cCREs and genes. However, more complex regulatory relations exist -e.g. indirect regulation- which multi-hop causality relationships could map. Our algorithm has grouped the elements that are related by some correlation in their cCRE activity and gene expression patterns. But we have not specified whether the relations between these elements are direct or somehow indirect. Neither were we able to determine the order in which such causality could exist. This is the main limitation inherited by the similarity kernel we have used, which is inversely proportional to the euclidean distance and thus, proportional to the Pearson correlation. Gene-gene regulatory network inference methods that rely on the Pearson correlation are called Correlation Networks [312]. These methods are agnostic to causality. More sophisticated probabilistic modeling like Bayesian networks [98] or differential equations [159] could improve fostering causality-aware gene-gene expression regulatory mechanisms. In this work, a basic mapping for GRN inference is presented with respect to probabilistic models.

4.6.2 Limitations of the specific biological inductive bias

In this chapter, we have studied diverse parametric combinations of a set of primitive feature spaces to form a set of multi-semantic combination feature spaces. The multi-semantic spaces served, in turn, as a generative probability distribution for the edges of a set of abstract homogeneous graphs. But we did not learn the parameters that created the mentioned combinations. We fixed them *a priori* taking into account specific biological criteria. As a result, the GERM clusters we have discovered plausibly identify groups of cCREs and genes between which specific types of gene expression regulatory relations exist. More specifically, in every learning iteration, the set of importance weights for each feature space and the repulsive and attractive forces were fixed, considering a set of simple rules.

As explained in section 4.2, given a set of cCREs that are equally distant in the genome to a gene or a set of uniformly expressed genes, we tighten the similarity between the embeddings of these elements considering:

1. The correlation of chromatin accessibility and acetylation profiles of each cCRE,
2. The inverse correlation of the chromatin accessibility and the methylation profiles of each cCRE,
3. The absolute value of the correlation between the gene-expression profile of the gene(s) and the chromatin accessibility profile of the cCRE(s),

These basic inductive biases drove the setting of the hyper-parameters we used. Despite these inductive biased turn efficient for finding the results exposed in section 4.3, we argue that these may limit the broader range of regulatory mechanisms that exist in nature and could be inferred from the data we used in this experiment. We claim that a broader set of gene regulatory relations could be fostered from these data by a deep learning pipeline that includes a bigger hypothesis space. We could model softer inductive biases as regularization terms of the reconstruction loss. This technique may converge to more expressive manifolds concerning the one presented in this work. We explain more on this research path in paragraph 5.3.3.

Chapter 5

Conclusions

This dissertation offered a comprehensive review of the synergies between deep learning and heterogeneous graph models. Two main paradigms of collaboration have been identified between these disciplines. The first one consists of enhancing the scalability, expressiveness and generalization power of graph algorithms through deep learning. The second is the augmented efficiency of solution-space exploration that heterogeneous graph modeled scenarios could help design of deep learning optimization pipelines.

This research also identified two open research opportunities where the studied synergisms could be helpful. The first one was the online optimization of service function chain deployment in virtualized content delivery networks for live-streaming. The second was the inference of developmental regulatory mechanisms between genes and cis-regulatory elements. The candidate demonstrated his proficiency in the research field by applying the synergisms identified in the first phase of the research to solve such open problems. The proposed solutions to the open problems were based on the het-graph model-based Deep Reinforcement Learning and het-graph Representation Learning. In this conclusive chapter, we first mention some theoretical aspects of this dissertation to summarize the contribution and its limits as a whole, we then review other application fields that could benefit from the proposed methodologies, and lastly, we explore further research directions that could enhance the solutions presented in this thesis.

5.1 Theoretical aspects

The graph-based architectural inductive biases that could be inserted in deep ANN architectures to help solve tasks in model-based scenarios are the essence of the first good practice that we have studied in this dissertation. The Geometric Deep Learning Blueprint [26] has cleverly formalized this type of synergism with the language of group theory. Instead, at the core of the second good practice, we have action or state decomposition that creates non-explicitly architectural inductive biases to direct the optimization paths of Deep ANN architectures. These type of synergism is more difficult to formalize in the language of, for example, Markov Decision Processes. However, some related formalizations exist. For example, the policy invariance theorem presented in [183].

We note that the two types of synergism we have identified are two special cases of a unique practice: inductive bias injection in deep learning pipelines. Inductive biases restrict the degrees of freedom or the dimension of the hypothesis space by imposing some constraints in the solution space of the function approximator. However, the usage of Deep ANNs, imposes itself a special kind of inductive bias. Attempts to explain the hypothesis space induced by deep ANNs have been made. One remarkable example is explaining them through spline theory [197]. We argue that this and other related studies could form the basis for the formalization of the restrictions of the hypothesis space that different architectural inductive biases over deep ANNs create. We are less confident about the possibility of formalization of non-architectural inductive biases as a whole, because of the multiple types of biases that could be injected: regularization, sampling techniques, etc.

Despite mathematical models of inductive biases, a class of more high-level approaches like the concept of symbolic behavior [72] and innateness [154] in artificial intelligence could also help to understand the commonalities between the two synergisms exposed in this dissertation. These approaches could also open the question of whether the studied synergism a neat dichotomy or if they must be considered as a unique ubiquitous synergism in AI. Lastly, an important observation is that mixing these two synergisms is a very effective good practice already in industrial production deployments. Examples of such mixed approaches include deep relational reinforcement learning [310], neural algorithmic reasoning [259], Graph Networks [17], among others.

This dissertation aimed to create a clear taxonomy of how AI designers could model inductive biases for industrial-scale deep learning pipelines. Good Old Fashioned AI could be used to bring efficiency to deep learning training cycles when model knowledge is given and some well-designed deep learning architectures that can empower algorithms to extend their application range. A formal high-level model of innateness and inductive biases is thus a possible extension of this research. The author limits to claim in this manuscript that the previously mentioned works could enlighten such a formalization.

5.2 New Application Fields of the proposed methods

5.2.1 Clustering of time-series data

The influence of one or multiple time-series in the evolution profile of another time-series might be complex and hardly captured by linear correlation functions. [92] Some statistics exist that can measure non-linear dependence probability between two random variables. [129] However, the task of complex non-linear dependence finding in real-world time-series might also benefit from the universal approximation capacity of deep artificial neural networks. We demonstrated this by presenting a new methodology for creating a unique low-dimensional and easy-to-cluster embedding of a heterogeneous network in chapter 4. This clustering method is based on deep graph auto-encoders and an iterative process for creating multi-semantic node embeddings from multiple feature spaces. We argue that such a technique could also be helpful to cluster other time-series data.

One sample field that usually correlates an independent time-series with another set of exogenous time-series is the modelisation of material degradation as a function of weather factors. Weather factors are usually obtained as multi-variate time-series and are often high-dimensional. Multiple studies are being carried out to study and model the performance degradation of industrial machinery as a function of these and other similar exogenous time-series like usage and maintenance profiles. These studies often rely on domain-specific knowledge to hypothesize the influence of various factors on the degradation profile of the independent variable. We argue that our method could be used in this research field to help clustering time-series data considering their non-linear dependences.

For example, the influence of weather and traffic under the evolution of pavement performance indexes is a current research area where recurrent neural networks [134, 252] and shallow-ML [162, 191, 267] techniques have produced outstanding results, even if, to the best of the candidate's knowledge, the current literature lacks to use the het-graph modeling of these environments. We argue that the problem of predicting the pavement performance degradation could be modeled as a het-graph containing multiple node types like pavement chunks, weather, and traffic sensors. Geographic proximity could be the primary edge type of this graph. The inductive graph bias possessed by the GNN architectures like those leveraged in our method could be helpful in this context. By creating a proper graph deep learning pipeline, road chunks from multiple geographical regions could be clustered as a function of both the similarity in the chunks' performance degradation profiles and the similarity in the correlations with exogenous time-series data. Other examples of time-series data that could be better analyzed through clustering techniques like ours come from the stock market analysis field, where multiple stock evolution profiles are known to be correlated to various exogenous time-series data. We argue that a het-graph modelisation like ours could

shed light on the detection of correlation patterns among groups of market shares. Moreover, these correlation patterns could be associated with domain-specific phenomena with the help of domain experts to converge to an inductive instrument that could help in the detection of irregularities and investment opportunities.

5.2.2 Social Network analysis

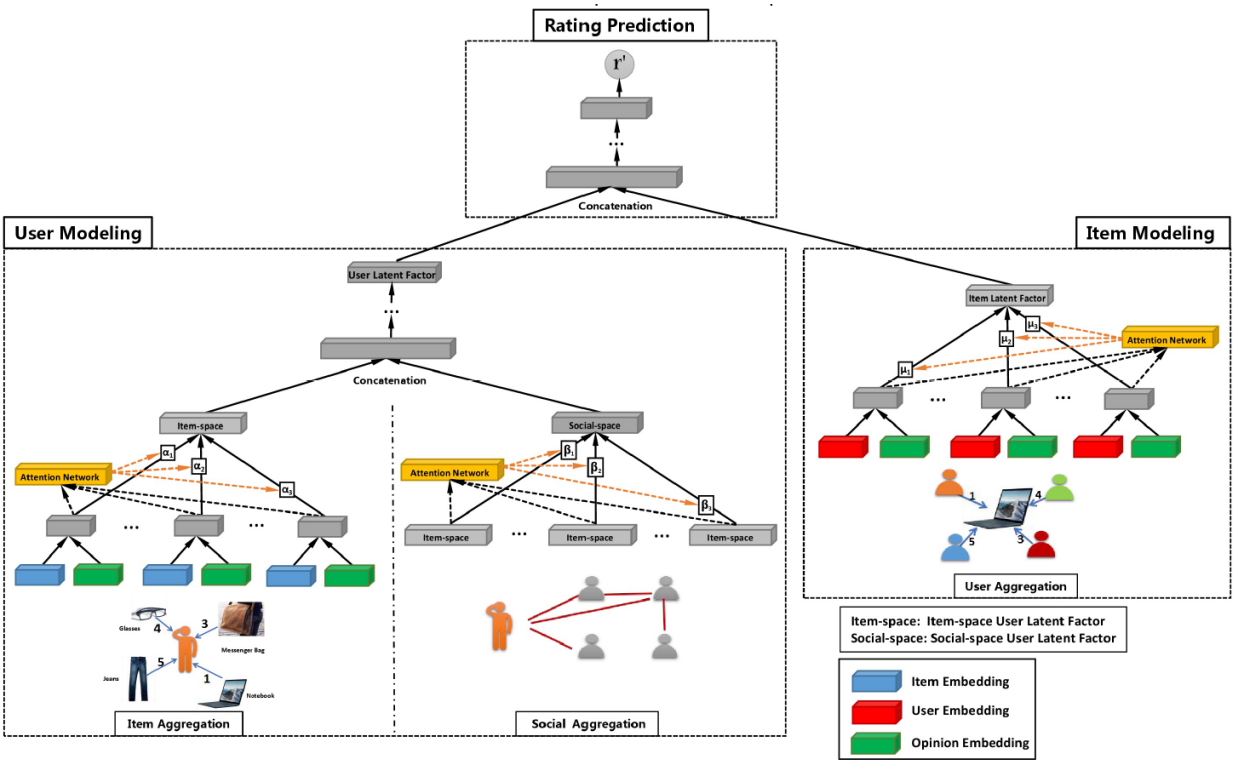

Figure 5.1. Deep learning-based representation learning of heterogeneous networks has been applied to the field of social network analysis. Social Networks are heterogeneous by nature because they include multiple types of nodes (individuals, events, items, fan pages, etc.) and edges (friendship, likes, associations, among others). This figure gives an overview of an algorithm that combines graph neural networks (GNN), attention-based mechanisms, and a hierarchical feature concatenation schema for studying a heterogeneous social network of individuals and items to create a social recommendation system. The algorithm presented in chapter 4 could enhance the final node embeddings by including the information from node-level features in the embedding space. This figure was obtained from [66]

