Graph-based learning and optimization

Taha Halal

université PARIS-SACLAY

Graph-based learning and optimization

Apprentissage et optimisation sur les graphes

Thèse de doctorat de l'université Paris-Saclay

École doctorale n° 580, sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat: Informatique
Graduate School: Informatique et sciences du numérique
Référent : Faculté des sciences d'Orsay

Thèse préparée au **Laboratoire interdisciplinaire des sciences du numérique** (Université Paris-Saclay, CNRS),
sous la direction de **Bogdan CAUTIS**, Professeur
et le co-encadrement de **Benoit GROZ**, Maître de conférences

Thèse soutenue à Paris-Saclay, le 17 Octobre 2024, par

Taha HALAL

Composition du jury

Membres du jury avec voix délibérative

Amel BOUZEGHOUB Professeure, Télécom SudParis	Présidente
Michalis VAZIRGIANNIS Professeur, Ecole Polytechnique, Palaiseau	Rapporteur & Examinateur
Cédric DU MOUZA Professeur, CNAM, Paris	Rapporteur & Examinateur
Chedy RAÏSSI Chercheur, INRIA Nancy Grand-Est, Université de Lorraine	Examinateur

THESE DE DOCTORAT

NNT : 2024UPASG043

université PARIS-SACLAY | ÉCOLE DOCTORALE Sciences et technologies de l'information et de la communication (STIC)

Titre: Apprentissage et optimisation sur les graphes
Mots clés: Apprentissage automatique; Réseaux de neurones en graphes; Réseaux Sociaux; Maximization d'influence; Graphe de connaissances; Prédiction de liens;

Résumé: Cette thèse utilise les réseaux neuronaux de graphes (GNNs) pour relever les défis des réseaux sociaux et des graphes de connaissances. Elle introduit des solutions novatrices pour la maximisation d'influence, identifiant les utilisateurs les plus influents dans les réseaux sociaux en considérant à la fois la structure du réseau et les sujets des messages. Les modèles proposés améliorent considérablement l'efficacité et sont particulièrement adaptés aux applications en temps réel. De plus, la thèse explore des méthodes pour prédire les liens manquants dans les graphes de connaissances avec des données limitées. Bien que cet aspect n'ait pas conduit à des avancées majeures, il a fourni des informations pertinentes et abouti au développement d'un prototype utilisé dans un projet réel. Globalement, cette thèse contribue à faire progresser notre compréhension des données complexes de graphes et offre des solutions pratiques pour des applications concrètes grâce à l'utilisation des GNNs.

Title: Graph-based learning and optimization
Keywords: Machine learning; Graph Neural Networks; Social Networks; Influence Maximization; Knowledge Graphs; Link Prediction;

Abstract: This research leverages Graph Neural Networks (GNNs) to address challenges in social networks and knowledge graphs. It introduces innovative solutions for influence maximization, identifying the most influential users in social networks by considering both network structure and message topics. The proposed models demonstrate significant improvements in efficiency and are particularly well-suited for real-time applications. Additionally, the research explores methods for predicting missing links in knowledge graphs under limited data conditions. While this aspect of the research didn't lead to major breakthroughs, it provided valuable insights and culminated in the development of a prototype utilized in a real-world project. Overall, this research contributes to advancing our understanding of complex graph data and offers practical solutions for real-world applications through the use of GNNs.

À ma famille, mon pilier

Remerciements

Ce manuscrit marque la fin de mon doctorat, une période à la fois exigeante et extrêmement enrichissante. J'ai passé près de quatre années à travailler dur, et ces années ont été synonymes d'une formidable évolution personnelle et professionnelle. Je n'aurais pas pu mener ce projet à bien sans le soutien exceptionnel de mes mentors, collaborateurs, amis et famille. Ils ont été là à chaque étape, me guidant et m'encourageant. Cette thèse leur est dédiée, en signe de ma profonde gratitude pour leur soutien inconditionnel.

Je tiens à exprimer ma plus profonde gratitude à mon directeur de thèse, Bogdan, pour son soutien indéfectible, sa patience inébranlable et ses précieux conseils toujours pertinents. Son expertise et sa vision m'ont été d'une aide précieuse tout au long de ce parcours. Je remercie également mon codirecteur, Benoît, pour sa présence constante, son souci du détail et son engagement envers l'excellence. Sa capacité à me pousser à toujours repousser mes limites a été une force dans la réalisation de ce travail. Je souhaite également remercier Bpifrance and IBM France pour avoir financé cette thèse dans le cadre du projet AIDA.

Je remercie du fond du cœur mes collègues du laboratoire, avec qui j'ai partagé bien plus que des heures de travail. Ces nouvelles amitiés ont illuminé mes journées, entre fous rires partagés, moments de découragement surmontés ensemble, pauses café interminables, parties endiablées de baby-foot et de billard, et la joie d'accueillir les papiers acceptés (et de surmonter les déceptions des refus). Un merci particulier à Armita, Alexandra et Lucas, ainsi qu'aux petits nouveaux Thibaut, Alan et Jonathan, pour leur soutien indéfectible et leur présence précieuse. Merci également à Joe, Fatiha et Emmanuel pour leur bienveillance. Je n'oublie pas non plus mes amis hors du labo, Nourredine, un grand soutien pendant cette thèse, oreille attentive et toujours de bon conseil, mes amis de longue date, Othmane, Amine, et Adam, mais aussi Manon et toute l'équipe de l'association Dans Ma Rue, Daniel, Anis et toute l'équipe de l'association Caradoc. Je remercie également toute l'équipe de boxe française du Club sportif de l'X, mes coachs, Pascal, Mathieu, Yann, Renaud, Nico, Alex, Gilles et mes coéquipiers, Mali, Max, Anne-Sophie, Gianni, Antonio, Vincent et les autres. Je n'oublie pas non plus mes amis de l'autre côté de la Méditerranée, Anas, Ayoub, Hamza, Oumayma, Maha, dont le soutien amical fut précieux dans les moments de doute et de fatigue, notamment lors de la rédaction de ce manuscrit. Merci à tous d'avoir toujours été là pour me soutenir, me rappeler que la vie est bien plus vaste que les murs du laboratoire, et m'aider à changer d'air. Vos encouragements, vos oreilles attentives et vos rires ont été un baume apaisant pendant ces années.

À ma famille, mon pilier indéfectible, je dédie cette thèse avec une gratitude infinie. Mes parents (Meriem & Abdelkhalek), ma tante préférée et mon oncle (Nozha & Ahmed). Votre amour inconditionnel, vos encouragements constants et votre présence bienveillante m'ont permis de surmonter les défis et de savourer chaque victoire. Vous avez toujours cru en moi, même lorsque j'ai douté, et votre soutien sans faille a été mon moteur principal. Du fond du cœur, merci d'être toujours là, présents et aimants, tout au long de cette aventure.

Et finalement, un mot pour ce jeune homme de 27 ans qui, un jour, a eu la folle audace de se lancer dans l'aventure d'une thèse. Une aventure guidée par une soif d'apprendre insatiable et le désir de toujours aller plus loin, de repousser les limites. Cette flamme, loin de s'éteindre, n'en est que plus vive aujourd'hui.

Abstract

Graphs are a fundamental data structure used to represent complex patterns in various domains. Graph Neural Networks (GNNs), a deep learning paradigm specifically designed for graph-structured data, offer a powerful deep learning solution for extracting meaningful patterns from these intricate relationships. This thesis explores the application of GNNs to address two key challenges: maximizing influence in social networks and predicting missing links in knowledge graphs with limited data. With applications ranging from optimizing public health campaigns and combating misinformation to knowledge base completion, this research addresses the need for computationally efficient and robust methods in these domains.

Influence maximization (IM) focuses on identifying the most influential nodes within a social network to maximize the spread of information or ideas. This thesis explores methods for tackling the IM problem, particularly in real-world scenarios with massive networks and diverse information themes. We build our models upon the S2V-DQN framework, a powerful approach that combines Deep Q-Networks for reinforcement learning with Structure2Vec for graph embedding. We first develop our IM-GNN model that incorporates advanced GNN features such as graph attention mechanisms and positional encoding, demonstrating competitive performance against existing learning-based and non-learning based methods for influence maximization.

We further extend our research to tackle Topic-aware Influence Maximization (TIM) where the spread of information is influenced by its thematic content, requiring models to consider not only network structure but also the topics of the messages being shared. This is where the limitations of traditional IM methods become apparent. Our TIM-GNN model effectively handles this complexity by incorporating topic-aware training and probabilistic methods for constructing topic-aware diffusion graphs. To address query latency concerns, we introduce TIM-GNN^x, which integrates cross-attention mechanisms and a pre-computed Q-matrix. Our experiments on real-world datasets demonstrate that our proposed model achieves competitive performance in terms of influence spread compared to state-of-the-art methods while also offering significant improvements in query time latency and robustness to changes in the diffusion graph. Notably, our TIM-GNN^x model strikes a balance between query efficiency and maximizing influence, making it particularly well-suited for real-time applications.

In the realm of knowledge graphs, we explore Few-Shot Link Prediction, where the goal is to predict missing relationships with limited training examples, which is crucial for addressing the long-tail phenomenon. This phenomenon refers to the fact that, in knowledge graphs, a large number of entities (nodes) and relations (edges) have very few connections or occurrences. This results in a distribution where a small number of popular entities or relations have many connections, while the vast majority have very few.

Our investigation focuses on the feasibility of integrating a path-based knowledge graph completion method (PathCon) with a meta-learning framework MetaR, to address the limitations of the latter. While our initial investigations did not yield significant improvements or notable scientific contributions, they provided valuable insights into the challenges of this task and informed the development of a prototype, deployed as an API, for the AIDA project. This prototype demonstrates the practical value of our research and paves the way for future explorations in this area.

Overall, this thesis contributes novel and efficient GNN-based solutions for influence maximization and explores promising directions for few-shot link prediction in knowledge graphs, pushing the boundaries of these research areas.

Résumé

Les graphes, structures de données fondamentales, sont utilisés pour représenter des schémas complexes dans divers domaines. Les réseaux de neurones graphiques (GNNs), un paradigme d'apprentissage profond conçu pour les données structurées en graphes, offrent une solution d'apprentissage profond efficace pour extraire des informations de ces relations complexes. Cette thèse explore l'application des GNNs pour relever deux défis clés : maximiser l'influence dans les réseaux sociaux et prédire les liens manquants dans les graphes de connaissances avec des données limitées. Avec des applications allant de l'optimisation des campagnes de santé publique et de la lutte contre la désinformation à la complétion des bases de connaissances, cette recherche répond au besoin de méthodes efficaces et robustes dans ces domaines.

La maximisation de l'influence (IM) se concentre sur l'identification des nœuds les plus influents au sein d'un réseau social pour maximiser la diffusion d'informations ou d'idées. Cette thèse explore des méthodes pour résoudre le problème de l'IM, en particulier dans des scénarios réels avec des réseaux massifs et divers thèmes d'information.

Nous construisons nos modèles en nous basant sur S2V-DQN, une approche qui combine les réseaux Deep Q-Networks pour l'apprentissage par renforcement avec Structure2Vec pour l'intégration de graphes. Nous développons d'abord notre modèle IM-GNN qui intègre des fonctionnalités GNN avancées telles que les mécanismes d'attention graphique et le codage positionnel, démontrant des performances concurrentielles par rapport aux méthodes existantes pour la maximisation de l'influence.

Nous étendons ensuite nos recherches pour aborder la maximisation de l'influence sensible au sujet (TIM) où la diffusion de l'information est influencée par son contenu thématique, exigeant que les modèles considèrent non seulement la structure du réseau mais aussi les sujets des messages partagés. C'est là que les limites des méthodes traditionnelles d'IM deviennent apparentes. Notre modèle TIM-GNN gère efficacement cette complexité en incorporant un entraînement sensible au sujet et des méthodes probabilistes pour construire des graphes de diffusion sensibles au sujet.

Pour résoudre les problèmes de latence des requêtes, nous introduisons TIM-GNNx, qui intègre des mécanismes d'attention croisée et une matrice Q précalculée. Nos expériences sur des ensembles de données réels démontrent que notre modèle atteint des performances concurrentielles en termes de diffusion d'influence par rapport aux méthodes de l'état de l'art tout en offrant des améliorations significatives en termes de latence et de robustesse. Notre modèle TIM-GNNx trouve un équilibre entre l'efficacité des requêtes et la maximisation de l'influence, ce qui le rend particulièrement adapté aux applications en temps réel.

Dans le domaine des graphes de connaissances, nous explorons la prédiction de liens à peu d'exemples, où l'objectif est de prédire les relations manquantes avec des exemples d'entraînement limités..

Notre étude se concentre sur la possibilité d'intégrer une méthode de complétion de graphe de connaissances basée sur les chemins, PathCon, avec un cadre de méta-apprentissage MetaR pour résoudre les limites de ce dernier. Bien que nos recherches initiales n'aient pas apporté d'améliorations significatives ou de contributions scientifiques notables, elles ont fourni des informations pertinentes sur les défis de cette tâche et ont éclairé le développement d'un prototype pour le projet AIDA. Ce prototype démontre la valeur pratique de nos recherches et ouvre la voie à de futures explorations dans ce domaine.

Dans l'ensemble, cette thèse apporte des solutions nouvelles et efficaces basées sur GNN pour la maximisation de l'influence et explore des pistes prometteuses pour la prédiction de liens à peu d'exemples dans les graphes de connaissances, repoussant les limites de ces domaines de recherche.

Résumé étendu

Les graphes constituent une structure de données fondamentale pour représenter des schémas complexes dans une variété de domaines. Les Réseaux de Neurones Graphiques (GNN), paradigme d'apprentissage profond spécifiquement conçu pour les données structurées en graphes, offrent une solution puissante pour extraire des motifs significatifs à partir de ces relations complexes. Cette thèse explore l'application des GNN pour relever deux défis majeurs : la maximisation de l'influence dans les réseaux sociaux et la prédiction de liens manquants dans les graphes de connaissances, notamment en contexte de données limitées. Ces recherches ont des implications importantes dans plusieurs domaines, allant de l'optimisation des campagnes de santé publique et de la lutte contre la désinformation à la complétion de bases de connaissances, en passant par la recommandation de produits et services. Ce travail répond ainsi au besoin croissant de méthodes robustes et performantes pour l'analyse de données complexes et interconnectées.

La maximisation de l'influence (IM) se concentre sur l'identification des nœuds les plus influents d'un réseau social afin de maximiser la diffusion d'informations ou d'idées. Cette thèse propose des solutions innovantes pour le problème de l'IM, en se concentrant sur les scénarios réalistes où les réseaux sont massifs et l'information est thématiquement diverse. Nos modèles reposent sur le cadre S2V-DQN, une approche combinant les Deep Q-Networks pour l'apprentissage par renforcement et Structure2Vec pour l'intégration de graphes. Cette approche permet d'apprendre des stratégies de sélection de nœuds optimisées en fonction de la structure du graphe et des dynamiques de diffusion. Nous introduisons IM-GNN, un modèle intégrant des mécanismes d'attention, un codage positionnel via le Laplacien magnétique, et l'ajout d'auto-boucles. Ces améliorations permettent de capturer des informations structurelles plus riches et d'améliorer la performance prédictive du modèle. L'évaluation sur des données réelles démontre la compétitivité de IM-GNN par rapport aux méthodes de l'état de l'art.

L'influence dans les réseaux sociaux est souvent modulée par la thématique de l'information diffusée. Pour refléter cette réalité, nous abordons la Maximisation de l'Influence Sensible au Thème (TIM), où le contenu thématique de l'information joue un rôle crucial dans le processus de diffusion. Nous proposons TIM-GNN, une extension de notre modèle qui intègre la dimension thématique dans l'apprentissage et la prédiction. Pour ce faire, nous utilisons des graphes de diffusion sensibles au thème construits à partir de données réelles de cascades d'information, en utilisant le modèle de factorisation de survie. De plus, nous introduisons TIM-GNNx, une version optimisée pour la latence des requêtes. TIM-GNNx utilise une matrice Q précalculée et des mécanismes d'attention croisée pour accélérer significativement le processus de sélection des nœuds influents, permettant ainsi des applications en temps réel. Nos expériences montrent que TIM-GNNx réalise un compromis optimal entre l'efficacité des requêtes et la performance en termes de propagation d'influence.

Dans le cadre du projet AIDA, nous nous sommes intéressés à la prédiction de liens dans les graphes de connaissances en contexte de données limitées, un problème crucial pour la complétion de bases de connaissances et la construction de modèles d'organisation d'entreprise. Le but est de prédire les relations manquantes entre les entités avec un nombre limité d'exemples d'apprentissage (apprentissage few-shot). Nous explorons l'intégration de PathCon, une méthode basée sur les chemins pour la complétion de graphes de connaissances, avec le framework MetaR, efficace en contexte de données limitées.. PathCon exploite la structure du graphe en modélisant le contexte relationnel et les chemins entre les entités, offrant ainsi une meilleure capacité de généralisation. Nous avons développé un prototype d'API pour la complétion de modèles organisationnels, démontrant la faisabilité de l'approche few-shot. Bien que les résultats de l'hybridation PathCon-MetaR n'aient pas montré de gains significatifs, nos analyses ont permis d'identifier des pistes d'amélioration pour les travaux futurs, notamment l'exploration de fonctions de score plus expressives et de nouvelles

stratégies d'intégration.

En conclusion, cette thèse démontre le potentiel des GNNs pour la maximisation d'influence et la prédiction de liens few-shot. Nos modèles TIM-GNN et TIM-GNNx offrent des solutions performantes, efficientes et robustes pour l'influence maximisée sensible au thème. Nos travaux sur la prédiction few-shot, bien que n'ayant pas abouti à des gains significatifs, ouvrent des pistes prometteuses pour l'intégration de méthodes structurelles avec le méta-apprentissage, et soulignent le potentiel des GNNs pour l'analyse de données graphées.

Leveraging Large Language Models (LLMs) in Thesis Writing

The field of artificial intelligence is rapidly evolving, with large language models leading the way in text generation capabilities. While my PhD thesis remained fundamentally a product of my own research and writing, as an AI researcher, I was naturally drawn to explore how these LLMs could augment my capabilities throughout the process, specifically utilizing models like GPT-4, Gemini Advanced, and Gemini 1.5. This chapter examines my experience using LLMs for specific tasks, as research assistants rather than original content generators, the ethical considerations involved, and the guidelines I developed to ensure academic integrity within this process.

Ethical considerations and guidelines From the beginning, I was aware of the ethical implications surrounding AI-assisted writing. These concerns include plagiarism, the potential spread of misinformation, and biases that LLMs may inherit from their training data. I established clear guidelines from the outset, positioning LLMs as tools to support my work, never to replace my own critical thinking. All LLM-generated content, particularly when used for factual information retrieval, was meticulously fact-checked using reliable sources. This ensured accuracy and maintained the academic rigor expected of a PhD thesis.

Practical use of LLMs Within these guidelines, I utilized LLMs for *research*, *writing*, and *code generation*. In terms of research, LLMs provided fresh perspectives and strengthened my arguments by brainstorming concepts and considering alternative viewpoints, serving as a good starting point for further evidence-based exploration rooted in scientific literature. For writing, LLMs not only identified grammatical errors and inconsistencies, but also suggested stylistic improvements. This significantly enhanced the clarity and conciseness of my writing. Finally, LLMs proved useful in generating code for automating simple but repetitive tasks, allowing me to focus on more complex aspects of my research.

Computational Ressources

This work was granted access to the HPC resources of IDRIS under the allocation AD011013756R1 made by GENCI.
We also used computational resources from the “Mésocentre” computing center of Université Paris-Saclay, CentraleSupélec and École Normale Supérieure Paris-Saclay supported by CNRS and Région Île-de-France (https://mesocentre.universite-paris-saclay.fr/).

Contents

Chapter 1

Introduction

The unprecedented surge in data generated across digital platforms calls for sophisticated analytical techniques to uncover hidden patterns within intricate network structures. Graph theory plays a fundamental role in deciphering these complex systems. Indeed, graphs offer a versatile way to represent diverse relationships and structures, making them essential tools in fields ranging from social network analysis to the study of biological networks and information retrieval systems.

1.1 Everything is connected

Graphs are fundamental data structures consisting of nodes (or vertices) and edges, which represent various types of relationships between these nodes. In many ways, graphs serve as the main modality of data [1], their ability to model a wide range of patterns observed in natural and artificial systems making them highly versatile. The adaptability of graphs in representing entities as diverse as molecules, social networks, and transportation systems highlights their significance, fostering keen interest from both scientific and industrial communities with diverse applications.

1.2 Knowledge Graphs: A distinct graph paradigm

Knowledge Graphs (KGs) are a unique type of graph designed to represent and connect information in a structured manner that mirrors human understanding. KGs consist of nodes representing entities (e.g., people, places, objects) and directed, labeled edges representing relationships between these entities (e.g., "located in", "works at", "is a"). Additionally, both nodes and edges in a KG can have attributes that provide further information. KGs find significant applications in areas such as semantic search, personalized recommendations, and data integration.

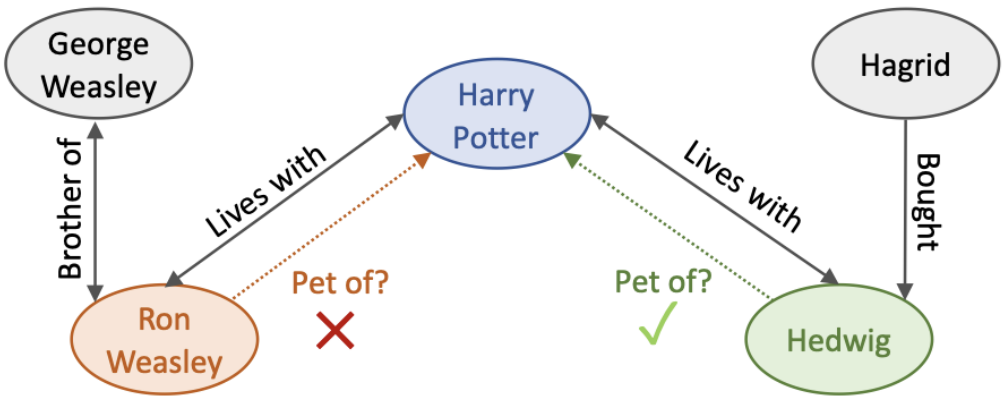


Figure 1.1: Link prediction in KGs, from [2].

1.3 Learning on Graphs : Graph Neural Networks

Graph Neural Networks (GNNs) represent a significant advancement in machine learning, specifically designed to interpret and process graph-structured data. Unlike traditional deep learning architectures that excel at handling sequential (e.g., text) or grid-like data (e.g., images), GNNs are tailored to extract insights from the inherent connectivity patterns within graphs. This flexibility enables GNNs to model complex relationships in a wide array of real-world domains where objects and their interactions can be naturally represented as graphs. GNNs operate by iteratively aggregating information from a node's neighborhood, allowing them to capture both local structural features and global dependencies within the graph.

1.4 Real World Applications of GNNs

The versatility and expressiveness of GNNs have led to their widespread adoption across a diverse spectrum of applications. Below, we explore key applications and highlight influential papers that advanced the use of GNNs in each domain:

Bioinformatics

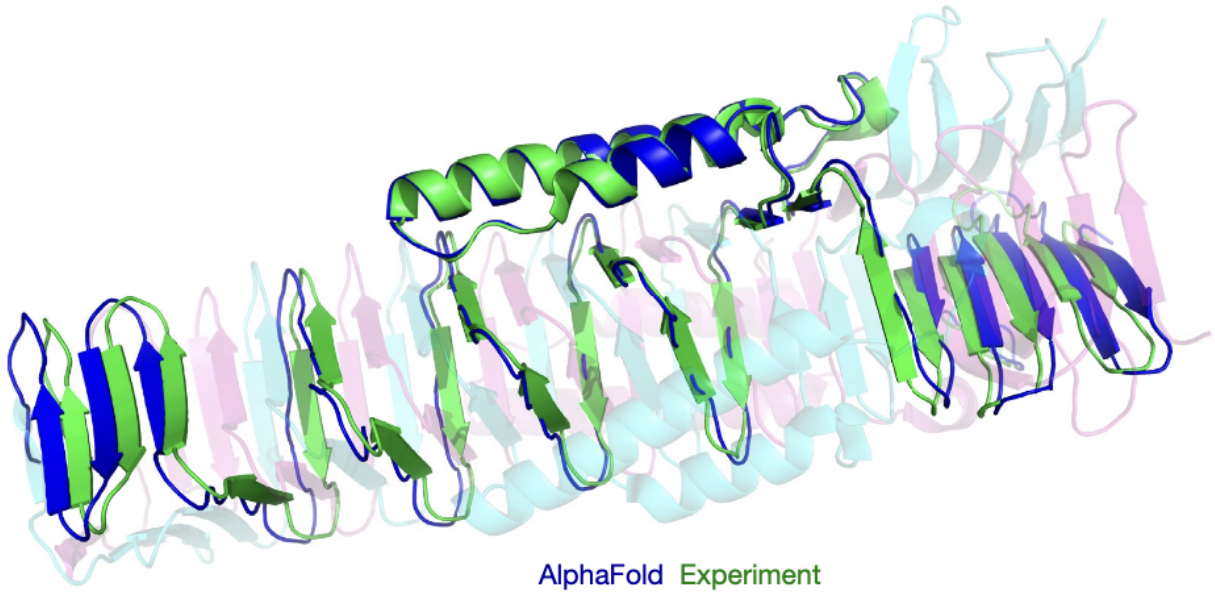


Figure 1.2: Correctly predicting a protein structure, PDB 6SKo, using AlphaFold (blue is predicted and green is experimental), from [3].

GNNs are revolutionizing bioinformatics by providing a powerful framework for modeling and analyzing complex biological systems. At their core, GNNs excel at learning from the intricate relationships within biological entities. A landmark achievement in this field is DeepMind's AlphaFold [3], which utilizes an attention-based GNN architecture to achieve unprecedented accuracy in protein structure prediction. The open-sourcing of protein structures predicted by AlphaFold, made available in the AlphaFold Protein Structure Database [4], further accelerated scientific discovery, fueling countless research efforts across the biological sciences. Beyond this, GNNs also advance drug discovery, predicting drug-target interactions [5] and even designing new molecules [6]. Furthermore, GNNs are being integrated with 3D geometric deep learning for applications that require a nuanced understanding of molecular geometries and properties, particularly in quantum chemistry and dynamic molecular simulations [7]. This integration enables the networks to effectively utilize spatial information, enhancing both the accuracy and interpretability of predictive models in complex chemical systems. Beyond molecular-level studies, GNNs are making strides in modeling cell-cell interactions within complex biological tissues and understanding gene regulatory networks. Innovative

approaches like GRGNN [8] exemplify this progress; the method effectively reconstructs gene regulatory networks by combining gene expression data with the power of graph neural networks. This provides a robust framework for exploring the intricate interactions that govern cellular behavior.

Social Networks

GNNs offer powerful tools for analyzing and understanding the dynamics of social networks. Their ability to model relationships and dependencies between users makes them ideal for tasks like community detection [9], identifying influential users, and viral marketing. For example, frameworks like the Community Embedding framework (ComE) [9] demonstrate how GNNs improve community detection by integrating it with node and community embeddings, leading to a more refined understanding of social groups. A key application area is influence maximization, where the goal is to select a small set of "seed" nodes that will maximize the spread of information or opinions through the network [10]. GNNs are proving invaluable in this domain, with works like ToupleGDD [11] demonstrating their ability to learn effective influence maximization strategies – a key focus of this thesis. ToupleGDD tackles the challenges of scalability and generalization in IM by using deep reinforcement learning and a unique GNN-based approach. Additionally, GNNs are also being used to address challenges in social recommender systems and to combat the spread of misinformation or harmful content within social networks.

Physics

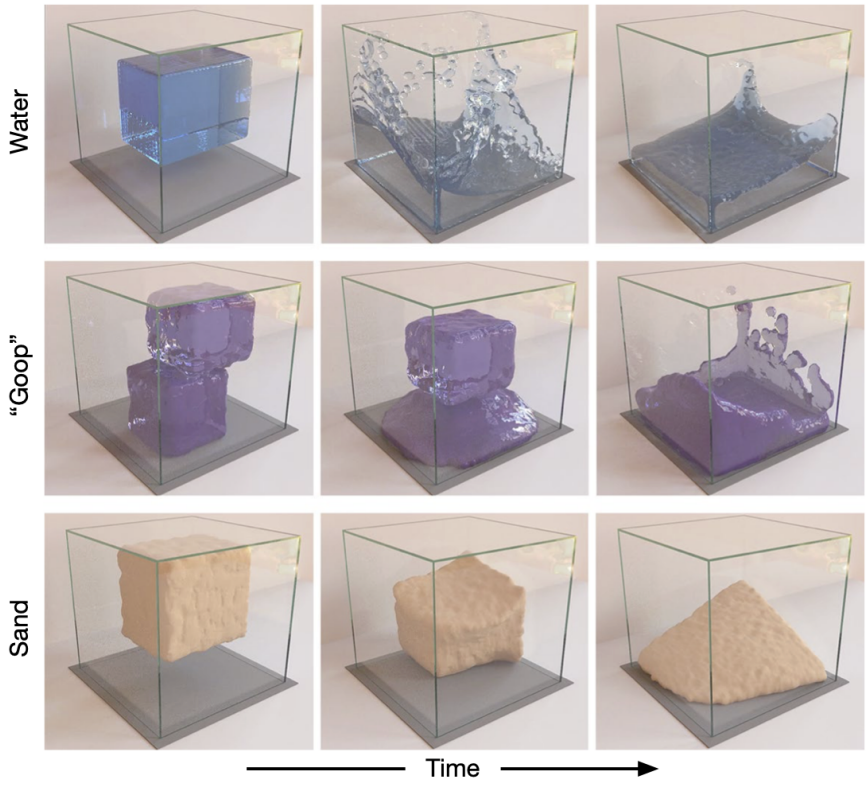


Figure 1.3: Rollouts of the GNS model from [12] where it learns to simulate rich materials at resolutions sufficient for high-quality rendering.

GNNs are transforming how we approach problems in physics. Their ability to learn and reason about complex interactions within physical systems makes them well-suited for tasks ranging from particle interaction simulations to modeling the dynamics of complex fluids. Building upon their foundational Graph Network framework [13], a team of researchers demonstrated the power of graph-based neural networks for learning complex physical interactions. This approach leads to highly accurate simulations [12]. More recently, GNNs have found success in high-energy physics for jet tagging and event reconstruction. Their ability to model the interconnected relationships between

particles makes them a powerful tool for analyzing the complex data generated in particle collider experiments [14]. Specifically, GNNs can improve the precision of reconstructing particle interactions within jets, including the identification of secondary vertices, which helps to distinguish the decays of different particle types [15]. Beyond particle physics, GNNs excel in quantum chemistry. They demonstrate their power by calculating atomic forces in large-scale simulations (e.g., ForceNet [16]) and their ability to model complex representations for predicting molecular properties (e.g., Symmetrical Graph Neural Network [17]). Furthermore, GNNs are enabling innovations in materials design and even the study of astrophysical phenomena.

Transportation Systems

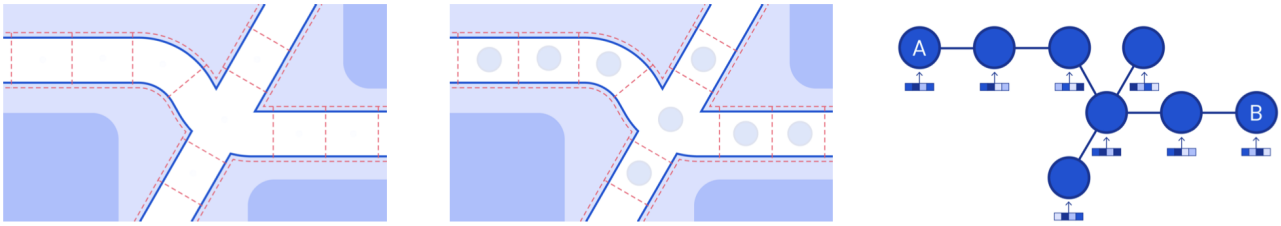


Figure 1.4: A road network (left) is transformed into a graph representation suitable for GNNs, from [18].

GNNs are revolutionizing the field of transportation systems by offering a data-driven approach to modeling complex transportation networks and solving related combinatorial optimization problems. Their capability to capture the inherent relationships between roads, intersections, and other entities within the network makes them ideal for tasks like traffic forecasting [19, 20]. For example, models like DCRNN [19] and T-GCN [20] leverage GNNs to analyze both spatial and temporal traffic patterns, leading to more accurate forecasts and better-informed decision-making in transportation management. Additionally, GNNs are used in route planning [21], optimizing routes for individuals or fleets based on complex, combinatorial travel choices. GNNs even find use in traffic light control, optimizing signal timings to maximize traffic flow [22]. A landmark achievement in this field is Google's deployment of a sophisticated GNN model to enhance ETA predictions within Google Maps. This work [18] has positively impacted millions of users and enterprises worldwide. The model's ability to learn complex patterns within traffic data has significantly reduced errors in ETA predictions (40+% in cities like Sydney), showcasing the power of GNNs in large-scale, real-world applications. Moreover, the field is expanding beyond traditional road networks; GNNs are being applied to multi-modal transportation systems, considering interactions between buses, trains, and even bike-sharing networks [23]. Furthermore, GNNs are being explored for real-time incident detection and routing adjustments during emergencies, potentially improving overall network resilience while addressing dynamic combinatorial problems.

Communication Networks

GNNs are finding widespread use in communication networks, where their ability to model relationships within the network topology offers advantages for various tasks. One key application is in resource allocation within wireless networks [24]. GNNs can learn the complex interactions between network elements, allowing them to optimize how resources like bandwidth and power are distributed among users. Specifically, methods like Aggregation Graph Neural Networks (Agg-GNNs) [24] facilitate decentralized decision-making, empowering individual devices to intelligently allocate resources based on local network conditions. This leads to improved network efficiency, adaptability to dynamic environments, and enhanced user experiences. Additionally, GNNs are proving invaluable for traffic routing [25]. By treating networks as graphs, GNNs can dynamically adapt routing strategies to real-time traffic conditions and changing topologies. This leads to more efficient network management and outperforms traditional routing methods. Furthermore, GNNs facilitate anomaly detection in communication networks by learning "normal" network behavior and identifying deviations that might signal security issues or technical failures.

Chip Design

GNNs are emerging as powerful tools within the integrated circuit design process. Their ability to model the complex interdependencies between components on a chip makes them suitable for various tasks. These chips, designed with the help of GNNs, are themselves central to AI development. One critical application is in physical design, specifically placement optimization [26]. Here, GNNs can be combined with reinforcement learning to automate and improve the placement of components on a chip. By representing the chip as a graph, GNN-powered models can learn to optimize placement decisions to minimize wirelength, congestion, and other critical design metrics [26]. This approach leads to faster chip design cycles, improved performance, and a reduction in the need for manual design steps. Additionally, GNNs are being explored for routing optimization, where they aid in finding efficient paths for the complex network of wires on a chip. Moreover, GNNs show promise in predicting high-level synthesis (HLS) performance metrics [27]. This allows designers to evaluate design choices early in the process, promoting rapid prototyping and reducing the need for time-consuming implementation cycles.

Computer Vision

GNNs are expanding the frontiers of computer vision by effectively representing and reasoning about the relationships inherent within images and videos. A key application area is object detection and scene understanding, where complex relationships between objects are essential. Specifically, methods like the Vision GNN (ViG) [28] offer a powerful solution, treating images as graphs rather than grids. This graph-based approach, using graph convolutions and feed-forward modules, outperforms CNNs and Transformers on image recognition and object detection – especially when handling irregular objects. Its flexibility makes it promising for real-world scenarios with complex scenes and non-uniform objects. Moreover, GNNs are demonstrating value in 3D computer vision by directly processing point clouds [29]. Techniques like Dynamic Graph CNN's EdgeConv operation capture local geometric features within point clouds, enabling superior object classification and segmentation compared to prior methods. Additionally, recent works explore the use of GNNs in video analysis, where they capture the temporal relationships between video frames for improved understanding of actions and events.

Knowledge Graph Completion

GNNs have become invaluable tools for knowledge graph completion (KGC), the task of predicting missing links or inferring new facts within a knowledge graph. GNNs learn rich representations of entities and their relationships within the graph's structure. Early works [30] demonstrated the potential of GNNs to handle complex relational knowledge. Recent advancements have significantly pushed the boundaries of GNNs in this area. One such innovation is the INDIGO approach [31], which directly encodes the knowledge graph into a GNN. This encoding eliminates the need for external scoring functions, leading to more efficient and generalizable link prediction in inductive settings. Similarly, the MA-GNN model [32] advances the field by employing multiple attention mechanisms to capture both global and local structural information, improving the model's ability to handle long-range dependencies for more complex reasoning. Another novel approach is the PathCon method [2], which adopts a new GNN approach for KGC. It leverages "relational context" via message passing to aggregate neighborhood information based on relation types, and further incorporates "relational paths" by modeling and assessing the importance of various paths connecting entity pairs for relation prediction. Finally, other works have focused on integrating logical rules with GNNs [33], enhancing interpretability and robustness of predictions.

