Quest: Journal of Geometry, Mathematical and Quantum Physics
e-ISSN: 3064-7282 Vol.1 Issue 3|2024

Study on Application of Graph Theory in Artificial Intelligence (AI)

Ashok Kumar Mahato, Rahul Das, Suresh Kumar Sahani
Department of science and technology, Rajarshi Janak University, Janakpur, Nepal

Abstract: Mathematics play significant role in different fields/area. One of the major area of mathematics is Graph theory, In computer science, graph theory is an essential tool, particularly for artificial intelligence (AI). This study investigates the use of graph theory in several artificial intelligence (AI) fields, such as graph coloring algorithms, natural language processing (NLP), recommendation systems, and graph neural networks (GNNs). We'll show how comprehensive connection modeling and scalability in graph-based recommendation systems improve content customization, and how GNNs are excellent at managing complicated relational data in domains like traffic management, social network analysis, and drug development. We also investigate the use of graph theory to natural language processing (NLP), specifically as it relates to knowledge graphs that enhance chatbot performance, search engine accuracy, and semantic search. Furthermore, we examine how graph coloring techniques are applied in real-world situations like scheduling and register allocation, emphasizing how well they work to optimize resource usage and solve problems. This thorough analysis highlights graph theory's revolutionary influence on developing AI technologies and resolving practical issues, demonstrating its important function in both theoretical and practical contexts.

Keywords: Graphs, Graph theory, Artificial intelligence, Graph neural network, Recommendation system, NPL, Graph coloring.

Introduction

Graph Theory: The graph theory's beginnings can be traced back to the 1735 Konigsberg Bridge problem. This issue led to the idea of the Eulerian graph. Euler researched the Konigsberg bridge problem and created a structure known as the "Eulerian graph" to address it. Around 1840, A.F. Mobius gave rise to the concepts of a complete and a bipartite graph, and Kwiatkowski demonstrated that they were planar through the lens of recreational issues. The idea of a tree, which is a connected graph without Karl Kirchhoff introduced bicycles in 1845, and he used theoretical concepts to design them. while computing currents in electrical networks or circuits. Thomas Guthrie discovered the infamous "color problem" in 1852. Next, in 1856, Thomas. By observing bicycles on a polyhedral, P. Kirkman and William Hamilton established the notion known as the Hamiltonian graph. It tears that visited specific places precisely once. A puzzling problem was mentioned by H. Dudeney in 1913. Despite being a contrived problem, the color problem wasn't solved until a century later by Kenneth Heinrich Haken and Appel. This moment is recognized as the creation of graph theory.

Graph: A diagram with points connected by lines is a useful tool for easily describing many real-world situations. As an illustration, the points might stand for people, while the lines could depict friendships or communication centers connected by communication links. These diagrams

primarily focus on whether two points are related, rather than how they are connected. This notion gives rise to the mathematical notion of a graph.

A graph is a picture or diagram made up of a number of vertices and edges connecting specific pairings of these vertices.

A graph G=[V(G), E(G)] can be written mathematically. The two finite sets V(G) and E(G) are defined as follows:

V(G)= Vertex set of graph G

E(G) = Edge set of graph G such that e of E(G) allocated on an unordered pair of vertices (u, v) known as e's end vertices.

Graphs get their name from their ability to be graphically represented, and it is this ability to do so that allows us to better comprehend many of their characteristics. A line denotes an edge, and a point represents a vertex. combining the points that stand in for its ends.

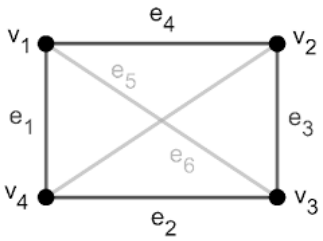

Fig*: graph with 4 vertex and 6 edges*

Artificial Intelligence: The branch of computer science known as artificial intelligence (AI) seeks to build machines that are able to carry out tasks that ordinarily require human intelligence. Artificial intelligence (AI) systems can be made to think and act like humans, or they can be programmed to follow logical reasoning and make performance-maximizing judgments. Autonomous vehicles are one application of AI. These cars employ artificial intelligence (AI) to sense their surroundings using cameras and sensors, process data to comprehend traffic and road conditions, and make decisions in real time—like changing lanes or stopping at a red light—so they can drive safely without any assistance from a human. Self-driving automobile AI blends rational intelligence (selecting the best routes based on data) with human-like behavior (obeying traffic laws).