Social recommendation systems [66], and user identity linkage [319] are some examples of social network analysis algorithms based on deep learning and het-graph-RL. Social Networks are, in fact, heterogeneous because they include various types of nodes (individuals, events, items, fan pages, etc.) and edges (friendship, likes, associations, among other interactions). This heterogeneity is the main challenge the research community addresses when trying to produce node embeddings that capture the complex topological relationships of these networks. In the case of recommender systems, for example, Fan *et al.* [66] noted that, when producing embeddings, both user-item and user-user interactions need to be considered by the representation learning algorithm. This condition holds especially if the embeddings should capture efficient topological features from the het-graph. Fan *et al.* modeled a het-graph-RL that included hierarchical attention-based mechanisms to construct a social recommender system. Each edge type is analyzed separately by a specific GNN module in this work. Then, the output of every module is concatenated to form a comprehensive embedding vector used to predict user ratings of items. Feature concatenation is done hierarchically to inject inductive biases that permit to learn users' social and individual preferences. Remarkably, the

attention-based mechanism learns to give the right importance weight to some associations in the graph with respect to others.

The work in [66] considered only the topological relationships among nodes, ignoring the node-level features. The authors of this work claim that the inclusion of node-level characteristics in their presented social recommender system would, in fact, improve its prediction performance in recommendation downstream tasks, and their call for future work mentions this need. We argue that our algorithm could be useful to answer this call because, apart from taking into account topological information, it combines various types of node-level characteristics to create multi-semantic embedding vectors. However, as exposed in paragraph 1.2.1 of this dissertation, our approach is not the unique deep-learning-based het-graph-RL algorithm that considers node-level features to produce the final embeddings. Other recent approaches with node-level feature inclusion could be adopted in this task [313, 271].

Many deep het-graph-RL algorithms were presented in paragraph 1.2.1 that use attention-based mechanisms to learn the importance distribution between node-level feature types and instances. However, in some network-related environments, this distribution may not be complex and dynamic: it could be the same for every node and be noted by domain experts. We argue that in these cases, industrial environments may prefer to use algorithms like the one presented in this research, whose main advantage over the attention-based ones is architectural simplicity. We can state this because our algorithm produces the final embeddings using a single deep graph auto-encoder at the cost of model-based parameter setting for the feature combination phase. Thus, the number of parameters to optimize is necessarily less concerning attention-based het-graph-RL approaches.

5.2.3 Online resource allocation optimization for cloud-hosted vCDNs

In chapter 3, we have proposed an AI-driven SFC deployment algorithm for live-video delivery in virtualized CDNs. This algorithm delegates the processing of live-streaming video requests to distributed chains of VNF containers. In the use-case presented, these containers are deployed on a cloud-hosted virtualized network infrastructure as in [136]. In the proposed vCDN ETSI standard architecture [62], the orchestrator module would host and run our DRL module. The orchestrator could use our algorithm to individuate near-to-optimal SFC deployment policies. On the other hand, in the mentioned standard, the VIM module is in charge of resource allocation processes to permit horizontal and vertical scaling of the underlying NFVI.

We have evaluated our SFC deployment solution with a greedy and reactive algorithm for resource allocation, which was specified in section 3.2.4. Dynamic policies for optimal resource allocation in vCDNs are instead an active area of research. [286] Joint optimization of SFC deployment and resource allocation has also been investigated, and practical solutions have been published for vCDNs [290] and general-case virtualized network systems [133]. Also, DRL-based resource-allocation agents have proven capable of learning proactive and long-term convenient policies in mobile edge computing [37], computing offloading [46], among other scenarios [228]. Instead, we propose investigating the DRL-based optimization of SFC Deployment in cloud-hosted vCDN systems. By applying a well-designed DRL-based resource-allocation algorithm, we argue that the SFC deployment policies presented in chapter 3 would be further optimized in terms of both QoS and costs. The main difficulty of designing a VNF resource adapter in our case is the high dimension of the action space. Such a space consists of the resource quantities that we want to assign to each resource type of every VNF in every node of the vCDN distributed system. Even if we could discretize such an action space to some extent, we should model a fine-grained adapter agent capable of precisely adapting the resource provision for each container.

We argue that DRL designers could obtain promising guidelines for designing a resource-allocation DRL agent in a cloud-hosted vCDN environment using the design strategy adopted in chapter 3. More specifically, designers could follow the intuition of shaping rewards based on the potential contribution of states to the final objective. We now provide some hints for shaping the rewards in this way.

We argue that a good technique to design an efficient resource allocation/adaptation agent is to serialize the action-space into single VNF resource-adaptation actions. In the particular het-graph model that we have made, each VNF module's utilization and resource provision in every hosting node is tracked as a node-level feature. We could use this information to memorize the utilization immediately after performing adaptation actions and after the system has served some incoming requests. We could then exploit the utilization's initial and final values to give a specialized reward to the adaptation action performed. More specifically, we could hypothesize the following situation: We perform a resource adaptation to a specific VNF instance f_k^i, where k is the VNF type, and i is the hosting node that instantiates the VNF. Then, we could shape the rewards as a function of the coherence of the adaptation concerning the utilization changes after the request serving period, as schematized in Table 5.1.

Table 5.1. A hint to the design of a reward shaping strategies for online VNF resource allocation actions considering the vCDN model presented in chapter 3

Adaptation Action	Resulting Conditions after serving period	Reward	Reward Meaning
$\delta_c > 0$	$\delta_\mu > 0$	$10 \cdot \delta_c$	very good
$\delta_c > 0$	$\delta_\mu = 0$	$-\frac{\delta_c}{10}$	bad
$\delta_c > 0$	$\delta_\mu < 0$	$-\delta_c$	very bad
$\delta_c < 0$	$\delta_\mu > 0$	δ_c	very bad
$\delta_c < 0$	$\delta_\mu = 0$	$-\frac{\delta_c}{10}$	good
$\delta_c < 0$	$\delta_\mu < 0$	$-10\delta_c$	very good
$\delta_c = 0$	$\delta_\mu > 0$	$-\delta_\mu$	bad
$\delta_c = 0$	$\delta_\mu = 0$	$r > 0$	good
$\delta_c = 0$	$\delta_\mu < 0$	δ_μ	bad

The hints on reward shaping provided in Table 5.1 are just one example of how the availability of model-based knowledge can help to create efficient exploration biases on a DRL agent. As explained in chapter 1 the availability of reward policies like the one in Table 5.1 is the main factor that permits the designers to create biased state-space representations that are light and, at the same time effective.

5.3 Future Research Directions

5.3.1 Automatic Curriculum Learning in Heterogeneous Graph RL

In chapter 4 we presented an algorithm for deep learning-based heterogeneous graph representation learning. In particular, we used manifold learning in the form of a probabilistic graph generative algorithm. Specifically, we created a graph where the relationships between nodes corresponded to a regularized version of the euclidean distance relationship between points. This generative approach helped create a sparse graph from the initial non-graph formatted dataset. Our algorithm iteratively

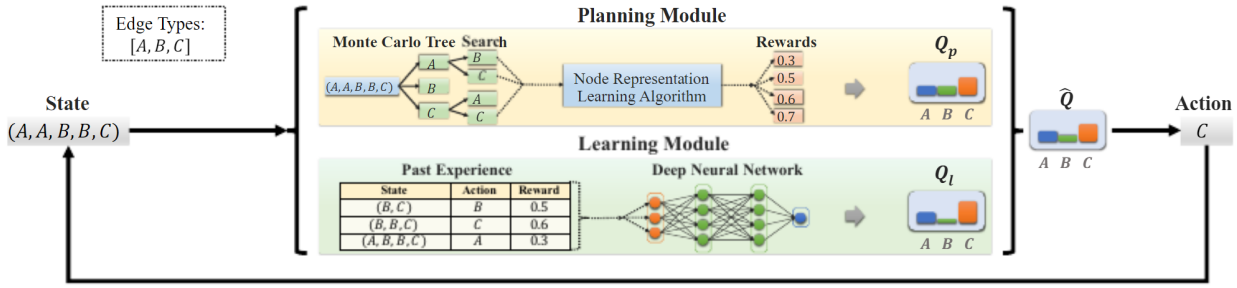

Figure 5.2. One main research direction we pursue is the incorporation of automatic hyperparameter optimization (HPO) routines for our heterogeneous graph representation learning (het-graph-RL) algorithm. This figure is obtained from [209] where authors used Monte Carlo tree search (MCTS) to help a DRL agent learn the optimal curriculum for embedding star networks. In other words, the training dataset for the embedding algorithm is divided into edge-type-specific chunks, and a DRL agent learns the correct sequence of training datasets to learn. We are investigating ways to extend a similar auto-HPO procedure to our dynamic-sparsity H-graph embedding algorithm to reduce the effort of manually finding the optimal HPO for each specific downstream task.

generates various graphs with increasing sparsity. The topological features of the generated graphs are then combined to produce the final node embeddings. Finally, we performed proper hyperparameter optimization to find the best results in the specific downstream task we faced. Despite many secondary hyper-parameters optimized using domain knowledge, the sequence of sparsity values for the iterative graph generation constitutes the main dynamic hyperparameter of the proposed algorithm. In our experiment, we have optimized the values for this dynamic hyperparameter on a trial-and-error basis.

We argue that further development of our heterogeneous graph embedding technique moves in the direction of automatic graph machine learning [321]. In particular, we are studying how to exploit Deep Reinforcement Learning (DRL) for Hyperparameter Optimization (HPO) routines of the sequence of sparsity values. DRL has been applied in the past for learning the optimal order of training tasks for the embedding of heterogeneous star networks [209]. Star networks are a sub-class of heterogeneous networks that have two types of nodes: center nodes and attribute nodes. Attribute nodes have multiple sub-classes, and each center node is linked to various attribute nodes through correspondent edges.

In [209], the proposed graph representation learning of star networks is also iterative: the embedding is created throughout various learning iterations, each phase concentrating on learning the information encoded by a particular type of attribute node. Authors in [209] face the problem of automatically learning the best sequence of attribute node-types that the embedding module needs to learn. Their objective is to converge to center-node embeddings suitable for a particular downstream task. Qu *et al.* propose the usage of Monte Carlo tree search (MCTS) to efficiently explore the high-dimensional action space that such a problem implies. Remarkably, they have tested their algorithm on various star network databases and downstream tasks. Inspired by the work in [209], we are investigating how to incorporate a DRL agent into our het-graph-RL algorithm to automatically learn the optimal sequence of values for the sparsity parameter.