1.5 Contributions

This thesis investigates the application of GNNs to enhance predictive capabilities in two distinct domains. First, we demonstrate how GNNs can predict the dynamics of influence spread in social

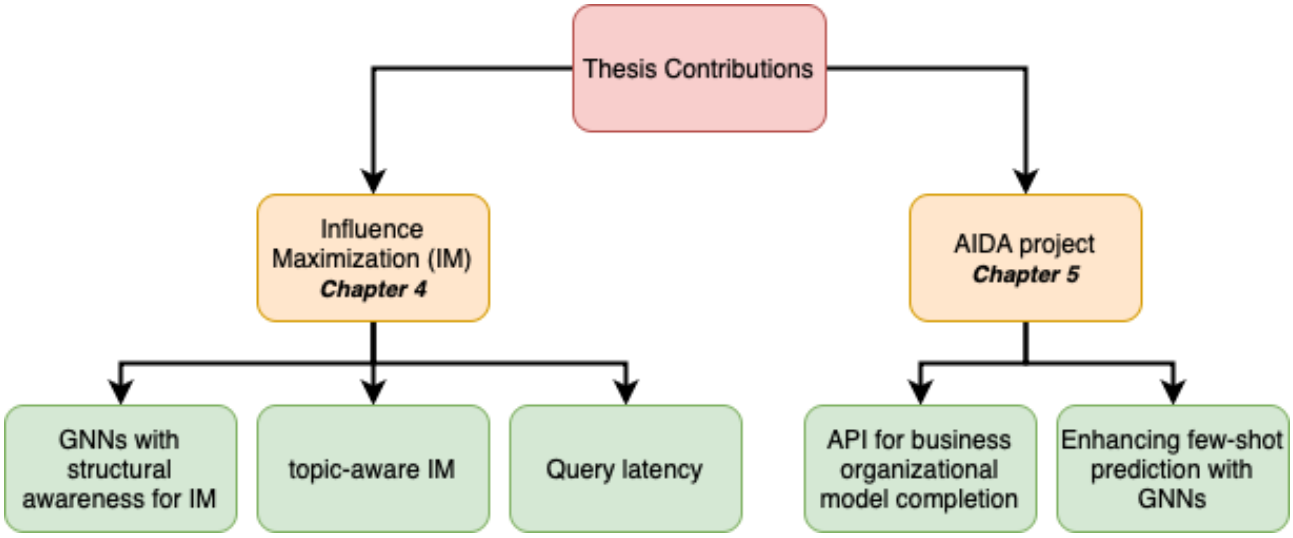


Figure 1.5: Thesis contributions: a visual breakdown by chapter.

networks, developing robust models that effectively leverage complex relationships within graph-structured social network data. Second, motivated by the AIDA project, we explore the use of GNNs in KG-based business organizational model recommendations. Our initial prototype utilized a non-GNN based few-shot link prediction model, demonstrating the feasibility of this approach, subsequently leading us to investigate how GNNs could further enhance the model's performance. This thesis makes several key contributions, outlined below and further illustrated in Figure 1.5.

1.5.1 Influence Maximization

This thesis advances the field of Influence Maximization (IM) in social networks. IM focuses on identifying the most influential nodes within a network to maximize the spread of information or ideas. Our work contributes to this field in several key ways. Firstly, we begin with an IM adaptation of S2V-DQN (S2V-DQN-IM) as a proof-of-concept. Secondly, we introduce IM-GNN, which combines attentive Graph Neural Networks (GATs), the graph magnetic Laplacian as a positional encoding and self-edges. Importantly, for our comparison of IM-GNN with SOTA baseline methods, we use public, real-world datasets and diffusion graphs built from cascades, i.e., with data-based diffusion probabilities. Thirdly, we expand our framework with TIM-GNN to consider the topical aspects of real-world influence spread. For a realistic evaluation, from public data, we extract topic-aware diffusion graphs from information cascades, using the survival factorization framework [34]. Additionally, we present TIM-GNN^x, which incorporates cross-attention mechanisms to address real-world IM query requirements. These contributions offer novel, scalable, and efficient solutions based on Deep Reinforcement Learning and Graph Neural Networks to tackle the complex challenge of Influence Maximization, pushing the boundaries of this research area.

1.5.2 Few-shot Link Prediction in Knowledge Graph

AIDA is a collaborative public-private project in the field of artificial intelligence (AI). It is a joint effort by IBM, Université Paris-Saclay, Softeam, DecisionBrain and STET, with the ambition to develop a learning platform that will allow companies to improve their performance by integrating artificial intelligence into their operational systems. The project is expected to benefit from the latest research advancements from Université Paris-Saclay.

Within the AIDA project, we develop a prototype, deployed as an API, to deliver KG-based recommendations for business organizational models. For that purpose, we adapt the MetaR model [35], a non-GNN based model, for our use case. We then delve into the challenges associated with few-shot link prediction in knowledge graphs and propose several innovative solutions to enhance this model with GNNs. We first adapt the PathCon model to the MetaR framework for few-shot link prediction. PathCon, known for its ability to capture graph structure, leverages relational context and paths for enhanced reasoning. Our adaptation integrates PathCon's GNN message passing and

path modeling into MetaR's Meta-Learning framework for few-shot link prediction, resulting in comparable performance to the original MetaR model. Building upon this, we introduce experimental hybrid models that explore different ways to combine PathCon with the MetaR model, seeking potential performance gains. We investigate various strategies for this integration. These strategies include an adaptive attention mechanism to dynamically balance the contributions of each model based on the context, closer alignment of MetaR with PathCon's relational reasoning, and a Mixture of Experts model [36] where each method is specialized and deployed selectively based on the prediction context. While the hybrid models did not consistently outperform the individual models, our investigation revealed potential reasons for this outcome. The inductive, structure-focused nature of PathCon might not be fully compatible with MetaR's task-adaptive, gradient-based meta-learning approach. Additionally, the limitations of the TransE scoring function, particularly in representing complex relational patterns, could have hampered the models' synergy. Finally, PathCon's reliance on diverse relational paths might have been hindered by the limited support set examples in the few-shot setting. These findings suggest that future research should explore either more compatible learning paradigms, more expressive scoring functions, or more data-efficient ways to utilize path information.

1.6 Thesis Structure

This thesis explores the intersection of graph neural networks, reinforcement learning, and meta-learning to solve key problems within graphs, specifically influence maximization in social graphs and few-shot link prediction in knowledge graphs. The structure is as follows:

- Chapter 2 provides a comprehensive survey of existing research in combinatorial optimization, influence maximization, GNNs, reinforcement learning, few-shot link prediction, and meta-learning in knowledge graphs.
- Chapter 3 introduces core notations, formal definitions, and in-depth explanations of the foundational techniques employed in this work, including GNNs, reinforcement learning, meta-learning.
- Chapter 4 advances the field of Influence Maximization in social networks. This chapter introduces novel deep reinforcement learning and graph neural network-based approaches that outperform existing methods in performance, latency, and robustness. Key contributions include the integration of graph attention mechanisms, positional encodings and self-edges to an existing deep learning framework, the modeling of topic-aware influence spread, and the optimization of query latency for practical IM applications.
- Chapter 5 discusses the development of a prototype for deploying KG-based business organizational model recommendations as an API. It also explores challenges in few-shot link prediction for knowledge graphs and proposes several ideas. The PathCon method is adapted to a Meta-Learning framework to enhance its few-shot capabilities while effectively leveraging KG structure. Additionally, a hybrid model combining the adapted PathCon with the MetaR algorithm is developed. While experiments did not reveal a conclusive synergy between these models, the chapter analyzes potential reasons for this outcome.
- Chapter 6 summarizes the thesis's key discoveries and the significance of its contributions and suggests promising avenues for extending this research or addressing its limitations.

Chapter 2

Literature Review

2.1 Combinatorial Optimization

Combinatorial optimization is a cornerstone of applied mathematics and computer science, concerned with finding optimal solutions from a finite set of possibilities subject to constraints. The field's rich history includes seminal works establishing the theoretical underpinnings. The Traveling Salesman Problem, formulated by Dantzig, Fulkerson, and Johnson [37], exemplifies the challenges of combinatorial optimization with its exponentially growing solution space. In parallel, George Dantzig's development of the Simplex algorithm [38] revolutionized the field of linear programming, providing a foundation upon which later techniques could build. For complex combinatorial problems, branch-and-bound algorithms [39] offer systematic search procedures. However, the inherent computational difficulty often necessitates heuristics and metaheuristics such as simulated annealing [40], evolutionary algorithms [41], and tabu search [42] to find good solutions in reasonable time.

The rise of machine learning, particularly Graph Neural Networks, has transformed combinatorial optimization. GNNs excel at learning complex patterns from graph-structured data, making them naturally suited to represent problems with entities and relationships [43]. Recent work has highlighted the power of attention-based GNNs for finding good solutions to the Traveling Salesperson Problem, getting close to optimal results for problems up to 100 nodes [44]. Moreover, using reinforcement learning to train GNNs for combinatorial optimization has shown significant promise [45].

A recent survey [46] offers a panoramic view of the field, outlining how GNNs present a compelling approach for addressing challenging optimization problems. The survey highlights the capabilities of GNNs that make them particularly well-suited for CO tasks. Specifically, the permutation invariance of GNNs aligns with the need to find solutions independent of input order. Moreover, GNNs' strength in relational reasoning allows them to uncover complex patterns within the problem structure. This survey details how GNNs can be applied directly as learned solvers to find solutions or can be integrated with traditional solvers to enhance efficiency and guide the search process. Crucially, this comprehensive survey offers a unified perspective of the emerging field, creating a taxonomy of GNN-CO methods while highlighting promising research directions and remaining challenges.

2.2 Influence Maximization in Social Graphs

Influence maximization. The problem was first formalized in [10], and shown to be NP-hard, for diffusion models such as Linear Threshold (LT) and Independant Cascades (IC). Exploiting the monotonicity and submodularity of the influence spread function, they show that a simple greedy strategy is within $(1 - 1/e)$ of the optimal. Since the diffusion models are stochastic, expensive Monte Carlo simulations are needed to estimate the spread (marginal gain) of a candidate seed. Regarding the traditional (vanilla) IM formulation, the majority of the studies that followed focused on improving upon efficiency. For instance, the CELF method of [47] further exploits submodularity, leading to far fewer Monte Carlo diffusion simulations in practice. The IMM algorithm of [48] introduces a set of estimation techniques based on martingales, which enable to maintain the worst-case guarantees of the greedy approach, while increasing the efficiency (near-linear time). IMM builds upon the reverse influence sampling (RIS) approach from [49]. Instead of simulating influence spread forward from

potential seed nodes, RIS traces influence paths backward from randomly selected nodes to identify influential sources. IMM uses this concept to generate reverse reachable (RR) sets and then selects the seed set that overlaps the most with these sets. While these and numerous other good heuristics have been proposed in recent years, their applicability in real-world IM settings at scale remains problematic, due to the intrinsic hardness of the combinatorial problem combined with the stochasticity of diffusion models. For a more in-depth comparison of traditional IM methods, we refer the reader to the recent studies of [50] and [51].

Topic-aware IM (TIM). Vanilla IM models do not capture variations in diffusion patterns, essen-

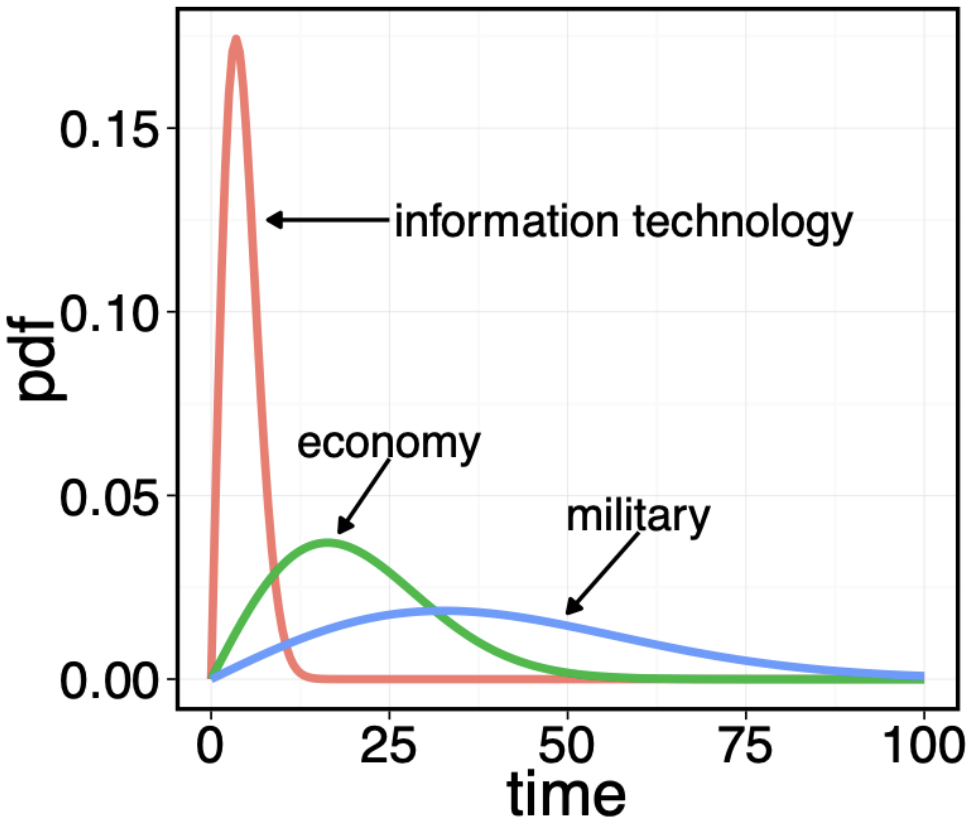


Figure 2.1: Posts of different topical features have a different spreading rate, reflected in the different shapes of the fitted Rayleigh distribution of transmission time, from [52]

tially treating diffusion items as a black box (ignoring content), yet topical features have been shown to play a major role in how information may spread virally [52]. Extending the vanilla IC model, Topic-aware Independent Cascades (TIC) diffusion models have been proposed initially in [53] and further refined in [34]. In short, TIC assumes that each edge is labelled by a *probability diffusion vector* of dimension d (number of topics), with each component denoting the activation probability from the source node to the target one under a specific topic. While capturing more closely real-world diffusions than vanilla IM models, TIC has the downside of bringing an additional level of complexity to an already challenging problem. Algorithms for online topic-aware IM have been proposed, based on TIC, following generally a strategy of pre-processing and indexing topic-agnostic IM results, such as the INFLEX approach of [54], as well as [55] or [56]. We refer the reader to [57] for a survey on IM in contexts involving other relevant dimensions of social networks, such as topics, location, or time. We contribute to this line of research by proposing a DRL-based approach, predicting marginal spread gain under TIC, based jointly on the diffusion medium and the topical profile of the item to spread.

S2V-DQN. Designing good heuristics for combinatorial optimization problems on graphs often requires tremendous specialized knowledge and trial-and-error. [45] presents a new perspective on combinatorial optimization problems in general. They use a powerful end-to-end trainable approach called S2V-DQN, merging the strengths of Deep Q-Network (DQN) in reinforcement learning with Structure2Vec's (S2V) capability in graph embedding. It acts as a meta-algorithm, capable of automating the design of heuristics and incrementally constructing a solution for hard combinatorial

optimization problems on graphs. *In our work, we adopt and adapt this framework to the problem of topic-aware IM.*

Learning-based methods for IM. We have seen in recent years an inflation of learning-based methods for IM, alas, with many of them suffering in our view from similar conceptual, methodological, or scientific rigour shortcomings. A recent survey of such approaches can be found in [58]. We focus our discussion on the deep reinforcement learning line of research that builds upon the meta-algorithm S2V-DQN, adapting it to the IM problem. PIANO is a DQN-based IM algorithm that was proposed by [59], and the first study to use the S2V-DQN framework for IM. PIANO trains the DQN model on small, topologically coherent samples of the diffusion graph before evaluating it on the full graph. With a focus on generalizability, PIANO also proposes to maintain a collection of pre-computed models to avoid retraining. The authors claim superior efficiency and comparable result quality with state-of-the-art classical IM methods. However, we found serious issues while trying to reproduce the paper's results on benchmark IM datasets. Our basic algorithmic design (S2V-DQN-IM) for vanilla IM is quite similar to PIANO conceptually, which allows us to follow the performance gains brought to it by our progressively refined methods IM-GNN and TIM-GNN. DIEM [60] is the only study to apply the S2V-DQN framework to topic-aware IM. However, it advances only marginally the general understanding of the problem space and potential solutions. This is due to experimental limitations and unspecified details. For example, DIEM is evaluated on undisclosed Twitter data, under an ad-hoc topic-aware model, with artificial topology-based diffusion probabilities. We could not obtain the DIEM code or data from the authors, hence the results are not reproducible. Overall, PIANO and DIEM have similar limitations. They both evaluate on artificial diffusion graphs, which limits their relevance for practical IM scenarios. Instead, we focus on cascade-based diffusion graphs, built from public datasets of cascades. Regarding how these studies extend and improve upon the S2V-DQN framework, many unclear aspects remain. In fact, PIANO does not discuss this aspect at all (incidentally, does not cite [45]), although it adopts its conceptual framework.

ToupleGDD [11] is an S2V-DQN-like framework for vanilla IM that couples three GNNs enhanced with an attention mechanism for network embedding. It has a particular focus on generalizability, and therefore trains on small, randomly generated graphs and with a small seed set budget. This allows it to be evaluated on a multitude of diffusion graphs and for various budgets. GCOMB is an approach proposed in [61], which, unlike S2V-DQN, does not use an end-to-end architecture. It start from the observation that pruning the search space is as important as the prediction of the solution. GCOMB works by first using a graph convolutional network (GCN) to identify promising nodes (as seed set candidates); the GCN learns embeddings of the promising nodes in a supervised manner. Then, it uses a Q-learning component to predict the solution set from the promising nodes. The mixture of supervised and RL makes GCOMB on one hand lightweight (fewer parameters), but on the other hand highly dependent on the quality of the supervised stage. The DeepIM approach of [62] tackles vanilla IM on a fixed graph using an autoencoder to compress seed sets, training the network to optimize the spread prediction from the reconstructed seed set. The optimal seed set is then inferred through gradient descent in the low-dimension space. By design, one particular focus of DeepIM is generalizability under various diffusion models, which may impact its predictive performance in certain settings. While alternative diffusion models are beyond our scope, we see this additional level of robustness – across diffusion models – as an interesting direction for future research. *ToupleGDD, GComb, and DeepIM are the most recent and, in our view, the best representatives of the state-of-the-art on learning-based IM; we will employ them as baseline methods.*

Other related work. [63] proposes RL4IM, an RL approach for contingency-aware IM, where the seeded nodes accept their role with a certain probability. In particular, they focus on *reward shaping* since, due to uncertainty on the seeds' willingness to spread information, it is no longer possible to use the marginal spread of a newly selected node as the immediate reward at each step. When contingency is relaxed, RL4IM becomes the basic S2V-DQN we compare with. The IMINFECTOR approach of [64] solves vanilla IM by extracting influencer and susceptible node embeddings from cascades. The rationale is that such deep-learning based representations can capture high-order correlations between users. They rely on such embeddings to make predictions on influencer-influencee spread probability and influencers' spread magnitude, used as the main ingredients in an adapted greedy algorithm for seed selection. The DeepIS approach of [65] predicts the *susceptibility* of target nodes to be influenced by a seed set. They approach the problem as a node regression task in the GNN framework.

Positioning w.r.t state-of-the-art. While there is an extensive and diverse work on new DRL methods for graph CO problems (e.g., for vanilla IM), in particular building upon the simple, generic S2V-DQN framework, our work represents the first attempt to leverage DRL for topic-aware IM. We argue that a DRL-based approach within this framework is only justifiable if there is a large space of potential IM queries (hence topic-aware), which can be answered in real-time. The aforementioned DRL-based solutions for vanilla IM cannot be directly applied nor easily adapted to address the challenges of topic-aware IM. In essence, TIM-GNN^x is the first approach addressing the three critical challenges for practical applicability of learning-based IM solutions: predictive *accuracy* over a broad (topic-aware) query space, *low latency* over large diffusion graphs, and *robustness* to graph changes. As illustrated in our experiments, existing methods fall short in at least one of these aspects. Importantly, in these experiments we incorporate realistic diffusion graphs, either topic-agnostic or topic-aware, in contrast with the commonly used uniform or trivalency probability models of most prior works (including all DRL-based methods for vanilla IM).

2.3 Graph Neural Networks and Message Passing

GNNs have emerged as a powerful deep learning paradigm for extracting insights from graph-structured data. Unlike traditional machine learning techniques originally designed for tabular, sequential, or image data, GNNs are specifically tailored to handle the complexities of interconnected, non-Euclidean data, making them uniquely powerful in domains where relationships between entities are important. The core mechanism behind GNNs is the concept of message passing, which enables efficient information sharing and aggregation within a graph's intricate topology.

2.3.1 Foundational Works and Key Developments

The development of GNNs demonstrates a clear drive toward scalable, expressive deep learning architectures for graph-structured data. Early work, pioneered by [66], modeled graph dependencies using a recurrent neural network-based approach. This design allowed node representations to evolve iteratively based on the states of their neighbors, capturing the inherent dependencies within graphs. Building on this capability, [67] sought to adapt the power of convolutional neural networks (CNNs) to the graph domain. Their spectral approach, leveraging the graph Laplacian and Fourier transform, provided a mathematical basis for graph convolutions. However, the global nature of the spectral filters created scalability challenges. Recognizing these limitations, subsequent research shifted towards spatial methods that directly operate on graph neighborhoods. [68] formalized this trend. By defining graph operations in terms of message passing between nodes, this approach offered a flexible and scalable foundation for graph representation learning. [69] improved the spatial approach with Graph Convolutional Networks (GCNs). By approximating spectral convolutions with a localized first-order approach, GCNs improved scalability without sacrificing the ability to capture neighborhood information. Further refinement came with GATs [43]. Integrating attention mechanisms to dynamically weigh the importance of a node's neighbors significantly increased GNNs' expressive power for complex relational patterns. Addressing large and dynamic graphs, [70] introduced GraphSAGE, an inductive learning paradigm. GraphSAGE generates node embeddings by sampling and aggregating features from local neighborhoods, making it suitable for graphs where the full structure might be unknown or constantly evolving.

A crucial aspect of GNN research lies in understanding their theoretical limitations and expressivity. The Weisfeiler-Lehman (WL) graph isomorphism test provides a fundamental benchmark for measuring the ability of a GNN to distinguish between non-isomorphic graphs. Research has shown that standard message passing GNNs are at most as powerful as the WL test [71]. More recent work seeks to enhance the expressive power of GNNs, potentially enabling them to surpass the limitations imposed by the WL test [72].

Pushing the boundaries of GNNs, recent works explore mechanisms for enhanced collaboration between nodes. One approach to improving GNNs efficiency involves reducing the number of layers required to learn features from distant neighborhoods. [73] introduces learnable structural and positional representations, allowing nodes to aggregate information from throughout the graph with fewer message-passing steps. Another fascinating development is Cooperative Graph Neural Net-

works [74], where nodes collaborate strategically in their communication patterns, demonstrating improved performance. Unlike traditional GNNs where communication follows the fixed graph structure, nodes in this framework learn to actively choose whether to send messages, receive messages, both or none. This strategic communication allows for targeted information flow, potentially reducing computational bottlenecks while increasing the expressive power of the model.

2.4 Reinforcement Learning in Graph Analysis

The synergy of RL and graph analysis gives rise to a fertile research field with the potential to tackle intricate graph-based tasks with increased automation and adaptability.

2.4.1 Reinforcement Learning

Reinforcement learning (RL) is a powerful machine learning paradigm where an agent learns through interaction with its environment. Unlike supervised learning, the agent isn't given explicit correct answers, but instead, receives rewards (or punishments) for its actions. The goal is to learn a strategy, called a policy, that maximizes the cumulative reward over time. RL draws inspiration from how humans and animals learn – through trial and error, discovering which actions yield the best outcomes in different scenarios.

Early RL research focused on tabular methods such as Q-learning [75], where agents learn values for each state-action pair. These methods, while effective in simple environments, struggle with the curse of dimensionality as the state and action spaces grow. To address this, function approximation methods, utilizing neural networks, have become increasingly popular. Deep Q-Networks [76] were a breakthrough in this area, demonstrating the ability to learn complex policies directly from sensory inputs i.e raw pixel data.

However, DQN initially faced challenges with stability and convergence. Subsequent research addressed these issues through various enhancements. Double DQN [77] mitigated the overestimation bias of Q-value present in DQN by separating the process of choosing an action from the process of evaluating its value. Dueling DQN [78] further improved performance by separating the Q-value output into state value (how valuable is being in a particular state) and action advantage (how much better a specific action is compared to the others in that state). Prioritized Experience Replay [79] tackled the issue of correlated data and unstable learning by storing past experiences and using them for training updates while introducing a mechanism to prioritize the replay of experiences that contribute most to learning, leading to faster convergence. These advancements, along with others, have solidified DQN's position as a foundational algorithm in deep RL and paved the way for further exploration in the field. In this thesis, we mostly use *Prioritized Double DQN*.

Despite their success, DQN and similar value-based methods can suffer from instability and slow convergence. Policy gradient methods [80] offer an alternative, directly optimizing the policy parameters to maximize expected return. Recent advancements such as the Proximal Policy Optimization (PPO) algorithm [81] have improved the stability and efficiency of policy gradient methods, making them a popular choice for continuous control tasks.

2.4.2 RL and GNNs

GNNs excel in learning expressive representations of nodes, edges, and whole graphs. The integration of GNNs and RL creates a powerful framework where the GNN can inform the RL agent's state representation, and the RL agent optimizes decision-making on the graph structure. This combination proves effective in applications like combinatorial optimization on graphs [45], recommender systems, and molecule generation.

2.5 Few-Shot Link Prediction in Knowledge Graphs

Knowledge Graphs have become essential tools for representing and reasoning about real-world information, but a core challenge lies in completing these graphs by predicting missing links between

entities. This task, known as link prediction, has traditionally relied on methods, like TransE [82] and ComplEx [83], requiring extensive training data, leading to limitations when encountering the "long tail" phenomenon – the abundance of entities and relations with sparse observations. As a response, the field of few-shot link prediction has gained prominence, aiming to make accurate predictions using only a handful of training examples.

Few-shot link prediction approaches draw inspiration from meta-learning, leveraging its principles of learning to learn and rapid adaptation to new tasks. Meta-learning based models like MetaR [35] and GMatching [84] respectively utilize gradient-based and metric-based meta-learning to acquire generalizable strategies for adapting to new tasks with limited data. MetaR proposes a meta relational learning framework that aims to transfer relation-specific meta information from a support set to a query set. It leverages relation meta to transfer common and important information, and gradient meta to accelerate learning. GMatching, on the other hand, learns a matching metric by considering both the learned embeddings and one-hop graph structures.

Beyond these established techniques, recent research has introduced novel approaches that further advance the field. PathCon [2] employs relational message passing, focusing on edge features (relation types) and iteratively passing messages among edges to aggregate neighborhood information. This allows the model to capture both the relational context and paths within the knowledge graph, leading to strong performance and interpretable results, particularly in sparse KGs. Connection Subgraph Reasoner (CSR) [85] addresses the limitations of meta-learning by directly predicting the target few-shot task without relying on manually curated sets of training tasks. By modeling a shared *connection subgraph* between support and query triplets, CSR implements a principle called *eliminative induction*. This principle, rooted in scientific reasoning, involves proposing multiple hypotheses and then systematically eliminating those that contradict observed data. In the context of CSR, the algorithm explores various connection subgraphs within the knowledge graph, treating them as potential hypotheses for explaining the relations in the support set. Subgraphs inconsistent with any of the support triplets are discarded, leading to a refined hypothesis represented by the shared connection subgraph. This effectively identifies the underlying logical pattern that implies the existence of the target triplet. This approach offers a more generalizable and adaptable solution to few-shot learning.

Few-shot link prediction methods, and knowledge graph completion techniques in general, are rapidly improving. Researchers are exploring new architectures, pre-training strategies, and inductive reasoning mechanisms to build stronger models. Future directions may include incorporating external knowledge sources to enrich the models, developing hybrid approaches that combine different learning paradigms, and designing data-efficient models for knowledge graph reasoning.

Chapter 3

Preliminaries

3.1 Notations

Symbol	Definition
G	A graph
V	The set of nodes (vertices) in a graph
E	The set of edges in a graph
P	Probability function on the set of edges
(u, v)	An edge between nodes u and v
$p_{u,v}$	Topic-agnostic diffusion probability along edge (u, v)
$\mathbf{p}_{u,v}$	Topic-aware diffusion probability vector for edge (u, v) in TIC (dimension d, one element per topic)
$p^z_{u,v}$	Diffusion probability along edge (u, v) for topic z in TIC
S	A seed set of nodes
k	Seed set budget
$\sigma(S)$	Spread of influence from seed set S
d	Number of topics
$\overrightarrow{\gamma}$	Item: a topic distribution vector
$Q = (\overrightarrow{\gamma}, k)$	IM query: topic distribution and budget
$\sigma(S\|\overrightarrow{\gamma})$	Spread of influence from seed set S for item $\overrightarrow{\gamma}$ in TIC
M	Predictive model for marginal spread gain
$M(\overrightarrow{\gamma}, S_0, s)$	Predicted marginal spread gain by model M for item $\overrightarrow{\gamma}$, partial solution S_0, and candidate node s

3.2 Graph Neural Networks

Graph Neural Networks have emerged as a powerful tool for analyzing and extracting insights from graph-structured data. To effectively utilize GNNs, it is crucial to grasp the fundamental concepts that underpin their design and functionality.

3.2.1 Key aspects of GNNs

Some core principles in understanding GNNs are permutation invariance and message passing.

Permutation invariance This principle dictates that the output of a GNN should remain unaffected by the order in which nodes are presented. This property is essential for GNNs because graphs, unlike sequences or grids, do not have a fixed or inherent ordering of nodes. The lack of a canonical node order necessitates that GNNs operate in a way that is insensitive to the specific arrangement of

Symbol	Definition
x_v	Binary indicator for node v being in partial solution S_0
$\mathcal{N}(v)$	Set of neighboring nodes of node v in the directed graph
$\hat{Q}$	Action-value function parameterized by GNN
relu(z)	Rectified Linear Unit activation function: relu$(z) = \max(0, z)$
t	Time step or iteration
$e^t_{u,v}$	Attention score between nodes u and v at iteration t
$\alpha^t_{u,v}$	Normalized attention weight between nodes u and v at iteration t
L_N	Normalized combinatorial Laplacian matrix
$L(q)_N$	Normalized magnetic Laplacian matrix with potential q
$\tilde{A}$	Symmetrized adjacency matrix
$\tilde{D}$	Degree matrix of the symmetrized adjacency matrix
b	Number of base items
T	Number of message passing iterations
α	Learning rate
γ	Discount factor in reinforcement learning
T_i	Task i in meta-learning
D_i	Dataset for task i in meta-learning
L_{T_i}	Loss function for task T_i
f_{θ_i}	Model after being updated for task T_i with parameters θ_i
θ	Initial model parameters to be optimized

nodes. This leads to the use of aggregation functions, such as summation or averaging, that combine information from neighboring nodes without being influenced by the order in which those neighbors are considered.

Message passing GNNs operate through an iterative process where nodes exchange information, or "messages" with their neighbors to update their representations. This message passing framework is the core mechanism by which GNNs learn and represent information within a graph. The process involves several key phases: First, each node gathers information from its neighbors in a process known as neighborhood aggregation. These collected messages are then transformed, often using neural networks, to extract meaningful features and patterns. Finally, the aggregated and transformed information is used to update the representation of the node itself. By iteratively exchanging and refining information, nodes in a GNN gradually develop a comprehensive understanding of their local environment and their position within the broader graph structure.

3.2.2 Three flavours of GNN layers

GNNs can be categorized into different "flavours" [86] based on the specific mechanisms they employ for aggregating and transforming information.

Convolutional GNNs This flavour employs a fixed weighting scheme to aggregate features from neighboring nodes.

$$h_u = \phi\left(x_u, \bigoplus_{v \in N_u} c_{uv}\psi(x_v)\right)$$

where h_u is the updated feature vector for node u, x_u is the original feature vector for node u, N_u represents the neighborhood of node u, c_{uv} is a fixed weight signifying the importance of node v to node u. Most commonly, ψ and ϕ are learnable affine transformations with activation functions; e.g. $\psi(x) = \mathbf{W}x + b$; $\phi(x, z) = \sigma(\mathbf{W}x + \mathbf{U}z + b)$ where $\mathbf{W}$, $\mathbf{U}$, b are learnable parameters and σ is an activation function such as the rectified linear unit. The additional input of x_u to ϕ represents an optional *skip-connection*, which is often very useful. Finally, $\bigoplus$ is a permutation-invariant aggregation function

(e.g., sum, mean, or maximum).

This flavour is similar to convolutional operations in CNNs, where fixed filters aggregate information from local regions.

Attentional GNNs This flavour utilizes a learnable self-attention mechanism to determine the importance of neighboring nodes dynamically.

$$h_u = \phi\left(x_u, \bigoplus_{v \in N_u} a(x_u, x_v)\psi(x_v)\right)$$

where $a(x_u, x_v)$ represents the attention coefficient between nodes u and v, computed by a learnable function a
This flavour offers more flexibility than the convolutional approach, allowing the network to focus on relevant neighbors based on their features.

Message-passing GNNs This flavour computes and exchanges arbitrary vector-valued messages between nodes.

$$h_u = \phi\left(x_u, \bigoplus_{v \in N_u} \psi(x_u, x_v)\right)$$

where $\psi(x_u, x_v)$ represents the message sent from node v to node u, computed by a learnable function ψ
This flavour is the most expressive but also computationally demanding, as it requires computing and storing vector messages for each edge in the graph.

There is a representational containment between these flavours, with convolutional being the least expressive and message-passing being the most expressive. The choice of GNN flavour depends on the specific task and the desired trade-off between expressivity and computational efficiency.

3.2.3 Node representation learning in graphs

Several other important ideas contribute to the effectiveness and versatility of GNNs.

Graph sampling When dealing with large-scale graphs, GNNs face computational challenges due to the vast number of nodes and edges involved in message passing. Graph sampling techniques offer a solution by strategically selecting a smaller, representative subset of nodes and edges for processing. This approach reduces computational costs, improves efficiency, and minimizes memory usage while aiming to preserve the essential structural information of the full graph. Various sampling methods exist, each with its own strengths and weaknesses. Node sampling techniques focus on selecting a subset of nodes and their corresponding edges, with options ranging from simple random selection to more sophisticated approaches based on node features or centrality measures. For instance, Breadth-First Search (BFS) sampling [87] starts from a randomly chosen node and systematically explores its neighboring nodes, then their neighbors, and so on, creating a subgraph that captures the local structure around the starting point. Alternatively, edge sampling techniques concentrate on selecting a subset of edges and their incident nodes, often using random selection or walk-based methods that traverse the graph along specific paths.

Node embedding techniques play a critical role in GNNs by learning to represent each node as a low-dimensional vector that captures its structural role and relationships within the graph. These dense vector representations encode valuable information about the graph's structure, enabling downstream tasks like node classification, link prediction, and graph classification. Several methods exist for generating node embeddings, each with its own approach to capturing structural information. Random walk [88] based embeddings, such as DeepWalk [89] and Node2Vec [90], simulate random

walks on the graph and learn representations based on the co-occurrence patterns of nodes within these walks. Matrix factorization based embeddings, like Laplacian Eigenmaps [91] and GraRep [92], leverage matrix factorization of graph-related matrices to obtain node embeddings that reflect the graph's underlying structure. Deep learning based embeddings, often generated by GNNs themselves, propagate information through the graph to learn representations that consider both local neighborhoods and broader connectivity patterns. Importantly, GNNs can be initialized using embeddings from other embedding techniques. This leverages pre-existing structural knowledge, potentially leading to faster convergence, more robust representations, and a reduced likelihood of overfitting.

3.2.4 Learning tasks over graphs

Learning tasks over graphs span multiple levels of complexity, encompassing node-level predictions, the analysis of relationships between nodes, and inferences about the entire graph structure.

Node-Level Tasks Node-level tasks focus on predicting properties or labels associated with individual nodes in the graph. Examples include predicting *the class label of a node*, such as identifying spam accounts in a social network or classifying the topic of a research paper in a citation network. Additionally, node-level tasks can involve predicting *a continuous value associated with a node*, like estimating the creditworthiness of a borrower in a financial transaction network.