A major component of artificial intelligence (AI) is graph theory. It demonstrates how ideas from graph theory are frequently applied to computer science applications, such as artificial intelligence (AI), in order to model and solve complicated issues. AI systems can process and reason about information more easily when relationships and dependencies between data elements are organized using graph theory. For example, shortest path and graph coloring algorithms are used in AI activities including learning, optimization, and decision-making. Additionally, graph databases (like Neo4j) facilitate the querying and storing of graph-structured data, which helps artificial intelligence (AI) comprehend complex relationships and perform better.

Applications

1. Recommendation system:

When a user consumes or watches content, a recommendation system provides tailored suggestions. This system makes recommendations depending on how the content is used, user information, and/or content ratings. We suggest a new approach to address the aforementioned problems: a graph-based recommendation system. It includes graph algorithms, multiple recommendation systems, and assessed user content data stored inside a graph structure. The scalability and diversity of relationship modeling of the graph-based recommendation system make it superior to the current one. This distinction in data storage efficiency is what sets a graph-based recommendation apart from the current recommendation methodology. To forecast the

similarity of collaborative filtering procedures, the graph-based recommendation system does not create a sparse matrix.

Recommendation Process:

Data Gathering using Internet Scraping: Reviews, ratings, tags, movies, users, and other data are gathered from the internet. Utilizing this data, the recommendation service is constructed.

Creating and Importing Knowledge Graph Models: The various movie-related data (e.g., actors, tags, new movies, reviews, etc.) are connected by a knowledge graph. The graph is constantly being updated by web scraping, which adds new movies and pertinent metadata.

Finding Candidates for Recommendations: A graph is defined by a graph recommendation engine using the relationships between metadata (e.g., user-movie interactions).

Similarity-Based suggestions: A graph depicting user-movie interactions is used to make suggestions based on how similar users' preferences are. To improve suggestion accuracy, people with similar tastes are grouped together using community discovery.

Methods of Graph Analysis: Users can receive suggestions based on the significance of directors, actors, and other movie related properties thanks to Centrality Analysis (PageRank). Users can search for movies based on particular themes or genres by using tags in the graph. Personalized Suggestions for Films:

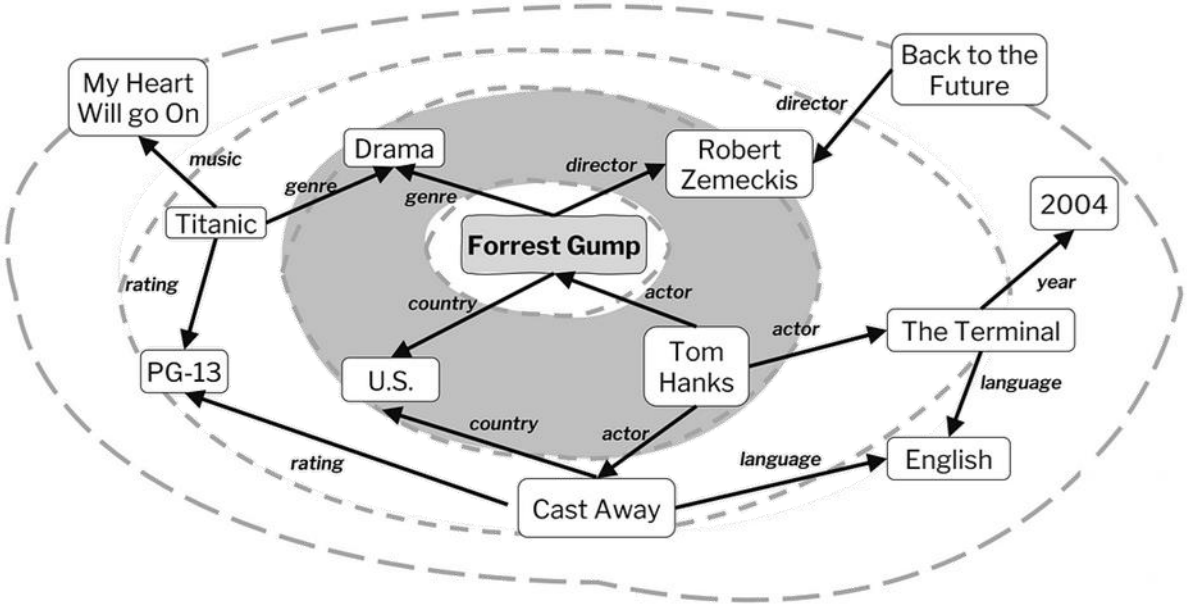

Fig: Movie recommendation system

A recommendation algorithm based on graphs is used by major video streaming services like Netflix. Additionally, it's utilized on e-commerce websites like eBay and Auction, among others. The market has already seen widespread use of distributed storage techniques utilizing graph technology connected to Elasticsearch or Hadoop's Hbase. Advances in recommendation algorithms have been studied and numerous studies on graph distribution storage have been released recently. As a result, it is evident that graph technology has more to offer the world and that the demand for diverse data analysis has grown quickly.

2. Graph Neural Network:

Items in the real world are frequently described in terms of their relationships with other items. A graph is a natural way to represent a collection of things and the relationships among them. For more than ten years, researchers have created neural networks known as graph neural networks, or GNNs, that function on graph data. Their expressive strength and skills have expanded with recent

improvements. Practical applications are beginning to appear in fields including traffic prediction, physics simulations, antibiotic discovery, and fake news detection.

Since many data sources are in a graph-based format, relationships and dependencies between items can be captured by representing them as nodes (vertices) connected by edges. Traditional fully connected networks struggle to analyze these kinds of datasets because of their intricate structure. relationship graphs are a better way to depict the complex relationship structure seen in these datasets. As a result, modeling these kinds of datasets with graph neural networks is a potential option.

Numerous use cases using graph-structured data can be addressed by GNNs. Among the most well-known instances is social media analysis. Because of the size of their user bases, users of Facebook, Instagram, and other platforms can be considered as individual nodes in a graph with their profile details acting as attributes. With the help of their friend lists, we can build a sizable graph. Here is a famous case where fully connected neural networks are outperformed by graph neural networks.

Because of its capacity to handle relational data, Graph Neural Networks (GNNs) are finding increased use across other domains. GNNs models are used in molecular graphs for drug discovery and biology to predict the properties of compounds and find possible candidates for drugs. By utilizing user interactions, they improve community detection and friend suggestions in social network analysis. By simulating linguistic links, GNNs enhance semantic parsing and document categorization for natural language processing. Moreover, GNNs in traffic management examine road networks for route optimization and traffic prediction, improving the effectiveness of transportation. Following figure shows that the traffic management by using graph neural network.

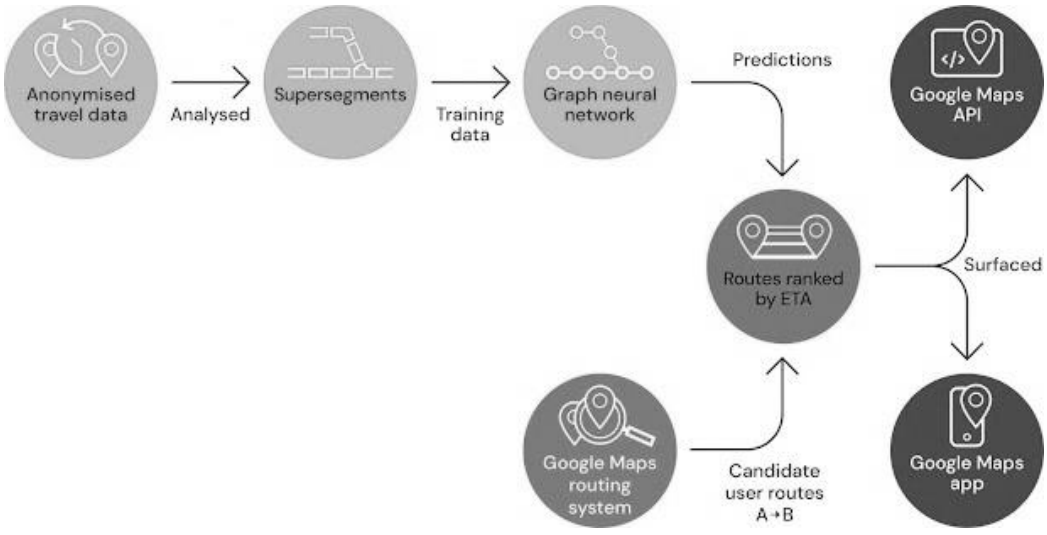

Fig: Traffic management analysis

It's also evident that Graph Neural Networks (GNNs) are used in a variety of ways by AWS (Amazon Web Services) to satisfy client demands. Amazon Neptune is one instance. It is a fully managed graph database service that simplifies the development and operation of highly connected data processing systems. GNNs are used by Neptune to index and query chart data, giving users fast access to relationships between items in their data for analysis. Amazon SageMaker is a further example. Customers can train and implement machine learning models at scale with our fully managed machine learning service. Customers can train models for tasks like link prediction, node classification, and chart classification using SageMaker's integrated GNN algorithm.