Our deep-het-graph-RL algorithm uses a dynamic sparsity hyper-parameter $k_i, \forall i \in I$, where I is the number of iterations in which we generate a graph with the correspondent sparsity and refine the embedding. This hyper-parameter is meant to follow a crescent path $k_0, k_1, ..., K_I$ so that $K_0 < k_1 < ... < K_I$. The sparsity increment, however, is not necessarily a constant value. At each iteration i, the sparsity parameter is limited by the interval $[k_{i-1}, N]$, where N is the number of nodes in the network. In this case, the action space could be prohibitively large to explore with a pure model-free DRL approach. We propose then to learn the optimal sequence of k_i, by creating an MDP model where model the state an action space as follows:

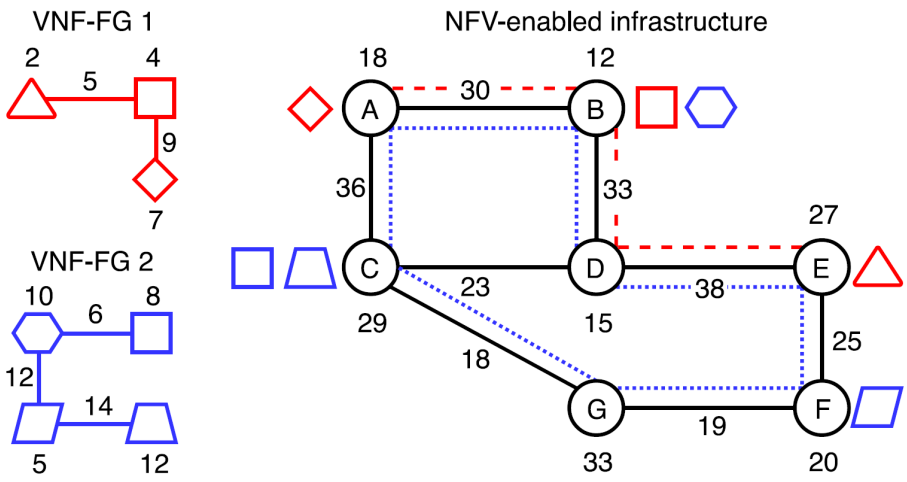

Figure 5.3. A schematic representation of the task of allocating and chaining virtual network function forwarding graphs (VNF-FG). Multiple VNF types are represented by geometrical shapes. In the left side of the image we have the specifications of the two VNF-FG that are deployed in the right side, which contains the NFVI graph with the corresponding node and link capacities. The VNF-FG embedding problem can be divided in two sub-tasks: VNF allocation (place the requested VNF modules in nodes of the NFVI) and FG chaining (composing the best NVFI paths for connecting the deployed VNF modules taking into account the requisites of the virtual links in the VNF-FG). Inspired by recent success of the application of relational deep reinforcement learning, we propose to investigate on a model-free DRL tecnique for jointly optimize these two tasks together. This figure was obtained from [228]

- The states consist of the autoencoder parametric state at the end of every optimization iteration,
- the actions consist of the sparsity **augmentation** to perform before the next learning iteration.

Moreover, we argue that we could exploit MCTS if we limit the action-space, i.e., the options of the sparsity augmentation.

5.3.2 AI-driven VNF Forwarding Graph-Embedding

In this dissertation, we have investigated the problem of live-streaming SFC Deployment in cloud-hosted virtualized network infrastructures. We assumed that, in this kind of NFVI environment, the cloud provider is responsible for network-level routing. Consequently, in our model, we were able to abstract the network-level routing between any two VNF modules into a single application-level connection between the correspondent VNF hosting nodes, as explained in section 3.2.1.

Suppose we take into consideration the network level routing problem instead. In that case, we could further enhance the optimization of QoS by creating long-term convenient routing policies with the help of DRL. Given a particular network infrastructure that hosts the vCDN, data transportation from a source VNF to a destination VNF could follow multiple routes. Each route could imply a specific transportation delay. Consequently, routing optimization at the substrate network level could enhance the QoS in a vCDN. The joint problem of SFC deployment and routing in virtualized network systems is referred to as VNF Forwarding Graph-Embedding (VNF-FGE), and a lot of research effort has been devoted to it [228]. Usually, the problem is divided into the placement and chaining tasks. Placement involves where to deploy the VNFs, and chaining is related to routing content between them.

DRL approaches to VNF-FGE have been successfully applied in the recent years [9, 210]. These approaches model the placement-related action as a permutation of the nodes in the network. For each VNF in the SFC chain, the placement task is performed by choosing the first available node taking into account the permutation order proposed by the agent. By doing so, the allocation

failures are prevented from happening, and the agent does not learn from these transitions. We argue that, even if these works converge to optimal VNF-FGE policies, the design of the exploration dynamic injects a bias that could imply sub-optimality of the achieved results. Inspired by the advances in relational deep reinforcement learning [310, 58], we propose a novel research direction which is to reduce the biased design of the action-space by letting the DRL agent learn long-term convenient allocating and chaining actions together. We argue that an interesting research path for this goal is graph convolutional reinforcement learning [119].

Graph convolutional reinforcement learning (GCRL) is a multi-agent DRL algorithm that permits the agents to consider the topological characteristics of an environment modeled as a dynamic graph. Topological characteristics of the environment are abstracted using the neural message passing framework. As explained in section 1.2.1, the latter produces a smoothed representation of the features of a node combining each node's feature vector with an aggregation of the features of the neighborhood nodes. The work in [119] proposes to implement the aggregation operator based on multi-head attention [15]. This attention mechanism permits the agent to privilege some neighboring nodes with respect to others when aggregating features in the neighborhood.

We argue that a multi-agent reinforcement learning algorithm could help design an algorithm for the joint optimization of VNF placement and chaining, i.e., optimization of VNF-FGE. One agent could focus on the placement phase and the other on the chaining problem collaboratively. To prevent inefficient exploration biases in these agents, inspired by the work in 1.2.1, we propose to design a state-space with the help of graph reinforcement learning with attention-based aggregation of local state features. Such a design would help abstract topological characteristics of the network environment, and we could improve the effectiveness of a more model-free exploration with respect to current existent approaches for DRL based VNF-FGE.

5.3.3 Relational GCNs and regularized unsupervised rewards for more general gene-expression regulation mechanism fostering

In chapter 4, we have delivered a deep learning pipeline that learns to foster mechanisms of gene expression regulation. The presented algorithm relied on a loss function definition that permitted us to discern the importance of various original feature spaces for inferring a specific type of Gene regulatory mechanisms. Still, it also created a strong inductive bias that restricted the class of GERM clusters we could find, as explained in paragraph 4.6.2. In short, the expressive power of the final manifold was biased to accommodate the statistical characteristics of the particular biological drivers that governed the loss function desing process.

As mentioned in paragraph 1.2.1, the Relational Graph Convolutional Network (R-GCN) architecture [229] is a flexible graph neural network that learns to differentiate between types of relationships for producing the final node embeddings through gradient descent. Other types of architectures are also available that, for example, permit learning to give importance to the neighbors of each node as a function of the features of the node instances themselves [260]. We ask ourselves if our DeepReGraph algorithm could somehow benefit from these kinds of flexible architectures that learn how to automatically combine different original feature spaces.

The most challenging part of such a setting would be the definition of the loss function that drives learning the node embeddings for the gene-CRE graph. We could create a monolithic loss function like the one we used in paragraph 4.2.7. Still, we would again create a strong inductive bias as we did with the current implementation of DeepReGraph. This limit falls from the fact that we should fix *a-priori* the combination scheme of the set of original feature spaces of the graph to reduce them to a unique feature space under which the reconstruction loss is minimized. By doing this, we would again bias the learning to a particular class of regulatory relationships, and thus we would not have gained more expressive flexibility than DeepReGraph. Perhaps we would only augment the learning parameters of the model and thus, reduce the current data efficiency. In synthesis, whether it comes from the definition of the loss function or data pre-processing, we should reduce the strength of the inductive biases that drive our manifold learning. We could make

this reduction only by obtaining learning reward signals directly from biological criteria that mirror a wide range of candidate regulatory mechanisms between genes and CREs.

In the last years, robust algorithms for gene set enrichment analysis (GSEA) [247] and motif enrichment analysis (MEA) [161] have been proposed. Powerful and easily-accessible GSEA libraries permit obtaining ratings for a set of genes as a function of how it is related to different genetic functions. GSEA algorithms are dynamic in that they can use data from various instances of next-generation sequencing experiments to quantify the significance with which a set of genes is related to a wide set of genetic functions. [124]. These genetic functions could be, for example, indicators of phenotypical differentiation or other important developmental mechanisms. Similar dynamicity characterizes most of the current MEA approaches. [150] These libraries permit quantifying the significance of associations between candidate CRE regions and a wide range of well-known transcription factors (TF). We claim that an intelligent combination of the statistical outcomes of MEA and GSEA could solve the previously-mentioned challenge.

In a reinforcement learning setting, MEA and GSEA could constitute efficient environments that provide reward signals for various candidate associations or heterogeneous clusters of genes and cCREs. Given any cluster containing genes and cCREs, these reward signals could quantify the significance of the functional association of the inner gene set and the TF association of the cCRE set. These associations should no more be restricted to a particular sub-class of regulatory mechanisms in that they will give high rewards to any association as a function of its biological significance. In other words, we are no more interested in particular associations but in every association, as long as it is biologically significant. We argue that, by carefully designing these rewards, we could create more expressive end-to-end manifold learning pipelines for heterogeneous gene-CRE graphs. In this new inference paradigm, we would learn the combination parameters of different original feature spaces through gradient descent. Moreover, the learned manifolds' spatial distribution would mirror a broader range of gene regulatory mechanisms.

Bibliography

[1] Abdallah, M., Griwodz, C., Chen, K.-T., Simon, G., Wang, P.-C., and Hsu, C.-H. Delay-Sensitive video computing in the cloud: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications* (2018).

[2] Afshar, R. R., Rhuggenaath, J., Zhang, Y., and Kaymak, U. A reward shaping approach for reserve price optimization using deep reinforcement learning. In *2021 International Joint Conference on Neural Networks (IJCNN)* (2021).

[3] Agarap, A. F. Deep learning using rectified linear units (ReLU). *arXiv e-prints* (2018).

[4] Ahmed, A., Shervashidze, N., Narayanamurthy, S., Josifovski, V., and Smola, A. J. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on World Wide Web*, WWW '13. Association for Computing Machinery, New York, NY, USA (2013).

[5] Akinosho, T. D., Oyedele, L. O., Bilal, M., Ajayi, A. O., Delgado, M. D., Akinade, O. O., and Ahmed, A. A. Deep learning in the construction industry: A review of present status and future innovations. *Journal of Building Engineering* (2020).

[6] Alberts, B. *Molecular Biology of the Cell*. Garland Science (2017).

[7] Ammanabrolu, P. and Riedl, M. Playing text-adventure games with graph-based deep reinforcement learning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota (2019). Available from: https://aclanthology.org/N19-1358, doi:10.18653/v1/N19-1358.

[8] Amodio, M. and Krishnaswamy, S. MAGAN: Aligning biological manifolds. In *Proceedings of the 35th International Conference on Machine Learning* (edited by J. Dy and A. Krause), vol. 80 of *Proceedings of Machine Learning Research*. PMLR (2018).