Edge-Level Tasks Edge-level tasks deal with predicting properties or labels associated with the connections between nodes. This can involve predicting the existence or likelihood of a connection between two nodes, which can be applied to recommend connections in social networks or predict protein-protein interactions. Assigning a label or category to an edge, such as determining the type of relationship between two entities in a knowledge graph, falls under *edge classification*. Additionally, *edge regression* tasks involve predicting a continuous value associated with an edge, such as estimating the strength of a social tie or the weight of a transportation link.

Graph-level tasks When we move beyond analyzing individual nodes and edges to understanding entire graphs, GNNs face a new set of challenges. Graph-level tasks require the model to reason about the holistic properties of the graph structure. For instance, classifying a molecule as possessing a specific pharmaceutical property, like inhibiting a particular enzyme, based solely on its atomic structure represented as a graph, exemplifies a *graph classification* task. In contrast, *graph regression* involves predicting a continuous value associated with the entire graph, such as estimating the precise binding affinity of a molecule to a target protein, a crucial factor in drug discovery. Beyond classification and regression, grouping similar graphs together based on their structural properties and features, known as *graph clustering*, can be helpful for identifying patterns in collections of molecules, ultimately enabling the discovery of molecules with similar pharmaceutical properties.

To capture this crucial structural information, GNNs can be designed to incorporate higher-order variants capable of recognizing complex substructures within the graph. One powerful tool for achieving this is the Laplacian matrix [93], a mathematical representation of the graph's connectivity. The Laplacian encodes essential information about the graph's structure, enabling GNNs to learn representations that reflect the overall graph properties. By leveraging the Laplacian, GNNs can enforce smoothness in node representations, ensuring that connected nodes have similar representations. This is particularly helpful for tasks like graph classification, where similar graphs should be grouped together. Additionally, the eigenvalues and eigenvectors of the Laplacian matrix reveal important structural properties like connectivity and diameter, further enhancing the GNN's understanding of the graph.

3.3 Reinforcement Learning

3.3.1 Learning through Trial-and-Error

Reinforcement Learning (RL) [94] is a dynamic subset of machine learning where an agent learns by actively interacting with its environment. Unlike supervised learning, which relies on labeled data,

or unsupervised learning, which seeks patterns in unlabeled data, RL focuses on learning through trial and error. The agent learns through a system of rewards and penalties, gradually improving its decision-making strategy to maximize its long-term gains.

3.3.2 The RL Framework

In reinforcement learning, an **agent**, the decision-making entity, interacts with an **environment** by making choices called **actions**. The **state** represents the agent's current situation within the environment, and each action can potentially change that state. The environment provides **rewards** as feedback, indicating the success or value of an action taken in a particular state. The agent's overarching goal is to learn an optimal **policy**, essentially a function that maps states to actions, that dictates which action to take in each possible state to maximize cumulative rewards over time.

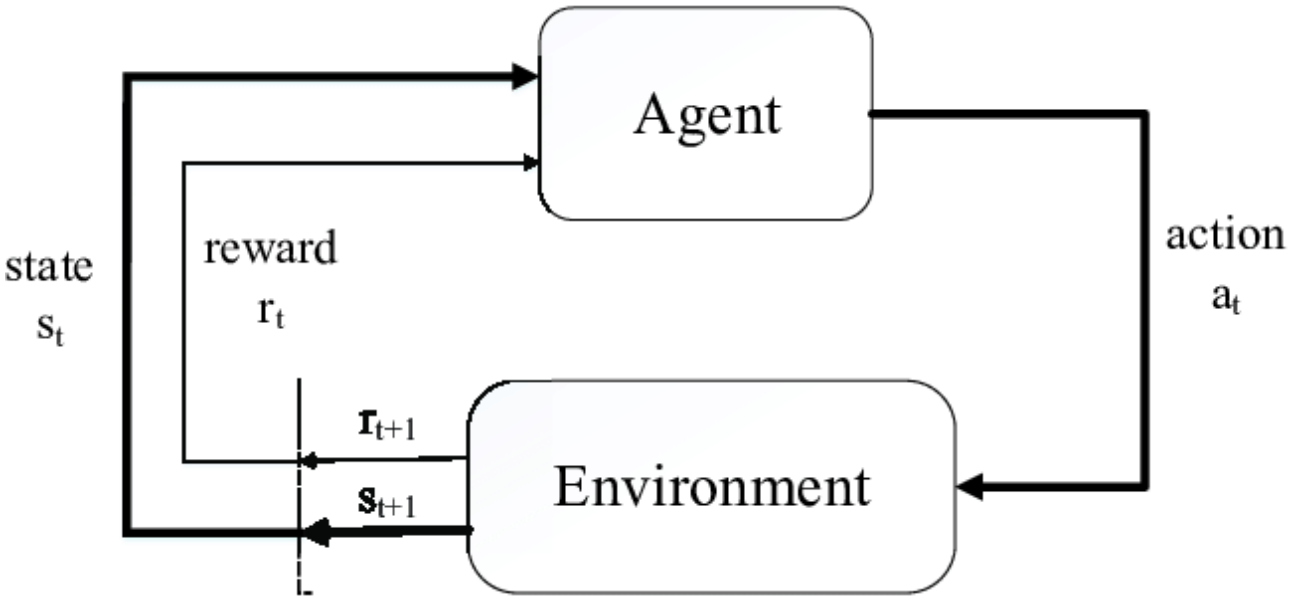


Figure 3.1: The reinforcement learning framework : At time-step t, the agent observes a representation of the environment's state S_t, selects an action A_t and receives a reward R_t.

The interaction between the agent and environment is typically modeled as a Markov Decision Process (MDP):

A Markov Decision Process is defined by a tuple:

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$$

where $\mathcal{S}$ is the set of all possible states, referred to as the state space; $\mathcal{A}$ is the set of all possible actions, referred to as the action space; **p** is the transition function where $p(s'|s, a)$ gives the probability of transitioning to state s' when taking action a in state s; and **r** is the reward function where $r(s, a)$ gives the immediate reward received for taking action a in state s.

3.3.3 Returns and Episodes

The agent's interaction with the environment is structured into episodes.

An episode is a sequence of states, actions, and rewards, starting from an initial state and ending in a terminal state or after a predetermined number of steps.

An episode can be represented as:

$$S_t \xrightarrow{A_t} S_{t+1} \xrightarrow{A_{t+1}} S_{t+2} \xrightarrow{A_{t+2}} \cdots$$

where t is a time step.

The agent's goal is to maximize the cumulative reward, known as the return.

The return (G_t) at time step t is the sum of rewards obtained from that time step onwards.

Often, a discounted return is used to prioritize immediate rewards:

$$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots,$$

where $\gamma \in (0, 1)$ is the discount factor.

3.3.4 Q-Learning: A Foundational Algorithm

The Value Function:

In Q-learning, we estimate the expected future reward using the Q-value, represented by $Q(s, a)$. This value estimates how good it is to take action a in state s, and then follow the agent's current policy.

Learning by Updating:

Q-learning updates Q-values iteratively using:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

where α is the learning rate, r the immediate reward after taking action a in state s, γ the discount factor and $\max_{a'} Q(s', a')$ the maximum estimated future reward from the next state s'.

Goal:

By repeatedly updating Q-values, they become accurate estimates of the true rewards. The agent improves its policy by choosing actions with the highest Q-values in each state.

3.3.5 Deep Q-Networks

The DQN algorithm addresses the challenges of high-dimensional state spaces by using deep neural networks to approximate the Q-function. In complex environments with many states and actions, tabular Q-learning becomes impractical. Deep neural networks provide a powerful function approximation tool, where the network takes the current state as input and outputs the estimated Q-values for each possible action. DQN's also incorporate experience replay, where past experiences (state, action, reward, next state) are stored in a buffer. During training, random samples from this buffer are used, improving learning stability and breaking the correlations that can arise in sequential data.

The DQN algorithm seeks to learn the optimal Q-function, which estimates the expected cumulative reward of taking an action in a given state and following an optimal policy thereafter. Formally, the optimal Q-function is defined as:

$$Q^*(s, a) = \mathbb{E}\left[r(s, a) + \gamma \max_{a'} Q^*(s', a')\right] \tag{3.1}$$

where s' represents the state resulting from taking action a in state s.

During training, the network's parameters are adjusted to minimize the difference between the predicted Q-value and a target Q-value derived from the Bellman equation [95]. This is achieved by minimizing the following loss function:

$$L(\theta) = \mathbb{E}\left[\left(y^{DQN} - Q(s, a; \theta)\right)^2\right] \tag{3.2}$$

where $y^{DQN} = r + \gamma \max_{a'} Q(s', a'; \theta)$ is the target, calculated using the current network itself. The expectation $\mathbb{E}[\ldots]$ is typically approximated using a mini-batch of experiences sampled from the replay buffer.

3.3.6 Balancing Exploration and Exploitation

A crucial challenge in reinforcement learning is striking the right balance between exploration and exploitation. Exploration involves trying new actions to potentially discover better long-term strategies, while exploitation leverages known information to maximize immediate rewards.

In this work, we employ the **epsilon-greedy** (ε-greedy) strategy for balancing exploration and exploitation. With probability ε, a random action is selected (exploration). Otherwise, with probability $1 - \varepsilon$, the action with the highest estimated Q-value is chosen (exploitation). The value of ε often decays over time, encouraging initial exploration and favoring exploitation as learning progresses.

While effective, it's important to acknowledge that other exploration strategies exist, each with its own advantages and trade-offs:

- **Boltzmann Exploration:** Actions are chosen probabilistically based on their Q-values, ensuring all actions have some chance of being selected.
- **Upper Confidence Bound (UCB):** Balances Q-value estimates with uncertainty, favoring exploration of less-tried actions.
- **Curiosity:** Intrinsically rewards the agent for discovering surprising outcomes.

3.3.7 Some extensions to DQN

Several extensions have been proposed to enhance the performance and stability of the basic DQN algorithm:

Double DQN (DDQN):

DDQN tackles the issue of overestimation bias potentially present in Q-learning methods. It achieves this by decoupling the action selection and action evaluation processes. DDQN uses the current Q-network to choose the best action, but relies on a separate target network for evaluating the Q-value of that action. This helps reduce overoptimism and improves stability:

$$y^{DDQN} = r + \gamma Q(s', \arg\max_{a'} Q(s', a'; \theta); \theta') \tag{3.3}$$

Where θ represents the current Q-network's parameters and θ' represents the parameters of a target network which is periodically updated with the current network's weights.

Prioritized Experience Replay:

Instead of uniformly sampling from past experiences, prioritized experience replay prioritizes transitions that have a high Temporal Difference (TD) error. The TD error signifies a large discrepancy between the estimated Q-value and the target Q-value, making these experiences more informative for learning. Prioritizing these transitions leads to more efficient use of data.

$$TDError = |Q(s, a; \theta) - y^{DQN}| \tag{3.4}$$

Dueling Networks:

This architecture enhances Q-networks by explicitly separating the estimation of state values and action advantages:

- **State-value function (V(s)):** Represents the overall value of being in a particular state.
- **Advantage function (A(s, a)):** Represents the relative advantage of choosing a specific action within that state. The advantage function is directly related to the Q-value function through the following equation:

$$Q(s, a) = V(s) + A(s, a)$$

This relationship highlights that the Q-value of an action can be decomposed into the value of being in the state and the advantage of taking that specific action in that state.

This separation can improve generalization and help the agent focus on the value of states, especially when multiple actions might yield similar outcomes. However, directly using the equation above can lead to identifiability issues (i.e., an infinite number of values for $V(s)$ and $A(s,a)$ can result in the same $Q(s,a)$). To address this, the Q-value in a dueling network can be calculated as follows:

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + \left(A(s,a;\theta,\alpha) - \frac{1}{|\mathcal{A}|} \sum_{a' \in \mathcal{A}} A(s,a';\theta,\alpha) \right) \tag{3.5}$$

Where θ represents the shared parameters between the value and advantage streams, and α and β are parameters specific to each stream. $\mathcal{A}$ denotes the set of all possible actions in the given environment.

By subtracting the average advantage from the advantage term, we force the advantage function to be zero-centered, improving the stability and learning dynamics of the network.

3.4 Building real world diffusion graphs for IM

To understand how information propagates through online social networks, we analyze diffusion cascades. These cascades represent sequences of information transmissions between users, where one user's action (e.g., sharing a post) influences others to take similar actions. From these cascades, we can infer the likelihood of information diffusion on a network. In the following, we explore two approaches for modeling information diffusion on social graphs, considering both vanilla (i.e., topic-agnostic) and topic-aware IM settings.

Topic-Agnostic Setting:

In this setting, we focus on the structural properties of the network, neglecting the thematic content of the information being diffused. One popular technique for constructing edge weights in topic-agnostic models is based on [96] (also employed by [97]). This method considers the frequency of reposts between users and the average delay between reposts to weight the edges in the cascade graph:

$$p(u,v) = \left(\frac{A_{v2u}}{A_v} \right) \exp\left(-\frac{\bar{D}}{\delta} \right)$$

where $p(u,v)$ represents the weight of the edge between users u and v. This weight is calculated based on multiple factors, including A_{v2u}, the number of times user u reposted content from user v, and A_v, the total number of posts created by user v. Additionally, $\bar{D}$ signifies the average time delay between a repost from user v and a repost by user u. The parameter δ controls the decay rate of the edge weights with respect to this time delay, affecting how the interaction strength diminishes over time.

Topic-Aware Setting:

This setting incorporates the thematic content of the information being diffused. Survival Factorization [34], a powerful framework for modeling topic-aware information diffusion, addresses the complexities of understanding how information spreads through networks by incorporating social influence patterns, topical structures, and temporal dynamics. This is achieved through a low-dimensional latent space that captures key aspects of the diffusion process.

The model acknowledges that information cascades are driven by the content and themes they represent. Each cascade is associated with a specific topic, drawn from a multinomial distribution over a set of K topics.

Survival Factorization recognizes the heterogeneity of users within a network. Each user is assigned two vectors within the latent space, representing their authoritativeness and susceptibility across the K topics.

- **Authoritativeness ($A_{v,k}$):** This reflects the user's ability to influence others on a given topic. Users with high authoritativeness are more likely to trigger the adoption of information within their network.
- **Susceptibility ($S_{u,k}$):** This signifies the user's openness to being influenced by others on a specific topic. Users with high susceptibility are more likely to adopt information when exposed to it.

The interplay between authoritativeness and susceptibility determines the pairwise transmission rates, which are modeled using the following equation:

$$\lambda_{v,u,k} = A_{v,k} \cdot S_{u,k}$$

This equation implies that the probability of user v influencing user u regarding topic k depends on both v's expertise on the topic and u's receptiveness to it.

Information cascades unfold over time, and the model incorporates this temporal aspect by employing a survival analysis framework. Specifically, the Weibull distribution is used to model the delay between the activation times of users within a cascade:

$$f(t_u(c)|t_v(c), \lambda_{v,u,k}) = Weib(\Delta_c^{u,v}; \lambda_{v,u,k}, \rho)$$

where $f(t_u(c)|t_v(c), \lambda_{v,u,k})$ represents the probability density function of user u's activation time and $Weib(\Delta_c^{u,v}; \lambda_{v,u,k}, \rho)$ denotes the Weibull distribution. The Weibull distribution is characterized by $\Delta_c^{u,v} = t_u(c) - t_v(c)$ representing the time delay between user activations, $\lambda_{v,u,k}$ serving as the scale parameter reflecting the transmission rate within topic k, and ρ acting as the shape parameter influencing the shape of the probability density function and capturing the temporal dynamics of activation.

This allows the model to capture the dynamics of information propagation and predict future adoption behaviors based on the observed temporal patterns.

The Survival Factorization model combines these elements through a generative process. First, a topic is selected for the cascade. Then, side information (such as hashtags) is generated based on the chosen topic using a Poisson language model. Finally, user activation times are generated according to the Weibull distribution, with the transmission rates determined by the user's authoritativeness and susceptibility on the given topic.

3.5 Graph Laplacian methods

In GNNs, effectively representing the relative positions of nodes within a graph is crucial for capturing structural information and enabling the model to reason about relationships between nodes. Graph positional encoding (GPE) techniques address this need by embedding positional information into node features, similar to positional encodings in transformer models for natural language processing. In this thesis, we explore the contrasting roles of two mathematical tools —the widely used combinatorial Laplacian and the more recently developed magnetic Laplacian— within the context of specific GPE methods.

Combinatorial Laplacian

The combinatorial Laplacian provides insights into the connectivity and structure of a graph. Its eigenvectors and eigenvalues reveal important graph characteristics, such as connected components, community structure, and diffusion behavior.

To utilize the combinatorial Laplacian for GPE, a common approach involves:

- **Symmetrization of the Adjacency Matrix:**. This step ensures the Laplacian matrix is symmetric and positive semi-definite, guaranteeing real-valued eigenvectors that form an orthogonal basis. This property is crucial for positional encoding as it allows for meaningful comparisons between node positions.

$$\tilde{A}_{i,j} = \frac{1}{2}(A_{i,j} + A_{i,j}), 1 \leq i, j \leq n \tag{3.6}$$

Since the weights are positive diffusion probabilities, the degree matrix can be expressed as:

$$\tilde{D}_{i,i} = \sum_{j=1}^{n} \tilde{A}_{i,j} = \frac{1}{2} \sum_{j=1}^{n} (A_{i,j} + A_{i,j}), 1 \leq i \leq n \tag{3.7}$$

- **Degree Normalization:** Normalizing the Laplacian by the degree matrix accounts for the varying degrees of nodes, preventing bias towards highly connected nodes.

$$L_N = I - (\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}) \tag{3.8}$$

- **Eigenvector Selection:** The eigenvectors, except for the first (constant) one, are used as positional encodings, capturing structural information based on the graph's connectivity.

Magnetic Laplacian

The magnetic Laplacian [98] extends the concept of the combinatorial Laplacian to directed graphs by incorporating edge direction information using complex numbers. This makes it conceptually similar to sinusoidal positional encodings used in transformers, which encode sequence order information.

Key points regarding the magnetic Laplacian for GPE are:

- **Complex Eigenvectors:** Due to its construction, the magnetic Laplacian has complex eigenvectors. However, the element-wise modulus of these eigenvectors provides real-valued positional encodings.
- **Encoding Directionality:** The magnetic Laplacian encodes the directionality of edges, allowing the GNN to distinguish between incoming and outgoing connections, which is crucial for capturing information flow within the graph.

$$L_N^{(q)} = I - (\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}) \odot \exp\left(i\Theta^{(q)}\right) \tag{3.9}$$

  with the Hadamard product $\odot$, element-wise $\exp$, $i = \sqrt{-1}$, $\Theta_{u,v}^{(q)} = 2\pi q(A_{u,v} - A_{v,u})$, and potential $q \geq 0$.

- **Relationship to Combinatorial Laplacian:** In the special case of an undirected graph, the magnetic Laplacian reduces to the combinatorial Laplacian, highlighting its generalizing property.

Advantages and Considerations:

Both Laplacian-based GPE methods offer valuable tools for incorporating structural information into GNNs. The choice between them depends on the nature of the graph and the specific task. For undirected graphs, the combinatorial Laplacian provides a simple and efficient approach. For directed graphs, the magnetic Laplacian offers a more nuanced representation by capturing directionality, potentially leading to improved performance in tasks where edge direction is important.

3.6 Meta-Learning

Meta-learning, also known as "learning to learn", is a field of machine learning that focuses on developing models capable of rapidly acquiring new skills or adapting to changing environments with minimal additional training. Its central goal is to extract knowledge from previous experiences and apply it to accelerate and enhance learning on future, unseen tasks [99].

Meta-learning differs from traditional approaches that concentrate on optimizing performance for a single task. Instead, it emphasizes the learning process itself. A meta-learning model is exposed to a diverse set of learning tasks, enabling it to excel on new tasks, even with limited training examples.

In meta-learning, tasks are sampled from a task distribution. A task, denoted as T_i, is characterized by two fundamental components: a dataset and a loss function. The dataset, represented as $D_i = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, comprises a collection of input-output pairs specific to the task and is divided into two subsets: a support set and a query set. The *support set* provides a limited number of labeled examples, acting as a training set for the model to learn the task. The *query set*, on the other hand, contains examples used to evaluate the model's performance after it has learned from the support set. Finally, the loss function L_i measures the model's overall performance on the given task, typically by assessing its predictions on the query set against the true labels.

Meta-learning encompasses a diverse range of approaches for learning and adaptation across tasks, with gradient-based and metric-based methods being two prominent examples

3.6.1 Gradient-Based Meta-Learning

Gradient-based meta-learning aims to accelerate adaptation across various tasks. The Model-Agnostic Meta-Learning (MAML) algorithm is a prominent example [99].

MAML aims to find a set of initial model parameters that serve as a good starting point for learning new tasks with minimal gradient updates. Instead of training a separate model for each task, MAML seeks a single model that can be easily fine-tuned to perform well on any task from the distribution.

MAML aims to minimize the following loss function:

$$\min_{\theta} \sum_{T_i \sim p(T)} L_{T_i}(f_{\theta_i'}) = \min_{\theta} \sum_{T_i \sim p(T)} L_{T_i}(f_{\theta - \alpha \nabla_\theta L_{T_i}(f_\theta)}) \tag{3.10}$$

where θ represents the initial model parameters to be optimized, T_i is a task sampled from a distribution of tasks $p(T)$, L_{T_i} is the task-specific loss function measuring the model's performance on task T_i, and $f_{\theta_i'}$ is the model after being updated for task T_i with parameters θ_i' obtained through gradient steps on the task. This equation encapsulates the goal of finding initial parameters θ that, after minor adjustments for a new task, lead to the lowest possible loss on that task, thereby ensuring rapid learning and adaptation capabilities.

Inner and Outer Loop

MAML operates with two levels of learning:

- **Inner loop (Task-specific adaptation):** The model uses the support set to update its parameters through gradient descent, minimizing the task-specific loss L_{T_i}. This process allows the model to adapt to the specific task at hand.

  $$\theta_i' = \theta - \alpha \nabla_\theta L_{T_i}(f_\theta) \tag{3.11}$$

  where α is the inner-loop learning rate. This results in task-specific parameters θ_i'.

- **Outer loop (Meta-update):** The model's performance on the query set, after adapting to each task in the inner loop, is used to update the initial model parameters θ. This step aims to improve the model's ability to learn quickly and effectively across a variety of tasks.

  $$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{T_i \sim p(T)} L_{T_i}(f_{\theta_i'}) \tag{3.12}$$

  where β is the meta-learning rate.

3.6.2 Metric-Based Meta-Learning

Metric-based meta-learning is a powerful paradigm within meta-learning that focuses on learning effective distance metrics or similarity functions. This approach is essential for scenarios where training data is limited, such as in few-shot learning. By learning to accurately compare instances, these models gain the ability to generalize well from a small number of examples.

At the heart of metric-based meta-learning lies the concept of embedding data points into a representational space. The key characteristic of this space is that the distances between these embedded

points directly correspond to their semantic similarity. A well-trained model within this paradigm demonstrates two key capabilities. First, it can effectively discriminate between different classes, ensuring that examples from distinct categories are well-separated within the embedding space. Second, it exhibits the ability to cluster similar examples, drawing semantically related data points closer together within this representational landscape.

Formalization

The fundamental goal of metric-based meta-learning is to learn a function f that maps instances x_i and x_j into an embedding space. A distance metric d is then used to quantify their similarity:

$$d(f(x_i), f(x_j)) \tag{3.13}$$

Typically, the embedding function f is parameterized by a neural network, and common choices for the distance metric d include Euclidean distance or cosine distance.

Chapter 4

Influence Maximization in Social Graphs

4.1 Introduction

Social media has redefined the ways in which information spreads. By word-of-mouth mechanisms, viral spread can happen rapidly and at unprecedented scale. Whether for advertising or political / public-awareness campaigns, understanding and taking advantage of this highly effective medium for information diffusion is paramount. Unsurprisingly, the key for a successful diffusion campaign is to know where to start, i.e., to choose the best sources (*spread seeds*) from which to initiate the dissemination of information. Under the generic name of Influence Maximization (in short, IM), we have seen in recent years many advances, at the intersection of combinatorial optimization on graphs and network analysis [10].

Despite diverse formulations, at their core, most IM studies share a common focus: select an optimal set of spread seed nodes of size k (the budget) in a *diffusion graph* – a directed graph whose edges are labelled by a diffusion probability. Optimality may have various definitions, but usually boils down to maximizing the expected spread under a specific diffusion model such as Independent Cascades (IC).

While conceptually simple and extensively studied, IM remains notoriously difficult, with few solutions being truly applicable in real-life scenarios. IM is in general NP-hard [10], and most studies exploit the monotonicity and submodularity of the spread objective, which guarantee the greedy selection algorithm to achieve at least 63% of the optimal spread. However, the greedy approach itself remains computationally prohibitive, as one key step therein – the estimation of the *marginal spread gain* for a candidate seed w.r.t. to the current partial solution – is #P-hard and requires in practice expensive sampling steps (Monte Carlo, RR sets, etc) [47, 48].

As a possible direction for IM solutions that can be applied *at scale* to answer in *real-time* IM queries, we consider in this thesis an approach that aims to replace the expensive estimation of marginal gain by a prediction thereof. In short, we want to pre-train a model that can predict *at query time* a node's quality in a given *context*, namely for a given diffusion graph, IM query, and partial seed set. With likely but hopefully acceptable loss in effectiveness, the trade-off would be to gain significantly in efficiency.

In essence, we aim to train a predictive model $\mathcal{M}$ for marginal gain, by DRL. However, training the model is computationally expensive, especially on large graphs. We argue that this upfront cost can be justified only if it is amortized over multiple queries. Our approach is thus akin to some of the existing IM solutions, which pre-compute and index key information to speed up computation at query time (but the key information in our framework is a model trained by DRL). More precisely, we propose a 3-staged approach.

1. in a *training stage*, we build the predictive model $\mathcal{M}$ using well-chosen and manageable samples of the diffusion graph $\mathcal{G}$; this stage may be slow.
2. in a *pre-query / embedding stage*, we build the representation of the entire input graph $\mathcal{G}$ (on which IM queries are to be computed), to be given as input to $\mathcal{M}$; this stage may be quite fast, yet not real-time.
3. in a *query stage*, IM queries Q would be answered, using $\mathcal{G}$'s embedding and $\mathcal{M}$; this must be fast for each query.

In light of this, two important questions need to be clarified:

Q1 As our primary motivation stems from the scalability and efficiency limitations of classic IM solutions, *is the idea of training such a model $\mathcal{M}$ justified for real-time IM querying*?

Q2 As social graphs are highly dynamic, *how robust can the model $\mathcal{M}$ be, with respect to diffusion graph changes*?

We believe that [Q1] can be answered positively only if we have a large space of potential IM queries. Vanilla IM queries (i.e., under a traditional IM formulation, without other context dimensions such as location, topic, or time) simply amount to a budget value k, but most diffusions in social media are *topic-aware*, requiring a larger space of potential IM queries. Indeed, in real-world applications, diffusions convey messages with various topical distributions, making it necessary to handle a much larger space of potential IM queries. Therefore, our thesis is that answering Q1 positively – for efficient / real-time query response – within the DRL framework *hinges* on topic-aware diffusion models.

Topic-awareness: topic-aware IM is not simply a sub-problem of traditional IM. It is a significantly more complex formulation thereof, where diffusion probabilities are determined by the content being shared. Existing solutions for vanilla IM cannot be directly applied nor easily adapted to address the challenges of topic-aware IM. This is analogous to how non-learning based vanilla IM methods required innovative extensions and new ideas, as in INFLEX, to handle the topic-aware case.

Regarding [Q2], $\mathcal{M}$ must be robust to graph evolution: if trained on a snapshot of the graph at time t, it must be able to make predictions on future snapshots of the same diffusion medium, which most likely remain structurally similar. We stress that we focus on *robustness* instead of *generalizability* – i.e., the ability of $\mathcal{M}$ to make predictions on other graphs, unseen at training and maybe structurally different – as we believe the former is more important for practical scenarios. E.g., there is little practical interest in training on a Twitter diffusion graph and testing on a Meta Threads one.

Our contributions. We revisit in this thesis the generic framework S2V-DQN of [45], which designs heuristic algorithms for graph-based combinatorial optimization problems using DRL.

IM-GNN. We begin with an IM adaptation of S2V-DQN (**S2V-DQN-IM**) as a proof-of-concept. Through progressive refinements, we evolve this initial adaptation into our first approach, **IM-GNN**, which shows competitive performance w.r.t. the existing learning-based methods for *vanilla IM*. In particular, we integrate in **IM-GNN** attentive GNNs and positional encoding with the novel graph magnetic Laplacian [98]. Importantly, for our comparison of IM-GNN with SOTA baseline methods, we use public, real-world datasets and diffusion graphs built from cascades, i.e., with *data-based* diffusion probabilities. In contrast, the graphs used in the evaluation of existing methods are mostly artificial and topology-bound, with uniform, trivalency, or degree-based edge probabilities.

TIM-GNN. Building on the confirmation of IM-GNN's effectiveness, we extend our framework to incorporate *topic-awareness*, leading to our method **TIM-GNN**. For a realistic evaluation, from public data, we extract topic-aware diffusion graphs from information cascades, using the survival factorization framework of [34]. We assess the performance of TIM-GNN on (i) effectiveness (spread), (ii) efficiency (query latency), and (iii) robustness to changes in the diffusion graph. The existing baseline methods we use for comparison are either *topic-agnostic learning-based* or *non-learning based topic-aware*. Our experimental results show that TIM-GNN can meet the stringent requirements of real-world applications, being superior to the state-of-the-art, albeit relatively slow at query time.

TIM-GNN^x. Finally, to improve latency, we incorporate *cross-attention mechanisms*. We refine TIM-GNN to accurately predict a high-quality ranking of seed nodes for any query (item and budget combination), by using a pre-computed Q-matrix derived from well-chosen representative *base items*. The resulting approach (**TIM-GNN^x**) allows for real-time yet effective assessment of influence spread, significantly reducing the need for extensive graph message passing operations during each assessment. As a result, we mostly preserve the robust performance of our first-cut topic-aware algorithm TIM-GNN, while achieving a $10x - 20x$ query time speed-up.

4.2 Problem Definition

Diffusion networks and IM. A social (diffusion) network is commonly defined as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{P})$, where $\mathcal{V}$ denotes the nodes (users), $\mathcal{E}$ is the set of edges between them, and $\mathcal{P}$ is a probability func-

tion on $\mathcal{E}$, such that information propagates along an edge (u, v) according to the edge probability (or weight) $p_{u,v}$. Influence can spread through this network according to a *diffusion model*. A diffusion model is a set of rules and probabilities that govern how information or influence propagates through the network. Common models include Independent Cascade, Linear Threshold, and their variations. Given a social network and a diffusion model, *Influence Maximisation* aims to select a set of seed nodes $S \subseteq \mathcal{V}$, of size at most k, such that the expected spread of influence starting from S (or the expected number of activated nodes) is maximized.

Independent Cascades (IC). IC is the most well-known diffusion model. In IC, each node can be in one of two states: active or inactive. At the initial time step, only the nodes in the seed set S are active. At each subsequent time step, all nodes that transitioned from inactive to active at the previous time step will independently make a unique attempt to activate each of their inactive neighbors. A neighbor will be activated with probability equal to the weight of the edge between the two nodes. The propagation ends when no nodes can activate any of their neighbors. The *spread* $\sigma(S)$ is the number of nodes that are activated at the end of the propagation.

Topic-aware IM. As an extension to IC, we consider the TIC diffusion model, initially proposed in [53], which takes into consideration the topical description of the information being diffused. TIC assumes that each edge in the graph may spread information pertaining to a certain number d of topics: namely, $(u, v) \in \mathcal{E}$ is associated with a vector $\mathbf{p}_{u,v} = \left(p^1_{u,v}, p^2_{u,v}, ..., p^d_{u,v}\right)$, $0 \leq \mathbf{p}_{u,v} \leq 1$, where $p^z_{u,v}$ is the weight associated to topic z, denoting the probability for topic z that user u activates user v.

Given an item, i.e., a topic distribution vector $\overrightarrow{\gamma}$ as the information that is diffused, $\overrightarrow{\gamma} = \left(\gamma^1, \gamma^2, ..., \gamma^d\right)$ s.t. $\sum_{1 \leq i \leq d} \gamma^i = 1$, for each edge (u, v), the propagation probability along that edge w.r.t. $\overrightarrow{\gamma}$ is:

$$p_{u,v}(\overrightarrow{\gamma}) = \langle \mathbf{p}_{u,v}, \overrightarrow{\gamma} \rangle = \mathbf{p}^{\top}_{u,v} \overrightarrow{\gamma} \tag{4.1}$$

It is this propagation probability that will be used for edge (u, v), as described before, in a *TIC diffusion process*. When, for a given item $\overrightarrow{\gamma}$ the Eq. (4.1) is applied to all the edges of $\mathcal{G}$, we say the item is *projected* on $\mathcal{G}$, leading to a topic-agnostic diffusion graph (having a single diffusion probability per edge). In topic-aware IM, the objective is then to find the best seed set given an item-budget query $\mathcal{Q} = (\overrightarrow{\gamma}, k)$, and we can define $\sigma(S|\overrightarrow{\gamma})$ similarly.

Greedy algorithm and diffusion simulations. As the objective function in IM is monotone submodular, the general approach for finding an approximate solution is based on the greedy algorithm. It selects at each step a new seed node that yields the largest marginal gain on expected spread w.r.t. the current partial solution. However, computing the expected spread is #P-hard for IC (and clearly for TIC as well), so most traditional IM research works focus on approximation methods. One such method is to simulate r random cascades from a given seed node and average the number of influenced nodes to approximate the marginal spread.

We can formally define topic-aware IM as follows:

Problem 1 (Topic-aware IM) *Given the topic-aware network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{P})$ and a query $\mathcal{Q} = (\overrightarrow{\gamma}, k)$, find a seed set $G^* = \arg\max_S \sigma(S|\overrightarrow{\gamma})$, where $S \subset \mathcal{V}$, $|S| = k$.*

In the greedy seed selection algorithm, the node with the largest maximal gain can be defined as follows:

Problem 2 (Seed node selection) *In the setting of Problem 1, given a partial solution S', $|S'| < k$, of the greedy seed selection, find the node $s \in \mathcal{V} - S'$ with the largest marginal spread gain w.r.t S', i.e., $\arg\max_s [\sigma(S' \cup \{s\}|\overrightarrow{\gamma}) - \sigma(S'|\overrightarrow{\gamma})]$.*

Finally, our goal is to substitute any expensive simulation-based estimation of marginal spread with a prediction thereof, given by a model $\mathcal{M}$ trained on the diffusion graph.

Problem 3 (Marginal gain prediction) *For a known topic-aware network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, build a predictive model $\mathcal{M}$ that – for any query $\mathcal{Q} = (\overrightarrow{\gamma}, k)$ and partial greedy solution S' – predicts the marginal spread gain w.r.t. S' for any node $s \in \mathcal{V} - S'$,*

$$\mathcal{M}(\overrightarrow{\gamma}, S', s) \approx [\sigma(S' \cup \{s\}|\overrightarrow{\gamma}) - \sigma(S'|\overrightarrow{\gamma})] \tag{4.2}$$

Our focus is on solving in a satisfactory manner Problem (3), s.t. Problem (1) can be answered more efficiently, by substituting the spread σ-expression in Problem (2) with an $\mathcal{M}$-prediction thereof.

4.3 Proposed Approach

We present our successive adaptations of the DRL framework S2V-DQN [45], starting with two models for vanilla IM – **S2V-DQN-IM** (SOTA) and **IM-GNN** (ours) – and then the two models we propose for topic-aware IM, **TIM-GNN** and **TIM-GNN**^{x}.

4.3.1 DRL formulation

We model the decision making procedure as a Markov Decision Process (MDP) and define the states, actions, and rewards according to the RL framework:

- **State**: The state S' corresponds to the current partial solution. A node therein is represented by its embedding.
- **Action**: The action consists in adding a node v to to the current partial solution, for which the agent receives a reward from the environment and moves to a new state.
- **Reward**: The reward function $r(S', v)$ for state S' after selecting the new seed v and transitioning to state $S' \cup \{v\}$ is the marginal spread gain, i.e., $\sigma(S' \cup \{n\}) - \sigma(S')$
- **Environment**: The environment consists of a diffusion graph with a diffusion model (IC or TIC in this work).
- **Policy**: Based on $\hat{Q}$, we select actions by ϵ-greedy policy

$$\pi(v|S) = \begin{cases} \arg\max_{v \in \overline{S'}} \hat{Q}(S', v) \text{ with probability } 1 - \epsilon \\ \text{sample uniformly a node } v \in \overline{S'} \text{ otherwise.} \end{cases}$$

  This favors exploration and enables the model to test different actions, independently of its own predictions.