3. Natural Language Processing:

As the name implies, "natural language" refers to language written or spoken by people as opposed to languages utilized for computer programming or communication. The subfield of artificial intelligence (AI), which includes natural language processing (NLP), computer science is the study of symbolic inference by use of computers and the representation of symbolic knowledge for inference-making purposes. Understanding natural language by computers is one of the most challenging issues in artificial intelligence. because of the diversity, complexity, and irregularity of human language, as well as the philosophical meaning issues.

Graphs are essential for improving Natural Language Processing (NLP) capabilities because they offer a connected and organized data foundation. In a variety of NLP applications, this paradigm makes a substantial contribution to both the production and comprehension of human language.

Application of NPL using graphs

Search engines: In order to provide more precise and contextually relevant search results, modern search engines make use of knowledge graphs to comprehend and interpret user queries. Search engines can return results that most closely fit the user's requirements by comprehending the intent and context of user searches.

Chatbots and Virtual Assistants: Chatbots and virtual assistants interpret user queries and deliver precise, context-sensitive responses by utilizing knowledge graphs. For example, the chatbot uses its knowledge graph to propose restaurants based on user history, location, and preferred cuisine when a user asks for recommendations.

Semantic Search in Enterprise Systems: By comprehending the context and connections between various data points, knowledge graphs play a crucial role in semantic search applications within enterprise systems, assisting in the discovery of information across enormous corporate databases.

Language Translation Services: Knowledge graphs help translators translate words more accurately and naturally by helping them grasp the subtle cultural and contextual differences between other languages.

In general, knowledge graphs give NLP applications a depth of comprehension and sophistication that improves the intuitiveness, accuracy, and human-like quality of interactions with AI systems.

4. Graph Coloring Algorithm

Beginning with vertex 0, give each vertices a different color. Verify whether the adjacent vertices have the same color before assigning them one. Indicate which color assignment is a component of the solution if it does not conflict with the requirements. Finding a vertex-coloring of a graph can be done simply by conducting a systematic search among all mappings from the set of vertices to the set of colors.

Sometimes the sequence in which vertices are processed determines how many colors are used. Think about the following two graphs, for instance. Vertices 3 and 4 are switched on the graph on the right side, as you can see. The graph on the left can be colored with three colors if we take into account the vertices 0, 1, 2, 3, and 4. However, we require 4 colors if we take into account the vertices 0, 1, 2, 3, and 4 in the right graph.

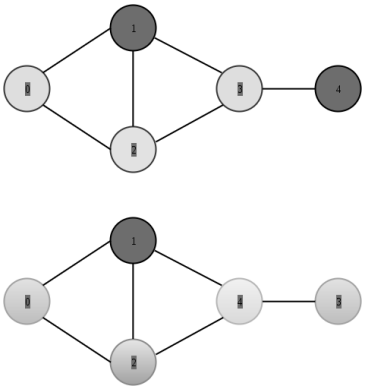

Fig: left graph with 3 color and Right graph with 4 color

Application of graph coloring

Scheduling algorithms: let's say you have a number of workers and a set of jobs to complete. You need to allocate each worker a task within a specific time slot (let's suppose each job takes one time slot for simplicity). Although jobs can be planned in any sequence, there is a possibility that two jobs in tandem won't be scheduled for the same time slot if they depend on the same resource, for example. Every work in the matching graph has a vertex, and every pair of jobs that conflicts has an edge. With an infinite workforce, the graph's chromatic number represents the ideal time to complete every task without encountering any conflicts.

Register allocation: a computer software called a compiler converts high-level languages like C, Java, or OCaml's source code into machine code. This is often accomplished in a few phases, one of which is assigning registers to the program's most frequently used variables while storing the other ones in memory. This may be modeled as a graph coloring issue, in which the compiler creates an interference graph with symbolic registers for vertices and edges connecting nodes as needed simultaneously. The variables can be kept in k registers if the graph can be colored with k different hues.