[9] Anh Quang, P. T., Hadjadj-Aoul, Y., and Outtagarts, A. Evolutionary Actor-Multi-Critic model for VNF-FG embedding. In *2020 IEEE 17th Annual Consumer Communications Networking Conference (CCNC)*. ieeexplore.ieee.org (2020).

[10] Arai, S. Effects of shaping a reward on multiagent reinforcement learning. *Multi-Agent Applications with Evolutionary Computation and Biologically Inspired Technologies* (2011).

[11] Argelaguet, R., Arnol, D., Bredikhin, D., Deloro, Y., Velten, B., Marioni, J. C., and Stegle, O. MOFA+: a statistical framework for comprehensive integration of multi-modal single-cell data. *Genome Biol.* (2020).

[12] Arulkumaran, K., Deisenroth, M. P., Brundage, M., and Bharath, A. A. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine* (2017).

[13] Asmuth, J., Littman, M. L., and Zinkov, R. Potential-based shaping in model-based reinforcement learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence* (2008).

[14] Bagavathi, A. and Krishnan, S. Multi-Net: A scalable multiplex network embedding framework. In *Complex Networks and Their Applications VII*. Springer International Publishing (2019).

[15] Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014). arXiv:1409.0473.

[16] Barabási, A.-L. Network science. *Philosophical Transcations of the Royal Society A. Mathematical, Physical and Engineering Sciences* (2013).

[17] Battaglia, P. W., et al. Relational inductive biases, deep learning, and graph networks (2018). arXiv:1806.01261.

[18] Baxevanis, A. D., Bader, G. D., and Wishart, D. S. *Bioinformatics*. John Wiley & Sons (2020).

[19] Beck, M. T. and Botero, J. F. Coordinated allocation of service function chains. In *2015 IEEE Global Communications Conference (GLOBECOM)* (2015).

[20] Belkin, M. and Niyogi, P. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS'01. MIT Press, Cambridge, MA, USA (2001).

[21] Bellman, R. A markovian decision process. *Journal of Mathematics and Mechanics* (1957).

[22] Bellman, R. and Kalaba, R. E. *Dynamic programming and modern control theory*, vol. 81. Citeseer (1965).

[23] Benkacem, I., Taleb, T., Bagaa, M., and Flinck, H. Optimal VNFs placement in CDN slicing over Multi-Cloud environment. *IEEE Journal on Selected Areas in Communications* (2018).

[24] Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems* (edited by C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger), vol. 26. Curran Associates, Inc. (2013).

[25] Borgatti, S. P. and Halgin, D. S. On network theory. *Organization Science* (2011).

[26] Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine* (2017).

[27] Budhkar, S. and Tamarapalli, V. An overlay management strategy to improve QoS in CDN-P2P live streaming systems. *Peer-to-Peer Networking and Applications* (2020).

[28] Butte, A. J. and Kohane, I. S. Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements. In *Biocomputing 2000*, pp. 418–429. World Scientific (1999).

[29] Buyya, R., Pathan, M., and Vakali, A. *Content Delivery Networks*. Springer Science & Business Media (2008).

[30] Cai, H., Zheng, V. W., and Chang, K. C.-C. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering* (2018).

[31] Çalişir, S. and Pehlivanoğlu, M. K. Model-Free reinforcement learning algorithms: A survey. In *2019 27th Signal Processing and Communications Applications Conference (SIU)*. ieeexplore.ieee.org (2019).

[32] Cao, S., Lu, W., and Xu, Q. GraRep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15. Association for Computing Machinery, New York, NY, USA (2015).

[33] Cevallos Moreno, J. F., Sattler, R., Caulier Cisterna, R. P., Ricciardi Celsi, L., Sánchez Rodríguez, A., and Mecella, M. Online service function chain deployment for Live-Streaming in virtualized content delivery networks: A deep reinforcement learning approach. *Future Internet* (2021).

[34] Cevallos Moreno, J. F., Zarrineh, P., Sánchez-Rodríguez, A., and Mecella, M. DeepReGraph co-clusters temporal gene expression and cis-regulatory elements through heterogeneous graph representation learning. *F1000Res.* (2022).

[35] Chang, S., Han, W., Tang, J., Qi, G.-J., Aggarwal, C. C., and Huang, T. S. Heterogeneous network embedding via deep architectures. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15. Association for Computing Machinery, New York, NY, USA (2015).

[36] Chen, F., Wang, Y.-C., Wang, B., and Kuo, C.-C. J. Graph representation learning: a survey. *APSIPA Transactions on Signal and Information Processing* (2020).

[37] Chen, J., Xing, H., Xiao, Z., Xu, L., and Tao, T. A DRL agent for jointly optimizing computation offloading and resource allocation in MEC. *IEEE Internet of Things Journal* (2021).

[38] Chen, X., Guo, J., Zhu, Z., Proietti, R., Castro, A., and Yoo, S. J. B. Deep-RMSA: A Deep-Reinforcement-Learning routing, modulation and spectrum assignment agent for elastic optical networks. In *2018 Optical Fiber Communications Conference and Exposition (OFC)* (2018).

[39] Cho, H., Berger, B., and Peng, J. Compact integration of Multi-Network topology for functional analysis of genes. *Cell Systems* (2016).

[40] Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN Encoder-Decoder for statistical machine translation. *arXiv [cs.CL]* (2014).

[41] CISCO. Cisco global cloud index: Forecast and methodology (2016). Available from: https://virtualization.network/Resources/Whitepapers/0b75cf2e-0c53-4891-918e-b542a5d364c5_white-paper-c11-738085.pdf.

[42] Cisco, V. Cisco visual networking index: Forecast and trends (2017). Available from: https://cyrekdigital.com/uploads/content/files/white-paper-c11-741490.pdf.

[43] Coffman, T., Greenblatt, S., and Marcus, S. Graph-based technologies for intelligence analysis. *Communications of the ACM* (2004).

[44] Cozzo, E., de Arruda, G. F., Rodrigues, F. A., and Moreno, Y. Multilayer networks: Metrics and spectral properties. *Understanding Complex Systems* (2016).

[45] Dai, R., Liu, G., Wang, Z., Kan, B., and Yuan, C. A novel Graph-Based energy management system. *IEEE Transactions on Smart Grid* (2020).

[46] Dai, Y., Zhang, K., Maharjan, S., and Zhang, Y. Edge intelligence for Energy-Efficient computation offloading and resource allocation in 5G beyond. *IEEE Transactions on Vehicular Technology* (2020).

[47] Das, R., Dhuliawala, S., Zaheer, M., Vilnis, L., Durugkar, I., Krishnamurthy, A., Smola, A., and McCallum, A. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *arXiv [cs.CL]* (2017). arXiv:1711.05851.

[48] De Moor, B. J., Gijsbrechts, J., and Boute, R. N. Reward shaping to improve the performance of deep reinforcement learning in perishable inventory management. *European Journal of Operational Research* (2022).

[49] Demirci, S. and Sagiroglu, S. Optimal placement of virtual network functions in software defined networks: A survey. *Journal of Network and Computer Applications* (2019).

[50] Devlin, S. M. and Kudenko, D. Dynamic Potential-Based reward shaping. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*. IFAAMAS, ESP (2012).

[51] Dieye, M., Ahvar, S., Sahoo, J., Ahvar, E., Glitho, R., Elbiaze, H., and Crespi, N. CPVNF: Cost-Efficient proactive VNF placement and chaining for Value-Added services in content delivery networks. *IEEE Transactions on Network and Service Management* (2018).

[52] Dong, S., Wang, P., and Abbas, K. A survey on deep learning and its applications. *Computer Science Review* (2021).

[53] Dong, X. and Weng, Z. The correlation between histone modifications and gene expression. *Epigenomics* (2013).

[54] Dong, Y., Chawla, N. V., and Swami, A. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17. Association for Computing Machinery, New York, NY, USA (2017).

[55] Dong, Y., Hu, Z., Wang, K., Sun, Y., and Tang, J. Heterogeneous network representation learning. In *IJCAI*, vol. 20. cs.ucla.edu (2020).

[56] Dong, Y., Tang, X., and Yuan, Y. Principled reward shaping for reinforcement learning via lyapunov stability theory. *Neurocomputing* (2020).

[57] Dulac-Arnold, G., Evans, R., Sunehag, P., and Coppin, B. Reinforcement learning in large discrete action spaces. *ArXiv* (2015).

[58] Džeroski, S., De Raedt, L., and Driessens, K. Relational reinforcement learning. *Machine Learning* (2001).

[59] Eraslan, G., Avsec, Ž., Gagneur, J., and Theis, F. J. Deep learning: new computational modelling techniques for genomics. *Nature Reviews Genetics* (2019).

[60] Ernst, J. and Kellis, M. ChromHMM: automating chromatin-state discovery and characterization. *Nature Methods* (2012).

[61] Ernst, J., Nau, G. J., and Bar-Joseph, Z. Clustering short time series gene expression data. *Bioinformatics* (2005).

[62] Etsi, G. 001,"network functions virtualization (NFV): Use cases", 10-2013. Available from: https://www.etsi.org/deliver/etsi_gs/nfv/001_099/001/01.01.01_60/gs_nfv001v010101p.pdf.

[63] Eubank, S. Network based models of infectious disease spread. *Japanese Journal of Infectious Diseases* (2005).

[64] Fan, Q., Yin, H., Min, G., Yang, P., Luo, Y., Lyu, Y., Huang, H., and Jiao, L. Video delivery networks: Challenges, solutions and future directions. *Computers & Electrical Engineering* (2018).

[65] Fan, S., Zhu, J., Han, X., Shi, C., Hu, L., Ma, B., and Li, Y. Metapath-guided heterogeneous graph neural network for intent recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, KDD '19. Association for Computing Machinery, New York, NY, USA (2019).

[66] Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., and Yin, D. Graph neural networks for social recommendation. In *The World Wide Web Conference*, WWW '19. Association for Computing Machinery, New York, NY, USA (2019).

[67] Fei, X., Liu, F., Xu, H., and Jin, H. Adaptive VNF scaling and flow routing with proactive demand prediction. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications* (2018).

[68] Filelis-Papadopoulos, C. K., Endo, P. T., Bendechache, M., Svorobej, S., Giannoutakis, K. M., Gravvanis, G. A., Tzovaras, D., Byrne, J., and Lynn, T. Towards simulation and optimization of cache placement on large virtual content distribution networks. *Journal of Computational Science* (2020).

[69] Francois-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G., and Pineau, J. An introduction to deep reinforcement learning. *arXiv* (2018). arXiv:1811.12560.

[70] Gao, W., Wu, H., Siddiqui, M. K., and Baig, A. Q. Study of biological networks using graph theory. *Saudi Journal of Biological Sciences* (2018).

[71] Gao, Y. and Toni, F. Potential based reward shaping for hierarchical reinforcement learning. In *Twenty-Fourth International Joint Conference on Artificial Intelligence* (2015).

[72] Garnelo, M. and Shanahan, M. Reconciling deep learning with symbolic artificial intelligence: representing objects and relations. *Current Opinion in Behavioral Sciences* (2019), artificial Intelligence.

[73] Gil Herrera, J. and Botero, J. F. Resource allocation in NFV: A comprehensive survey. *IEEE Transactions on Network and Service Management* (2016).

[74] Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning* (edited by D. Precup and Y. W. Teh), vol. 70 of *Proceedings of Machine Learning Research*. PMLR (2017).