In the following, we concisely describe the Prioritized Double DQN model, we refer the reader to chapter 3 for more details. We use the term episode to refer to a complete sequence of actions, from empty to complete solution (seed set), with each step representing one seed selection. In the simplest DQN formulation [100], the agent selects an action ϵ- greedily, based on the current state and the action values, and adds a transition $(S_t, A_t, R_{t+1}, \gamma_{t+1}, S_{t+1})$ to a replay memory buffer. The buffer or experience replay increases data efficiency by allowing to use the same sample in multiple updates. It also helps reduce the correlation between samples used in the updates, hence it helps reduce variance. The parameters are updated by minimizing the following loss function:

$$(R_{t+1} + \gamma_{t+1} \max_{a'} q_{\overline{\theta}}(S_{t+1}, a\prime) - q_\theta(S_t, A_t))^2 \tag{4.3}$$

where t is a time step randomly sampled from the replay buffer, and $\overline{\theta}$ represents the parameters of a target network, a periodic copy of the online one θ, which is not optimized. Prioritized Double DQN adopts *double Q-learning* to address an overestimation bias of Q-learning, and *prioritized replay* improves data efficiency by replaying more often a transition from which there is more to learn.

4.3.2 S2V-DQN-IM: A Foundational Approach

S2V-DQN-IM serves as our initial model, stemming from the S2V-DQN framework of [45] and being tailored for vanilla IM. This basic version involves training on sub-graphs extracted from the input diffusion graph, and is followed in the PIANO approach of [59]. A key component of S2V-DQN-IM's architecture is a Graph Neural Network employing the Structure2Vec framework [101]. In this context, GNNs encompass a range of techniques that leverage topology-based node representation learning via message passing, converting the structural information of nodes in a graph into vector representations, with S2V being one such technique. S2V-DQN-IM incrementally constructs the seed set for IM. Each iteration involves selecting a candidate node guided by an action-value function Q that is derived from GNN-produced node embeddings. In short, Q quantifies the nodes' potential contribution (the marginal spread gain) in the context of the current partial solution.

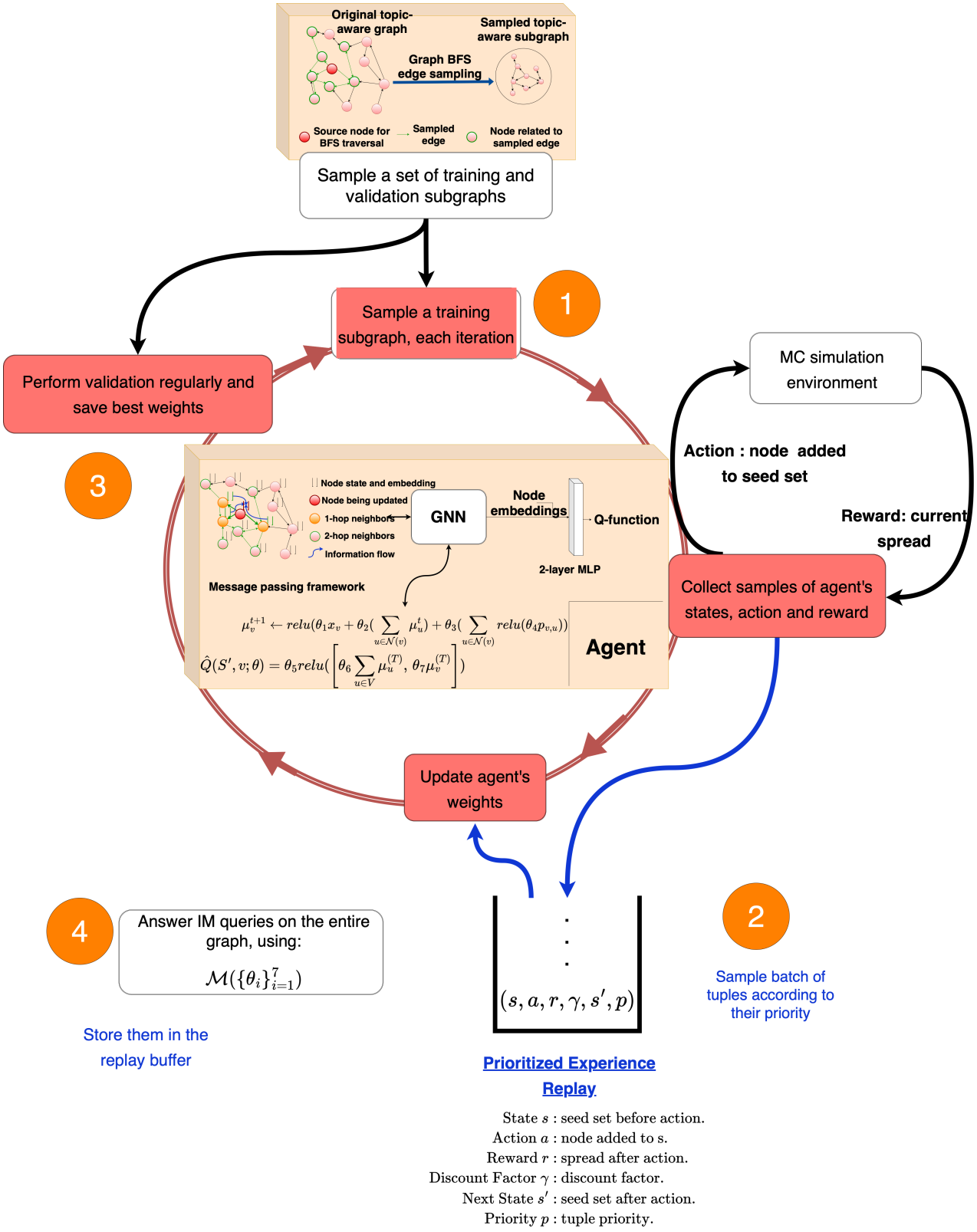


Figure 4.1: Overview of S2V-DQN-IM.

GNN and node embedding. The GNN architecture directly contributes to the parametrization of the Q-function, denoted as $\hat{Q}$. This involves computing a p-dimensional feature embedding μ_v for each node $v \in \mathcal{V}$, taking into account the partial solution S'. The embedding process is as follows:

$$\mu_v^{t+1} \leftarrow relu(\theta_1 x_v + \theta_2(\sum_{u \in \mathcal{N}(v)} \mu_u^t) + \theta_3(\sum_{u \in \mathcal{N}(v)} relu(\theta_4 p_{v,u}))) \quad (4.4)$$

Here, $\theta_1 \in \mathbb{R}^p$, $\theta_2 \in \mathbb{R}^{p \times p}$, $\theta_3 \in \mathbb{R}^{p \times \frac{p}{2}}$, and $\theta_4 \in \mathbb{R}^{\frac{p}{2}}$ are the *model parameters*, p being the embedding

size and $\mu_v^0 = \mathbf{0}$. We denote the set of neighboring nodes of node v in the directed graph as $\mathcal{N}(v)$, where u is considered a neighbor of v if there exists a directed edge from v to u. The rectified linear unit $relu$ ($relu(z) = max(0, z)$) is applied element-wise. Lastly, x_v is a binary scalar that encodes the current partial solution S' such that $x_v = 1$ if $v \in S'$ and $x_v = 0$ otherwise.

Upon completing a number T of message passing iterations within the GNN, as outlined in Eq. (4.4), we obtain the final node vector embeddings, thereby defining the parametrization as follows:

$$\hat{Q}(S', v; \theta) = \theta_5 relu(\left[\theta_6 \sum_{u \in V} \mu_u^{(T)}, \theta_7 \mu_v^{(T)}\right]) \tag{4.5}$$

where $\theta_5, \theta_6, \theta_7 \in \mathbb{R}^{p \times \frac{p}{2}}$, and $[.,.]$ is the concatenation operator. A 2-layer MLP reduces $\hat{Q}$ to one value per node. The model $\mathcal{M}$, comprising the set of parameters $\{\theta_i\}_{i=1}^7$ and the two-layer MLP, is trained in an end-to-end fashion using DRL techniques. More precisely, similar to [45], and subsequently to [59] and [102], we use a *Prioritized Double DQN* [79] to learn $\hat{Q}$.

Overview of the S2V-DQN-IM framework. We provide a comprehensive overview of S2V-DQN-IM in Fig. 4.1, which will serve as the basis for illustrating our successive adaptations of this framework, towards ultimately an effective and query-time efficient topic-aware IM solution. The process begins with **Step (1)**, where each time a subgraph is sampled[1] from the diffusion graph for training purposes. This strategy serves a dual purpose. First, in alignment with [45], it enables training on a variety of instances from the combinatorial optimization problem, which are representative of a consistent distribution, thereby enhancing the model's robustness. Second, considering the often very large size of social graphs, this approach addresses the scalability challenges faced by Graph Neural Networks when applied to large networks. In **Step (2)**, the agent gathers learning samples from these sub-graphs. This is achieved by selecting seed nodes and conducting diffusion simulations to compute the corresponding spread. The accumulated learning samples are stored in a *replay buffer*, which serves as an essential repository during the learning phase of training, contributing significantly to a more efficient use of past experience. The model undergoes continuous validation throughout the learning process (**Step (3)**), where it is tested on larger sub-graph samples. This step is crucial for selecting the most effective model for subsequent evaluation. Finally, in **Step (4)**, the trained model can be used to answer vanilla IM queries. Specifically, it determines an optimal seed set for the entire diffusion graph, for a given query (budget constraint) k. The seed nodes are selected in k successive steps, at each step taking into account the current partial solution. Further details of the algorithm are provided below in Algorithm 1.

4.3.3 IM-GNN: Advanced GNN Features

IM-GNN extends S2V-DQN-IM with some advanced GNN features: GATs [103], positional encodings [104], and self-edges. GATs enable the model to discern the significance of node relationships, while positional encodings facilitate the understanding of node positions in the graph, evolving towards a transformer-like architecture. Self-edges can represent a node's inherent properties, which may not be captured otherwise.

Attention mechanisms. Central to the transformer architecture [105], attention mechanisms in GNNs assign varying importance to neighboring nodes' features.

$$e_{uv}^t = \theta_8^T \left[\mu_u^t, \mu_v^t\right] \tag{4.6}$$

where $\theta_8 \in \mathbb{R}^{2p}$ is a trainable parameter and $e_{uv}^t \in \mathbb{R}$ indicates the importance of node v's features for node u. Then,

$$\alpha_{uv}^t = \frac{\exp(e_{uv}^t)}{\sum_{v \in \mathcal{N}(u)} \exp(e_{uv}^t)} \tag{4.7}$$

where α_{uv}^t is the softmax of e_{uv}^t over u's neighbors. The normalized attention weights are used to compute a linear combination of the features, during message passing aggregation. So Eq. (4.4)

[1]We use Breadth First Search (BFS) sampling, which starts from a random seed node and systematically explores its neighboring nodes, then their neighbors, and so on.

Algorithm 1 S2V-DQN-IM training

Input: Diffusion graph $\mathcal{G}$, experience replay memory Mem initialized to capacity n, learning period L, number of episodes E and steps T
Output: model $\mathcal{M}$'s parameters Θ

for $i = 1$ to I **do**
  Sample a diffusion subgraph g from $\mathcal{G}$
  for $e = 1$ to E **do**
    Initialize the state to empty $S_1 = ()$
    for $t = 1$ to T **do**
      $v_t = \begin{cases} \arg\max_{v \in \overline{S_t}} \hat{Q}(S_t, v) \text{ with probability } 1 - \epsilon \\ \text{random node } v \in \overline{S} \text{ otherwise} \end{cases}$
      Add node v_t to partial seed set $S_{t+1} = (S_t, v_t)$
      if $t \geq n$ **then**
        Add $(S_{t-n}, A_{t-n}, \sum_{i=t-n}^{t} R_i, \gamma, S_t)$ to Mem
      end if
      if $i \times e \times t \mod L = 0$ **then**
        Sample random batch from $\mathcal{B} \sim_{\text{iid.}} Mem$
        Update parameters Θ following Eq.(4.3)
      end if
      if (spread moving average $<$ threshold) **then**
        continue
      end if
    end for
  end for
  return Θ
end for

becomes:

$$\mu_v^{t+1} \leftarrow relu(\theta_1 x_v + \theta_2(\sum_{u \in \mathcal{N}(v)} \alpha_{vu}^t \mu_u^t) + \theta_3(\sum_{u \in \mathcal{N}(v)} \alpha_{vu}^t relu(\theta_4 p_{v,u})) \quad (4.8)$$

Positional encoding. In IM-GNN, we explore two advanced positional encodings for directed diffusion graphs: the combinatorial Laplacian and the magnetic Laplacian [98]. The combinatorial Laplacian, using a symmetrized adjacency matrix, provides insights into the graph's connectivity and structure. The magnetic Laplacian, introducing complex numbers to encode edge direction, gives a more comprehensive representation of directed graphs. This approach aligns conceptually with sinusoidal positional encodings used in transformers [104], offering a structure-aware encoding that acknowledges directedness.

Self-edges. IM-GNN also integrates self-edges in message passing, to capture the nodes' self-interactions. This improves the model's predictive capabilities, despite increasing computational demands. Specifically, it entails the adjustment of Eq. (4.8) to ensure that $u \in \mathcal{N}(u)$ with $p_{u,u} = 0$ for each node u in the considered graph.

4.3.4 TIM-GNN: IM in the TIC model

In the first topic-aware adaptation of our model, TIM-GNN, we enhance IM-GNN (the version using positional encoding with the magnetic Laplacian) to focus on effectively handling topic-aware scenarios. Recall from Sec. 4.2 that, in the topic-aware setting, an IM query $\mathcal{Q}$ consists of a seed set budget k and an item, $\mathcal{Q} = (\vec{\gamma}, k)$. Spread is now determined by the TIC diffusion model, and we need to account for this in our approach for predicting marginal spread gain. To align with the topic-aware setting, the main modification we perform here is in the subgraph sampling scheme. While the BFS subgraph sampling is common to the two previous approaches, its integration in TIM-GNN requires now *uniform sampling of items* and their projection onto the sampled subgraphs. These projected

graphs are then stored and sampled during the training and validation phases. This sampling adaptation ensures that TIM-GNN remains robust in handling topic-specific diffusion patterns.

Referring to Eq. (7), within the trainable parameters of TIM-GNN, θ_3 and θ_4 are specifically related to diffusion probabilities. This new scheme ensures that these parameters are such that the model can generalize across the topic space. The other parameters are optimized by the same process as in IM-GNN, benefiting from the fact that only the diffusion probabilities depend on the item's topical distribution, but the topology of the graph remains unchanged.

4.3.5 TIM-GNN^x: Optimization for query latency

In the development of TIM-GNN^x, our objective was to enhance the efficiency of our topic-aware IM method, specifically focusing on *query latency*. We achieve this by integrating the topic-aware diffusion aspects *directly* into the training process such that, at query time, for any query $\mathcal{Q} = (\overrightarrow{\gamma}, k)$, the projection of the item $\overrightarrow{\gamma}$ onto the diffusion graph is no longer necessary. In other words, the model will be able to predict marginal spread gain directly based on the query and topic-aware diffusion graph. To obtain this capability, we rely in particular on *cross-attention mechanisms*.

To our knowledge, TIM-GNN^x is the first to address the critical challenges detailed in Sec. 4.1: predictive *accuracy* over a broad (topic-based) query space, low *latency* over large topic-aware diffusion graphs, *robustness* to graph changes (see the experiments on graph perturbation below). The flow of TIM-GNN^x is as follows.

Selection of base items. We begin by creating a set of b *base items*, where $b \geq d$ (recall d is the number of topics). Intuitively, these base items should encapsulate the topical diversity of the existing items. One straightforward and data-agnostic method to design the base items is by one-hot encoding, where each item is exclusively associated with a single topic. Assuming the set of possible items known (potentially large, but given), a more precise and data-driven selection of the base items consists in running a clustering algorithm over the training items ($K-Means++$ in our experiments), and using the resulting centroids over the training items. For each of these selection strategies, we further explored two variations: one where the base items are treated as *learnable parameters*, and another where they are static.

Q-matrix computation: For each base item, we apply the previous model TIM-GNN to generate a corresponding Q-function. This process results in a Q-matrix of dimensions (b, $\mathcal{V}$), encapsulating the influence values across different base items and nodes.

Cross-attention mechanism. One key innovation in TIM-GNN^x is the integration of a cross-attention mechanism. This mechanism takes as input the pre-computed Q-matrix (computed once) and the specific item $\overrightarrow{\gamma}$ for which a prediction is required, and its role is to dynamically extract the relevant information from the Q-matrix, in order to output a Q-vector specific to $\overrightarrow{\gamma}$.

Training complexity vs. query latency. The attention mechanism is trained end-to-end, trading training complexity for query latency. Indeed, on one hand, by leveraging the summarized information in the Q-matrix, we can make real-time predictions, bypassing the need for expensive graph projection computations at query time. On the other hand, the cross-attention mechanisms and the processing of the base projected graphs bring significant computational overhead and memory requirements. In practice, with limited resources, they may require simplifications in other aspects of the training architecture, potentially leading to a decrease in prediction quality[2]. Fig. 4.2 summarizes the training vs. query time operational differences between TIM-GNN and TIM-GNN^x.

[2]In our experimental environment, some of the features introduced in IM-GNN, such as self-edges and positional encoding, had to be silenced, along with a reduction in batch size, embedding dimensions and graph sampling size.

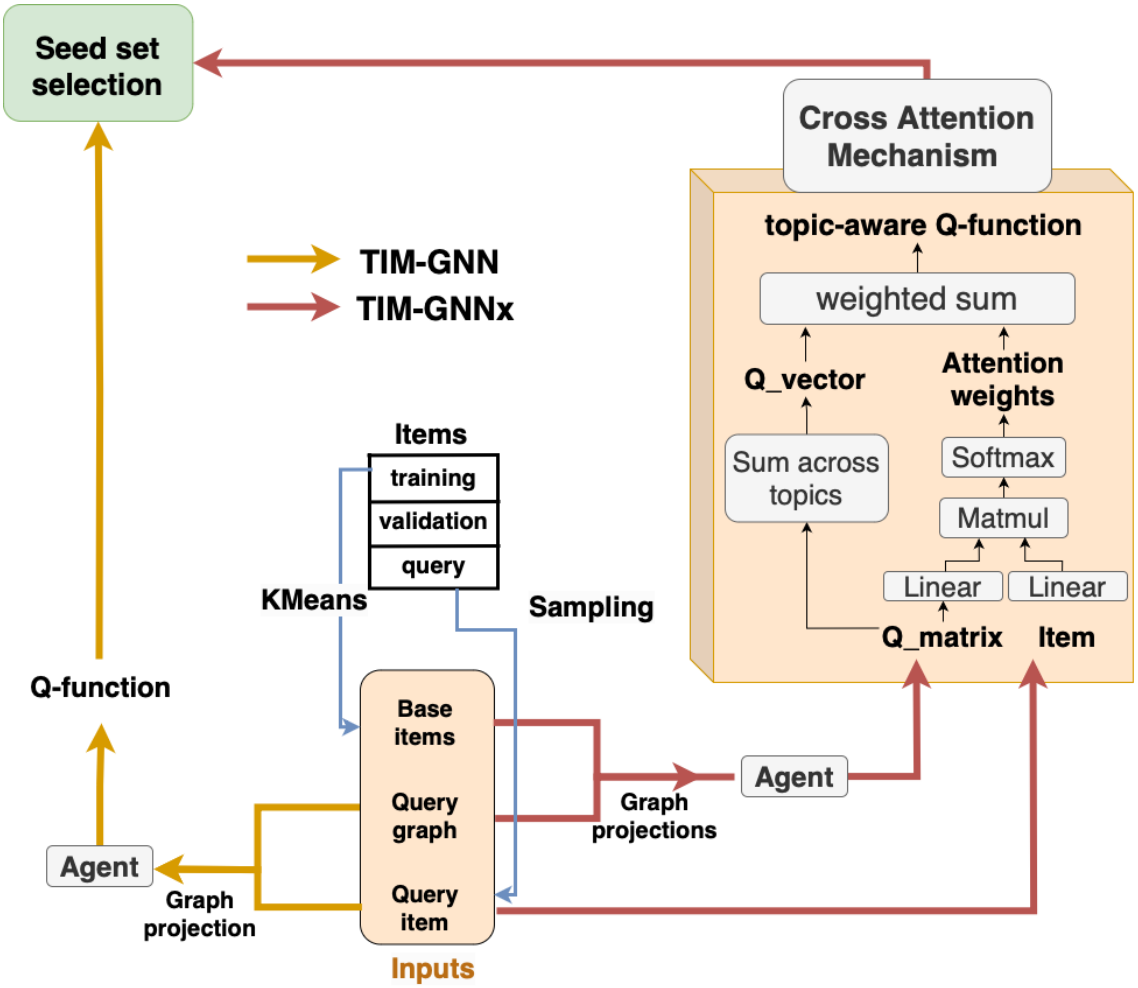


Figure 4.2: Comparison TIM-GNN vs TIM-GNNx.

4.4 Experiments

Hardware and implementation details. We run our experiments using HPC resources, mainly on (i) Octo-GPU SXM4 80 GB A100 (eight-GPU accelerated compute nodes Nvidia A100 SXM4 80 GB GPUs), with AMD Milan EPYC 7543 processors (32 cores at 2,80 GHz), so 64 cores and 512 GB per node, running RedHat v.8.4 and Slurm v.21.08.8. Our implementation is in JAX [106] and Jraph [107]. [3]

Datasets for S2V-DQN-IM We used the public datasets detailed in table 4.1 to replicate the results from [59]:

Dataset	nodes	edges	type
HepPh	34K	421K	Directed
DBLP	317K	1.05M	Undirected
LiveJournal	4.85M	69M	Directed
Orkut	3.07M	117.1M	Undirected

Table 4.1: Datasets in [59].

Datasets for IM-GNN : Data-based Sina Weibo We used a publicly available microblogging dataset from **Sina Weibo** [108]; Weibo is the main Chinese microblogging website. Starting from the

[3] The source code, data pre-processing and statistics, as well as the "glue" code and references to the respective implementations for SurvivalFactorization and baseline methods can be found at https://anonymous.4open.science/r/Submission-ecmlpkdd/.

cascades, we employed the edge weighting technique from [96] (also used in [97]).

$$p(u,v) = (A_{v2u}/A_v) \times e^{-\frac{\bar{D}}{\delta}} \tag{4.9}$$

where A_{v2u} is the number of times u reposted from v, A_v is the total number of posts of v, and $\bar{D}$ is the average time it takes for u to repost from v. On the resulting graph, we filtered out the edges with probability less than 0.01 and we kept its 5-core. The final graph has 32K nodes and 204K edges.

Datasets for TIM-GNN For the extraction of topic-aware diffusion graphs, we used three publicly available real-world datasets containing diffusion cascades, used in the study of [34]. They come from the following applications: **Weibo**, **Flixster** (a dataset from a social movie platform that allows users to find movies, connect with friends, discuss and rate movies), and **MemeTracker clustered** (a dataset that tracks phrases and quotes over online-news providers and blogs).

We applied the following pre-process /train / validate / test setup. We **filter** the cascade collection by removing small cascades, rare users, cascades with short content, etc, and we fix the number of topics, following [34]'s experimental findings. We apply the **survival factorization** model [34] to infer from the cascades matrices of *authoritativeness* A and *susceptibility* S for the nodes, as well as a matrix of word relevance w.r.t. topics. We get the **topic-aware probabilities** $p_{u,v}$ of our graph by a simple product of the vectors $A[u]$ and $S[v]$. We create the **items** (topic vectors), using the relevance matrix, by summing the topic distributions for each word in a cascade and normalizing to 1. Once we have the graph and items, we apply KMeans++ clustering to the items, and select the resulting 100 centroids as *representative items*. Among these representative items, we select the 5 items maximizing the Wasserstein distance between the probability distributions of the respective projected graphs, as our **evaluation items**. We also sample 10 items randomly for validation, while the remaining items are used for training. We detail further our pre-processing steps for these datasets below for each of the datasets.

For Sina Weibo, he dataset [108] is a network consisting of more than 1.7 million nodes and 0.4 billion edges, accompanied by a set of 300000 retweet cascades. We removed the cascades that have less than 10 activations and less than 2 words (in Chinese) and the users who belong to less than 7 cascades. We obtained a total of 139306 cascades, 61802 nodes, and 172539 words. After applying Survival Factorization for 6 topics, we construct our graph by retaining the corresponding edges from the original graph, removing all zero probabilities, and taking the largest weakly connected component. The final graph has 61330 nodes and 1980944 edges.

For Flixster, the raw dataset was cleaned by removing the cascades (movies) that have less than 20 occurences and the users that rated less than 15 movies. We obtained a total of 10739 cascades and 39294 users. After applying Survival Factorization for 10 topics, we construct our graph by retaining the corresponding edges from the original graph, removing all zero probabilities, and taking the largest weakly connected component. The final graph has 22428 nodes and 90961 edges.

For Memetracker, the raw dataset was cleaned by removing the cascades that have less than 10 activations and less than 10 words and the users who belong to less than 20 cascades. We obtained a total of 26474 cascades, 8299 nodes and 22193 words. After applying Survival Factorization for 8 topics, given the absence of an available MemeTracker graph, we proceed to define our graph by choosing edges that meet a specific probability threshold among all the possible combinations. We take the largest weakly connected component. The final graph has 8299 nodes and 380621 edges.

The probabilities are rescaled, to enhance diffusion in the graphs while keeping the distribution of probabilities consistent across different topics. For Sina Weibo, a simple translation to set the maximal probability to 0.9 was done. For MemeTracker and Flixster, we use the following formula

$$1 + (\frac{augmentation\_factor}{topic\_sum})$$

(factors 10^3, and 10^6 respectively). The final graphs are represented in Table 4.2.

Dataset	nodes	edges	Topics
Sina Weibo	61.3K	1.98M	6
Flixster	22.4K	90.9K	10
MemeTracker	8.2K	380K	8

Table 4.2: Datasets for topic-aware IM

Baseline methods. In our experimental analysis, we compared several IM methods, including both heuristic and learning-based approaches. The baselines, besides random selection, are as follows. **MaxDegree** selects the k nodes with highest outdegree. **MaxOutweight** selects the k nodes with the largest total weight of their outgoing edges; when the probabilities are uniform, it is equivalent to MaxDegree. **ToupleGDD** is the vanilla IM algorithm from [11]; since its result is non-deterministic, this operation is performed 20 times, for statistical relevance, and we take the minimum, mean, and maximum spread values. **GComb** is the vanilla IM algorithm from [61]. **DeepIM** is the approach of [62], solving vanilla IM using an autoencoder to compress seed sets, training the network to optimize the spread prediction from the reconstructed seed set. **S2V-DQN** is [11]'s implementation of a simple S2V-DQN adaptation for IM, trained on random graphs. **IMM** [48] is used as the state-of-the-art IM algorithm; its output can be seen as an optimal spread result, to which the heuristics and learning-based methods should get as close as possible.

Training. From a given diffusion network, we generate a set of 100 randomly sampled training graphs by BFS. In each training iteration, a new subgraph is sampled, with approx. 20% of the edges from the original network. This allows the model to continuously gather diverse samples of activation states and corresponding rewards, which are then stored in a buffer. Subsequent training phases utilize these buffered samples to update the model's weights.

Validation & testing. We generate another set of 10 graphs with an edge count of 30% of the original graph. We validate the model every 3rd training iteration by evaluating it on these held-out graphs and taking the mean of the spread as a measure of performance. We save the best weights of the model and replace them whenever a better version appears. Finally, the best model selected by validation can be evaluated at query time on the entire graph.

S2V-DQN-IM. Our first goal was first to reproduce the basic S2V-DQN framework adapted to IM, also called PIANO in [59]. We used the known IM benchmarking datasets **HepPH**, **DBLP**, **LiveJournal**, **Orkut** (all from the SNAP repository). We report here our results on HepPh, DBLP and LiveJournal (We faced an Out of Memory issue for Orkut dataset in our setup). As it is unclear from [59] what diffusion probabilities were used for these graphs, we considered either uniform (0.5) or trivalency (0.1, 0.01, 0.001) values.From Fig. 4.3, we can draw the following conclusions: (i) the performance of S2V-DQN-IM and ToupleGDD are close on uniform (only case where DeepIM also does well) and trivalency graphs, with a slightly better performance for S2V-DQN-IM, and (ii) simple heuristics like MaxDegree and MaxOutweight perform quite well.

From Fig. 4.4 and Fig. 4.5, we can draw the following conclusions: (i) except for DBLP trivalency, the performance of all algorithms and heuristics on both DBLP and LiveJournal datasets seems to stabilize around a certain spread value as the seed set size increases and exhibit similar performance, which is unusual and may be attributed to the large size of the datasets. (ii) On the DBLP trivalency model, GCOMB appears to outperform other algorithms, while IMM encounters out-of-memory errors on both DBLP and LiveJournal datasets. (iii) Simple heuristics like MaxDegree and MaxOutweight consistently perform well, particularly on the DBLP trivalency model.

IM-GNN. In datasets with uniform or trivalency probability, we have by design a high correlation between the nodes with high degree and those with high total out-weight, hence MaxDegree and MaxOutweight will show similar performance. In realistic diffusion graphs, built from cascades, i.e., with *data-based* diffusion probabilities, this is not necessarily the case. IM-GNN was evaluated on realistic diffusion graphs, using previously described **data-based Sina Weibo**.

Recall that IM-GNN stands for S2V-DQN-IM enhanced with attention mechanisms, positional encoding, and self-edges.

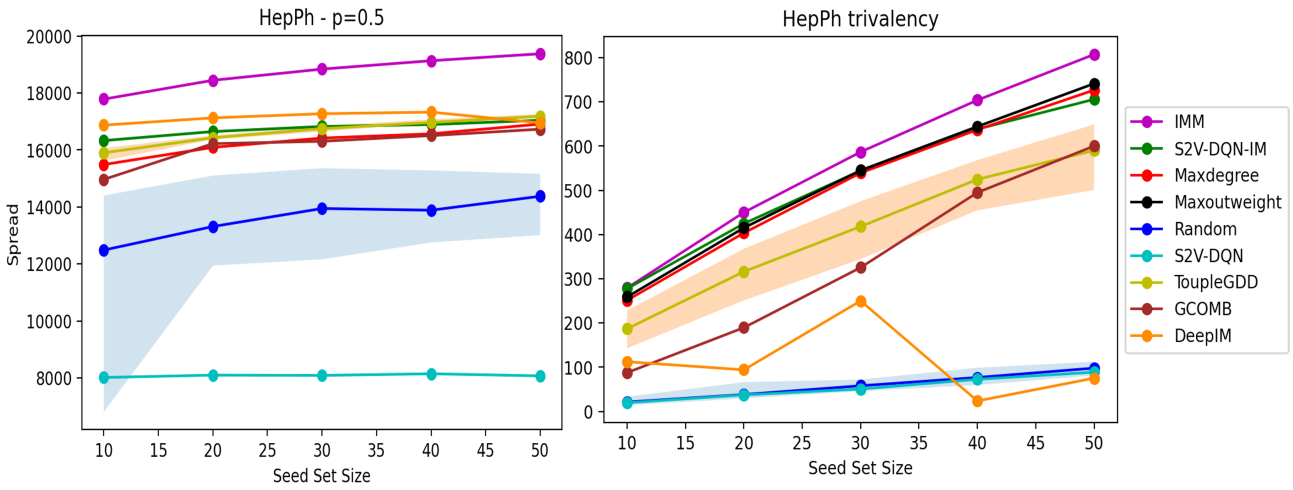


Figure 4.3: Results on Heph (uniform – p=0.5 left, and trivalency right).

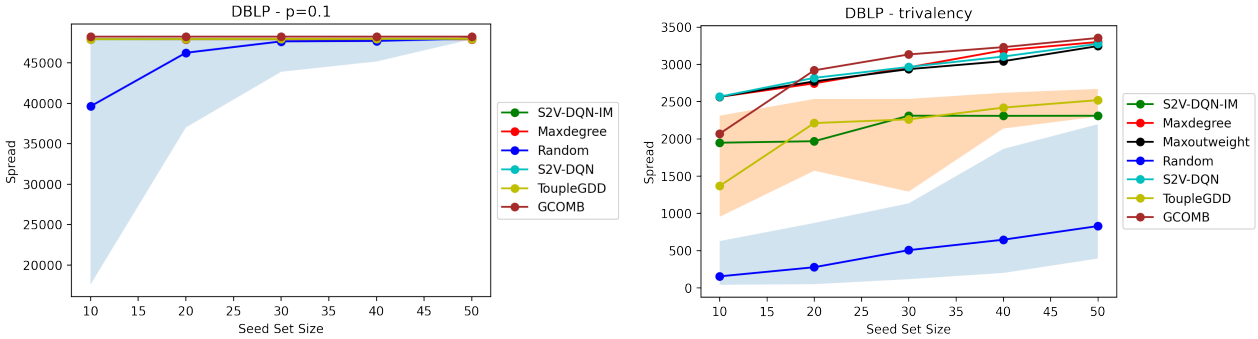


Figure 4.4: Results on DBLP (uniform – p=0.5 left, and trivalency right).

In an *ablation study*, we tested all the configurations on the Weibo graph (Fig. 4.6). The results show that (i) when IM-GNN has no attention, the magnetic Laplacian shows superior performance, while when combined with self-edges, it has a synergistic effect and proves to be very efficient, (ii) the combinatorial Laplacian also improves performance compared to the vanilla model, while random-walks positional encoding hurts the performance, (iii) when the attention mechanism is used, it increases the performance but reduces the impact of positional encoding (this may depend on the graph features, so we still consider the magnetic Laplacian and self-edges as useful model additions), (iv) as we compare the models with magnetic Laplacian and self-edges, with / without attention, with our baselines (Fig. 4.7), our performance with attention is now above the one of the heuristics (and GComb), comparable to ToupleGDD and, expectedly, below IMM (slightly). ToupleGDD outperforms S2V-DQN-IM on real graphs, perhaps due to its training on random graphs. DeepIM performs well (on comparison to the baselines) only on uniform probabilities, but not so well (below baselines) in general.

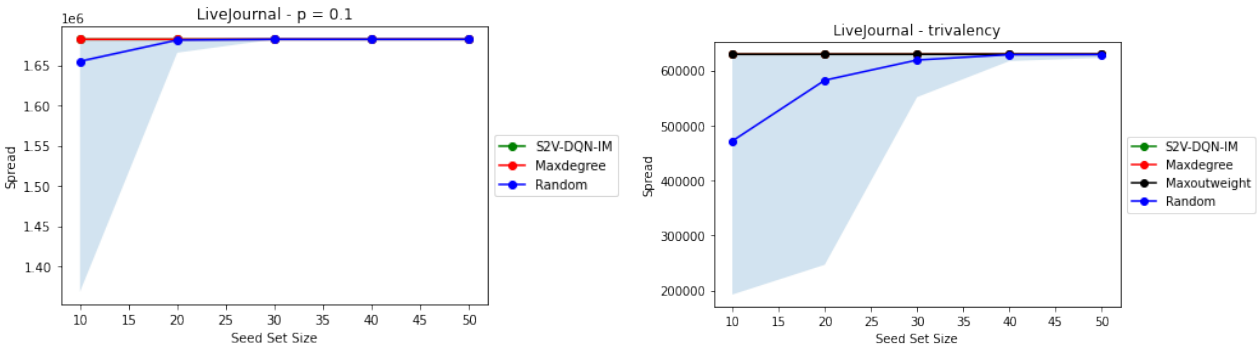


Figure 4.5: Results on LiveJournal (uniform – p=0.5 left, and trivalency right).

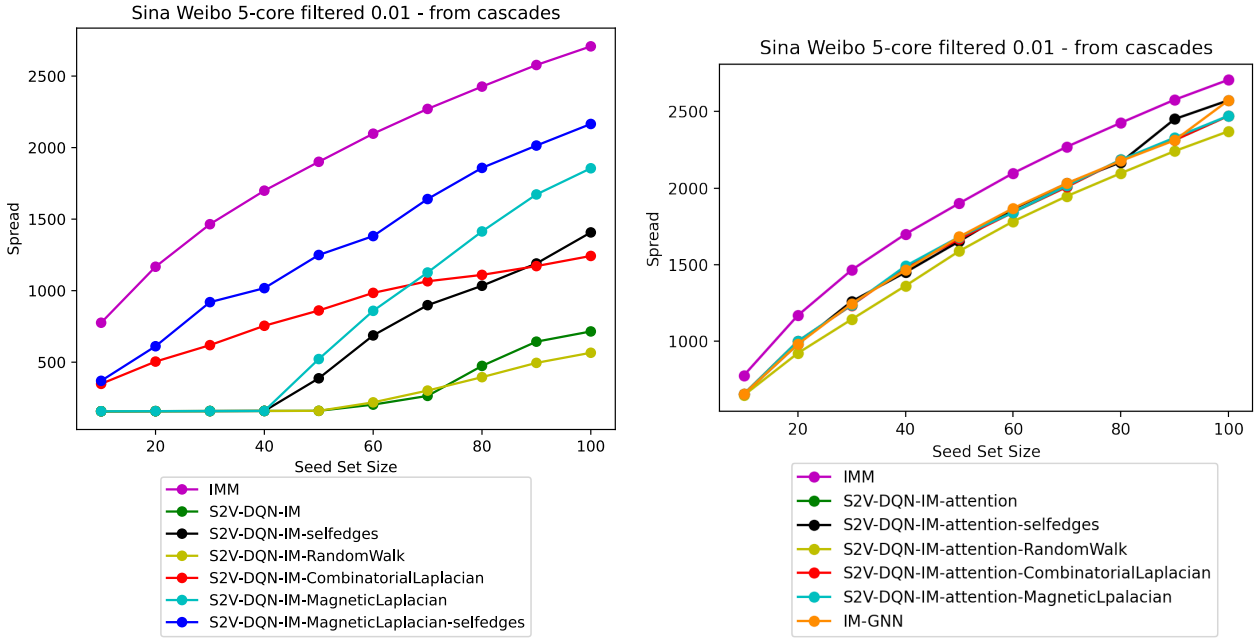


Figure 4.6: Spread obtained on Sina Weibo, with variants of S2V-DQN-IM (left) and with attention (right).