There are uses for pattern matching in graph coloring as well.

The process of giving each vertex in a graph a different color so that no two neighboring vertices have the same color is known as graph coloring. The variation in color needs between two comparable graphs illustrates how the order in which vertices are processed affects the algorithm's performance. Graph coloring is used in many different contexts, such as register allocation in compilers (variables are assigned to restricted registers) and scheduling algorithms (tasks must be allotted time slots without conflicts). Pattern matching and other optimization issues also employ graph coloring.

Problem:

You have eight jobs, each represented as a vertex in a graph, in an AI-driven task scheduling system. Certain pairs of tasks cannot be planned at the same time because some tasks depend on the accomplishment of others. The edges connecting the vertices provide the dependence restrictions. The edges are as follows:

Task 1 depends on Task 2.

Task 2 is dependent upon Tasks 3 and 4.

Tasks 5 and 6 are necessary for Task 4.

Tasks 7 and 8 are necessary for Task 6.

The objective is to figure out the bare minimum of time slots needed to plan every activity so that no two dependent tasks—that is, no two neighboring vertices with the same color—are scheduled at the same time.

Solution:

The jobs and their dependencies can be shown as a graph with eight vertices and the following edges:

(1, 2), (2, 3), (2, 4), (4, 5), (4, 6), (6, 7), (6, 8)

There are no cycles in this graph, which resembles a tree.

Finding the fewest colors required to color the graph while making sure that no two nearby vertices—which stand for dependent tasks—share the same color is the challenge.

Since no two neighboring vertices would need the same color, a tree is a bipartite graph and may be colored with a maximum of two colors. This is due to the fact that vertices in a bipartite network may always be divided into two groups so that there are no connections connecting vertices in the same group.

This graph has a chromatic number of 2. As a result, two time slots are the bare minimum needed to plan any job without creating a conflict.

This issue simulates an artificial intelligence task scheduling situation in which jobs need to be scheduled into non-conflicting time slots according to dependencies. In a parallel processing environment, we may maximize system efficiency and guarantee appropriate resource use by reducing the number of time slots (colors).

Conclusion

Graph theory continues to be a potent and indispensable instrument in the field of artificial intelligence, greatly augmenting the capacities and efficacy of many AI applications. This study has illustrated the ways in which graph-based approaches, such as recommendation systems, graph neural networks, and natural language processing, aid in the resolution of intricate, practical issues. Graph theory is used in recommendation systems to enhance customization through complex relationship modeling, and graph neural networks provide cutting edge solutions for traffic control, medication discovery, and social network analysis. Graph theory use in NLP also improves contextual comprehension and search accuracy. Graph coloring methods are explored in more detail to show how useful they are for scheduling and register allocation. All things considered, the use of graph theory to AI enhances theoretical understanding and offers workable answers to important problems. The ideas and methods of graph theory will continue to be crucial for fostering creativity and accomplishing technological advances as AI advances.

References

1. van Steen, M. (2010). *Graph theory and complex networks: An introduction*. CRC Press.
2. Deo, N. (2012). *Graph theory with applications to engineering and computer science*. Dover Publications.
3. Wu, Z., Lima, J., & Ba, J. (2021). Graph neural networks: A review of methods and applications. *IEEE Transactions on Neural Networks and Learning Systems*.
4. West, D. B. (2001). *Introduction to graph theory* (2nd ed.). Prentice Hall.
5. Russell, S., & Norvig, P. (2020). *Artificial intelligence: A modern approach* (4th ed.). Pearson.
6. Wu, Z., Souvenir, R., & Yan, X. (2019). Graph neural networks: A review of methods and applications. *IEEE Transactions on Neural Networks and Learning Systems, 31*(8), 2570-2583.
7. Yao, L., & Huang, C. (2017). Graph-based recommendation systems: A survey. *ACM Computing Surveys (CSUR), 50*(6), 1-35.

8. Li, S., Zhang, Y., & Li, X. (2019). Traffic prediction with graph convolutional networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(01), 7371-7378.
9. Zhang, Y., & Wang, J. (2018). Graph-based neural networks for natural language processing: A survey. *Journal of Computer Science and Technology*, *33*(5), 913-936.
10. Wang, X., Zhang, X., & Li, X. (2020). A survey of recommendation algorithms based on graph theory. *ACM Computing Surveys (CSUR)*, *53*(1), 1-35.