[75] Gomaa, W. E. Modeling gene regulatory networks: A survey. In *2011 9th IEEE/ACS International Conference on Computer Systems and Applications (AICCSA)* (2011). doi:10.1109/AICCSA.2011.6126584.

[76] Gorkin, D. U., et al. An atlas of dynamic chromatin landscapes in mouse fetal development. *Nature* (2020).

[77] Gosak, M., Markovič, R., Dolenšek, J., Slak Rupnik, M., Marhl, M., Stožer, A., and Perc, M. Network science of biological systems at different scales: A review. *Phys. Life Rev.* (2018).

[78] Gosavi, A. Reinforcement learning: A tutorial survey and recent advances. *INFORMS Journal on Computing* (2009).

[79] Gourdin, E., Maillé, P., Simon, G., and Tuffin, B. The economics of CDNs and their impact on service fairness. *IEEE Transactions on Network and Service Management* (2017).

[80] Goyal, P. and Ferrara, E. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems* (2018).

[81] Grondman, I., Busoniu, L., Lopes, G. A. D., and Babuska, R. A survey of Actor-Critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems Man and Cybernetics* (2012).

[82] Grover, A. and Leskovec, J. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16. Association for Computing Machinery, New York, NY, USA (2016).

[83] Grześ, M. Reward shaping in episodic reinforcement learning. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '17. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2017).

[84] Grzes, M. and Kudenko, D. Plan-based reward shaping for reinforcement learning. In *2008 4th International IEEE Conference Intelligent Systems* (2008).

[85] Grzes, M. and Kudenko, D. Theoretical and empirical analysis of reward shaping in reinforcement learning. In *"2009 International Conference on Machine Learning and Applications"* (2009).

[86] Gu, S., Holly, E., Lillicrap, T., and Levine, S. Deep reinforcement learning for robotic manipulation. *arXiv preprint arXiv:1610. 00633* (2016).

[87] Gupta, L., Jain, R., Erbad, A., and Bhamare, D. The P-ART framework for placement of virtual network services in a multi-cloud environment. *Computer Communications* (2019).

[88] Gyawali, S., Qian, Y., and Hu, R. Q. Resource allocation in vehicular communications using graph and deep reinforcement learning. In *2019 IEEE Global Communications Conference (GLOBECOM)* (2019).

[89] Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems* (edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett), vol. 30. Curran Associates, Inc. (2017).

[90] Hamilton, W. L. *Graph Representation Learning*. Morgan & Claypool Publishers (2020).

[91] Hamilton, W. L., Ying, R., and Leskovec, J. Representation learning on graphs: Methods and applications. *IEEE Data Engineering Bulletin* (2017). Available from: http://sites.computer.org/debull/A17sept/p52.pdf.

[92] Han, Z., Zhao, J., Leung, H., Ma, K. F., and Wang, W. A review of deep learning models for time series prediction. *IEEE Sensors Journal* (2021).

[93] Hasselt, H. Double q-learning. *Advances in Neural Information Processing Systems* ("2010").

[94] Hasselt, H. v., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16. AAAI Press (2016).

[95] Hausberg, J. P., Liere-Netheler, K., Packmohr, S., Pakura, S., and Vogelsang, K. Research streams on digital transformation from a holistic business perspective: a systematic literature review and citation network analysis. *Journal of Business Economics and Management* (2019).

[96] Heinz, S., et al. Simple combinations of lineage-determining transcription factors prime cis-regulatory elements required for macrophage and B cell identities. *Molecular Cell* (2010).

[97] Herbaut, N. *Collaborative Content Distribution over a VNF-as-a-Service platform.* Ph.D. thesis, Université de Bordeaux (2017).

[98] Hill, S. M., Lu, Y., Molina, J., Heiser, L. M., Spellman, P. T., Speed, T. P., Gray, J. W., Mills, G. B., and Mukherjee, S. Bayesian inference of signaling network topology in a cancer cell line. *Bioinformatics* (2012).

[99] Holmes, S. H. and Huber, W. *Modern Statistics for Modern Biology.* Cambridge University Press (2018).

[100] Hu, B., Shi, C., Zhao, W. X., and Yu, P. S. 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*, KDD '18. Association for Computing Machinery, New York, NY, USA (2018).

[101] Hu, B., Zhang, Z., Shi, C., Zhou, J., Li, X., and Qi, Y. Cash-Out user detection based on attributed heterogeneous information network with a hierarchical attention mechanism. *AAAI* (2019).

[102] Hu, L., Li, C., Shi, C., Yang, C., and Shao, C. Graph neural news recommendation with long-term and short-term interest modeling. *Information Processing & Management* (2020).

[103] Hu, L., Xu, S., Li, C., Yang, C., Shi, C., Duan, N., Xie, X., and Zhou, M. Graph neural news recommendation with unsupervised preference disentanglement. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics.* Association for Computational Linguistics, Online (2020).

[104] Hu, Y., Hua, Y., Liu, W., and Zhu, J. Reward shaping based federated reinforcement learning. *IEEE Access* (2021).

[105] Huang, H., Hu, Z., Lu, Z., and Wen, X. Network-Scale traffic signal control via multiagent reinforcement learning with deep spatiotemporal attentive network. *IEEE Transactions on Cybernetics* (2021).

[106] Huang, R., Guan, W., Zhai, G., He, J., and Chu, X. Deep graph reinforcement learning based intelligent traffic routing control for Software-Defined wireless sensor networks. *NATO Advanced Science Institutes Series E: Applied Sciences.* (2022).

[107] Huang, W., Zhu, H., and Qian, Z. AutoVNF: An automatic resource sharing schema for VNF requests. *Journal of Internet Services and Information Security* (2017).

[108] Huang, Y.-A., Pan, G.-Q., Wang, J., Li, J.-Q., Chen, J., and Wu, Y.-H. Heterogeneous graph embedding model for predicting interactions between TF and target gene. *Bioinformatics* (2022).

[109] Huynh-Thu, V. A. and Sanguinetti, G. Gene regulatory network inference: An introductory survey. *Methods in Molecular Biology* (2019).

[110] Ibn-Khedher, H., Abd-Elrahman, E., Kamal, A. E., and Afifi, H. OPAC: An optimal placement algorithm for virtual CDN. *Computer Networks* (2017).

[111] Ibn-Khedher, H., Hadji, M., Abd-Elrahman, E., Afifi, H., and Kamal, A. E. Scalable and cost efficient algorithms for virtual CDN migration. In *2016 IEEE 41st Conference on Local Computer Networks (LCN)*. ieeexplore.ieee.org (2016).

[112] Ito, M., He, F., and Oki, E. Robust optimization model for probabilistic protection under uncertain virtual machine capacity in cloud. In *2020 16th International Conference on the Design of Reliable Communication Networks DRCN 2020* (2020).

[113] Jaderberg, M., Simonyan, K., Zisserman, A., and Kavukcuoglu, K. Spatial transformer networks. *Advances in Neural Information Processing Systems* (2015).

[114] Jagtap, S., Pirayre, A., Bidard, F., Duval, L., and Malliaros, F. BRANet: Graph-based integration of multi-omics data with biological a priori for regulatory network inference. In *17th International Conference on Computational Intelligence Methods for Bioinformatics and Biostatistics (CIBB)* (2021).

[115] Jahromi, N. T., Kianpisheh, S., and Glitho, R. H. Online VNF placement and chaining for value-added services in content delivery networks. In *2018 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)* (2018).

[116] Jayender, P. and Kundu, G. K. Intelligent ERP for SCM agility and graph theory technique for adaptation in automotive industry in india. *International Journal of System Assurance Engineering and Management* (2021).

[117] Ji, H., Zhu, J., Wang, X., Shi, C., Wang, B., Tan, X., Li, Y., and others. Who you would like to share with? a study of share recommendation in social e-commerce. *Proceedings of the AAAI Conference on Artificial Intelligence* (2021).

[118] Jia, Y., Wu, C., Li, Z., Le, F., and Liu, A. Online scaling of NFV service chains across Geo-Distributed datacenters. *IEEE/ACM Transactions on Networking* (2018).

[119] Jiang, J., Dun, C., Huang, T., and Lu, Z. Graph convolutional reinforcement learning. In *International Conference on Learning Representations* (2020). Available from: https://openreview.net/forum?id=HkxdQkSYDB.

[120] Kaelbling, L. P., Littman, M. L., and Moore, A. W. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research* (1996).

[121] Khezri, H. R., Moghadam, P. A., Farshbafan, M. K., Shah-Mansouri, V., Kebriaei, H., and Niyato, D. Deep reinforcement learning for dynamic reliability aware NFV-Based service provisioning. In *2019 IEEE Global Communications Conference (GLOBECOM)*. ieeexplore.ieee.org (2019).

[122] Khoshraftar, S. and An, A. A survey on graph representation learning methods. *arXiv* (2022). arXiv:2204.01855.

[123] Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Sallab, A. A. A., Yogamani, S., and Pérez, P. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems* (2021).

[124] Korotkevich, G., Sukhov, V., Budin, N., Shpak, B., Artyomov, M. N., and Sergushichev, A. Fast gene set enrichment analysis. *BioRxiv* (2021).

[125] Kramer, O. K-Nearest neighbors. In *Dimensionality Reduction with Unsupervised Nearest Neighbors* (edited by O. Kramer), pp. 13–23. Springer Berlin Heidelberg, Berlin, Heidelberg (2013).

[126] Kulkarni, T. D., Narasimhan, K., Saeedi, A., and Tenenbaum, J. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in Neural Information Processing Systems* (2016).

[127] Kutschenreiter-Praszkiewicz, I. Graph-Based decision making in industry. *Graph Theory - Advanced Algorithms and Applications* (2018).

[128] Kuyumcu, A. and Garcia-Diaz, A. A polyhedral graph theory approach to revenue management in the airline industry. *Computers & Industrial Engineering* (2000).

[129] La Rocca, M. and Vitale, L. Clustering time series by nonlinear dependence. In *Mathematical and Statistical Methods for Actuarial Sciences and Finance*. Springer International Publishing (2021).

[130] Le, Q., Miralles-Pechuán, L., Kulkarni, S., Su, J., and Boydell, O. An overview of deep learning in industry. *Data Analytics and AI* (2020).

[131] LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *Nature* (2015).

[132] Levine, M. and Davidson, E. H. Gene regulatory networks for development. *Proceedings of the National Academy of Sciences U. S. A.* (2005).

[133] Li, J., Zhang, N., Ye, Q., Shi, W., Zhuang, W., and Shen, X. Joint resource allocation and online virtual network embedding for 5G networks. In *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*. ieeexplore.ieee.org (2017).

[134] Li, J., Zhang, Z., Wang, X., and Yan, W. Intelligent decision-making model in preventive maintenance of asphalt pavement based on PSO-GRU neural network. *Advanced Engineering Informatics* (2022).

[135] Li, X., Darwich, M., Bayoumi, M., and Salehi, M. A. Cloud-Based video streaming services: A survey. *arXiv* (2020). arXiv:2011.14976.

[136] Li, X., Darwich, M., Salehi, M. A., and Bayoumi, M. A survey on cloud-based video streaming services. In *Advances in Computers*, vol. 123. Elsevier (2021).

[137] Li, X., Zhang, H., and Zhang, R. Adaptive graph Auto-Encoder for general data clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* (2021).