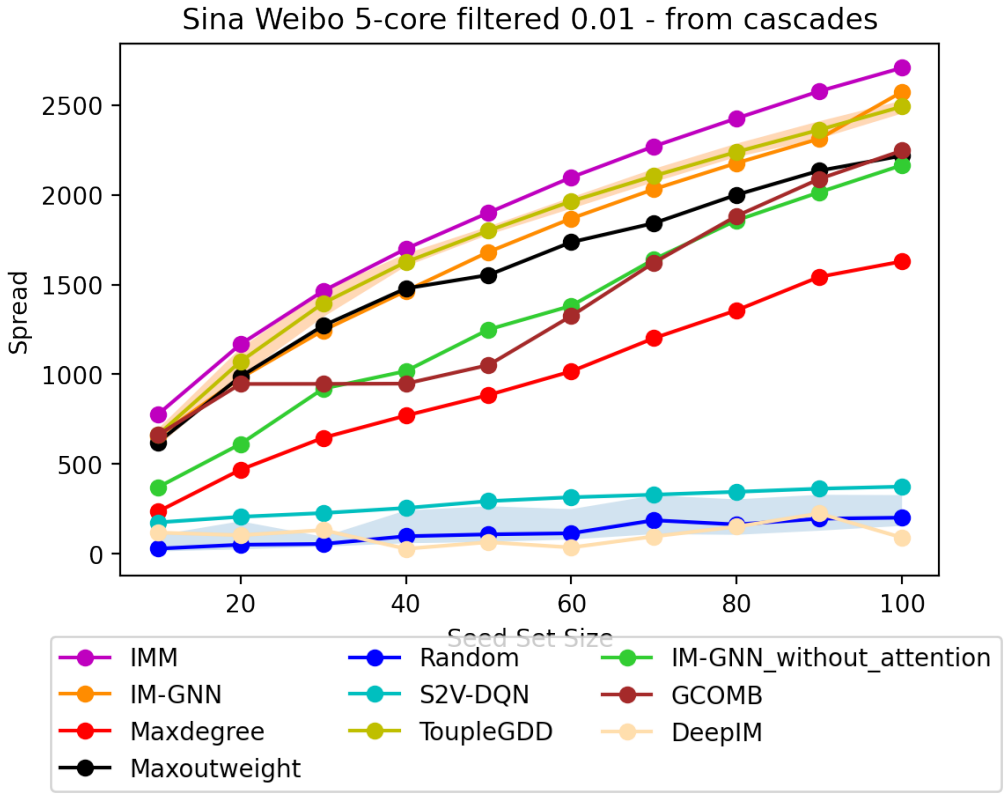


Figure 4.7: Spread obtained on Sina Weibo - comparison IM-GNN and baselines.

TIM-GNN and TIM-GNN^x Recall that, in terms of architecture, TIM-GNN enhances IM-GNN by following a topic-aware training approach, in order to learn across diverse topic-dependent diffusions.

We evaluate our methods on 15 projected graphs, using the 5 evaluation items on each of topic-aware graphs: **Sina Weibo**, **Flixster**, and **MemeTracker**. For each projected graph, we compare (i) the baselines, (ii) IM-GNN trained on the projected graphs, (iii) TIM-GNN trained directly in topic-aware manner, and (iv) the state-of-the-art INFLEX topic-aware model of [54]. Recall that INFLEX builds an index based on IMM results for 30 random training items, and then aggregates results from this index to answer any other topic-aware query.

We present the results for our two topic-aware approaches, TIM-GNN and TIM-GNN^x in error bar plots in Fig. 4.8 and Fig. 4.9. Individual plots for each of the 5 evaluation items and 3 topic-aware datasets are relegated to an appendix. We can make the following key observations: (i) **(Fig. 4.8 - left)** TIM-GNN shows similar performance compared to IM-GNN, competitive performance compared to IMM and INFLEX, and often superior performance compared to ToupleGDD and GComb, (ii) MaxOutweight performs surprisingly well on these datasets, (iii) **(Fig. 4.8 - right)** We observe that TIM-GNN^x generally maintains comparable performance levels to TIM-GNN. Notably, the KMeans++ variant of TIM-GNN^x demonstrates improved performance over the one-hot variant, with small advantage for the learnable version. It is important to note the performance deviation encountered by TIM-GNN^x on the Sina Weibo dataset. Given the size and complexity of this topic-aware graph, this is due to the additional training complexity and to our current training setting, leading to a loss of critical information necessary for the algorithm to effectively optimize its IM objective[4]; (iv) **(Fig. 4.9)** when we compare the query latency, we can see TIM-GNN is slower than INFLEX but much faster than ToupleGDD (also an S2V-DQN-based model). Finally, TIM-GNN^x is much faster at query time than TIM-GNN, with a 10x-20x speedup. Furthermore, unlike TIM-GNN, which needs a separate prediction for each seed set size k, TIM-GNN^x predicts the Q function just once, allowing for the selection of all seed nodes in a single step. Overall, this approach can yield query latency improvements of several orders of magnitude, especially when dealing with diverse queries.

[4]The model reduction we employed in this dataset was too drastic (embedding size 16 instead 32, batch size 24 instead of 32, graph sampling size 10% instead of 20%).

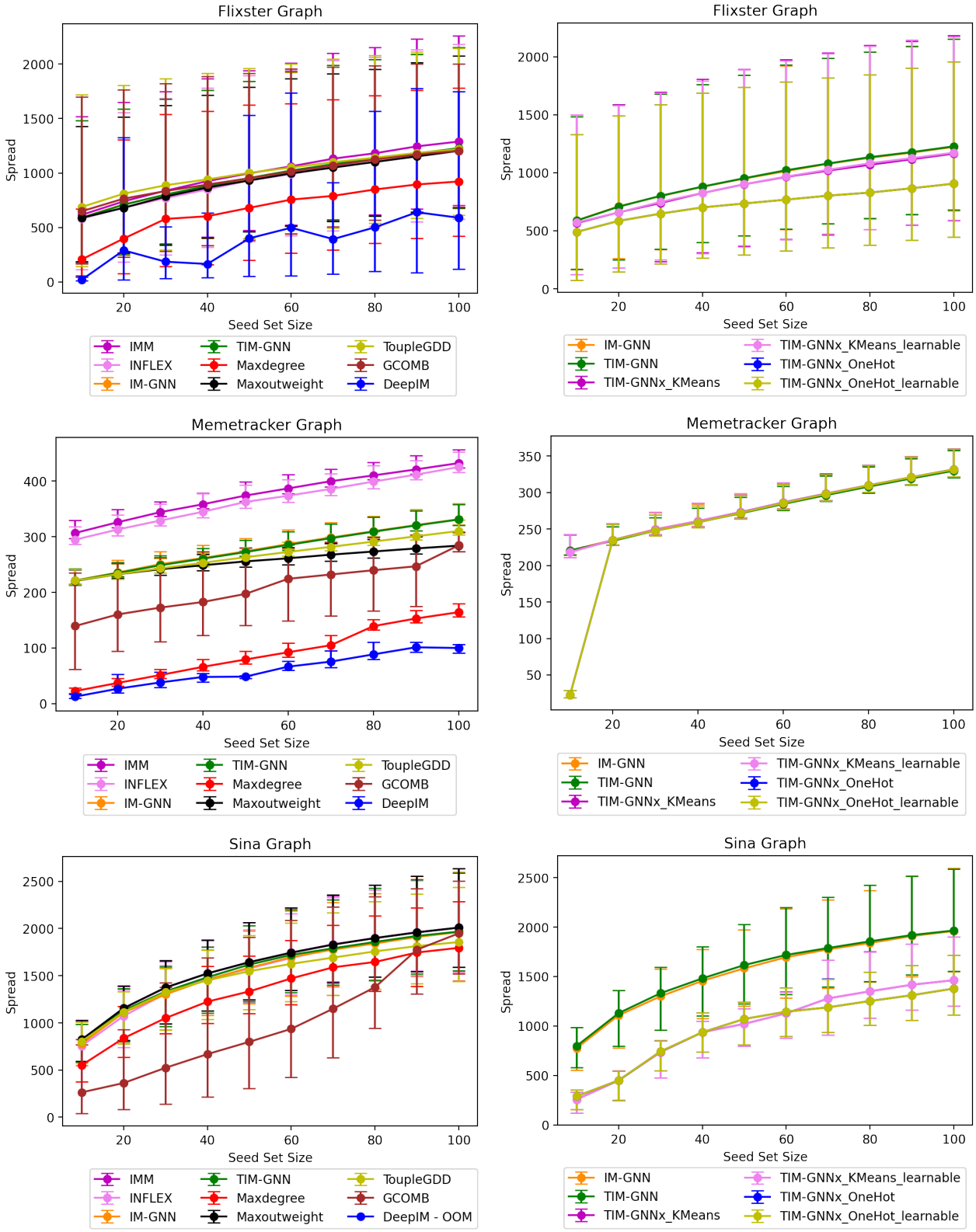


Figure 4.8: Results on Flixter (top row), MemeTracker (mid row), and Sina Weibo (bottom row) – Comparative results across five items – TIM-GNN vs. baselines (left) and variants of TIM-GNN^x (right).

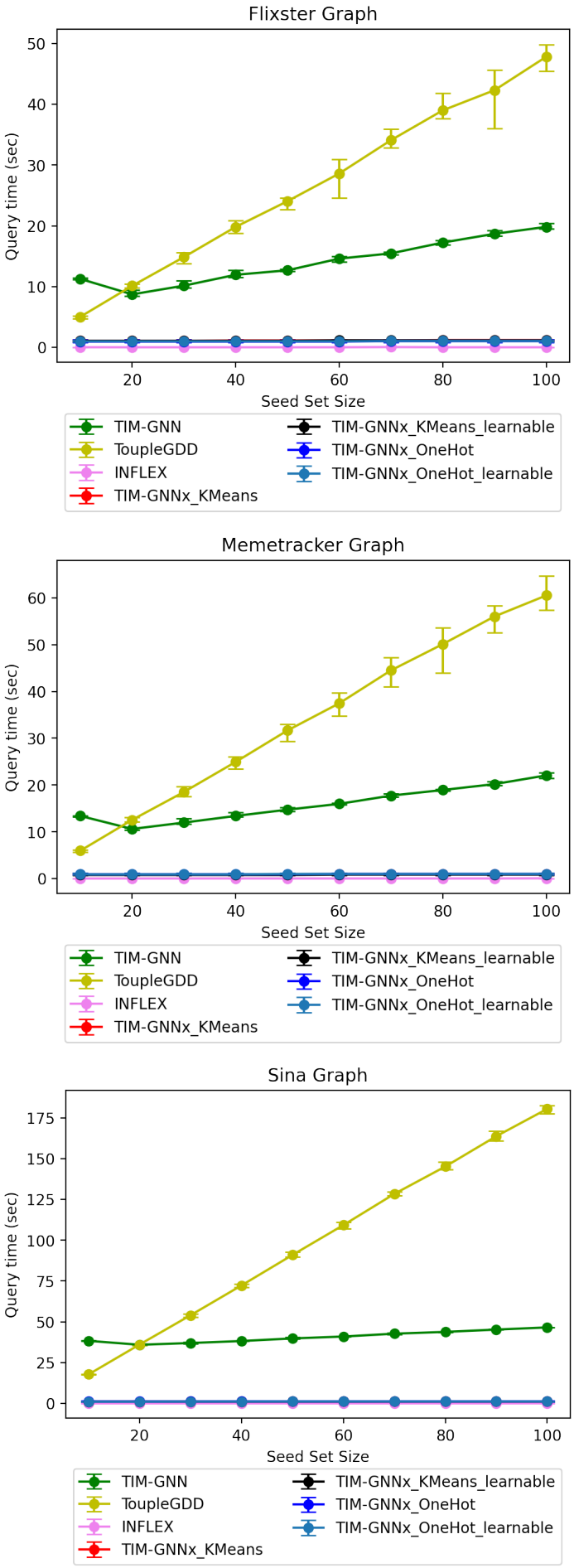


Figure 4.9: Results on Flixter (top row), MemeTracker (mid row), and Sina Weibo (bottom row) – Comparative results across five items – query time latency.

Robustness of TIM-GNN^x. The goal of this experiment is to analyze how TIM-GNN^x and INFLEX, the two low-latency topic-aware methods, may be affected by changes in diffusion probability values. Our thesis is that INFLEX is not adapted to dynamic diffusion graphs, since it only uses an index pre-constructed with results from an IM method such as IMM, instead of the actual graph. Therefore, whenever the graph changes, even slightly, these changes are taken into account by INFLEX only if the index is rebuilt, which is of course computationally expensive.

We perform this experiment on Sina Weibo (for item 2), Memetracker (for item 5) and Flixster (for item 3). We first evaluate the robustness of INFLEX and TIM-GNN^x when facing a limited, *single-node* change, as follows. For each item, we first project the item on the topic-aware input graph, then, we select the best node, n_1. For INFLEX, the best node is selected from the seed set for k=10, by choosing the node with most impact on the performance loss. For TIM-GNN^x, since this model provides a ranking, we just choose the node with highest Q-value. We then *perturb* outgoing probabilities of n_1, by setting them to a low ϵ value. Recall that, unless the index is reconstructed, INFLEX will return the same seed set, regardless of the perturbation. For INFLEX's performance, we will therefore evaluate the spread on the perturbed graph of a seed set obtained on the unperturbed one.

We also evaluate the robustness of INFLEX and TIM-GNN^x, for a larger scale, *multi-node* change, which can make the spread of INFLEX arbitrarily low, as follows. Instead of a single node, we select all the nodes from INFLEX's seed set for $k = 100$ and we perturb their outgoing diffusion probabilities to a very small value. The same is done for TIM-GNN^x 100 best nodes. As a golden standard, we evaluate the spread obtained by IMM on the perturbed graphs. We present the results in Figures 4.10-4.12, for one item per dataset (3 plots in total). Overall, we can observe that, for the single-node perturbation (denoted $1Per$), there is a decrease in performance for all algorithms, but the relative performance remains coherent with our previous results. Finally, when we perturb the seed set of INFLEX for $k = 100$ (denoted as $100Per$), we can observe that TIM-GNN^x maintains a good performance and stays competitive with IMM while by design INFLEX has no spread at all besides the seed set.

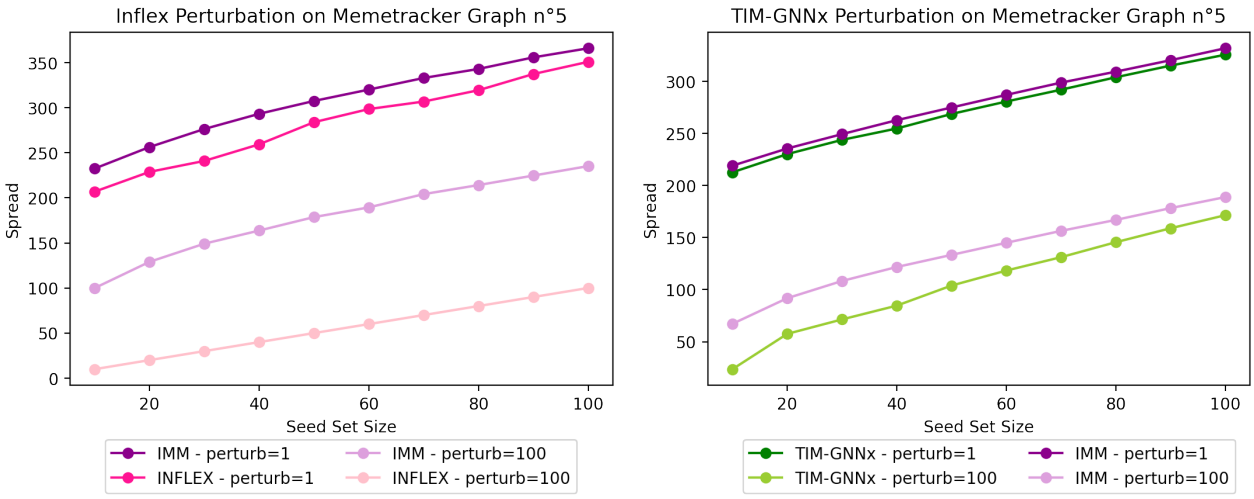


Figure 4.10: Perturbation – INFLEX left, TIM-GNNx right.

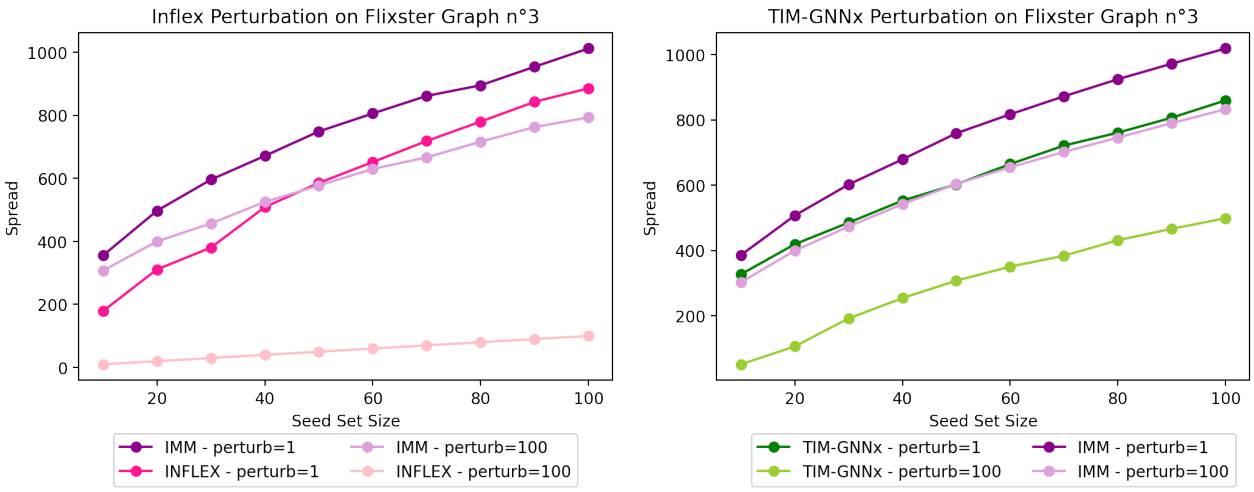


Figure 4.11: Perturbation – INFLEX left, TIM-GNN^x right.

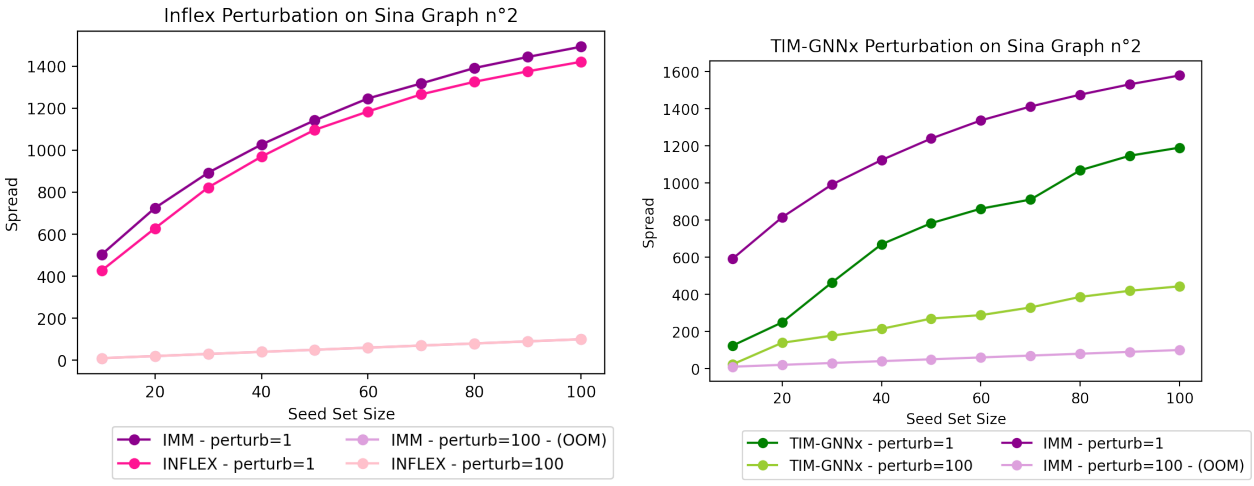


Figure 4.12: Perturbation – INFLEX left, TIM-GNN^x right.

Time comparison on topic-aware IM. Recall we can distinguish three key time components: training, pre-query processing, query latency. Training time is less critical whenever the model is robust, allowing for infrequent retraining. Pre-query time is also likely manageable, *if not required for each query* (unlike in all the topic-agnostic DRL methods when used in the topic-aware setting, i.e., by *projecting* each time the query topic vector on the topic-aware graph). While non-learning based IM methods like IMM don't have the first 2 stages, they do incur high query latency, which is the main motivation of DRL-based methods: predict marginal spread for faster response time, with a potential loss in spread effectiveness. Recall that Fig. 4.9, showed query latency; therein IMM was omitted, as it doesn't even fit the scale. Next, we describe *training time* and *memory footprint* for the various DRL-based models. GComb, TGDD, and DeepIM all require <10h (with some variations depending on the dataset) for training, while TIM-GNN (resp. TIM-GNN^x) <30h (resp. <35h). We stress that their larger training time is to be expected, as topic-awareness is integrated in training, but, importantly, this is mitigated by the robustness performance. All models have a <50GB average memory footprint.

4.5 Conclusion

We present in this thesis chapter DRL-based solutions for IM, building upon the S2V-DQN generic framework, which we progressively refine towards our main contributions, TIM-GNN and TIM-GNN^x, for topic-aware IM. The latter model learns by observing diverse topic-aware diffusions during training. While TIM-GNN performs well in terms of topic-aware spread effectiveness, its latency at query time remains rather high. To address this issue, in TIM-GNN^x, by using cross-attention mechanisms, we integrate the topic-aware dimension directly in the training process. This trades complexity at the training stage for query latency. We show TIM-GNN^x maintains comparable overall spread performance as its predecessor, while achieving a 10x-20x speed-up.

To our knowledge, TIM-GNN^x is the first to address the three critical challenges for the practical applicability of learning-based IM methods: predictive accuracy over a broad (topic-based) query space, low latency over large diffusion graphs, and robustness to graph changes, as demonstrated in our empirical evaluation on real-world data and diffusion graphs built from real cascades. By directly predicting topic-aware marginal spread gain, TIM-GNN^x avoids (i) expensive diffusion simulations (as IMM or similar methods), (ii) expensive topic / graph projection (as GComb, ToupleGDD, DeepIM, TIM-GNN), and (iii) expensive topic-aware indexing, which requires a static graph (as INFLEX).

Table 4.3 summarizes how these methods behave w.r.t. spread, query latency, robustness to graph changes, and topic-awareness.

	IMM	GComb	ToupleGDD	DeepIM	INFLEX	TIM-GNN	TIM-GNN^x
Spread	✓	✓	✓	✓	✓	✓	✓
Latency	×	×	×	×	✓	×	✓
Scale	×	✓	✓	✓	×	✓	✓
Robust	–	–	–	–	×	✓	✓
Topic-aware	–	–	–	–	✓	✓	✓

Table 4.3: Summary of methods behaviour.

Chapter 5

Few-Shot Link Prediction in Knowledge Graphs

5.1 Introduction

KGs have emerged as powerful tools for representing and organizing information in a structured manner, enabling efficient knowledge retrieval and reasoning. Link prediction, the task of inferring missing connections between entities within a KG, plays a crucial role in KG completion and powers various downstream applications.

Traditional link prediction methods have achieved remarkable success, but they often struggle when dealing with entities or relations that have limited data representation, a common issue in real-world knowledge graphs due to the long-tail distribution. This phenomenon, illustrated in Figure 5.1, describes the prevalence of entities (nodes) and relations (edges) with sparse connections or occurrences, making it difficult to discern meaningful patterns.

Few-shot link prediction tackles this challenge by enabling models to make accurate predictions with very few training examples. Few-shot learning techniques, such as meta-learning and transfer learning, offer promising avenues for addressing this problem by leveraging knowledge acquired from previous experiences to efficiently adapt to new tasks with minimal data.

This chapter delves into the realm of few-shot link prediction in knowledge graphs, exploring the limitations of existing approaches and proposing novel methods to enhance predictive capabilities. We begin by examining the challenges within the AIDA project, where the goal is to develop an AI-powered platform for improving organizational models for companies. Subsequently, we analyze the MetaR framework, a prominent non-GNN based meta-learning approach for few-shot link prediction, highlighting its strengths and weaknesses. Finally, we investigate the potential of integrating the PathCon method, which leverages knowledge graph structure for prediction, with the MetaR framework to further improve performance and address the limitations of existing techniques.

5.2 Problem Definition in AIDA Project

In today's complex business environments, ensuring accuracy and completeness within organizational models is critical for efficiency. Missing data in these models can hinder decision-making and the optimal deployment of resources. Currently, manual methods for filling in these gaps are often time-consuming and prone to error. Our task within the AIDA project was to develop an AI-powered platform to address the challenge of improving organizational models for companies (Figure 5.2).

The core of the problem lies in the efficient retrieval and integration of missing information. This requires an AI system capable of swiftly pinpointing gaps within the organization model and then intelligently sourcing the correct information. Such sourcing could involve searching internal enterprise databases or engaging with experts to provide data customized to the specific company.

We propose a novel approach to improve organizational models by leveraging the power of few-shot link prediction in knowledge graphs. This choice is motivated by two key factors: the suitability of knowledge graphs for representing business models and the limitations of data availability in our project.

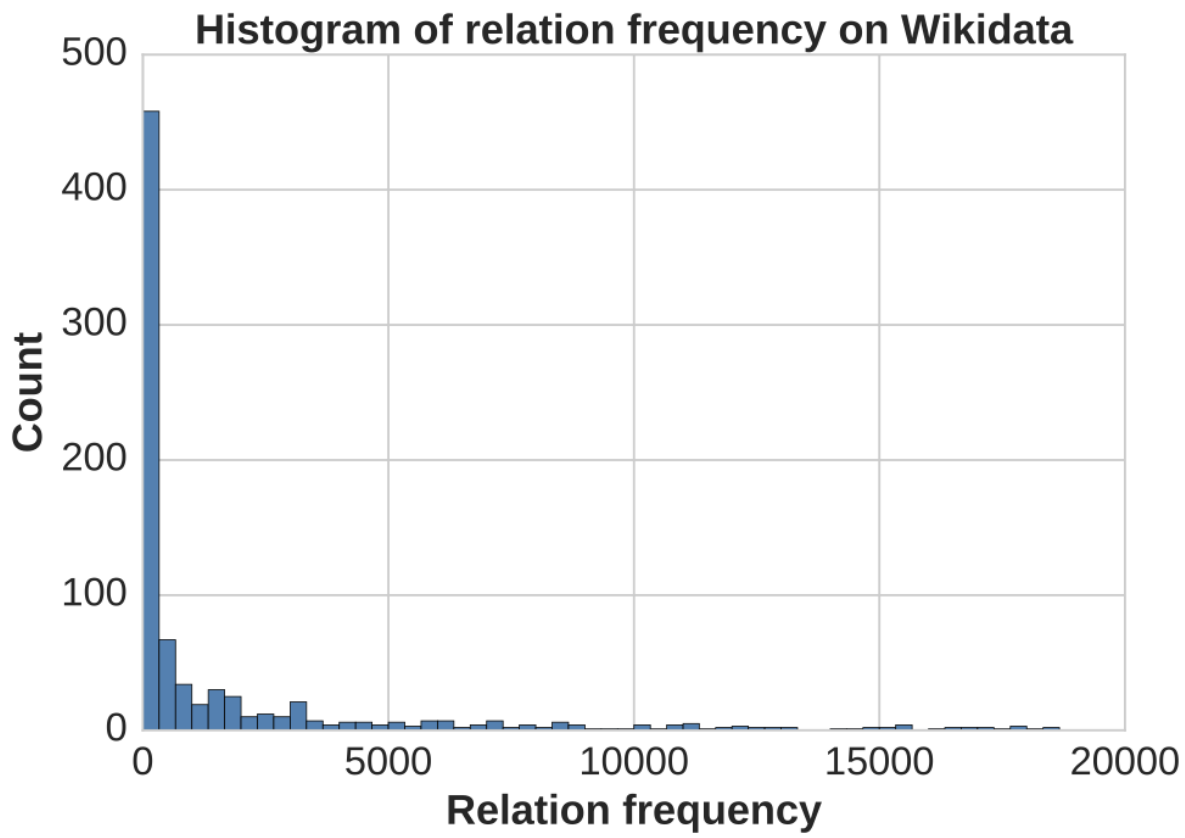


Figure 5.1: Illustration of the long-tail distribution: In this histogram of relation frequencies in Wikidata [109], there are a large portion of relations that only have a few triples; figure from [84].

Why Knowledge Graphs?

Knowledge graphs provide a structured and intuitive way to represent the complex relationships within an organization, like the one depicted in figure 5.2. By depicting entities like “RH Accueil”, “Paie”, and “Spécialiste IT” as nodes, and their relationships as edges, we can capture the organizational structure effectively. For the hierarchical aspect, we replaced using boxes to depict departments like “Service Administratif”, “Service Support”, and “Service des Opérations” by employing entities and relationships. For example, “Paie”, “RH Accueil”, and “RH Contract” would be individual entities, linked to a “Service Administratif” entity through a “belongs_to” relationship, providing a more nuanced and accurate picture of the organization, fully represented as a knowledge graph.

The Need for Few-Shot Learning

The nature of our project presented a significant challenge: the absence of readily available data on organizational structures. Gathering comprehensive datasets for such purposes can be time-consuming and expensive. To overcome this hurdle, we turned to few-shot learning. This technique allows our models to learn and predict missing links (relationships) between entities using only a small number of examples. This is particularly valuable in our scenario where data is scarce, enabling us to still extract meaningful insights and enhance our understanding of the organization.

Our solution

This thesis examines the feasibility of state-of-the-art few-shot link prediction techniques to accurately and efficiently infer missing relationships within organizational models. We focus on developing a user-friendly API that enables loading business organizational model files for predictions, employing an adapted version of MetaR. The API also facilitates necessary retraining when new, unseen entities

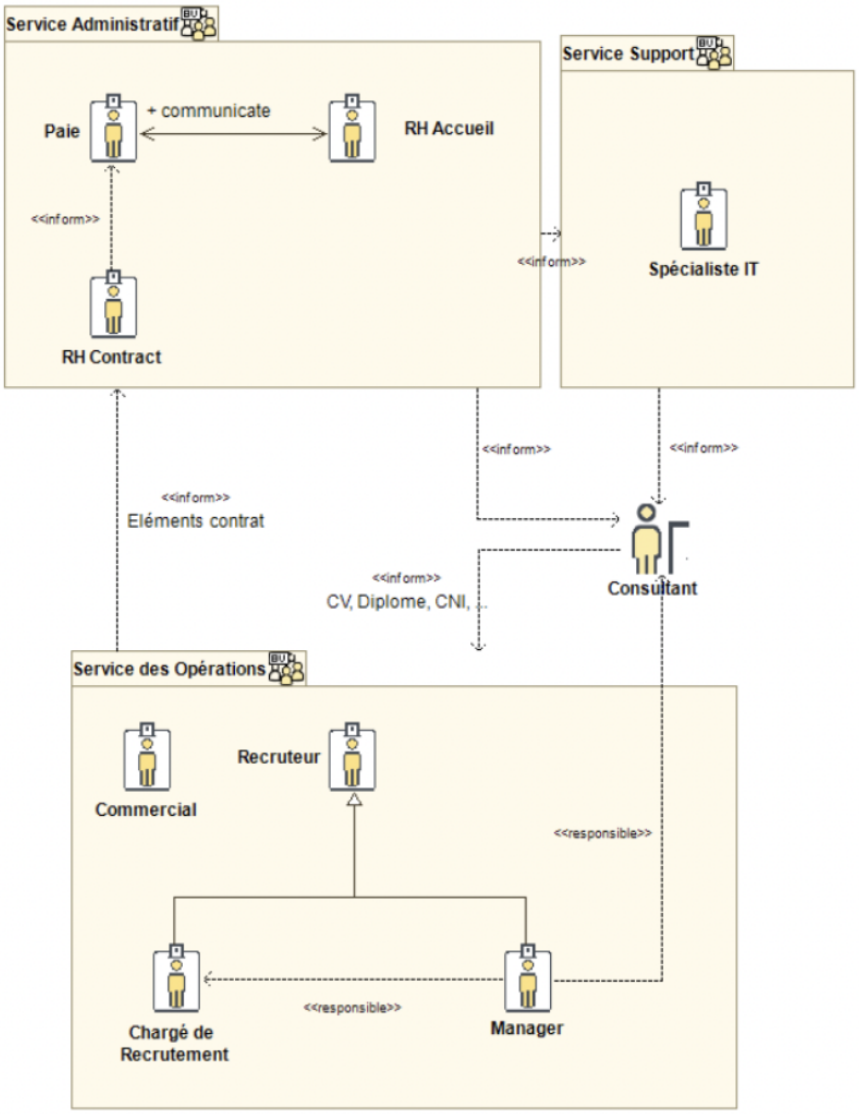


Figure 5.2: Example of organization diagram, from project AIDA.

are introduced into the models. A detailed discussion of our work will be presented in subsequent sections.

5.3 The MetaR framework : applying MAML to KGs

The Meta Relational Learning (MetaR) framework [35] is designed to address the challenge of few-shot link prediction taks in KGs by leveraging a Meta-Learning framework. This task involves predicting new triples about a relation r with only a few observed instances. MetaR achieves this by extracting and transferring **relation-specific meta information** from a support set to a query set, enabling the model to learn and generalize effectively from limited data.

MetaR utilizes two types of relation-specific meta information. **Relation Meta** represents the underlying connection between head and tail entities for a specific relation. It is extracted from the support set and transferred to the query set to guide predictions. **Gradient Meta**, on the other hand, captures the loss gradient of the relation meta, indicating how it should be adjusted to minimize the loss function. This enables rapid adaptation of the relation meta within a task based on the observed instances.

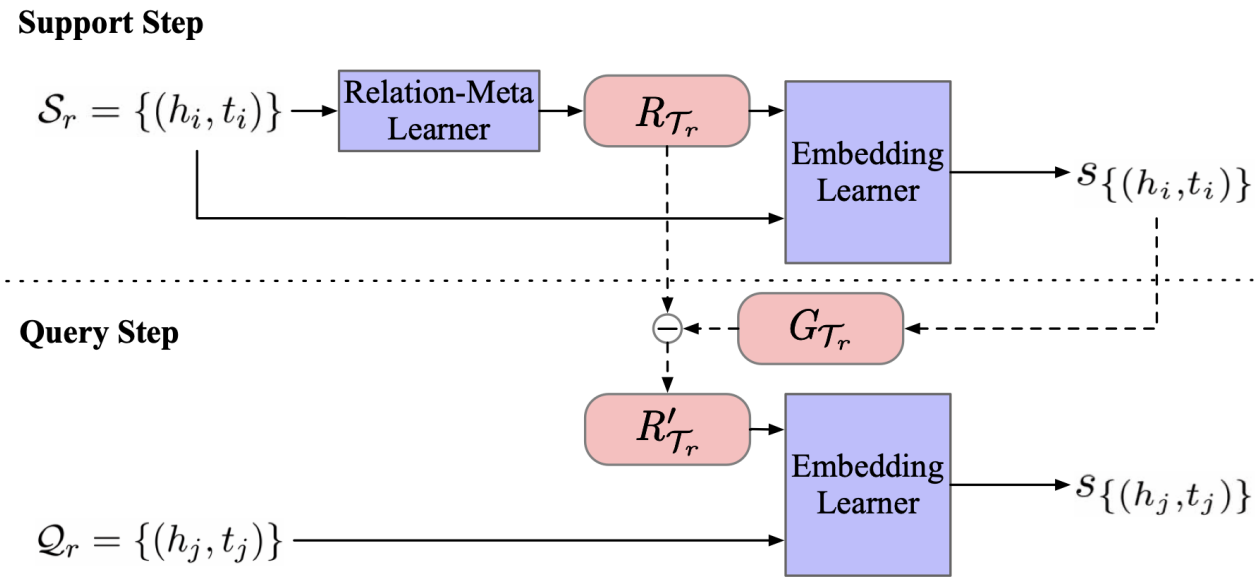


Figure 5.3: Overview of MetaR from [35]. $T_r = \{S_r, Q_r\}$, R_{T_r} and R'_{T_r} represent relation meta and updated relation meta, and G_{T_r} represents gradient meta.