[138] Li, Y. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274* (2017). arXiv:1701.07274.

[139] Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. S. Gated graph sequence neural networks. In *4th International Conference on Learning Representations, ICLR, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings* (edited by Y. Bengio and Y. LeCun) (2016). Available from: http://arxiv.org/abs/1511.05493.

[140] Li, Y., Wu, F.-X., and Ngom, A. A review on machine learning principles for multi-view biological data integration. *Briefings in Bioinformatics* (2018).

[141] Liao, Y., Smyth, G. K., and Shi, W. The R package rsubread is easier, faster, cheaper and better for alignment and quantification of RNA sequencing reads. *Nucleic Acids Res.* (2019).

[142] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *4th International Conference on Learning Representations, ICLR* (2016). Available from: http://arxiv.org/abs/1509.02971.

[143] Liu, J., Huang, Y., Singh, R., Vert, J.-P., and Noble, W. S. Jointly embedding multiple single-cell omics measurements. *Algorithms in Bioinformatics* (2019).

[144] Liu, X., Yu, Y., Guo, C., Sun, Y., and Gao, L. Full-text based context-rich heterogeneous network mining approach for citation recommendation. In *IEEE/ACM Joint Conference on Digital Libraries* (2014).

[145] Lu, Y., Xie, R., Shi, C., Fang, Y., Wang, W., Zhang, X., and Lin, L. Social influence attentive neural network for Friend-Enhanced recommendation. In *Machine Learning and Knowledge Discovery in Databases: Applied Data Science Track*. Springer International Publishing (2021).

[146] Luckow, A., Cook, M., Ashcraft, N., Weill, E., Djerekarov, E., and Vorster, B. Deep learning in the automotive industry: Applications and tools. In *2016 IEEE International Conference on Big Data* (2016).

[147] Lukovszki, T. and Schmid, S. Online admission control and embedding of service chains. In *Structural Information and Communication Complexity*. Springer International Publishing (2015).

[148] Luong, N. C., Hoang, D. T., Gong, S., Niyato, D., Wang, P., Liang, Y.-C., and Kim, D. I. Applications of deep reinforcement learning in communications and networking: A survey. *IEEE Communications Surveys Tutorials* (2019).

[149] Ma, K. J., Bartoš, R., and Bhatia, S. A survey of schemes for internet-based video delivery. *Journal of Network and Computer Applications* (2011).

[150] Machanick, P. and Bailey, T. L. Meme-chip: motif analysis of large dna datasets. *Bioinformatics* (2011).

[151] Mao, B., Fadlullah, Z. M., Tang, F., Kato, N., Akashi, O., Inoue, T., and Mizutani, K. Routing or computing? the paradigm shift towards intelligent computer network packet transmission based on deep learning. *IEEE Transactions on Computers* (2017).

[152] Mao, W., Wang, L., Zhao, J., and Xu, Y. Online fault-tolerant VNF chain placement: A deep reinforcement learning approach. In *2020 IFIP Networking Conference (Networking)* (2020).

[153] Marcheggiani, D. and Titov, I. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark (2017). Available from: https://aclanthology.org/D17-1159, doi:10.18653/v1/D17-1159.

[154] Marcus, G. Innateness, alphazero, and artificial intelligence. *arXiv preprint arXiv:1801.05667* (2018).

[155] Marotta, A. and Kassler, A. A power efficient and robust virtual network functions placement problem. In *2016 28th International Teletraffic Congress (ITC 28)* (2016).

[156] Marotta, A., Zola, E., D'Andreagiovanni, F., and Kassler, A. A fast robust optimization-based heuristic for the deployment of green virtual network functions. *Journal of Network and Computer Applications* (2017).

[157] Matias, J., Garay, J., Toledo, N., Unzilla, J., and Jacob, E. Toward an SDN-enabled NFV architecture. *IEEE Communications Magazine* (2015).

[158] McDowell, I. C., Manandhar, D., Vockley, C. M., Schmid, A. K., Reddy, T. E., and Engelhardt, B. E. Clustering gene expression time series data using an infinite gaussian process mixture model. *PLOS Computational Biology* (2018).

[159] McGoff, K. A., Guo, X., Deckard, A., Kelliher, C. M., Leman, A. R., Francey, L. J., Hogenesch, J. B., Haase, S. B., and Harer, J. L. The local edge machine: inference of dynamic models of gene regulation. *Genome biology* (2016).

[160] McInnes, L., Healy, J., and Melville, J. UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426* (2018). arXiv:1802.03426.

[161] McLeay, R. C. and Bailey, T. L. Motif enrichment analysis: a unified framework and an evaluation on chip data. *BMC bioinformatics* (2010).

[162] Meegoda Jay N. and Gao Shengyan. Roughness progression model for asphalt pavements using Long-Term pavement performance data. *Journal of Transportation Engineering* (2014).

[163] Mei, X., Cai, X., Yang, L., and Wang, N. Relation-aware heterogeneous graph transformer based drug repurposing. *Expert Systems with Applications* (2022).

[164] Merkwirth, C. and Lengauer, T. Automatic generation of complementary descriptors with molecular graph networks. *Journal of Chemical Information and Modeling* (2005).

[165] Michailidis, G. and d'Alché Buc, F. Autoregressive models for gene regulatory network inference: Sparsity, stability and causality issues. *Mathematical biosciences* (2013).

[166] Michelle M. Li, Kexin Huang, M. Zitnik. Graph representation learning in biomedicine. *arXiv* (2021). arXiv:2104.04883.

[167] Min, E., Guo, X., Liu, Q., Zhang, G., Cui, J., and Long, J. A survey of clustering with deep learning: From the perspective of network architecture. *IEEE Access* (2018).

[168] Mingshuo, N., Dongming, C., and Dongqi, W. Reinforcement learning on graphs: A survey. *arXiv preprint arXiv:2204.06127* (2022). arXiv:2204.06127.

[169] Mishra, M., Nayak, J., Naik, B., and Abraham, A. Deep learning in electrical utility industry: A comprehensive review of a decade of research. *Engineering Applications of Artificial Intelligence* (2020).

[170] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *Proceedings of The 33rd International Conference on Machine Learning* (edited by M. F. Balcan and K. Q. Weinberger), vol. 48 of *Proceedings of Machine Learning Research*. PMLR, New York, New York, USA (2016).

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

[172] Mnih, V., et al. Human-level control through deep reinforcement learning. *Nature* (2015).

[173] Molina, B., Palau, C. E., and Esteve, M. Modeling content delivery networks and their performance. *Computer Communications* (2004).

[174] Molina, B., Palau, C. E., and Esteve, M. CDN modeling and performance. *Next Generation Content Delivery Infrastructures* (2012).

[175] Mora, A., Sandve, G. K., Gabrielsen, O. S., and Eskeland, R. In the loop: promoter–enhancer interactions and bioinformatics. *Briefings in Bioinformatics* (2015).

[176] Mosavi, A., Faghan, Y., Ghamisi, P., Duan, P., Ardabili, S. F., Salwana, E., and Band, S. S. Comprehensive review of deep reinforcement learning methods and applications in economics. *Science in China, Series A: Mathematics* (2020).

[177] Munkhbat, B. *A Computational Simulation Model for Predicting Infectious Disease Spread Using the Evolving Contact Network Algorithm*. University of Massachusetts Libraries (2019).

[178] Murphy, R. L., Srinivasan, B., Rao, V., and Ribeiro, B. Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs. In *International Conference on Learning Representations* (2019). Available from: https://openreview.net/forum?id=BJluy2RcFm.

[179] Nachum, O., Norouzi, M., Xu, K., and Schuurmans, D. Bridging the gap between value and policy based reinforcement learning. In *Advances in neural information processing systems*, vol. 30 (2017).

[180] Nascimento, M. C. V. and de Carvalho, A. C. P. L. F. Spectral methods for graph clustering – a survey. *European Journal of Operational Research* (2011).

[181] Nayak, K. and Panigrahy, S. K. Application of machine learning to improve tourism industry. *Design of Intelligent Applications Using Machine Learning and Deep Learning Techniques* (2021).

[182] Newman, M. *Networks*. Oxford University Press (2018).

[183] Ng, A. Y., Harada, D., and Russell, S. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning*, vol. 99 (1999).

[184] Nguyen, N. D. and Wang, D. Multiview learning for understanding functional multiomics. *PLOS Computational Biology* (2020).

[185] Nguyen, T. T. and Reddi, V. J. Deep reinforcement learning for cyber security. *IEEE Transactions on Neural Networks and Learning Systems* (2021). doi:10.1109/TNNLS.2021.3121870.

[186] Ni, J., Chang, S., Liu, X., Cheng, W., Chen, H., Xu, D., and Zhang, X. Co-Regularized deep Multi-Network embedding. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE (2018).

[187] Nickel, M., Tresp, V., and Kriegel, H.-P. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11. Omnipress, Madison, WI, USA (2011).

[188] Nourani, E. GoVec: Gene ontology representation learning using weighted heterogeneous graph and Meta-Path. *Journal of Computational Biology* (2021).

[189] Nunner, H., Buskens, V., and Kretzschmar, M. A model for the co-evolution of dynamic social networks and infectious disease dynamics. *Comput Soc Netw* (2021).

[190] Obara, M., Kashiyama, T., and Sekimoto, Y. Deep reinforcement learning approach for train rescheduling utilizing graph theory. In *2018 IEEE International Conference on Big Data (Big Data)* (2018).

[191] Onayev, A. and Swei, O. IRI deterioration model for asphalt concrete pavements: capturing performance improvements over time. *Construction and Building Materials* (2021).

[192] Oskolkov, N. How exactly umap works (2019). Available from: https://towardsdatascience.com/how-exactly-umap-works-13e3040e1668.

[193] Ou, M., Cui, P., Pei, J., Zhang, Z., and Zhu, W. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16. Association for Computing Machinery, New York, NY, USA (2016).

[194] Özgül, O. F., Bardak, B., and Tan, M. A convolutional deep clustering framework for gene expression time series. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (2021).

[195] Pagiatakis, C., Musolino, E., Gornati, R., Bernardini, G., and Papait, R. Epigenetics of aging and disease: a brief overview. *Aging Clinical and Experimental Research* (2021).

[196] Pandey, S., Choi, M. J., and Park, S. The evolution of over the top (OTT): Standardization, key players and challenges. *Majlesi Journal of Electrical Engineering* (2019).

[197] Parhi, R. and Nowak, R. D. What kinds of functions do deep neural networks learn? insights from variational spline theory. *SIAM Journal on Mathematics of Data Science* (2022).

[198] Pathan, A. M. K. and Buyya, R. A taxonomy and survey of content delivery networks. *Grid Computing and Distributed Systems* (2007).

[199] Pavlopoulos, G. A., Secrier, M., Moschopoulos, C. N., Soldatos, T. G., Kossida, S., Aerts, J., Schneider, R., and Bagos, P. G. Using graph theory to analyze biological networks. *BioData Mining* (2011).

[200] Pei, J., Hong, P., Pan, M., Liu, J., and Zhou, J. Optimal VNF placement via deep reinforcement learning in SDN/NFV-Enabled networks. *IEEE Journal on Selected Areas in Communications* (2020).

[201] Pei, J., Hong, P., Xue, K., and Li, D. Resource aware routing for service function chains in SDN and NFV-Enabled network. *IEEE Transaction on Services Compututing* (2018).

[202] Pei, J., Hong, P., Xue, K., and Li, D. Efficiently embedding service function chains with dynamic virtual network function placement in Geo-Distributed cloud system. *IEEE Transactions on Parallel and Distributed Systems* (2019).

[203] Pérez, S., Arroba, P., and Moya, J. M. Energy-conscious optimization of edge computing through deep reinforcement learning and two-phase immersion cooling. *Future Generation Computer Systems* (2021).

[204] Perozzi, B., Al-Rfou, R., and Skiena, S. DeepWalk: online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '14. Association for Computing Machinery, New York, NY, USA (2014).

[205] Pham, T., Tran, T., Phung, D., and Venkatesh, S. Column networks for collective classification. In *Thirty-First AAAI Conference on Artificial Intelligence* (2017).

[206] Pratapa, A., Jalihal, A. P., Law, J. N., Bharadwaj, A., and Murali, T. M. Benchmarking algorithms for gene regulatory network inference from single-cell transcriptomic data. *Nature Methods* (2020).

[207] Ptashne, M. and Gann, A. *Genes & signals*, vol. 2. Cold Spring Harbor Laboratory Press Cold Spring Harbor, NY: (2002).

[208] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017).

[209] Qu, M., Tang, J., and Han, J. Curriculum learning for heterogeneous star network embedding via deep reinforcement learning. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18. Association for Computing Machinery, New York, NY, USA (2018).

[210] Quang, P. T. A., Hadjadj-Aoul, Y., and Outtagarts, A. A deep reinforcement learning approach for VNF forwarding graph embedding. *IEEE Transactions on Network and Service Management* (2019).

[211] Rao, A., Vg, S., Joseph, T., Kotte, S., Sivadasan, N., and Srinivasan, R. Phenotype-driven gene prioritization for rare diseases using graph convolution on heterogeneous networks. *BMC Med. Genomics* (2018).

[212] Rao, R. V. *Decision Making in the Manufacturing Environment: Using Graph Theory and Fuzzy Multiple Attribute Decision Making Methods*. Springer, London (2007).

[213] Reis, J., Rocha, M., Phan, T. K., Griffin, D., Le, F., and Rio, M. Deep neural networks for network routing. In *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE (2019).

[214] REPORTS AND DATA. Bioinformatics market by products & services, by application, and by end-use and by region forecast to 2028 (2021). Available from: https://www.reportsanddata.com/download-summary-form/1403.

[215] Reznik, Y., Teixeira, T., and Peck, R. On multiple media representations and CDN performance. In *Proceedings of the 1st Mile-High Video Conference* (2022).

[216] Ristevski, B. A survey of models for inference of gene regulatory networks. *Nonlinear Analysis: Modelling and Control* (2013).

[217] Robinson, D. *Content Delivery Networks: Fundamentals, Design, and Evolution*. John Wiley & Sons (2017).

[218] Ross-Innes, C. S., et al. Differential oestrogen receptor binding is associated with clinical outcome in breast cancer. *Nature* (2012).

[219] Rothenberg, E. V. Causal gene regulatory network modeling and genomics: Second-Generation challenges. *Journal of Computational Biology* (2019).

[220] Sadavare, A. B. and Kulkarni, R. V. A review of application of graph theory for network. *Journal of Chemical Information and Modeling* (2012).

[221] Sadeghi, M. and Armanfard, N. Deep clustering with self-supervision using pairwise data similarities. *TechRxiv: preprint* (2021).

[222] Saint-André, V. Computational biology approaches for mapping transcriptional regulatory networks. *Computational and Structural Biotechnology Journal* (2021).

[223] Sánchez-García, R. J., Cozzo, E., and Moreno, Y. Dimensionality reduction and spectral properties of multilayer networks. *Physical Review E* (2014).

[224] Santos, G. L., Kelner, J., Sadok, D., and Endo, P. T. Using reinforcement learning to allocate and manage SFC in cellular networks. In *2020 16th International Conference on Network and Service Management (CNSM)* (2020).

[225] Sato, R. A survey on the expressive power of graph neural networks. *arXiv preprint arXiv:2003.04078* (2020). arXiv:2003.04078.

[226] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Transactions on Neural Networks and Learning Systems* (2009).

[227] Schäfer, J. and Strimmer, K. An empirical bayes approach to inferring large-scale gene association networks. *Bioinformatics* (2005).

[228] Schardong, F., Nunes, I., and Schaeffer-Filho, A. NFV resource allocation: a systematic review and taxonomy of VNF forwarding graph embedding. *Computer Networks* (2021).

[229] Schlichtkrull, M., Kipf, T. N., Bloem, P., van den Berg, R., Titov, I., and Welling, M. Modeling relational data with graph convolutional networks. In *The Semantic Web*. Springer International Publishing (2018).

[230] Schlitt, T. and Brazma, A. Current approaches to gene regulatory network modelling. *BMC Bioinformatics* (2007).

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

[232] Sewak, M. Policy-Based reinforcement learning approaches. In *Deep Reinforcement Learning: Frontiers of Artificial Intelligence* (edited by M. Sewak), pp. 127–140. Springer Singapore, Singapore (2019).

[233] Shaikh, S. Deep reinforcement learning with accelerated reward function technique for robotics task planning. *San Jose State University ProQuest Dissertations Publishing* (2021).

[234] Shen, Y., Chen, J., Huang, P.-S., Guo, Y., and Gao, J. M-walk: Learning to walk over graphs using monte carlo tree search. *arXiv* (2018). arXiv:1802.04394, doi:10.48550/ARXIV.1802.04394.

[235] Shi, C., Wang, X., and Yu, P. S. Heterogeneous graph representation for industry application. *Artificial Intelligence: Foundations, Theory, and Algorithms* (2022).

[236] Shi, C., Wang, X., and Yu, P. S. *Heterogeneous Graph Representation Learning and Applications*. Springer Singapore (2022).

[237] Siemers, M., Lazaratos, M., Karathanou, K., Guerra, F., Brown, L. S., and Bondar, A.-N. Bridge: A Graph-Based algorithm to analyze dynamic H-Bond networks in membrane proteins. *Journal of Chemical Theory and Computation* (2019).

[238] Sima, C., Hua, J., and Jung, S. Inference of gene regulatory networks using Time-Series data: A survey. *Current Genomics* (2009).

[239] Singhal, A., Sinha, P., and Pant, R. Use of deep learning in modern recommendation system: A summary of recent works. *arXiv* (2017). arXiv:1712.07525.

[240] Sinha, K., Sodhani, S., Dong, J., Pineau, J., and Hamilton, W. L. CLUTRR: A diagnostic benchmark for inductive reasoning from text. 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).

[241] Sokolov, V. Discussion of 'deep learning for finance: deep portfolios'. *Applied Stochastic Models in Business and Industry* (2017).

[242] Stanley, J. S., 3rd, Gigante, S., Wolf, G., and Krishnaswamy, S. Harmonic alignment. *Proceedings of the 2021 SIAM International Conference on Data Mining* (2020).

[243] Steck, H., Baltrunas, L., Elahi, E., Liang, D., Raimond, Y., and Basilico, J. Deep learning for recommender systems: A netflix case study. *AI Magazine* (2021).

[244] Stocker, V., Smaragdakis, G., Lehr, W., and Bauer, S. The growing complexity of content delivery networks: Challenges and implications for the internet ecosystem. *Telecomm. Policy* (2017).

[245] StriveCast. What is a content delivery network (CDN)? (2017). Accessed: 2021-10-10. Available from: https://strivecast.com/content-delivery-network-cdn/.

[246] Stuart, T., et al. Comprehensive integration of Single-Cell data. *Cell* (2019).

[247] Subramanian, A., et al. Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences* (2005).

[248] Sun, C., Bi, J., Zheng, Z., and Hu, H. SLA-NFV: an SLA-aware high performance framework for network function virtualization. In *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16. Association for Computing Machinery, New York, NY, USA (2016).

[249] Sutton, R. S. Learning to predict by the methods of temporal differences. *Machine Learning* (1988).

[250] Sutton, R. S., Barto, A. G., et al. *Introduction to reinforcement learning*, vol. 2. MIT press Cambridge (1998).

[251] Szepesvári, C. Algorithms for reinforcement learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* (2010).

[252] Tabatabaee, N., Ziyadi, M., and Shafahi, Y. Two-stage support vector classifier and recurrent neural network predictor for pavement performance modeling. *Journal of Infrastructure Systems* (2013).

[253] Tahghigh Jahromi, N. *Towards the Softwarization of Content Delivery Networks for Component and Service Provisioning.* Ph.D. thesis, Concordia University (2018).

[254] Tan, H., Tang, W., Fan, X., Jing, Q., and Bi, J. SERL: Semantic-Path biased representation learning of heterogeneous information network. In *Knowledge Science, Engineering and Management.* Springer International Publishing (2018).

[255] Tang, B., Pan, Z., Yin, K., and Khateeb, A. Recent advances of deep learning in bioinformatics and computational biology. *Frontiers in Genetics* (2019).

[256] Tesauro, G. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Comput.* (1994).

[257] Tomassilli, A., Giroire, F., Huin, N., and Pérennes, S. Provably efficient algorithms for placement of service function chains with ordering constraints. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications* (2018).

[258] Van der Maaten, L. and Hinton, G. Visualizing data using t-SNE. *Journal of Machine Learning Research* (2008).

[259] Veličković, P. and Blundell, C. Neural algorithmic reasoning. *Patterns* (2021).

[260] Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *6th International Conference on Learning Representations, ICLR* (2018).

[261] Vesselinova, N., Steinert, R., Perez-Ramirez, D. F., and Boman, M. Learning combinatorial optimization on graphs: A survey with applications to networking. *IEEE Access* (2020).

[262] Vetrík, T. Extended study of biological networks using graph theory. *Proyecciones (Antofagasta)* (2021).

[263] Villiers, B. d., de Villiers, B., and Sabatta, D. Hindsight reward shaping in deep reinforcement learning. In *2020 International SAUPEC/RobMech/PRASA Conference* (2020).

[264] Virmani, N., Salve, U. R., Kumar, A., and Luthra, S. Analyzing roadblocks of industry 4.0 adoption using graph theory and matrix approach. *IEEE Transactions on Engineering Management* (2021).

[265] Waltz, M. and Fu, K. A heuristic approach to reinforcement learning control systems. *IEEE Transactions on Automatic Control* (1965).

[266] Wang, B., Sun, Y., Duong, T. Q., Nguyen, L. D., and Hanzo, L. Risk-Aware identification of highly suspected COVID-19 cases in social IoT: A joint graph theory and reinforcement learning approach. *IEEE Access* (2020).

[267] Wang, C., Xu, S., and Yang, J. Adaboost algorithm in artificial intelligence for optimizing the IRI prediction accuracy of asphalt concrete pavement. *Sensors* (2021).

[268] Wang, S., Chen, X., and Xiong, S. Attention based reinforcement learning with reward shaping for knowledge graph reasoning. In *Natural Language Processing and Chinese Computing*. Springer International Publishing (2021).

[269] Wang, X., Gong, Y., Yi, J., and Zhang, W. Predicting gene-disease associations from the heterogeneous network using graph embedding. In *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)* (2019).