Framework Components

MetaR is comprised of two main modules:

- **Relation-Meta Learner**: This module extracts relation meta from the support set. In a K-shot link prediction task, $T_r = \{S_r, Q_r\}$, the relation meta is learned from a support set S_r, which contains K entity pairs. For each entity pair (h_i, t_i) in the support set, the relation meta is computed using a neural network R on the concatenated embeddings of these entities. The final relation meta R_{T_r} for the task is obtained by averaging the individual entity-pair specific relation meta:

$$R_{T_r} = \frac{\sum_{i=1}^{K} R(h_i, t_i)}{K}. \tag{5.1}$$

- **Embedding Learner**: This module evaluates the truth value of triples in the support and query sets using entity embeddings and relation meta. Inspired by TransE [82], the score function for an entity pair (h_i, t_i) under relation meta R_{T_r} is defined as:

$$s(h_i, t_i) = \|h_i + R_{T_r} - t_i\|, \tag{5.2}$$

  where $\|x\|$ denotes the L2 norm of vector x.

The loss function employs a contrastive approach. In essence, it aims to minimize the distance between the embeddings of actual entity pairs associated with the relation and maximize the distance for randomly corrupted triples (negative samples). These negative samples are created by replacing either the head or tail entity with a randomly chosen entity from the KG. This process allows the model to differentiate genuine relationships from spurious connections.

Specifically, the loss function for the support set S_r is calculated as:

$$L(S_r) = \sum_{(h_i,t_i)\in S_r} [\gamma + s(h_i, t_i) - s(h_i, t'_i)]_+, \tag{5.3}$$

where $[\cdot]_+$ represents the positive part, γ is the margin, and $s(h_i, t'_i)$ is the score for a negative sample (h_i, t'_i).

The gradient meta G_{T_r} is obtained by calculating the gradient of the loss with respect to the relation meta:

$$G_{T_r} = \nabla_{R_{T_r}} L(S_r). \tag{5.4}$$

The relation meta is then updated using the gradient meta:

$$R'_{T_r} = R_{T_r} - \beta G_{T_r}, \tag{5.5}$$

where β is the step size. Finally, the updated relation meta R'_{T_r} is used to score triples corresponding to the same relation in the query set Q_r and compute the loss for training.

Training Objective

The overall training objective of MetaR is to minimize the sum of query set losses across all tasks in a minibatch:

$$L = \sum_{(S_r, Q_r) \in T_{train}} L(Q_r), \tag{5.6}$$

where T_{train} represents the set of training tasks, and $L(Q_r)$ is the loss function for the query set Q_r of a specific task T_r. The loss function $L(Q_r)$ is calculated similarly to $L(S_r)$, using the updated relation meta R'_{T_r} to score the triples in the query set.

This objective function ensures that the model learns to predict new triples accurately for unseen relations based on limited support set examples, while simultaneously optimizing the overall knowledge representation for all relations in the training set.

5.4 A functional API for business organizational model completion

Our approach centers on harnessing few-shot link prediction, specifically utilizing the MetaR framework, to complete and enhance organizational models. In collaboration with Softeam, we developed a small dataset of French organizational models. This dataset was crucial for pre-training our model and evaluating the quality of its recommendations.

The model first calculates embeddings for various entities using a powerful language model like Camembert. While entity embeddings can be learned directly, utilizing an LLM significantly enhances the model's accuracy. We leverage the MetaR framework, which employs an encoder-decoder paradigm to learn representations of relationships between these entities. MetaR's gradient-based meta-learning approach enables it to learn effectively from the limited examples provided by our dataset, transferring knowledge from support samples to queries.

This way, our model is able to score and rank potential relationships represented as triplets (subject-predicate-object). This allows us to identify the most probable connections based on their scores. To generate recommendations, we rank triplets absent from the original knowledge graph and suggest those with higher rankings than the ground truth triplets.

5.5 Overcoming MetaR's Limitations with PathCon

5.5.1 Limits of MetaR for Few-Shot Link Prediction

The MetaR model [35] has been a prominent approach for few-shot link prediction in knowledge graphs. However, recent studies have shed light on its limitations and inherent biases. The following delves into the constraints of MetaR, drawing insights from the analysis presented in [110].

First, MetaR, along with other few-shot link prediction models, predominantly leverages **positional information** about entities rather than capturing the underlying **structural information** or logical patterns of relations. This bias arises from the formulation of the few-shot task itself, where models receive a limited number of randomly sampled support set examples. Consequently, MetaR primarily exploits **coarse-grained signals** such as community structures within the knowledge graph.

It struggles to discern **fine-grained relational semantics**, such as symmetry or transitivity, which require a more comprehensive understanding of the logical patterns governing the relations.

MetaR Equation Analysis: The core of MetaR lies in learning a function, *RelLearner*, that maps a support set S_i characterizing relation r_i to a low-dimensional embedding:

$$r_i = RelLearner(\{(E(h_k), E(t_k))\}_{k=1}^{K}) \tag{5.7}$$

where E is an entity encoder and (h_k, r_i, t_k) represents a support set triple. This relation embedding r_i is then updated using the gradient of the support set loss $L(S_i)$:

$$r_i' = r_i - \eta \nabla_{r_i} L(S) \tag{5.8}$$

While this meta-gradient update mechanism provides some relation-specific information, it is insufficient for capturing complex logical patterns, especially when the support set is limited in size and diversity.

In conclusion, the MetaR model, while pioneering in the domain of few-shot link prediction, exhibits limitations in its ability to learn and exploit fine-grained relational semantics. Its reliance on positional information and the constraints imposed by support set sampling hinder its effectiveness in capturing the underlying logical patterns of relations, particularly in scenarios with limited support set examples.

5.5.2 PathCon : Exploiting Knowledge Graph Structure for Prediction

PathCon is a novel knowledge graph completion model that leverages the inherent structure of knowledge graphs for enhanced prediction accuracy. Unlike traditional embedding-based KGC models, PathCon operates solely on edge features (relation types), eliminating the need for entity IDs. This makes PathCon ideal for **inductive learning** i.e handling unseen entities during inference, enabling generalization to new data. Moreover, it also allows for **explainability**, offering insights into predictions by capturing correlations among relation types.

PathCon achieves this by modeling two crucial aspects of the knowledge graph structure: Relational context and relational paths.

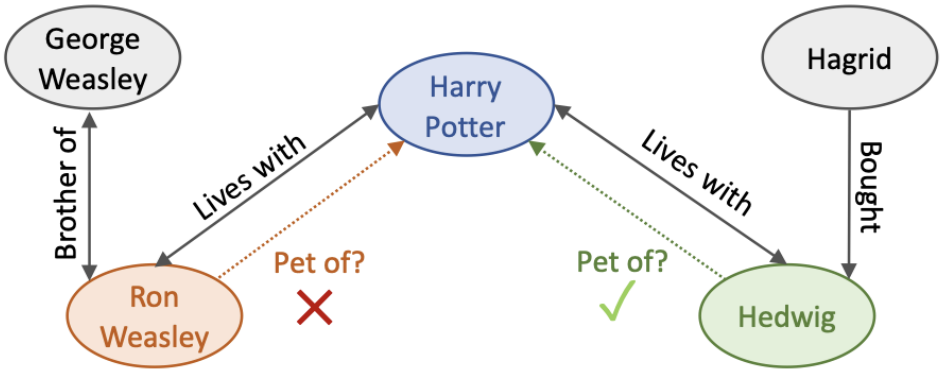


Figure 5.4: Importance of relational context of an entity, from [2].

Relational context captures the nature of entities by considering their neighboring relations. PathCon utilizes alternate relational message passing to aggregate information from multi-hop neighboring edges. This framework iteratively updates edge representations by exchanging messages between nodes and edges:

- **Edge to node aggregation:**

$$m_v^i = \sum_{e \in N(v)} s_e^i \tag{5.9}$$

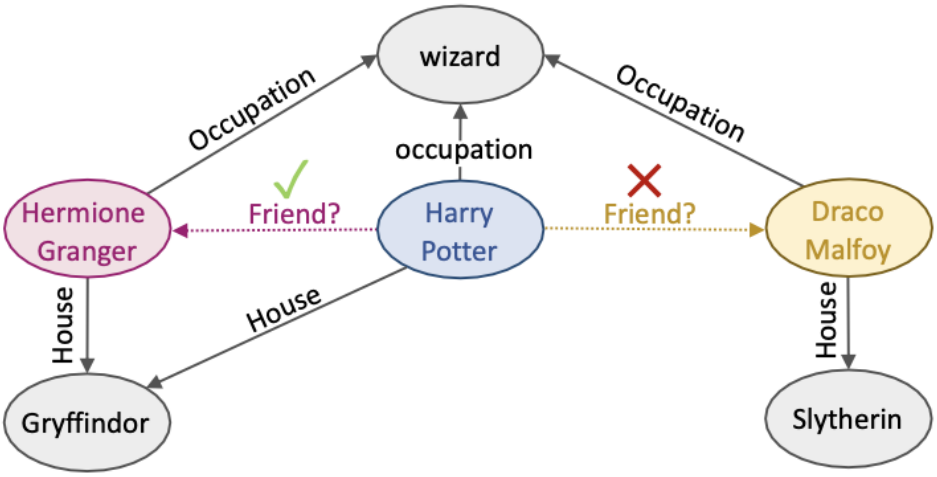


Figure 5.5: Importance of relational paths between entities, from [2].

- **Edge update:**

$$s_e^{i+1} = \sigma([m_v^i, m_u^i, s_e^i] \cdot W_i + b_i), \quad v, u \in N(e) \tag{5.10}$$

In these equations, m_v^i represents the message stored at node v, and s_e^i represents the hidden state of edge e, both at iteration i. $N(v)$ denotes the set of neighboring edges of node v, while $N(e)$ denotes the set of endpoint nodes of edge e. W_i and b_i are the learnable transformation matrix and bias, respectively, used at iteration i. $\sigma(\cdot)$ represents a nonlinear activation function, and $[\cdot]$ denotes the concatenation function. Finally, $s_e^0 = x_e$ is the initial feature of edge e, which can be taken as the one-hot identity vector of the relation type associated with edge e.

After K iterations, the final messages m_h^{K-1} and m_t^{K-1} represent the context of head and tail entities, respectively, and are combined to obtain the context representation of the entity pair (h, t):

$$s^{(h,t)} = \sigma([m_h^{K-1}, m_t^{K-1}] \cdot W_{K-1} + b_{K-1}) \tag{5.11}$$

Relational paths capture the relative positions of entities by considering the paths connecting them in the knowledge graph. PathCon identifies all relational paths between head and tail entities, ignoring node identities, and assigns each path an embedding vector. An attention mechanism is then employed to selectively aggregate these path representations based on the context information:

$$\alpha_P = \frac{\exp(s_P^\top s^{(h,t)})}{\sum_{P \in P_{h \to t}} \exp(s_P^\top s^{(h,t)})} \tag{5.12}$$

$$s_{h \to t} = \sum_{P \in P_{h \to t}} \alpha_P s_P \tag{5.13}$$

where:

$P_{h \to t}$ is the set of all paths from h to t. α_P is the attention weight of path P. s_P is the embedding vector of path P. $s_{h \to t}$ is the aggregated representation of relational paths.

The context and path representations are combined to predict the relation between head and tail entities:

$$p(r|h, t) = \text{SoftMax}(s^{(h,t)} + s_{h \to t}) \tag{5.14}$$

This allows PathCon to exploit both local neighborhood information and global connectivity patterns within the knowledge graph for accurate and interpretable predictions.

Variants of PathCon

PathCon can be implemented with several design alternatives. When modeling relational paths, PathCon can represent them using either a dedicated embedding vector for each unique path or by encoding the sequence of relations in a path using a Recurrent Neural Network (RNN). Additionally, the aggregation of multiple paths between an entity pair can be done either through a simple mean operation or through an attention mechanism that leverages the context representation of the entity pair to weight the importance of each path.

In our setup, we mostly use the RNN variant since we observed similar performance with embedding vectors but much shorter training time.

5.6 A Hybrid Approach for Few-Shot Link Prediction

Building upon the strengths of both PathCon and MetaR, we developed a hybrid model that aimed to address the limitations of each individual model and improve the overall performance of few-shot link prediction in knowledge graphs. This hybrid approach sought to leverage PathCon's ability to capture relational context and structural information, alongside MetaR's meta-learning capabilities for rapid adaptation to new relations with limited data.

To formally represent the integration of PathCon and MetaR for link prediction, we define the following notations:

- $\mathbf{R_{PathCon}} \in \mathbb{R}^{B \times d}$: The tensor of PathCon relation embeddings, where B is the batch size (B different relations) and d is the embedding dimension.
- $\mathbf{R_{MetaR}} \in \mathbb{R}^{B \times d}$: The tensor of MetaR relation embeddings.
- $\mathbf{R_{base}} \in \mathbb{R}^{N \times d}$: The base relation embeddings matrix, where N is the total number of relations. These are learnable embeddings.
- $\mathbf{E_{support}} \in \mathbb{R}^{B \times |S| \times N_e \times d}$: The tensor of support set entity embeddings, where $|S|$ is the size of the support set and N_e is the number of entities (N_e=2, for head and tail entities).

$\mathbf{E_{support}}$ is sampled from a learnable entity embeddings tensor $\mathbf{E_{base}} \in \mathbb{R}^{M \times d}$, where M is the total number of entities. It is important to note that by incorporating PathCon into the MetaR framework, the model is no longer inductive since we need the entities ids to select their embeddings from $\mathbf{E_{base}}$ which is initialized at the beginning of the learning process.

Integration Strategies

We explored two main avenues for integrating PathCon with MetaR: *adapting PathCon to the MetaR framework*, making it few-shot link prediction capable, and *combining the newly adapted PathCon with the MetaR model* for enhanced performance.

Adapting PathCon to the MetaR framework We employed PathCon to analyze the knowledge graph structure and generate a probability distribution $P_{PathCon}(r|h,t)$ over potential relations given head entity h and tail entity t, which is then used to weight the base relation embeddings $\mathbf{R_{base}}$ to obtain the PathCon embeddings $\mathbf{R_{PathCon}}$:

$$\mathbf{R_{PathCon}} = P_{PathCon}(r|h,t) \cdot \mathbf{R_{base}}$$

The probability distribution $P(r|h,t)$ has dimensions (B, N), allowing it to be directly multiplied with the base relation embeddings. This weighting scheme, also known as soft selection, is used instead of selecting the relation with the maximum probability to ensure gradient flow during training. Ensuring gradient flow is essential for effectively training the model, as it is the mechanism by which errors are backpropagated and the model parameters are updated during optimization. Without gradient flow, learning is impossible.

Combining PathCon with the MetaR model Several methods were explored to effectively integrate the relation embeddings generated by PathCon with MetaR relation embeddings:

1. **Cross-Attention Mechanism:** This method involved introducing an attention mechanism that dynamically weighs the contributions of PathCon and MetaR during the prediction process. The attention mechanism takes into account the specific query and the knowledge graph context to determine the relative importance of each model's output.

$$\begin{aligned}
\mathbf{Q} &= \mathbf{R_{PathCon}} \cdot \mathbf{W_q} \in \mathbb{R}^{B\times d} \\
\mathbf{K} &= \mathbf{R_{MetaR}} \cdot \mathbf{W_k} \in \mathbb{R}^{B\times d} \\
\mathbf{V} &= \mathbf{R_{MetaR}} \cdot \mathbf{W_v} \in \mathbb{R}^{B\times d} \\
\mathbf{S} &= \frac{\mathbf{QK^T}}{\sqrt{d}} \in \mathbb{R}^{B\times B} \\
\mathbf{A} &= softmax(\mathbf{S}) \in \mathbb{R}^{B\times B} \\
\mathbf{R_{attn}} &= \mathbf{AV} \in \mathbb{R}^{B\times d}
\end{aligned}$$

   where $\mathbf{W_q}, \mathbf{W_k}, \mathbf{W_v} \in \mathbb{R}^{d\times d}$ are the weight matrices for query, key, and value in the attention mechanism.

   PathCon and MetaR outputs are projected to obtain query, key, and value vectors. Attention scores are calculated using scaled dot-product attention. These scores are then normalized to obtain attention weights, which are used to compute a weighted sum of the value vectors.

   The final relation representation is the attention-weighted MetaR output $\mathbf{R_{attn}}$ which is used to predict the most likely relation between entities h and t.

2. **MetaR Alignment:** This strategy involved modifying the MetaR model to better align with PathCon's relational reasoning. Specifically, we integrated PathCon embeddings directly into MetaR's relation learning process. Recall that MetaR typically learns a relation representation r_i (equation 5.7) from the embeddings of head and tail entities in the support set, as shown in the following:

$$r_i = RelLearner((E(h_k), E(t_k))_{k=1}^K)$$

   where $RelLearner$ is the relation learner module within MetaR, $E(h_k)$ and $E(t_k)$ are the embeddings of the head and tail entities for the k-th instance in the support set and K is the number of instances in the support set.

   To incorporate PathCon's structural information, we augment the input to the $RelLearner$ module by taking the PathCon relation embedding ($\mathbf{R_{PathCon}}$) as an additional input with the entity embeddings (by concatenation). The modified equation becomes:

$$r_i = RelLearner((E(h_k), \mathbf{R_{PathCon}}, E(t_k))_{k=1}^K) \tag{5.15}$$

   This modification allows MetaR to leverage the relational context captured by PathCon directly during the relation learning process. By feeding both entity and PathCon relation embeddings to the relation learner, we encourage MetaR to learn relation representations that are more informed by the graph structure.

3. **Mixture of Experts:** The Mixture of Experts (MoE) architecture has emerged as a transformative approach in the field of LLMs. Inspired by the notion that a complex task can be efficiently addressed by dividing it among specialized experts, MoE introduces a paradigm shift in model design. Instead of relying on a single network, MoE models comprise multiple expert networks, each specializing in a specific aspect of the task. A gating network dynamically selects which experts to consult for a given input, effectively routing information to the most relevant subnetworks. This approach has revolutionized LLMs by allowing for significant increases in model

capacity without a proportional increase in computational cost. By activating only a subset of experts for each input, MoE models achieve remarkable computational efficiency while maintaining high performance.

The potential of MoE to leverage specialized knowledge from different experts motivated us to explore this architecture for integrating PathCon and MetaR. We hypothesized that treating PathCon and MetaR as separate experts would allow us to capitalize on their respective strengths for improved link prediction. A gating network would be used to dynamically determine the contribution of each expert (PathCon and MetaR) based on the specific query and knowledge graph context. In our implementation, it is a simple neural network consisting of a linear layer followed by a softmax activation. The corresponding equations are as follows:

$$\begin{aligned}
\mathbf{E_{flat}} &= \text{flatten}(\mathbf{E_{support}}) \in \mathbb{R}^{B\times|S|\times N_e d} \\
\mathbf{G} &= softmax(\mathbf{E_{flat}} \cdot \mathbf{W_g} + \mathbf{b_g}) \in \mathbb{R}^{B\times|S|\times E} \\
\mathbf{G_{avg}} &= \frac{1}{|S|}\sum_{s\in S}\mathbf{G}^{(s)} \in \mathbb{R}^{B\times E}
\end{aligned}$$

where $\mathbf{W_g} \in \mathbb{R}^{N_e d\times E}$ is the weight matrix of the linear layer and $\mathbf{b_g} \in \mathbb{R}^{E}$ is the bias vector of the linear layer.

The support set embeddings are flattened, and the gating mechanism is applied to produce gating weights. These weights are then averaged over the support set dimension, where E is the number of experts (E = 2).

$$\begin{aligned}
\mathbf{E} &= [\mathbf{R_{PathCon}}, \mathbf{R_{MetaR}}] \in \mathbb{R}^{B\times E\times d} \\
\mathbf{R_{mix}} &= \mathbf{G_{avg}} \cdot \mathbf{E} \in \mathbb{R}^{B\times d}
\end{aligned}$$

The PathCon and MetaR outputs are stacked, and the gating weights are used to compute a weighted combination of the expert outputs.

The final relation representation is the gated combination of PathCon and MetaR outputs $\mathbf{R_{mix}}$ which is used to predict the most likely relation between entities h and t.

5.7 Experiments

5.7.1 Evaluation Setup

To assess the performance of our proposed approach, combining MetaR and PathCon, we utilized the MetaR framework for benchmarking.[1] Evaluation was conducted on two knowledge graph datasets: NELL-One [84] and Wiki-One [84]. We employed the following standard metrics for knowledge graph completion:

- **Mean Reciprocal Rank (MRR):** Measures the average reciprocal rank of the correct answer.
- **Hits@10:** Percentage of cases where the correct answer is within the top 10 predictions.
- **Hits@5:** Percentage of cases where the correct answer is within the top 5 predictions.
- **Hits@1:** Percentage of cases where the correct answer is ranked first.

[1]The code combining PathCon to the original MetaR framework can be found here : https://anonymous.4open.science/r/Submission-/.

5.7.2 Comparison of Datasets : NELL-One and Wiki-One

The two datasets used in this study, NELL-One and Wiki-One, exhibit significant differences that contribute to the varying performance of models.

- **Sparsity:** Wiki-One has a higher proportion of entities appearing in only one triple during training (82.8%) compared to NELL-One (37.1%).
- **Number of tasks:** NELL-One has a smaller number of training tasks (51) compared to Wiki-One (133).

Table 5.1 presents a more detailed comparison between the two datasets.

Table 5.1: NELL-One and Wiki-One Datasets Comparison.

Feature	NELL-One	Wiki-One
# Entities	68,545	4,838,244
# Relations	358	822
# Triples	181,109	5,859,240
One-shot entities (%)	37.1	82.8

5.7.3 Results and Discussion - Adapting PathCon to the MetaR framework

Table 5.2 presents a summary of the experimental results.

Table 5.2: Summary of behaviour across two datasets.

Model	MRR	Hits@10	Hits@5	Hits@1
NELL-One - 5-shot				
MetaR	0.2530	0.4370	0.3490	0.155
PathCon - RNN & attention	**0.2620**	**0.4520**	**0.3780**	**0.159**
Wiki-One - 5-shot				
MetaR	**0.2220**	0.3200	**0.2760**	**0.167**
PathCon - RNN & mean	0.2070	**0.3230**	0.2560	0.152

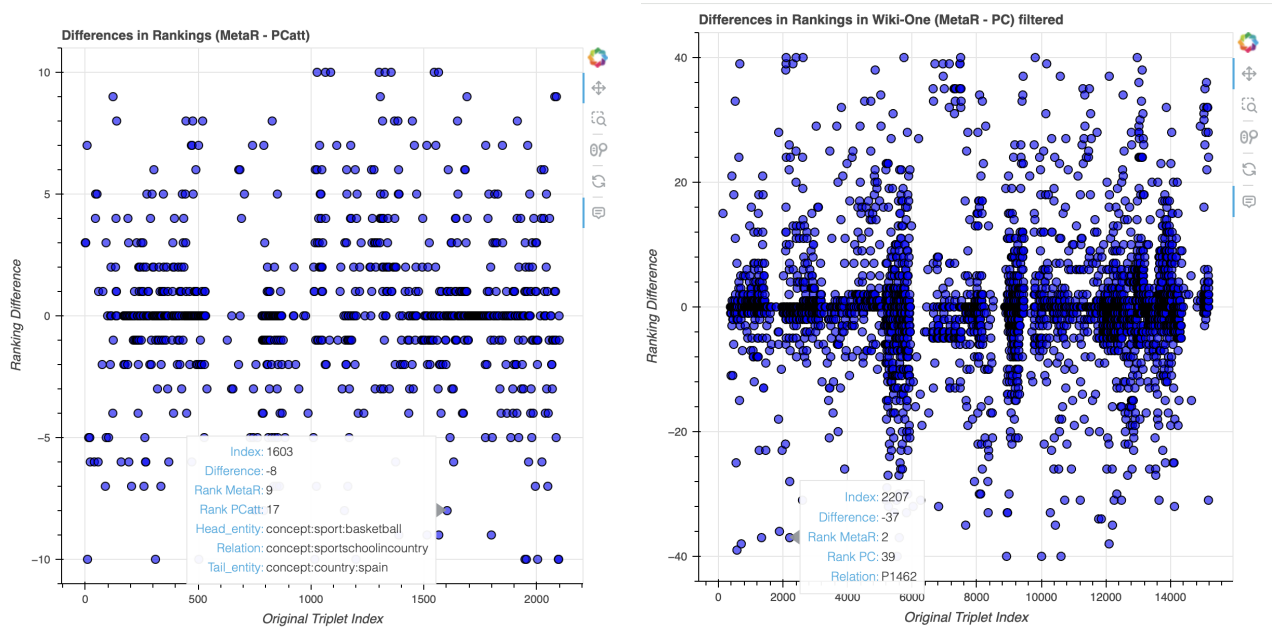
Key observations:

- **NELL-One Dataset:** Our PathCon implementation (RNN & attention) demonstrates superior performance over MetaR across all metrics on the NELL-One dataset.
- **Wiki-One Dataset:** The results on Wiki-One are less clear-cut. MetaR has a higher MRR and Hits@1, while PathCon (RNN & mean) shows a marginally better Hits@10 score.

5.7.4 Triplet-Level Analysis:

To gain a deeper understanding of the models' behavior, we conducted a triplet-level analysis examining the differences in how MetaR and PathCon rank each individual triplet in the two datasets. Figure 5.6 illustrates these differences through a scatter plot, where the x-axis represents the original triplet index and the y-axis depicts the ranking difference (calculated as MetaR rank minus PathCon rank).

The scattered distribution of points across both positive and negative regions of the y-axis reveals a crucial observation: the relative performance of MetaR and PCatt exhibits significant variability depending on the specific triplet under consideration. This suggests that each model possesses distinct



(a) Rankings of ground-truth triplets for MetaR and PathCon (RNN & attention) on NELL-One dataset.

(b) Rankings of ground-truth triplets for MetaR and PathCon (RNN & mean) on Wiki-One dataset.

Figure 5.6: Comparative analysis of rankings on NELL-One and Wiki-One datasets.

strengths and weaknesses, excelling in different scenarios within the knowledge graph completion task.

Despite the observed variations in individual triplet performance, the overall distribution of points appears roughly balanced between the positive and negative regions, with 283 positive differences and 264 negative differences for NELL-One for example. This balance indicates that while MetaR and PathCon might favor different types of triplets, their overall performance remains comparable across the entire dataset, which is coherent with our previous results. However, a clear pattern behind these individual triplet differences remains elusive, as variations occur even within triplets sharing the same relation type.

The contrasting performance profiles observed at the triplet level offer a compelling opportunity: the potential to leverage the complementary strengths of both models through combination. By integrating MetaR and PathCon, we can envision a system that capitalizes on their individual advantages, mitigating their respective weaknesses and achieving superior accuracy and robustness in few-shot link prediction tasks.

5.7.5 Results and Discussion - Combining PathCon with MetaR

Table 5.3 presents a summary of the experimental results.

Table 5.3: Summary of behaviour across two datasets.

Model	MRR	Hits@10	Hits@5	Hits@1
NELL-One - 5-shot				
MetaR	0.2530	0.4370	0.3490	0.155
PathCon - RNN & attention	0.2620	**0.4520**	**0.3780**	0.159
Cross-Attention Mechanism	0.2570	0.4500	0.3590	0.155
MetaR Alignment	0.2550	0.4370	0.3580	0.155
Mixture of Experts	**0.2640**	0.4360	0.3500	**0.173**

While combining PathCon's relational path modeling with MetaR's meta-learning approach yielded some improvements in the case of the Mixture of Experts architecture, the gains remain modest and potentially dataset-specific (NELL-One). Consequently, these results are not definitive and do not conclusively demonstrate a synergistic effect from merging the two models.

5.7.6 A Discussion of Hybrid Model Performance on Real and Synthetic Datasets:

Despite our efforts to enhance few-shot link prediction using the MetaR framework and PathCon, the results remained inconclusive. Various combinations of MetaR and PathCon, including different relation learners and attention mechanisms, failed to significantly improve prediction scores.

Experiments on synthetic datasets: To delve deeper, we conducted experiments on synthetically generated triplets, encompassing positional, symmetric, and transitive relations, mirroring the methodology from [110]. Our aim was to assess whether PathCon's context and relation utilization could effectively capture the underlying structure of these synthetic relations.

The results on synthetic data were equally inconclusive, failing to demonstrate a clear advantage of PathCon in leveraging structural information. This lack of improvement leads us to question the efficacy of the current framework and explore potential explanations for these limitations.

Possible Explanations for Inconclusive Results:

Inadequacy of the Scoring Mechanism: The use of TransE for scoring triplets might be a limiting factor. TransE, due to its simplistic additive scoring function, encounters difficulty in accurately representing hierarchical relations like "is_a" or "part_of". The model's reliance on vector addition fails to capture the inherent directionality and transitivity crucial for these relations. For example, if "Cat is_a Mammal" and "Mammal is_a Animal", TransE struggles to infer the transitive relation "Cat is_a Animal". Its vector space representation tends to position the embedding for "Cat" closer to "Animal" offset by the relation embedding, rather than directly near "Animal" as implied by transitivity. In a few-shot scenario, where only a handful of examples are available, this limitation is amplified, as the model cannot effectively discern these intricate patterns from limited data. Exploring more expressive scoring functions, such as those based on graph neural networks or complex embedding models, could potentially improve the ability to discern and exploit structural information.

Incompatibility of Learning Paradigms: A fundamental incompatibility might exist between PathCon's inductive, structure-focused approach and MetaR's task-adaptive, gradient-based meta-learning. PathCon excels in leveraging global graph structure, while MetaR prioritizes rapid adaptation to new relations based on limited local information. This potential mismatch could hinder the effective integration of their learned representations.

Insufficient Path Diversity: PathCon's performance counts on the diversity and representativeness of relational paths it discovers. In few-shot scenarios, with limited support set examples, the model might explore only a restricted set of paths, failing to capture the full relational context. This could lead to inaccurate predictions, especially for relations that rely on longer or less frequent paths.

5.8 Future Work: Exploring Direct Decoding Approaches

While this thesis focused on models employing scoring functions for link prediction, the rapidly evolving field of knowledge graph completion offers exciting alternative avenues for future research. One promising direction lies in exploring direct decoding approaches, exemplified by the INDIGO model [31]. Unlike scoring function-based methods, INDIGO directly interprets the learned representations within its GNN architecture to predict missing links, eliminating the need for a separate scoring mechanism. Building upon these advancements, future work could investigate hybrid models that combine the strengths of scoring function-based approaches like MetaR and PathCon with the direct decoding mechanisms pioneered by INDIGO. Such a hybrid model could leverage the context and structural information captured by PathCon alongside the meta-learning capabilities of MetaR, while benefiting from INDIGO's direct decoding strategy. This fusion of different learning paradigms

has the potential to significantly enhance link prediction performance, pushing the boundaries of knowledge graph completion and unlocking new possibilities for knowledge-driven applications.

Chapter 6

Conclusions and Future Work

6.1 Summary of Findings

This thesis explores the potential of graph neural networks in addressing two crucial challenges: influence maximization in social networks and few-shot link prediction in knowledge graphs. We demonstrate the effectiveness of GNNs in capturing intricate relationships within graph-structured data, leading to novel solutions that advance the state-of-the-art in both domains.

In the realm of influence maximization, we introduce a series of progressively refined models, culminating in TIM-GNN^x, a topic-aware influence maximization solution that outperforms existing methods in performance, latency, and robustness. Our approach leverages the strengths of deep reinforcement learning and graph neural networks to efficiently predict marginal spread gain. By integrating graph attention mechanisms, positional encodings, and a novel cross-attention scheme, TIM-GNN^x effectively handles the complexities of topic-aware diffusion while achieving significant speed-ups in query time. This makes TIM-GNN^x a compelling solution for real-time influence maximization applications, offering a promising avenue for optimizing information dissemination strategies in large-scale social networks. However, the scalability of TIM-GNN^x on very large and complex topic-aware graphs remains a challenge, which could be addressed in the future by exploring graph sampling techniques, distributed training strategies, or alternative DRL algorithms that offer improved scalability and convergence properties.

For few-shot link prediction in knowledge graphs, we delve into the limitations of existing meta-learning approaches and explore the potential of integrating the PathCon method, which excels in capturing structural information within knowledge graphs, with the MetaR framework. While our initial investigations yielded promising results on certain datasets, demonstrating the feasibility of adapting PathCon for few-shot link prediction, our hybrid models integrating PathCon and MetaR did not consistently achieve the desired performance gains. This highlights the complexities of combining inductive, structure-focused approaches with task-adaptive, gradient-based meta-learning.

Despite their distinct applications, influence maximization and few-shot link prediction share a compelling synergy. Both rely on extracting meaningful patterns from limited data, leveraging GNNs and their message-passing framework to effectively represent relationships and make predictions. Furthermore, the successful integration of attention mechanisms in both domains is a key example of this synergy: The cross-attention technique, initially used for integrating PathCon and MetaR in few-shot link prediction, was subsequently explored as a method for efficiently handling diverse queries in the TIM-GNNx model for influence maximization. This example underscores the valuable cross-pollination of ideas between seemingly distinct research areas.

6.2 Limitations and Future Research Directions

Despite the successes achieved, our work also reveals limitations and paves the way for future research directions. In the case of influence maximization, the current reliance on S2V-DQN as the foundational framework can be further enhanced by exploring more sophisticated GNN architectures. In particular, integrating cooperative GNNs [74], which enable strategic communication patterns between nodes, could significantly improve the model's ability to capture complex diffusion dynamics.

These models offer a more nuanced and flexible approach to information propagation compared to the standard message passing schemes used in S2V-DQN. By allowing nodes to dynamically choose their communication strategies based on their state and the states of their neighbors, cooperative GNNs could potentially unlock new levels of performance and efficiency in influence maximization tasks.

Regarding few-shot link prediction, future work should explore alternative scoring functions that are more expressive and capable of capturing complex relational semantics. Moving beyond the simplistic additive scoring of TransE, recent advancements in GNN-based link prediction, such as the INDIGO model, which directly encodes the knowledge graph into a GNN, offer promising avenues. By combining the structural insights from PathCon with the direct decoding capabilities of INDIGO, we envision a hybrid model that can overcome the limitations of scoring function-based methods and achieve superior accuracy and interpretability in few-shot link prediction. This integration could significantly enhance knowledge graph completion, enabling more robust and versatile knowledge-driven applications.

6.3 Publications

- "Topic-Aware Influence Maximization with Deep Reinforcement Learning and Graph Attention Networks" currently under review in the European Conference on Machine Learning and Data Mining (Journal track - ECML PKDD 2025).
- "Structure-Aware Meta-Learning for Few-Shot Knowledge Graph Link Prediction" accepted in the International Workshop on Resource-Efficient Learning for Knowledge Discovery, colocated with KDD 2024.

Appendix A

Supplementary Material

In this appendix, we present the plots for each of the five evaluation items and three topic-aware datasets (Flixster, Memetracker and Sina Weibo), as only error bar plots were presented in this thesis.