[270] Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P., and Yu, P. S. Heterogeneous graph attention network. In *The World Wide Web Conference*, WWW '19. Association for Computing Machinery, New York, NY, USA (2019).

[271] Wang, X., Liu, N., Han, H., and Shi, C. Self-supervised heterogeneous graph neural network with co-contrastive learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21. Association for Computing Machinery, New York, NY, USA (2021).

[272] Wang, Y., Yao, H., and Zhao, S. Auto-encoder based dimensionality reduction. *Neurocomputing* (2016).

[273] Wang, Z., Gerstein, M., and Snyder, M. RNA-Seq: a revolutionary tool for transcriptomics. *Nature Reviews Genetics* (2009).

[274] Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning* (edited by M. F. Balcan and K. Q. Weinberger), vol. 48 of *Proceedings of Machine Learning Research*. New York, New York, USA (2016).

[275] Wang, Z., Zhang, J., Feng, J., and Chen, Z. Knowledge graph embedding by translating on hyperplanes. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 1 (2014).

[276] Watkins, C. J. C. H. and Dayan, P. Q-learning. *Machine Learning* (1992).

[277] Welling, M. and Kipf, T. N. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR* (2017).

[278] West, D. M. The evolution of video streaming and digital content delivery (2014). Available from: https://www.brookings.edu/wp-content/uploads/2016/06/West_Evolution-of-VideoStreaming-and-Digital-Content-Delivery_Final.pdf.

[279] Wittkopp, P. J. and Kalay, G. Cis-regulatory elements: molecular mechanisms and evolutionary processes underlying divergence. *Nature Reviews Genetics* (2011).

[280] Wowza Media Systems. 4 tips for sizing streaming server hardware (2017). Available from: https://www.wowza.com/blog/4-tips-for-sizing-streaming-server-hardware.

[281] Wu, T., et al. clusterprofiler 4.0: A universal enrichment tool for interpreting omics data. *The Innovation* (2021).

[282] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* (2021).

[283] Xi, J. and Yu, Z. *Unsupervised Learning Models for Unlabeled Genomic, Transcriptomic & Proteomic Data*. Frontiers Media SA (2022).

[284] Xia, K., Sacco, C., Kirkpatrick, M., Saidy, C., Nguyen, L., Kircaliali, A., and Harik, R. A digital twin to train deep reinforcement learning agent for smart manufacturing plants: Environment, interfaces and intelligence. *Journal of Manufacturing Systems* (2021).

[285] Xiao, Y., Zhang, Q., Liu, F., Wang, J., Zhao, M., Zhang, Z., and Zhang, J. NFVdeep: adaptive online service function chain deployment with deep reinforcement learning. In *Proceedings of the International Symposium on Quality of Service*, no. Article 21 in IWQoS '19. Association for Computing Machinery, New York, NY, USA (2019).

[286] Xie, Y., Liu, Z., Wang, S., and Wang, Y. Service function chaining resource allocation: A survey. *arXiv* (2016). arXiv:1608.00095.

[287] Xie, Y., Yu, B., Lv, S., Zhang, C., Wang, G., and Gong, M. A survey on heterogeneous network representation learning. *Pattern Recognition* (2021).

[288] Xiong, W., Hoang, T., and Wang, W. Y. DeepPath: A reinforcement learning method for knowledge graph reasoning. *arXiv* (2017). arXiv:1707.06690.

[289] Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.-I., and Jegelka, S. Representation learning on graphs with jumping knowledge networks. In *Proceedings of the 35th International Conference on Machine Learning* (edited by J. Dy and A. Krause), vol. 80 of *Proceedings of Machine Learning Research*. PMLR (2018).

[290] Yala, L., Frangoudis, P. A., Lucarelli, G., and Ksentini, A. Cost and availability aware resource allocation and virtual function placement for CDNaaS provision. *IEEE Transactions on Network and Service Management* (2018).

[291] Yan, S., Xu, D., Zhang, B., Zhang, H.-J., Yang, Q., and Lin, S. Graph embedding and extensions: a general framework for dimensionality reduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2007).

[292] Yan, Z., Ge, J., Wu, Y., Li, L., and Li, T. Automatic virtual network embedding: A deep reinforcement learning approach with graph convolutional networks. *IEEE Journal on Selected Areas in Communications* (2020).

[293] Yang, B., Fu, X., Sidiropoulos, N. D., and Hong, M. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *Proceedings of the 34th International Conference on Machine Learning* (edited by D. Precup and Y. W. Teh), vol. 70 of *Proceedings of Machine Learning Research*. PMLR (2017).

[294] Yang, B., Yih, W.-T., He, X., Gao, J., and Deng, L. Embedding entities and relations for learning and inference in knowledge bases. *2nd International Conference on Learning Representations, ICLR* (2014).

[295] Yang, C., Xiao, Y., Zhang, Y., Sun, Y., and Han, J. Heterogeneous network representation learning: A unified framework with survey and benchmark. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[296] Yang, C., Xiao, Y., Zhang, Y., Sun, Y., and Han, J. Heterogeneous network representation learning: Survey, benchmark, evaluation, and beyond. *Computing Research Repository, CoRR* (2020). Available from: https://arxiv.org/abs/2004.00216, arXiv:2004.00216.

[297] Yang, K. and Toni, L. GRAPH-BASED RECOMMENDATION SYSTEM. In *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. ieeexplore.ieee.org (2018).

[298] Yang, S., Yang, B., Kang, Z., and Deng, L. IHG-MA: Inductive heterogeneous graph multi-agent reinforcement learning for multi-intersection traffic signal control. *Neural Netw.* (2021).

[299] Yang, T., Hu, L., Shi, C., Ji, H., Li, X., and Nie, L. HGAT: Heterogeneous graph attention networks for semi-supervised short text classification. *ACM Transactions on Information Systems* (2021).

[300] Yi, B., Wang, X., Li, K., Das, S. k., and Huang, M. A comprehensive survey of network function virtualization. *Computer Networks* (2018).

[301] Yi, H.-C., You, Z.-H., Huang, D.-S., and Kwoh, C. K. Graph representation learning in bioinformatics: trends, methods and applications. *Briefings in Bioinformatics* (2021).

[302] Yin, Y., Ji, L.-X., Zhang, J.-P., and Pei, Y.-L. DHNE: Network representation learning method for dynamic heterogeneous networks. *IEEE Access* (2019).

[303] Yu, C., Liu, J., Nemati, S., and Yin, G. Reinforcement learning in healthcare: A survey. *ACM Computing Surveys* (2021).

[304] Yu, L., Sun, L., Du, B., Liu, C., Lv, W., and Xiong, H. Heterogeneous graph representation learning with relation awareness. *IEEE Transactions on Knowledge and Data Engineering* (2022).

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

[306] Yuan, M., Pun, M., Chen, Y., Wang, D., and Li, H. Multimodal reward shaping for efficient exploration in reinforcement learning. *Computing Research Repository, CoRR* (2021). Available from: https://arxiv.org/abs/2107.08888, arXiv:2107.08888.

[307] Yuan, Y. and Bar-Joseph, Z. Deep learning for inferring gene relationships from single-cell expression data. *Proceedings of the National Academy of Sciences U. S. A.* (2019).

[308] Yun, S., Jeong, M., Kim, R., Kang, J., and Kim, H. J. Graph transformer networks. *Advances in Neural Information Processing Systems* (2019).

[309] Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. Deep sets. *Advances in neural information processing systems* (2017).

[310] Zambaldi, V., et al. Relational deep reinforcement learning. *arXiv* (2018). arXiv:1806.01830.

[311] Zboril, E., Yoo, H., Chen, L., and Liu, Z. Dynamic interactions of transcription factors and enhancer reprogramming in cancer progression. *Frontiers in Oncology* (2021).

[312] Zhang, B. and Horvath, S. A general framework for weighted gene co-expression network analysis. *Statistical applications in genetics and molecular biology* (2005).

[313] Zhang, C., Song, D., Huang, C., Swami, A., and Chawla, N. V. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19. Association for Computing Machinery, New York, NY, USA (2019).

[314] Zhang, C., Swami, A., and Chawla, N. V. CARL: Content-Aware representation learning for heterogeneous networks. *arXiv* (2018). arXiv:1805.04983.

[315] Zhang, C., Swami, A., and Chawla, N. V. SHNE: Representation learning for Semantic-Associated heterogeneous networks. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, WSDM '19. Association for Computing Machinery, New York, NY, USA (2019).

[316] Zhang, D. and Bailey, C. P. Obstacle avoidance and navigation utilizing reinforcement learning with reward shaping. *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications II* (2020).

[317] Zhang, D., Yin, J., Zhu, X., and Zhang, C. MetaGraph2Vec: Complex semantic path augmented heterogeneous network embedding. In *Advances in Knowledge Discovery and Data Mining*. Springer International Publishing (2018).

[318] Zhang, R.-X., Ma, M., Huang, T., Pang, H., Yao, X., Wu, C., Liu, J., and Sun, L. Livesmart: A QoS-Guaranteed Cost-Minimum framework of viewer scheduling for crowdsourced live streaming. In *Proceedings of the 27th ACM International Conference on Multimedia*, MM '19. Association for Computing Machinery, New York, NY, USA (2019).

[319] Zhang, W., Shu, K., Liu, H., and Wang, Y. Graph neural networks for user identity linkage. *arXiv preprint arXiv:1903.02174* (2019). arXiv:1903.02174.

[320] Zhang, Z., Cui, P., and Zhu, W. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[321] Zhang, Z., Wang, X., and Zhu, W. Automated machine learning on graphs: A survey. *arXiv* (2021). arXiv:2103.00742.

[322] Zhang, Z., Zhang, D., and Qiu, R. C. Deep reinforcement learning for power system applications: An overview. *CSEE Journal of Power and Energy Systems* (2020).

[323] Zhao, K., Bai, T., Wu, B., Wang, B., Zhang, Y., Yang, Y., and Nie, J.-Y. Deep adversarial completion for sparse heterogeneous information network embedding. In *Proceedings of The Web Conference 2020*, pp. 508–518. Association for Computing Machinery, New York, NY, USA (2020).

[324] Zhao, M., He, W., Tang, J., Zou, Q., and Guo, F. A comprehensive overview and critical evaluation of gene regulatory network inference technologies. *Briefings in Bioinformatics* (2021).

[325] Zheng, B., Sage, M., Sheppeard, E. A., Jurecic, V., and Bradley, A. Engineering mouse chromosomes with Cre-loxP: range, efficiency, and somatic applications. *Molecular and Cellular Biology* (2000).

[326] Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. Graph neural networks: A review of methods and applications. *AI Open* (2020).

[327] Zhou, J., Liu, L., Wei, W., and Fan, J. Network representation learning: From preprocessing, feature extraction to node embedding. *ACM Computing Surveys* (2022).

[328] Zhou, K., Zhang, S., Wang, Y., Cohen, K. B., Kim, J.-D., Luo, Q., Yao, X., Zhou, X., and Xia, J. High-quality gene/disease embedding in a multi-relational heterogeneous graph after a joint matrix/tensor decomposition. *Journal of Biomedical Informatics* (2022).

[329] Zhou, S., Bu, J., Wang, X., Chen, J., and Wang, C. HAHE: Hierarchical attentive heterogeneous information network embedding. *arXiv* (2019). arXiv:1902.01475.