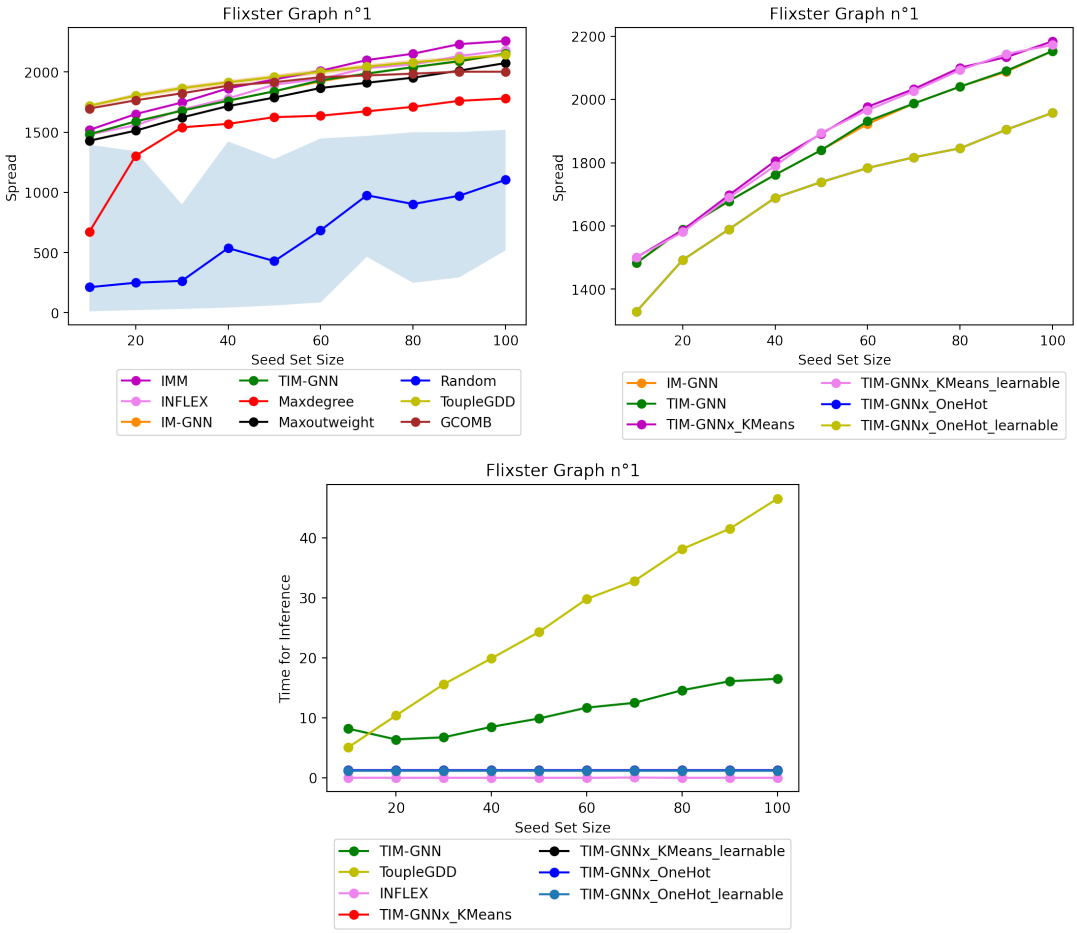


Figure 1: Results on projected graph n°1 - Flixster. Comparing TIM-GNN with baselines (left), variants of TIM-GNN^x (right), and query time (down).

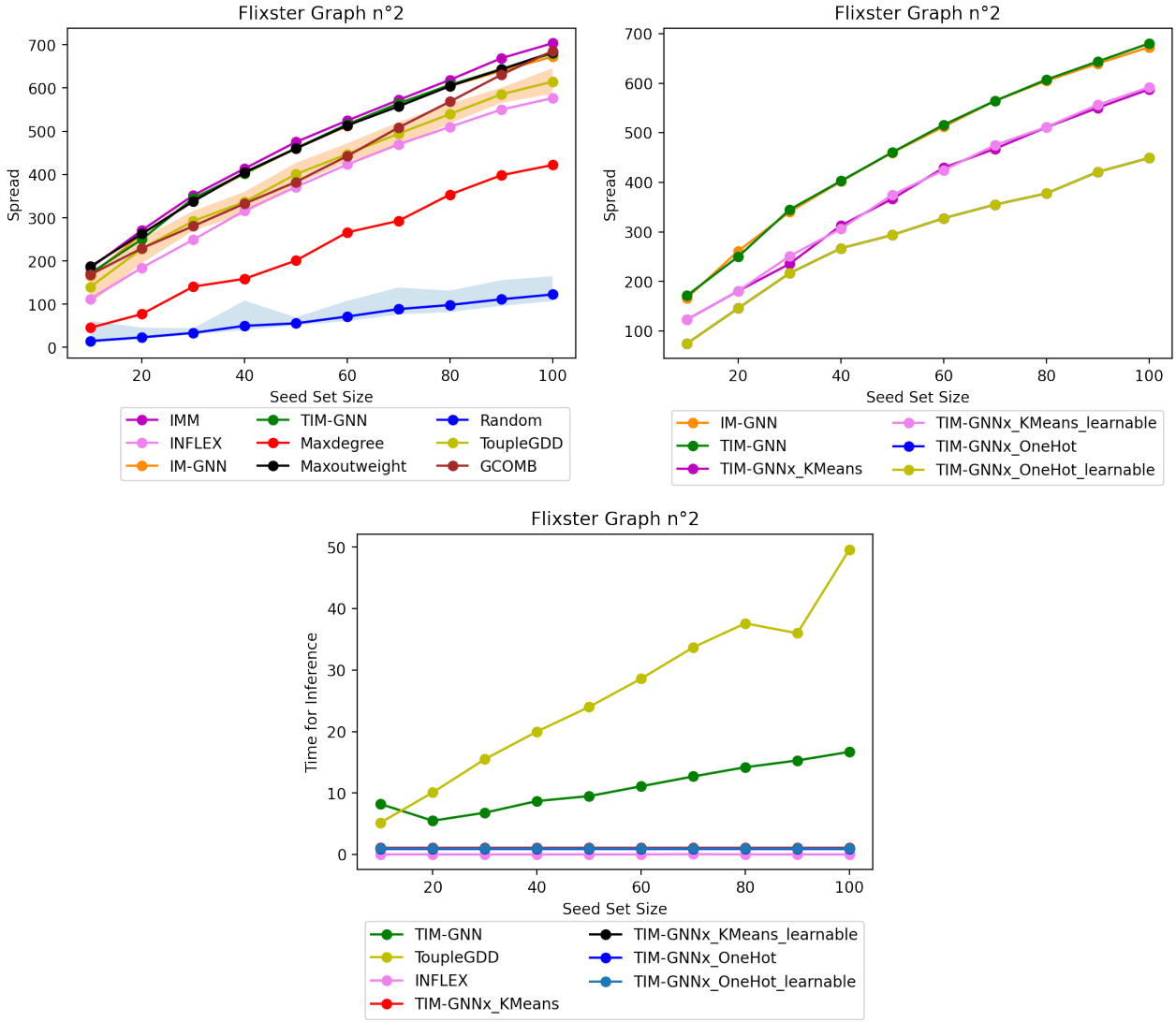


Figure 2: Results on projected graph n°2 - Flixster. Comparing TIM-GNN with baselines (left), variants of TIM-GNN^x (right), and query time (down).

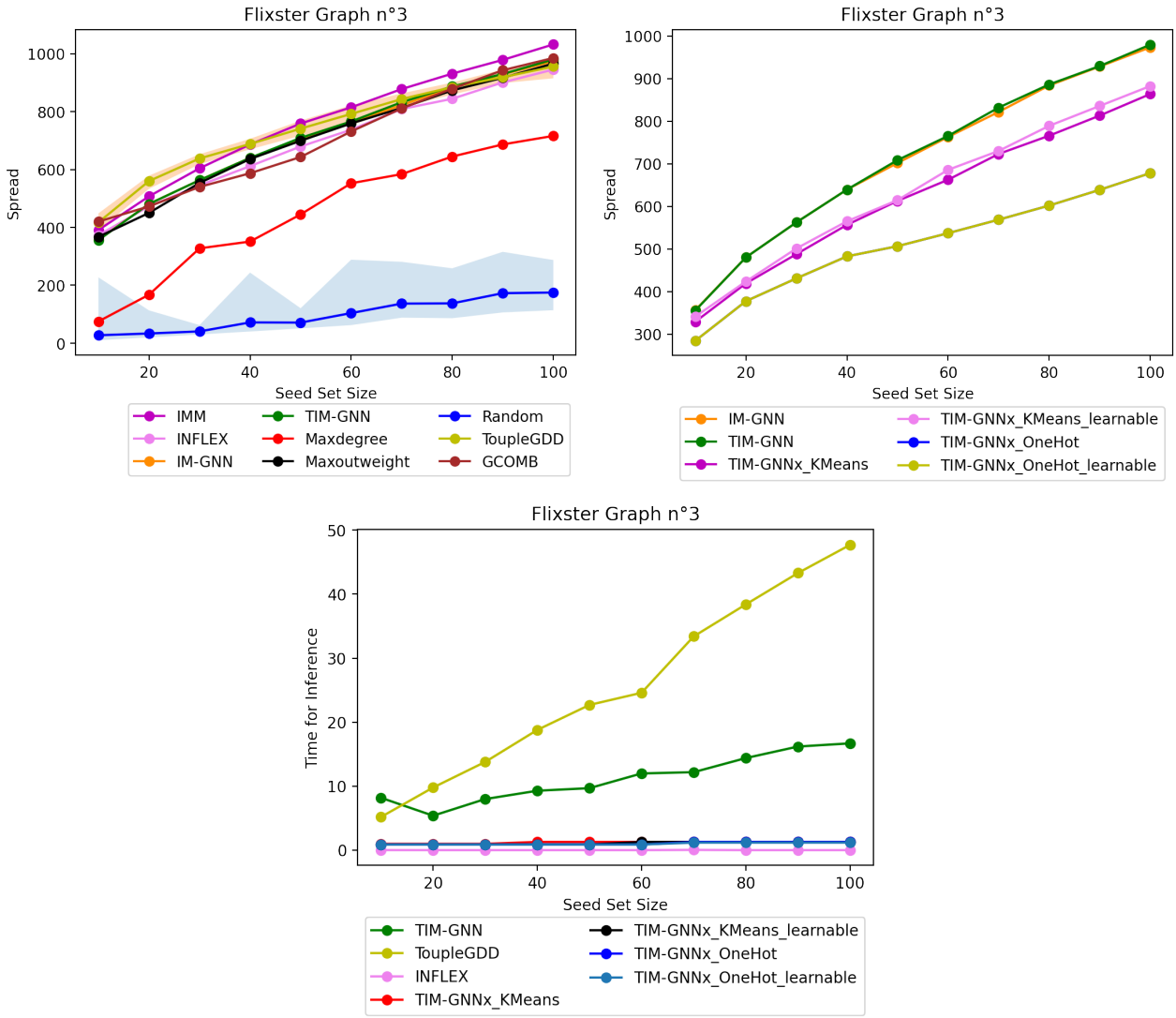


Figure 3: Results on projected graph n°3 - Flixster. Comparing TIM-GNN with baselines (left), variants of TIM-GNN^x (right), and query time (down).

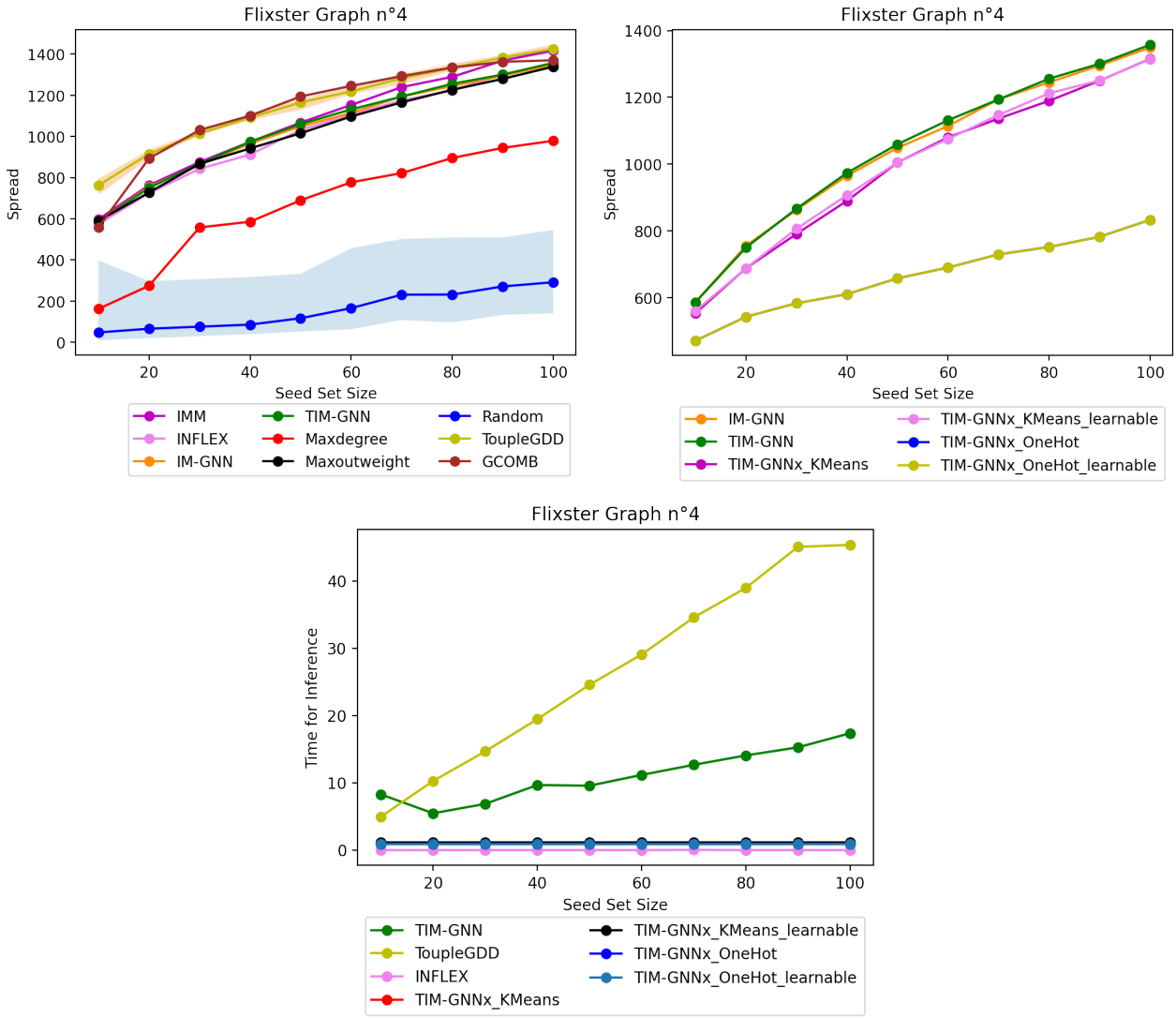


Figure 4: Results on projected graph n°4 - Flixster. Comparing TIM-GNN with baselines (left), variants of TIM-GNN^x (right), and query time (down).

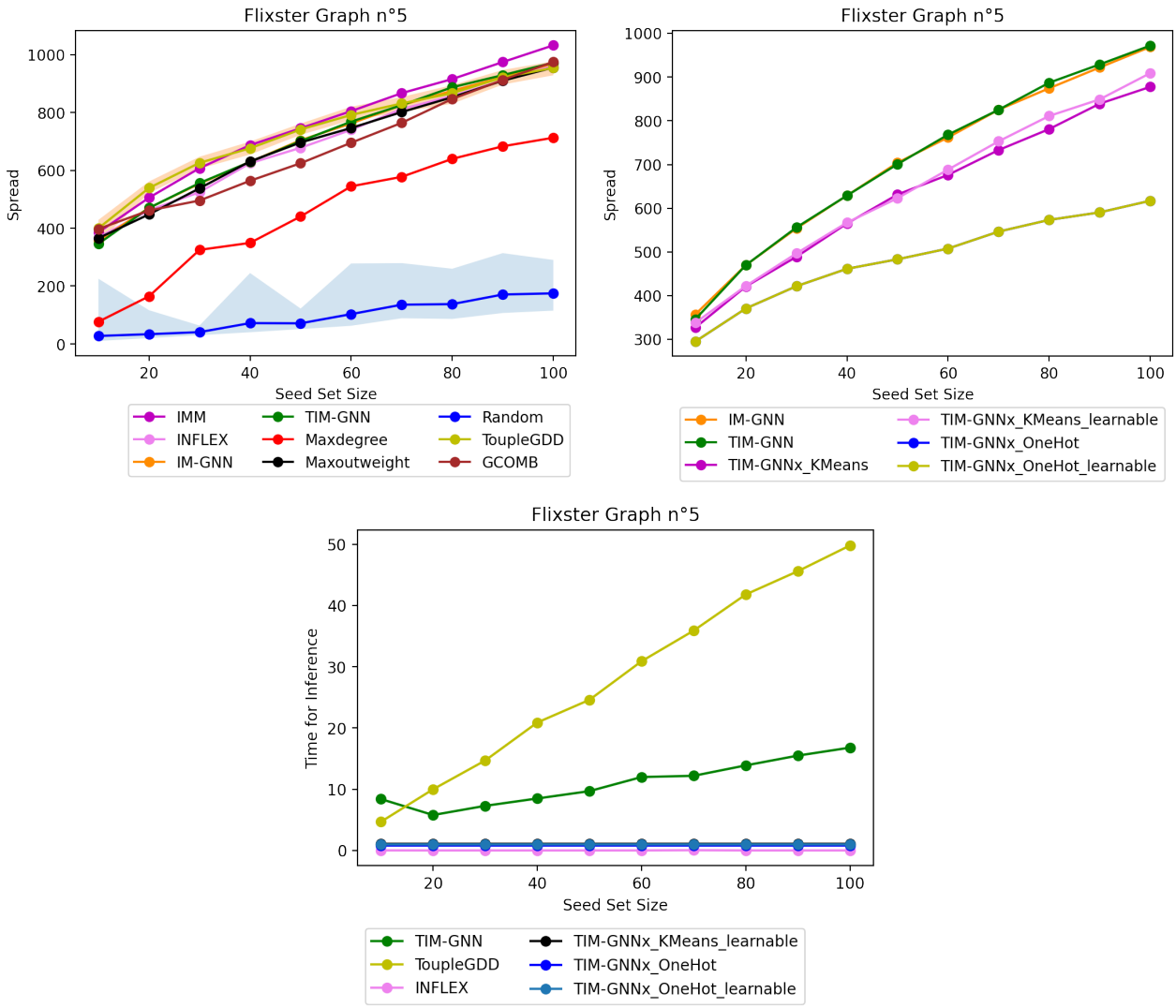


Figure 5: Results on projected graph n°5 - Flixster. Comparing TIM-GNN with baselines (left), variants of TIM-GNN^x (right), and query time (down).

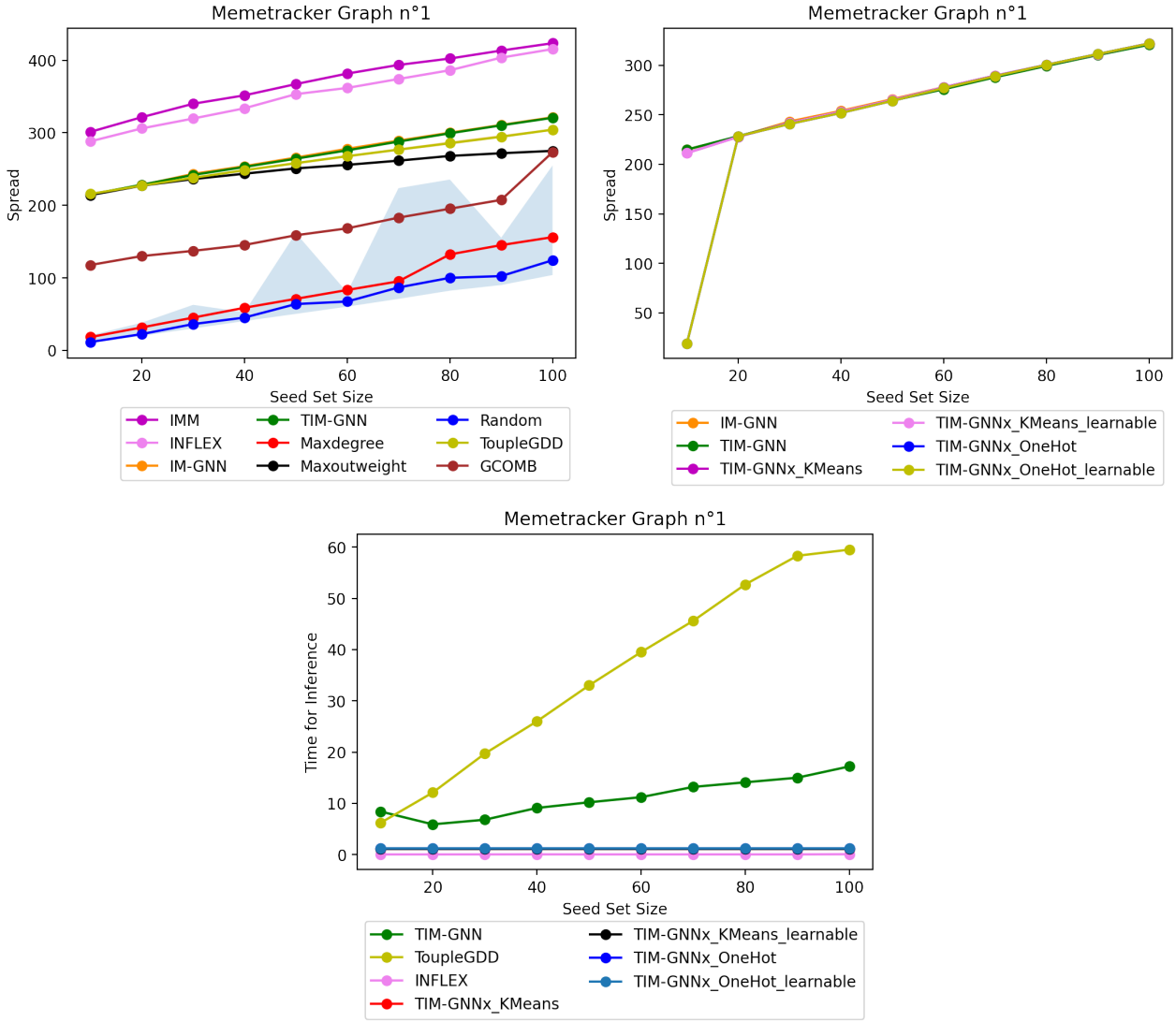


Figure 6: Results on projected graph n°1 - Memetracker. Comparing TIM-GNN with baselines (left), variants of TIM-GNN^x (right), and query time (down).

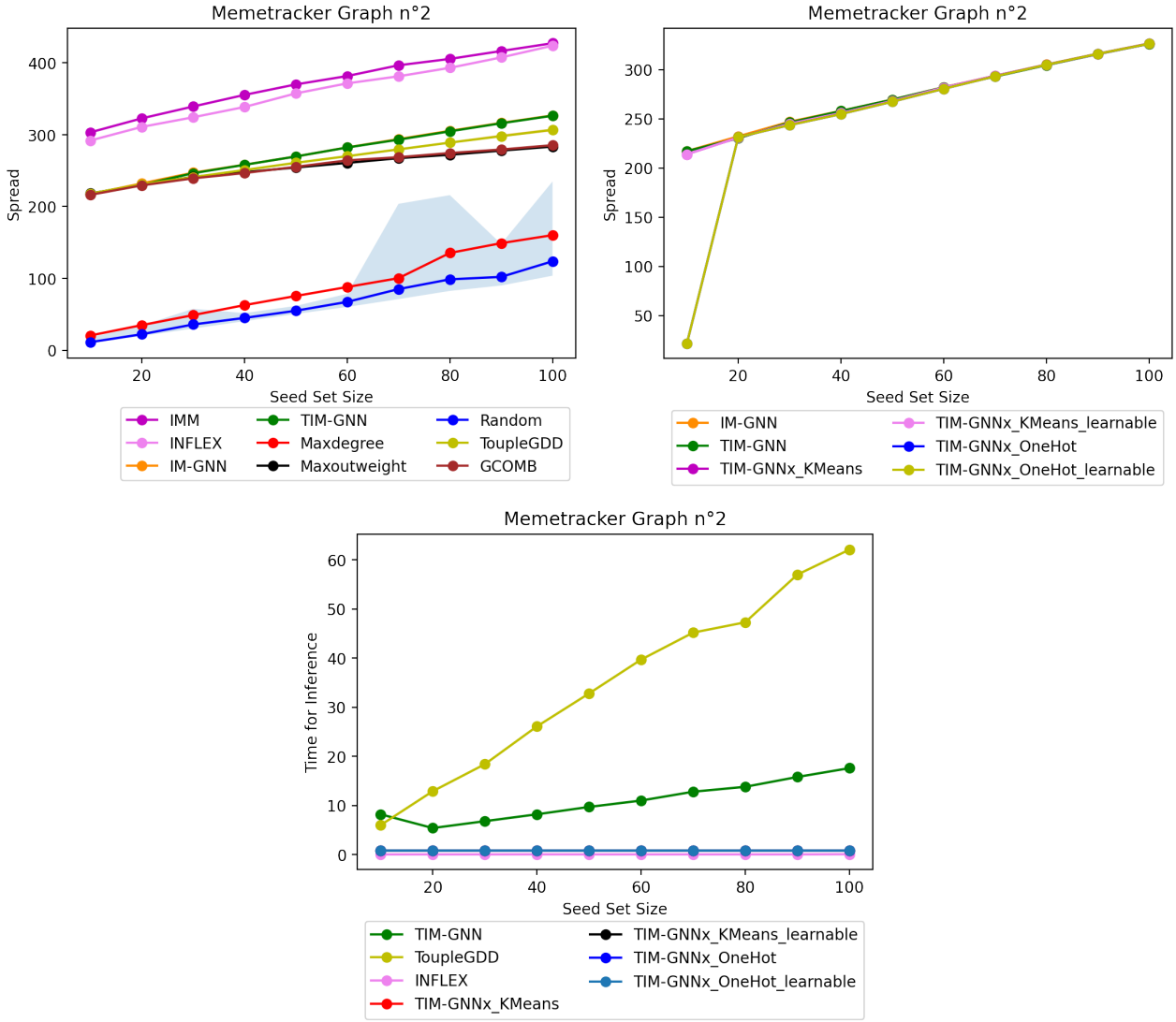


Figure 7: Results on projected graph n°2 - Memetracker. Comparing TIM-GNN with baselines (left), variants of TIM-GNN^x (right), and query time (down).

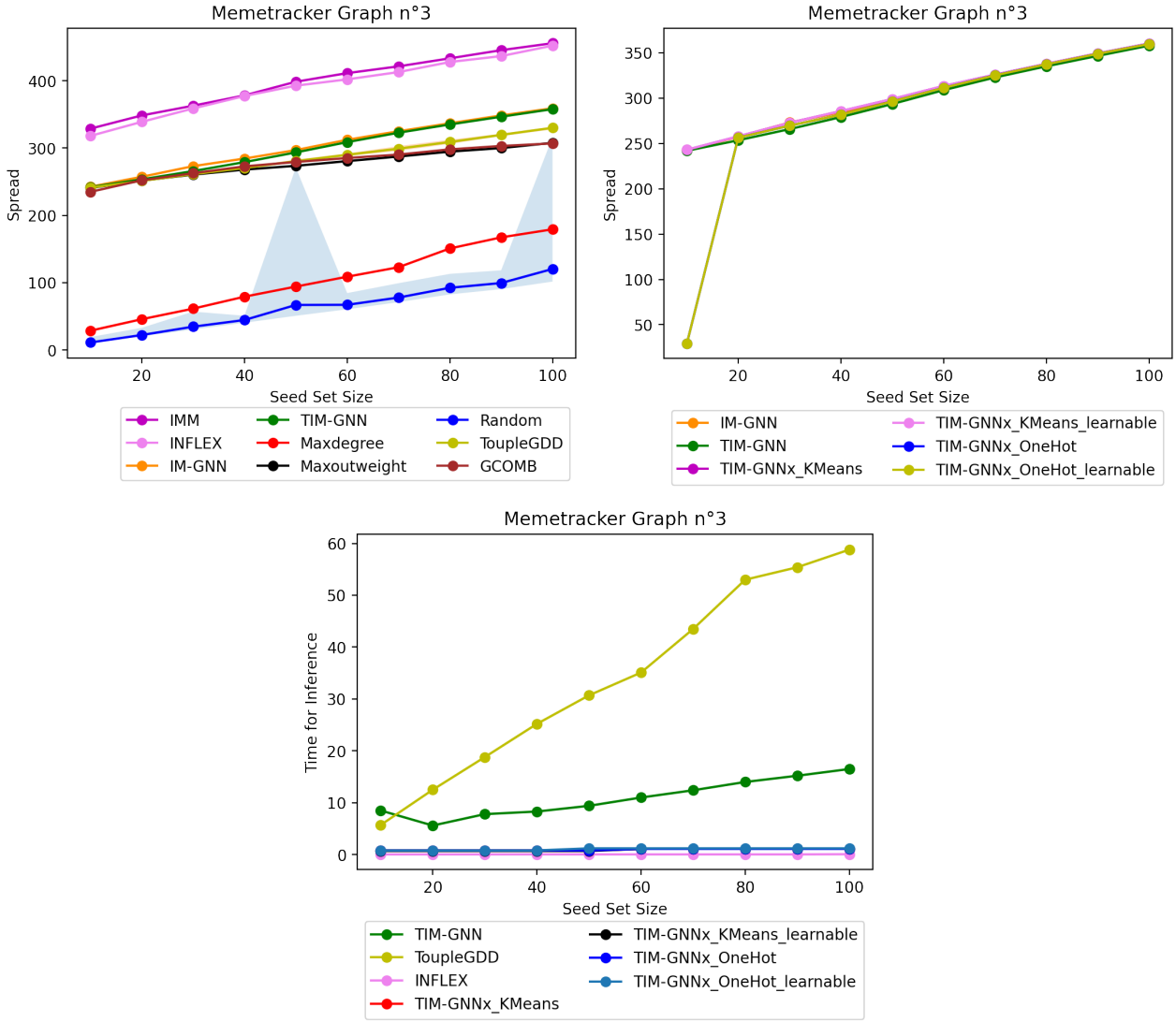


Figure 8: Results on projected graph n°3 - MemeTracker. Comparing TIM-GNN with baselines (left), variants of TIM-GNN^x (right), and query time (down).

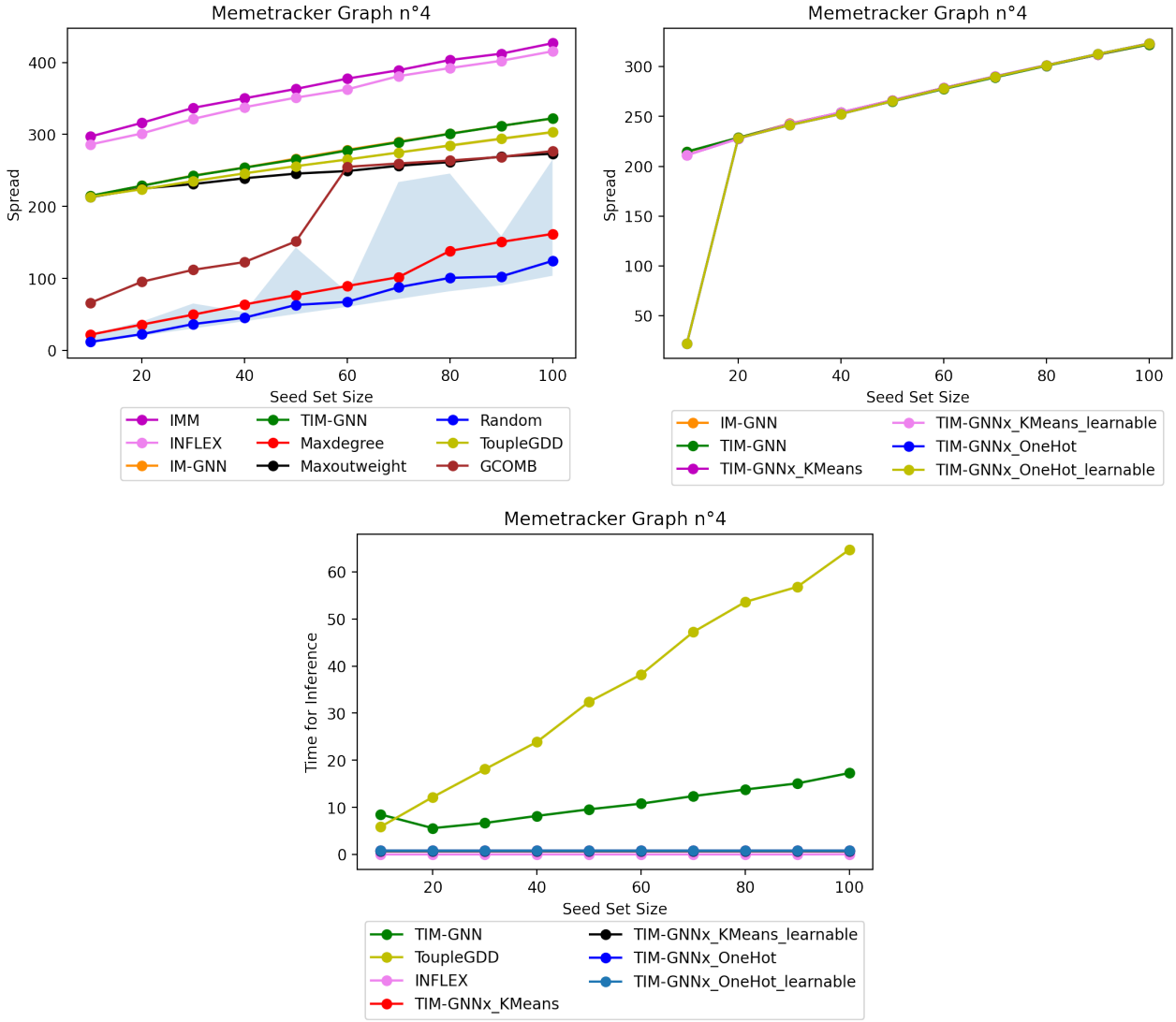


Figure 9: Results on projected graph n°4 - Memetracker. Comparing TIM-GNN with baselines (left), variants of TIM-GNN^x (right), and query time (down).

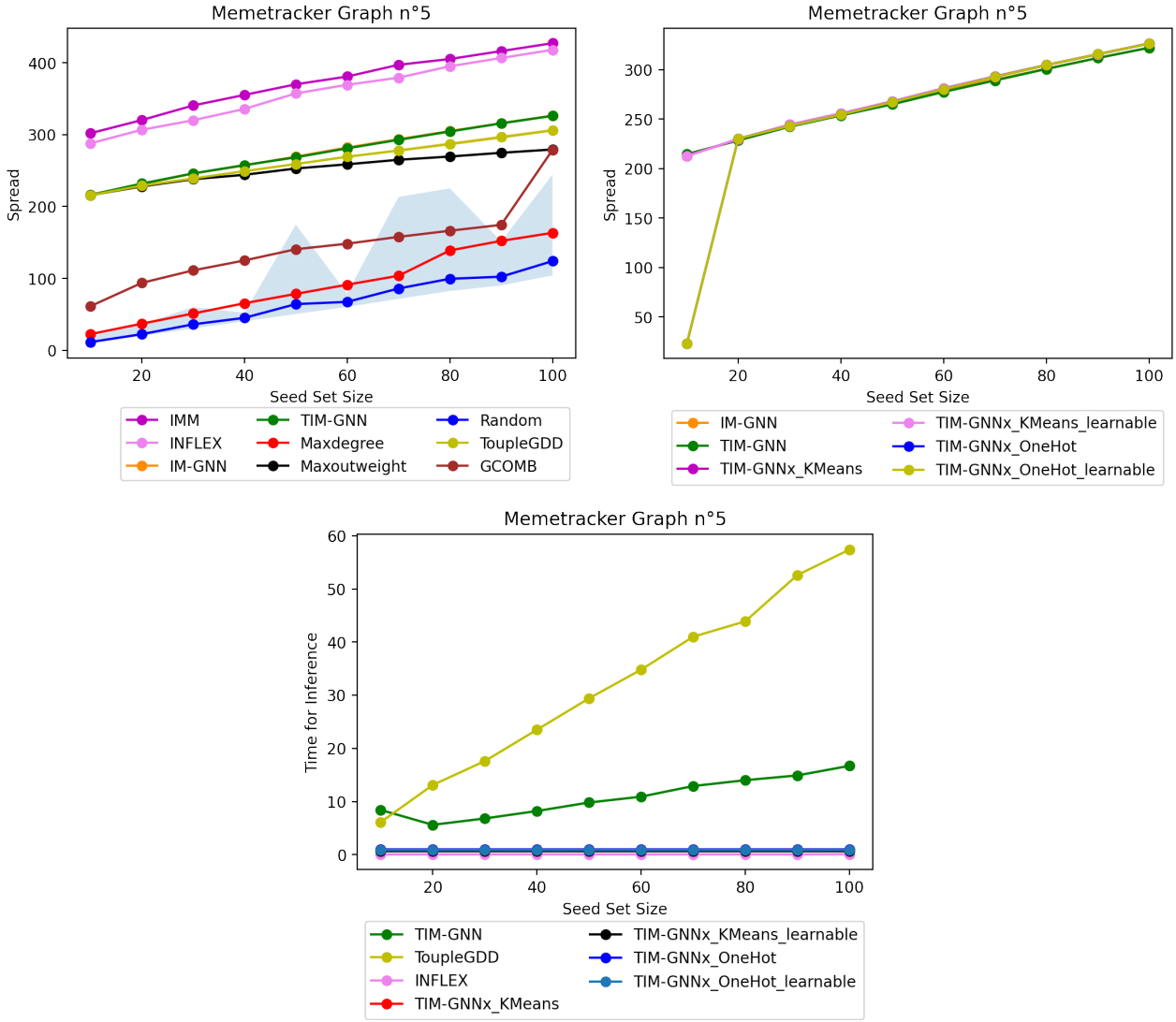


Figure 10: Results on projected graph n°5 - Memetracker. Comparing TIM-GNN with baselines (left), variants of TIM-GNN^x (right), and query time (down).

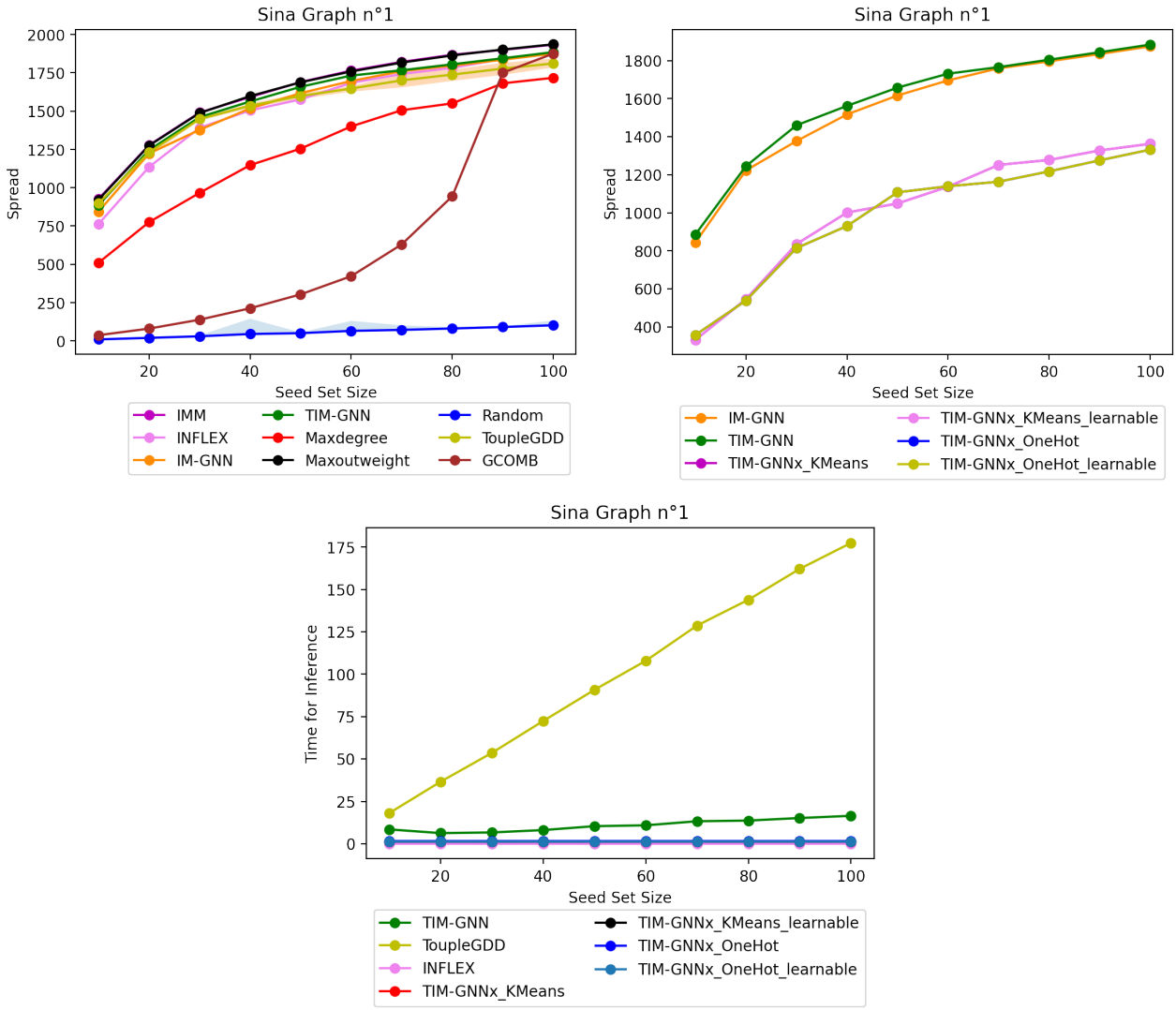


Figure 11: Results on projected graph n°1 - Sina Weibo. Comparing TIM-GNN with baselines (left), variants of TIM-GNN^x (right), and query time (down).

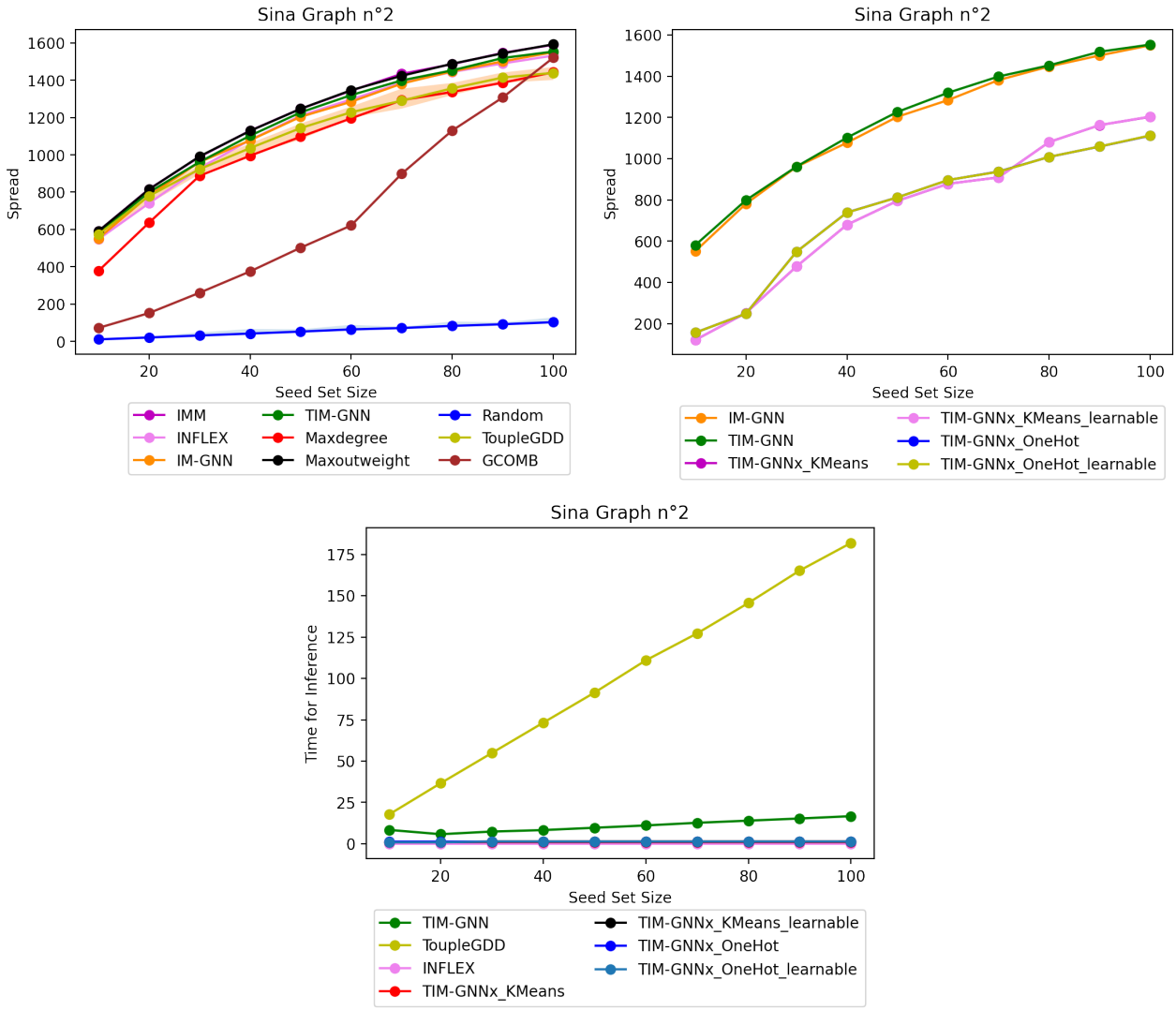


Figure 12: Results on projected graph n°2 - Sina Weibo. Comparing TIM-GNN with baselines (left), variants of TIM-GNN^x (right), and query time (down).

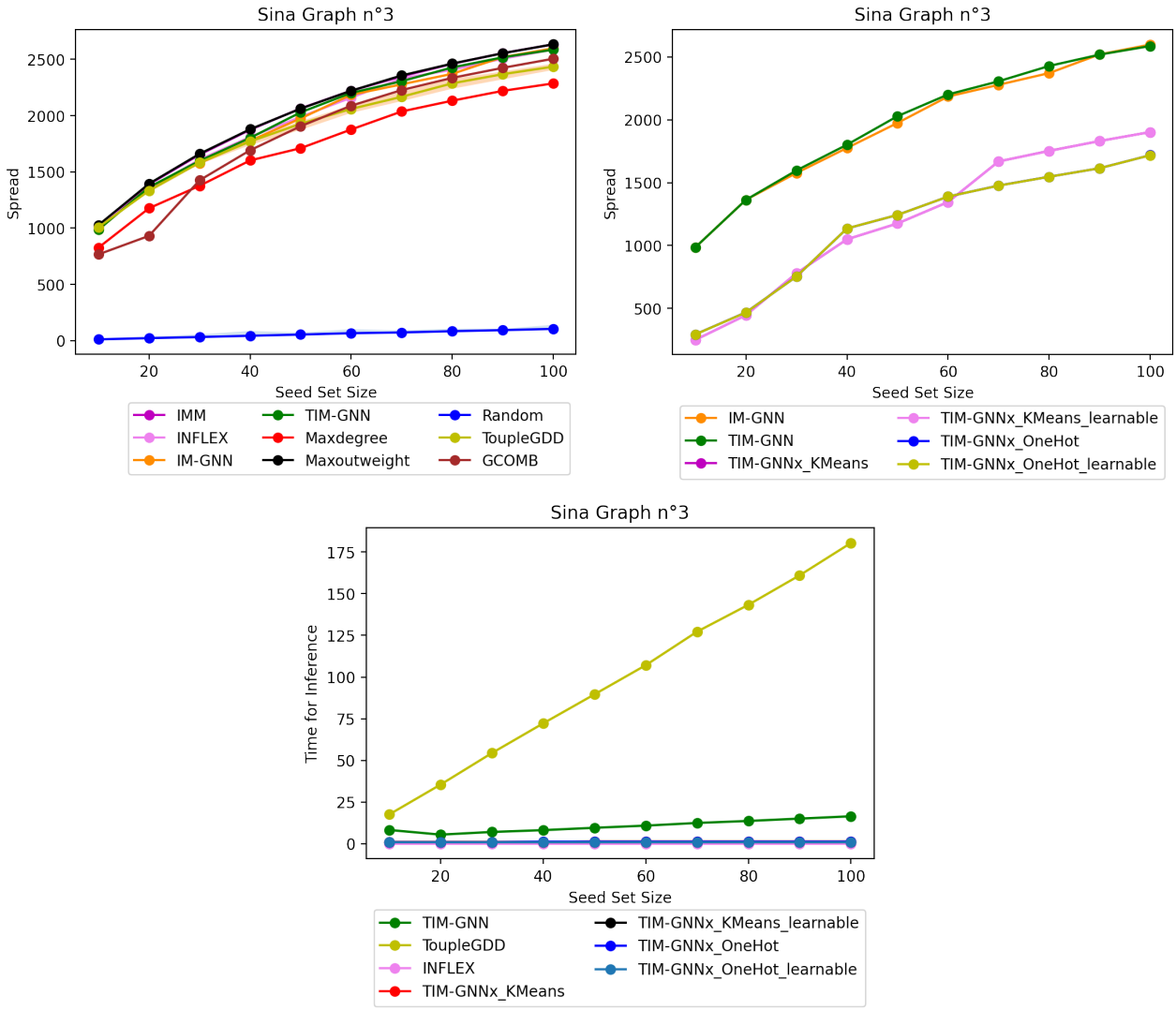


Figure 13: Results on projected graph n°3 - Sina Weibo. Comparing TIM-GNN with baselines (left), variants of TIM-GNN^x (right), and query time (down).

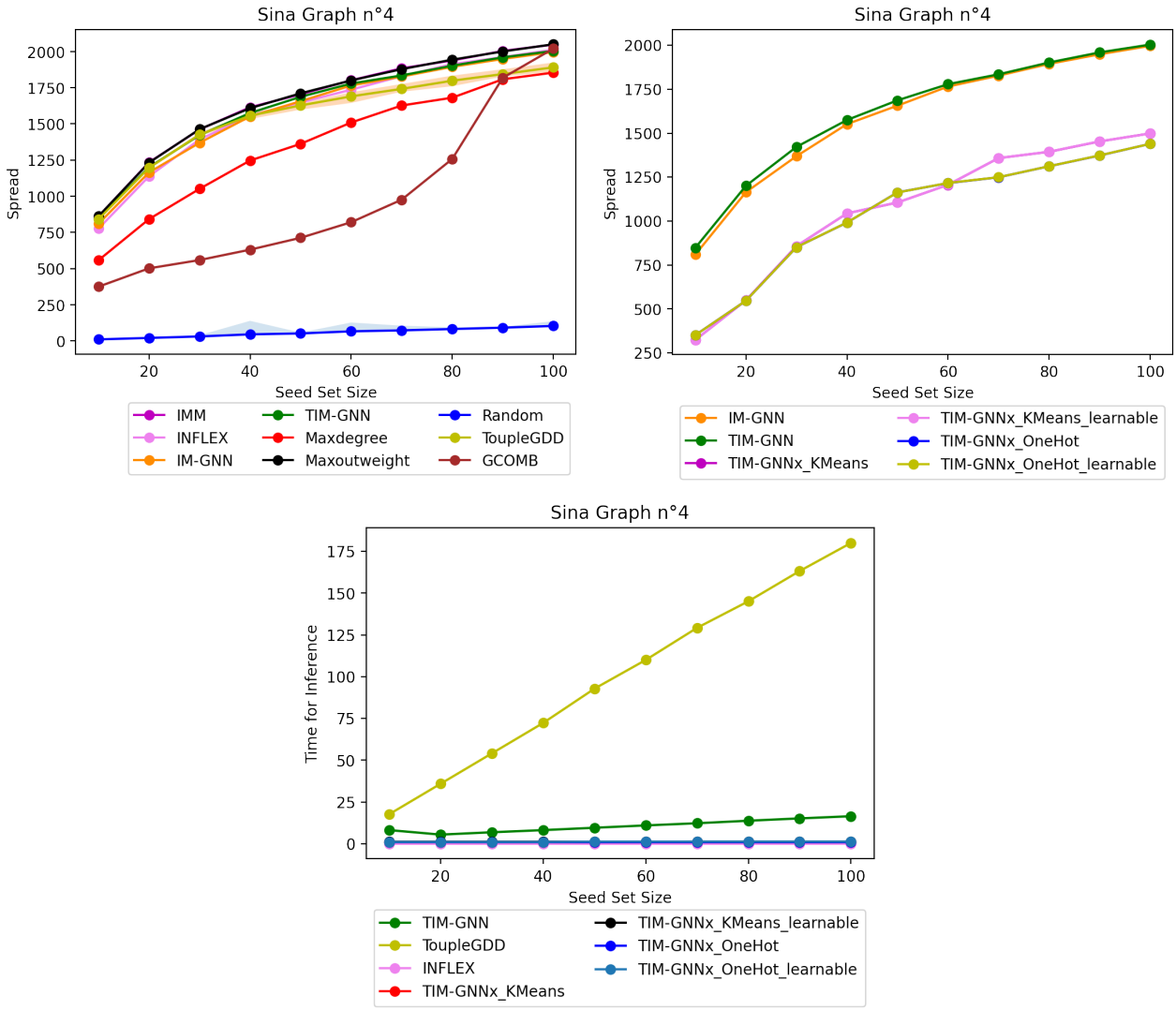


Figure 14: Results on projected graph n°4 - Sina Weibo. Comparing TIM-GNN with baselines (left), variants of TIM-GNN^x (right), and query time (down).

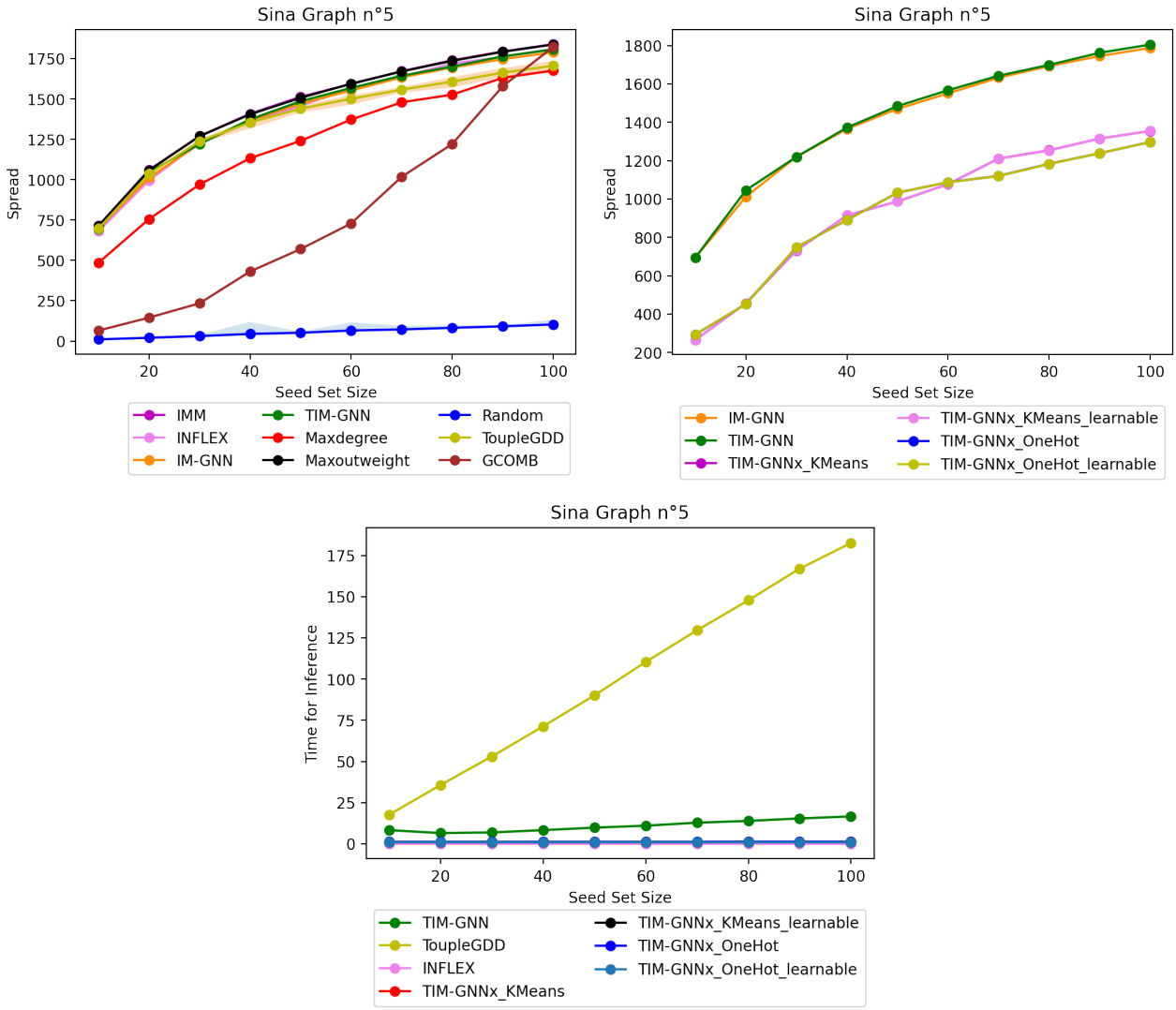


Figure 15: Results on projected graph n°5 - Sina Weibo. Comparing TIM-GNN with baselines (left), variants of TIM-GNN^x (right), and query time (down).

Bibliography

[1] P. Veličković, "Everything is connected: Graph neural networks," *Current Opinion in Structural Biology*, vol. 79, p. 102538, Apr. 2023.

[2] H. Wang, H. Ren, and J. Leskovec, "Entity context and relational paths for knowledge graph completion," *CoRR*, vol. abs/2002.06757, 2020.

[3] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. Kohl, A. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, and D. Hassabis, "Highly accurate protein structure prediction with alphafold," *Nature*, vol. 596, pp. 1–11, 07 2021.

[4] M. Varadi, S. Anyango, M. Deshpande, S. Nair, C. Natassia, G. Yordanova, D. Yuan, O. Stroe, G. Wood, A. Laydon, A. Žídek, T. Green, K. Tunyasuvunakool, S. Petersen, J. Jumper, E. Clancy, R. Green, A. Vora, M. Lutfi, M. Figurnov, A. Cowie, N. Hobbs, P. Kohli, G. Kleywegt, E. Birney, D. Hassabis, and S. Velankar, "AlphaFold Protein Structure Database: massively expanding the structural coverage of protein-sequence space with high-accuracy models," *Nucleic Acids Research*, vol. 50, pp. D439–D444, 11 2021.

[5] M. Tsubaki, K. Tomii, and J. Sese, "Compound–protein interaction prediction with end-to-end learning of neural networks for graphs and sequences," *Bioinformatics*, vol. 35, pp. 309–318, 07 2018.

[6] J. M. Stokes, K. Yang, K. Swanson, W. Jin, A. Cubillos-Ruiz, N. M. Donghia, C. R. MacNair, S. French, L. A. Carfrae, Z. Bloom-Ackermann, V. M. Tran, A. Chiappino-Pepe, A. H. Badran, I. W. Andrews, E. J. Chory, G. M. Church, E. D. Brown, T. S. Jaakkola, R. Barzilay, and J. J. Collins, "A deep learning approach to antibiotic discovery," *Cell*, vol. 180, no. 4, pp. 688–702.e13, 2020.

[7] K. Atz, F. Grisoni, and G. Schneider, "Geometric deep learning on molecular representations," 2021.

[8] J. Wang, A. Ma, Q. Ma, D. Xu, and T. Joshi, "Inductive inference of gene regulatory network using supervised and semi-supervised graph neural networks," *Computational and structural biotechnology journal*, vol. 18, pp. 3335–3343, 2020.

[9] S. Cavallari, V. W. Zheng, H. Cai, K. C.-C. Chang, and E. Cambria, "Learning community embedding with community detection and node embedding on graphs," p. 377–386, 2017.

[10] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," p. 137–146, 2003.

[11] T. Chen, S. Yan, J. Guo, and W. Wu, "Touplegdd: A fine-designed solution of influence maximization by deep reinforcement learning," *IEEE Transactions on Computational Social Systems*, pp. 1–12, 2023.

[12] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. Battaglia, "Learning to simulate complex physics with graph networks," vol. 119, pp. 8459–8468, 13–18 Jul 2020.

[13] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. F. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, Ç. Gülçehre, H. F. Song, A. J. Ballard, J. Gilmer, G. E. Dahl, A. Vaswani, K. R. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu, "Relational inductive biases, deep learning, and graph networks," *CoRR*, vol. abs/1806.01261, 2018.

[14] S. Gong, Q. Meng, J. Zhang, H. Qu, C. Li, S. Qian, W. Du, Z.-M. Ma, and T.-Y. Liu, "An efficient lorentz equivariant graph neural network for jet tagging," *Journal of High Energy Physics*, vol. 2022, July 2022.

[15] J. Shlomi, S. Ganguly, E. Gross, K. Cranmer, Y. Lipman, H. Serviansky, H. Maron, and N. Segol, "Secondary vertex finding in jets with neural networks," *The European Physical Journal C*, vol. 81, no. 6, pp. 1–12, 2021.

[16] W. Hu, M. Shuaibi, A. Das, S. Goyal, A. Sriram, J. Leskovec, D. Parikh, and C. L. Zitnick, "Forcenet: A graph neural network for large-scale quantum calculations," *CoRR*, vol. abs/2103.01436, 2021.

[17] S. Ye, J. Liang, R. Liu, and X. Zhu, "Symmetrical graph neural network for quantum chemistry with dual real and momenta space," *The Journal of Physical Chemistry A*, vol. 124, no. 34, pp. 6945–6953, 2020. PMID: 32786228.

[18] A. Derrow-Pinion, J. She, D. Wong, O. Lange, T. Hester, L. Perez, M. Nunkesser, S. Lee, X. Guo, B. Wiltshire, P. W. Battaglia, V. Gupta, A. Li, Z. Xu, A. Sanchez-Gonzalez, Y. Li, and P. Velickovic, "ETA prediction with graph neural networks in google maps," *CoRR*, vol. abs/2108.11482, 2021.

[19] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," 2018.

[20] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, and H. Li, "T-GCN: A temporal graph convolutional network for traffic prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 9, pp. 3848–3858, 2020.

[21] K. Lei, P. Guo, Y. Wang, X. Wu, and W. Zhao, "Solve routing problems with a residual edge-graph attention neural network," *Neurocomputing*, vol. 508, pp. 79–98, 2022.

[22] X. Hu, C. Zhao, and G. Wang, "A traffic light dynamic control algorithm with deep reinforcement learning based on gnn prediction," 2020.

[23] H. Liu, J. Han, Y. Fu, J. Zhou, X. Lu, and H. Xiong, "Multi-modal transportation recommendation with unified route representation learning," *Proc. VLDB Endow.*, vol. 14, p. 342–350, nov 2020.

[24] Z. Wang, M. Eisen, and A. Ribeiro, "Learning decentralized wireless resource allocations with graph neural networks," *IEEE Transactions on Signal Processing*, vol. 70, pp. 1850–1863, 2022.

[25] O. Hope and E. Yoneki, "Gddr: Gnn-based data-driven routing," pp. 517–527, 2021.

[26] A. Mirhoseini, A. Goldie, M. Yazgan, J. W. Jiang, E. M. Songhori, S. Wang, Y. Lee, E. Johnson, O. Pathak, A. Nazi, J. Pak, A. Tong, K. Srinivasa, W. Hang, E. Tuncer, Q. V. Le, J. Laudon, R. Ho, R. Carpenter, and J. Dean, "A graph placement methodology for fast chip design," *Nat.*, vol. 594, no. 7862, pp. 207–212, 2021.

[27] N. Wu, H. Yang, Y. Xie, P. Li, and C. Hao, "High-level synthesis performance prediction using gnns: Benchmarking, modeling, and advancing," *CoRR*, vol. abs/2201.06848, 2022.

[28] K. Han, Y. Wang, J. Guo, Y. Tang, and E. Wu, "Vision GNN: an image is worth graph of nodes," 2022.

[29] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," 2019.

[30] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," pp. 593–607, 2018.

[31] S. Liu, B. Grau, I. Horrocks, and E. Kostylev, "Indigo: Gnn-based inductive knowledge graph completion using pair-wise encoding," *Advances in Neural Information Processing Systems*, vol. 34, pp. 2034–2045, 2021.

[32] H. Xu, J. Bao, and W. Liu, "Double-branch multi-attention based graph neural network for knowledge graph completion," pp. 15257–15271, July 2023.

[33] A. Anil, V. Gutiérrez-Basulto, Y. Ibañéz-García, and S. Schockaert, "Inductive knowledge graph completion with gnns and rules: An analysis," 2024.

[34] N. Barbieri, G. Manco, and E. Ritacco, "Survival factorization on diffusion networks," pp. 684–700, 2017.

[35] M. Chen, W. Zhang, W. Zhang, Q. Chen, and H. Chen, "Meta relational learning for few-shot link prediction in knowledge graphs," *CoRR*, vol. abs/1909.01515, 2019.

[36] S. Masoudnia and R. Ebrahimpour, "Mixture of experts: a literature survey," *Artificial Intelligence Review*, vol. 42, no. 2, pp. 275–293, 2014.

[37] G. Dantzig, R. Fulkerson, and S. Johnson, "Solution of a large-scale traveling-salesman problem," *Journal of the operations research society of America*, vol. 2, no. 4, pp. 393–410, 1954.

[38] G. B. Dantzig, *Linear Programming and Extensions*. Princeton: Princeton University Press, 1963.

[39] A. H. Land and A. G. Doig, "An automatic method of solving discrete programming problems," *Econometrica*, vol. 28, no. 3, pp. 497–520, 1960.

[40] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[41] J. H. Holland, "Genetic algorithms," *Scientific American*, vol. 267, no. 1, pp. 66–73, 1992.

[42] F. Glover and M. Laguna, "Tabu search," pp. 2093–2229, 1998.

[43] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, *et al.*, "Graph attention networks," *stat*, vol. 1050, no. 20, pp. 10–48550, 2017.

[44] W. Kool, H. Van Hoof, and M. Welling, "Attention, learn to solve routing problems!," *arXiv preprint arXiv:1803.08475*, 2018.

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

[46] Q. Cappart, D. Chételat, E. B. Khalil, A. Lodi, C. Morris, and P. Veličković, "Combinatorial optimization and reasoning with graph neural networks," *Journal of Machine Learning Research*, vol. 24, no. 130, pp. 1–61, 2023.

[47] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, "Cost-effective outbreak detection in networks," p. 420–429, 2007.

[48] Y. Tang, Y. Shi, and X. Xiao, "Influence maximization in near-linear time: A martingale approach," p. 1539–1554, 2015.

[49] C. Borgs, M. Brautbar, J. T. Chayes, and B. Lucier, "Influence maximization in social networks: Towards an optimal algorithmic solution," *CoRR*, vol. abs/1212.0884, 2012.

[50] N. Ohsaka, "The solution distribution of influence maximization: A high-level experimental study on three algorithmic approaches," pp. 2151–2166, 2020.

[51] A. Arora, S. Galhotra, and S. Ranu, "Debunking the myths of influence maximization: An in-depth benchmarking study," pp. 651–666, 2017.

[52] N. Du, L. Song, H. Woo, and H. Zha, "Uncover topic-sensitive information diffusion networks," pp. 229–237, 2013.

[53] N. Barbieri, F. Bonchi, and G. Manco, "Topic-aware social influence propagation models," *Knowl. Inf. Syst.*, vol. 37, no. 3, pp. 555–584, 2013.

[54] Ç. Aslay, N. Barbieri, F. Bonchi, and R. Baeza-Yates, "Online topic-aware influence maximization queries," pp. 295–306, 2014.

[55] W. Chen, T. Lin, and C. Yang, "Real-time topic-aware influence maximization using preprocessing," vol. 9197, pp. 1–13, 2015.

[56] S. Chen, J. Fan, G. Li, J. Feng, K. Tan, and J. Tang, "Online topic-aware influence maximization," *Proc. VLDB Endow.*, vol. 8, no. 6, pp. 666–677, 2015.

[57] Y. Li, J. Fan, Y. Wang, and K. Tan, "Influence maximization on social graphs: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 10, pp. 1852–1872, 2018.

[58] Y. Li, H. Gao, Y. Gao, J. Guo, and W. Wu, "A survey on influence maximization: From an ml-based combinatorial optimization," *ACM Trans. Knowl. Discov. Data*, vol. 17, jul 2023.

[59] H. Li, M. Xu, S. S. Bhowmick, J. S. Rayhan, C. Sun, and J. Cui, "Piano: Influence maximization meets deep reinforcement learning," *IEEE Transactions on Computational Social Systems*, pp. 1–13, 2022.

[60] S. Tian, P. Zhang, S. Mo, L. Wang, and Z. Peng, "A learning approach for topic-aware influence maximization," pp. 125–140, 07 2019.

[61] S. Manchanda, A. MITTAL, A. Dhawan, S. Medya, S. Ranu, and A. Singh, "Gcomb: Learning budget-constrained combinatorial algorithms over billion-sized graphs," vol. 33, pp. 20000–20011, 2020.

[62] C. Ling, J. Jiang, J. Wang, M. T. Thai, R. Xue, J. Song, M. Qiu, and L. Zhao, "Deep graph representation learning and optimization for influence maximization," vol. 202, pp. 21350–21361, 2023.

[63] H. Chen, W. Qiu, H. Ou, B. An, and M. Tambe, "Contingency-aware influence maximization: A reinforcement learning approach," 2021.

[64] G. Panagopoulos, F. D. Malliaros, and M. Vazirgiannis, "Multi-task learning for influence estimation and maximization," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 9, pp. 4398–4409, 2022.

[65] W. Xia, Y. Li, J. Wu, and S. Li, "Deepis: Susceptibility estimation on social networks," pp. 761–769, 2021.

[66] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[67] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," 2014.

[68] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," *CoRR*, vol. abs/1704.01212, 2017.

[69] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *CoRR*, vol. abs/1609.02907, 2016.

[70] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *CoRR*, vol. abs/1706.02216, 2017.

[71] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," 2019.

[72] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe, "Weisfeiler and leman go neural: higher-order graph neural networks," 2019.

[73] V. P. Dwivedi, A. T. Luu, T. Laurent, Y. Bengio, and X. Bresson, "Graph neural networks with learnable structural and positional representations," *CoRR*, vol. abs/2110.07875, 2021.

[74] B. Finkelshtein, X. Huang, M. Bronstein, and İsmail İlkan Ceylan, "Cooperative graph neural networks," 2023.

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

[76] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[77] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," *CoRR*, vol. abs/1509.06461, 2015.

[78] Z. Wang, N. de Freitas, and M. Lanctot, "Dueling network architectures for deep reinforcement learning," *CoRR*, vol. abs/1511.06581, 2015.

[79] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2016.

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

[81] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.

[82] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," vol. 26, 2013.

[83] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," *CoRR*, vol. abs/1606.06357, 2016.

[84] W. Xiong, M. Yu, S. Chang, X. Guo, and W. Y. Wang, "One-shot relational learning for knowledge graphs," pp. 1980–1990, Oct.-Nov. 2018.

[85] Q. Huang, H. Ren, and J. Leskovec, "Few-shot relational reasoning via connection subgraph pretraining," 2022.

[86] M. M. Bronstein, J. Bruna, T. Cohen, and P. Velickovic, "Geometric deep learning: Grids, groups, graphs, geodesics, and gauges," *CoRR*, vol. abs/2104.13478, 2021.

[87] A. Bundy and L. Wallen, *Breadth-First Search*, pp. 13–13. Berlin, Heidelberg: Springer Berlin Heidelberg, 1984.

[88] L. László, "Random walks on graphs: A survey, combinatorics, paul erdos is eighty," *Bolyai Soc. Math. Stud.*, vol. 2, 01 1993.

[89] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," *CoRR*, vol. abs/1403.6652, 2014.

[90] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," *CoRR*, vol. abs/1607.00653, 2016.

[91] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[92] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," p. 891–900, 2015.

[93] F. R. K. Chung, *Spectral Graph Theory*. American Mathematical Society, 1997.

[94] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, second ed., 2018.

[95] R. Bellman, *Dynamic Programming*. Princeton University Press, 1957.

[96] A. Goyal, F. Bonchi, and L. V. Lakshmanan, "Learning influence probabilities in social networks," p. 241–250, 2010.

[97] G. Panagopoulos, F. Malliaros, and M. Vazirgiannis, "DiffuGreedy: An Influence Maximization Algorithm based on Diffusion Cascades," Dec. 2018.

[98] Y. He, M. Permultter, G. Reinert, and M. Cucuringu, "Msgnn: A spectral graph neural network based on a novel magnetic signed laplacian," 2022.

[99] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," *CoRR*, vol. abs/1703.03400, 2017.

[100] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[101] H. Dai, B. Dai, and L. Song, "Discriminative embeddings of latent variable models for structured data," *CoRR*, vol. abs/1603.05629, 2016.

[102] S. Tian, S. Mo, L. Wang, and Z. Peng, "Deep reinforcement learning-based approach to tackle topic-aware influence maximization," *Data Science and Engineering*, vol. 5, no. 1, pp. 1–11, 2020.

[103] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," 2018.

[104] S. Geisler, Y. Li, D. Mankowitz, A. T. Cemgil, S. Günnemann, and C. Paduraru, "Transformers meet directed graphs," 2023.

[105] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[106] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018.

[107] J. Godwin*, T. Keck*, P. Battaglia, V. Bapst, T. Kipf, Y. Li, K. Stachenfeld, P. Veličković, and A. Sanchez-Gonzalez, "Jraph: A library for graph neural networks in jax.," 2020.

[108] J. Zhang, B. Liu, J. Tang, T. Chen, and J. Li, "Social influence locality for modeling retweeting behaviors," p. 2761–2767, 2013.

[109] D. Vrandečić and M. Krötzsch, "Wikidata: a free collaborative knowledgebase," *Commun. ACM*, vol. 57, p. 78–85, sep 2014.

[110] D. Jambor, K. Teru, J. Pineau, and W. L. Hamilton, "Exploring the limits of few-shot link prediction in knowledge graphs," pp. 2816–2822, Apr. 2021.